

***Department Of Computer Science***

***Academic Year: 2021***

***Mcs Final (Morning)***

***Data Mining And Warehousing***

***Project Report***

***Entitled: Movie Recommendation System***

***SUBMITTED BY:***

*Name**Seat No*

* *Syeda Abeer Fatima P19101070*
* *Neha Adil P19101049*
* *Arsalan Rashid P19101010*
* *Abdul Moeez P19101002*

***INDEX***

|  |  |  |
| --- | --- | --- |
| S.NO | TOPIC | PAGE NO. |
| I | Introduction | 3 |
| II. | Literature Review | 5 |
| III. | Data Set | 8 |
| IV. | Data Analytics And Visualization | 9 |
| V. | Result And Conclusion | 40 |
| VI. | References | 41 |
| VII. | Appendix | 42 |

INTRODUCTION

Data Mining Concepts:

Let we first talk about the concept of data mining. Data mining is a field of joining of computer science and statistics used to determine patterns in the information set. The core aim of the data mining progression is to extract the useful information from the account of data and mold it into a comprehensible structure for future use. There are different process and practices used to carry out data mining successfully. Data mining is an iterative process within which progress is well-defined by discovery, through either automatic or manual approaches. Data mining is most useful in an exploratory examination scenario in which there are no fixed notions about what will constitute an "interesting" outcome. Data mining is the search for new, valuable, and nontrivial information in large volumes of data. It is a cooperative effort of individuals and computers. Best results are accomplished by balancing the knowledge of human experts in describing problems and goals with the search proficiencies of computers.

In practice, the two main goals of data mining tend to be prediction and description. Prediction includes using some variables or fields in the data set to predict unknown or future values of other variables of concentration. Description, on the other hand, emphases on discovery of patterns describing the data that can be interpreted by humans. Therefore, it is likely to place data-mining actions into one of two sorts:

* Predictive data mining, which produces the model of the system defined by the given data set, or
* Descriptive data mining, which produces new, nontrivial material created on the existing data set.

Data Mining Techniques:

Data mining implement its technique from various research areas, including statics machine learning, database systems, rough sets, visualization and neural networks.

1. **Statistical Approach**: Statistical models are assembled from a set of training data. Several statistical tools have been used for data mining including, Bayesian network, correlation analysis, regression analysis and cluster analysis.
2. **Machine Learning Approach**: The utmost mutual machine learning approaches used for data mining include conceptual learning, inductive concept learning and decision tree induction. By succeeding the path from root to leaf node an objects class can be determine by decision tree. Decision trees are brought from the training set and decision trees give classification rules.

In today's digital world where there is a boundless diversity of content consumed such as books, videos, articles, Films, etc., concluding material of one's choice has turn out to be an infallible task. Digital content on the other hand Providers want to involve more and more users in their service for maximum time. Where is it the recommender system comes into picture where content providers guide users by content User choice in this paper, we have proposed a movie recommendation system. Recommendation system supports users to find and select items (e.g., books, movies, restaurants) from a large Number available on the web or other electronic information sources. Given a large set of objects and a Describing user need, they offer the user a small set of items that are well suited description.

The key area of this project is to build a recommendation engine that recommends movies to users. This R project is designed to help you understand the functioning of how a recommendation system works. Before moving ahead in this movie recommendation system project, you need to know what recommendation system means.

What is a Recommendation System?

A recommendation system delivers ideas to the users through a filtering process that is built on user preferences and browsing history. The data about the user is taken as an input. The information is taken from the input that is in the form of browsing data. This information mirrors the former practice of the product as well as the allocated ratings. A recommendation system is a platform that provides its users with several contents based on their favorites and likings. A recommendation system takes the information about the user as an input. The recommendation system is an application of the [*machine learning algorithms*](https://data-flair.training/blogs/machine-learning-algorithms/).

Recommendation System is a major area which is very popular and useful for people to take proper decision. It is a method that helps user to find out the information which is beneficial for the user from variety of data available. When it comes to Movie Recommendation System, recommendation is done based on similarity between users (Collaborative Filtering) or by considering particular user's activity (Content Based Filtering) which he wants to engage with. So to overcome the limitations of collaborative and content based filtering generally, combination of collaborative and content based filtering is used so that a better recommendation system can be developed. Also, various similarity measures are used to find out similarity between users for recommendation. Various companies like face book which recommends friends, LinkedIn which recommends job, Pandora recommends music, Netflix recommends movies, Amazon recommends products etc. use recommendation system to increase their profit and also benefit their customers.

LITERATURE REVIEW

Nowadays, the recommendation system has made finding the things easy that we need. Movie recommendation systems purpose is helping movie fans by telling what movie to watch deprived of having to go through the lengthy process of selecting from a bulky set of movies which go up to thousands and millions that is time consuming and confusing. In this article, our aim is to lessen the human energy by suggesting movies based on the user’s interests.

Over the past decade, a large number of recommendation systems for a variety of domains have been developed and are in use. These recommendation systems use a variety of methods such as content-based approach, collaborative approach, knowledge-based approach, utility-based approach, hybrid approach, etc.

(Manoj Kumar, D.K Yadav, Ankur Singh, Vijay Kr. Gupta, August 2015) presented the movie recommendation system a MOVREC which is based on collaborative filtering approach. Collaborative filtering makes use of information provided by user. That information is analyzed and a movie is recommended to the users which are arranged with the movie with highest rating first. The system also has a provision for user to select attributes on which he wants the movie to be recommended. (Luis M. de Campos, Juan M. Fernández-Luna \*, Juan F. Huete, Miguel A. Rueda-Morales;, 2010) has analyzed two traditional recommender systems i.e., content based filtering and collaborative filtering. As both of them have their own drawbacks he proposed a new system which is a combination of Bayesian network and collaborative filtering. The proposed system is optimized for the given problem and provides probability distributions to make useful inferences. A hybrid system has been presented by ( Harpreet Kaur Virk, Er. Maninder Singh, April 2015). A mix of collaborative as well as content filtering algorithm is used by the system. The user specific information or item specific information is clubbed to form a cluster by (Utkarsh Gupta1 and Dr Nagamma Patil2, 2015) using chameleon. This is an efficient technique based on Hierarchical clustering for recommender system. ( Urszula Kużelewska, 2014) proposed clustering as a way to deal with recommender systems. Two methods of computing cluster representatives were presented and evaluated. Centroid-based solution and memory-based collaborative filtering methods were used as a basis for comparing effectiveness of the proposed two methods. The result was a significant increase in the accuracy of the generated recommendations when compared to just centroid-based method. (Costin-Gabriel Chiru, Vladimir-Nicolae Dinu , Ctlina Preda, Matei Macri, 2015) proposed Movie Recommender, a system which uses the information known about the user to provide movie recommendations. This system attempts to solve the problem of unique recommendations which results from ignoring the data specific to the user. To predict the difficulty level of each case for each trainee Hongli LIn et al. proposed a method called contentboosted collaborative filtering (CBCF). There are various types of recommender systems with different approaches and some of them are classified in this paper:

1. **Content-Based Filtering:**

In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in form of keywords (or features). A set of assigned keyword (terms, feature) might be seen by user profile which algorithm collects from the user interest. A set of keywords (or features) of an item is the Item profile.

* Advantages of content-based filtering are:
* They capable of recommending unrated items.
* We can easily explain the working of recommender system by listing the Content feature of an item.
* Content-based recommender system use needs only the rating of the concerned user, and not any other user of the system.
* Disadvantages of content-based filtering are:
* It does not work for a new user who has not rated any item yet as enough rating are required content-based recommender evaluates the user preferences and provide accurate recommendations.
* No recommendation of serendipitous items.
* Limited Content Analysis- The recommend does not work if the system fails to distinguish the items that a user likes from the items that he does not like.

1. **Collaborative filtering-based systems**:

Content based engine algorithm has several limitations. It is only capable of suggesting movies which are close to a certain movie. Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Collaborative filtering system recommends items based on similarity measures between users and/or items. The system recommends items preferred by similar users. This is based on the scenario where a person asks his friends, who have similar tastes, to recommend him some movies. It is basically of two types:

1. **User based filtering**:

These systems recommend products to the user that similar users may have liked. We can use either Pearson correlation or cosine similarity to measure similarity between two users. Each row represents a user and the column represent the different movie in the matrix and the last one record that show the similarity between the user and the target user is except in the matrix. The rating which is given by the user to that movie is represented by each cell. Although computing user-based Collaborative filtering is very simple, it suffers from several problems. A main issue is that users' preferences may change over time. It indicates that precompiling the matrix based on their neighboring users may lead to bad performance. To tackle this problem, we can apply item-based Collaborative filtering.

1. **Item based filtering:**

Instead of measuring the similarity between users, the item-based Collaborative filtering recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item- based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based Collaborative filtering does. It successfully avoids the problem posed by dynamic user preference as item-based Collaborative filtering is more static. However, several problems remain for this method. First, the main issue is scalability. The computation grows with both the customer and the product. O(mn) with m users and n items is the worst case complexity. In addition, sparsity is another concern.

* Advantages of collaborative filtering-based systems:
* It is dependent on the relation between users which implies that it is content-independent.
* They can make real quality assessment of items by considering other people’s experience.
* Disadvantages of collaborative filtering are:
* Initial rater problem: Collaborative filtering systems cannot provide recommendations for new items because there is no user rating on which to base a prediction.
* Gray Sheep: For a CF based system to work; a group with similar characteristics is required. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.

A recommendation system also finds a similarity between the different products. For example, Netflix Recommendation System provides you with the recommendations of the movies that are similar to the ones that have been watched in the past. Furthermore, there is a collaborative content filtering that provides you with the recommendations in respect with the other users who might have a similar viewing history or preferences. 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.

Collaborative filtering is one of the most effective and adequate technique used in recommendation. The fundamental aim of the recommendation is to provide prediction of the different items in which a user would be interested in based on their preferences. Recommendation systems based on collaborative filtering techniques are able to provide approximately accurate prediction when there is enough data. User based collaborative filtering techniques have been very powerful and success in the past to recommend the items based on user's preferences. But, there are also some certain challenges such as scalability and sparsity of data which increases as the number of users and items increases. In a large website, it is difficult to find the interested information in a certain time. But the recommendation system filters out information and items that are best suitable for us. Although there are different recommendation approaches, yet collaborative filtering technique is very popular because of the effectiveness. In this work, movie recommender system has been described, which basically uses item-based technique of collaborative filtering to provide the recommendations of items, which is dynamic and will learn from the positive feedback.

**DATA SET**

A Data set is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. Data sets describe values for each variable for unknown quantities such as height, weight, temperature, volume, etc. of an object or values of random numbers.

Any expert system relies heavily on an extensive dataset. In order to get reliable results, it is important that we have a good dataset. Looking through some freely available movie datasets and In order to build our recommendation system, we have used the Movie Lens Dataset. You can find the movies.csv and ratings.csv file that we have used in our Recommendation System Project [*here*](https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFlDW2jdQGD/view). The data consists of 105339 ratings applied over 10329 movies.

Our data set contains 10329 different movies from from 1995 till 2015. The data set included variables such as: movie id, movie title, movie year, and genre of the movie. Movie id is basically used to uniquely identify a particular movie. Film genres are categories that define a movie based on its narrative elements. Each genre is unique in the types of stories they tell.13 classic movie genres are action, romance, thriller, horror, adventure, comedy, drama, fantasy, musicals, mystery, sports science fiction. Ratings.csv includes the column of rating of the particular movie.

