

Listener Music Preference Using **Association Rules in R**

Course: DA5020 38540

Name: Anupaa B Ramesh

NU id: 001253433

Module: Term Project

Problem Statement for The Project

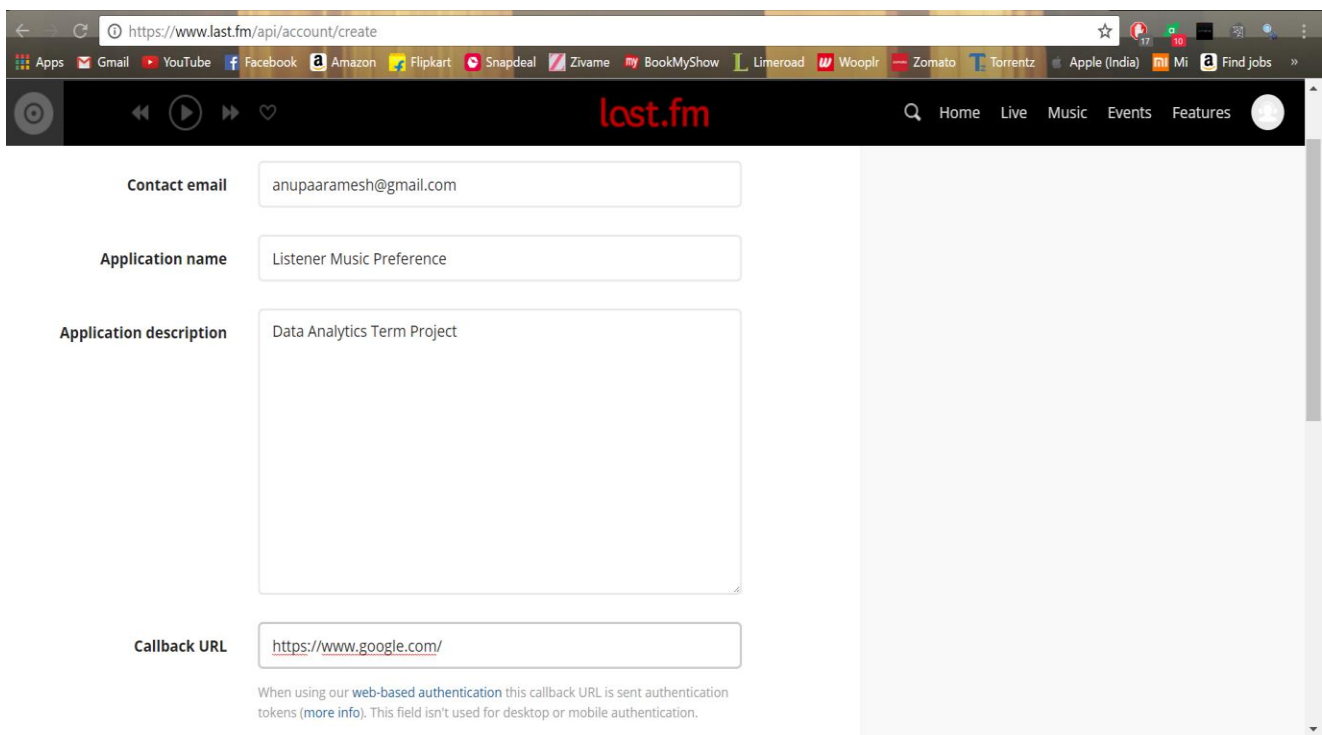
To understand what music each listener prefers to listen, all the data regarding the person's online choices relevant to music is collected. This recorded data will help send out the "recommended music" that each listener is likely to enjoy. Also, this enables to come up with a focused marketing strategy that sends out advertisements for music that a listener may wish to buy. However, this results in wasting money on scarce advertising.

Proposed Solution:

Suppose that we are provided with data from a music community site, giving us details of each user. This will further help us get access on a log of every artist that the listed users have downloaded on their computer. With this data, we will also get information on the demographics of the listed users (such as age, sex, location, occupation, and interests). The objective of obtaining this data lies in building a system that recommends new music to the users in this listed community. From the available information, it is usually not difficult to determine the support for various individual artists (that is, the frequencies of a specific music genre/artist or song that a user is listening to) as well as the joint support for pairs (or larger groupings) of artists.

One such music community site that gives us access to a variety of such music related data is **lastfm**. This data from **lastfm** can be accessed through their web based API. Then with the use of Association Rules available in R we can develop relationships between variables of the dataset.

First, we need to create an API for the site lastfm which can be accessed through the link <http://www.last.fm/api/intro> . This link has all the information regarding the API creation the request styles, the authentication requirements and the various API methods available. The root URL for the lastfm API is located at <http://ws.audioscrobbler.com/2.0/> .Below is a screenshot of the information entered to create and API.



