Contents

[2 Objectives: 4](#_Toc106140482)

[3 What is Recommendation System? 4](#_Toc106140483)

[4 Dataset used: 4](#_Toc106140484)

[5 Essential Libraries: 5](#_Toc106140485)

[*5.1* *library(recommenderlab)* 5](#_Toc106140486)

[5.2 library(ggplot2) 5](#_Toc106140487)

[5.3 library(data.table) 5](#_Toc106140488)

[5.4 library(reshape2) 5](#_Toc106140489)

[6 Importing data: 5](#_Toc106140490)

[7 Data Pre-processing: 7](#_Toc106140491)

[7.1 One-Hot Encoding: 7](#_Toc106140492)

[7.2 Dcast: 10](#_Toc106140493)

[7.3 lapply(): 12](#_Toc106140494)

[8 Exploring Similar Data 13](#_Toc106140495)

[9 Most Viewed Movies Visualization 16](#_Toc106140496)

[10 Heatmap of Movie Ratings 18](#_Toc106140497)

[11 Performing Data Preparation 19](#_Toc106140498)

[11.1 Selecting Useful Data 19](#_Toc106140499)

[11.2 Data Normalization 22](#_Toc106140500)

[11.3 Performing Data Binarization 23](#_Toc106140501)

[12 Collaborative Filtering System 25](#_Toc106140502)

[13 Building the Recommendation System using R 25](#_Toc106140503)

[14 How to build Recommender System on dataset using R? 29](#_Toc106140504)

[15 Summary 30](#_Toc106140505)

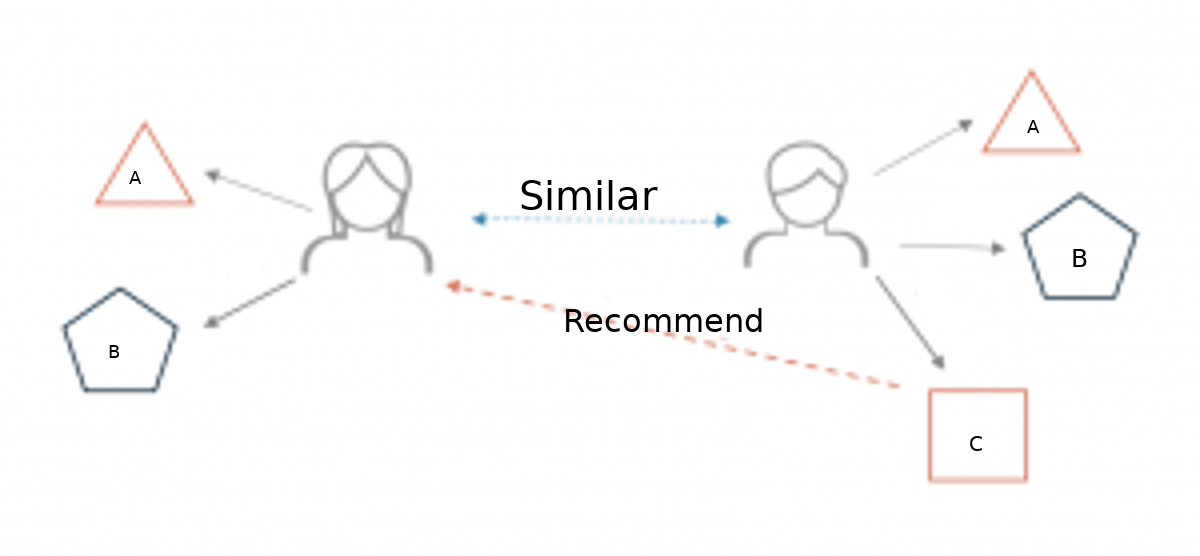
[16 Reference: 31](#_Toc106140506)

# Objectives:

The main goal of this machine learning project is to build a recommendation engine that recommends movies to users. This R project is designed to understand the functioning of a recommendation system. Here we developed an Item Based Collaborative Filter. This will help gain experience of implementing my R, Data Science, and Machine learning skills in a real-life project.

# What is Recommendation System?

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like.



# Dataset used:

To build our recommendation system, we have used the MovieLens Dataset. You can find the movies.csv and ratings.csv file that we have used in our Recommendation System Project. This data consists of 105339 ratings applied to over 10329 movies.













 















 



# Essential Libraries:

## *library(recommenderlab)*

This R package provides an infrastructure to test and develop recommender algorithms. The package supports rating(e.g., 1-5) and unary(0-1) data-sets.

## library(ggplot2)

ggplot2 is an R package dedicated to data visualization. It can greatly improve the quality and aesthetics of your graphics and will make you much more efficient in creating them.

