***Detailed steps to build Movie Recommendation System***

**Download your Data set from here:**

** **

**Step 0.0**:

First of all, you need to install R in your machine. You can download R form online and install if in your computer.

For installing in windows, you can follow: [https://www.youtube.com/watch?v=NZxSA80lF1I](mailto:https://www.youtube.com/watch?v=NZxSA80lF1I)

For installing in mac, you can follow: <mailto:https://www.youtube.com/watch?v=LanBozXJjOk>

**Step 0.1**:

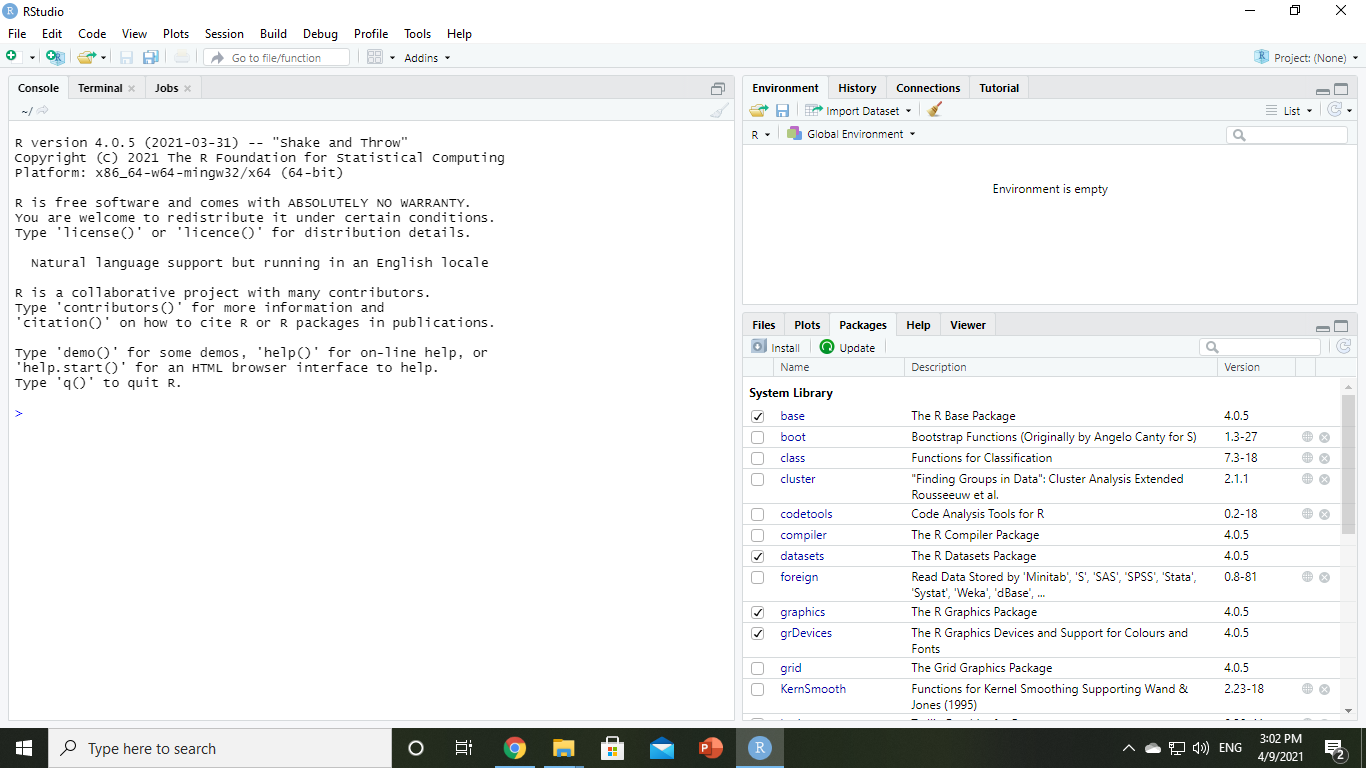
As we will code in R language, we need an IDE to do our programing. For this project we will use RStudio.

For installing in windows, you can follow: [https://www.youtube.com/watch?v=NZxSA80lF1I](mailto:https://www.youtube.com/watch?v=NZxSA80lF1I)

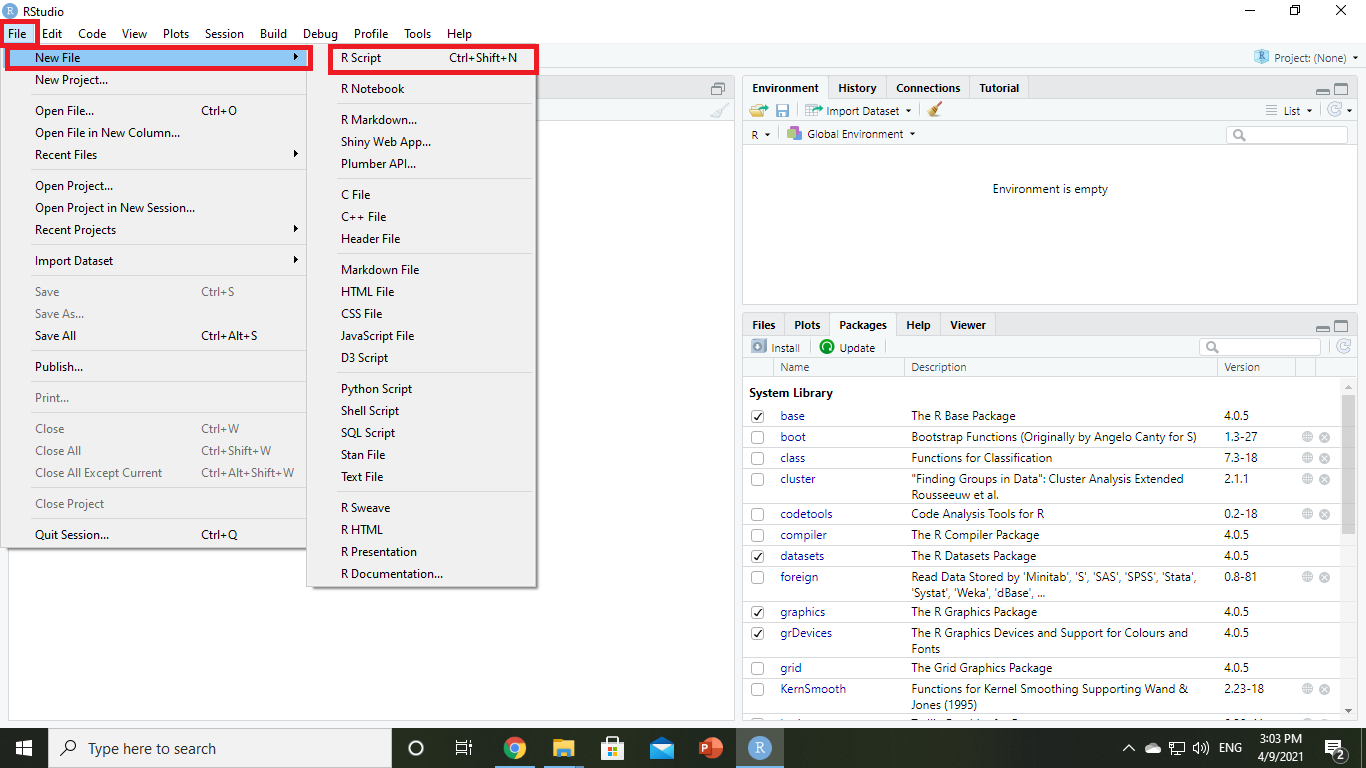
For installing in mac, you can follow: <mailto:https://www.youtube.com/watch?v=LanBozXJjOk>

**Step0.2:**

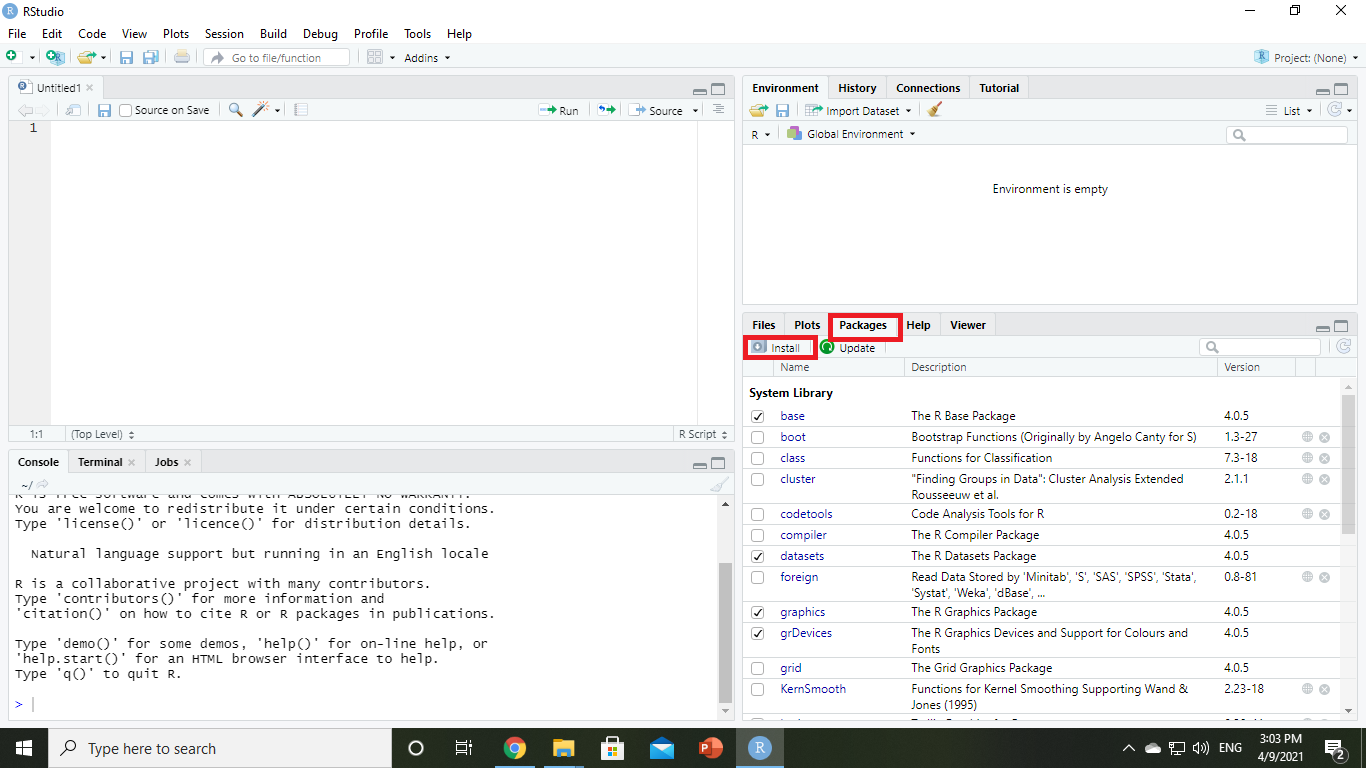
After installing Rstudio you will find the below environment



Now you need to start your R script. So that you should go like this picture from file.



