Movie Recommendation System: To understand the functioning of how a recommendation system works. Develop an Item BasedCollaborative Filter using Netflix dataset

A recommendation system provides suggestions to the users through a filtering process that is based on user preferences and browsing history. The information about the user is taken as an input. The information is taken from the input that is in the form of browsing data. There are two types of recommendation systems – Content-Based Recommendation System and Collaborative Filtering Recommendation. In this project of recommendation system in R, we will work on a collaborative filtering recommendation system and more specifically, ITEM based collaborative recommendation system.

CODE (R-STUDIO) : -

```
install.packages("recommenderlab")
library(recommenderlab)
library(ggplot2)
install.packages("data.table", dependencies=TRUE)
library(data.table)
library(reshape2)
movie_data <- read.csv("C:/Users/HP/Downloads/IMDB-Dataset/movies.csv",stringsAsFactors=FALSE)
rating_data <- read.csv("C:/Users/HP/Downloads/IMDB-Dataset/ratings.csv")
str(movie_data)
head(movie_data)
summary(rating_data)
head(rating_data)
movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors=FALSE)
library(data.table)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
                                        type.convert=TRUE),
                              stringsAsFactors=FALSE) #DataFlair
colnames(movie_genre2) <- c(1:10)
list_genre <- c("Action", "Adventure", "Animation", "Children",
                "Comedy", "Crime","Documentary", "Drama", "Fantasy",
                "Film-Noir", "Horror", "Musical", "Mystery","Romance",
                "Sci-Fi", "Thriller", "War", "Western")
genre_mat1 <- matrix(0,10330,18)
genre_mat1[1,] <- list_genre
colnames(genre_mat1) <- list_genre
for (index in 1:nrow(movie_genre2)) {
  for (col in 1:ncol(movie_genre2)) {
    gen_col = which(genre_mat1[1,] == movie_genre2[index,col]) #Author DataFlair
    genre_mat1[index+1,gen_col] <- 1
  }
}
```

```
genre_mat2 <- as.data.frame(genre_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list
for (col in 1:ncol(genre_mat2)) {
  genre_mat2[,col] <- as.integer(genre_mat2[,col]) #convert from characters to integers
}
str(genre_mat2)
SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[])
head(SearchMatrix)
ratingMatrix <- dcast(rating_data, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userIds
#Convert rating matrix into a recommenderlab sparse matrix
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")
ratingMatrix
recommendation_model <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
names(recommendation_model)
lapply(recommendation_model, "[[", "description")
recommendation_model$IBCF_realRatingMatrix$parameters
similarity_mat <- similarity(ratingMatrix[1:4, ],
                             method = "cosine",
                             which = "users")
as.matrix(similarity_mat)
image(as.matrix(similarity_mat), main = "User's Similarities")
movie_similarity <- similarity(ratingMatrix[, 1:4], method =
                               "cosine", which = "items")
as.matrix(movie_similarity)
image(as.matrix(movie_similarity), main = "Movies similarity")
rating_values <- as.vector(ratingMatrix@data)
unique(rating_values) # extracting unique ratings
Table_of_Ratings <- table(rating_values) # creating a count of movie ratings
Table_of_Ratings
library(ggplot2)
movie_views <- colCounts(ratingMatrix) # count views for each movie
table_views <- data.frame(movie = names(movie_views),
                          views = movie_views) # create dataframe of views
table_views <- table_views[order(table_views$views,
                                 decreasing = TRUE), ] # sort by number of views
table_views$title <- NA
for (index in 1:10325){
  table_views[index,3] <- as.character(subset(movie_data,
                                              movie_data$movieId == table_views[index,1])$title)
}
table_views[1:6,]
ggplot(table_views[1:6, ], aes(x = title, y = views)) +
  geom_bar(stat="identity", fill = 'steelblue') +
  geom_text(aes(label=views), vjust=-0.3, size=3.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Total Views of the Top Films")
image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")
movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
                              colCounts(ratingMatrix) > 50]
```

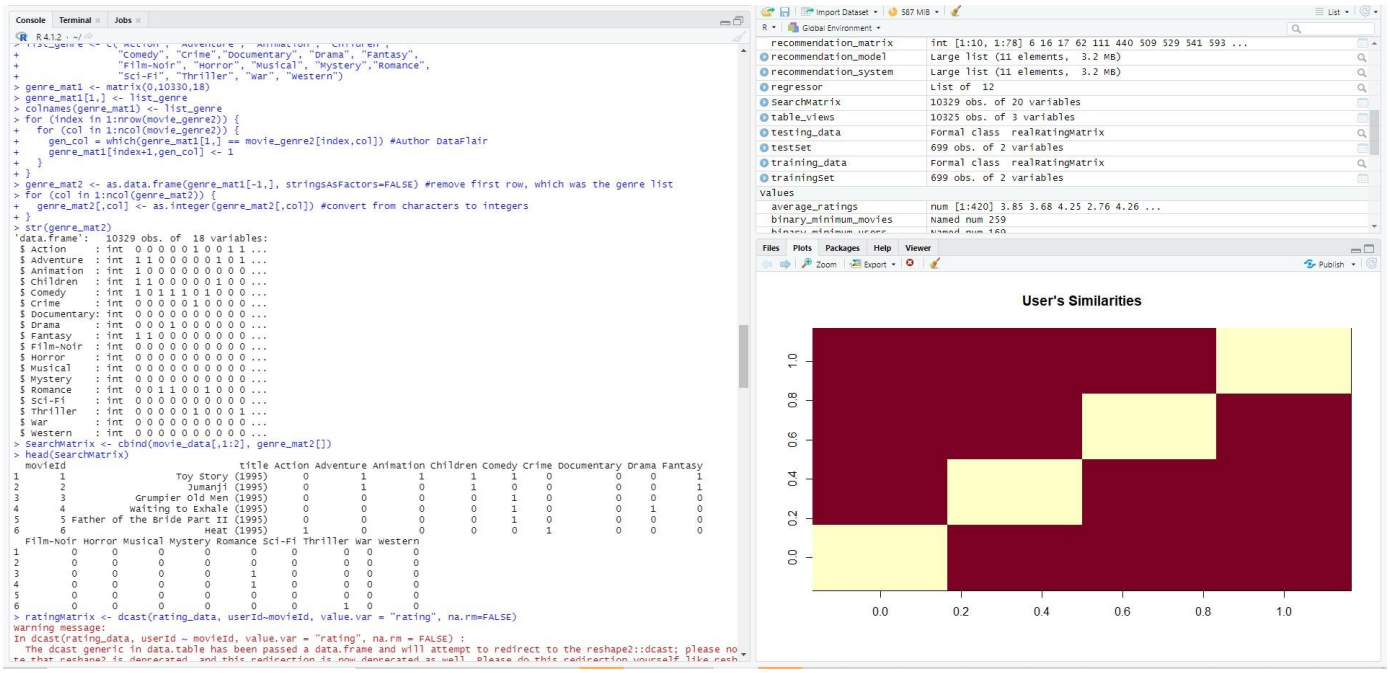
```
movie_ratings
minimum_movies<- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
     main = "Heatmap of the top users and movies")
average_ratings <- rowMeans(movie_ratings)
qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
ggtitle("Distribution of the average rating per user")
normalized_ratings <- normalize(movie_ratings)
sum(rowMeans(normalized_ratings) > 0.00001)
image(normalized_ratings[rowCounts(normalized_ratings) > minimum_movies,
                         colCounts(normalized_ratings) > minimum_users],
     main = "Normalized Ratings of the Top Users")
binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
binary_minimum_users <- quantile(colCounts(movie_ratings), 0.95)
#movies_watched <- binarize(movie_ratings, minRating = 1)
good_rated_films <- binarize(movie_ratings, minRating = 3)
image(good_rated_films[rowCounts(movie_ratings) > binary_minimum_movies,
                       colCounts(movie_ratings) > binary_minimum_users],
     main = "Heatmap of the top users and movies")
sampled_data<- sample(x = c(TRUE, FALSE),
                      size = nrow(movie_ratings),
                      replace = TRUE,
                      prob = c(0.8, 0.2))
training_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]
recommendation_system <- recommenderRegistry$get_entries(dataType ="realRatingMatrix")
recommendation_system$IBCF_realRatingMatrix$parameters
recommen_model <- Recommender(data = training_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model
class(recommen_model)
model_info <- getModel(recommen_model)
class(model_info$sim)
dim(model_info$sim)
top_items <- 20
image(model_info$sim[1:top_items, 1:top_items],
     main = "Heatmap of the first rows and columns")
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)

sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
top_recommendations <- 10 # the number of items to recommend to each user
predicted_recommendations <- predict(object = recommen_model,
                                     newdata = testing_data,
                                     n = top_recommendations)
predicted_recommendations
```

```
user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
movies_user1 <- predicted_recommendations@itemLabels[user1]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(movie_data,
                                     movie_data$movieId == movies_user1[index])$title)
}
movies_user2
recommendation_matrix <- sapply(predicted_recommendations@items,
                        function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with the
recommendations for each user
#dim(recc_matrix)
recommendation_matrix[,1:4]
```

