**BIG DATA PROJECT**

**ON**

**SOCIAL NETWORK BASED RECOMMENDED SYSTEM**

**- BY**

**SINDHOORI MUNAGALA**

**Course objective:**

The Project allows us to integrate our knowledge and skills obtained throughout the program. Emphasis is placed on Big Data practices, contributing in a team environment, and following the data science workflow. Upon completion, we can take a large real-world data set, improve its data quality, and use the data set to develop and improve the accuracy of a statistical-driven model that supports a real-world process, operational model, or business model.

**Title of the project:**

SOCIAL NETWORK BASED RECOMMENDED SYSTEM

**Group information:**

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Rohith Namani | C0717525 |
| Sindhoori Munagala | C0718158 |
| Sasirekha Kannan | C0716363 |

**Project Planning and Proposal:**

In this project we present a new paradigm of recommender systems which can utilize information in social networks, including user preferences, item's general acceptance, and influence from social friends. A probabilistic model is developed to make personalized recommendations from such information.

We have taken the scenario of Movie recommendation systems from social media data and predicting the Content based recommendations and User based recommendation systems based on that.

**Description of the problem:**

Collaborative filtering has many problems. Collaborative filtering has the data sparsity problem and the cold-start problem. It is difficult for recommender systems to accurately measure user similarities from those limited number of reviews. A related problem is the cold-start problem. Therefore, the system cannot accurately interpret this user's preference. First, in terms of the prediction accuracy, the additional information about users and their friends obtained from social networks improves the understanding of user behaviors and ratings. Therefore, we can model and interpret user preferences more precisely, and thus improve the prediction accuracy. Second, with friend information in social networks, it is no longer necessary to find similar users by measuring their rating similarity, because the fact that two people are friends already indicates that they have things in common. Thus, the data sparsity problem can be alleviated. Thus, the data sparsity problem can be alleviated.

**Project Deliverables:**

Social influence plays an important role in product marketing. However, it has rarely been considered in traditional recommender systems. In this project we demonstrate a new paradigm of recommender systems which can utilize information in social networks, including user preferences, item's general acceptance, and influence from social friends. A probabilistic model is developed to make personalized recommendations from such information. We extract data from a real online social network, and our analysis of this large dataset reveals that friends tend to select the same items and give similar ratings. Experimental results on this dataset show that our proposed system not only improves the prediction accuracy of recommender systems but also remedies the data sparsity and cold-start issues inherent in collaborative filtering.

**General Strategies:**

We followed 2 strategies

1. Content based filtering approach
2. User based filtering approach

**1. Content based filtering approach:**

Like the name suggests, the Content-based Filtering approach involves analyzing an item a user interacted with, and giving recommendations that are similar in **content** to that item. Content, in this case, refers to a set of attributes/features that describes your item. For a movie recommendation engine, a content-based approach would be to recommend movies that are of highest **similarity** based on its features, such as genres, actors, directors, year of production, etc. The assumption here is that users have preferences for a certain type of product, so we try to recommend a similar product to what the user has expressed liking for. Also, the goal here is to provide **alternatives** or **substitutes** to the item that was viewed.

**2. User based filtering approach:**

The User-Based Collaborative Filtering approach groups users according to prior usage behavior or according to their preferences, and then recommends an item that a similar user in the same group viewed or liked. To put this in layman terms, if user 1 liked movie A, B and C, and if user 2 liked movie A and B, then movie C might make a good recommendation to user 2. The User-Based Collaborative Filtering approach mimics how word-of-mouth recommendations work in real life.

In this post, I will use User-Based Collaborative Filtering to generate a top-10 recommendation list for users using the [recommender lab](http://cran.r-project.org/web/packages/recommenderlab/index.html) package available in R. The recommender lab package makes it easy to implement some of the popular collaborative filtering algorithms.

**Functional and Non-Functional Requirements:**

**Functional Requirements:**

* Cosine Similarity Algorithm
* Gathering Datasets from Movie lens

**Non- Functional Requirements:**

* Data Reliability
* R Libraries
* R Development environment.

**Literature Review:**

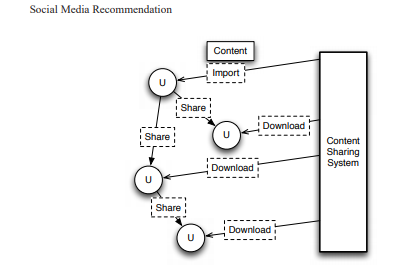
**Approaches:**

There are typically two types of algorithms for recommender systems ***–***

1. ***content-based methods***
2. ***collaborative filtering.***

Content-based methods measure the similarity of the recommended item (target item) to the ones that a target user (i.e., user who receives recommendations) likes or dislikes based on item attributes

Collaborative filtering finds users with tastes that are similar to the target users based on their past ratings which will then make recommendations to the target user based on the opinions of those similar users.



**Industry/Existing approach:**

Two approaches have been proposed:

1. The first approach is to condense the user/item rating matrix through dimensionality reduction techniques such as Singular Value Decomposition.
2. The second approach is to "enrich" the user/item rating matrix by 1) introducing default ratings or implicit user ratings, e.g., the time spent on reading articles; 2) using half-baked rating predictions from content-based methods

However, Traditional recommender systems do not take into consideration explicit social relations among users, yet the importance of social influence in product marketing has long been recognized.

**Improvisation:**

**Traditional Method:**

Despite all these efforts, recommender systems still face many challenging problems. Collaborative filtering has the data sparsity problem and the cold-start problem. It is difficult for recommender systems to accurately measure user similarities from those limited number of reviews. A related problem is the cold-start problem. Therefore, the system cannot accurately interpret this user's preference. First, in terms of the prediction accuracy, the additional information about users and their friends obtained from social networks improves the understanding of user behaviors and ratings. Therefore, we can model and interpret user preferences more precisely, and thus improve the prediction accuracy. Second, with friend information in social networks, it is no longer necessary to find similar users by measuring their rating similarity, because the fact that two people are friends already indicates that they have things in common. Thus, the data sparsity problem can be alleviated. Thus, the data sparsity problem can be alleviated.

**Industry/Existing Method:**

The recent emergence of online social networks (OSNs) gives us an opportunity to investigate the role of social influence in recommender systems. Members in those networks have their own personalized space where they not only publish their biographies, hobbies, interests, blogs, etc., but also list their friends. Furthermore, we propose to use the semantics of friend relationships and finer-grained user ratings to improve the prediction accuracy.