**Step 1:** First of all, we will import all the necessary libraries and packages. To do so you need to install the necessary package from Rstudio. You can go to the section like the below picture

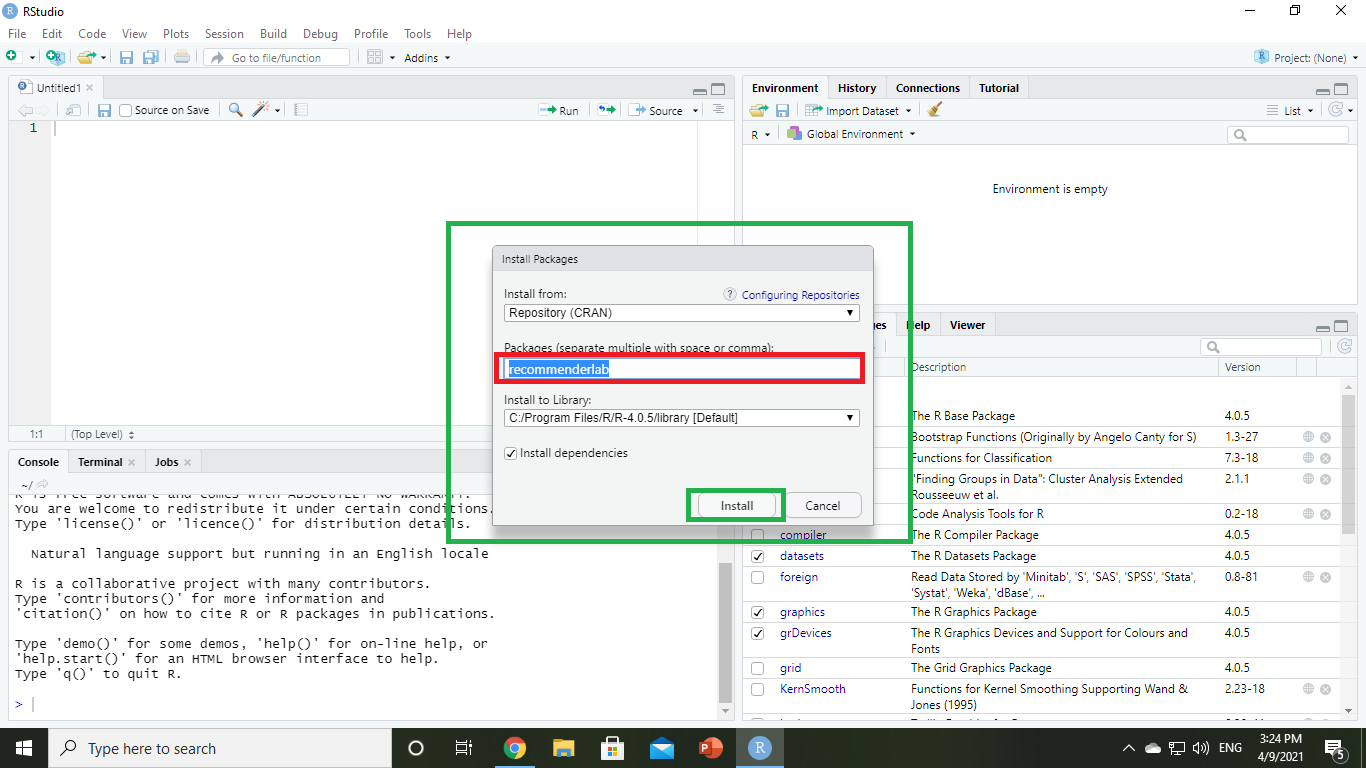


After clicking in the install option under package you will find this option to enter your library.



Here in the package section, you need to give your library name and install it. For this project we will need four libraries. Those are: **recommenderlab**, **ggplot2**, **data.table** and **reshape2**

You can install these package like below picture:

after click on install for every library, you will find some massage of installation like the below picture:



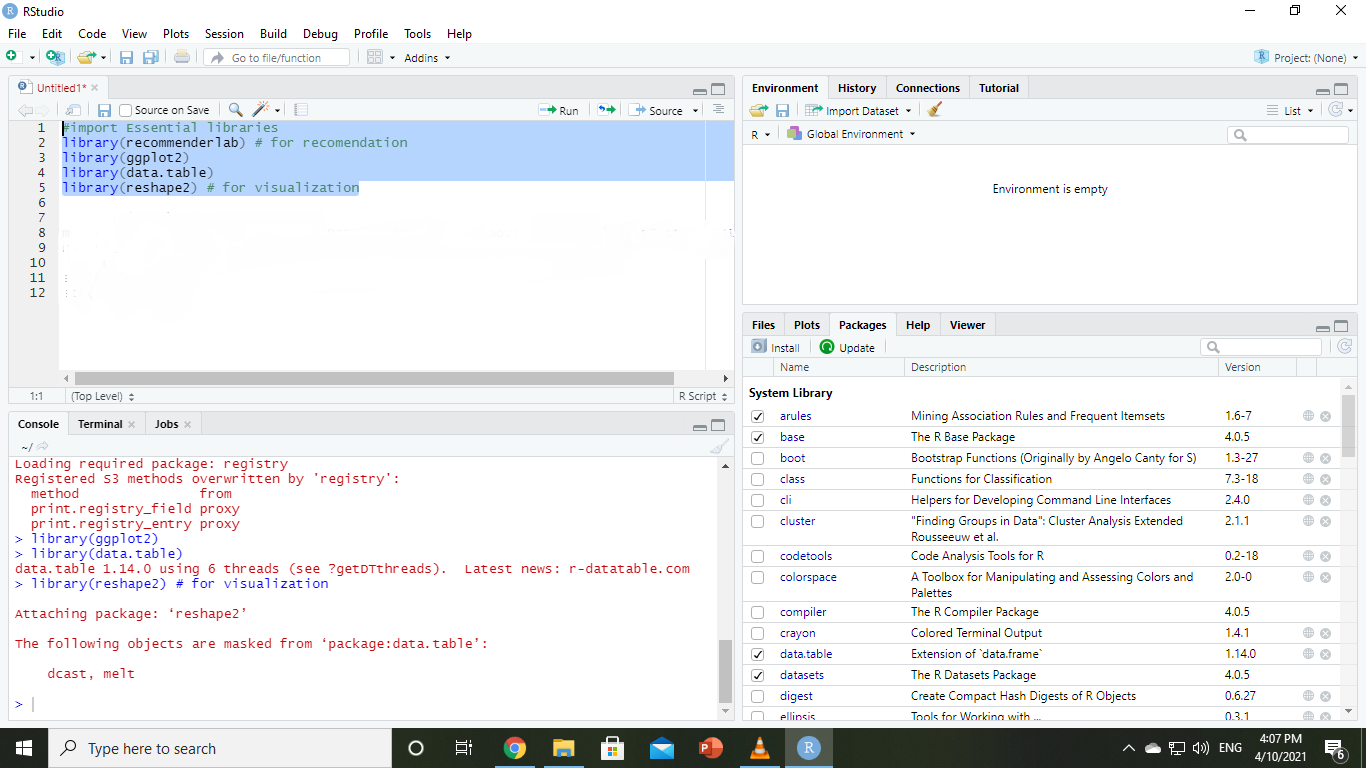
After successful completion of your library install you can start writing your code. The code should be:

***#import Essential libraries***

***library(recommenderlab) # for recomendation***

***library(ggplot2)***

***library(data.table)***

***library(reshape2) # for visualization*** 

**Step 2:** As we discuss we will use movies.csv and ratings.csv as our dataset, we will retrieve our data from movies.csv and ratings.csv into dataframes. Here we can also display information about movies if we want. (we have lots of function to overview the data e.g. summary(), str(), head() )

Here the for this part the code will be:

**# Retriving the data**

**movie\_data** <-**read.csv("H:/SEBPO/DataScience/IRMANA/movie.csv",stringsAsFactors=FALSE)**

**rating\_data** <- **read.csv("H:/SEBPO/Data Science/IRMANA/rating.csv")**

**#structure**

**str(movie\_data)**

**str(rating\_data)**

**#tabular view**

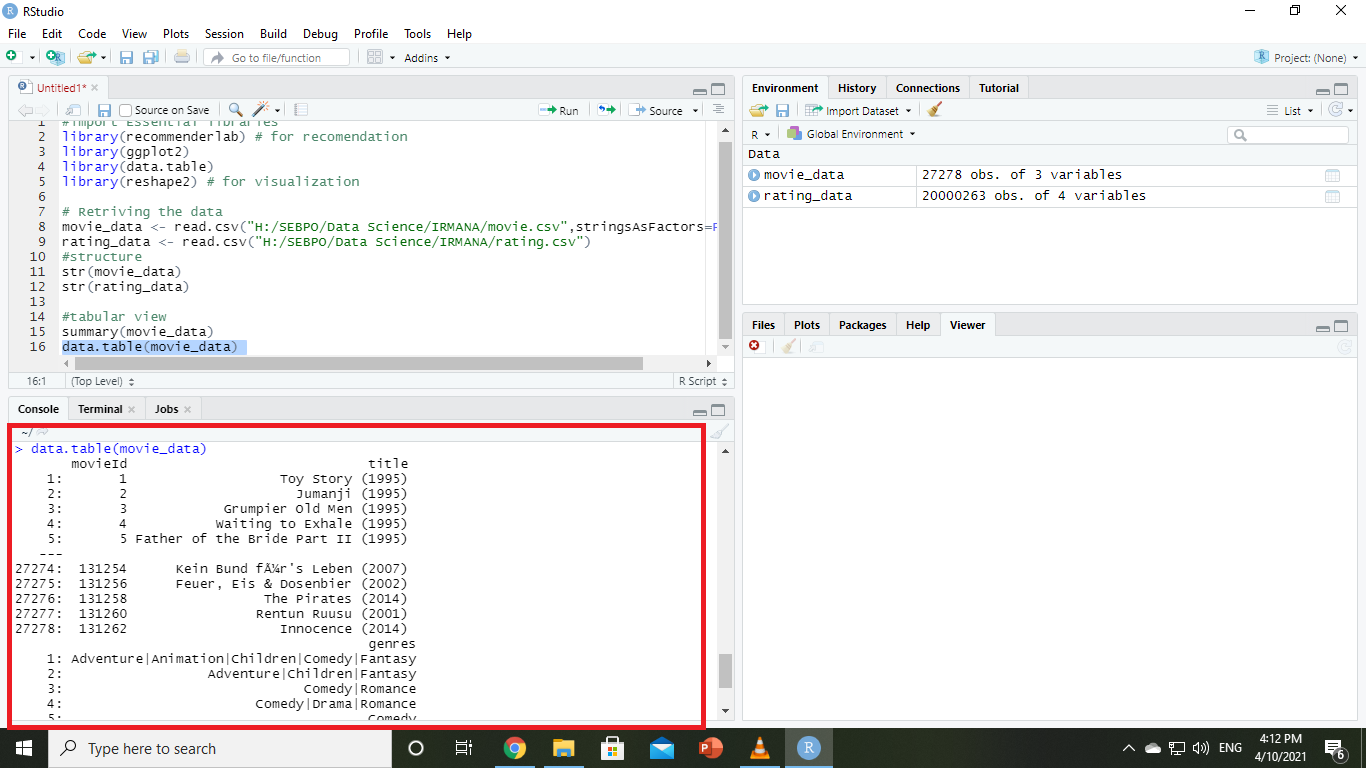
**summary(movie\_data)**

**data.table(movie\_data)**

Here inside read.csv you have to write the location where you movies.csv and ratings.csv is stored. In my case the directory was like this so I put that directory. From structure past str(movie\_data) and str(rating\_data) you will get your desired output:



To get the whole data you can get the tebular view summary(movie\_data) or data.table (movie\_data) . By this you will see the data.