## library(data.table)

data.table provides a high performance version of base R’s data.frame with syntax and feature enhancement for ease of use, convenience and programming speed.

.

## library(reshape2)

reshape2 is an R package written by Hadley Wickham that makes it easy to transform data between wide and long formats.

# Importing data:

In this step we will import our data from csv files

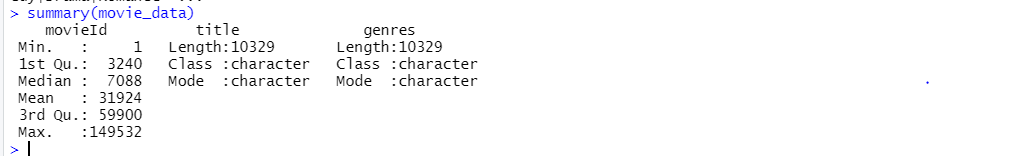
* + *setwd("/home/dataflair/data/movie\_data")*
  + *movie\_data <- read.csv("movies.csv",stringsAsFactors=FALSE)*
  + *rating\_data <- read.csv("ratings.csv")*

We can display the data frame by using str(movie\_data) and overview the summary of the movies using the summary() function. We will also use the head() function to print the first six lines of movie\_data

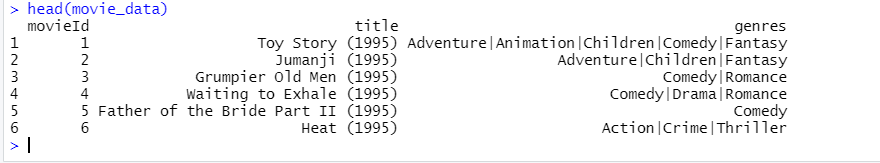
* str(movie\_data)



* summary(movie-data)



* head(movie\_data)



# Data Pre-processing:

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.

## One-Hot Encoding:

One hot encoding can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of a model.

* *movie\_genre <- as.data.frame(movie\_data$genres, stringsAsFactors=FALSE)*
* *movie\_genre2 <- as.data.frame(tstrsplit(movie\_genre[,1], '[|]',*

*type.convert=TRUE),*

*stringsAsFactors=FALSE)*

as. data. frame() function in R Programming Language is used to convert an object to data frame. These objects can be Vectors, Lists, Matrices, and Factors.

* *colnames(movie\_genre2) <- c(1:10)*

In this piece of code, we are creating a column of genre of all the listed movies in movie\_genre and in movie\_genre2 we are splitting genre into 10 columns, each genre in a column and then numbering the columns.

* *list\_genre <- c("Action", "Adventure", "Animation", "Children",*

*"Comedy", "Crime","Documentary", "Drama", "Fantasy",*

*"Film-Noir", "Horror", "Musical", "Mystery","Romance",*

*"Sci-Fi", "Thriller", "War", "Western")*

In this code we are making a list of all the possible genres of movies and stored in a variable list\_genre.

* *genre\_mat1 <- matrix(0,10330,18)*

Here we are making a matrix of size 18 and passing it 0 value.

* *genre\_mat1[1,] <- list\_genre*

Passing names of genre present in list to the first row of the matrix.

* *colnames(genre\_mat1) <- list\_genre*

Naming each column by the names of genre present in the list.

* *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*

*}*

*}*

In this for loop we will iterate through all the positions in the matrix and if the column name i.e., genre name matches the genre of the matrix we will mark it 1 otherwise it will remain 0. This is done to show the genre of the movie.

* *genre\_*mat2 *<- as.data.frame(genre\_mat1[-1,], stringsAsFactors=FALSE)*

This line of code will remove the first row, which was same as column name.

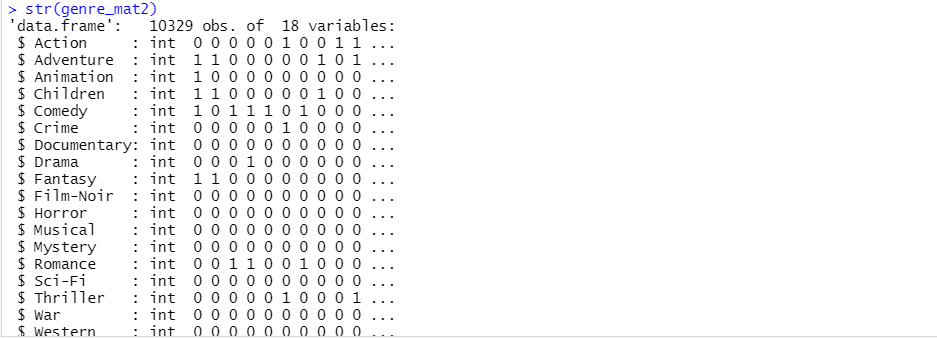
* *for (col in 1: ncol(genre\_mat2)) {*

*genre\_mat2[,col] <- as.integer(genre\_mat2[,col])*

*}*

Here we converted the character type data to integers.

