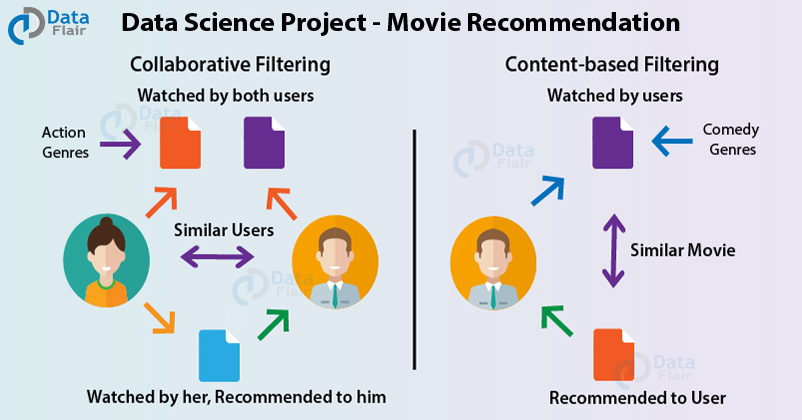
Machine Learning Project – Data Science Movie Recommendation System Project in R

BY [DATAFLAIR TEAM](https://data-flair.training/blogs/author/dfteam2/) · UPDATED · JULY 31, 2019

Have you ever been on an online streaming platform like Netflix, Amazon Prime, Voot? I watched a movie and after some time, that platform started recommending me different movies and TV shows. I wondered, how the movie streaming platform could suggest me content that appealed to me. Then I came across something known as Recommendation System. This system is capable of learning my watching patterns and providing me with relevant suggestions. Having witnessed the fourth industrial revolution where [***Artificial Intelligence***](https://data-flair.training/blogs/ai-tutorials-home/) and other technologies are dominating the market, I am sure that you must have come across a recommendation system in your everyday life. I am also sure that by this time curiosity must be getting the best of you. Therefore, in this Machine Learning Project, I will teach you to build your own recommendation system. So. let’s start.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/data-science-movie-recommendation-project.jpg)

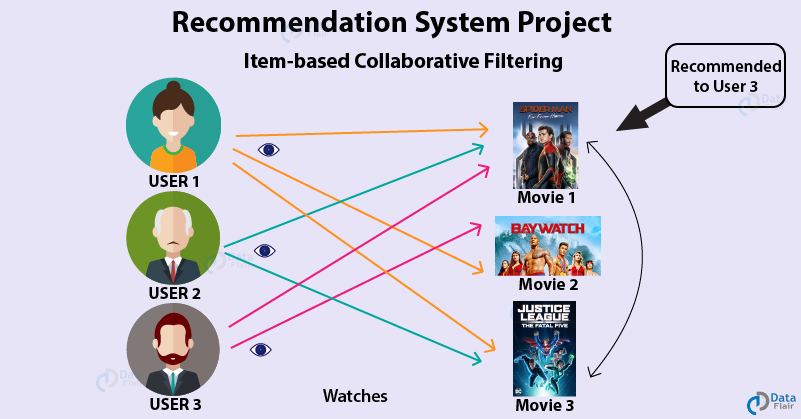
Movie Recommendation System Project using ML

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 help you understand the functioning of how a recommendation system works. We will be developing an Item Based Collaborative Filter. By the end of this tutorial, you will gain experience of implementing your R, Data Science, and ***Machine learning skills*** in a real-life project.

Before moving ahead in this movie recommendation system project in ML, you need to know what recommendation system means. Read below to find the answer.

What is a Recommendation System?

*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. This information reflects the prior usage of the product as well as the assigned ratings. A recommendation system is a platform that provides its users with various contents based on their preferences and likings. A recommendation system takes the information about the user as an input. The recommendation system is an implementation of the [***machine learning algorithms***](https://data-flair.training/blogs/machine-learning-algorithms/).

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/recommendation-system-project-in-R.png)

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.

***You must check***[***how Netflix recommendation engine works***](https://data-flair.training/blogs/data-science-at-netflix/)

How to build a Movie Recommendation System using Machine Learning

Dataset

In order 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 [***here***](https://drive.google.com/file/d/1Dn1BZD3YxgBQJSIjbfNnmCFlDW2jdQGD/view). This data consists of 105339 ratings applied over 10329 movies.

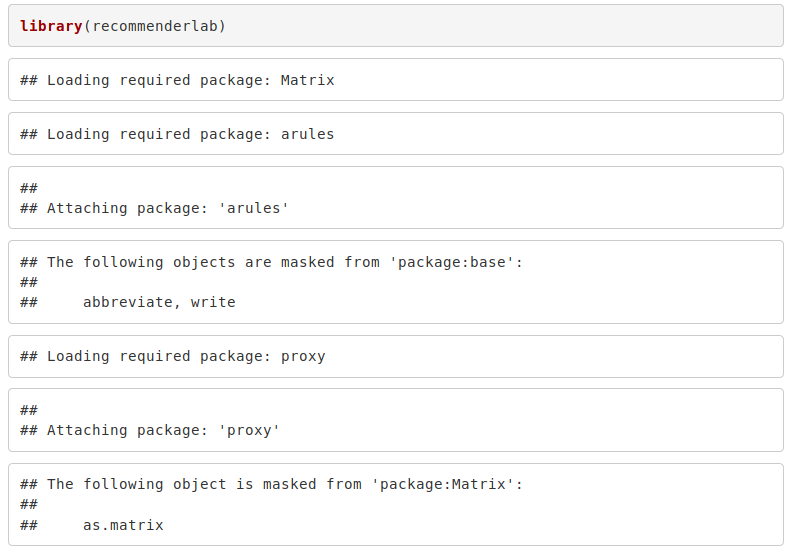
Importing Essential Libraries

In our Data Science project, we will make use of these four packages – *‘recommenderlab’, ‘ggplot2’, ‘data.table’ and ‘reshape2’.*

**Code:**

1. **library**(recommenderlab)

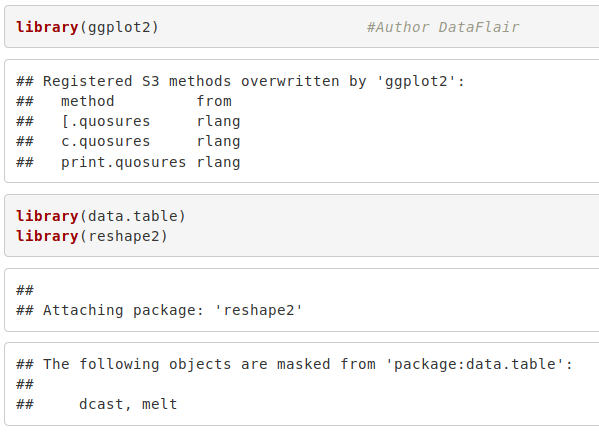
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/import-libraries-using-R-packages.png)

**Code:**

1. **library**(ggplot2) #Author DataFlair
2. **library**(data.table)
3. **library**(reshape2)

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/import-libraries-using-data-science-packages.png)

***Wait! Don’t forget to check our leading guide on***[***R programming classification***](https://data-flair.training/blogs/classification-in-r/)

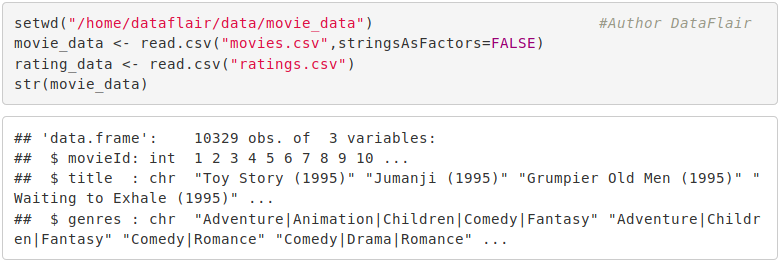
Retrieving the Data

We will now retrieve our data from movies.csv into movie\_data dataframe and ratings.csv into rating\_data. We will use the str() function to display information about the movie\_data dataframe.

**Code:**

1. **setwd**("/home/dataflair/data/movie\_data") #Author DataFlair
2. movie\_data <- read.**csv**("movies.csv",stringsAsFactors=FALSE)
3. rating\_data <- read.**csv**("ratings.csv")
4. **str**(movie\_data)

**Output Screenshot:**

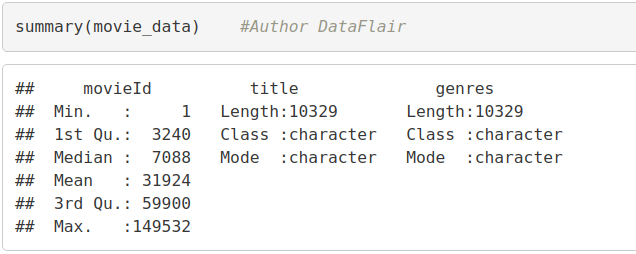
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/retrieving-the-data-from-dataframes.png)

We can 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

**Code:**

1. **summary**(movie\_data) #Author DataFlair

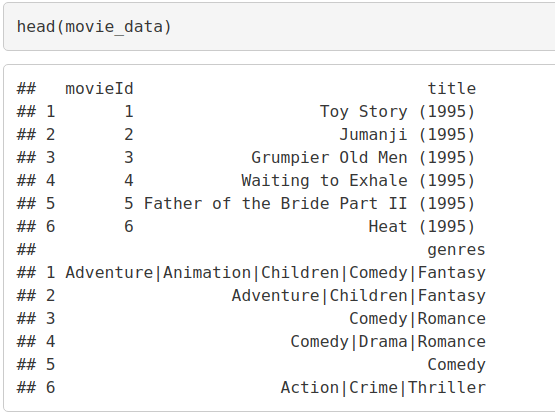
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/using-summary-function.png)