**Step 3:** In the movie dataframe, we will have 3 types of values which are movieId, title, genres. We will convert the genres present in the movie dataframe into a more usable format by the users. To do so, we will first create a [**one-hot encoding**](../Users/Taufiqul/Desktop/IRMANA%20R%20project/One-hot%20encoding%20in%20R_%20three%20simple%20methods%20_%20Data%20Tricks.html) to create a matrix that comprises of corresponding genres for each of the films.

At first, we will do pre process of data. The code and output will be like this:

**Code:**

***movie\_genre <- as.data.frame(movie\_data$genres, stringsAsFactors=FALSE)***

***library(data.table)***

***movie\_genre2 <- as.data.frame(tstrsplit(movie\_genre[,1], '[|]',***

***type.convert=TRUE),***

***stringsAsFactors=FALSE) #DataFlair***

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

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

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

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

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

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

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

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

***for (index in 1:nrow(movie\_genre2)) {***

***for (col in 1:ncol(movie\_genre2)) {***

***gen\_col = which(genre\_mat1[1,] == movie\_genre2[index,col]) #Author DataFlair***

***genre\_mat1[index+1,gen\_col] <- 1***

***}***

***}***

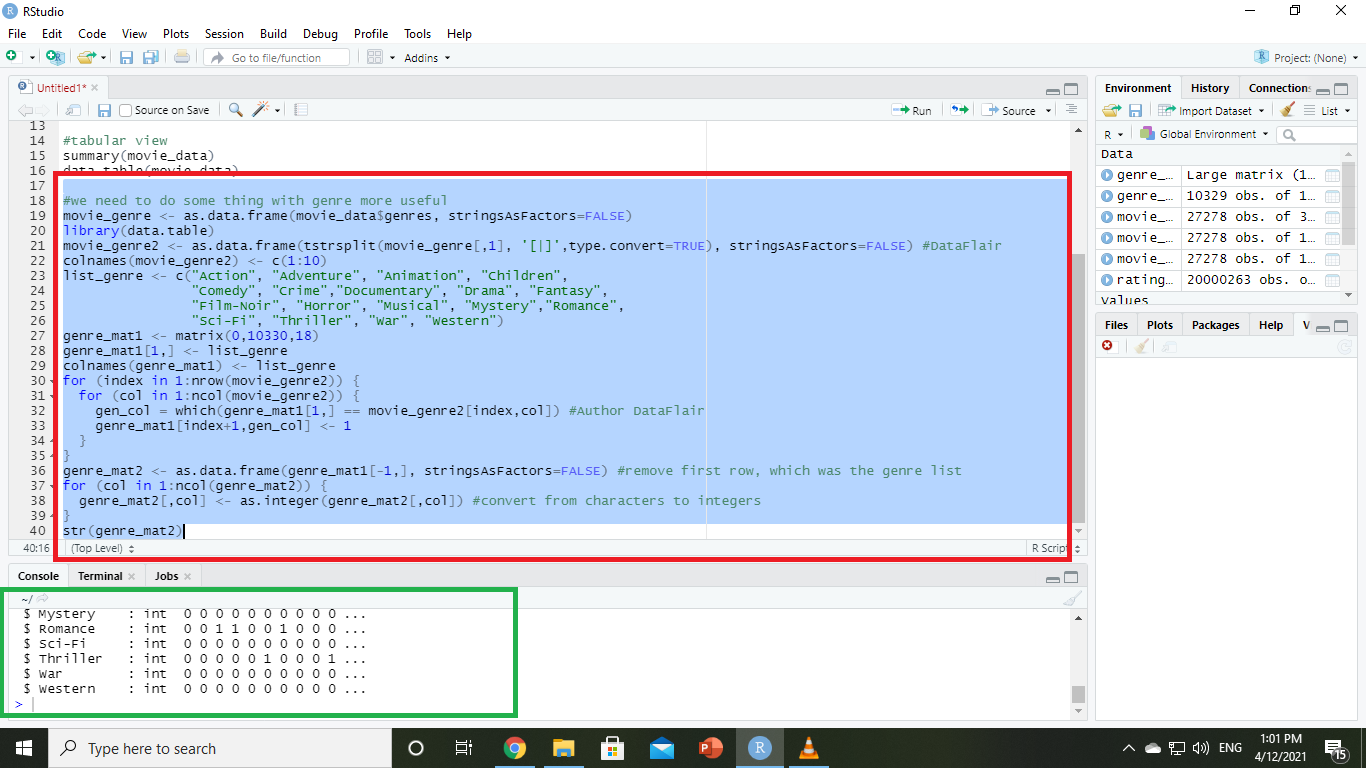
***genre\_mat2 <- as.data.frame(genre\_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list***

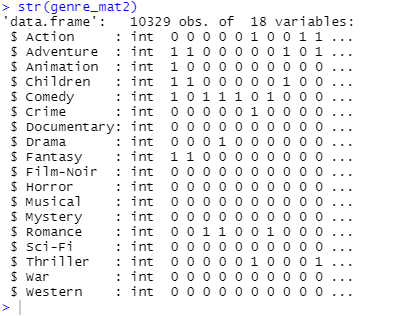
***for (col in 1:ncol(genre\_mat2)) {***

***genre\_mat2[,col] <- as.integer(genre\_mat2[,col]) #convert from characters to integers***

***}***

***str(genre\_mat2)***

******

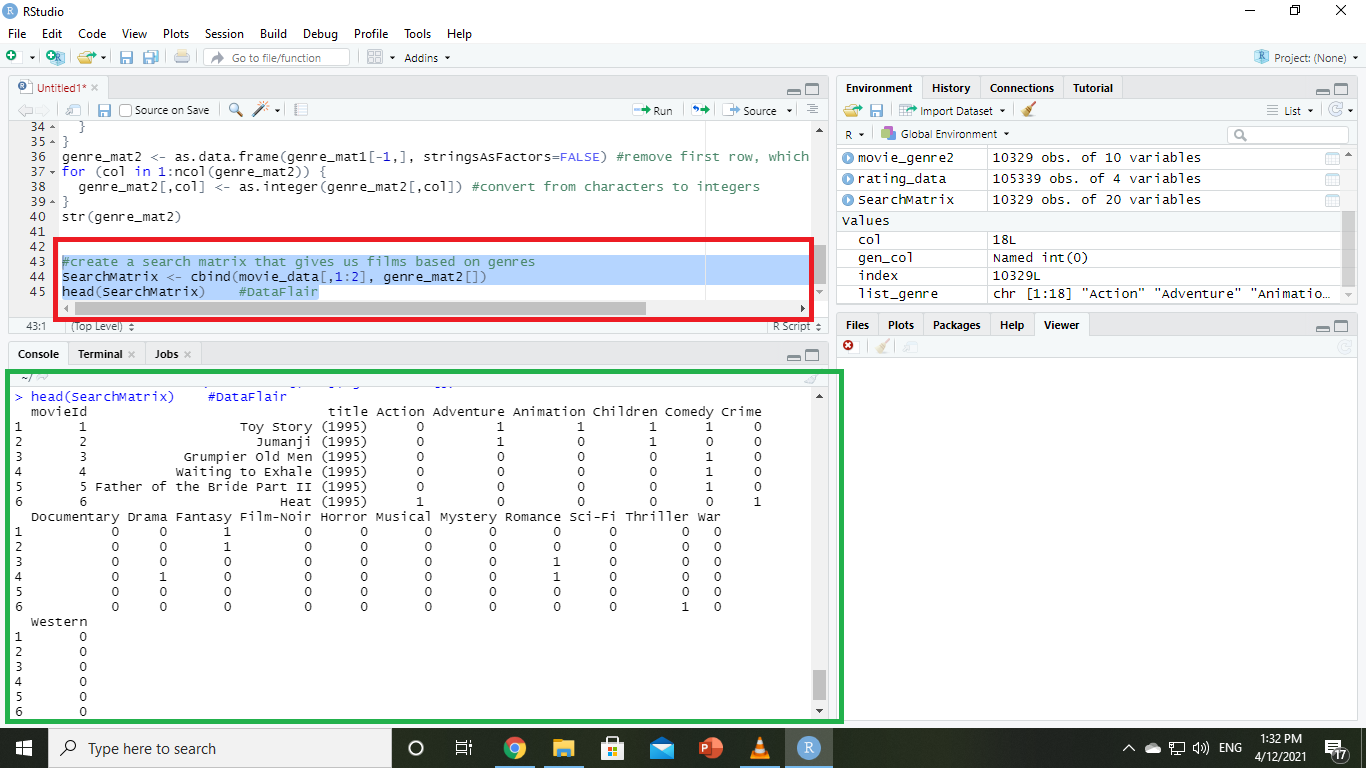
******

Now we will do the search Matrix:

**Code:**

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

***head(SearchMatrix)***

******

Many movies have several genres. So, lets create a sparse matrix for recommendation. we will use item based collaborative filtering