* *str(genre\_mat2)*

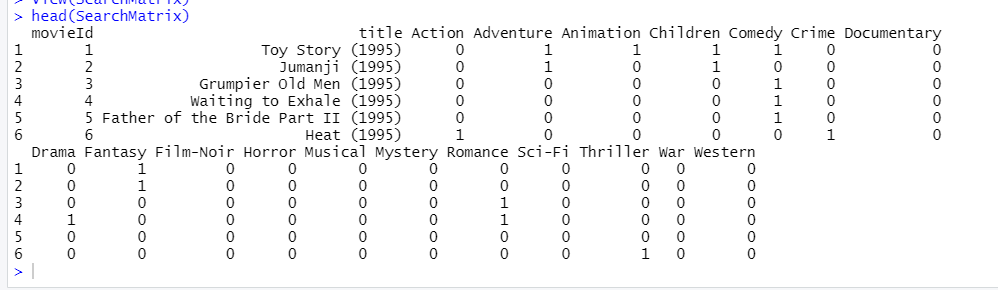
Here we will show the structure or dataframe of genre\_mat2 as follow:

* *SearchMatrix <- cbind(movie\_data[,1:2], genre\_mat2[])*

Here we are adding the first two columns of movies\_data at start of genre\_mat2. In this step of Data Pre-processing, 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.

* *head(SearchMatrix)*

To show the head of the data.



* *ratingMatrix <- dcast(rating\_data, userId~movieId, value.var = "rating", na.rm=FALSE)*

## Dcast:

If we have a dataset of long format, we are ready to reshape it with R. The melt and dcast functions for data.tables are for reshaping wide-to-long and long-to wide. We can use it to cast a molten data frame.

* *ratingMatrix <- as.matrix(ratingMatrix[,-1])*

As we have created a row of genre name, and it is same as column name so we will remove first row so we will left only with the column name.

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

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

* ratingMatrix <- as(ratingMatrix, "realRatingMatrix")

In this we are creating a matrix of rating, in corresponding matrix against a movie we placed rating of different userID.

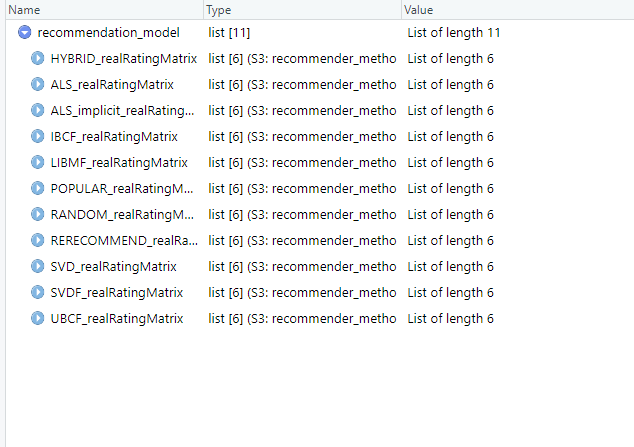
* ratingMatrix



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

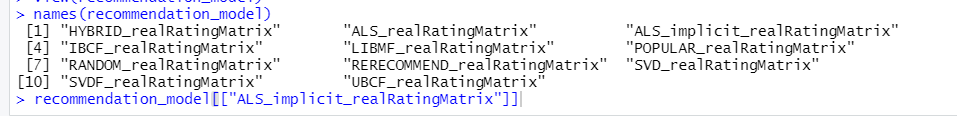
* *recommendation\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")*

We have a data frame of recommender registry; we will extract get\_entries column from it and assign it to recommendation\_model.



* *names(recommendation\_model)*

names() function is used to get the names of the objects.



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

## lapply():

The [lapply() function](https://www.geeksforgeeks.org/apply-a-function-over-a-list-of-elements-in-r-programming-lapply-function/) helps us in applying functions on list objects and returns a list object of the same length. The lapply() function in the R Language takes a list, vector, or data frame as input and gives output in the form of a list object. Since the lapply() function applies a certain operation to all the elements of the list it doesn’t need a Margin.

*recommendation\_model$IBCF\_realRatingMatrix$parameters*



We will implement a single model that is *Item-Based Collaborative Filtering.* Here we will display the parameters of the real rating matrix of the model.