**Code:**

1. **head**(movie\_data)

**Output Screenshot:**

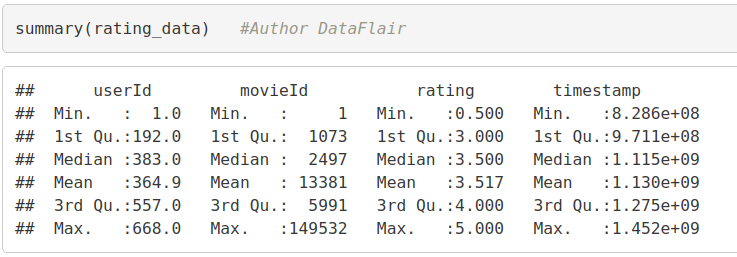
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/head-function-in-R-.png)

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

**Code:**

1. **summary**(rating\_data) #Author DataFlair

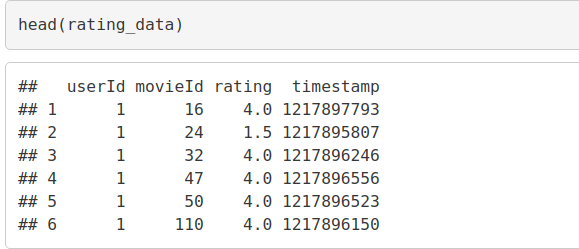
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/rating-head.png)

**Code:**

1. **head**(rating\_data)

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/rating-summary.png)

***Revise your R concepts with DataFlair for Free, checkout***[***120+ FREE R Tutorials***](https://data-flair.training/blogs/r-tutorials-home/)

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 films.

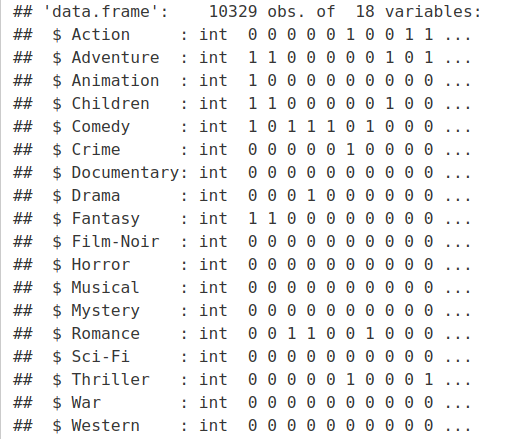
**Code:**

1. movie\_genre <- as.data.**frame**(movie\_data$genres, stringsAsFactors=FALSE)
2. **library**(data.table)
3. movie\_genre2 <- as.data.**frame**(**tstrsplit**(movie\_genre[,1], '[|]',
4. type.convert=TRUE),
5. stringsAsFactors=FALSE) #DataFlair
6. **colnames**(movie\_genre2) <- **c**(1:10)
7. list\_genre <- **c**("Action", "Adventure", "Animation", "Children",
8. "Comedy", "Crime","Documentary", "Drama", "Fantasy",
9. "Film-Noir", "Horror", "Musical", "Mystery","Romance",
10. "Sci-Fi", "Thriller", "War", "Western")
11. genre\_mat1 <- **matrix**(0,10330,18)
12. genre\_mat1[1,] <- list\_genre
13. **colnames**(genre\_mat1) <- list\_genre
14. **for** (index in 1:**nrow**(movie\_genre2)) {
15. **for** (col in 1:**ncol**(movie\_genre2)) {
16. gen\_col = **which**(genre\_mat1[1,] == movie\_genre2[index,col]) #Author DataFlair
17. genre\_mat1[index+1,gen\_col] <- 1
18. }
19. }
20. genre\_mat2 <- as.data.**frame**(genre\_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list
21. **for** (col in 1:**ncol**(genre\_mat2)) {
22. genre\_mat2[,col] <- as.**integer**(genre\_mat2[,col]) #convert from characters to integers
23. }
24. **str**(genre\_mat2)

**Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/creating-a-matrix-in-R.png)

**Output –**

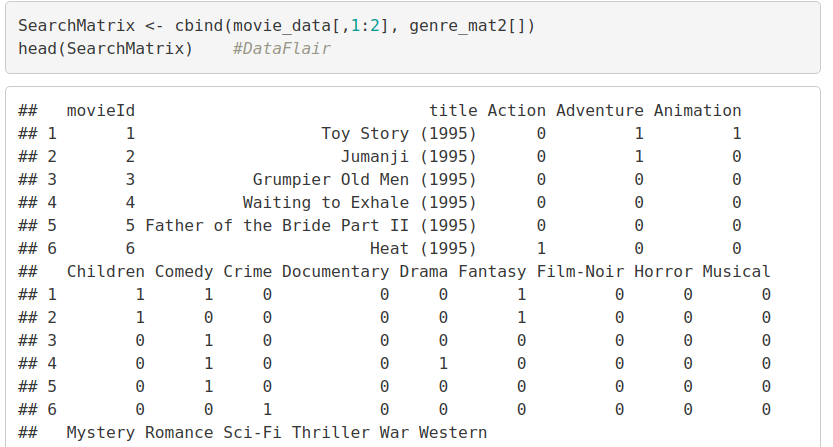
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/Create-matrix-output-in-R-recommendation-project.png)

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:**

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

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/create-search-matrix-in-r-recommendation-project.png)

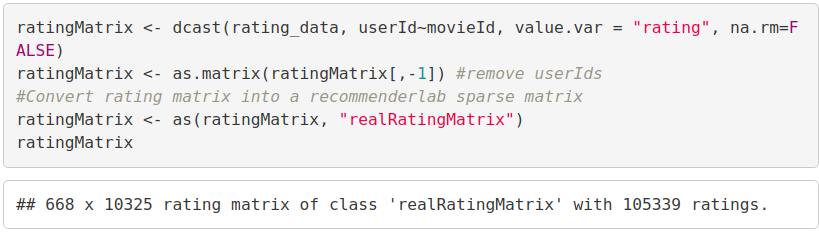
There are movies that have several genres, for example, Toy Story, which is an animated film also falls under the genres of Comedy, Fantasy, and Children. This applies to the majority of the films.

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:**

1. ratingMatrix <- **dcast**(rating\_data, userId~movieId, value.var = "rating", na.rm=FALSE)
2. ratingMatrix <- as.**matrix**(ratingMatrix[,-1]) #remove userIds
3. #Convert rating matrix into a recommenderlab sparse matrix
4. ratingMatrix <- **as**(ratingMatrix, "realRatingMatrix")
5. ratingMatrix

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/matrix-into-sparse-matrix.png)

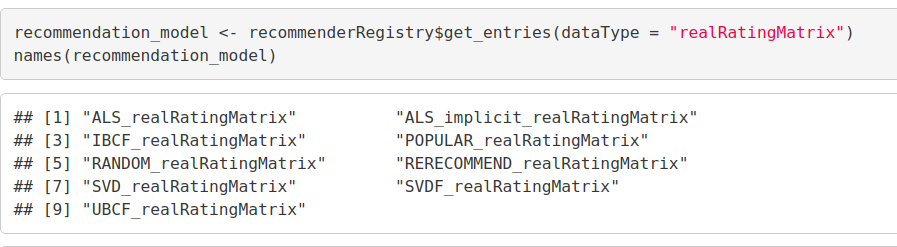
*Are you facing any trouble in implementing recommendation system project in R? Comment below, DataFlair Team is ready to help you.*

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

**Code:**

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

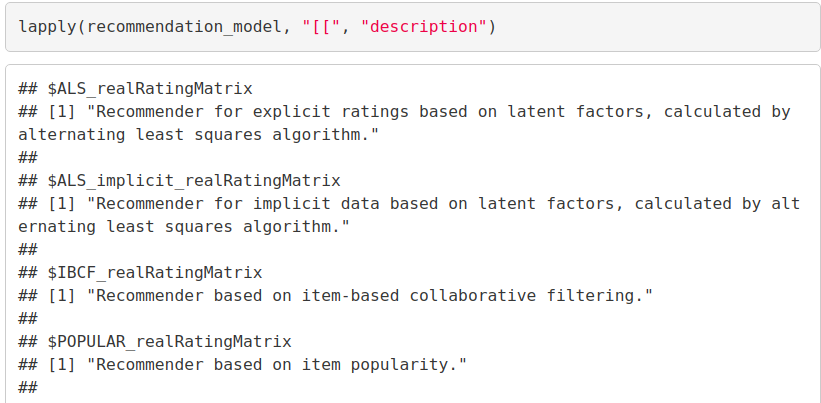
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/building-movies-recommendation-system.png)

**Code:**

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

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/recommendation-model.png)

***Want to become the next data scientist? Try out the best way and explore***[***Data Science Tutorials Series***](https://data-flair.training/blogs/data-science-tutorials-home/)***to learn Data Science in an easy way with DataFlair!!***

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

**Code:**

1. recommendation\_model$IBCF\_realRatingMatrix$parameters

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/implementing-single-model-in-data-science-project.png)