**Code:**

***# Many movies have several gnere. So, lets create a sparse matrix for recomendation***

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

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

***#Convert rating matrix into a recommenderlab sparse matrix***

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

***ratingMatrix***

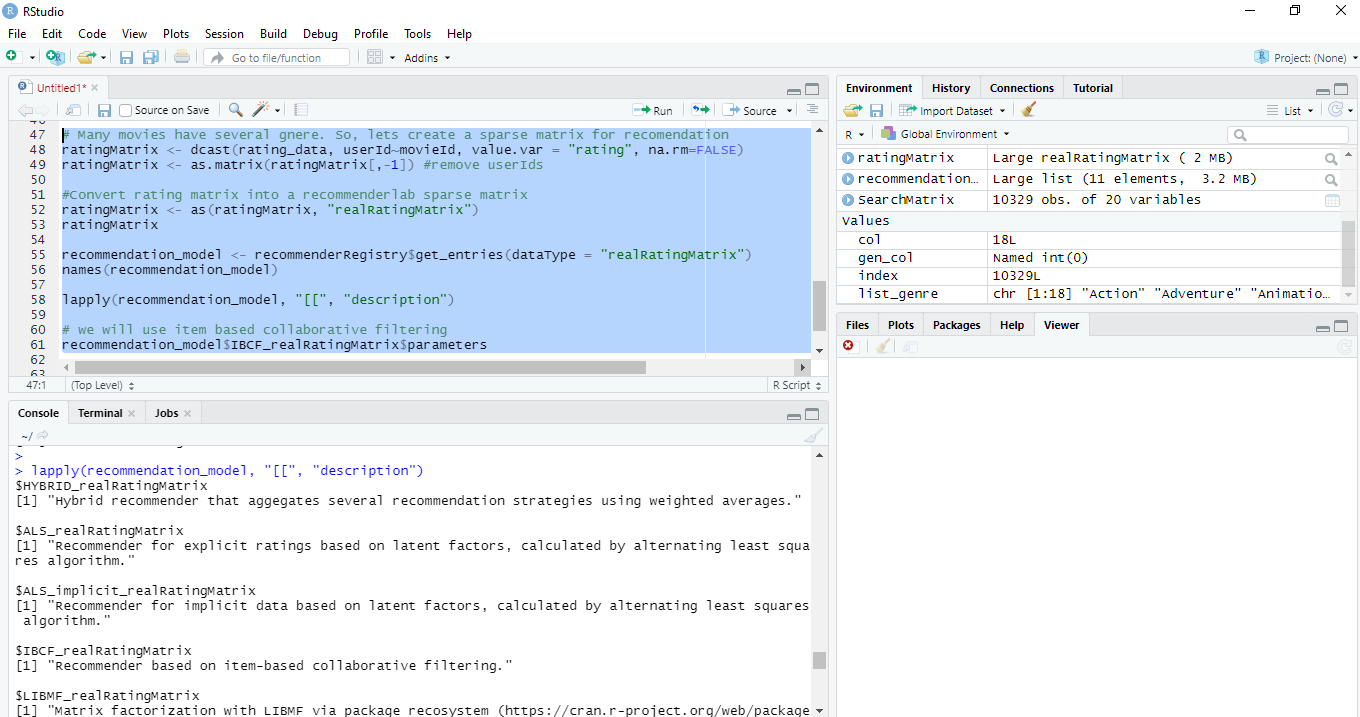
***recommendation\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")***

***names(recommendation\_model)***

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

***# we will use item based collaborative filtering***

***recommendation\_model$IBCF\_realRatingMatrix$parameters***

******

******

**Step 4:** After that, we will create a ‘search matrix’ that will allow us to perform an easy search of the films by specifying the genre present in our list. There could be movies that fall into several genres like one movie can be comedy, action, and fantasy.

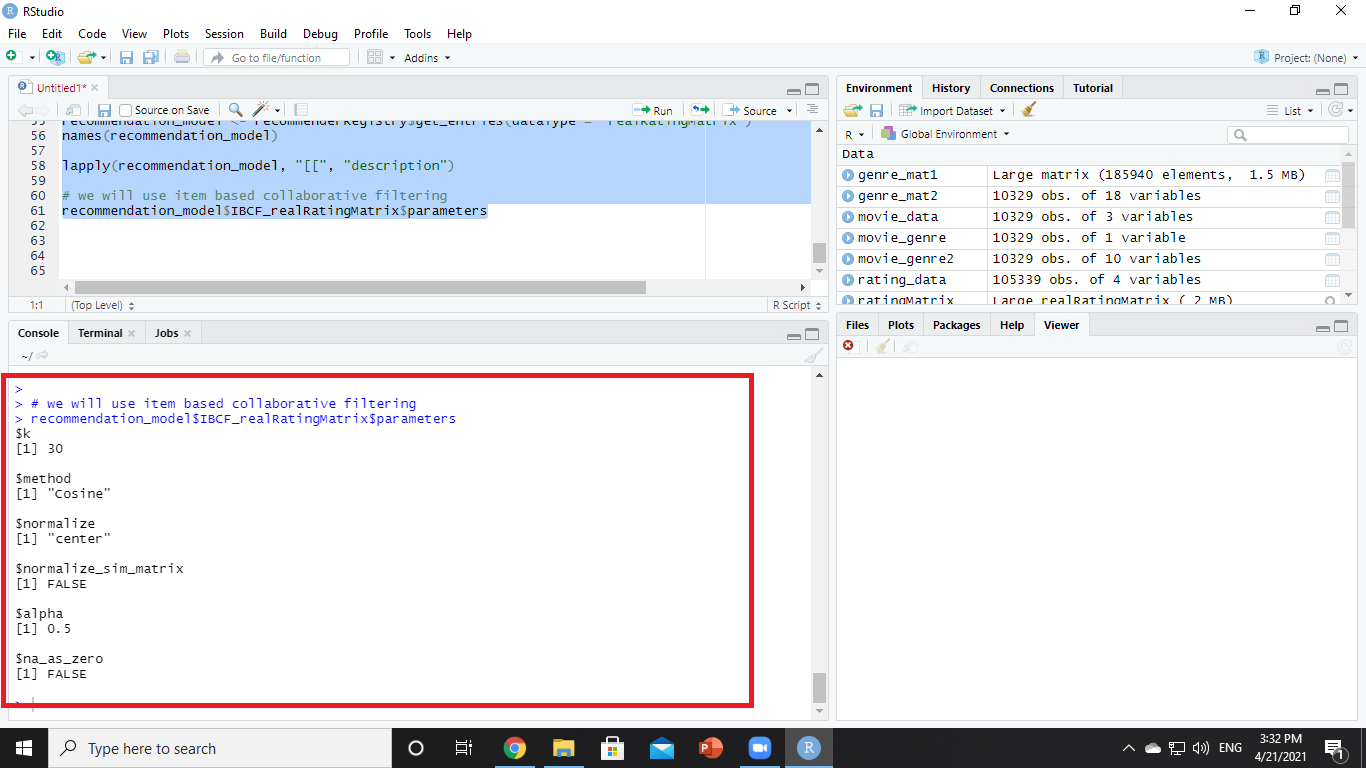
**Step 5:** For our movie recommendation system to make sense of our ratings through recommenderlabs (it is a keyword of R), we have to convert our matrix into a [**sparse matrix one**](../Users/Taufiqul/Desktop/IRMANA%20R%20project/Sparse%20Matrix%20Construction%20And%20Use%20In%20R%20-%20GormAnalysis.html).

**Step 6:** We will implement a single model in our R project – [**Item Based Collaborative Filtering**](../Users/Taufiqul/Desktop/IRMANA%20R%20project/Item-based%20collaborative%20filtering%20recommendation%20algorithms%20_%20Proceedings%20of%20the%2010th%20international%20conference%20on%20World%20Wide%20Web.html).

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

Code:

***recommendation\_model$IBCF\_realRatingMatrix$parameters***



Now we will Exploring Similar Data.

**Step 7:** Now we will need to explore similar data. So here we will observe suggesting movies to the users that are based on collecting preferences from many other users. For example, if 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 jacquard. (These are functions of R)

**Code:**

***#lets check similarity***

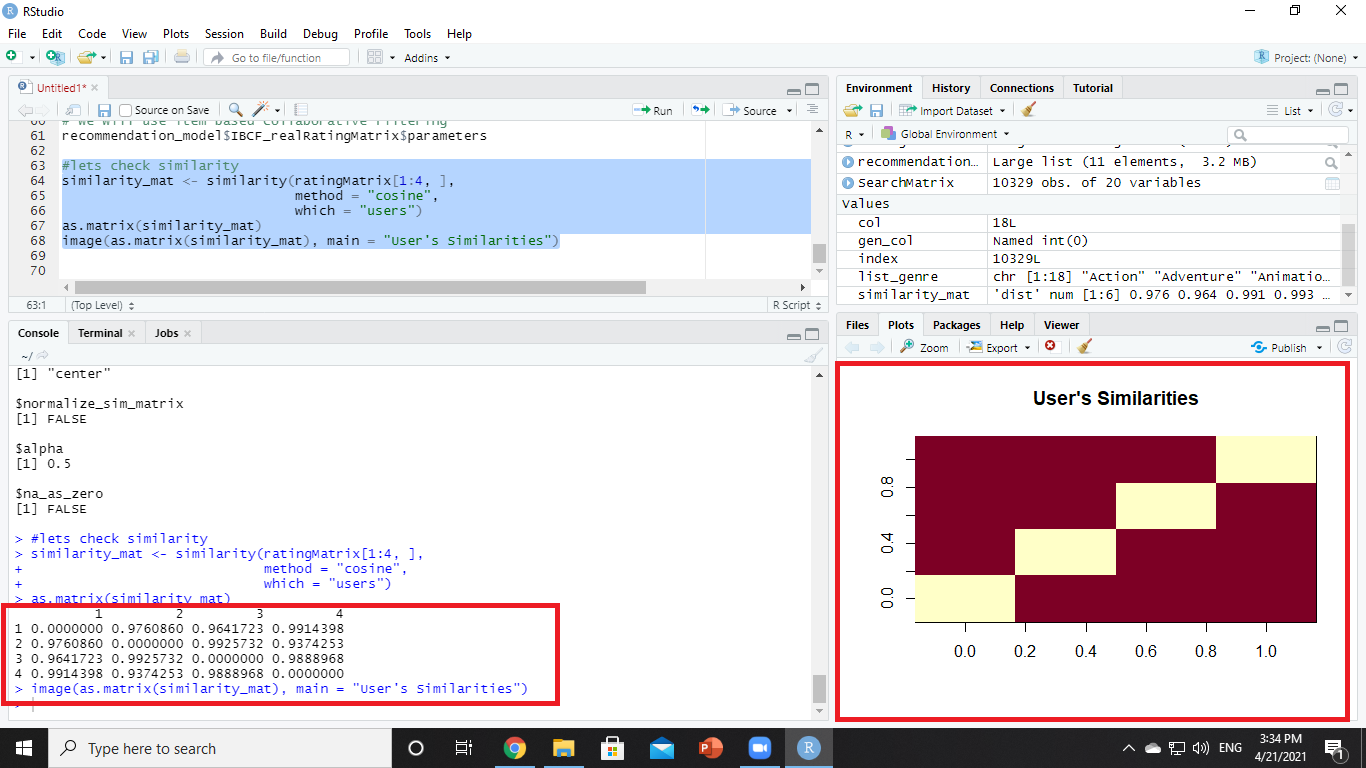
***similarity\_mat <- similarity(ratingMatrix[1:4, ],***

***method = "cosine",***

***which = "users")***

***as.matrix(similarity\_mat)***

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



**Step 8:** From the matrix, each row and column will represent a user and each cell in this matrix represents the similarity that is shared between the two users. We have taken four users and each cell in this matrix represents the similarity that is shared between the two users.