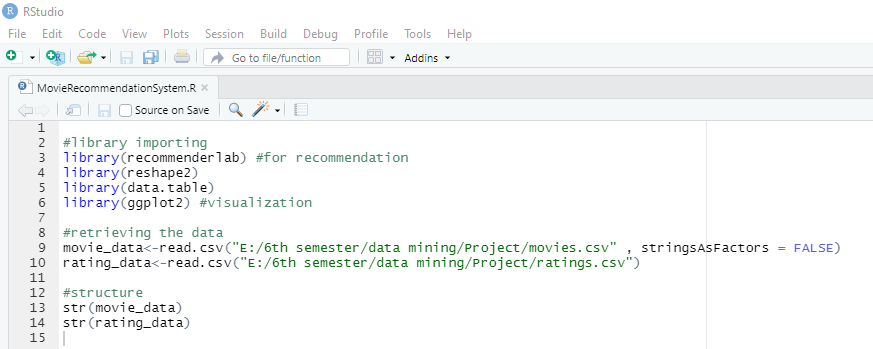
Different characteristics have different loads [66]. In our assessment we have found that the most fitting suggestions that can be delivered should be established on the assessments provided for the movies by existing customers, right now have given more importance to the rating characteristic than various properties. For the user to get the recommendation he has to rate at least 6 movies. If he/she is a new user and has not rated any movies then he is expected to search for a random movie or any movie of his interest in the search box and rate at least 6 movies. Only then the movies will be recommended to him/her.

DATA ANALYTICS AND VISUALIZATION

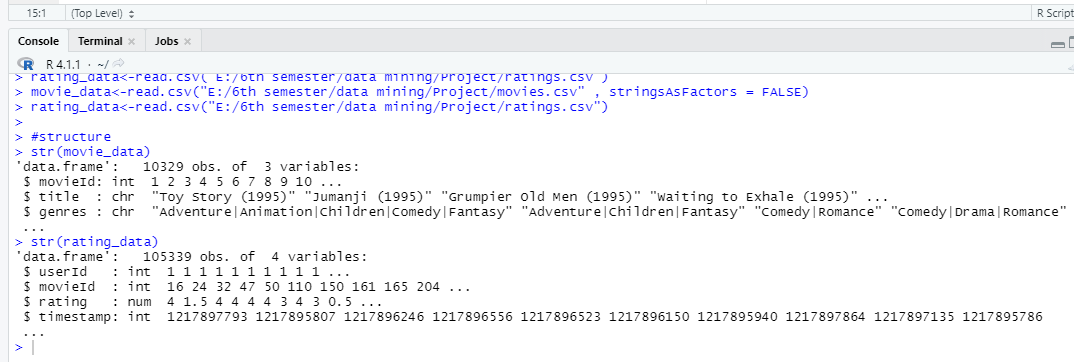
* IMPORTING ESSENTIAL LIBRARIES AND RETRIEVING DATA:

In our Data Science project, we used these four packages – ‘recommenderlab’, ‘ggplot2’, ‘data.table’ and ‘reshape2’. Then we retrieved our data from movies.csv into movie\_data dataframe and ratings.csv into rating\_data. We used the str() function to display information about both dataframes.

CODE

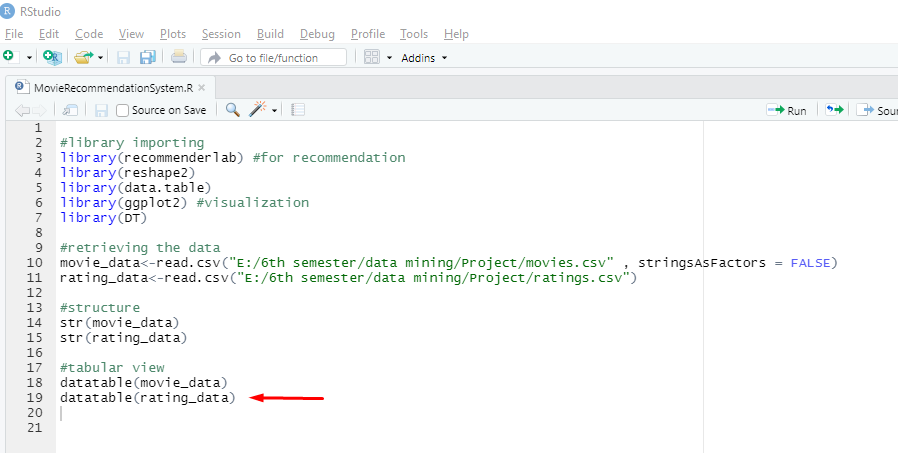


OUTPUT

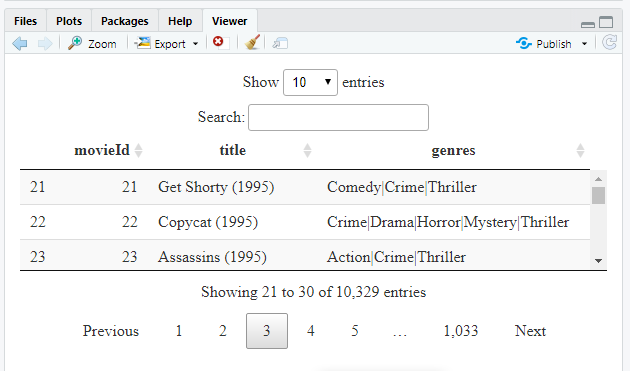


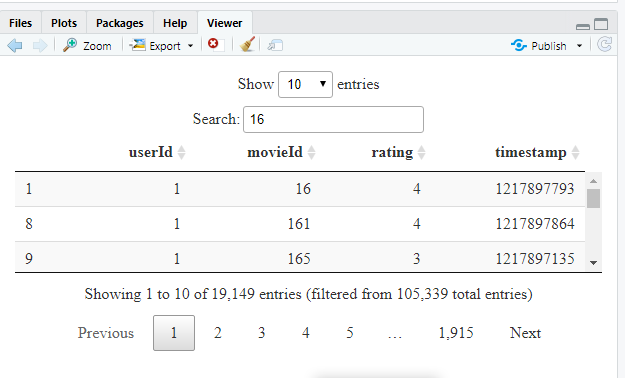
We are now able to display data in a tabular form:

CODE



OUTPUT (MOVIE DATA TABLE)

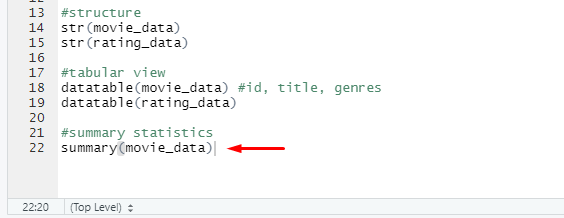




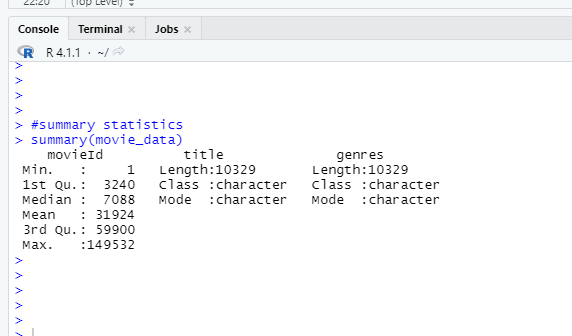
OUTPUT (RATING DATA TABLE)

Similarly, we can output the summary as well as the first six lines of the ‘movie\_data’ dataframe and ‘rating\_data’ dataframe –

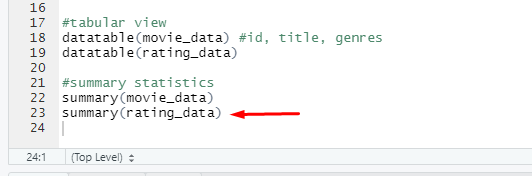
CODE



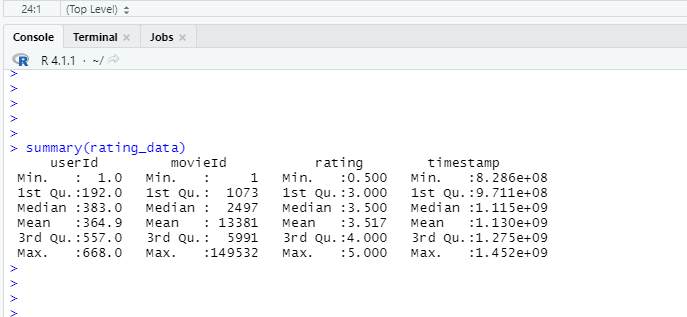
OUTPUT



CODE



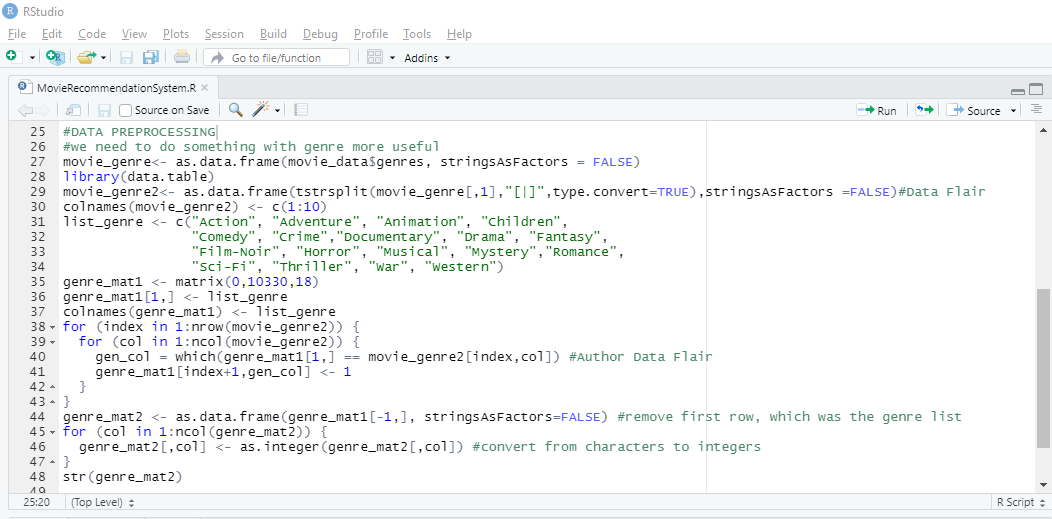
OUTPUT



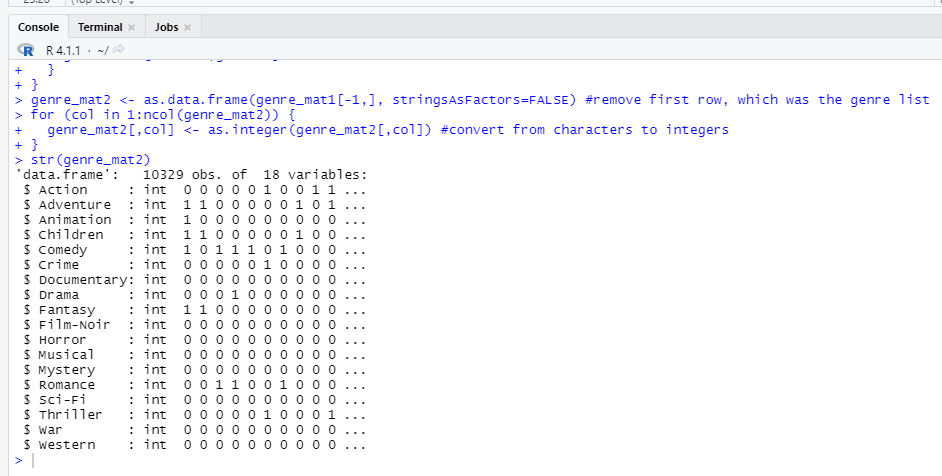
* DATA PRE-PROCESSING:

From the above 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 film.

CODE

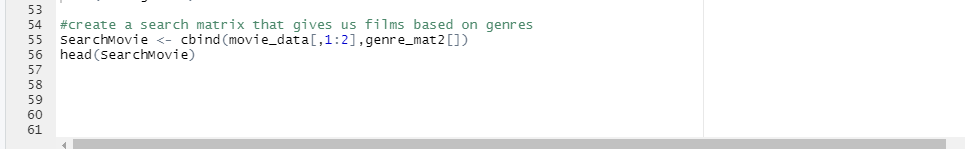


OUTPUT

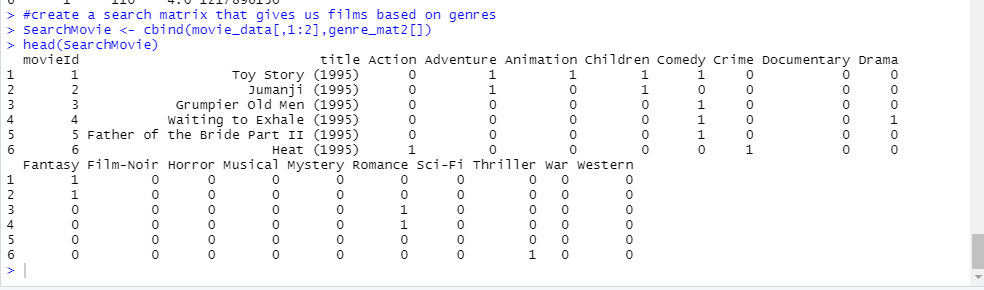


In the next step of Data Pre-processing of R project, we will create a ‘search matrix’ that will allow us to perform an easy search of the films by specifying the genre present in our list.