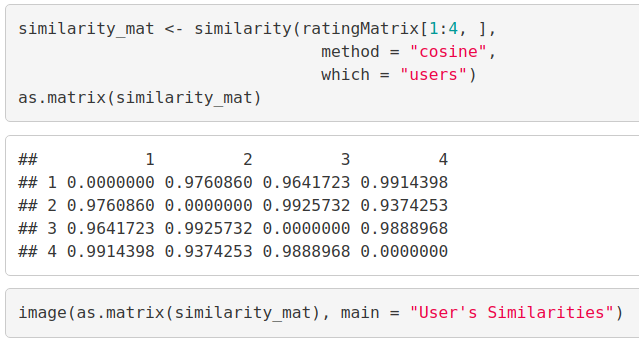
Exploring Similar Data

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:**

1. similarity\_mat <- **similarity**(ratingMatrix[1:4, ],
2. method = "cosine",
3. which = "users")
4. as.**matrix**(similarity\_mat)
5. **image**(as.**matrix**(similarity\_mat), main = "User's Similarities")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/using-collaborative-filtering-in-R.png)

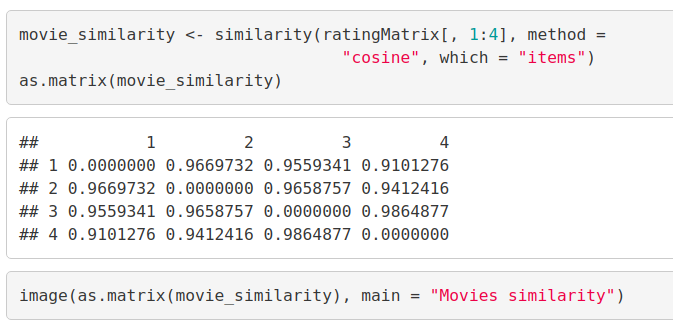
In the above matrix, each row and column represents a user. We have taken four users and each cell in this matrix represents the similarity that is shared between the two users.

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

**Code:**

1. movie\_similarity <- **similarity**(ratingMatrix[, 1:4], method =
2. "cosine", which = "items")
3. as.**matrix**(movie\_similarity)
4. **image**(as.**matrix**(movie\_similarity), main = "Movies similarity")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/delinate-the-similarity.png)

Let us now extract the most unique ratings –

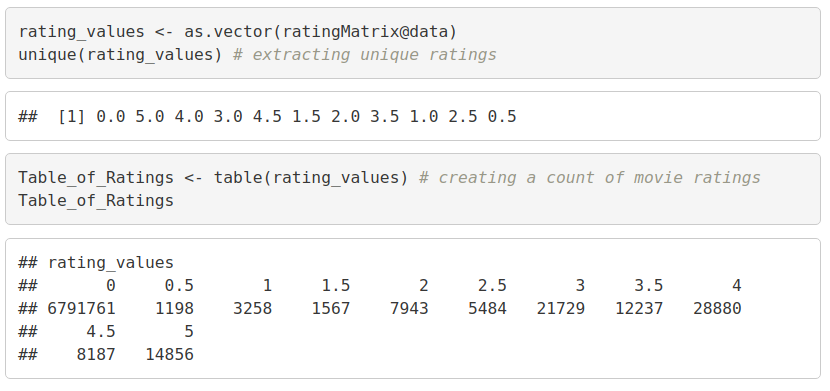
1. rating\_values <- as.**vector**(ratingMatrix@data)
2. **unique**(rating\_values) # extracting unique ratings

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

**Code:**

1. Table\_of\_Ratings <- **table**(rating\_values) # creating a count of movie ratings
2. Table\_of\_Ratings

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/rating-displaying-unique-ratings.png)

***This is the right time to check your R and Data Science Learning. Try these***[***latest interview questions***](https://data-flair.training/blogs/data-science-tutorials-home/#interview-questions)***and become a pro.***

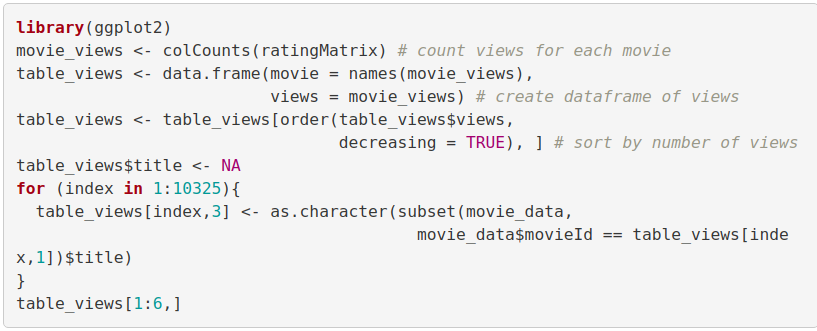
Most Viewed Movies Visualization

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.

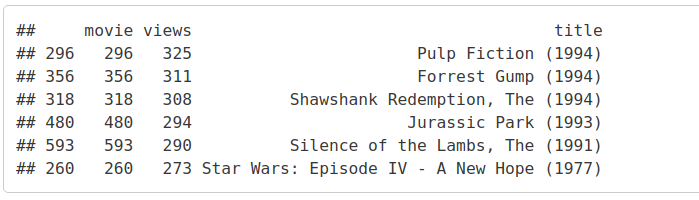
**Code:**

1. **library**(ggplot2)
2. movie\_views <- **colCounts**(ratingMatrix) # count views for each movie
3. table\_views <- data.**frame**(movie = **names**(movie\_views),
4. views = movie\_views) # create dataframe of views
5. table\_views <- table\_views[**order**(table\_views$views,
6. decreasing = TRUE), ] # sort by number of views
7. table\_views$title <- NA
8. **for** (index in 1:10325){
9. table\_views[index,3] <- as.**character**(**subset**(movie\_data,
10. movie\_data$movieId == table\_views[index,1])$title)
11. }
12. table\_views[1:6,]

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/counting-the-number-of-views-in-a-movie.png)

**Output –**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/most-viewed-movies-visualization.png)

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

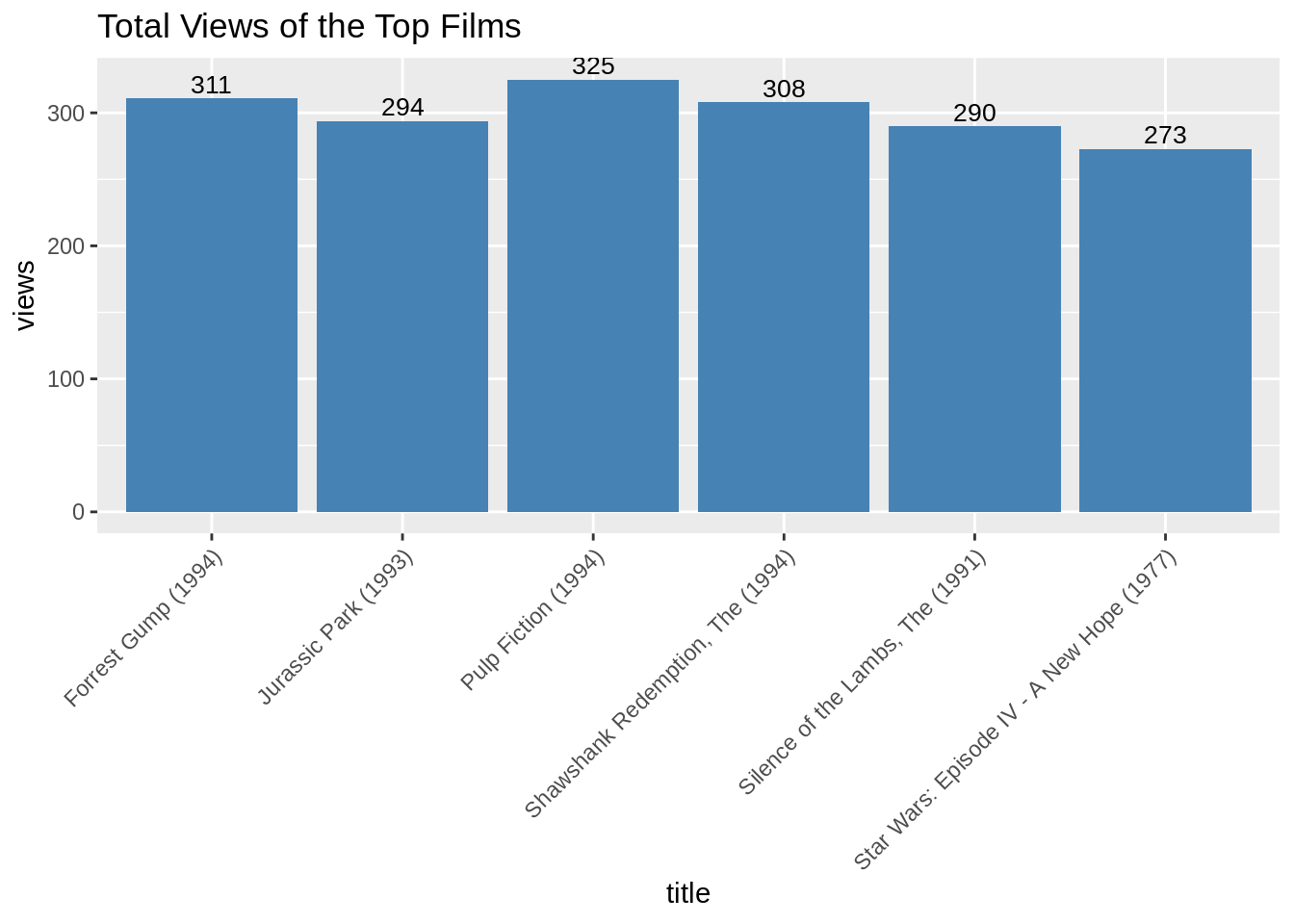
**Code:**

1. **ggplot**(table\_views[1:6, ], **aes**(x = title, y = views)) +
2. **geom\_bar**(stat="identity", fill = 'steelblue') +
3. **geom\_text**(**aes**(label=views), vjust=-0.3, size=3.5) +
4. **theme**(axis.text.x = **element\_text**(angle = 45, hjust = 1)) +
5. **ggtitle**("Total Views of the Top Films")