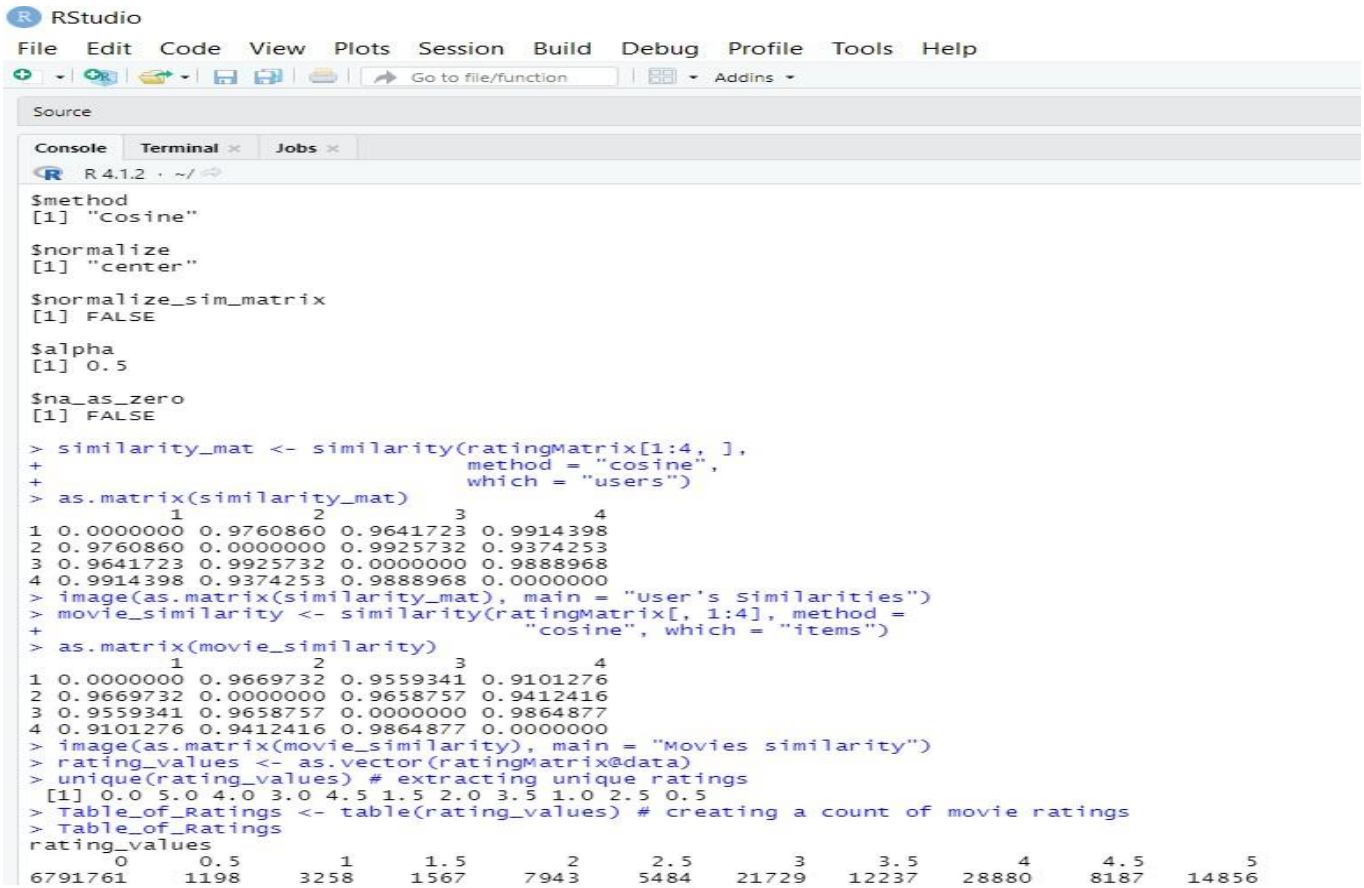
OUTPUTS

We first of all import the necessary libraries and extract our dataset that we are going to use. After that we start with data Pre-processing. From the above output table, we observe that the userId column, as well as the movieId column, consist of integers. Furthermore, we need to convert the genres present in the movie_data dataframe into a more usable format by the users. In order to do so, we will first create a one-hot encoding to create a matrix that comprises of corresponding genres for each of the films. In the next step of Data Pre-processing of R project, we will create a 'search matrix' that will allow us to perform an easy search of the films by specifying the genre present in our list. There are movies that have several genres, for example, Toy Story, which is an animated film also falls under the genres of Comedy, Fantasy, and Children. This applies to the majority of the films. For our movie recommendation system to make sense of our ratings through recommenderlabs, we have to convert our matrix into a sparse matrix one. This new matrix is of the class 'realRatingMatrix'.

```
> library(data.table)
data.table 1.14.2 using 2 threads (see ?getDTthreads).  Latest news: r-datatable.com

Attaching package: 'data.table'

The following objects are masked from 'package:reshape2':

    dcast, melt

Warning message:
package 'data.table' was built under R version 4.1.3
> library(reshape2)
> movie_data <- read.csv("C:\Users\HP\Downloads\IMDB-Dataset\movies.csv",stringsAsFactors=FALSE)
Error: '\U' used without hex digits in character string starting ""C:\U"
> movie_data <- read.csv("C:/Users/HP/Downloads/IMDB-Dataset/movies.csv",stringsAsFactors=FALSE)
> rating_data <- read.csv("C:/Users/HP/Downloads/IMDB-Dataset/ratings.csv")
> str(movie_data)
'data.frame':   10329 obs. of  3 variables:
 $ movieId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ title  : chr  "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to Exhale (1995)" ...
 $ genres : chr  "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy" "Comedy|Romance" "Comedy|Drama|Ro
mance" ...
> head(movie_data)
  movieId                               title                                        genres
1       1                    Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy
2       2                      Jumanji (1995)                  Adventure|Children|Fantasy
3       3             Grumpier Old Men (1995)                               Comedy|Romance
4       4            Waiting to Exhale (1995)                         Comedy|Drama|Romance
5       5 Father of the Bride Part II (1995)                                       Comedy
6       6                           Heat (1995)                        Action|Crime|Thriller
> summary(rating_data)
     userId          movieId           rating        timestamp
 Min.   :  1.0   Min.   :     1   Min.   :0.500   Min.   :8.286e+08
 1st Qu.:192.0   1st Qu.:  1073   1st Qu.:3.000   1st Qu.:9.711e+08
 Median :383.0   Median :  2497   Median :3.500   Median :1.115e+09
 Mean   :364.9   Mean   : 13381   Mean   :3.517   Mean   :1.130e+09
 3rd Qu.:557.0   3rd Qu.:  5991   3rd Qu.:4.000   3rd Qu.:1.275e+09
 Max.   :668.0   Max.   :149532   Max.   :5.000   Max.   :1.452e+09
> head(rating_data)
  userId movieId rating  timestamp
1      1      16    4.0 1217897793
2      1      24    1.5 1217895807
3      1      32    4.0 1217896246
4      1      47    4.0 1217896556
5      1      50    4.0 1217896523
6      1     110    4.0 1217896150
```



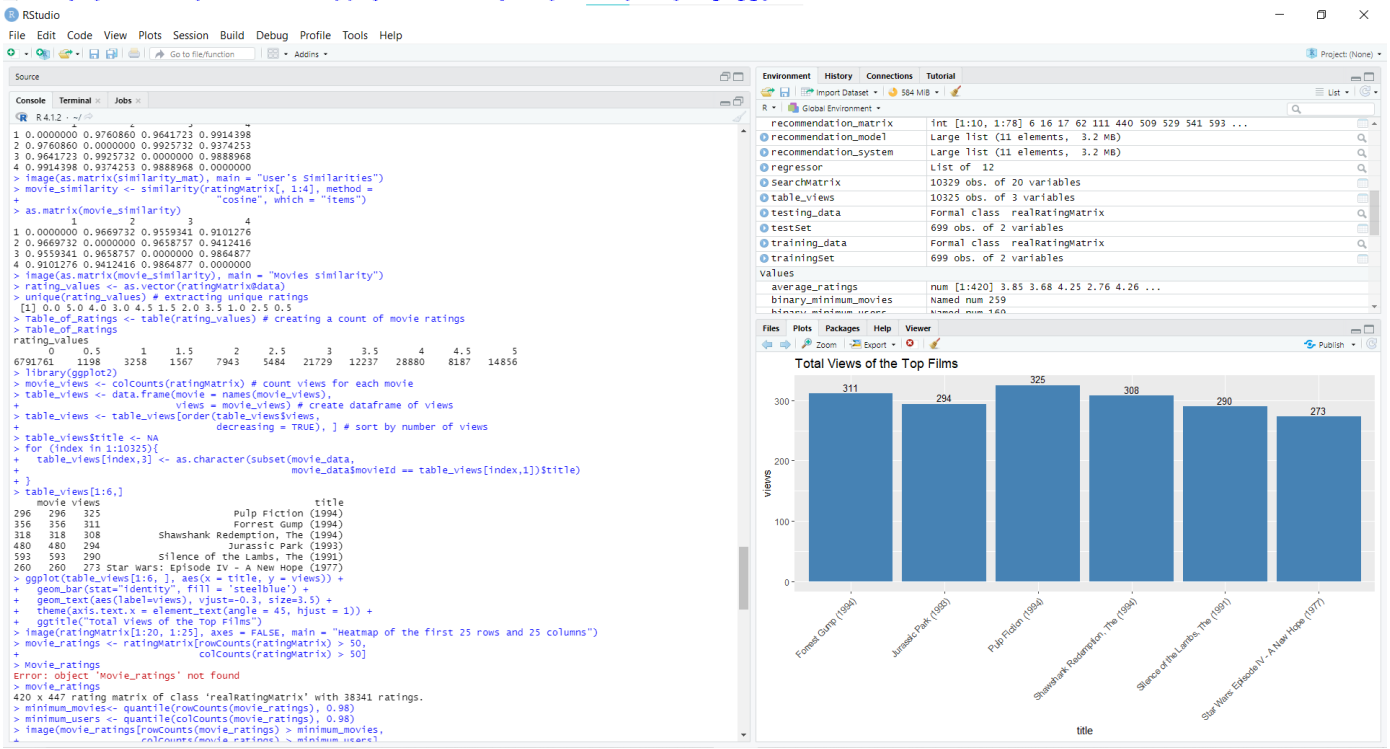
Collaborative Filtering involves suggesting movies to the users that are based on collecting preferences from many other users. For example, if a user A likes to watch action films and so does user B, then the movies that the user B will watch in the future will be recommended to A and vice-versa. Therefore, recommending movies is dependent on creating a relationship of similarity between the two users. With the help of recommenderlab, we can compute similarities using various operators like cosine, pearson as well as jaccard. We have taken four users and each cell in this matrix represents the similarity that is shared between the two users. Now, we will create a table of ratings that will display the most unique ratings.