# 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.

* *similarity\_mat <- similarity(ratingMatrix[1:4, ],*

*method = "cosine",*

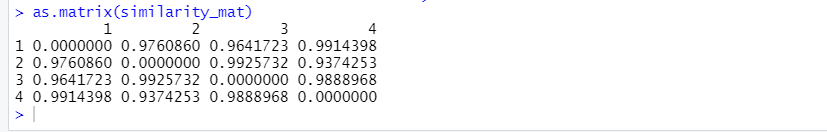
*which = "users")*

To find the similarity between the users we used the cosine method.

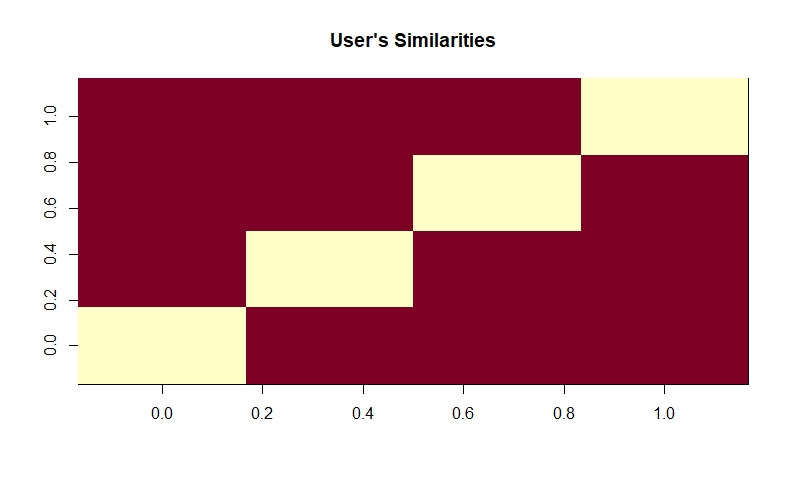
Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

* *as.matrix(similarity\_mat)*

We converted data table to matrix using as.matrix() function.



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



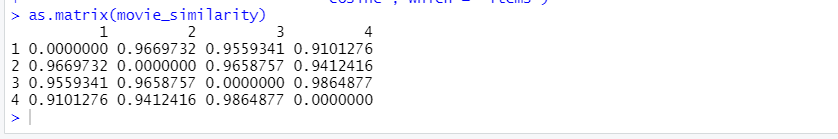
* *movie\_similarity <- similarity(ratingMatrix[, 1:4], method =*

*"cosine", which = "items")*

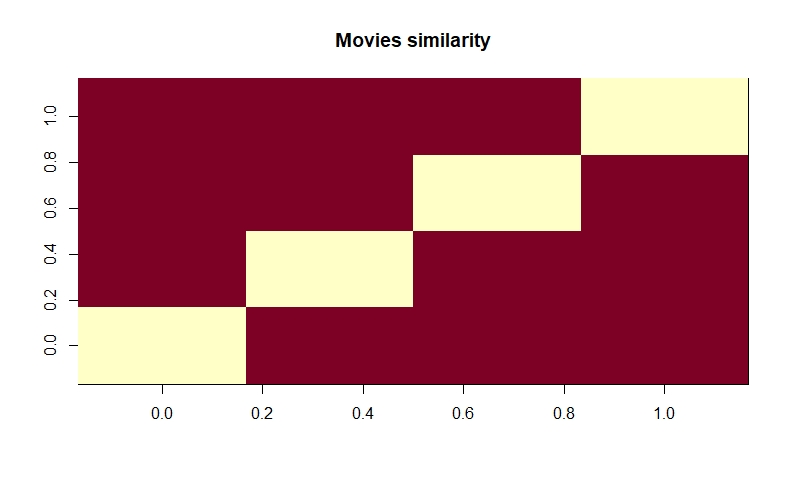
* *as.matrix(movie\_similarity)*

Similarly, we find movie similarity as well.

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



* *image(as.matrix(movie\_similarity), main = "Movies similarity")*



* *rating\_values <- as.vector(ratingMatrix@data)*

*Created a vector of rating values*

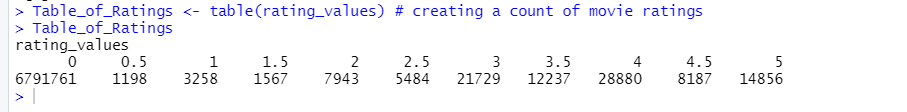
* *unique(rating\_values)*

*Extracted unique values of the ratings.*

**

* *Table\_of\_Ratings <- table(rating\_values*
* *Table\_of\_Ratings*

Calculated total values of unique rating values.