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/R-visualization-views-on-the-top-films.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/most-viewed-films-bar-plot-in-ML.png)

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

***If you are enjoying this Data Science Recommendation System Project, DataFlair brings another project for you –***[***Credit Card Fraud Detection using R***](https://data-flair.training/blogs/data-science-machine-learning-project-credit-card-fraud-detection/)***. Save the link, you can thank me later*😉**

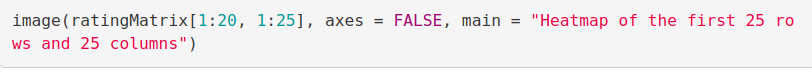
Heatmap of Movie Ratings

Now, in this data science project of Recommendation system, we will visualize a heatmap of the movie ratings. This heatmap will contain first 25 rows and 25 columns as follows –

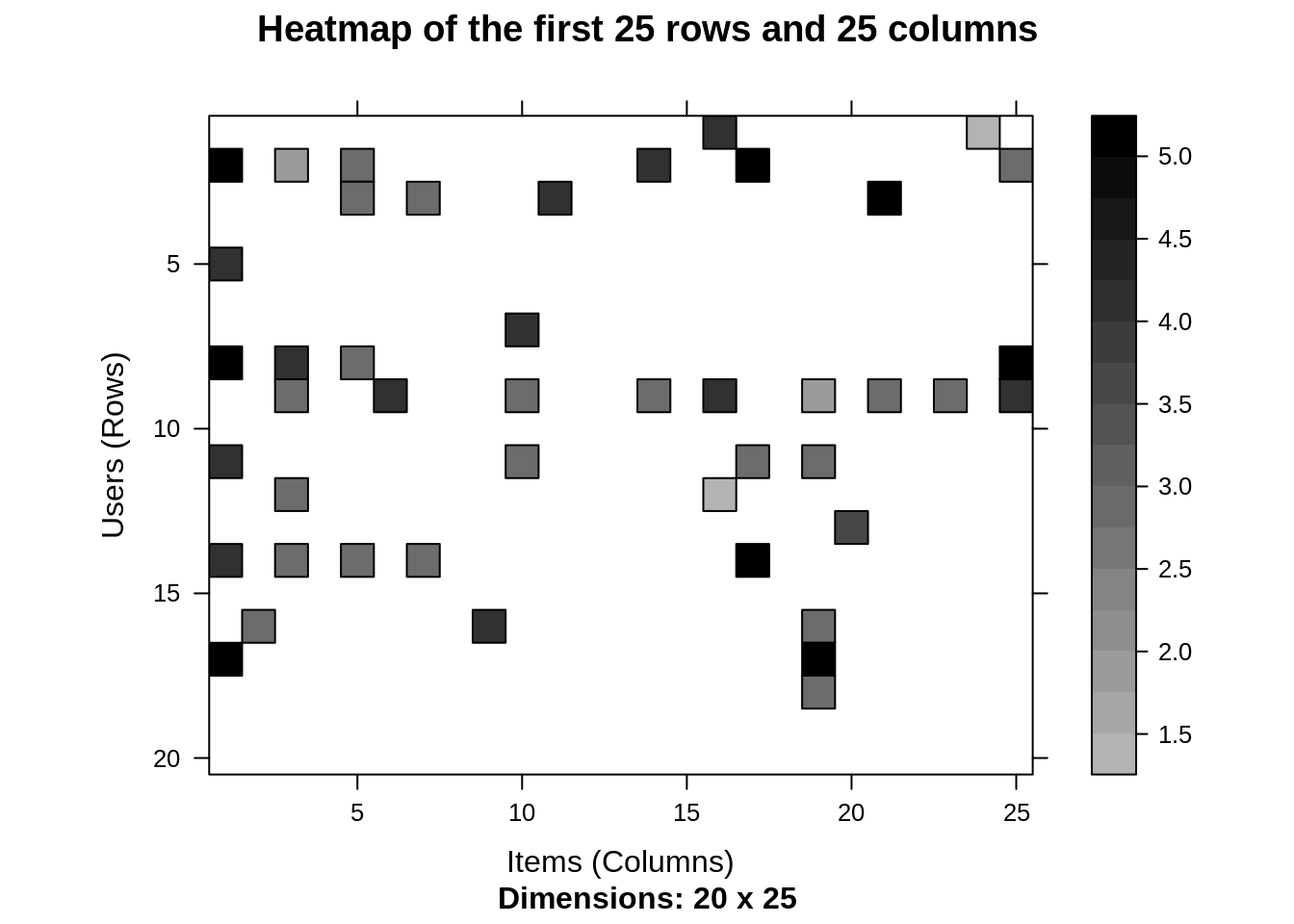
**Code:**

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

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/visualizing-heatmaps-of-movie-ratings.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/Output-of-R-heatmap-of-movie-ratings-visualization.png)

Performing Data Preparation

We will conduct data preparation in the following three steps –

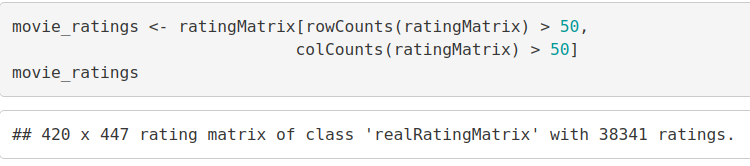
* Selecting useful data.
* Normalizing data.
* Binarizing the data.

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.

**Code:**

1. movie\_ratings <- ratingMatrix[**rowCounts**(ratingMatrix) > 50,
2. **colCounts**(ratingMatrix) > 50]
3. Movie\_ratings

**Output Screenshot:**

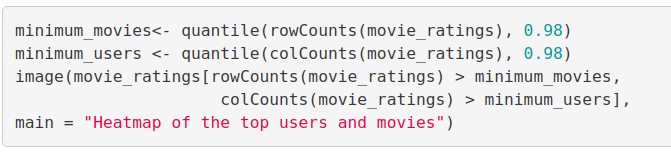
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/conducting-data-preparation-in-data-science-project.png)

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

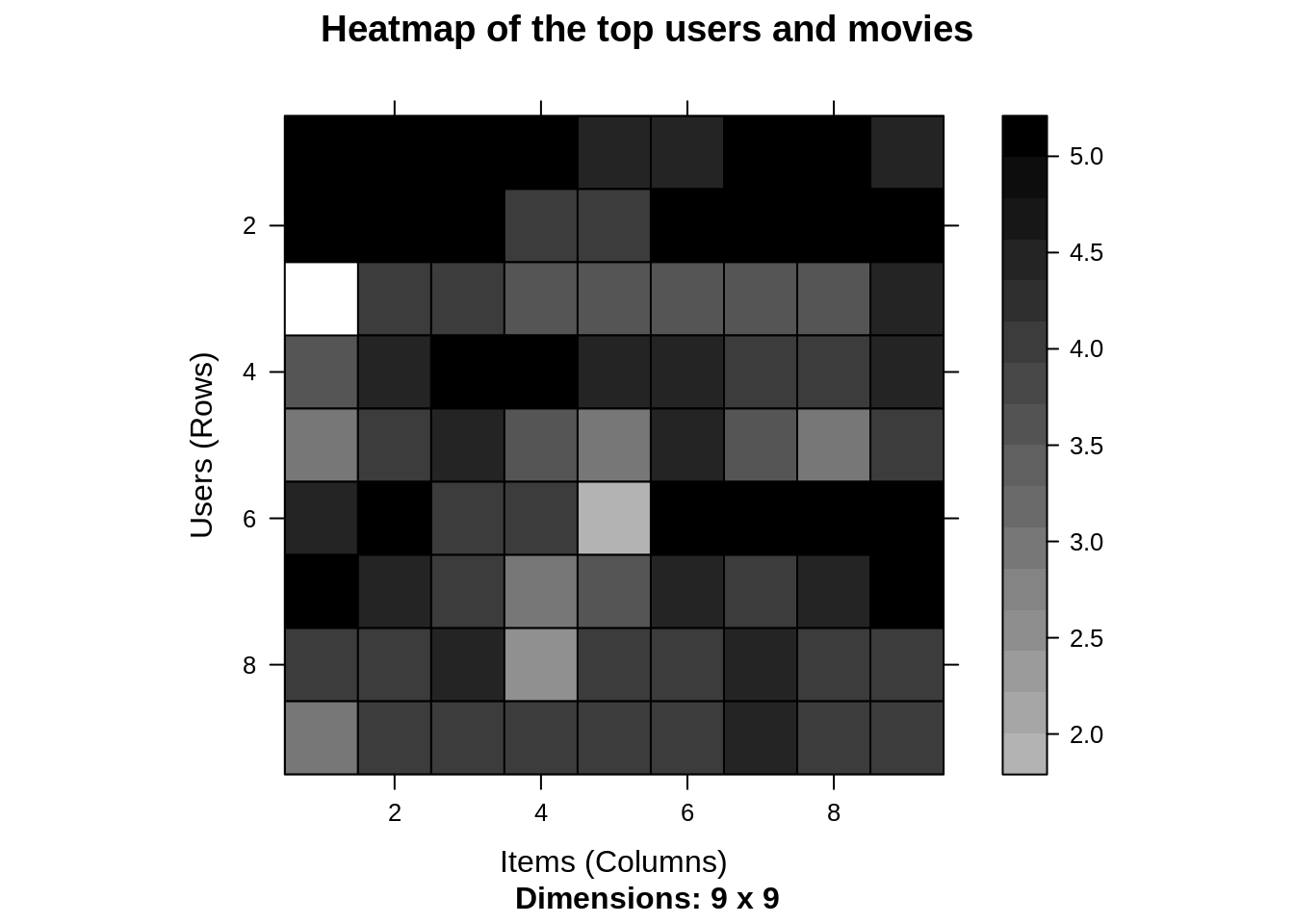
**Code:**

1. minimum\_movies<- **quantile**(**rowCounts**(movie\_ratings), 0.98)
2. minimum\_users <- **quantile**(**colCounts**(movie\_ratings), 0.98)
3. **image**(movie\_ratings[**rowCounts**(movie\_ratings) > minimum\_movies,
4. **colCounts**(movie\_ratings) > minimum\_users],
5. main = "Heatmap of the top users and movies")