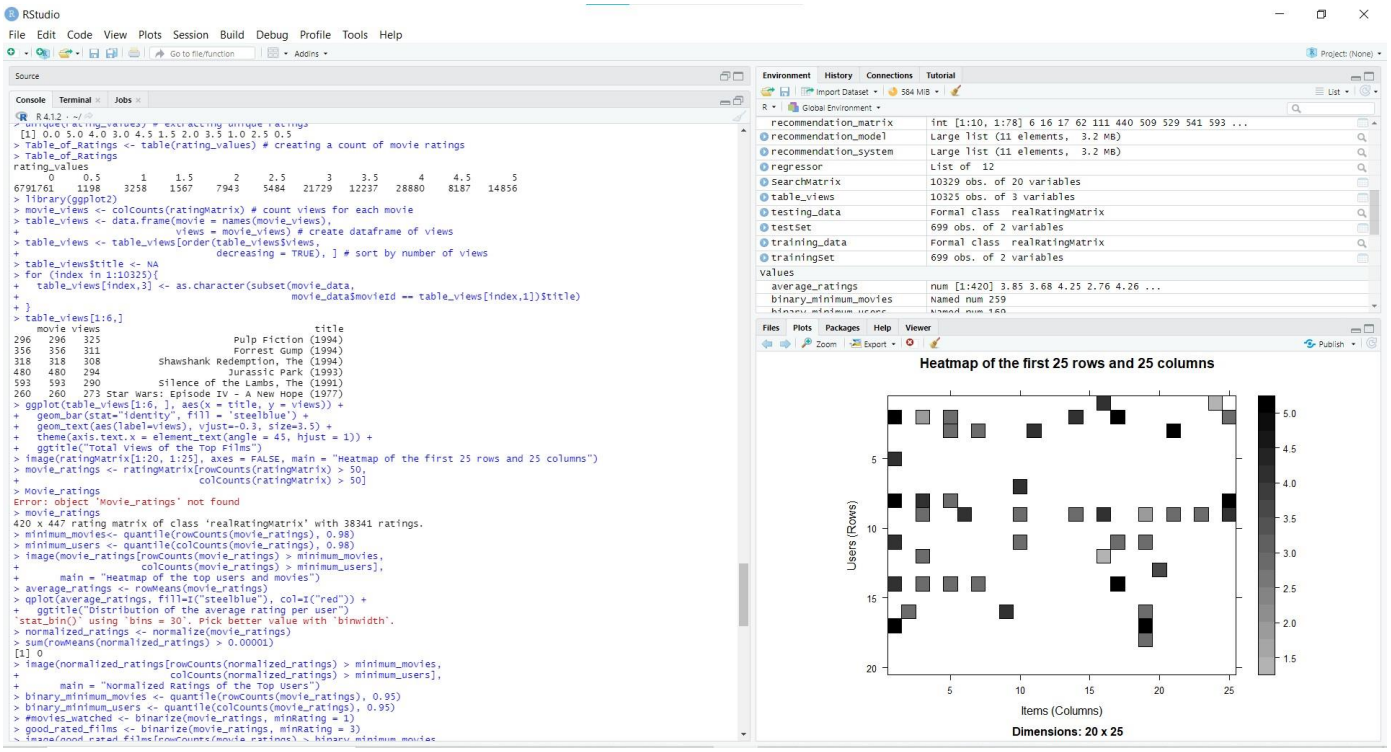
MOST VIEWED MOVIES VISUALISATION

We will first count the number of views in a film and then organize them in a table that would group them in descending order. From the above bar-plot, we observe that Pulp Fiction is the most-watched film followed by Forrest Gump.

```
> library(ggplot2)
> movie_views <- colCounts(ratingMatrix) # count views for each movie
> table_views <- data.frame(movie = names(movie_views),
+                           views = movie_views) # create dataframe of views
> table_views <- table_views[order(table_views$views,
+                                  decreasing = TRUE), ] # sort by number of views
> table_views$title <- NA
> for (index in 1:10325){
+   table_views[index,3] <- as.character(subset(movie_data,
+                                               movie_data$movieId == table_views[index,1])$title)
+ }
> table_views[1:6,]
     movie views                                      title
296    296   325                        Pulp Fiction (1994)
356    356   311                        Forrest Gump (1994)
318    318   308          Shawshank Redemption, The (1994)
480    480   294                      Jurassic Park (1993)
593    593   290          Silence of the Lambs, The (1991)
260    260   273 Star Wars: Episode IV - A New Hope (1977)
> ggplot(table_views[1:6, ], aes(x = title, y = views)) +
+   geom_bar(stat="identity", fill = 'steelblue') +
+   geom_text(aes(label=views), vjust=-0.3, size=3.5) +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
+   ggtitle("Total Views of the Top Films")
> image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")
> movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
+                               colCounts(ratingMatrix) > 50]
> Movie_ratings
Error: object 'Movie_ratings' not found
> movie_ratings
420 x 447 rating matrix of class 'realRatingMatrix' with 38341 ratings.
```



We will visualize a heatmap of the movie ratings. This heatmap will contain first 25 rows and 25 columns as follows –