# Most Viewed Movies 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.

* *movie\_views <- colCounts(ratingMatrix*)

counting views for each movie

* *table\_views <- data.frame(movie = names(movie\_views),*

*views = movie\_views)*

create table of views

* *table\_views <- table\_views[order(table\_views$views,*

*decreasing = TRUE), ]*

sort by number of views in descending order

* *table\_views$title <- NA*

*created a column of title in table\_view and pass it NA*

* *for (index in 1:10325){*

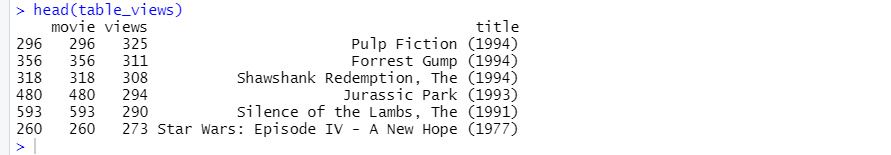
*table\_views[index,3] <- as.character(subset(movie\_data,*

*movie\_data$movieId == table\_views[index,1])$title)*

*}*

And here we passed the corresponding names of the movies to the title column.

* *table\_views[1:6,]*
* *head(table\_views)*



Visualize first 6 entries.

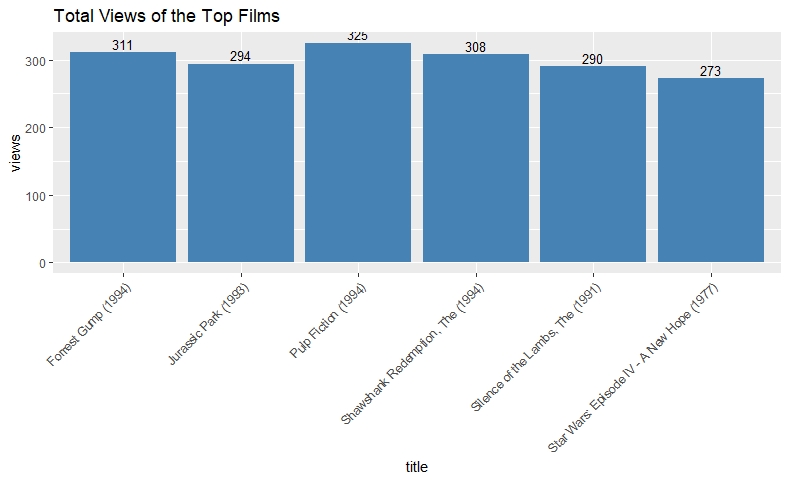
*ggplot(table\_views[1:6, ], aes(x = title, y = views)) +*

*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")*

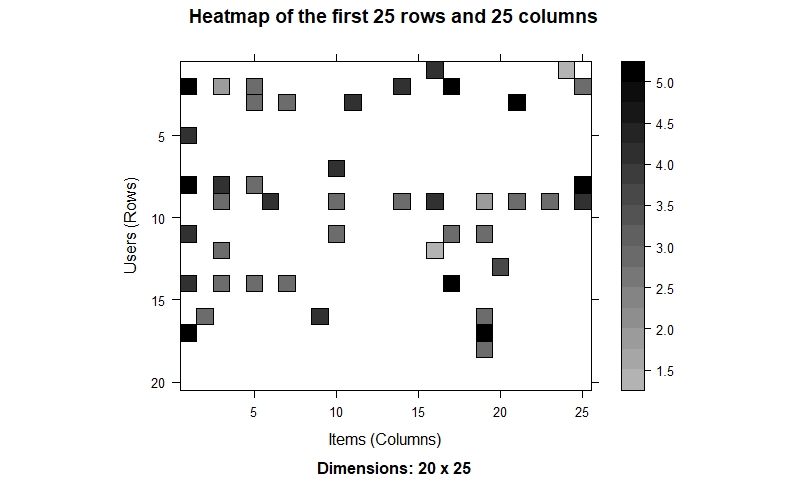
Now, we will visualize a bar plot for the total number of views of the top films. We will carry this out using ggplot.

# Heatmap of Movie Ratings

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

* *image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")*

x-axis shows users and y-axis shows items.



# Performing Data Preparation

We will conduct data preparation in the following three steps –

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

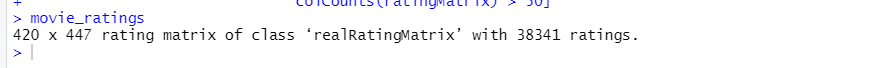
## Selecting Useful Data

* *movie\_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,*

*colCounts(ratingMatrix) > 50]*

* *movie\_ratings*

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 the same for minimum number of views that are per film. This way, we have filtered a list of watched films from least-watched ones.

**

From the above output of ‘movie\_ratings’, we observe that there are 420 users and 447 films as opposed to the previous 668 users and 10325 films.