Code:

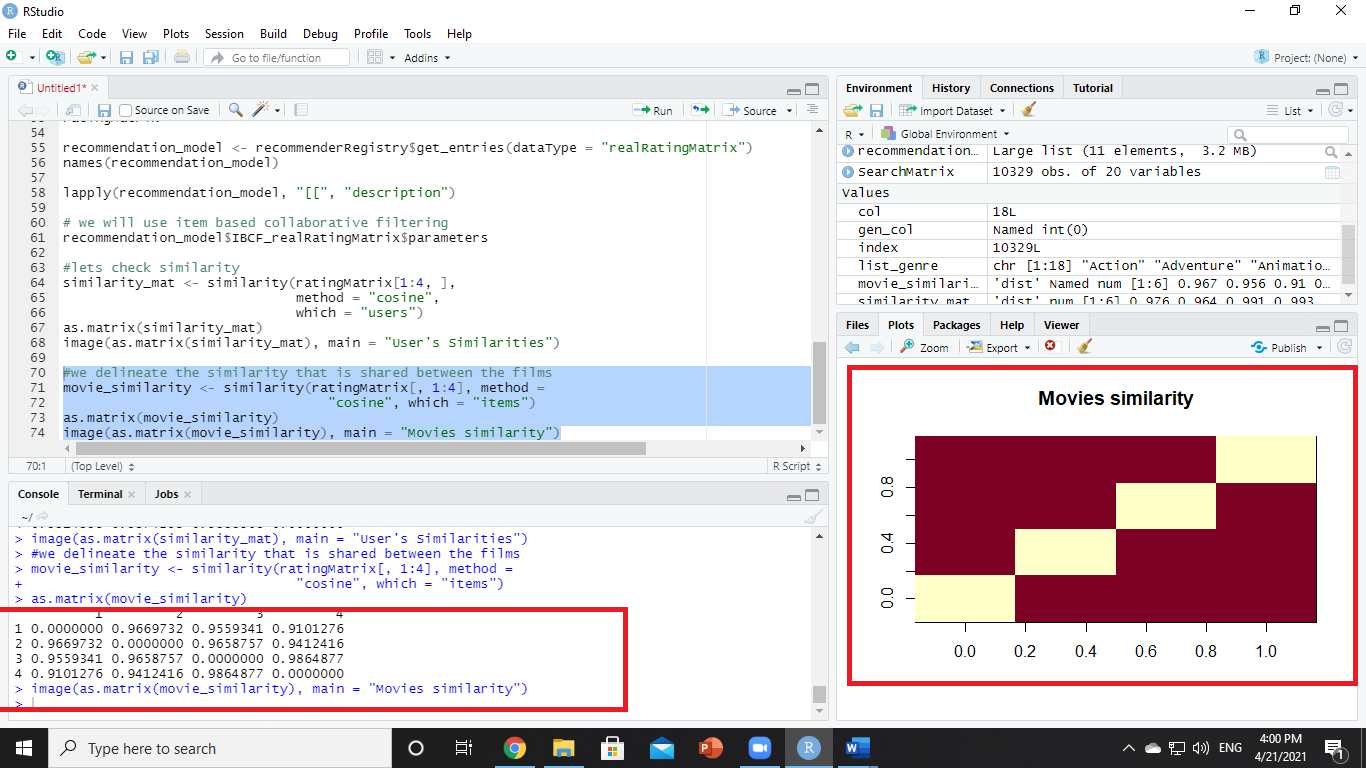
***#we delineate the similarity that is shared between the films***

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

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

***as.matrix(movie\_similarity)***

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



**Step 9:** Then, we delineate the similarity that is shared between the films.

**Code:**

***#we delineate the similarity that is shared between the films***

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

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

***as.matrix(movie\_similarity)***

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

**Step 10:** After that, we will create a table of ratings that will display the most unique ratings.

**Code:**

***#Let us now extract the most unique ratings***

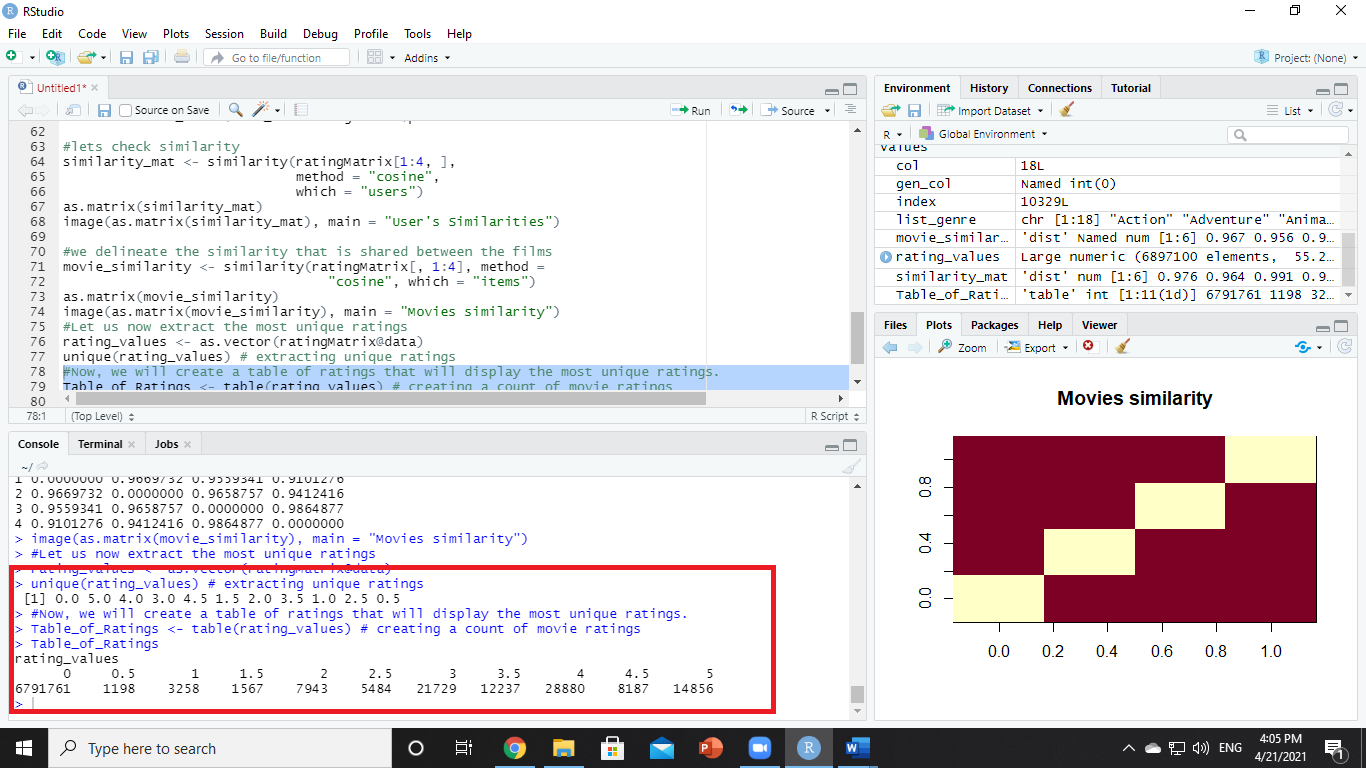
***rating\_values <- as.vector(ratingMatrix@data)***

***unique(rating\_values) # extracting unique ratings***

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

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

***Table\_of\_Ratings***



**Step 11:** Now we will **analyze the most viewed movie visualization** from our dataset. We will first count the number of views in a film and then organize them in a table.

Code:

***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***

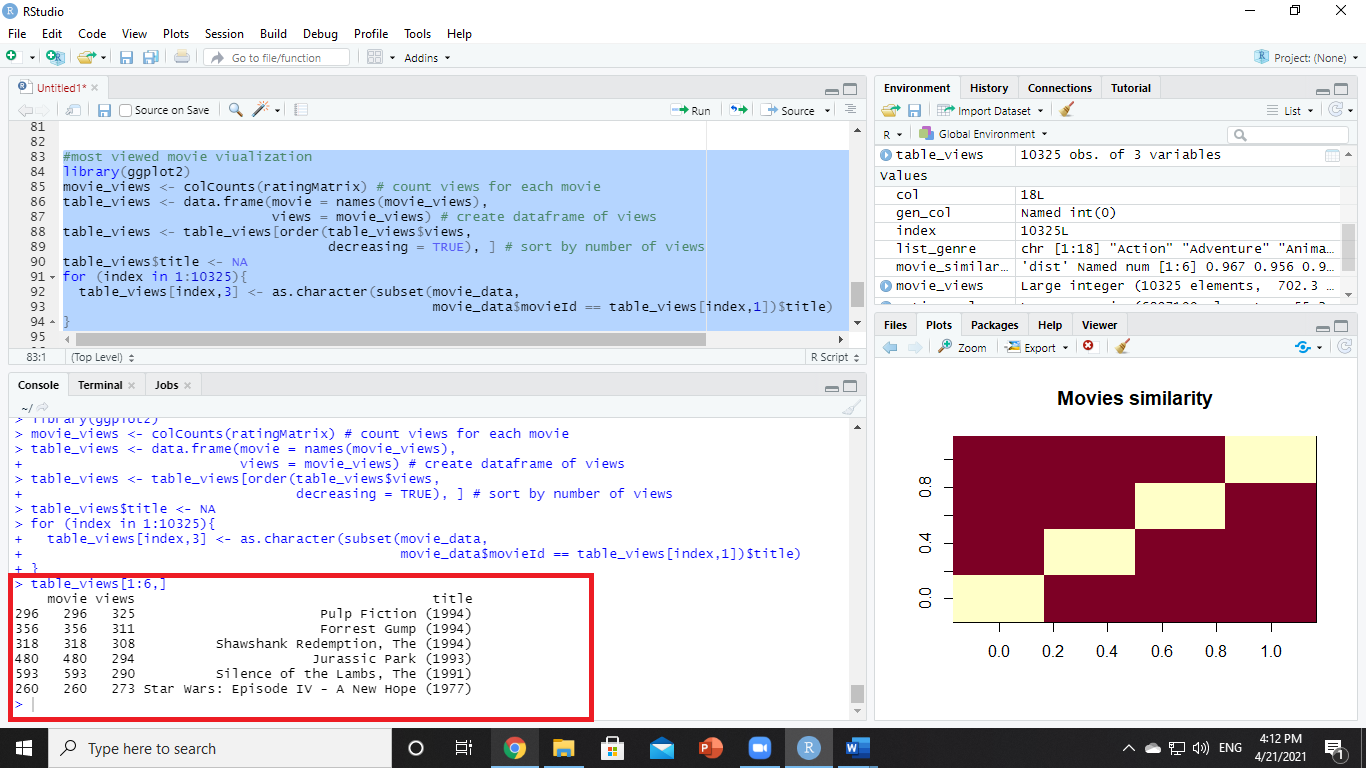
***for (index in 1:10325){***

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

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

***}***

***table\_views[1:6,]***



**Step 12:** Then, 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 chart***

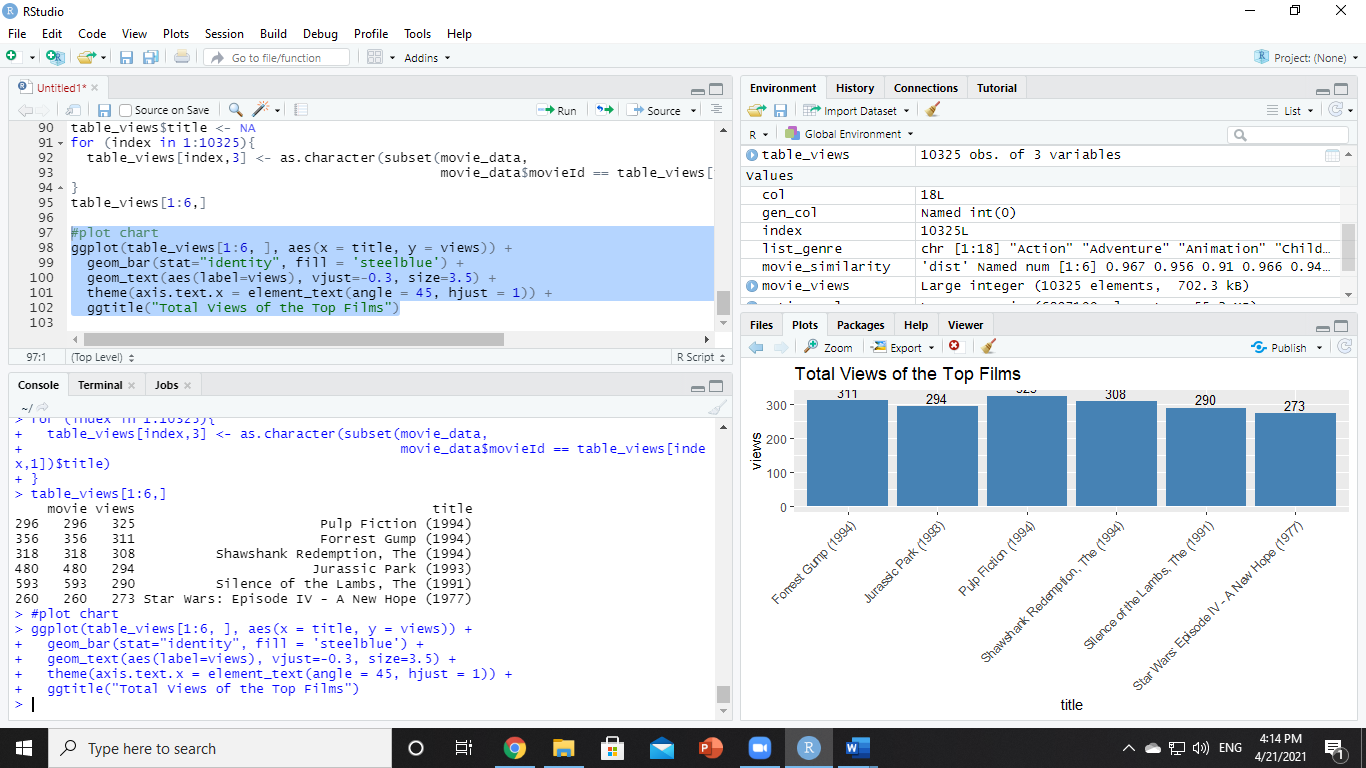
***ggplot(table\_views[1:6, ], aes(x = title, y = views)) +***

***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")***

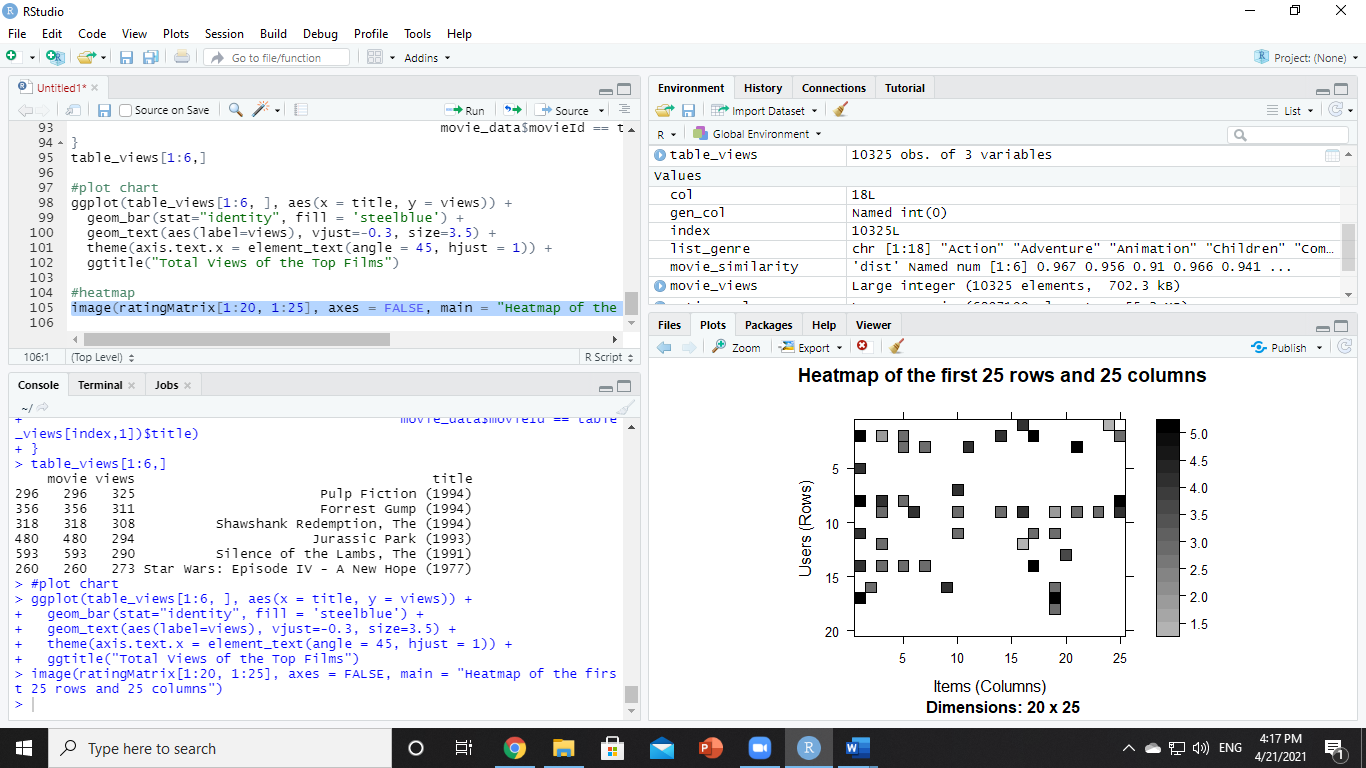


**Step 13:** After we can visualize a [**heatmap**](../Users/Taufiqul/Desktop/IRMANA%20R%20project/Heatmap%20in%20R_%20Static%20and%20Interactive%20Visualization%20-%20Datanovia.html) of the movie ratings.

Code :

***#heatmap***

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



**Step 14:** After that, we will visualize the **distribution of the average ratings per user**.

**Code:**

***#Performing Data Preparation***

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

***colCounts(ratingMatrix) > 50]***

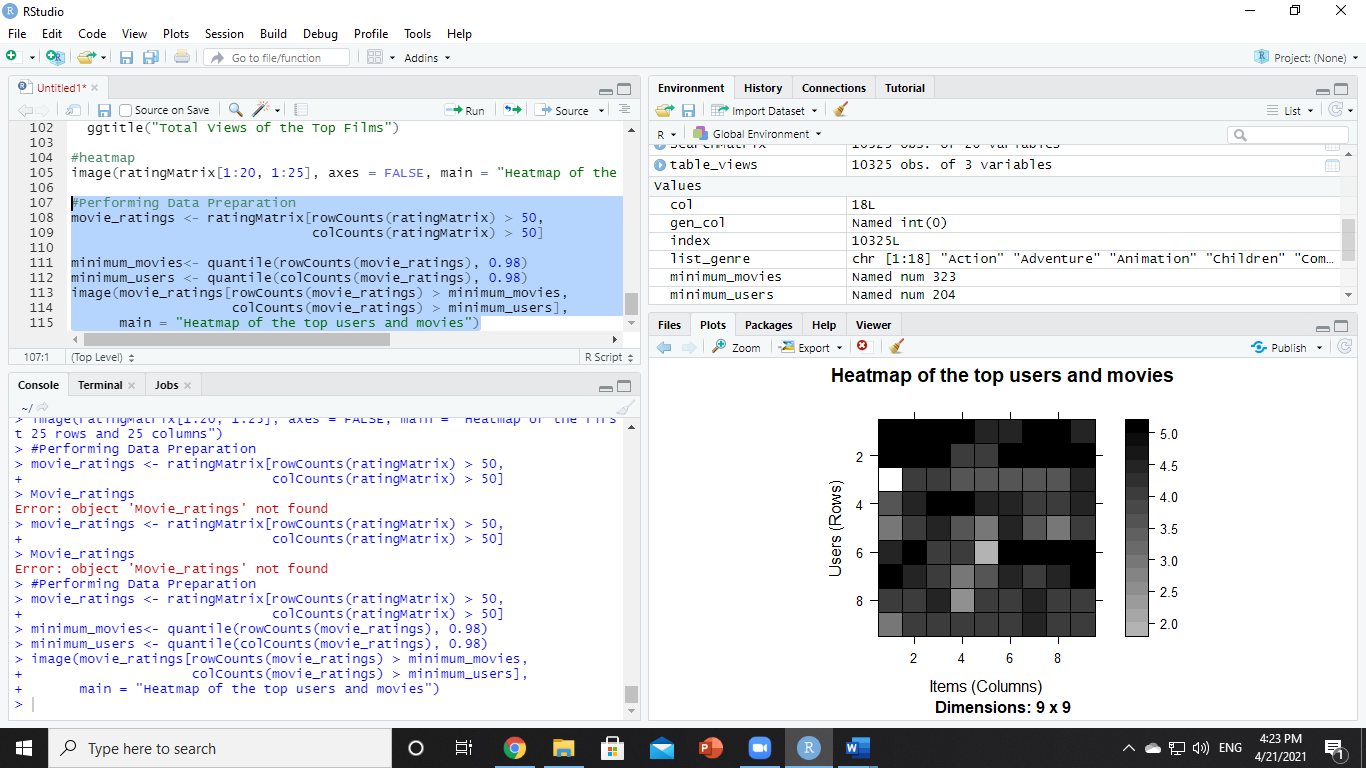
***minimum\_movies<- quantile(rowCounts(movie\_ratings), 0.98)***