CODE



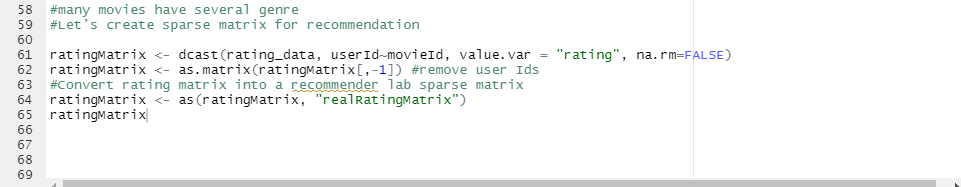
OUTPUT

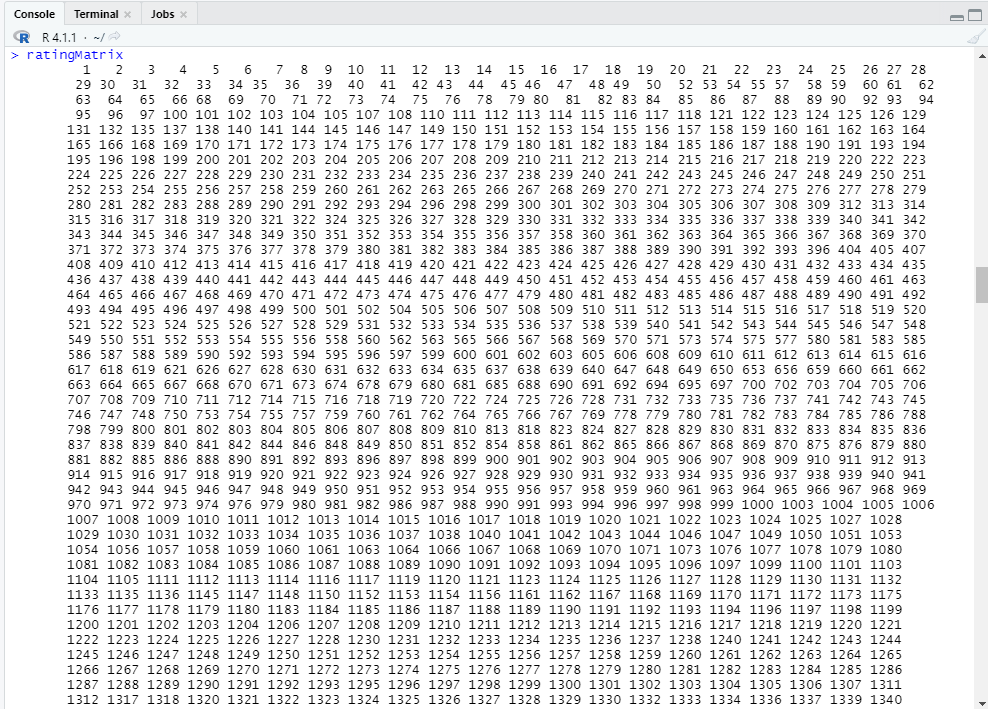


There are movies that have several genres, for example, Toy Story, which is an animated film also falls under the genres of Comedy, Fantasy, and Children. This applies to the majority of the films.

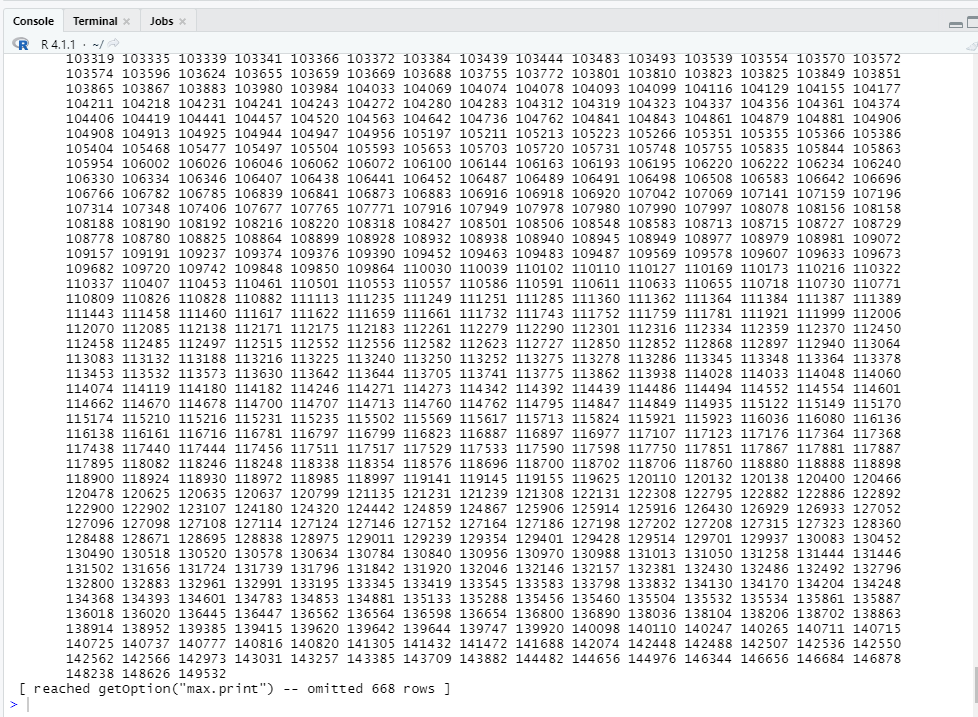
For our movie recommendation system to make sense of our ratings through recommenderlabs, we have to convert our matrix into a sparse matrix one. This new matrix is of the class ‘realRatingMatrix’. This is performed as follows:

CODE



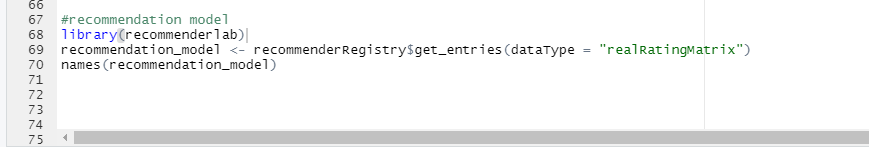


OUTPUT

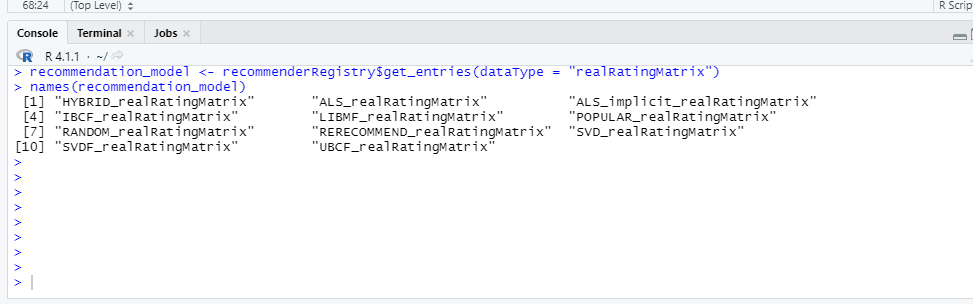


Let us now overview some of the important parameters that provide us various options for building recommendation systems for movies-

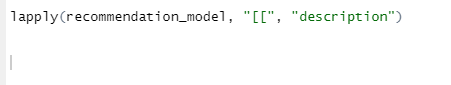
CODE



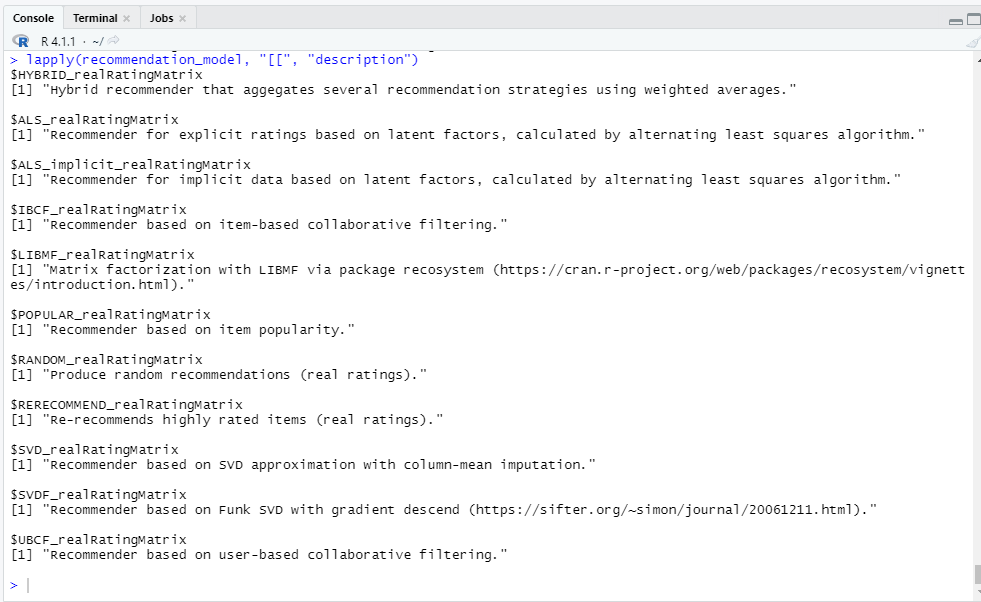
OUTPUT



CODE

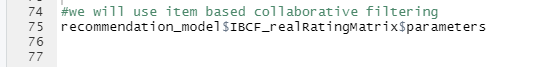


OUTPUT

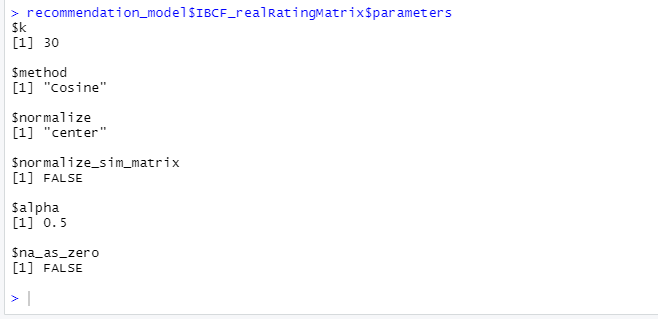


We will implement a single model in our R project – Item Based Collaborative Filtering.

CODE



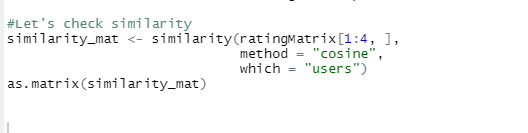
OUTPUT



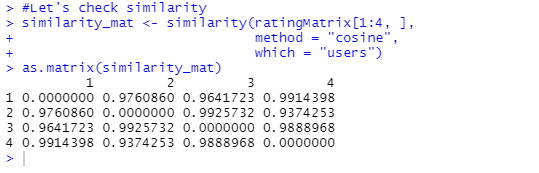
* EXPLORING SIMILAR DATA:

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.