* *minimum\_movies<- quantile(rowCounts(movie\_ratings), 0.98)*
* *minimum\_users <- quantile(colCounts(movie\_ratings), 0.98)*

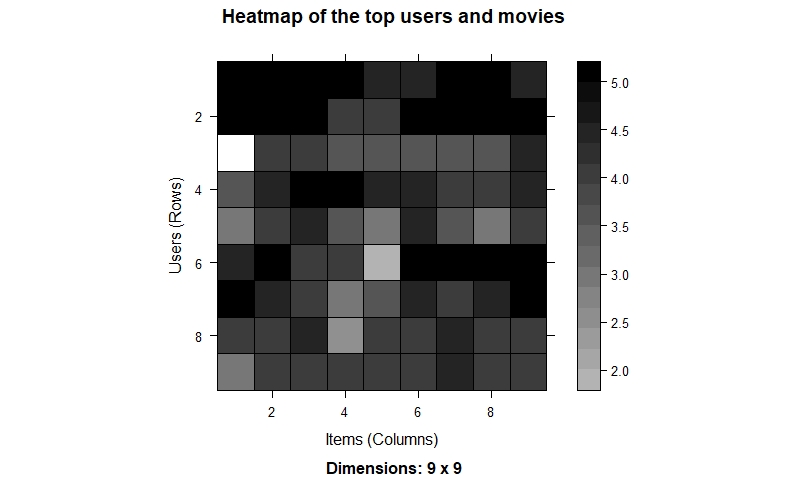
The generic function quantile produces sample quantiles corresponding to the given probabilities. The smallest observation corresponds to the probability of 0 and the largest to the probability of 1.

* *image(movie\_ratings[rowCounts(movie\_ratings) > minimum\_movies,*

*colCounts(movie\_ratings) > minimum\_users],*

*main = "Heatmap of the top users and movies")*

*We will create a Heatmap of top users and movies*

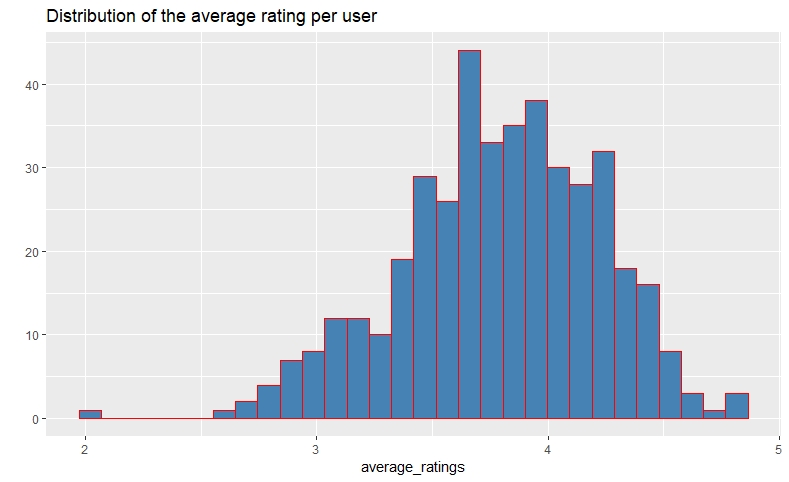


* *average\_ratings <- rowMeans(movie\_ratings)*

*Calculated average rating per user by finding means of each row.*

* *qplot(average\_ratings, fill=I("steelblue"), col=I("red")) +*

*ggtitle("Distribution of the average rating per user")*

**

## Data Normalization

In the case of some users, there can be high ratings or low ratings provided to all the watched films. This will act as a bias while implementing our model. 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.

* *normalized\_ratings <- normalize(movie\_ratings)*

Normalize numeric data to a given scale. Currently implemented for numeric vectors, numeric matrices and data frames.

* *sum(rowMeans(normalized\_ratings) > 0.00001)*

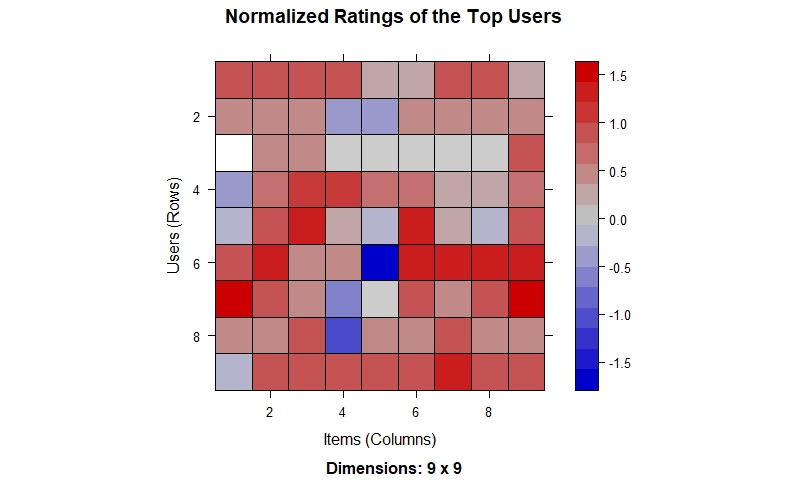
Sum() function is used to calculate the sum of vector elements.