**Solutions for the existing issues:**

We look at disambiguating terms in social media using the **Naive Bayes algorithm**, which is a powerful and surprisingly simple algorithm. Naive Bayes takes a few shortcuts to properly compute the probabilities for classification, hence the term *naive* in the name. Naive Bayes is a probabilistic model that is unsurprisingly built upon a naive interpretation of Bayesian statistics. Despite the naive aspect, the method performs very well in many contexts. It can be used for classification of many different feature types and formats, but we will focus on one in this article: binary features in the bag-of-words model.

**Bayes' theorem:**

For most of us, when we were taught statistics, we started from a frequentist approach. In this approach, we assume the data comes from some distribution and we aim to determine what the parameters are for that distribution. However, those parameters are (perhaps incorrectly) assumed to be fixed. We use our model to describe the data, even testing to ensure the data fits our model.

**Advantages of using this algorithm:**

* It is easy and fast to predict class of test data set. It also performs well in multi class prediction.
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
* It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption)

**Categories:**

The state of category is

1. Problem that may occur during gathering of data, cleaning and forecasting due to cold start problem. **(handled by Rohith - 717525)**

2. New strategies to solve the cold-start problem and predictive accuracy **(handled by Sasirekha Kannan-716363)**

3. Functional and non-functional requirements **(handled by Sindhoori – 718158)**

**Related research papers:**

1. **A Tag and Social Network Based Recommender System:**

**Sogol Naseri Ryerson University**

Nowadays, people are looking for their desired items on the internet and making their selection decisions based on factors such as ratings and reviews on the candidate items from other customers, recommendation from other people or from recommender systems. The choice of the RS depends upon what a user wants to purchase. Suppose that a colleague suggests you watch a movie and then you visit your favourite online movie website. After entering the name of the movie in the search box, it presents the top-10 movies matched with the searched keywords. In a specific area of the webpage called "Users Who Watched This Movie Also Watched," a list of movies that you might like is displayed. If you are a regular registered user of this type of online movie website, such a personalized list of recommendations will appear automatically as soon as you enter the website.

**The approach:**

**1) Tags connect users and items:** Tags are helpful for users to find items and similarly tags are useful for items in a way that items could connect users. For example, the “software” tag could be found easily by a user searching for bookmarks about software

**2) Tags connect items:** Different items may share the same tags. For example: in the Last.fm website different songs may share common tags. So, it means that these songs could be categorized into the same cluster. Then users could find other songs based on their tag (interest)

**3) Tags connect users:** Different users may use the same tag for items. So, users with similar interests will be recognized through their shared common tags.

**2. Social Media Recommendation**

**Zhi Wang, Wenwu Zhu, Peng Cui, Lifeng Sun and Shiqiang Yang**

Online social network is attracting more and more people in today’s Internet, where users can share and consume all kinds of multimedia contents. Social media sharing is based on online social network, where users can reach various contents shared by others. With the exponential growth in social media contents, such as images and videos generated by users, it is of great importance to study how to provide personalized contents in the social media service. Recommendation is foreseen to be one of the most important services that can provide such personalized multimedia contents to users. However, social media recommendation is different from traditional content recommendation in that social media recommendation needs to take not only the content information but also users’ social relationship and behavior into account.

**Approach:**

1.Importing recommendation which recommends users the contents to import to their profiles in online social network. Since in popular online social network systems such as Facebook and Twitter, contents (e.g., videos) are not hosted by the systems directly, instead, they are imported from other content sharing systems, the importing recommendation helps users in online social network to discover contents that they want to import to online social network, among all the contents from the external content sharing systems.

2. Sharing recommendation which recommends users the contents to share in online social network. After users have imported contents to online social network, such contents will be distributed through the social connections. In online social network, users who obtain the contents shared by their friends or people they follow, can further share the contents to their (other) friends or users following them, making the contents propagate in a cascade way. The sharing recommendation helps a user discover the contents that he/she wants to share, among all the contents shared from his/her friends.

**3. Improved Recommendations Based on Trust Relationships in Social Networks**

To tackle the existing downfalls of the recommendation system driven by social networking, i.e. cold start, sparsity and low accuracy issues; the author integrated a trust relationship recommendation system to enhance the recommendation performance and prediction accuracy.

**Approach:**

* The author defined the trust relationship and proposed an improvement recommendation approach by adopt an effective recommendation algorithm considering the characteristics using trust relationship formulation. On comparing, the proposed methods result an elevated accuracy and secures a better capability to handle the limitations of the existing recommendation system; i.e. cold start and sparsity.
* The proposed trust relationship is sub-categorized into DIRECT and INDIRECT trust. The trust relationship was formulated based on trust factor analysis. Using this thesis, the author defined a recommendation algorithm named IRATR.

**IMPACT ON INDUSTRY:**

Recommender systems play an important role in helping online users find relevant information by suggesting information of potential interest to them. Due to the potential value of social relations in recommender systems, social recommendation has attracted increasing attention in recent years.

Recommender systems take a large pool of available data and make the decision-making process easier by providing just a few targeted selections. A great example of a recommender system at work is [LinkedIn’s recommendation system](https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites/lutzfinger/2014/09/02/recommendation-engines-the-reason-why-we-love-big-data/&refURL=&referrer=#423da801077c) for people you might know. Instead of suggesting an unlimited number of possible connections (like the [500 million users](http://fortune.com/2017/04/24/linkedin-users/) currently registered on the site), the algorithm is able to narrow down the pool of availability to a few options based on Big Data that it collects so that you can connect with more people that you may actually know and grow your network on the site

**Detailed Designs:**

The implementation includes several modules and are explained as follows:

**Module 1: Collecting the dataset:**

The dataset used was from MovieLens it includes the User details and the ratings of the users.

**Module 2: Building a content based Recommendation system:**

For a movie recommendation engine, a content-based approach would be to recommend movies that are of highest **similarity** based on its features, such as genres, actors, directors, year of production, etc. The assumption here is that users have preferences for a certain type of product, so we try to recommend a similar product to what the user has expressed liking for. Also, the goal here is to provide **alternatives** or **substitutes** to the item that was viewed.

**Module 3: Data Pre-processing:**

To obtain the movie features matrix, the pipe-separated genres available in the movies dataset had to be split. The data.table package has a tstrsplit() function that works well here to perform string splits. A matrix will be generated out of it and includes all the movie genres and each genre is separated into columns.

**Module 4: Building a Binary Matrix:**

We need to build a Binary matrix to keep the ratings in a simple form and generate the recommendations. For this we used a function called **dcast()** which is present in **reshape2** package.

**Module 5: Building a user profile Matrix:**

We need to create the simple user profile matrix; User profile matrix is the dot product of the movie genre matrix and the binary ratings matrix and then calculate the dot product of the genre matrix and the ratings matrix and obtain the user profiles.