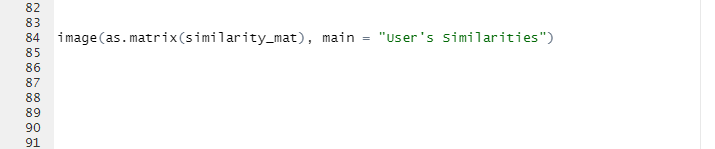
CODE



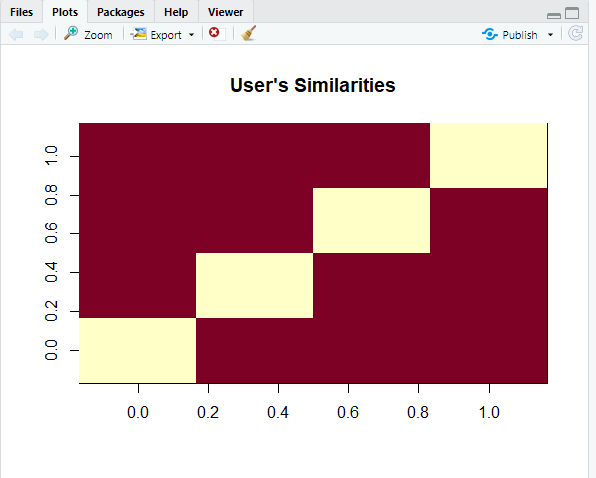
OUTPUT



CODE (SIMILARITY MATRIX)



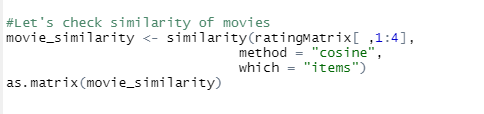
OUTPUT



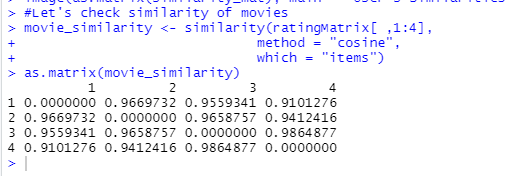
In the above matrix, each row and column represents a user. We have taken four users and each cell in this matrix represents the similarity that is shared between the two users.

Now, we delineate the similarity that is shared between the films –

CODE



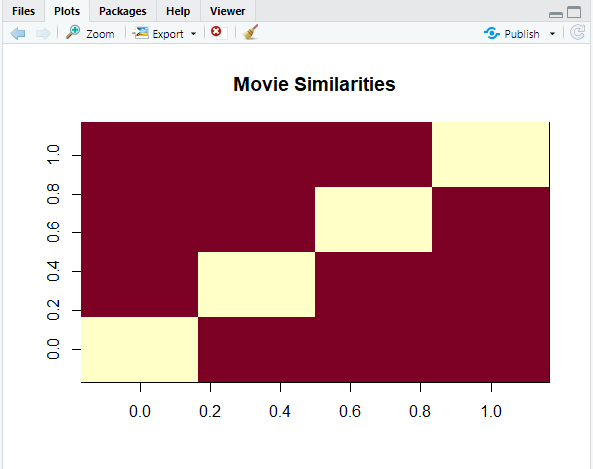
OUTPUT



CODE

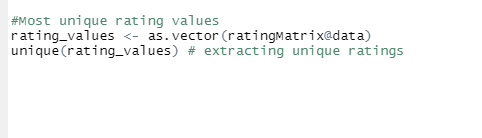


OUTPUT

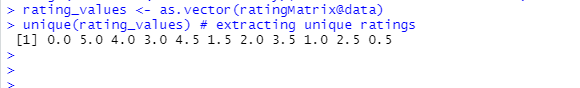


Let us now extract the most unique ratings –

CODE

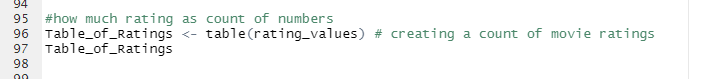


OUTPUT

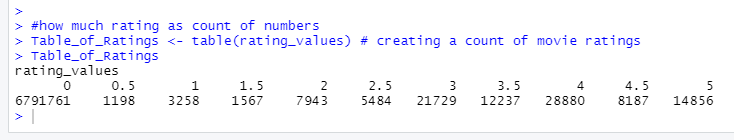


Now, we will create a table of ratings that will display the most unique ratings.

CODE



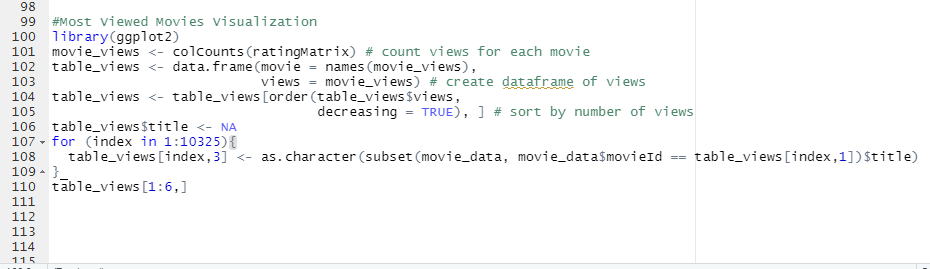
OUPUT



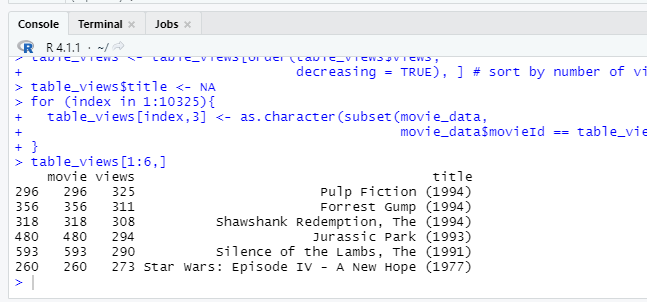
* MOST VIEWED MOVIE VISUALIZATION:

In this section of the machine learning project, we will explore the most viewed movies in our dataset. 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.

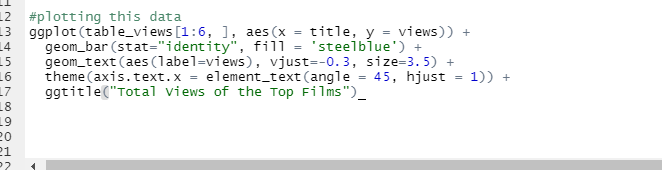
CODE



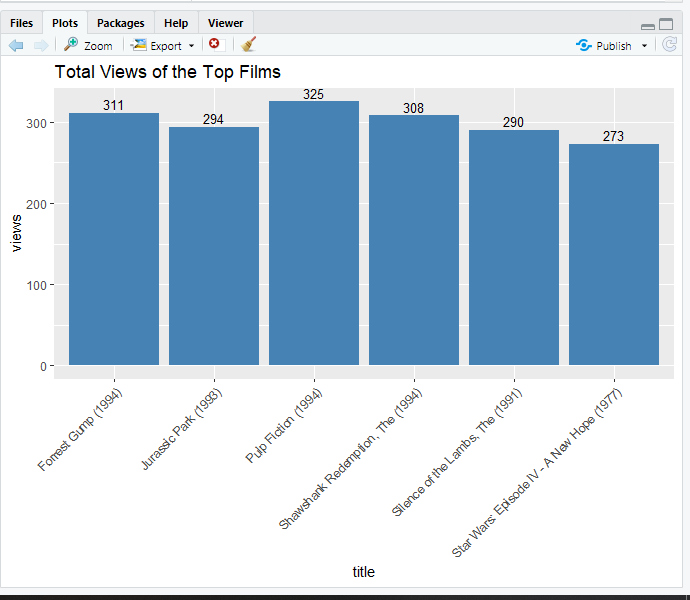
OUTPUT

Now, we will visualize a bar plot for the total number of views of the top films. We will carry this out using ggplot2.

CODE (PLOT)



OUTPUT

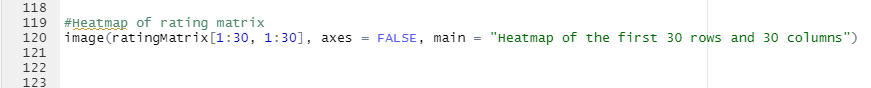


*From the above bar-plot, we observe that Pulp Fiction is the most-watched film followed by Forrest Gump.*

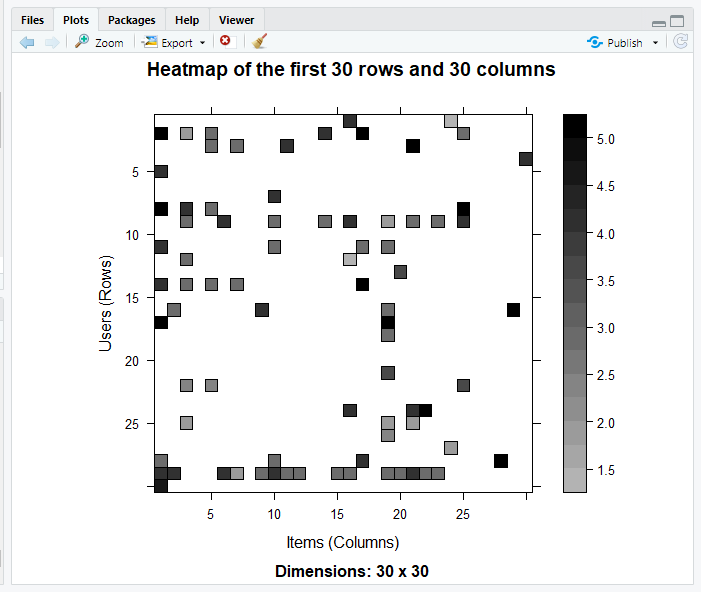
* HEAT MAP OF MOVIE RATING:

Now, in this data science project of Recommendation system, we will visualize a heatmap of the movie ratings. This heatmap will contain first 30 rows and 30 columns as follows –

CODE (HEATMAP)



OUPUT (HEATMAP)



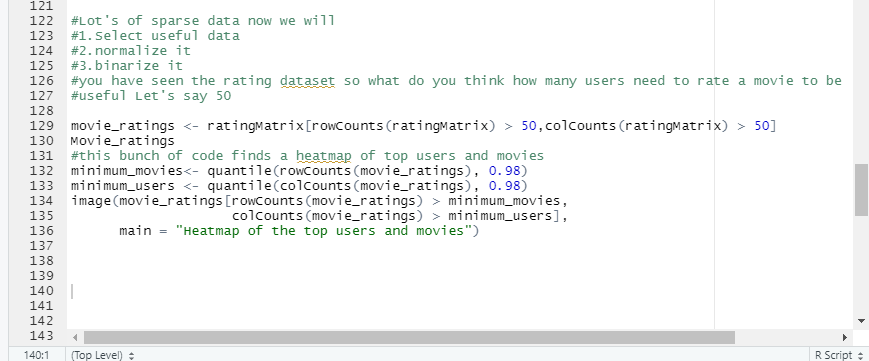
* PERFORMING DATA PREPRATION:

We will conduct data preparation in the following three steps –

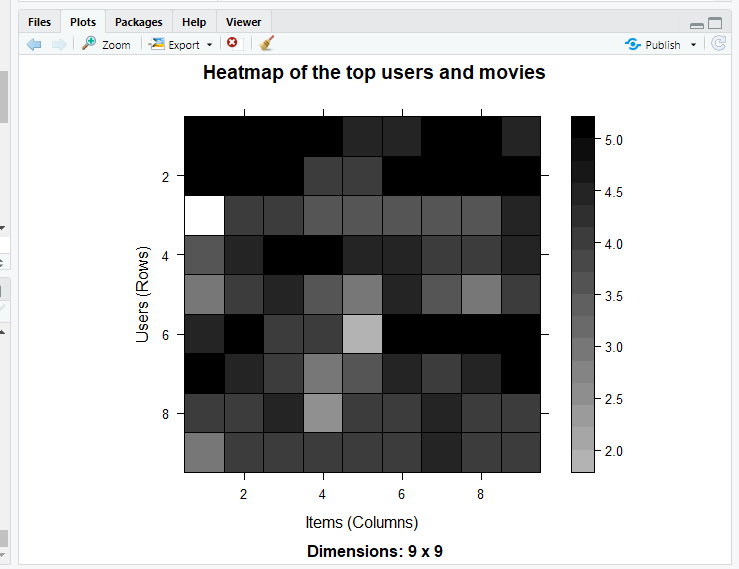
* Selecting useful data.
* Normalizing data.
* Binarizing the data.