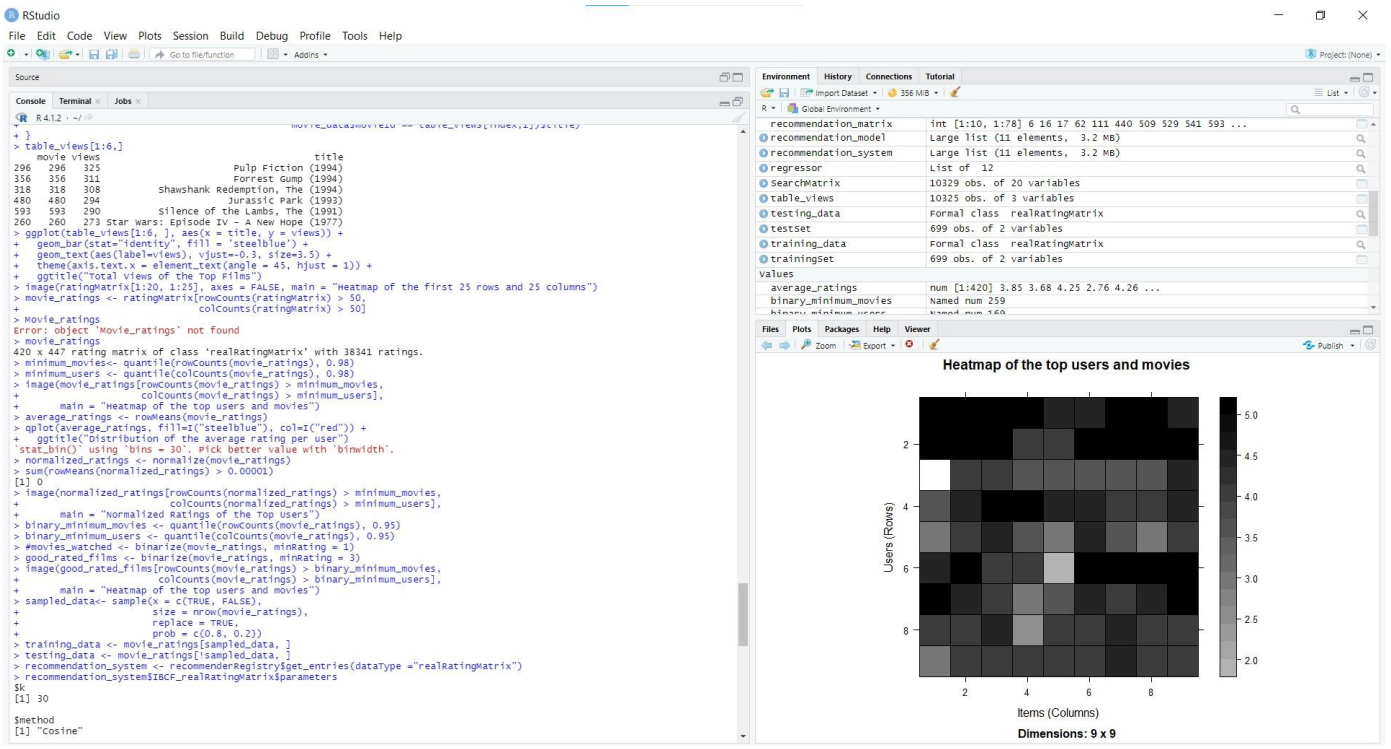


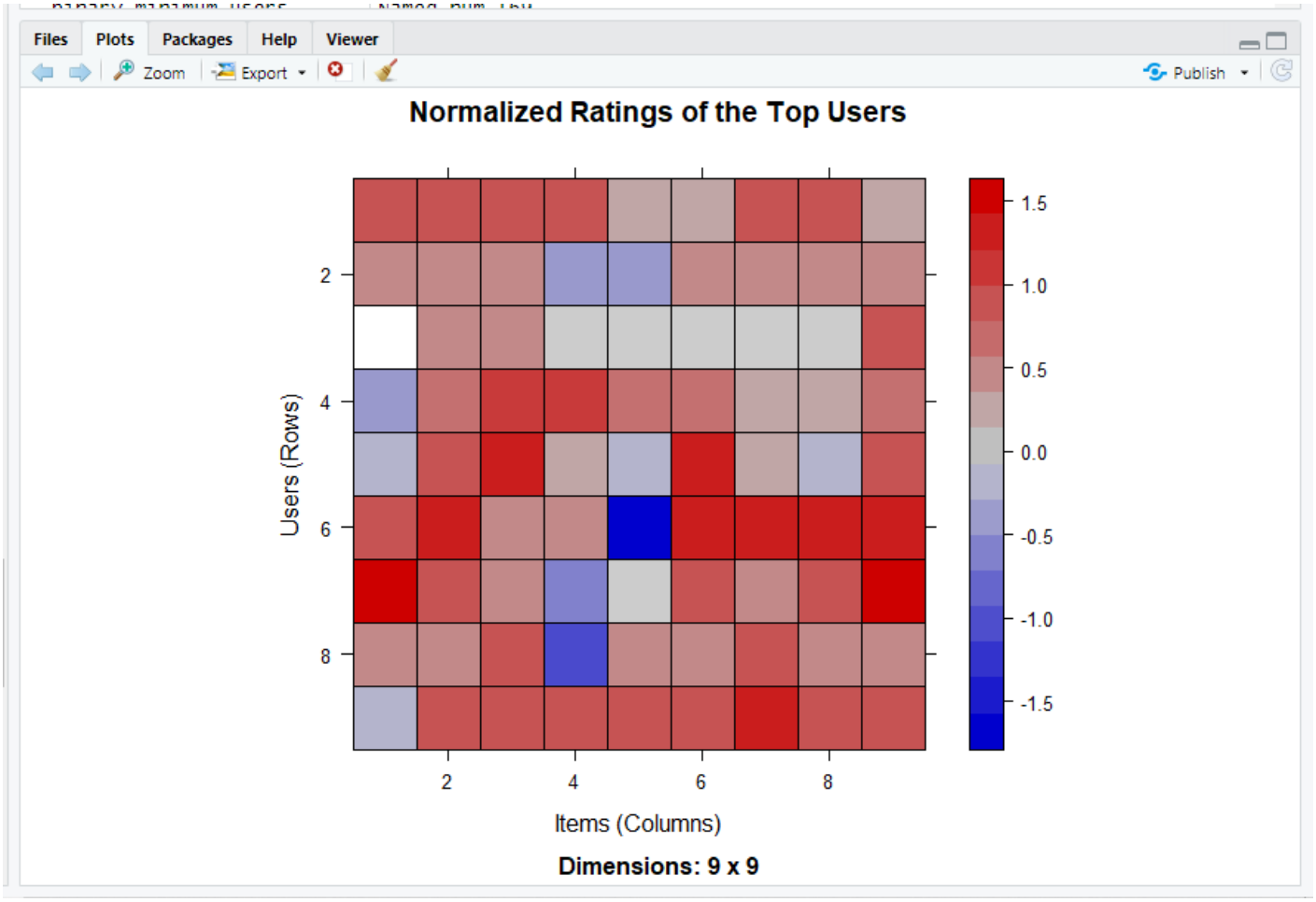
Performing Data Preparation

We will conduct data preparation in the following three steps –

- Selecting useful data.
- Normalizing data.
- Binarizing the data.

For finding useful data in our dataset, we have set the threshold for the minimum number of users who have rated a film as 50. This is also same for minimum number of views that are per film. This way, we have filtered a list of watched films from least-watched ones.