**Module 6: Calculating the Jaccard distance:**

We used the dist() function from the proxy library to calculate Jaccard Distance. We have now successfully generated some recommendations for the first user in the dataset. You can repeat this for every user in your dataset with a for loop to get recommendations for all your users

**Module 7: Building a User based Collaborative Filtering Approach:**

**Creation of the Recommender Model:**

The User-based Collaborative Filtering recommender model was created with recommenderlab with the below parameters and the ratings matrix:

**Method: UBCF**  
**Similarity Calculation Method: Cosine Similarity  
Nearest Neighbors: 30**

The predicted item ratings of the user will be derived from the 5 nearest neighbors in its neighborhood. When the predicted item ratings are obtained, the top 10 most highly predicted ratings will be returned as the recommendations.

**Data Collection and Management:**

The dataset used was from MovieLens website, and is publicly available from the below link.

<https://grouplens.org/datasets/movielens/>

In a bid to keep the recommender simple, I used the smallest dataset available (ml-latest-small.zip) –100,000 ratings and 2,488 tag applications applied to 8,570 movies by 706 users.

**Implementation:**

**Computing MovieLens Recommendation System in R**

The collaborative filtering approach considers only user preferences and does not consider the features or contents of the items (books or movies) being recommended. In this project, in order to recommend movies, we will use a large set of users preferences towards the movies from a publicly available movie rating dataset. The data collected for computing recommendation are movies and ratings which is available in

<https://grouplens.org/datasets/movielens/>

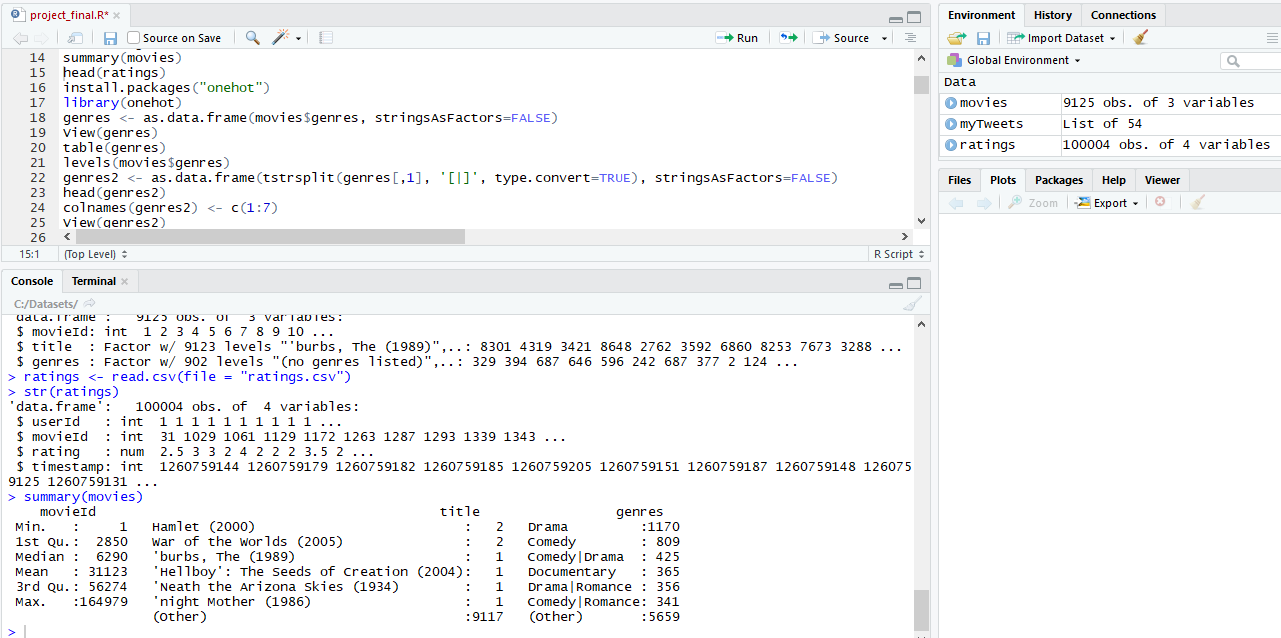
**Used Libraries:**

The following libraries were used in this project:

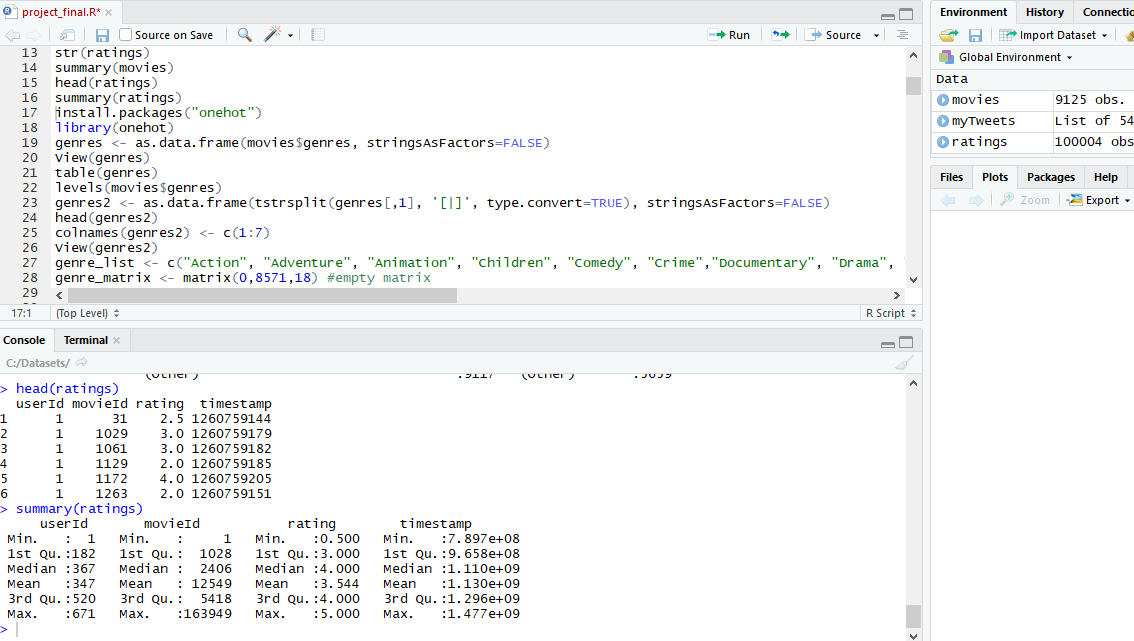
* library(recommenderlab)
* library(ggplot2)
* library(data.table)
* library(reshape2)

The data are contained in four files: *links.csv*, *movies.csv*, *ratings.csv* and *tags.csv*. We only use the files *movies.csv* and *ratings.csv* to build a recommendation system.