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.We can now delineate our matrix of relevant users as follows –

CODE

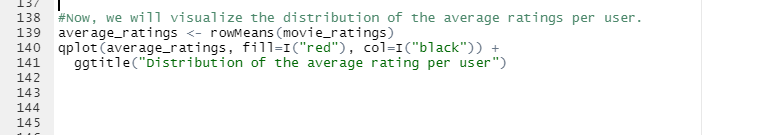


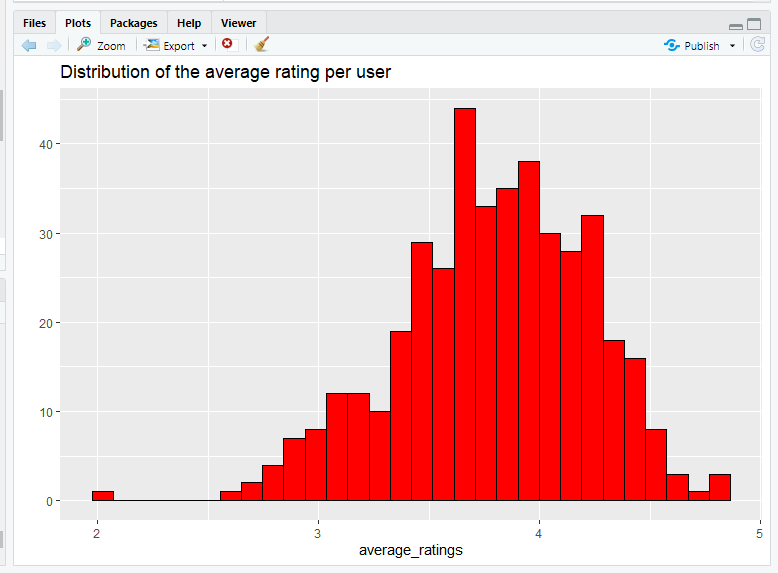
OUPUT (HEATMAP)



Now, we have visualized the distribution of the average ratings per user.

CODE



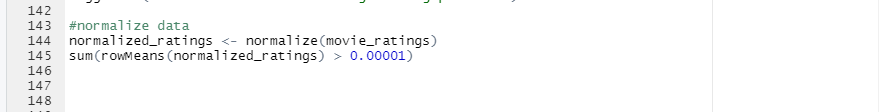


OUTPUT

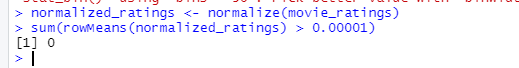
* DATA NORMALIZATION:

In the case of some users, there can be high ratings or low ratings provided to all of the watched films. This will act as a bias while implementing our model. In order to remove this, we normalize our data. Normalization is a data preparation procedure to standardize the numerical values in a column to a common scale value. This is done in such a way that there is no distortion in the range of values. Normalization transforms the average value of our ratings column to 0. We then plot a heatmap that delineates our normalized ratings.

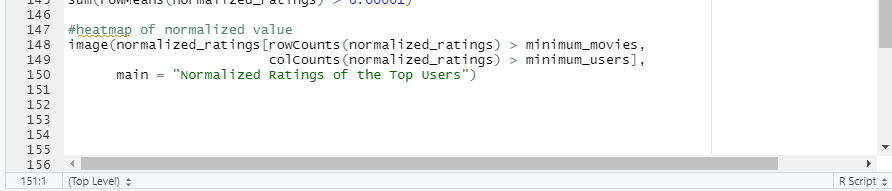
CODE



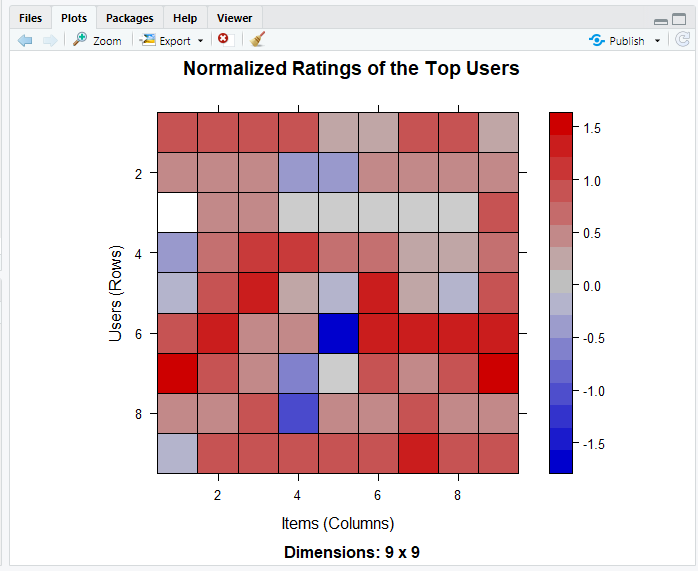
OUTPUT



CODE



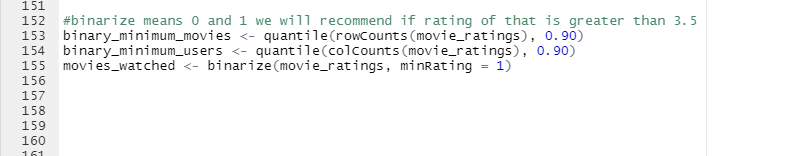
OUTPUT



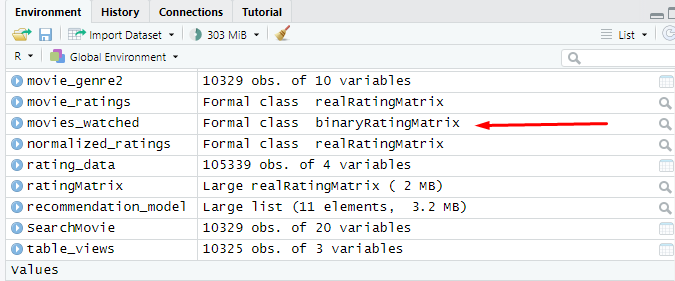
* PERFORMING DATA BINARIZATION:

In the final step of our data preparation in this data science project, we will binarize our data. Binarizing the data means that we have two discrete values 1 and 0, which will allow our recommendation systems to work more efficiently. We will define a matrix that will consist of 1 if the rating is above 3 and otherwise it will be 0.

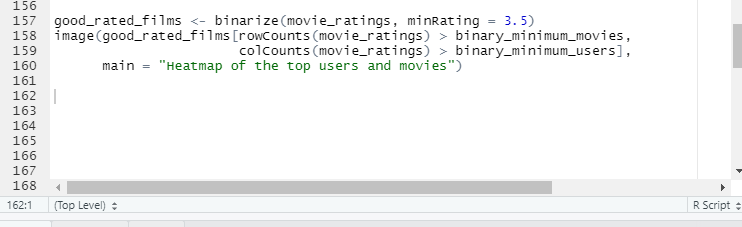
CODE

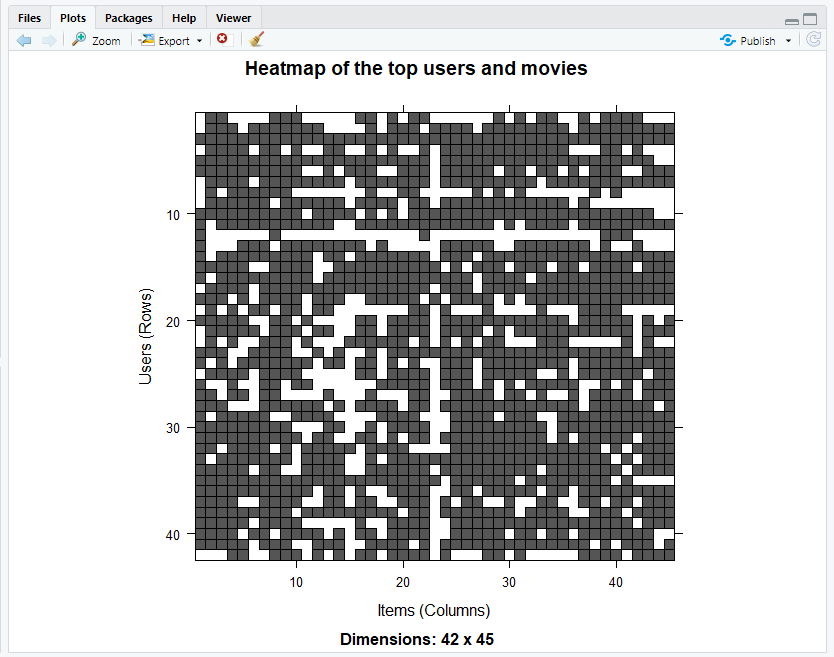


OUTPUT



CODE





OUTPUT

* COLLABRATIVE FILTERING SYSTEM:

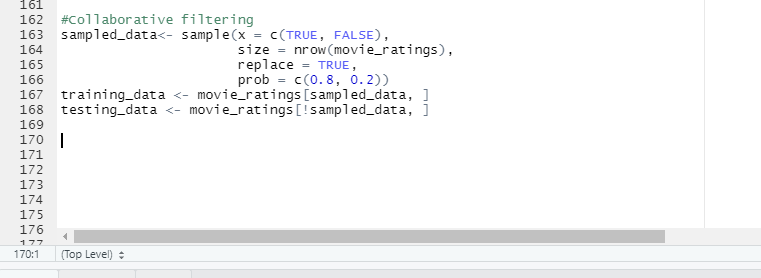
In this section of data science project, we will develop our very own Item Based Collaborative Filtering System. This type of collaborative filtering finds similarity in the items based on the people’s ratings of them. The algorithm first builds a similar-items table of the customers who have purchased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

CODE



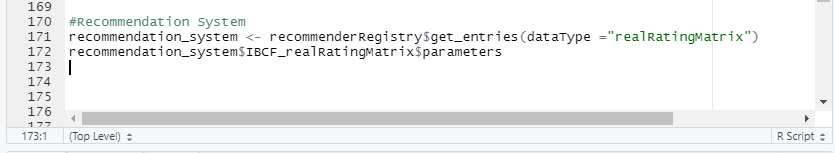
OUTPUT



* BUILDING THE RECOMMENDATION SYSTEM USING R:

We will now explore the various parameters of our Item Based Collaborative Filter. These parameters are default in nature. In the first step, k denotes the number of items for computing their similarities. Here, k is equal to 30. Therefore, the algorithm will now identify the k most similar items and store their number. We use the cosine method which is the default one but you can also use pearson method.

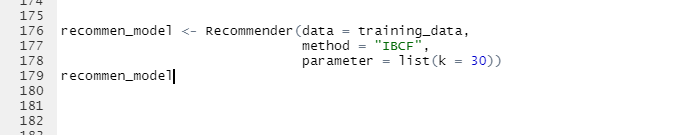
CODE



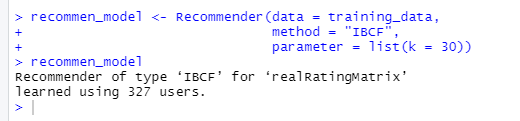


OUTPUT

CODE



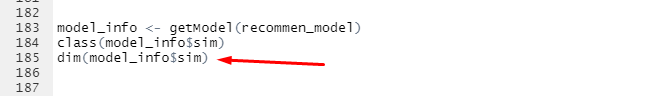
OUTPUT

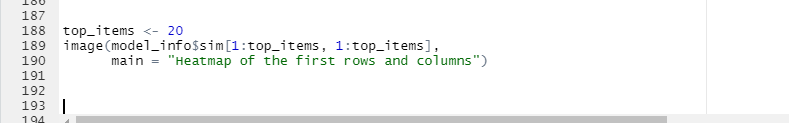


Let us now explore our data science recommendation system model as follows –

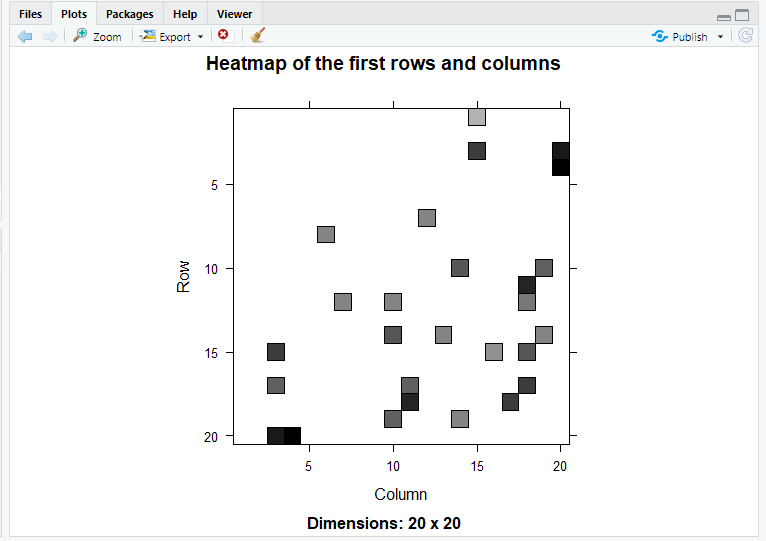
Using the getModel() function, we will retrieve the recommend\_model. We will then find the class and dimensions of our similarity matrix that is contained within model\_info. Finally, we will generate a heatmap, that will contain the top 20 items and visualize the similarity shared between them.