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/delinate-matrix-of-users.png)

**Output:**

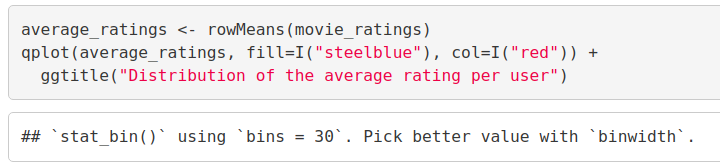
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/R-heatmap-for-top-users-and-films.png)

[***Data Visualization in R***](https://data-flair.training/blogs/data-visualization-in-r/)***– Learn the concepts in an easy way***

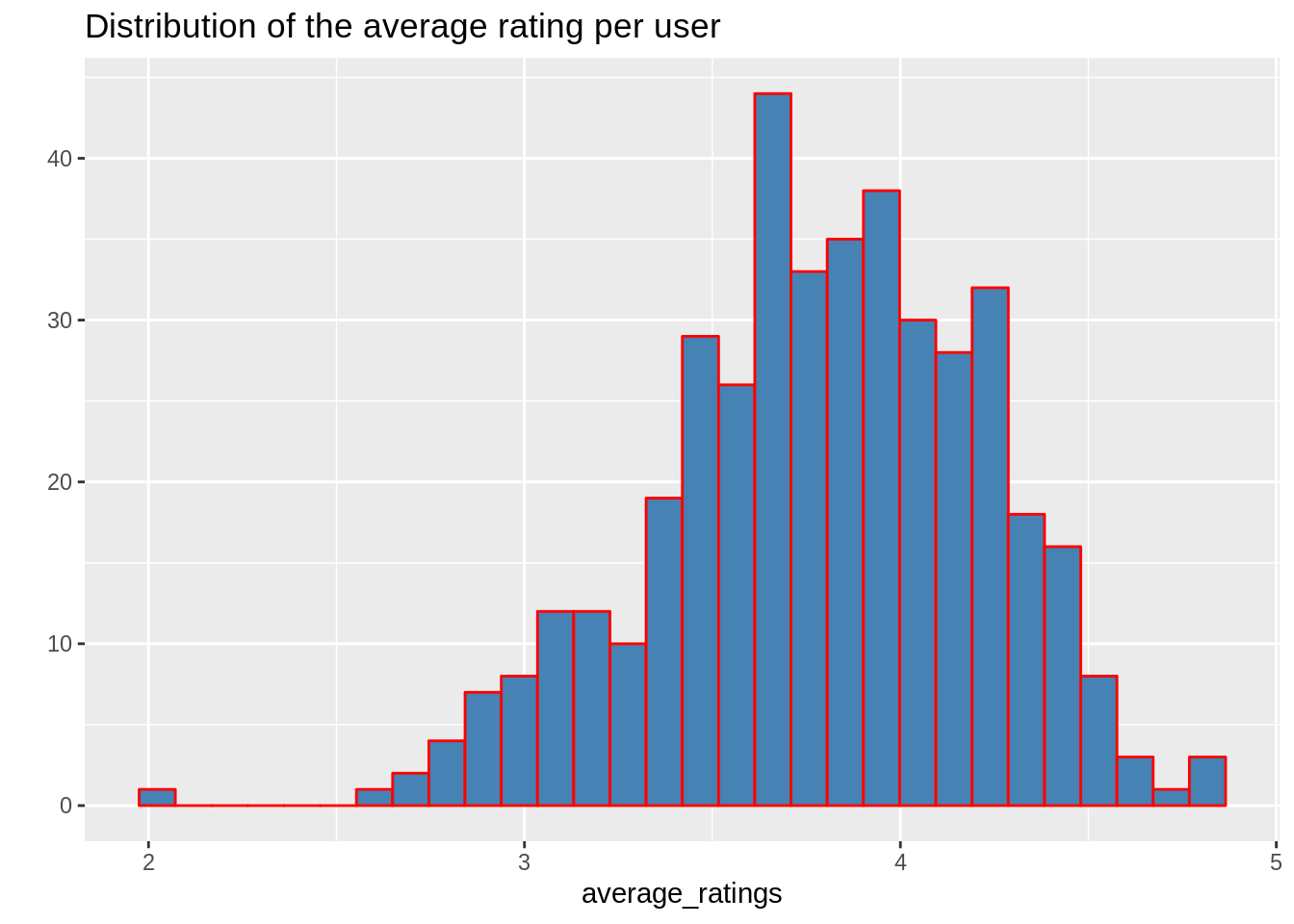
Now, we will visualize the distribution of the average ratings per user.

1. average\_ratings <- **rowMeans**(movie\_ratings)
2. **qplot**(average\_ratings, fill=**I**("steelblue"), col=**I**("red")) +
3. **ggtitle**("Distribution of the average rating per user")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/distribution-of-average-rating-per-userinput.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/Distribution-of-Average-Ratings-per-User.png)

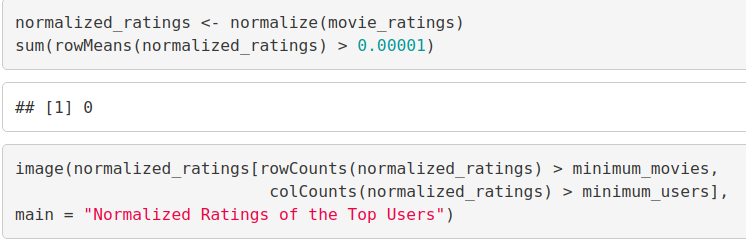
Data Normalization

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.