A summary of *movies* is given below, together with several first rows of a data frame



And here is a summary and a head of *ratings*:



Both *usersId* and *movieId* are presented as integers and should be changed to factors. Genres of the movies are not easily usable because of their format, we will deal with this in the next step.

**Data preprocessing**

Some pre-processing of the data available is required before creating the recommendation system. To obtain the movie features matrix, the pipe-separated genres available in the movies dataset had to be split. The data.table package has a tstrsplit() function that works well here to perform string splits.

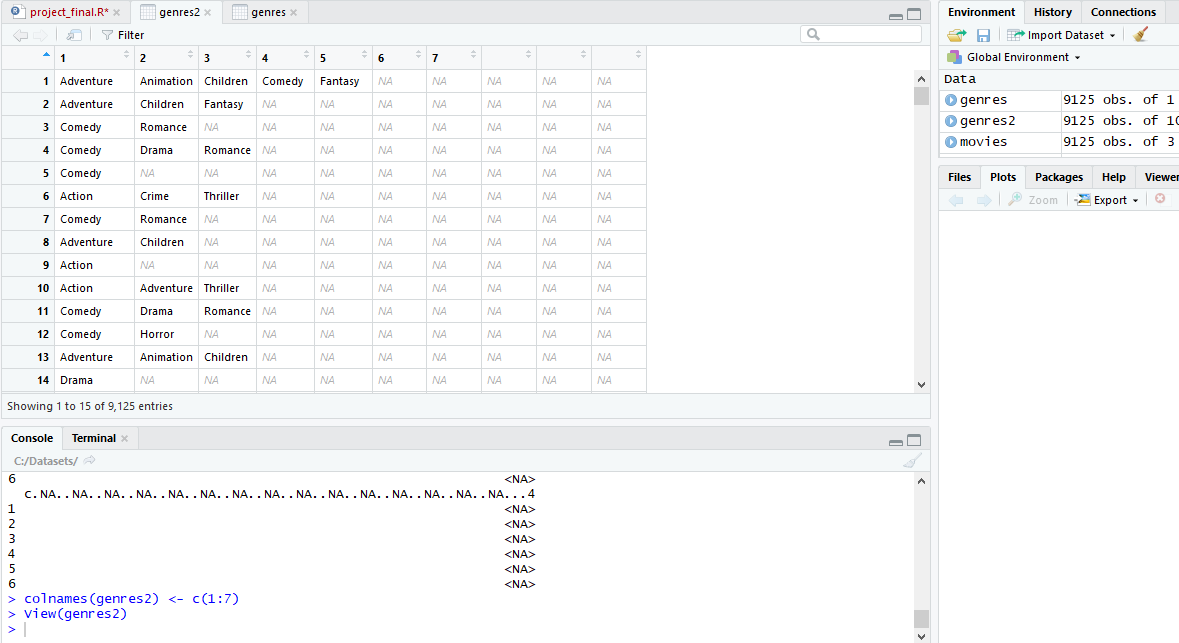
**genres <- as.data.frame(movies$genres, stringsAsFactors=FALSE)**

**library(data.table)**

**genres2 <- as.data.frame(tstrsplit(genres[,1], '[|]', type.convert=TRUE), stringsAsFactors=FALSE)**

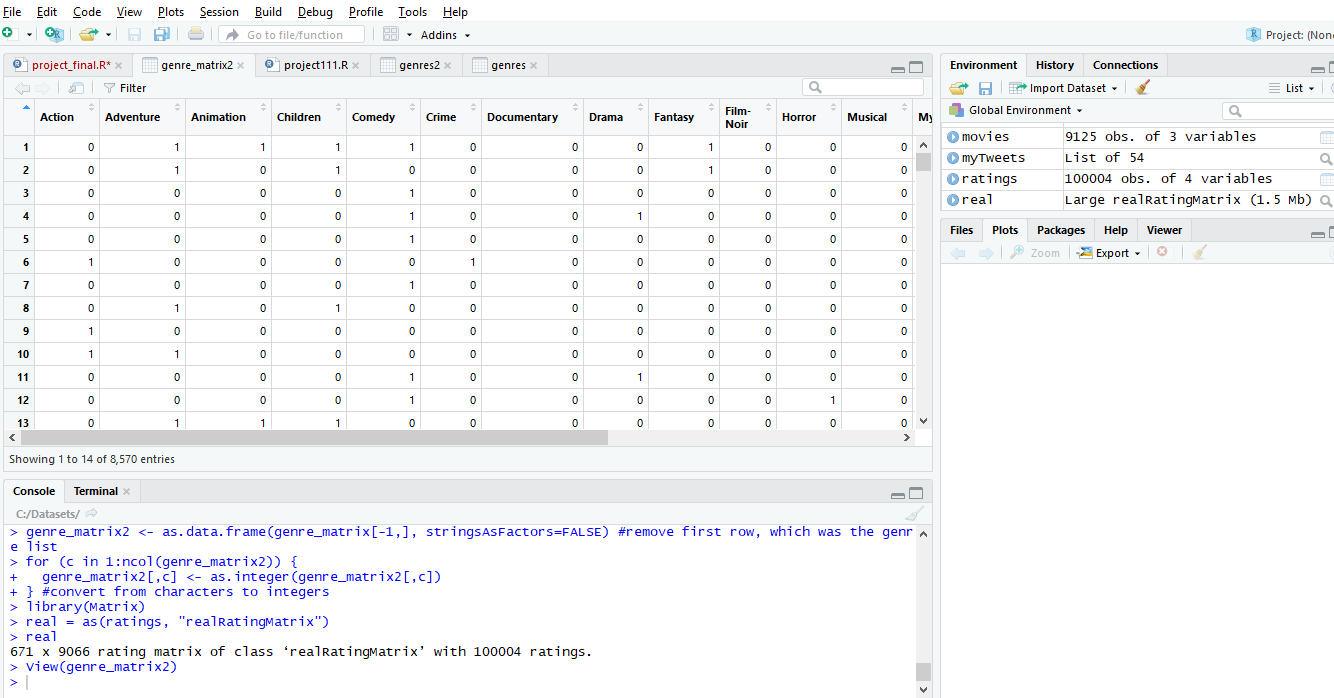
**colnames(genres2) <- c(1:7)**

This will give us a matrix that looks like this. This is basically movies$genres but each genre is separated into columns.