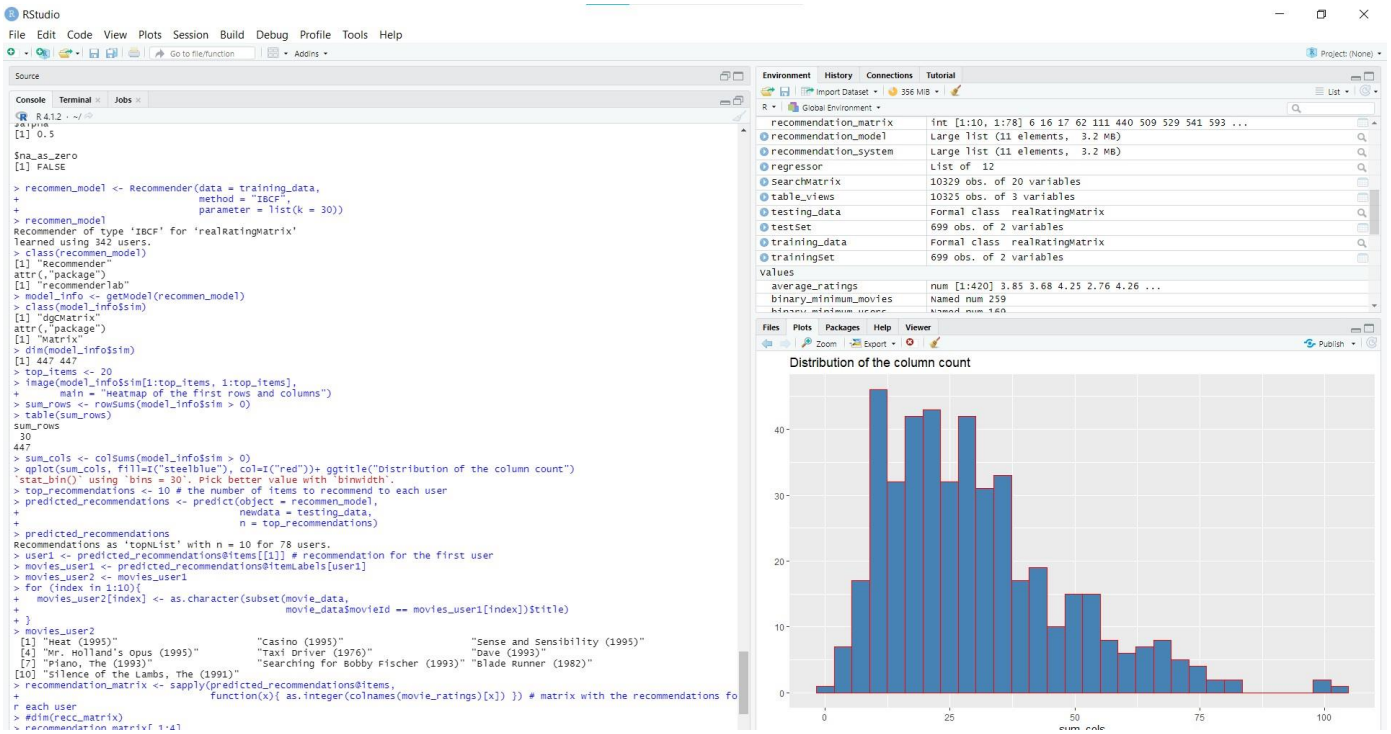
Collaborative Filtering System

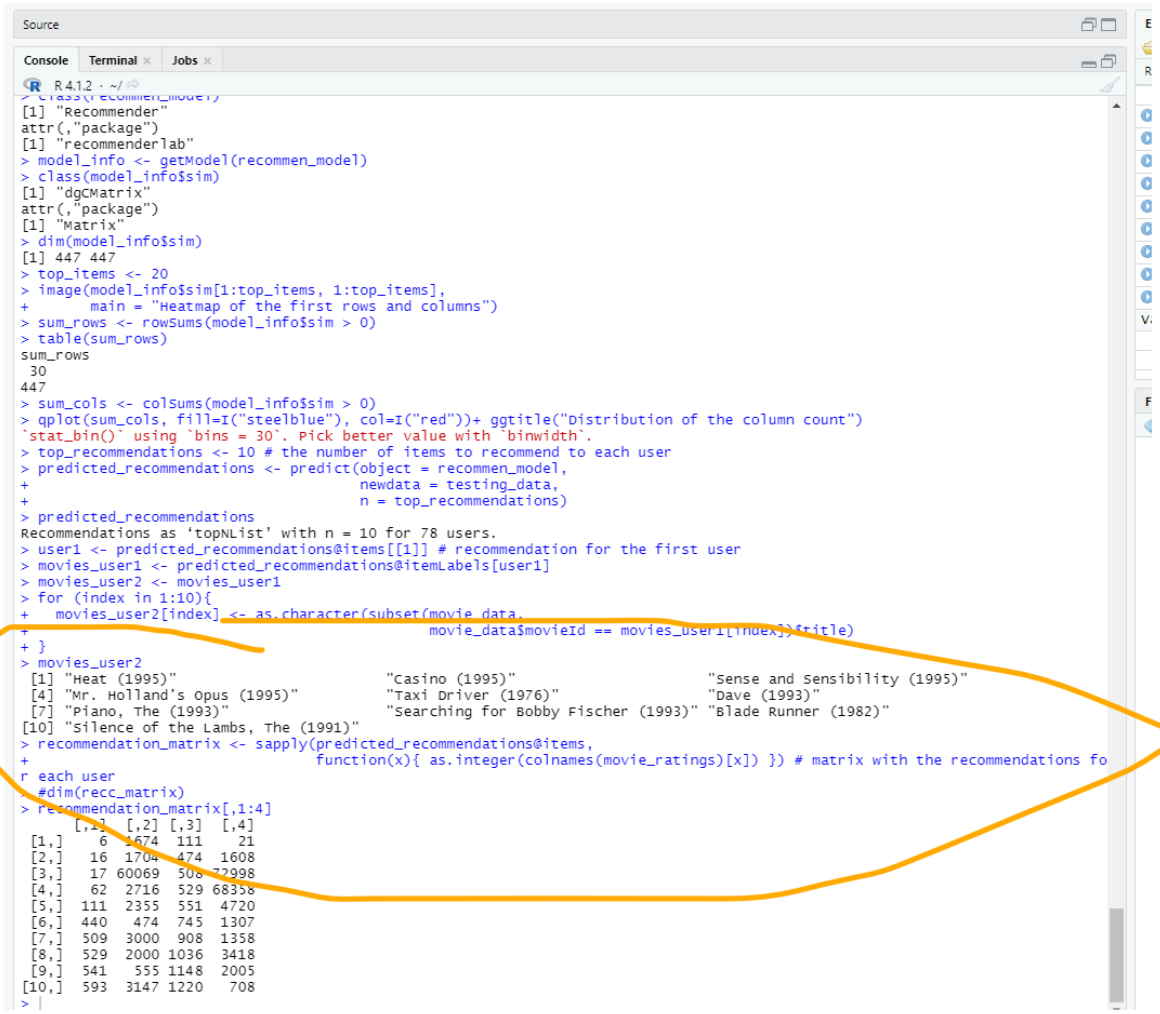
In this section of data science project, we will develop our very own Item Based Collaborative Filtering System. This type of collaborative filtering finds similarity in the items based on the people's ratings of them. The algorithm first builds a similar-items table of the customers who have purchased them into a combination of similar items. This is then fed into the recommendation system.

The similarity between single products and related products can be determined with the following algorithm –

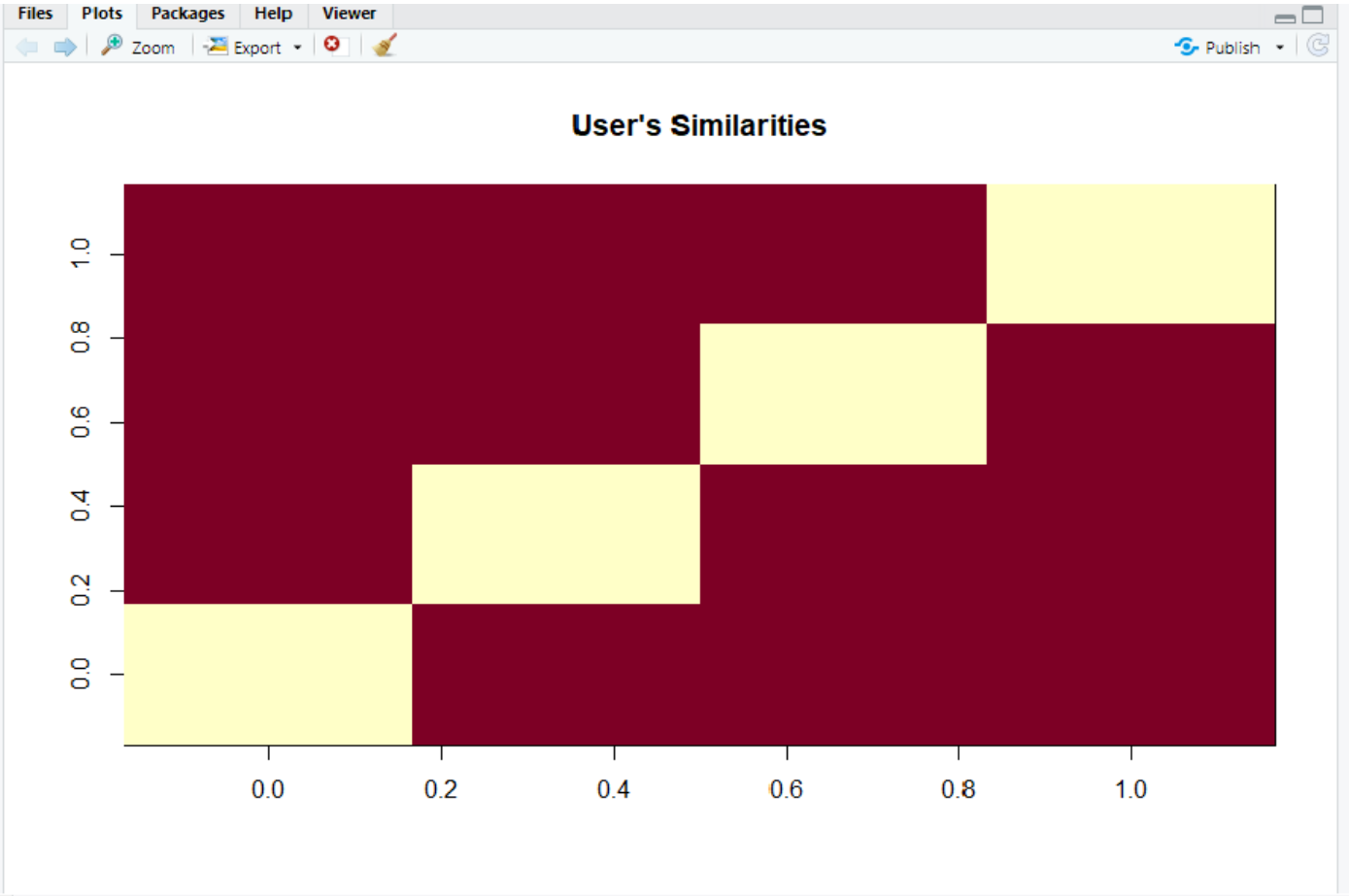
- For each Item i1 present in the product catalog, purchased by customer C.
- And, for each item i2 also purchased by the customer C.
- Create record that the customer purchased items i1 and i2.
- Calculate the similarity between i1 and i2.

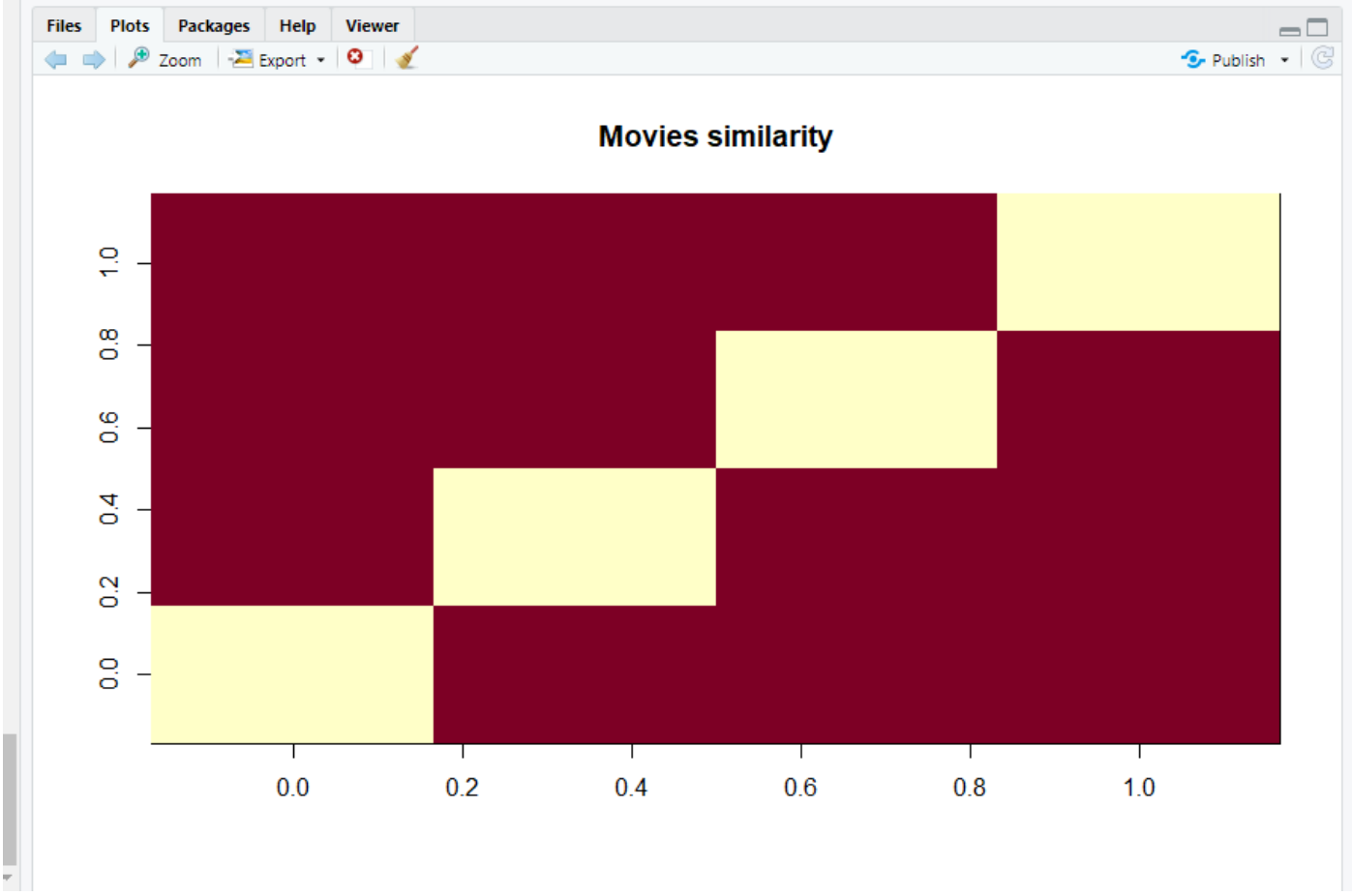
We will build this filtering system by splitting the dataset into 80% training set and 20% test set.