* *image(normalized\_ratings[rowCounts(normalized\_ratings) > minimum\_movies,*

*colCounts(normalized\_ratings) > minimum\_users],*

*main = "Normalized Ratings of the Top Users")*

Create the image of normalize rating



## 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.

* *binary\_minimum\_movies <- quantile(rowCounts(movie\_ratings), 0.95)*
* *binary\_minimum\_users <- quantile(colCounts(movie\_ratings), 0.95)*

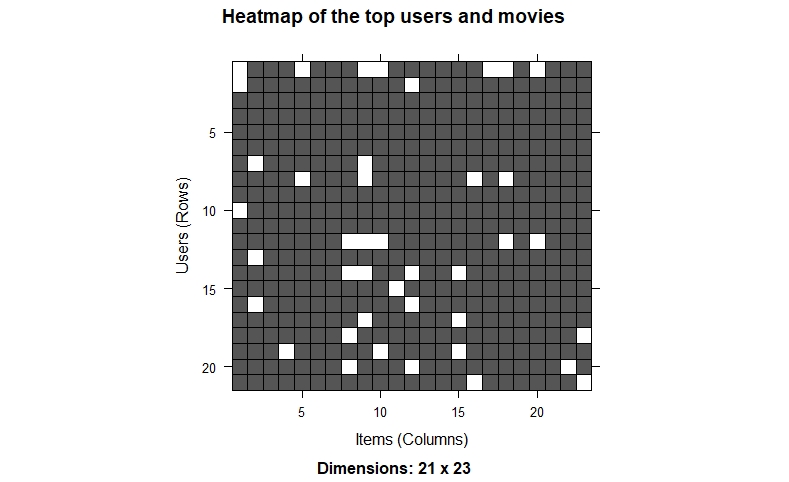
movies\_watched <- binarize(movie\_ratings, minRating = 1)

* *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")*

We will define a matrix that will consist of 1 if the rating is above 3 and otherwise it will be 0.

**

# Collaborative Filtering System

In this section of the data science project, we will develop 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 a record that the customer purchased items i1 and i2.
* Calculate the similarity between i1 and i2.
* *sampled\_data<- sample(x = c(TRUE, FALSE),*

*size = nrow(movie\_ratings),*

*replace = TRUE,*

*prob = c(0.8, 0.2))*

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

* *training\_data <- movie\_ratings[sampled\_data, ]*
* *testing\_data <- movie\_ratings[!sampled\_data, ]*

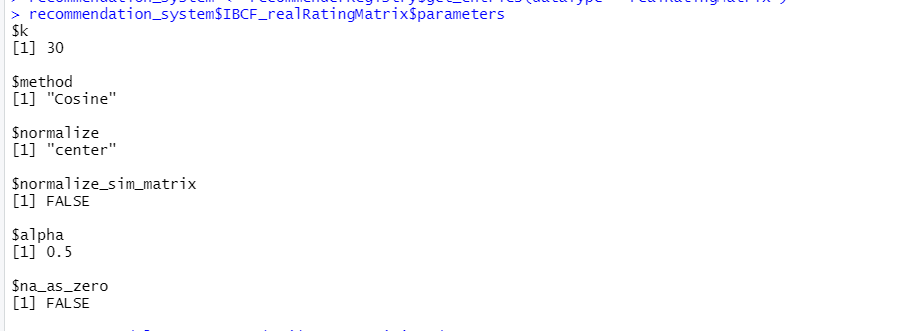
# Building the Recommendation System using R

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.

* *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)*

Using the getModel() function, we will retrieve the recommen\_model.

* *class(model\_info$sim)*

We will then find the class and dimensions of our similarity matrix that is contained within model\_info.