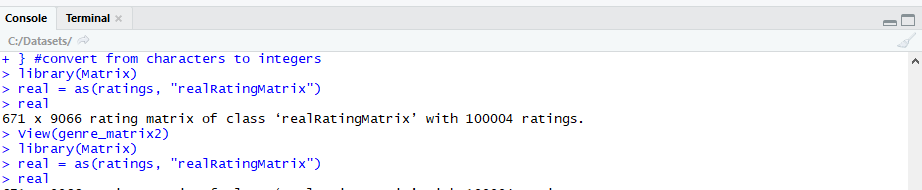
Then we create a matrix with columns representing every unique genre, and indicate whether a genre was present or not in each movie.

We have now obtained the movie genres matrix. Each column represents a unique movie genre, and each row is a unique movie. The table below just shows a preview of what the dataset looks like. We have 18 unique genres and 8570 unique movies.

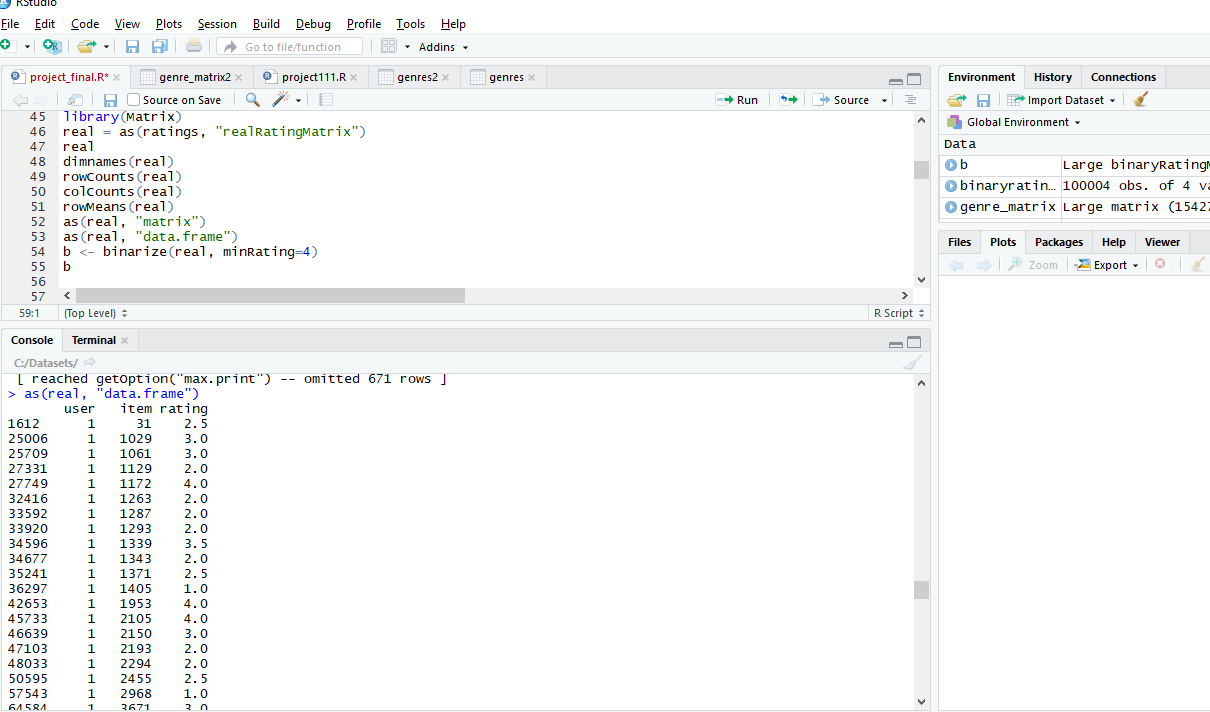


**Converting ratings matrix in a proper format**

To use the ratings data for building a recommendation engine with *recommenderlab*, I convert rating matrix into a sparse matrix of type *realRatingMatrix*.



A data frame is made that shows the user and items with rating, considering the minRating=4



Now, what we need is a user profile matrix. This can be easily done with the dcast() function in the reshape2 package. I first convert the ratings into a binary format to keep things simple. ratings of 4 and 5 are mapped to 1, representing likes, and ratings of 3 and below are mapped to -1, representing dislikes.