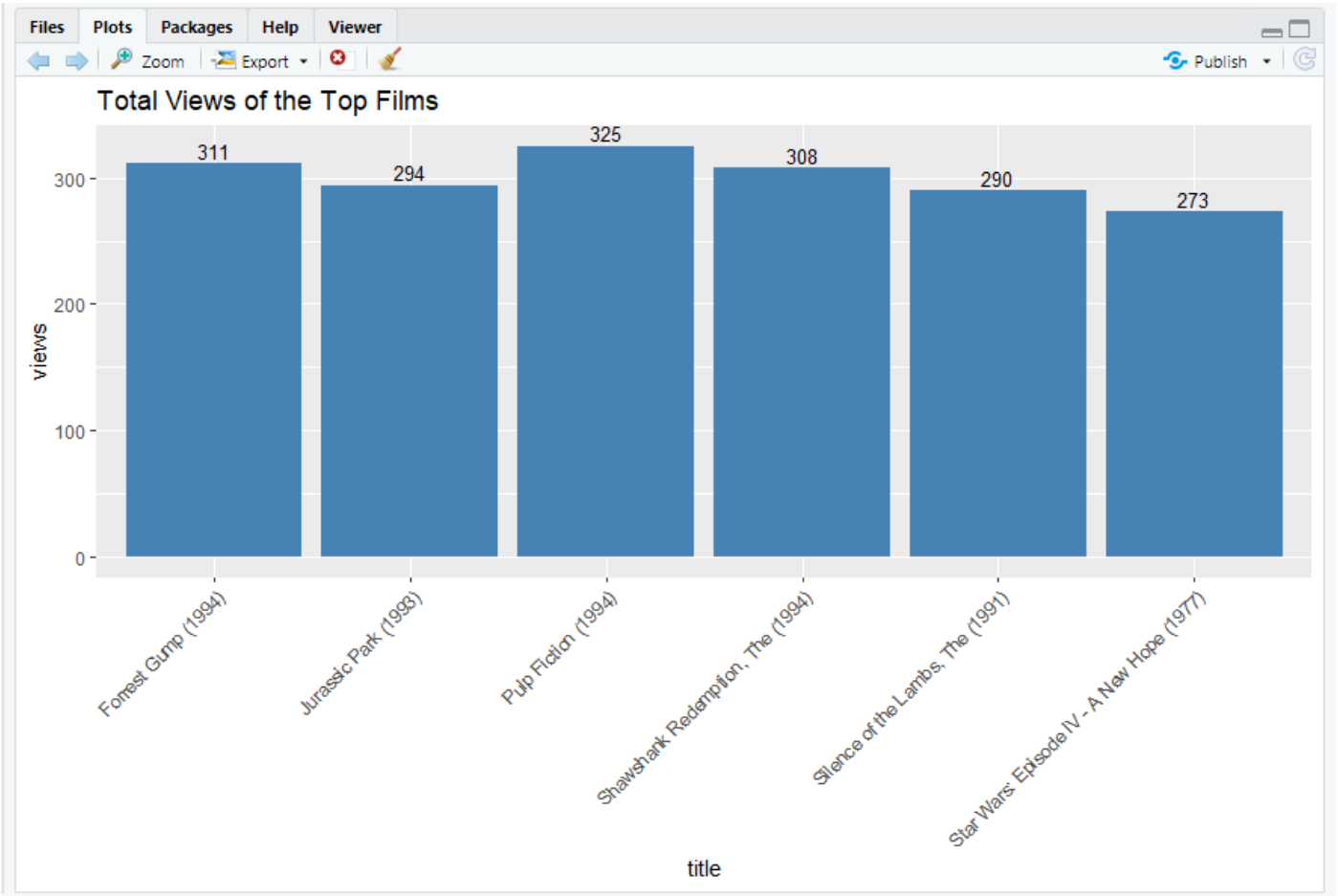

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
R 4.1.2 · ~/
$na_as_zero
[1] FALSE
> recommen_model <- Recommender(data = training_data,
+ method = "IBCF",
+ parameter = list(k = 30))
> recommen_model
Recommender of type 'IBCF' for 'realRatingMatrix'
learned using 342 users.
> class(recommen_model)
[1] "Recommender"
attr(,"package")
[1] "recommenderlab"
> model_info <- getModel(recommen_model)
> class(model_info$sim)
[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"
> dim(model_info$sim)
[1] 447 447
> top_items <- 20
> image(model_info$sim[1:top_items, 1:top_items],
+ main = "Heatmap of the first rows and columns")
> sum_rows <- rowSums(model_info$sim > 0)
> table(sum_rows)
sum_rows
30
447
> sum_cols <- colSums(model_info$sim > 0)
> qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> top_recommendations <- 10 # the number of items to recommend to each user
> predicted_recommendations <- predict(object = recommen_model,
+ newdata = testing_data,
+ n = top_recommendations)
> predicted_recommendations
Recommendations as 'topNList' with n = 10 for 78 users.
> user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
> movies_user1 <- predicted_recommendations@itemLabels[user1]
> movies_user2 <- movies_user1
> for (index in 1:10){
+ movies_user2[index] <- as.character(subset(movie_data,
+ movie_data$movieId == movies_user1[index])$title)
+ }
> movies_user2
[1] "Heat (1995)" "Casino (1995)" "Sense and Sensibility (1995)"
[4] "Mr. Holland's Opus (1995)" "Taxi Driver (1976)" "Dave (1993)"
[7] "Piano, The (1993)" "Searching for Bobby Fischer (1993)" "Blade Runner (1982)"
[10] "Silence of the Lambs, The (1991)"
> recommendation_matrix <- sapply(predicted_recommendations@items,
+ function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with the recommendations fo
r each user
> #dim(recc_matrix)
Environment History Connections Tutorial
Import Dataset 356 MiB
Global Environment
recommendation_matrix int [1:10, 1:78] 6 16 17 62 111 440 509 529 541 593 ...
recommendation_model Large list (11 elements, 3.2 MB)
recommendation_system Large list (11 elements, 3.2 MB)
regressor List of 12
SearchMatrix 10329 obs. of 20 variables
table_views 10325 obs. of 3 variables
testing_data Formal class realRatingMatrix
testSet 699 obs. of 2 variables
training_data Formal class realRatingMatrix
trainingSet 699 obs. of 2 variables
values
average_ratings num [1:420] 3.85 3.68 4.25 2.76 4.26 ...
binary_minimum_movies Named num 259
Files Plots Packages Help Viewer
Zoom Export Publish
Distribution of the column count
0 10 20 30 40
0 25 50 75 100
sum_cols