CODE



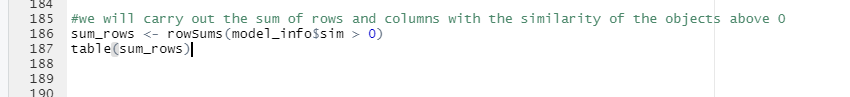
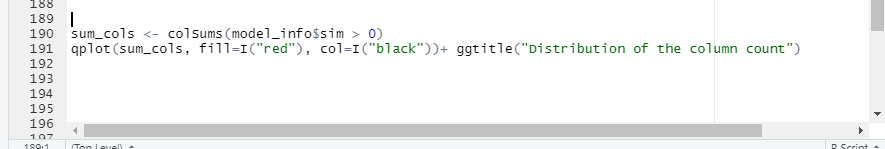


OUTPUT

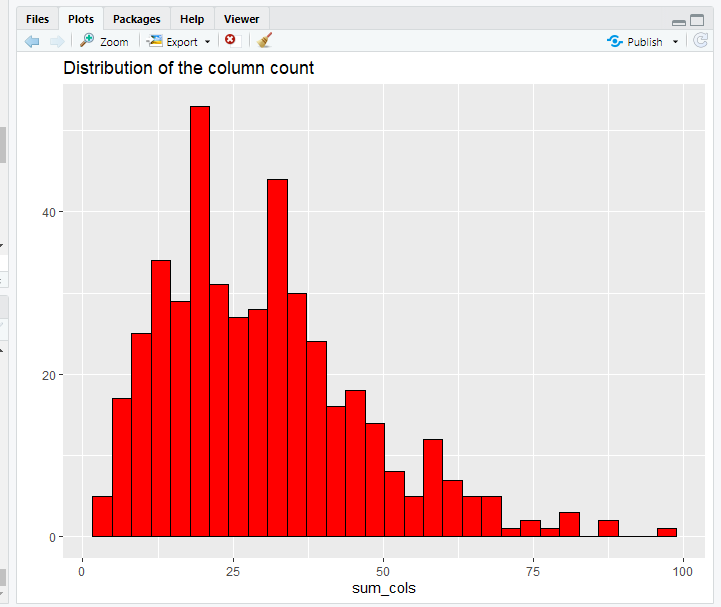


In the next step of ML project, we will carry out the sum of rows and columns with the similarity of the objects above 0. We will visualize the sum of columns through a distribution as follows –

CODE



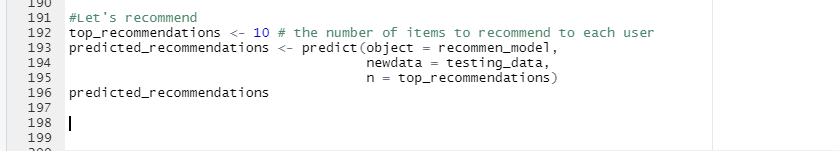
OUTPUT



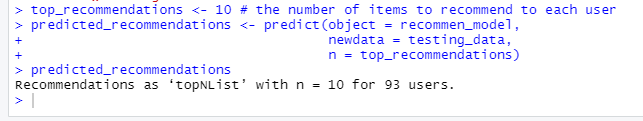
* HOW TO BUILD RECOMMENDER SYSTEM ON DATASET USING R?

We will create a top\_recommendations variable which will be initialized to 10, specifying the number of films to each user. We will then use the predict() function that will identify similar items and will rank them appropriately. Here, each rating is used as a weight. Each weight is multiplied with related similarities. Finally, everything is added in the end.

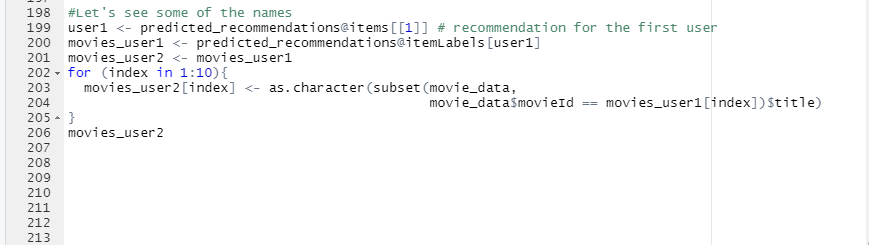
CODE



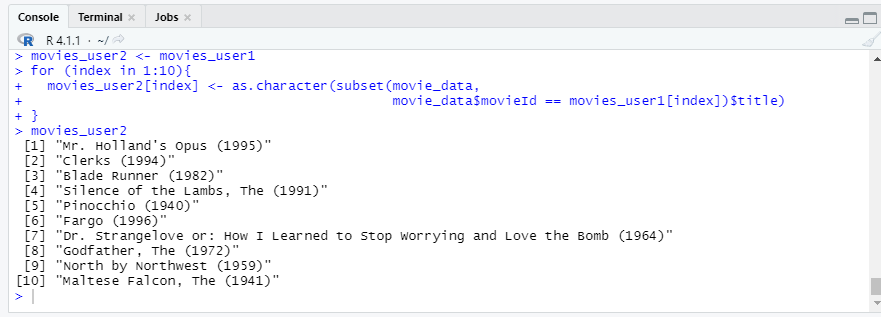
OUTPUT



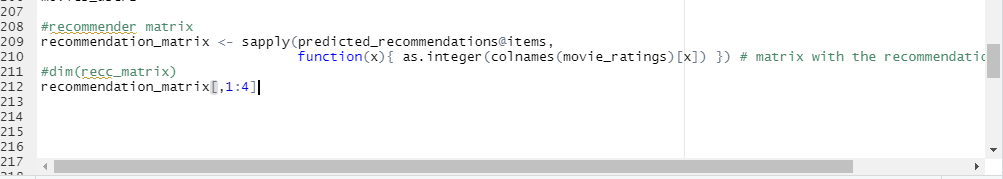
CODE



OUTPUT

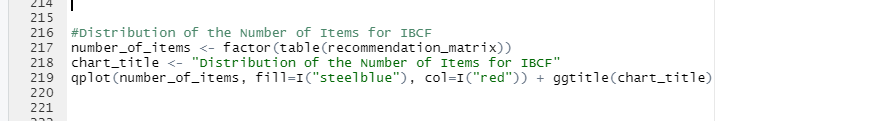


CODE



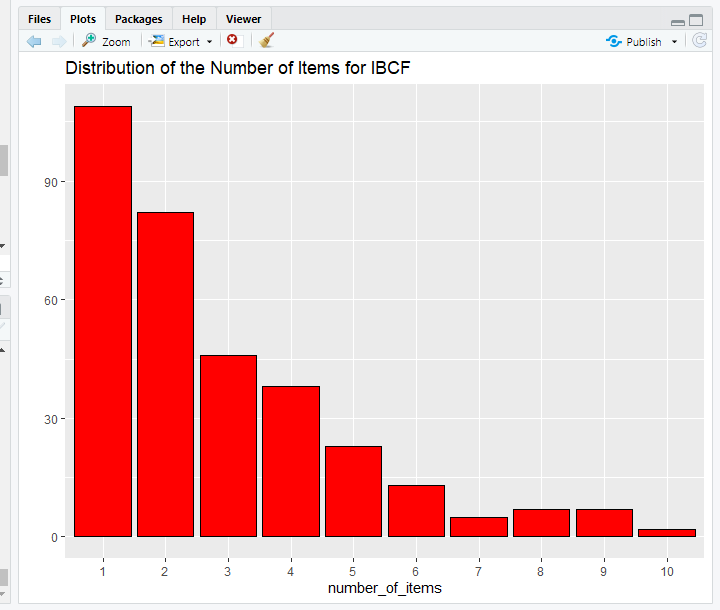
OUTPUT

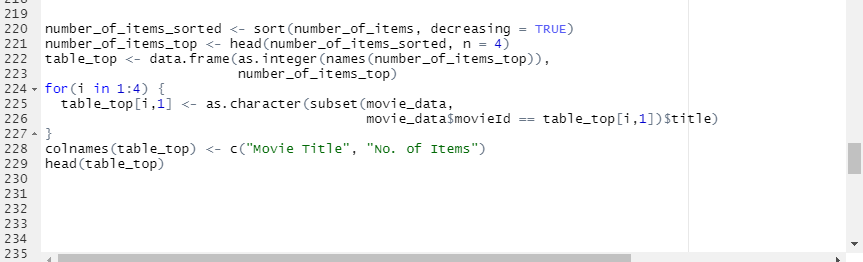




CODE

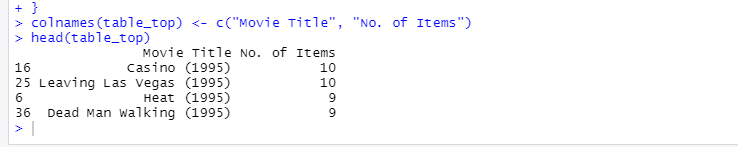
OUTPUT





CODE

OUTPUT



**RESULT AND CONCLUSION**

Recommendation Systems are the most popular type of machine learning applications that are used in all sectors. They are an improvement over the traditional classification algorithms as they can take many classes of input and provide similarity ranking based algorithms to provide the user with accurate results. These recommendation systems have evolved over time and have incorporated many advanced machine learning techniques to provide the users with the content that they want.

This recommendation system recommends different movies to users. Since this system is based on a collaborative approach, it will give progressively explicit outcomes contrasted with different systems that are based on the content-based approach. Content-based recommendation systems are constrained to people, these systems don't prescribe things out of the box. These systems work on individual users’ ratings, hence limiting your choice to explore more. While our system which is based on a collaborative approach computes the connection between different clients and relying upon their ratings, prescribes movies to others who have similar tastes, subsequently allowing users to explore more. It is a web application that allows users to rate movies as well as recommends them appropriate movies based on other's ratings.

Recommended systems open up new opportunities to obtain personal information over the Internet. It also helps reduce the problem of information overload which is a very common phenomenon with information retrieval systems and allows users to access products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user’s personal interests and based on his previous reviews, movies are recommended to users. This strategy helps in improving accuracy of the recommendations. It also helps in collecting authentic data with improved accuracy and makes the system more responsive. This approach overcomes the drawbacks of each individual algorithm and improves system performance. Techniques such as clustering, similarity, and classification are used to obtain better recommendations, thus reducing MAE and increasing accuracy and precision.

# References

Harpreet Kaur Virk, Er. Maninder Singh, April 2015. Analysis and Design of Hybrid Online Movie Recommender System. *International Journal of Innovations in Engineering and Technology (IJIET),* 5(2).

Urszula Kużelewska, 2014. Clustering Algorithms in Hybrid Recommender System on MovieLens Data. *Studies in Logic, Grammar and Rhetoric.*

Agarwal, S., 2013. *Data Mining: Data Mining Concepts and Techniques.* Katra, India, IEEE.