**binaryratings <- ratings**

**for (i in 1:nrow(binaryratings)){**

**if (binaryratings[i,3] > 3){**

**binaryratings[i,3] <- 1**

**}**

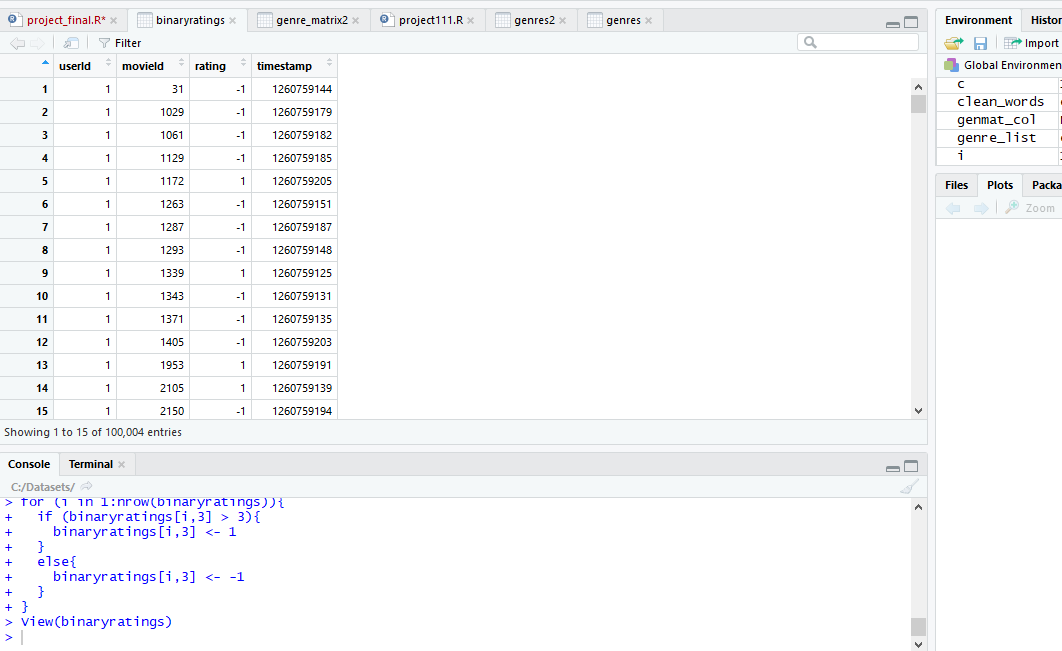
**else{**

**binaryratings[i,3] <- -1**

**}**

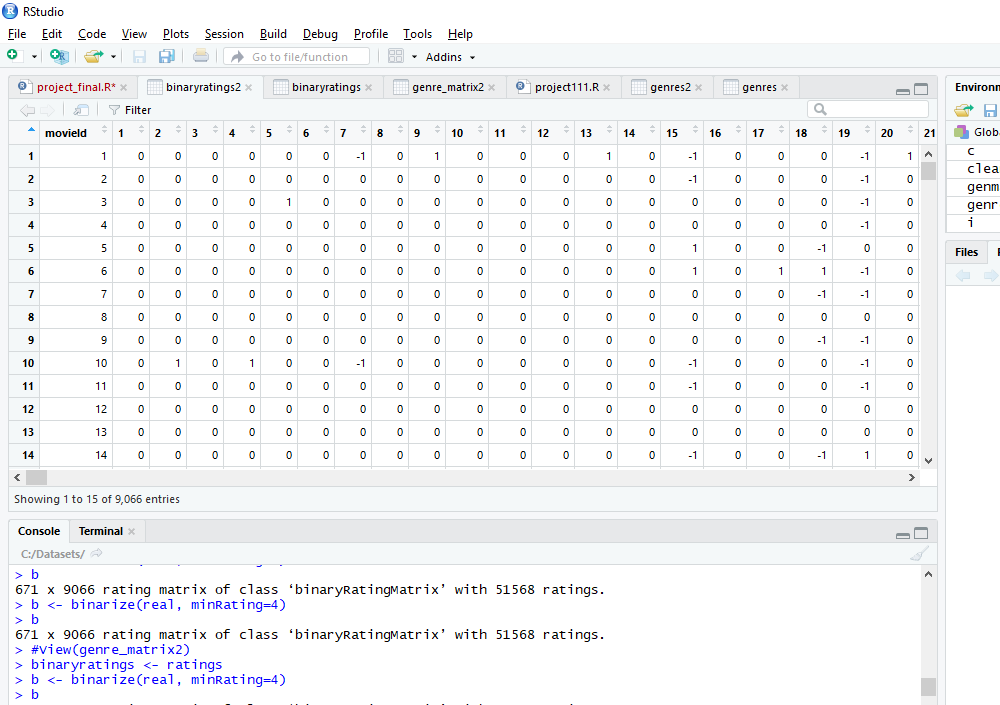
**}**

This is what the binaryratings dataset looks like now:



To obtain the binaryratings matrix in the correct format we need, I use the dcast() function in the reshape2 package. This basically transforms the data from a long format to a wide format. This also creates many NA values because not every user rated every movie. I substituted the NA values with 0.

Now we have the binaryratings matrix in the right format. This matrix has 8552 rows, representing the movieIds, and 706 cols, representing the userIds. The matrix now looks something like this:



Now we can calculate the dot product of the genre matrix and the ratings matrix and obtain the user profiles.

This user profiles shows the aggregated inclination of each user towards movie genres. Each column represents a unique userId, and positive values shows a preference towards a certain genre. The values were again simplified into a binary matrix — positive values were mapped to 1 to represent likes, negative values were mapped to 0 to represent dislikes.

**Now that we have the user profiles, we can go 2 ways from here.**  
1) Predict if a user likes an item based on the item descriptions (movie genres). This can be done by predicting user movie ratings.  
2) Assume that users like similar items, and retrieve movies that are closest in similarity to a user’s profile, which represents a user’s preference for an item’s feature.

We chose the second way, and decided to use Jaccard Distance to measure the similarity between user profiles, and the movie genre matrix. Jaccard Distance was my metric of choice for being suitable for binary data.

We used the dist() function from the proxy library to calculate Jaccard Distance. Unfortunately, it seems like it calculates the distance between rows from a single matrix, and we had 2 matrices. We decided to combine the genre matrix with the user profile matrix one at a time, and retrieve the minimum distance for each user. This calculation took quite a while in my local environment.

For the sake of simplicity, we will show how we did it with the first user in the dataset.

**#Calculate Jaccard distance between user profile and all movies**

library(proxy)

sim\_results <- dist(sim\_mat, method = "Jaccard")

sim\_results <- as.data.frame(as.matrix(sim\_results[1:8552]))

rows <- which (sim\_results == min(sim\_results))

#Recommended movies

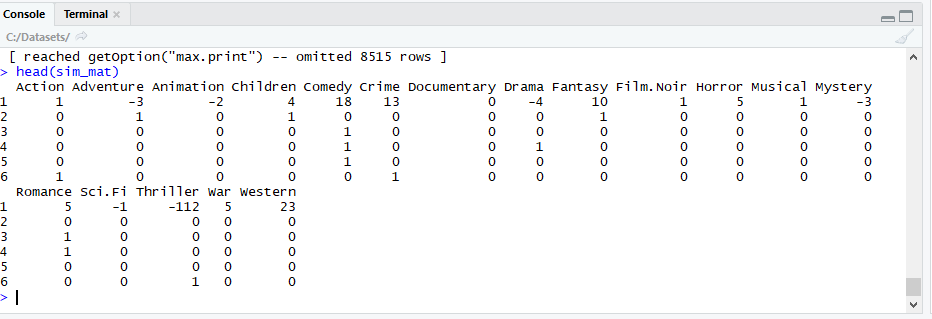
movies[rows,2]

We have now successfully generated some recommendations for the first user in the dataset. You can repeat this for every user in your dataset with a for loop to get recommendations for all your users.

Let’s look at the results.

This is the user profile we created for user 1. User 1 is inclined towards the following genres: Children, Documentary, Fantasy, Horror, Musical, War

> head(sim\_mat)



These were the movies recommended for User 1:

> #Recommended movies

> movies[rows,]



Now that we are done with a simple content-based recommender, let’s consider its strengths and weaknesses in general.

**Strengths**: Content-based recommender systems don’t require a lot of user data. You just need item data and you’re able to start giving recommendations to users. Also, your recommendation engine does not depend on lots of user data, so it is possible to give recommendations to even your first customer if you have adequate data to build his user profile.

**Weaknesses**: Your item data needs to be well distributed. It won’t be effective to have a content-based recommender if 80% of your movies are action movies. Also, the recommendations you get will likely be direct substitutes, and not complements, of the item the user interacted with.

**The User-Based Collaborative Filtering Approach**The User-Based Collaborative Filtering approach groups users according to prior usage behavior or according to their preferences, and then recommends an item that a similar user in the same group viewed or liked. To put this in layman terms, if user 1 liked movie A, B and C, and if user 2 liked movie A and B, then movie C might make a good recommendation to user 2. The User-Based Collaborative Filtering approach mimics how word-of-mouth recommendations work in real life.

In this project, we will use User-Based Collaborative Filtering to generate a top-10 recommendation list for users using the recommenderlab package available in R. The recommenderlab package makes it easy to implement some of the popular collaborative filtering algorithms.