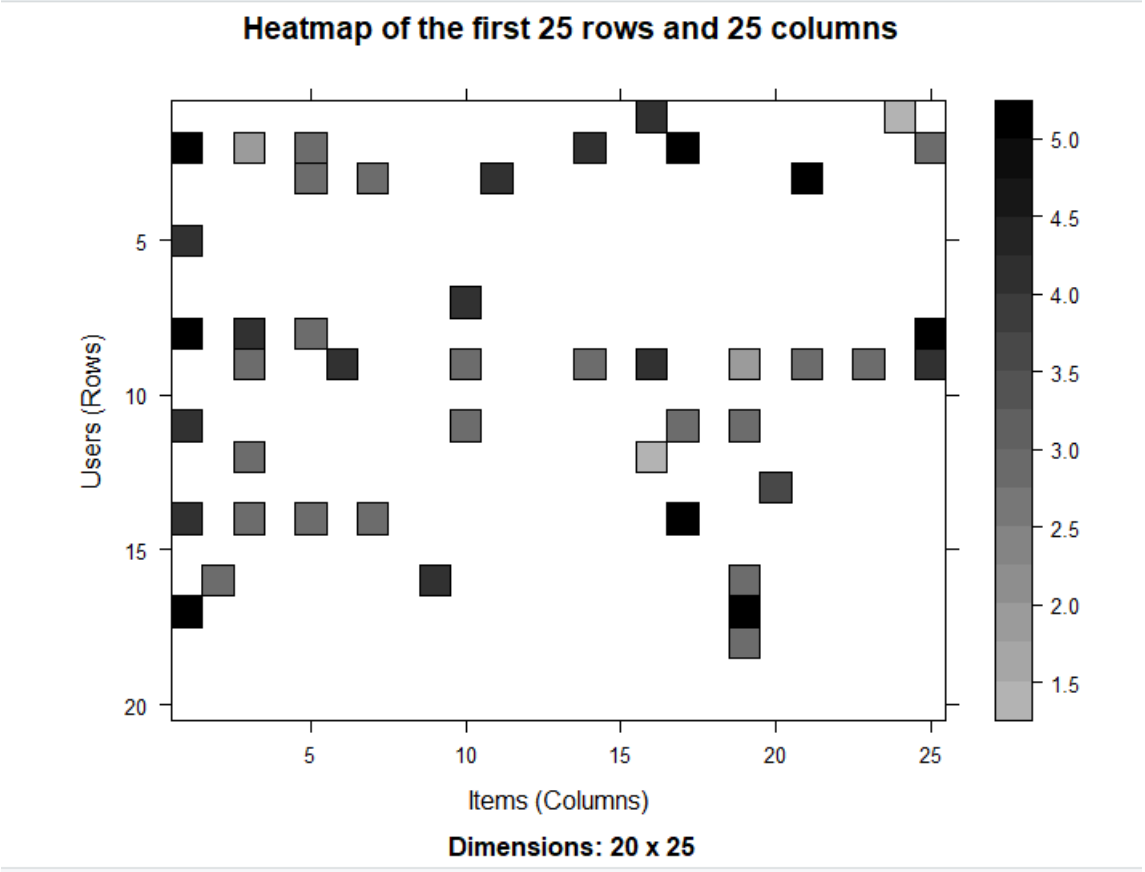

Source
Console Terminal Jobs
R 4.1.2 · ~/
[1] "Recommender"
attr(,"package")
[1] "recommenderlab"
> model_info <- getModel(recommen_model)
> class(model_info$sim)
[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"
> dim(model_info$sim)
[1] 447 447
> top_items <- 20
> image(model_info$sim[1:top_items, 1:top_items],
+ main = "Heatmap of the first rows and columns")
> sum_rows <- rowSums(model_info$sim > 0)
> table(sum_rows)
sum_rows
30
447
> sum_cols <- colSums(model_info$sim > 0)
> qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> top_recommendations <- 10 # the number of items to recommend to each user
> predicted_recommendations <- predict(object = recommen_model,
+ newdata = testing_data,
+ n = top_recommendations)
> predicted_recommendations
Recommendations as 'topNList' with n = 10 for 78 users.
> user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
> movies_user1 <- predicted_recommendations@itemLabels[user1]
> movies_user2 <- movies_user1
> for (index in 1:10){
+ movies_user2[index] <- as.character(subset(movie_data,
+ movie_data$movieId == movies_user1[index])$title)
+ }
> movies_user2
[1] "Heat (1995)" "Casino (1995)" "Sense and Sensibility (1995)"
[4] "Mr. Holland's Opus (1995)" "Taxi Driver (1976)" "Dave (1993)"
[7] "Piano, The (1993)" "Searching for Bobby Fischer (1993)" "Blade Runner (1982)"
[10] "Silence of the Lambs, The (1991)"
> recommendation_matrix <- sapply(predicted_recommendations@items,
+ function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with the recommendations fo
r each user
> #dim(recc_matrix)
> recommendation_matrix[,1:4]
[,1] [,2] [,3] [,4]
[1,] 6 1674 111 21
[2,] 16 1704 474 1608
[3,] 17 60069 508 72998
[4,] 62 2716 529 68358
[5,] 111 2355 551 4720
[6,] 440 474 745 1307
[7,] 509 3000 908 1358
[8,] 529 2000 1036 3418
[9,] 541 555 1148 2005
[10,] 593 3147 1220 708