Costin-Gabriel Chiru, Vladimir-Nicolae Dinu , Ctlina Preda, Matei Macri, 2015. *Movie Recommender System Using the User's Psychological Profile.* s.l., IEEE International Conference on ICCP.

F. Furtado, A. S., 2020. Movie recommendation system using machine learning. *International journal of research in Industrial Engineering.,* 9(1), pp. 84-98.

Kharita, M. K., Kumar, A. & Singh, P., 2018. *Item-Based Collaborative Filtering in Movie Recommendation in Real time.* Jalandhar, India, IEEE.

Luis M. de Campos, Juan M. Fernández-Luna \*, Juan F. Huete, Miguel A. Rueda-Morales;, 2010. “Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks”. *International Journal of Approximate Reasoning.*

Manoj Kumar, D.K Yadav, Ankur Singh, Vijay Kr. Gupta, August 2015. A Movie Recommender System MOVREC. *International journal of computer applications (0975-8887),* 24(3).

Poonam Sharma, L. Y., July 2020. Movie Recommendation System Using Item Based Collaborative Filtering. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST),* 8(4).

S. Gnanapriya, R. S. ,. G. S. D. ,. M. S. K., 2010. Data Mining Concepts and Techniques. *Data Mining and Knowledge Engineering,,* Volume Vol 2, 9 (2010) , pp. 256-263.

S.D.Gheware, A. K. S. T., October 2014. Data Mining: Task, Tools, Techniques and Applications. *International Journal of Advanced Research in Computer and Communication Engineering,* 3(10), p. 8096.

Utkarsh Gupta1 and Dr Nagamma Patil2, 2015. *Recommender System Based on Hierarchical Clustering Algorithm Chameleon.* s.l., IEEE International Advance Computing Conference(IACC). .

Project Of Movie Recommendation System

NEHA\_ADIL

12/5/2021

library(recommenderlab) #for recommendation

## Warning: package 'recommenderlab' was built under R version 4.1.2

## Loading required package: Matrix

## Loading required package: arules

## Warning: package 'arules' was built under R version 4.1.2

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

## Loading required package: proxy

## Warning: package 'proxy' was built under R version 4.1.2

##   
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':  
##   
## as.matrix

## The following objects are masked from 'package:stats':  
##   
## as.dist, dist

## The following object is masked from 'package:base':  
##   
## as.matrix

## Loading required package: registry

## Registered S3 methods overwritten by 'registry':  
## method from   
## print.registry\_field proxy  
## print.registry\_entry proxy

library(reshape2)

## Warning: package 'reshape2' was built under R version 4.1.2

library(data.table)

## Warning: package 'data.table' was built under R version 4.1.2

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:reshape2':  
##   
## dcast, melt

library(ggplot2) #visualization

## Warning: package 'ggplot2' was built under R version 4.1.2

library(DT)

## Warning: package 'DT' was built under R version 4.1.2

#retrieving the data  
movie\_data<-read.csv("E:/6th semester/data mining/Project/movies.csv" , stringsAsFactors = FALSE)  
rating\_data<-read.csv("E:/6th semester/data mining/Project/ratings.csv")  
  
#structure  
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|Romance" ...

str(rating\_data)

## 'data.frame': 105339 obs. of 4 variables:  
## $ userId : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ movieId : int 16 24 32 47 50 110 150 161 165 204 ...  
## $ rating : num 4 1.5 4 4 4 4 3 4 3 0.5 ...  
## $ timestamp: int 1217897793 1217895807 1217896246 1217896556 1217896523 1217896150 1217895940 1217897864 1217897135 1217895786 ...