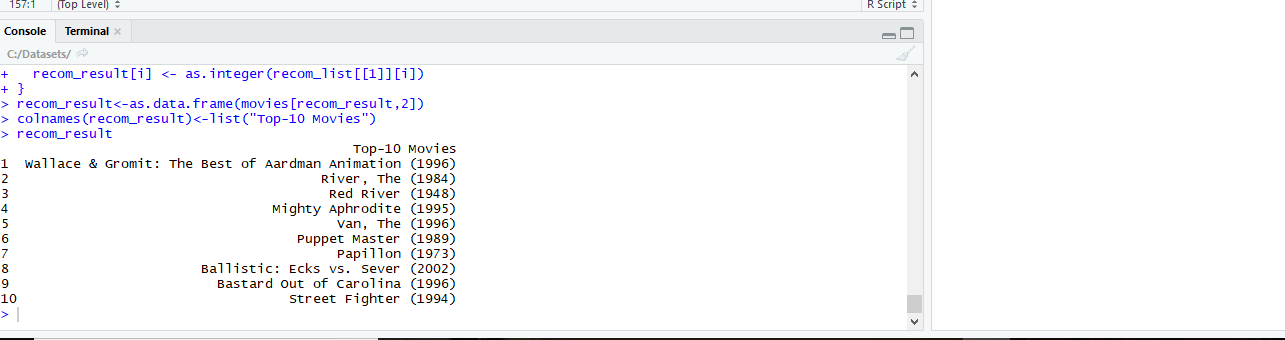
**Creation of the Recommender Model**The User-based Collaborative Filtering recommender model was created with recommenderlab with the below parameters and the ratings matrix:

Method: UBCF  
Similarity Calculation Method: Cosine Similarity  
Nearest Neighbors: 30

The predicted item ratings of the user will be derived from the 5 nearest neighbors in its neighborhood. When the predicted item ratings are obtained, the top 10 most highly predicted ratings will be returned as the recommendations.

**Normalize the data:**

And we have easily obtained the top 10 results for user 1! These were the movies recommended to user 1.



The recommenderlab package also provides an easy way to evaluate your model.

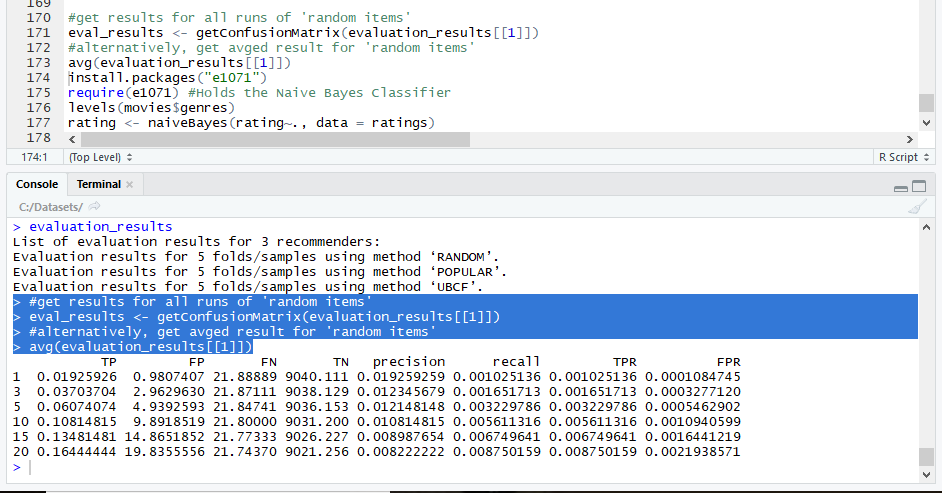
The evaluation results of the top-N recommender:

> #get results for all runs of 'random items'

> eval\_results <- getConfusionMatrix(evaluation\_results[[1]])

> #alternatively, get avged result for 'random items'

> avg(evaluation\_results[[1]])



Let’s look at the strengths and weaknesses of the User-based Collaborative Filtering approach in general.

**Strengths**: User-based Collaborative Filtering gives recommendations that can be complements to the item the user was interacting with. This might be a stronger recommendation than what a content-based recommender can provide as users might not be looking for direct substitutes to a movie they had just viewed or previously watched.

**Weaknesses**: User-based Collaborative Filtering is a type of Memory-based Collaborative Filtering that uses all user data in the database to create recommendations. Comparing the pairwise correlation of every user in your dataset is not scalable. If there were millions of users, this computation would be very time consuming. Possible ways to get around this would be to implement some form of dimensionality reduction, such as Principal Component Analysis, or to use a model-based algorithm instead. Also, user-based collaborative filtering relies on past user choices to make future recommendations. The implications of this is that it assumes that a user’s taste and preference remains constant over time, which might not be true and makes it difficult to pre-compute user similarities offline.

To classify the datasets, we introduce naïve bayes algorithm:

 We will use the Naive Bayes Technique to classify such users and ratings how well it performs.  
As we know, Bayes theorem is based on conditional probability and uses the formula

**P (A | B) = P(A) \* P(B | A) / P(B)**

We now know how this conditional probability comes from multiplication of events so if we use the general multiplication rule, we get another variation of the theorem that is, using P (A AND B) = P(A) \* P(B | A), we can obtain the value for conditional probability: P(B | A) = P(A AND B) / P(A) which is the variation of Bayes theorem.

Since P (A AND B) also equals P(B) \* P(A | B), we can substitute it and get back the original formula  
P(B | A) = P(B) \* P(A | B) / P(A)

Using this for each of the features among movies, users and ratings Naive Bayes algorithm will calculate the conditional probability of survival of the combination.