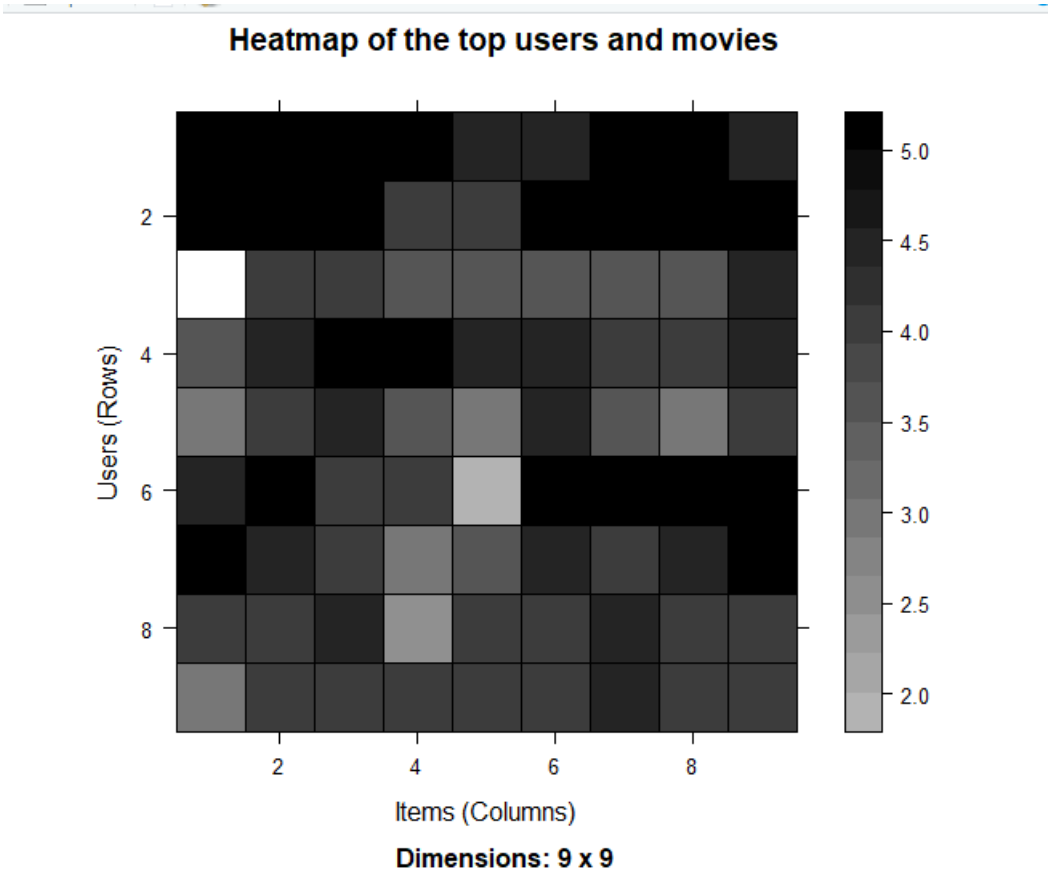
GRAPHS AND VISUALISATIONS

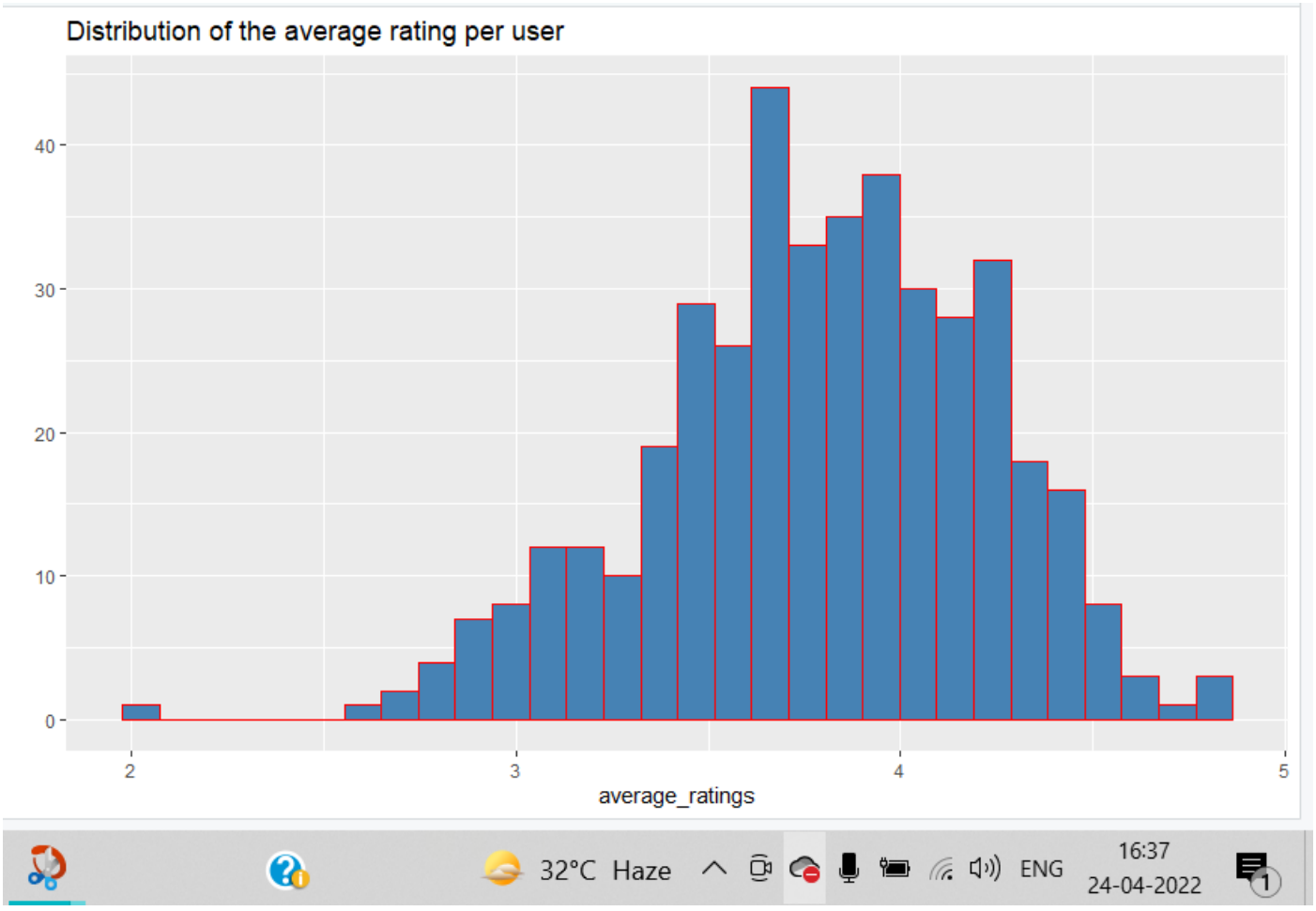


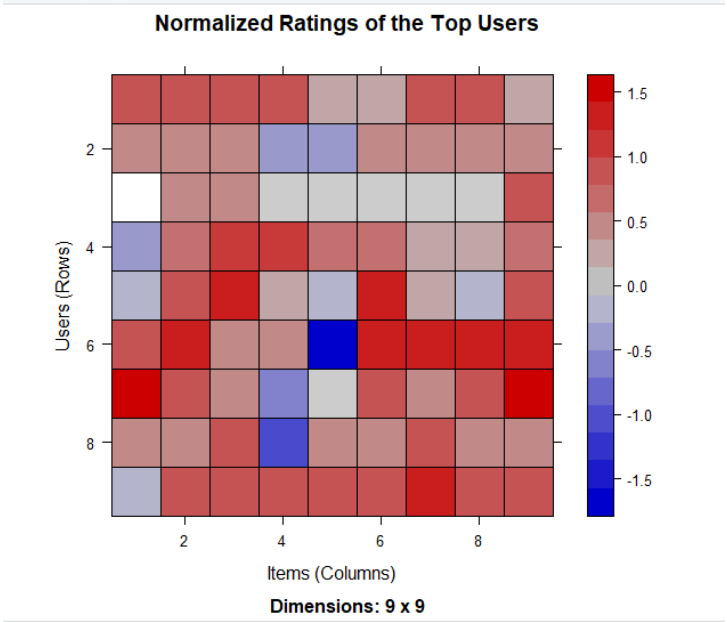


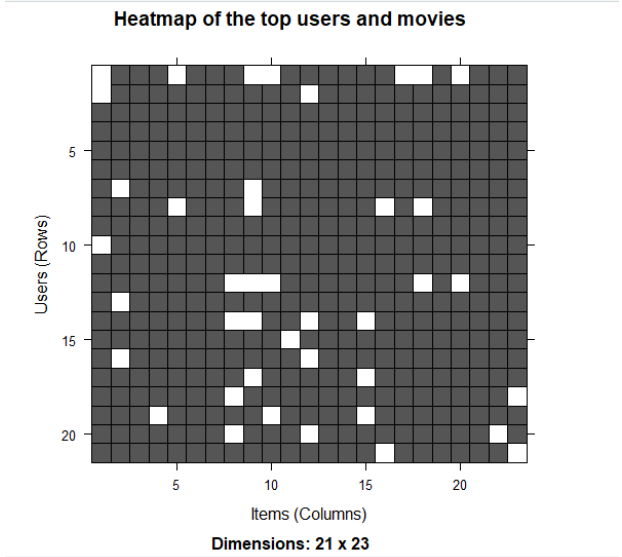

Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Total Views of the Top Films
311
294
325
308
290
273
300
200
100
0
views
Forrest Gump (1994)
Jurassic Park (1993)
Pulp Fiction (1994)
Shawshank Redemption, The (1994)
Silence of the Lambs, The (1991)
Star Wars: Episode IV - A New Hope (1977)
title

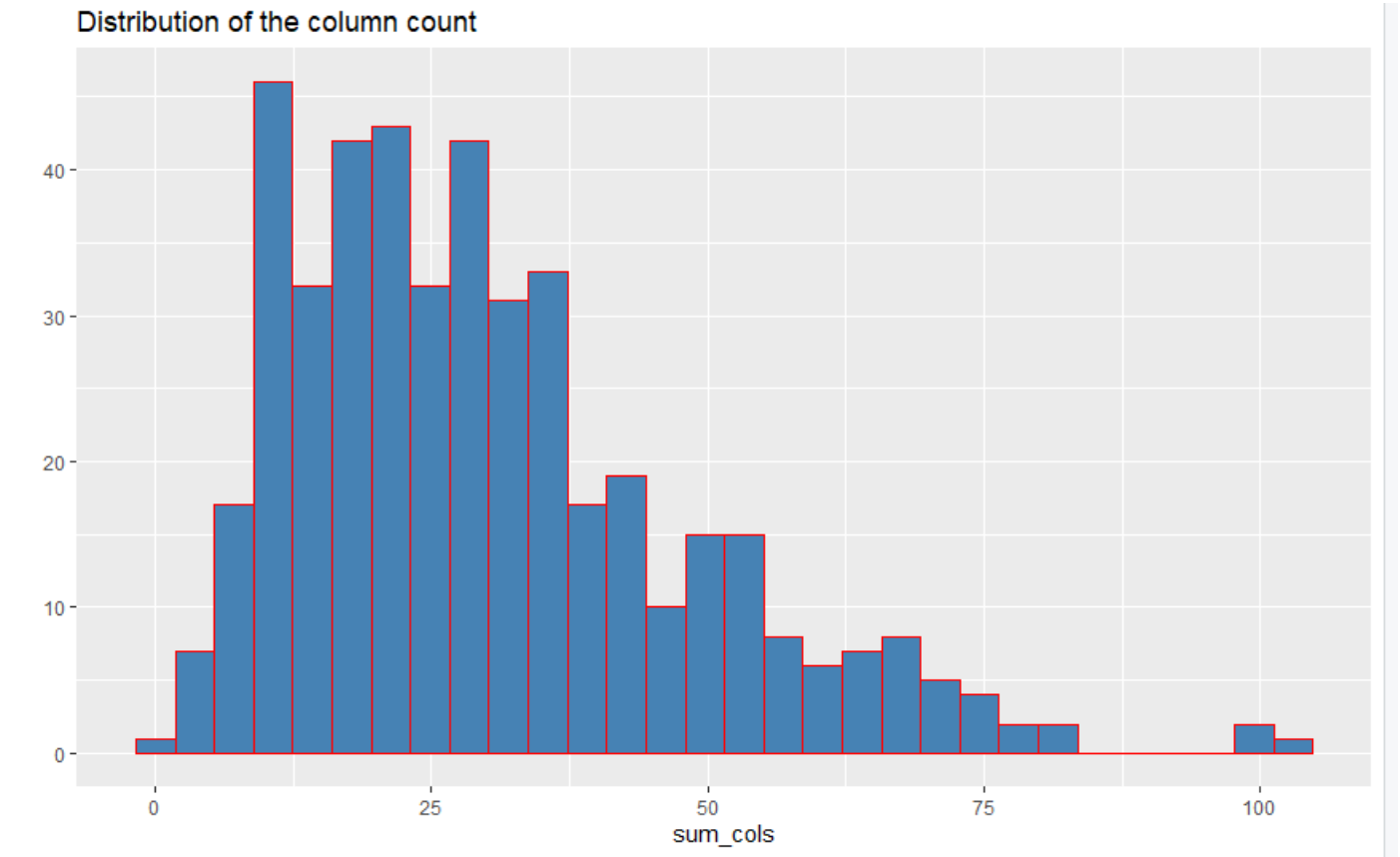

Heatmap of the first 25 rows and 25 columns
Users (Rows)
5
10
15
20
Items (Columns)
5
10
15
20
25
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Dimensions: 20 x 25












```
+                                    n = top_recommendations)
> predicted_recommendations
Recommendations as 'topNList' with n = 10 for 78 users.
> user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
> movies_user1 <- predicted_recommendations@itemLabels[user1]
> movies_user2 <- movies_user1
> for (index in 1:10){
+   movies_user2[index] <- as.character(subset(movie_data,
+                                              movie_data$movieId == movies_user1[index])$title)
+ }
> movies_user2
 [1] "Heat (1995)"                     "Casino (1995)"                   "Sense and Sensibility (1995)"
 [4] "Mr. Holland's Opus (1995)"       "Taxi Driver (1976)"              "Dave (1993)"
 [7] "Piano, The (1993)"               "Searching for Bobby Fischer (1993)" "Blade Runner (1982)"
[10] "Silence of the Lambs, The (1991)"
> recommendation_matrix <- sapply(predicted_recommendations@items,
+                                 function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with the recommendations fo
r each user
> #dim(recc_matrix)
> recommendation_matrix[,1:4]
     [,1] [,2] [,3] [,4]
```