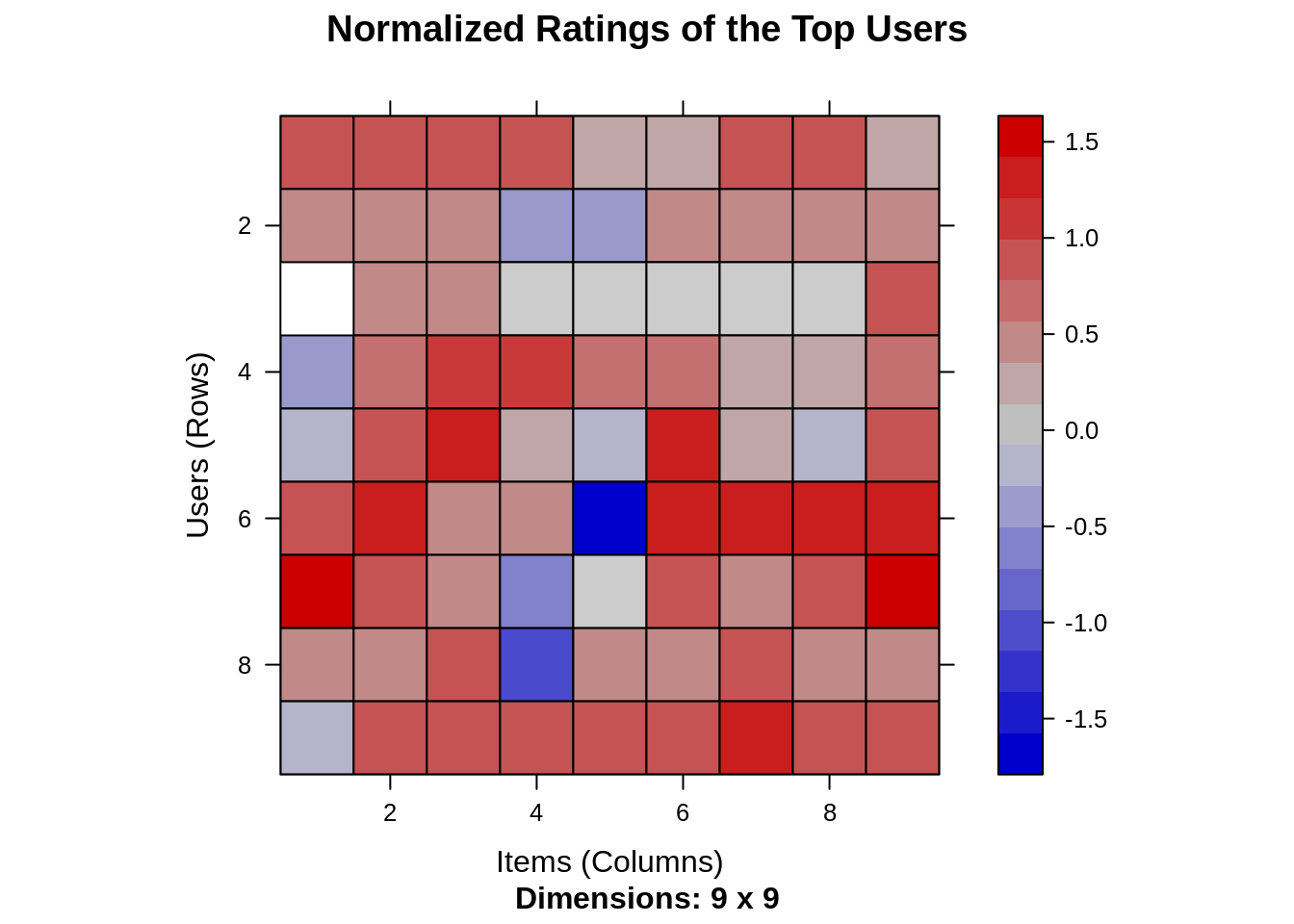
**Code:**

1. normalized\_ratings <- **normalize**(movie\_ratings)
2. **sum**(**rowMeans**(normalized\_ratings) > 0.00001)
3. **image**(normalized\_ratings[**rowCounts**(normalized\_ratings) > minimum\_movies,
4. **colCounts**(normalized\_ratings) > minimum\_users],
5. main = "Normalized Ratings of the Top Users")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/R-data-normalization.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/normaliztion-in-R-rating-of-top-users.png)

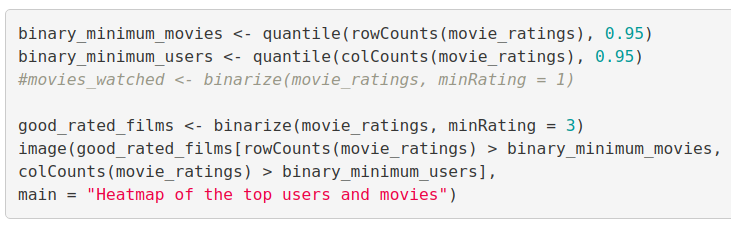
Performing Data Binarization

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.

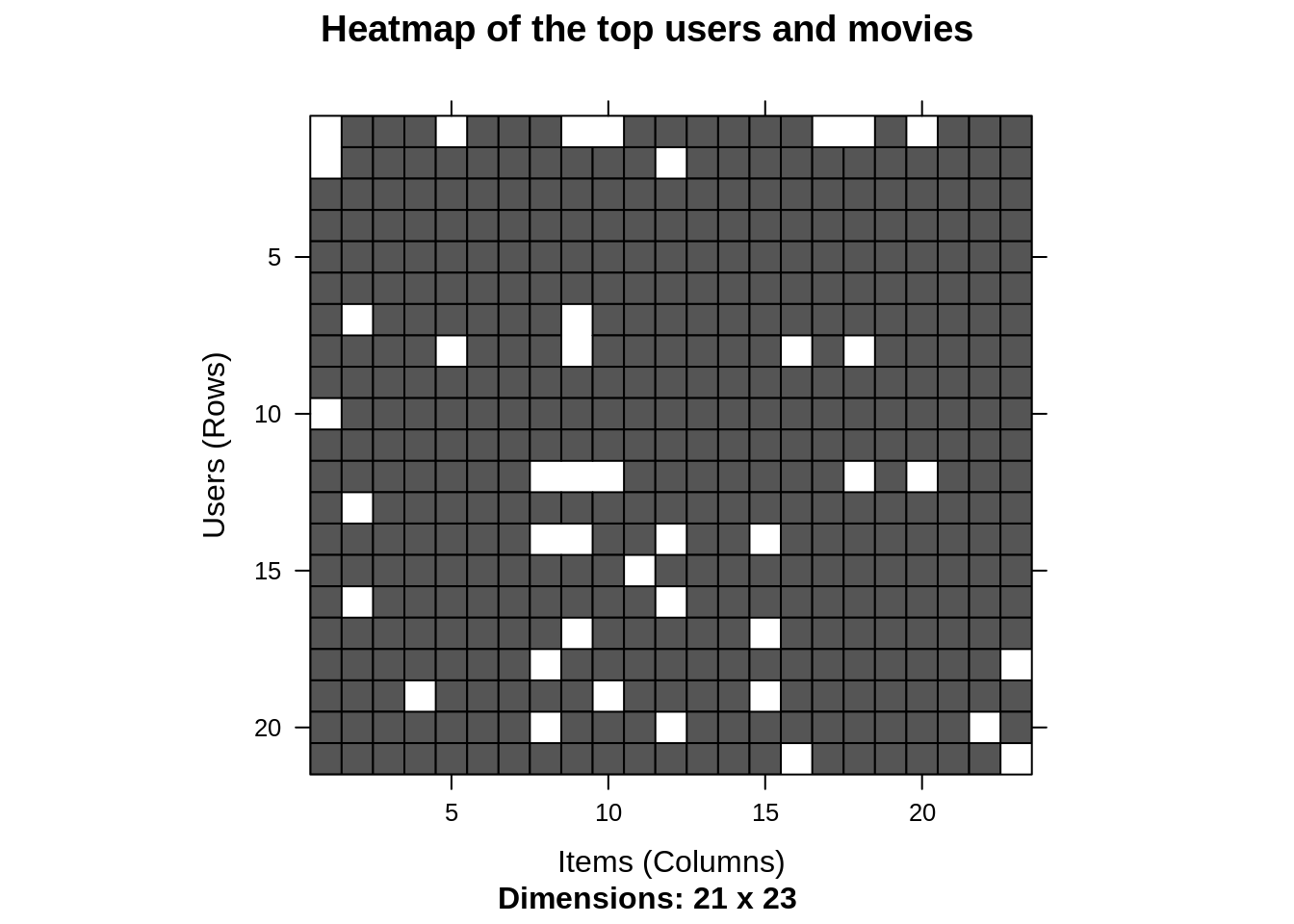
**Code:**

1. binary\_minimum\_movies <- **quantile**(**rowCounts**(movie\_ratings), 0.95)
2. binary\_minimum\_users <- **quantile**(**colCounts**(movie\_ratings), 0.95)
3. #movies\_watched <- binarize(movie\_ratings, minRating = 1)
4. good\_rated\_films <- **binarize**(movie\_ratings, minRating = 3)
5. **image**(good\_rated\_films[**rowCounts**(movie\_ratings) > binary\_minimum\_movies,
6. **colCounts**(movie\_ratings) > binary\_minimum\_users],
7. main = "Heatmap of the top users and movies")

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/defining-a-matrix-binarization-in-R.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/data-science-binarization-heatmap-in-R.png)

***Didn’t you check what’s trending on DataFlair? Check out the latest***[***Machine Learning Tutorial Series***](https://data-flair.training/blogs/machine-learning-tutorials-home/)***. Master all the ML concepts for FREE NOW!!***

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 –

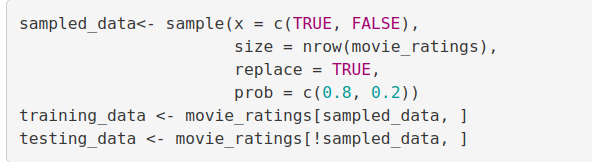
* For each Item i1 present in the product catalog, purchased by customer C.
* And, for each item i2 also purchased by the customer C.
* Create record that the customer purchased items i1 and i2.
* Calculate the similarity between i1 and i2.

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

**Code:**

1. sampled\_data<- **sample**(x = **c**(TRUE, FALSE),
2. size = **nrow**(movie\_ratings),
3. replace = TRUE,
4. prob = **c**(0.8, 0.2))
5. training\_data <- movie\_ratings[sampled\_data, ]
6. testing\_data <- movie\_ratings[!sampled\_data, ]

**Input Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/collaborative-filtering-in-R.png)

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:**

1. recommendation\_system <- recommenderRegistry$**get\_entries**(dataType ="realRatingMatrix")
2. recommendation\_system$IBCF\_realRatingMatrix$parameters

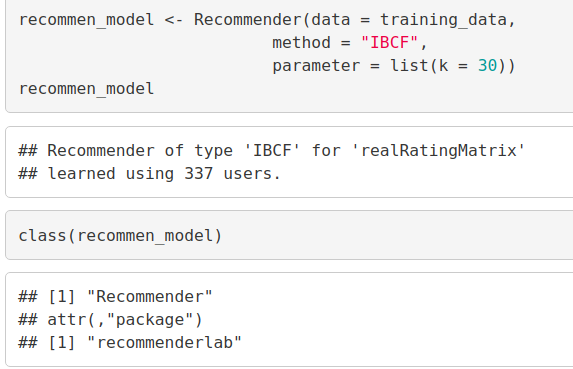
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/building-recommendation-engine-in-R.png)

**Code:**

1. recommen\_model <- **Recommender**(data = training\_data,
2. method = "IBCF",
3. parameter = **list**(k = 30))
4. recommen\_model
5. **class**(recommen\_model)

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/building-recommendation-system-in-R-.png)