> require(e1071) #Holds the Naive Bayes Classifier

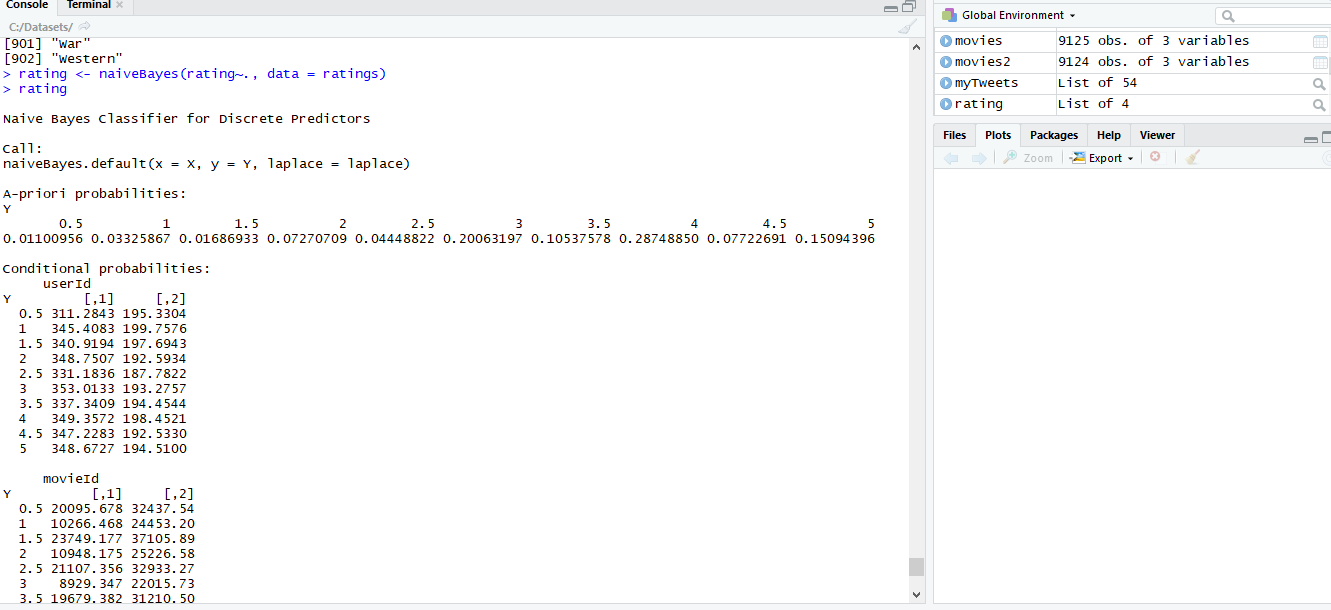
Loading required package: e1071

Warning message:

package ‘e1071’ was built under R version 3.4.4

we install “e01701” package that holds the naïve bayes classifier

The result is computed as follows:



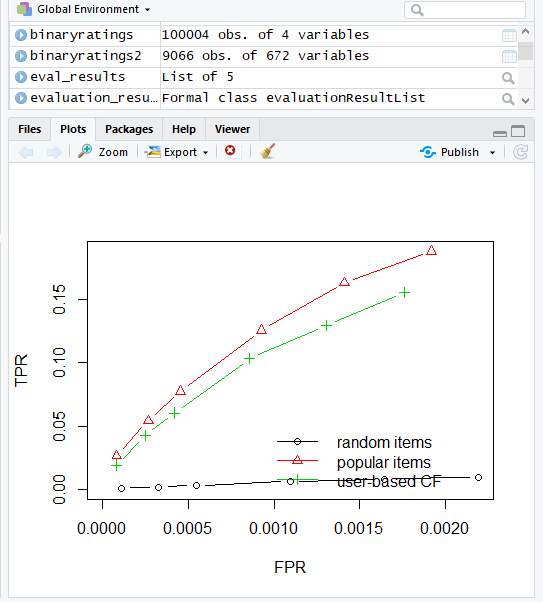
**Test cases:**

**Case 1: Evaluation for the rating matrix is done using “Cross Validation” method to predict the good rating.**

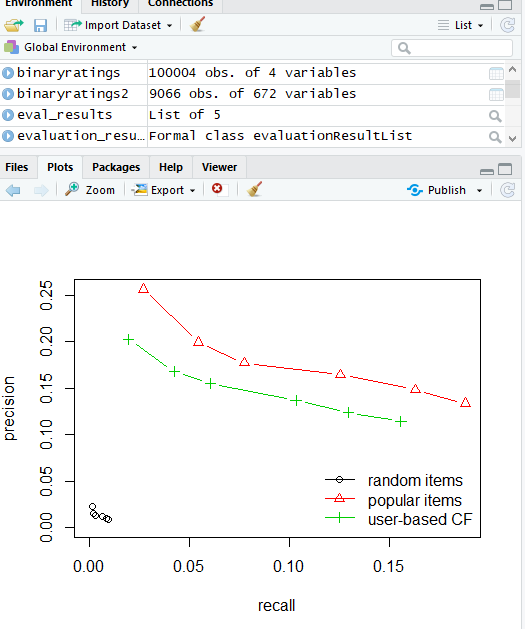
A good performance index is the area under the curve (AUC), that is, the area under the ROC curve. Even without computing it, the chart shows that the highest is UBCF with cosine distance, so it’s the best-performing technique.

The UBCF with cosine distance is still the top model. Depending on what is the main purpose of the system, an appropriate number of items to recommend should be defined.

A plot is made to measure the averaged ROC



Another plot is made to calculate the efficiency in data filtering by measuring the precision time and recall



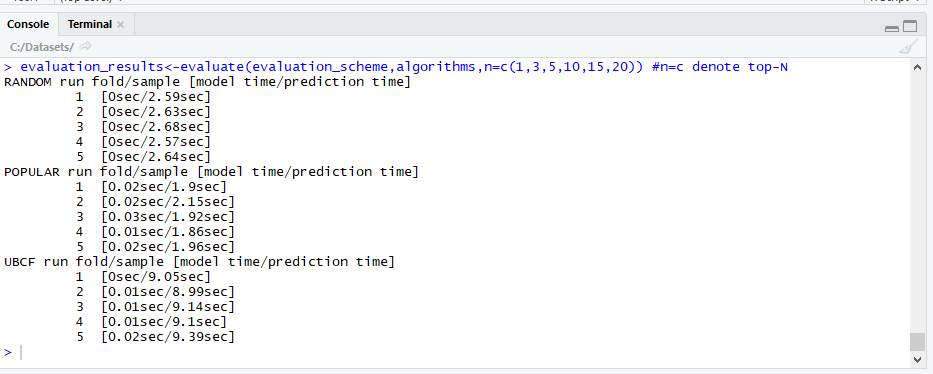
The result of the test case proves that it is an efficient filtering process for analysing large datasets.

**Test case 2:**

**Comparing the models:**

This test case is made to evaluate whether the data filtering based on user based collaborative filtering is efficient to plot the data when it is classified.

The parameters used to classify these values are classified as random items, popular items and user based CF



**Conclusion:**

Recommender systems play an important role in helping online users find relevant information by suggesting information of potential interest to them. Due to the potential value of social relations in recommender systems, social recommendation has attracted increasing attention in recent years.

Social references in recommender systems is highly efficient than normal recommender systems.

**References:**

1.Jianming He and Wesley W. Chu, http://www.cobase.cs.ucla.edu/tech-docs/jmhek/snrs.pdf

2.G. Adomavicius, and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering

## 3. https://muffynomster.wordpress.com/2015/06/07/building-a-movie-recommendation-engine-with-r/