**

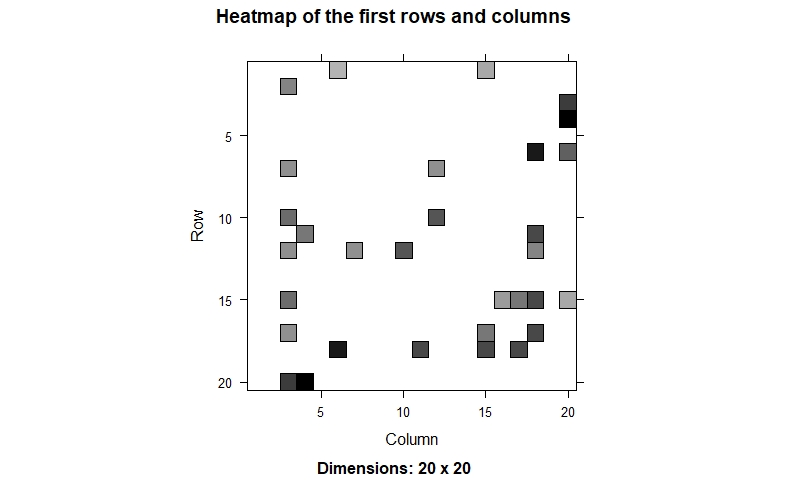
* *dim(model\_info$sim)*

**

* *top\_items <- 20*
* *image(model\_info$sim[1:top\_items, 1:top\_items],*

*main = "Heatmap of the first rows and columns")*

Finally, we will generate a heatmap, that will contain the top 20 items and visualize the similarity shared between them.

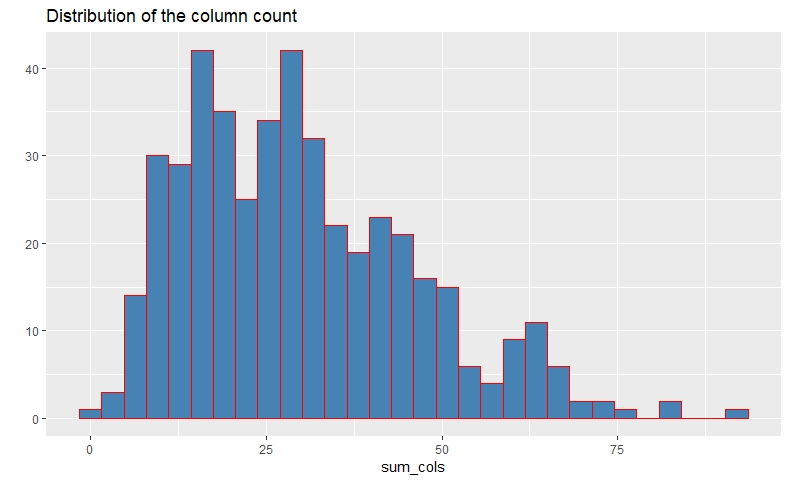
**

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)*

table() function in R Language is used to create a categorical representation of data with variable name and the frequency in the form of a table.

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



# 

# How to build Recommender System on dataset using R?

Here, each rating is used as a weight. Each weight is multiplied with related similarities. Finally, everything is added in the end.

* *top\_recommendations <- 10*

We will create a top\_recommendations variable which will be initialized to 10, specifying the number of films to each user.

* *predicted\_recommendations <- predict(object = recommen\_model,*

*newdata = testing\_data,*

*n = top\_recommendations)*

We will then use the predict() function that will identify similar items and will rank them appropriately.

* *predicted\_recommendations*

Display the information about model.

* *user1 <- predicted\_recommendations@items[[1]]*

recommendation for the first user

* *movies\_user1 <- predicted\_recommendations@itemLabels[user1]*

Movie user 1 watches

* *movies\_user2 <- movies\_user1*

Recommending to user movies that user 1 watch

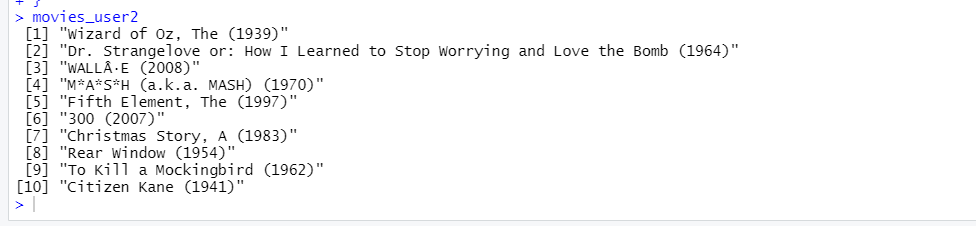
* *for (index in 1:10){*

*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]*

**

# Summary

*Recommendation Systems are the most famous kind of AI applications that are utilized in all areas. They are an improvement over the conventional arrangement calculations as they can take many classes of info and give comparability positioning-based calculations to furnish the client with exact outcomes. These proposal frameworks have developed over the long run and have consolidated many high-level AI methods to furnish the clients with their desired substance.*

# Reference:

https://www.rdocumentation.org/packages/recommenderlab/versions/0.2-7