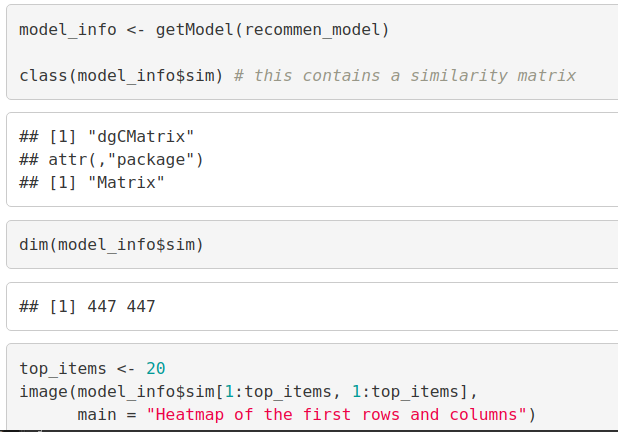
Let us now explore our data science recommendation system model as follows –

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

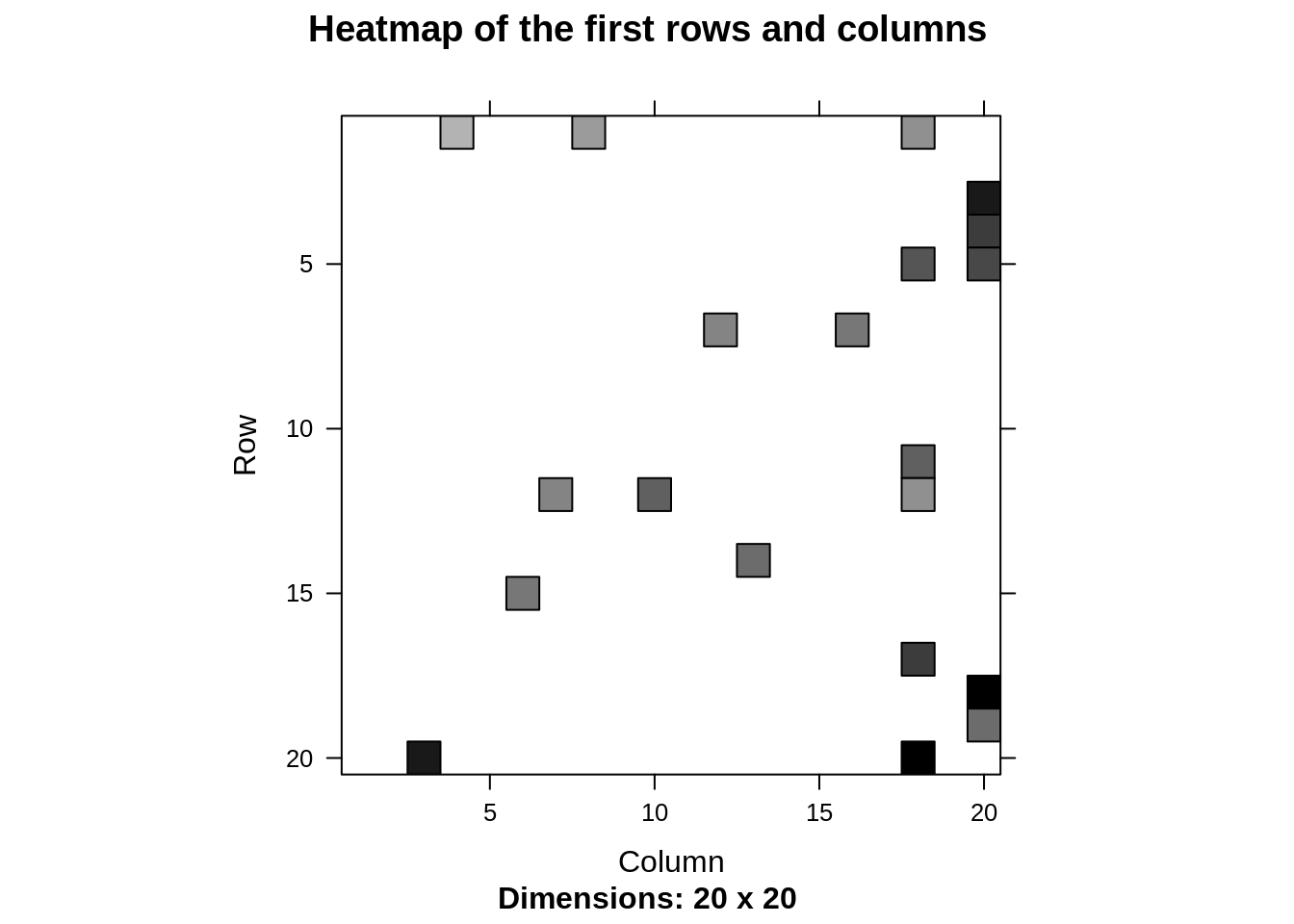
**Code:**

1. model\_info <- **getModel**(recommen\_model)
2. **class**(model\_info$sim)
3. **dim**(model\_info$sim)
4. top\_items <- 20
5. **image**(model\_info$sim[1:top\_items, 1:top\_items],
6. main = "Heatmap of the first rows and columns")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/similarity-visualization-in-R.png)

**Output:**

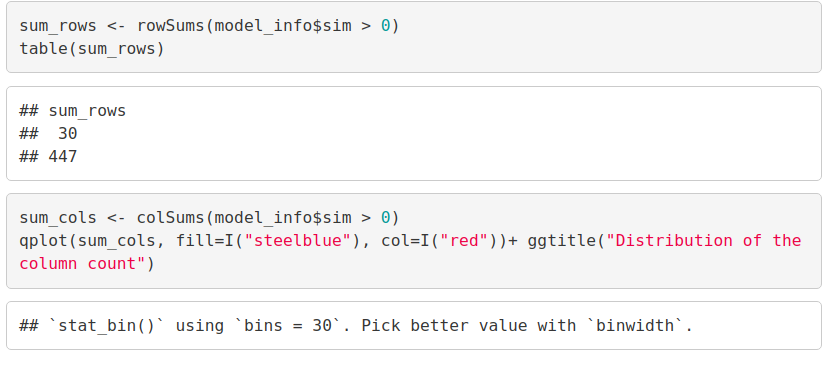
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/machine-learning-user-similarity-heatmap-in-R.png)

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 –

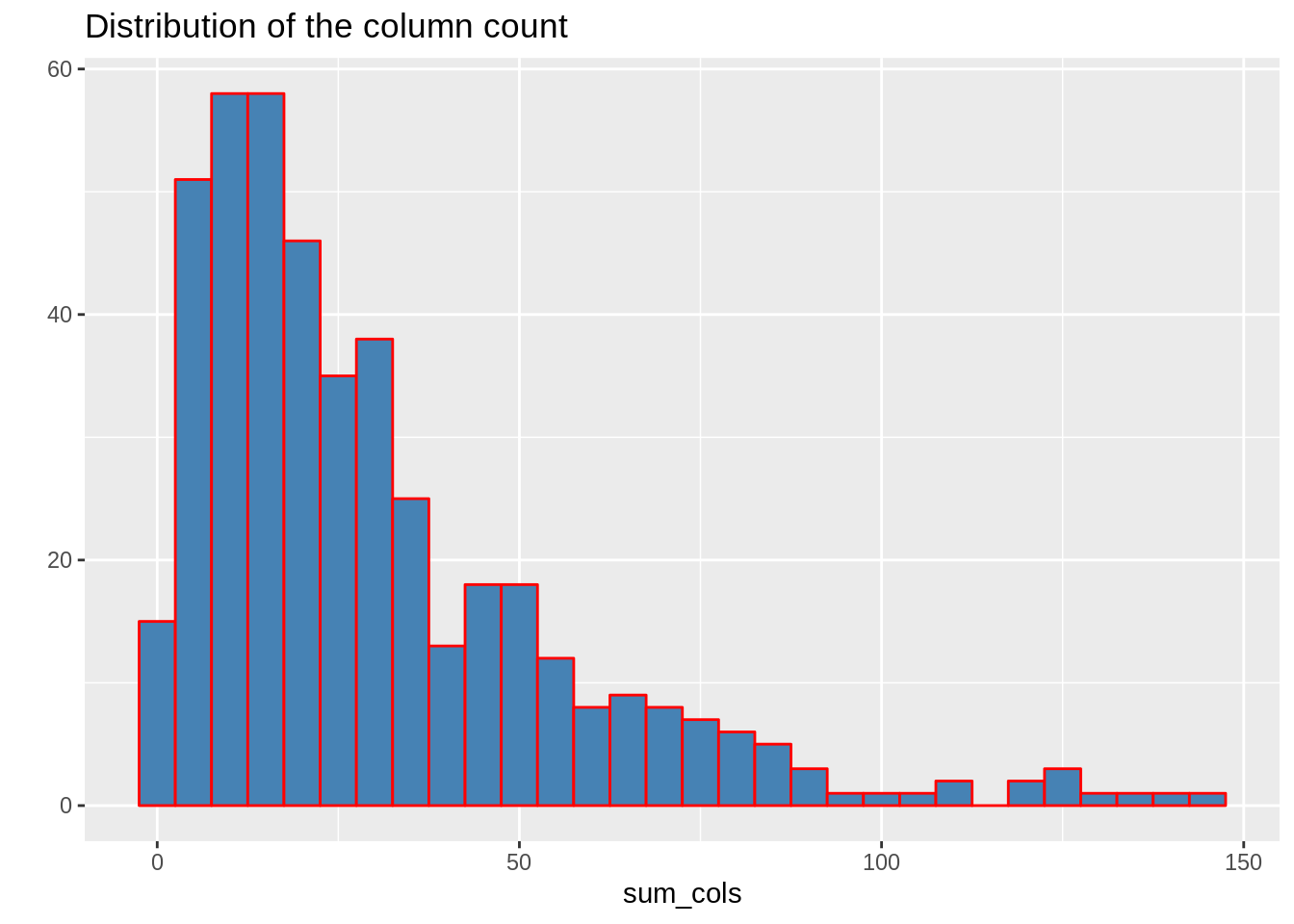
**Code:**

1. sum\_rows <- **rowSums**(model\_info$sim > 0)
2. **table**(sum\_rows)
3. sum\_cols <- **colSums**(model\_info$sim > 0)
4. **qplot**(sum\_cols, fill=**I**("steelblue"), col=**I**("red"))+ **ggtitle**("Distribution of the column count")

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/visualizing-the-sum-of-columns.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/data-science-distribution-visualization-in-R.png)

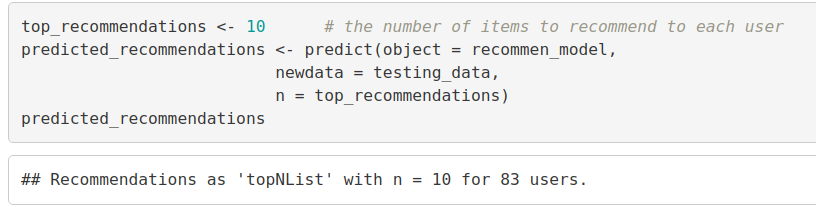
How to build Recommender System on dataset using R?