***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")***

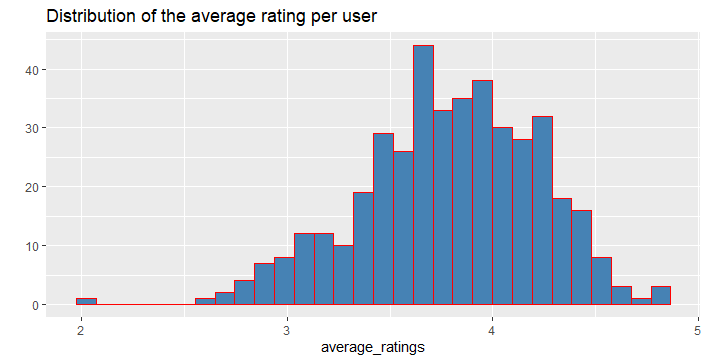


Code:

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

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

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



**Step 15:** We will conduct data preparation in the following three steps – Selecting useful data, Normalizing data, Binarizing the data.

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:**

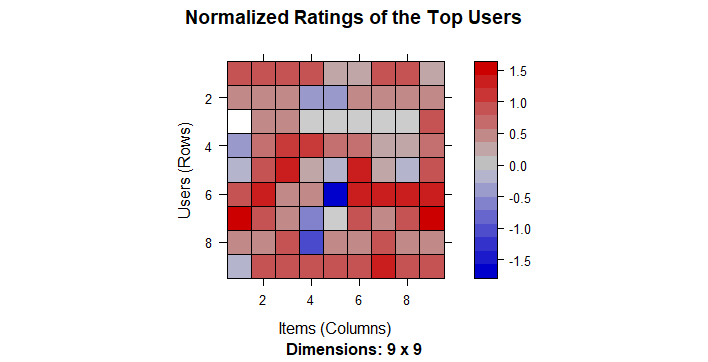
***normalized\_ratings <- normalize(movie\_ratings)***

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

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

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

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



**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:

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

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

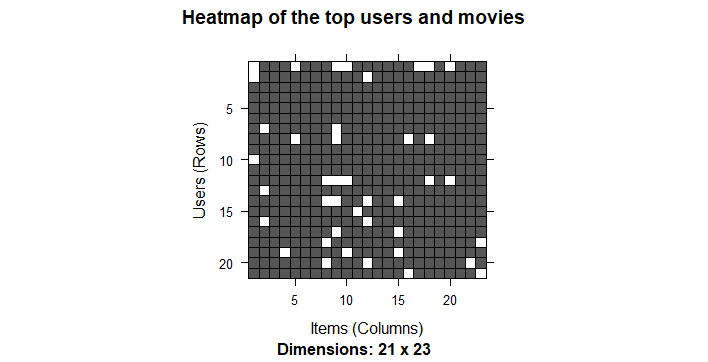
***#movies\_watched <- binarize(movie\_ratings, minRating = 1)***

***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")***



**Step 16:** We will now explore the various parameters of our Item Based Collaborative Filter. These parameters are default in nature.

Code:

***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, ]***

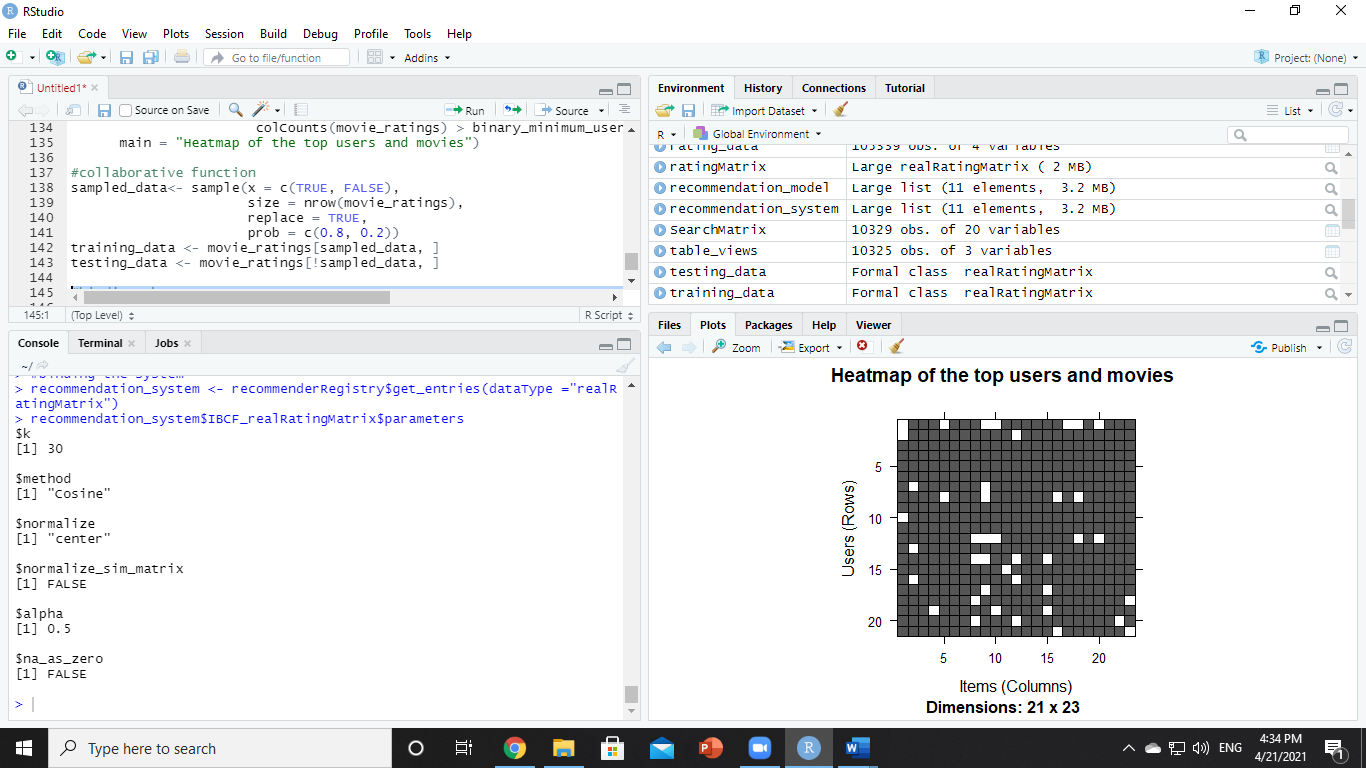
**Step 17:** Using the getModel() function, we will retrieve the recommendation model. We will then find the class and dimensions of our similarity matrix that is contained within model info.

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 :

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

***recommendation\_system$IBCF\_realRatingMatrix$parameters***



**Code:**

***#binding the system***

***recommen\_model <- Recommender(data = training\_data,***

***method = "IBCF",***

***parameter = list(k = 30))***

***recommen\_model***

***class(recommen\_model)***

***model\_info <- getModel(recommen\_model)***

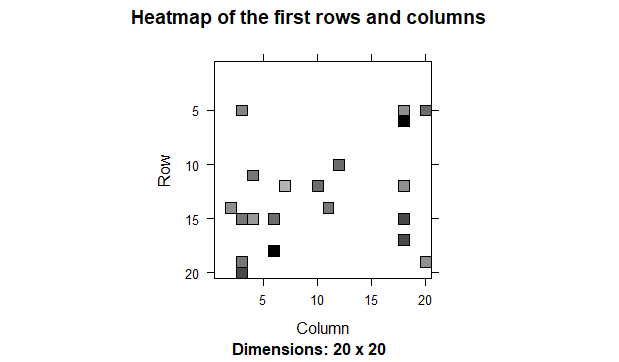
***class(model\_info$sim)***

***dim(model\_info$sim)***

***top\_items <- 20***

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

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



**Step 18:** Finally, we will generate a heatmap, that will contain the top items and visualize the similarity shared between them.

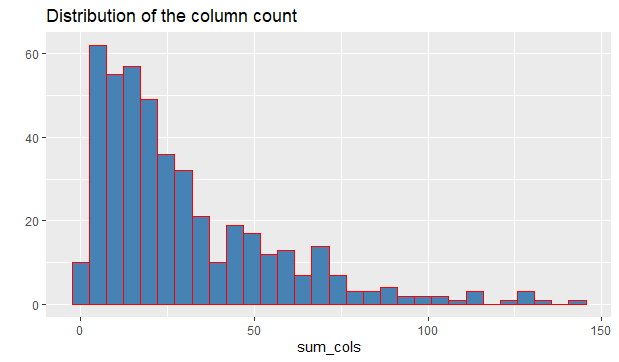
Code:

***sum\_rows <- rowSums(model\_info$sim > 0)***

***table(sum\_rows)***

***sum\_cols <- colSums(model\_info$sim > 0)***

***qplot(sum\_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")***



**Step 19:** In the next step of project, we will carry out the sum of rows and columns with the similarity of the objects. We will visualize the sum of columns through a distribution

**Code:**

***top\_recommendations <- 10 # the number of items to recommend to each user***

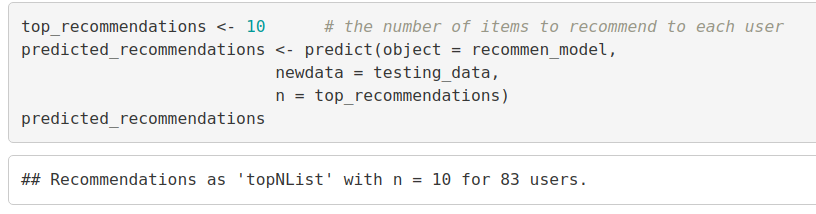
***predicted\_recommendations <- predict(object = recommen\_model,***

***newdata = testing\_data,***

***n = top\_recommendations)***

***predicted\_recommendations***

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:

***user1 <- predicted\_recommendations@items[[1]] # recommendation for the first user***

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

***movies\_user2 <- movies\_user1***

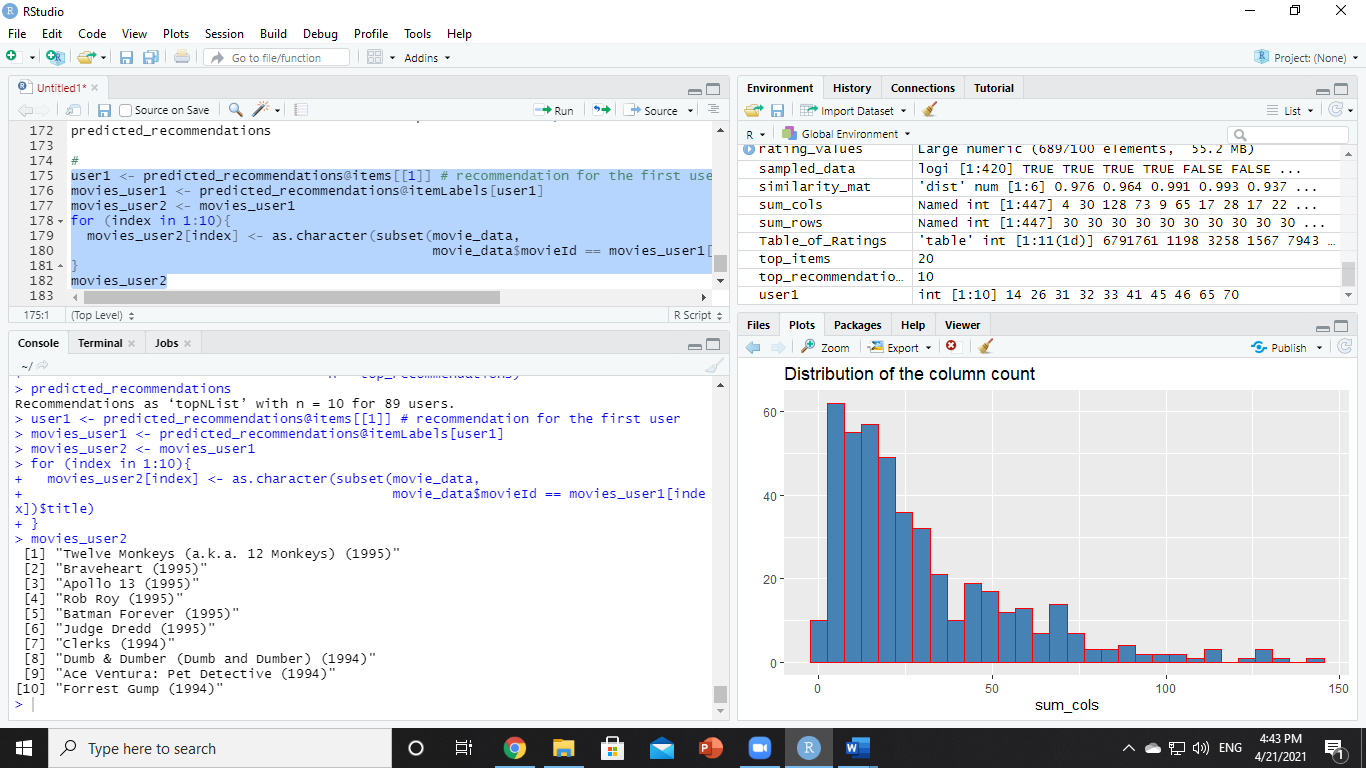
***for (index in 1:10){***

***movies\_user2[index] <- as.character(subset(movie\_data,***

***movie\_data$movieId == movies\_user1[index])$title)***

***}***

***movies\_user2***



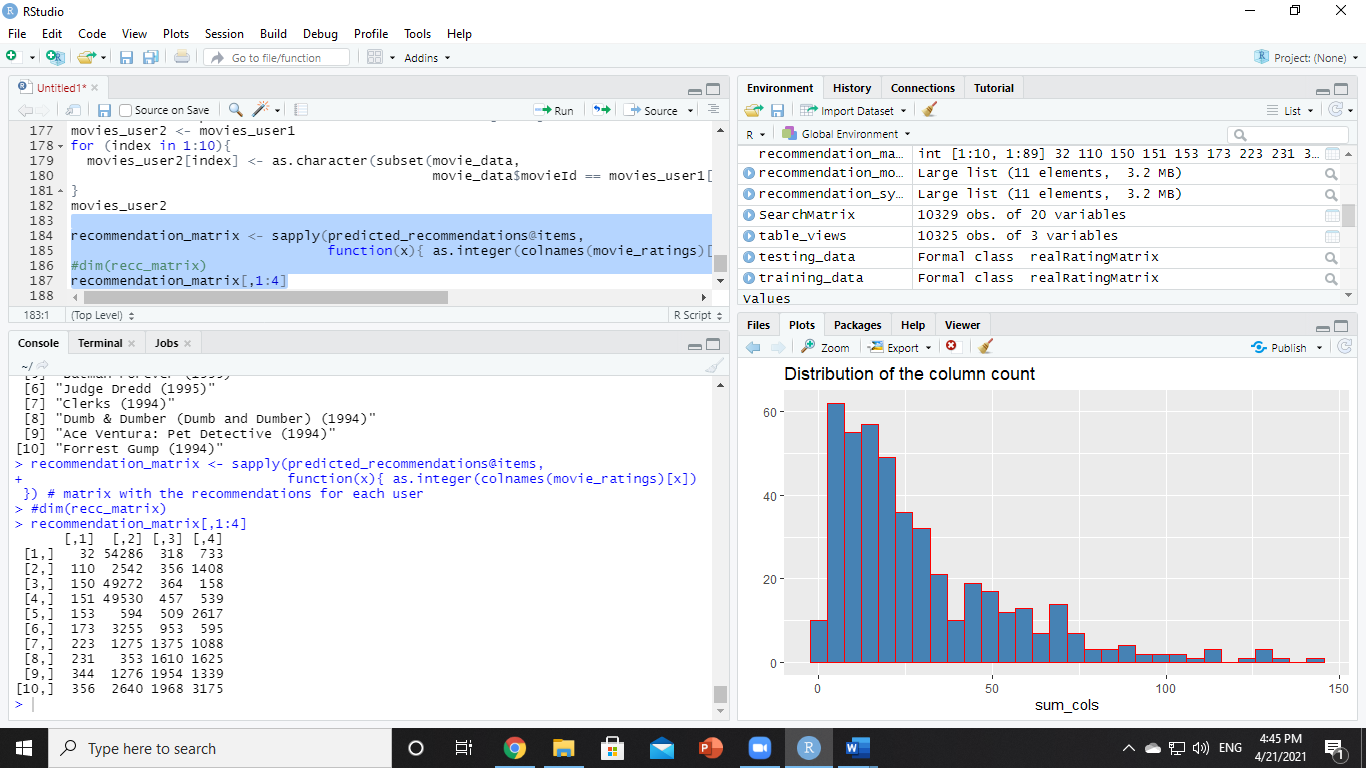
Code:

***recommendation\_matrix <- sapply(predicted\_recommendations@items,***

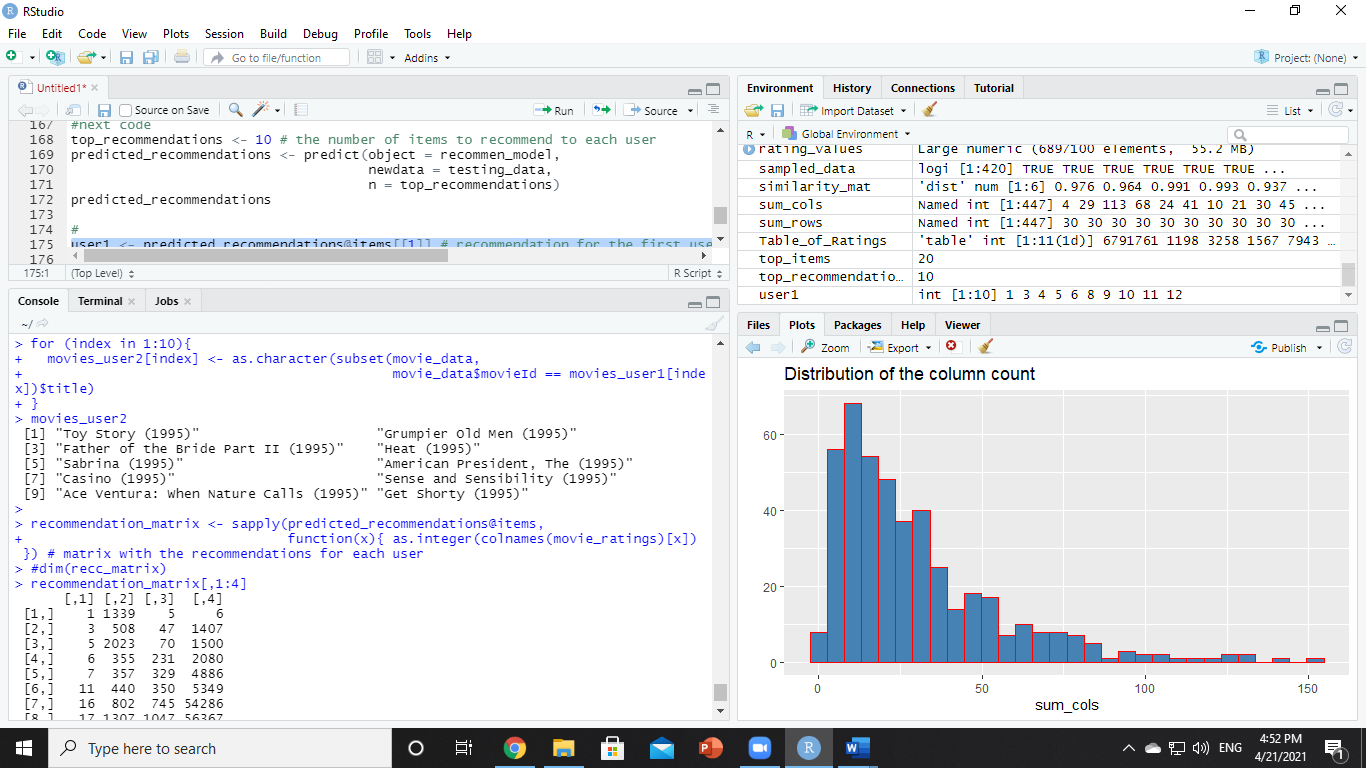
***function(x){ as.integer(colnames(movie\_ratings)[x]) }) # matrix with the recommendations for each user***

***#dim(recc\_matrix)***

***recommendation\_matrix[,1:4]***



Sample recommendation:



**The code is here. You can download it from here also:**