#tabular view #datatable(movie\_data) #id, title, genres #datatable(rating\_data)

```r  
#summary statistics  
summary(movie\_data)

## movieId title genres   
## Min. : 1 Length:10329 Length:10329   
## 1st Qu.: 3240 Class :character Class :character   
## Median : 7088 Mode :character Mode :character   
## Mean : 31924   
## 3rd Qu.: 59900   
## Max. :149532

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

#DATA PREPROCESSING  
#we need to do something with genre more useful  
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)  
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])  
 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)

## 'data.frame': 10329 obs. of 18 variables:  
## $ Action : int 0 0 0 0 0 1 0 0 1 1 ...  
## $ Adventure : int 1 1 0 0 0 0 0 1 0 1 ...  
## $ Animation : int 1 0 0 0 0 0 0 0 0 0 ...  
## $ Children : int 1 1 0 0 0 0 0 1 0 0 ...  
## $ Comedy : int 1 0 1 1 1 0 1 0 0 0 ...  
## $ Crime : int 0 0 0 0 0 1 0 0 0 0 ...  
## $ Documentary: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Drama : int 0 0 0 1 0 0 0 0 0 0 ...  
## $ Fantasy : int 1 1 0 0 0 0 0 0 0 0 ...  
## $ Film-Noir : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Horror : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Musical : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Mystery : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Romance : int 0 0 1 1 0 0 1 0 0 0 ...  
## $ Sci-Fi : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Thriller : int 0 0 0 0 0 1 0 0 0 1 ...  
## $ War : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Western : int 0 0 0 0 0 0 0 0 0 0 ...

#head of data  
head(movie\_data)

## movieId title  
## 1 1 Toy Story (1995)  
## 2 2 Jumanji (1995)  
## 3 3 Grumpier Old Men (1995)  
## 4 4 Waiting to Exhale (1995)  
## 5 5 Father of the Bride Part II (1995)  
## 6 6 Heat (1995)  
## genres  
## 1 Adventure|Animation|Children|Comedy|Fantasy  
## 2 Adventure|Children|Fantasy  
## 3 Comedy|Romance  
## 4 Comedy|Drama|Romance  
## 5 Comedy  
## 6 Action|Crime|Thriller

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

#create a search matrix that gives us films based on genres  
SearchMovie <- cbind(movie\_data[,1:2],genre\_mat2[])  
head(SearchMovie)

## movieId title Action Adventure Animation  
## 1 1 Toy Story (1995) 0 1 1  
## 2 2 Jumanji (1995) 0 1 0  
## 3 3 Grumpier Old Men (1995) 0 0 0  
## 4 4 Waiting to Exhale (1995) 0 0 0  
## 5 5 Father of the Bride Part II (1995) 0 0 0  
## 6 6 Heat (1995) 1 0 0  
## Children Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical  
## 1 1 1 0 0 0 1 0 0 0  
## 2 1 0 0 0 0 1 0 0 0  
## 3 0 1 0 0 0 0 0 0 0  
## 4 0 1 0 0 1 0 0 0 0  
## 5 0 1 0 0 0 0 0 0 0  
## 6 0 0 1 0 0 0 0 0 0  
## Mystery Romance Sci-Fi Thriller War Western  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 1 0 0 0 0  
## 4 0 1 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 1 0 0

#many movies have several genre  
#Let's create sparse matrix for recommendation  
ratingMatrix <- dcast(rating\_data, userId~movieId, value.var = "rating", na.rm=FALSE)

## Warning in dcast(rating\_data, userId ~ movieId, value.var = "rating", na.rm  
## = FALSE): The dcast generic in data.table has been passed a data.frame and  
## will attempt to redirect to the reshape2::dcast; please note that reshape2  
## is deprecated, and this redirection is now deprecated as well. Please do this  
## redirection yourself like reshape2::dcast(rating\_data). In the next version,  
## this warning will become an error.

ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove user Ids

#Convert rating matrix into a recommender lab sparse matrix  
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")  
ratingMatrix

## 668 x 10325 rating matrix of class 'realRatingMatrix' with 105339 ratings.

#recommendation model  
library(recommenderlab)  
recommendation\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")  
names(recommendation\_model)

## [1] "HYBRID\_realRatingMatrix" "ALS\_realRatingMatrix"   
## [3] "ALS\_implicit\_realRatingMatrix" "IBCF\_realRatingMatrix"   
## [5] "LIBMF\_realRatingMatrix" "POPULAR\_realRatingMatrix"   
## [7] "RANDOM\_realRatingMatrix" "RERECOMMEND\_realRatingMatrix"   
## [9] "SVD\_realRatingMatrix" "SVDF\_realRatingMatrix"   
## [11] "UBCF\_realRatingMatrix"

lapply(recommendation\_model, "[[", "description")

## $HYBRID\_realRatingMatrix  
## [1] "Hybrid recommender that aggegates several recommendation strategies using weighted averages."  
##   
## $ALS\_realRatingMatrix  
## [1] "Recommender for explicit ratings based on latent factors, calculated by alternating least squares algorithm."  
##   
## $ALS\_implicit\_realRatingMatrix  
## [1] "Recommender for implicit data based on latent factors, calculated by alternating least squares algorithm."  
##   
## $IBCF\_realRatingMatrix  
## [1] "Recommender based on item-based collaborative filtering."  
##   
## $LIBMF\_realRatingMatrix  
## [1] "Matrix factorization with LIBMF via package recosystem (https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html)."  
##   
## $POPULAR\_realRatingMatrix  
## [1] "Recommender based on item popularity."  
##   
## $RANDOM\_realRatingMatrix  
## [1] "Produce random recommendations (real ratings)."  
##   
## $RERECOMMEND\_realRatingMatrix  
## [1] "Re-recommends highly rated items (real ratings)."  
##   
## $SVD\_realRatingMatrix  
## [1] "Recommender based on SVD approximation with column-mean imputation."  
##   
## $SVDF\_realRatingMatrix  
## [1] "Recommender based on Funk SVD with gradient descend (https://sifter.org/~simon/journal/20061211.html)."  
##   
## $UBCF\_realRatingMatrix  
## [1] "Recommender based on user-based collaborative filtering."

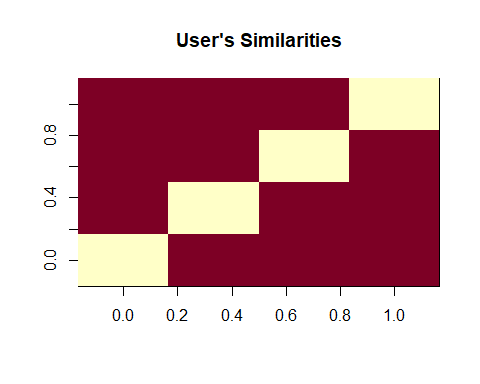
#we will use item based collaborative filtering  
recommendation\_model$IBCF\_realRatingMatrix$parameters

## $k  
## [1] 30  
##   
## $method  
## [1] "Cosine"  
##   
## $normalize  
## [1] "center"  
##   
## $normalize\_sim\_matrix  
## [1] FALSE  
##   
## $alpha  
## [1] 0.5  
##   
## $na\_as\_zero  
## [1] FALSE

#Let's check similarity of users we take only 4 users  
similarity\_mat <- similarity(ratingMatrix[1:4, ],  
 method = "cosine",  
 which = "users")  
as.matrix(similarity\_mat)

## 1 2 3 4  
## 1 0.0000000 0.9760860 0.9641723 0.9914398  
## 2 0.9760860 0.0000000 0.9925732 0.9374253  
## 3 0.9641723 0.9925732 0.0000000 0.9888968  
## 4 0.9914398 0.9374253 0.9888968 0.0000000

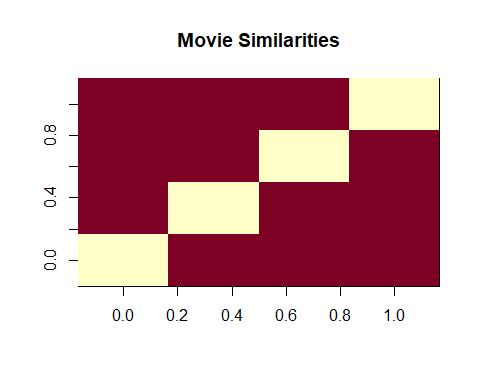
image(as.matrix(similarity\_mat), main = "User's Similarities")



#Let's check similarity of movies   
movie\_similarity <- similarity(ratingMatrix[ ,1:4],  
 method = "cosine",  
 which = "items")  
as.matrix(movie\_similarity)

## 1 2 3 4  
## 1 0.0000000 0.9669732 0.9559341 0.9101276  
## 2 0.9669732 0.0000000 0.9658757 0.9412416  
## 3 0.9559341 0.9658757 0.0000000 0.9864877  
## 4 0.9101276 0.9412416 0.9864877 0.0000000

image(as.matrix(movie\_similarity), main = "Movie Similarities")



#Most unique rating values  
rating\_values <- as.vector(ratingMatrix@data)  
unique(rating\_values) # extracting unique ratings

## [1] 0.0 5.0 4.0 3.0 4.5 1.5 2.0 3.5 1.0 2.5 0.5

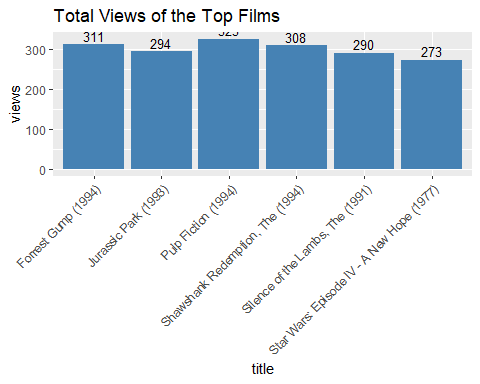
#how much rating as count of numbers  
Table\_of\_Ratings <- table(rating\_values) # creating a count of movie ratings  
Table\_of\_Ratings

## rating\_values  
## 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5   
## 6791761 1198 3258 1567 7943 5484 21729 12237 28880 8187   
## 5   
## 14856

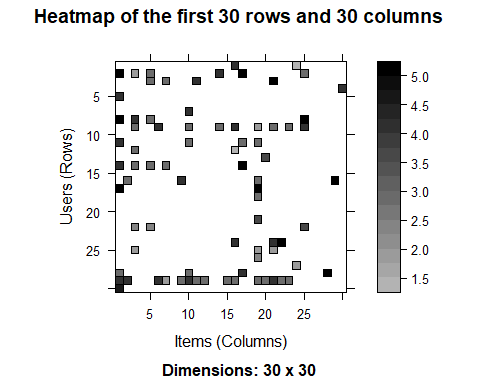
#Most Viewed Movies Visualization  
library(ggplot2)  
movie\_views <- colCounts(ratingMatrix) # count views for each movie  
table\_views <- data.frame(movie = names(movie\_views),  
 views = movie\_views) # create dataframe of views  
table\_views <- table\_views[order(table\_views$views,  
 decreasing = TRUE), ] # sort by number of views  
table\_views$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)

#plotting this data  
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")



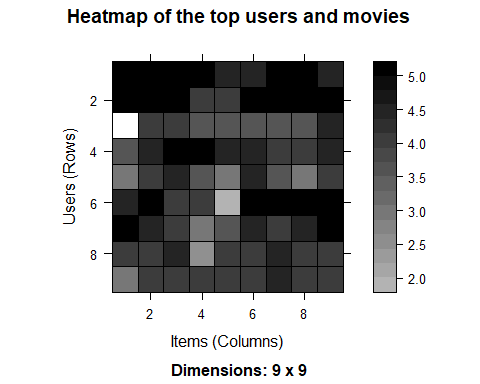
#Heat map of rating matrix  
image(ratingMatrix[1:30, 1:30], axes = FALSE, main = "Heatmap of the first 30 rows and 30 columns")



#Lot's of sparse data now we will  
#1.Select useful data  
#2.normalize it  
#3.binarize it  
#you have seen the rating dataset so what do you think how many users need to rate a movie to be  
#useful Let's say 50  
movie\_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,colCounts(ratingMatrix) > 50]  
movie\_ratings

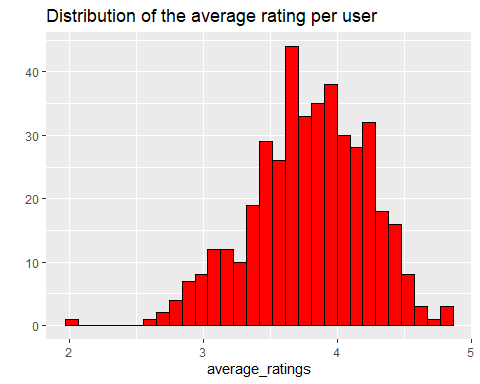
## 420 x 447 rating matrix of class 'realRatingMatrix' with 38341 ratings.

#this bunch of code finds a heat map of top users and movies  
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")



#Now, we will visualize the distribution of the average ratings per user.  
average\_ratings <- rowMeans(movie\_ratings)  
qplot(average\_ratings, fill=I("red"), col=I("black")) +  
 ggtitle("Distribution of the average rating per user")

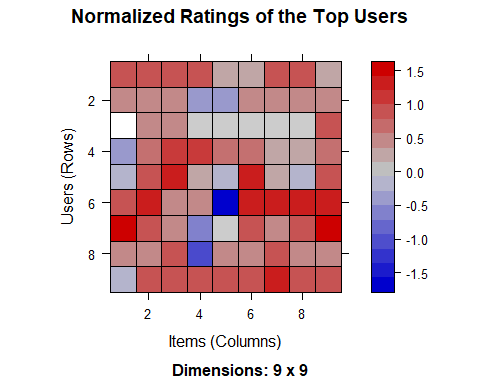
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



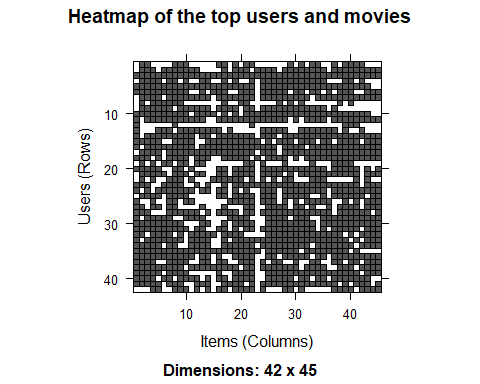
#normalize data  
normalized\_ratings <- normalize(movie\_ratings)  
sum(rowMeans(normalized\_ratings) > 0.00001)

## [1] 0

#heatmap of normalized value  
image(normalized\_ratings[rowCounts(normalized\_ratings) > minimum\_movies,  
 colCounts(normalized\_ratings) > minimum\_users],  
 main = "Normalized Ratings of the Top Users")



#binarize means 0 and 1 we will recommend if rating of that is greater than 3.5  
binary\_minimum\_movies <- quantile(rowCounts(movie\_ratings), 0.90)  
binary\_minimum\_users <- quantile(colCounts(movie\_ratings), 0.90)  
movies\_watched <- binarize(movie\_ratings, minRating = 1)  
  
good\_rated\_films <- binarize(movie\_ratings, minRating = 3.5)  
image(good\_rated\_films[rowCounts(movie\_ratings) > binary\_minimum\_movies,  
 colCounts(movie\_ratings) > binary\_minimum\_users],  
 main = "Heatmap of the top users and movies")



#Collaborative filtering  
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  
recommendation\_system <- recommenderRegistry$get\_entries(dataType ="realRatingMatrix")  
recommendation\_system$IBCF\_realRatingMatrix$parameters

## $k  
## [1] 30  
##   
## $method  
## [1] "Cosine"  
##   
## $normalize  
## [1] "center"  
##   
## $normalize\_sim\_matrix  
## [1] FALSE  
##   
## $alpha  
## [1] 0.5  
##   
## $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 335 users.

class(recommen\_model)

## [1] "Recommender"  
## attr(,"package")  
## [1] "recommenderlab"

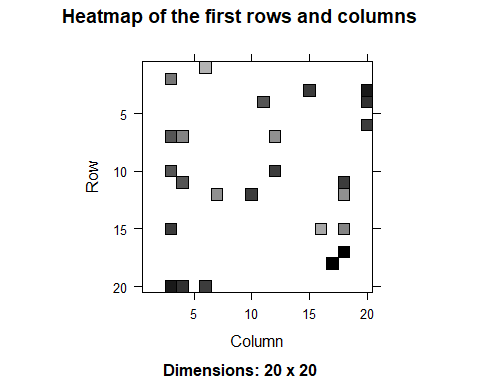
model\_info <- getModel(recommen\_model)  
class(model\_info$sim)

## [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")

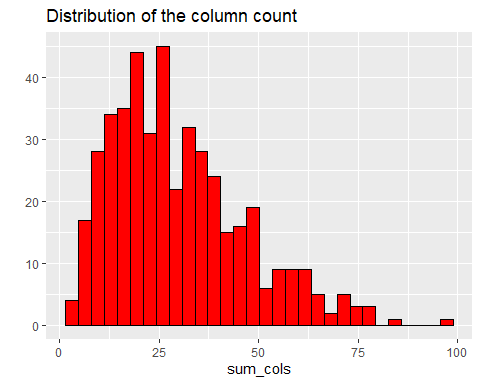


#we will carry out the sum of rows and columns with the similarity of the objects above 0  
sum\_rows <- rowSums(model\_info$sim > 0)  
table(sum\_rows)

## sum\_rows  
## 30   
## 447

sum\_cols <- colSums(model\_info$sim > 0)  
qplot(sum\_cols, fill=I("red"), col=I("black"))+ ggtitle("Distribution of the column count")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



#Let's recommend  
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 85 users.

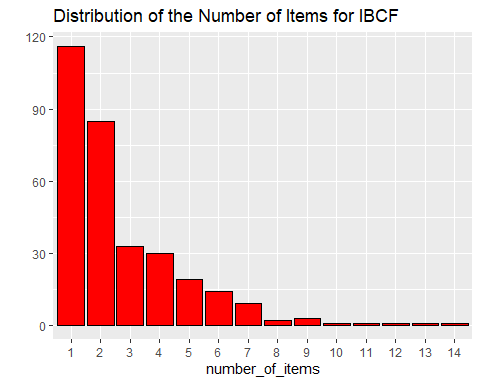
#Let's see some of the names  
user1 <- predicted\_recommendations@items[[1]] # recommendation for the first user  
movies\_user1 <- predicted\_recommendations@itemLabels[user1]  
movies\_user2 <- movies\_user1  
for (index in 1:10){  
 movies\_user2[index] <- as.character(subset(movie\_data,  
 movie\_data$movieId == movies\_user1[index])$title)  
}  
movies\_user2

## [1] "GoldenEye (1995)" "Get Shorty (1995)"   
## [3] "Leaving Las Vegas (1995)" "From Dusk Till Dawn (1996)"   
## [5] "French Kiss (1995)" "Crow, The (1994)"   
## [7] "Four Weddings and a Funeral (1994)" "Aladdin (1992)"   
## [9] "James and the Giant Peach (1996)" "Maltese Falcon, The (1941)"

#recommender matrix  
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]

## [,1] [,2] [,3] [,4]  
## [1,] 10 161 551 555  
## [2,] 21 592 555 1748  
## [3,] 25 1729 508 2324  
## [4,] 70 3578 165 3081  
## [5,] 236 2028 48394 3448  
## [6,] 353 593 4011 3897  
## [7,] 357 913 529 4306  
## [8,] 588 1704 1674 1704  
## [9,] 661 457 4022 1641  
## [10,] 913 1278 3578 3471

#Distribution of the Number of Items for IBCF  
number\_of\_items <- factor(table(recommendation\_matrix))  
chart\_title <- "Distribution of the Number of Items for IBCF"  
qplot(number\_of\_items, fill=I("red"), col=I("black")) + ggtitle(chart\_title)



number\_of\_items\_sorted <- sort(number\_of\_items, decreasing = TRUE)  
number\_of\_items\_top <- head(number\_of\_items\_sorted, n = 4)  
table\_top <- data.frame(as.integer(names(number\_of\_items\_top)),  
 number\_of\_items\_top)  
for(i in 1:4) {  
 table\_top[i,1] <- as.character(subset(movie\_data,  
 movie\_data$movieId == table\_top[i,1])$title)  
}  
colnames(table\_top) <- c("Movie Title", "No. of Items")  
head(table\_top)

## Movie Title No. of Items  
## 6 Heat (1995) 14  
## 923 Citizen Kane (1941) 13  
## 593 Silence of the Lambs, The (1991) 12  
## 508 Philadelphia (1993) 11