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:**

1. top\_recommendations <- 10 # the number of items to recommend to each user
2. predicted\_recommendations <- **predict**(object = recommen\_model,
3. newdata = testing\_data,
4. n = top\_recommendations)
5. predicted\_recommendations

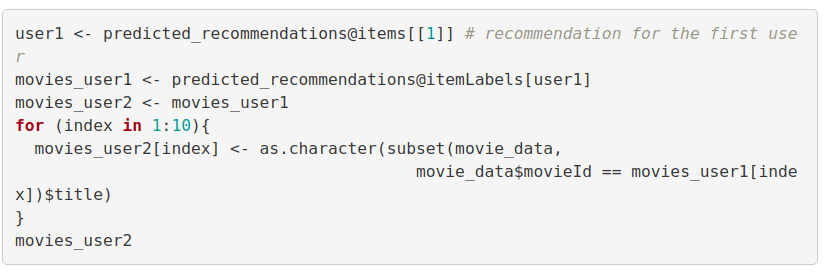
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/building-recommender-system-in-ml.png)

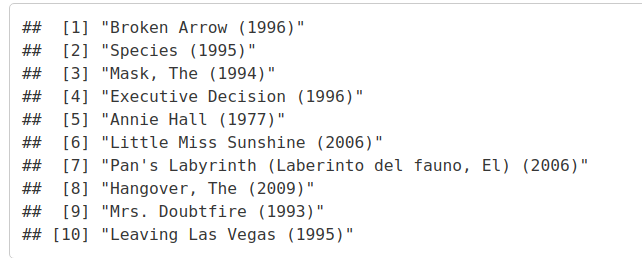
**Code:**

1. user1 <- predicted\_recommendations@items[[1]] # recommendation for the first user
2. movies\_user1 <- predicted\_recommendations@itemLabels[user1]
3. movies\_user2 <- movies\_user1
4. **for** (index in 1:10){
5. movies\_user2[index] <- as.**character**(**subset**(movie\_data,
6. movie\_data$movieId == movies\_user1[index])$title)
7. }
8. movies\_user2

**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/predict-function-to-create-recommender-system-.png)

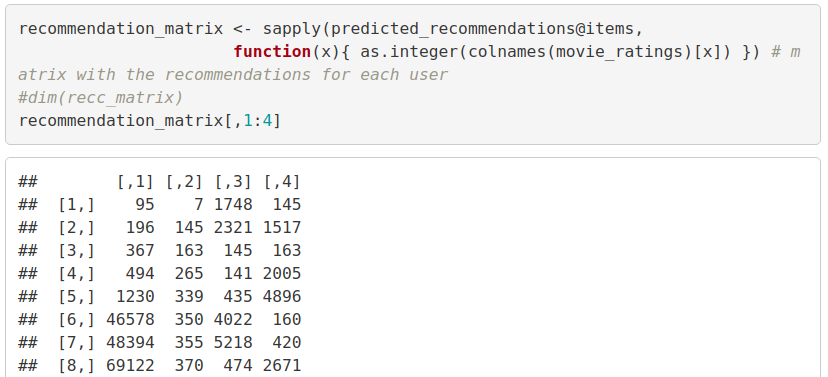
**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/implementing-recommender-system.png)

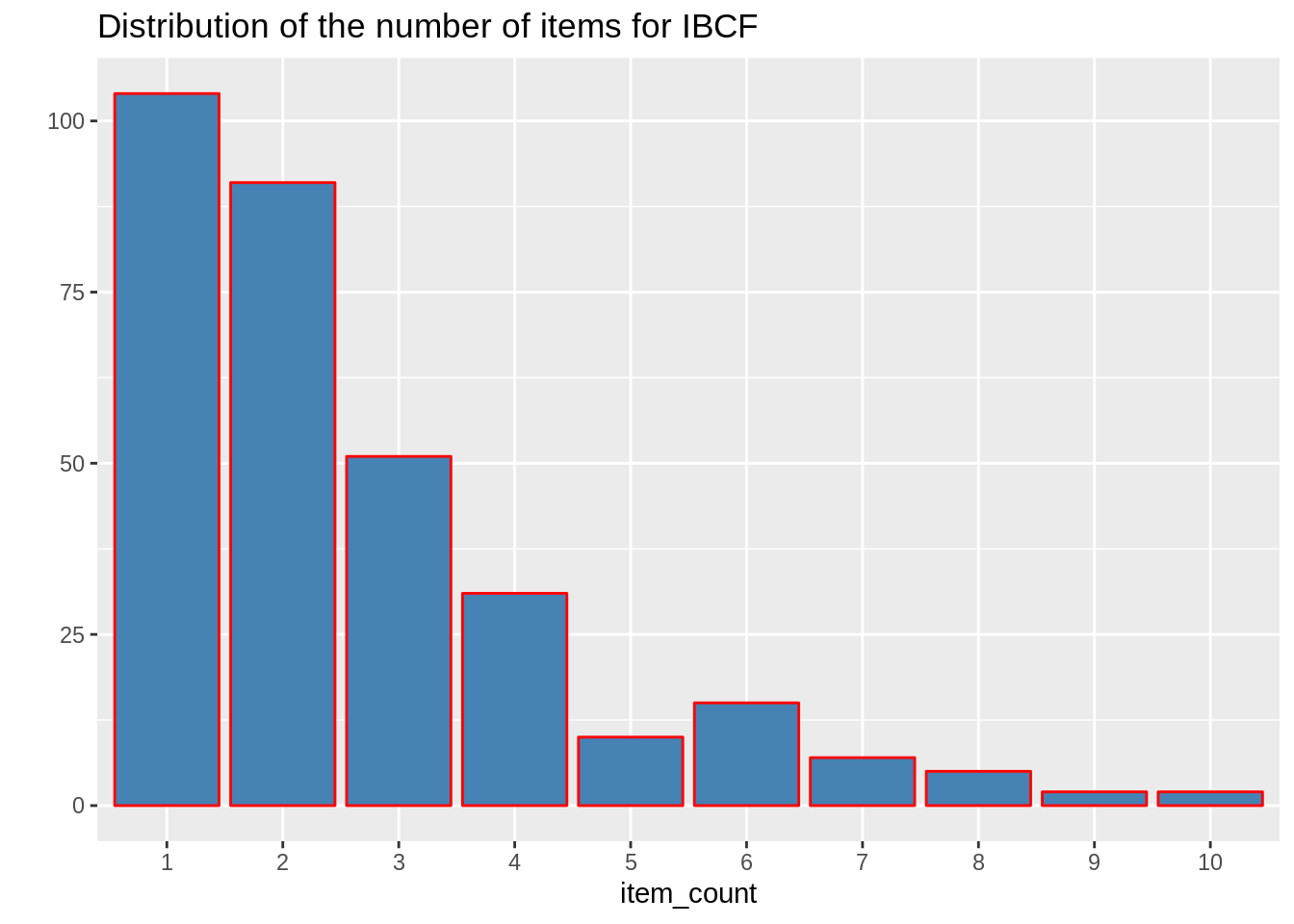
**Code:**

1. recommendation\_matrix <- **sapply**(predicted\_recommendations@items,
2. **function**(x){ as.**integer**(**colnames**(movie\_ratings)[x]) }) # matrix with the recommendations for each user
3. #dim(recc\_matrix)
4. recommendation\_matrix[,1:4]

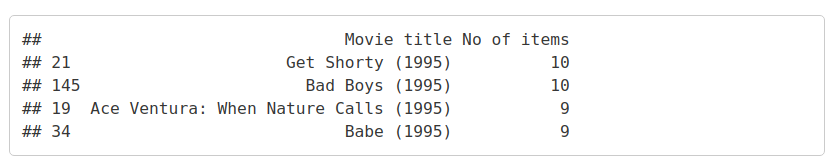
**Output Screenshot:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/distribute-visualization-in-R.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/Data-Science-Distribution-Visualisation-for-IBCF-in-R.png)

**Output:**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2019/07/R-distribute-visualization.png)

Summary

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.

Hola!! You completed this amazing Data Science Recommendation System Project. What next? Another project? Check [***Sentiment Analysis project in R***](https://data-flair.training/blogs/data-science-r-sentiment-analysis-project/)

Did you enjoy this R project? Have you built any Recommendation System in the past? Do share your experiences in the comments below.