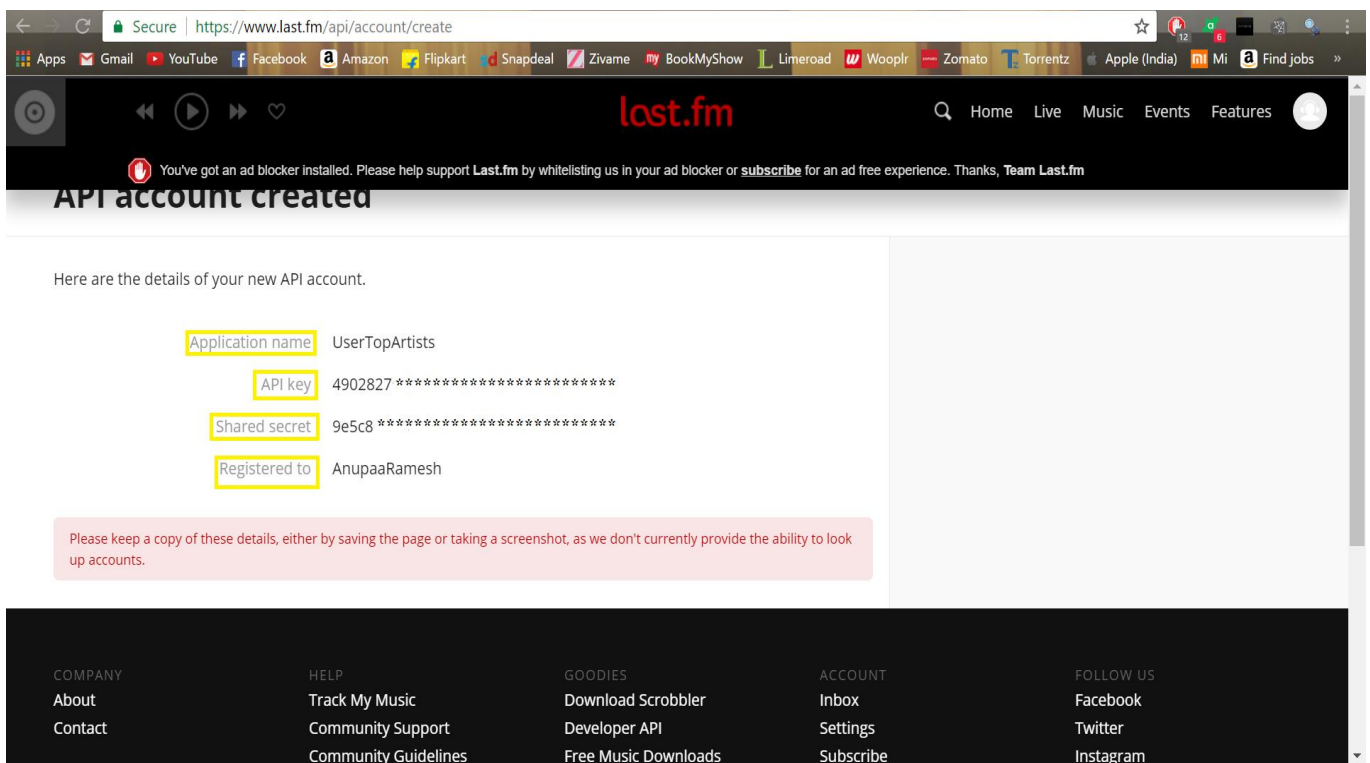
The screenshot displays the 'Create API Account' page on Last.fm. The browser's address bar shows the URL <https://www.last.fm/api/account/create>. The page features a dark navigation bar with the Last.fm logo and links to Home, Live, Music, Events, and Features. The form contains the following information:

- Contact email:** anupaaramesh@gmail.com
- Application name:** Listener Music Preference
- Application description:** Data Analytics Term Project
- Callback URL:** <https://www.google.com/>

Below the Callback URL field, a note states: "When using our [web-based authentication](#) this callback URL is sent authentication tokens ([more info](#)). This field isn't used for desktop or mobile authentication."

We need to provide a contact email address, a name for the application that we wish to create, a short description of the application and a callback URL. A callback URL is basically what the name suggests. It just redirects you to the page of the callback URL once the task you want to perform is completed.

Once the API account is successfully created we will receive the details regarding the application name, API key, shared secret and the name of the person to whom the API is registered to. Here is a screenshot of the same.



The API key which stands for application programming interface key is a code that can be passed to call a specific API which is identified by its user from the website. The API key also acts as a unique identifier and a secret token, this enables to keep track of how the API is being used and to make sure that it is not being abused by malicious users.

The data required for the to achieve the results of suggesting music to each listener based on their current choices contains the following fields. The user id, age, sex, country and the name of the artist. The data set contains information close to 300,000 records of song (or artistsr) selections that is listed per listening frequency given by 15,000 users. Each row of the data set contains the name of the artist that the user has been listening to. The first user is a German lady, who has listened to 16 artists. Below is a screenshot of the data that was imported from the lastfm API in a XML format.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

ramesh.a_TermProject.r* x lastfm x ramesh.a_FinalExam.r* x ramesh.a_hwr11.r x ramesh.a_hwr5.r x Final >>

Filter

	user	artist	sex	country
1	1	red hot chili peppers	f	Germany
2	1	the black dahlia murder	f	Germany
3	1	goldfrapp	f	Germany
4	1	dropkick murphys	f	Germany
5	1	le tigre	f	Germany
6	1	schandmaul	f	Germany
7	1	edguy	f	Germany
8	1	jack johnson	f	Germany
9	1	eluveitie	f	Germany
10	1	the killers	f	Germany
11	1	judas priest	f	Germany
12	1	rob zombie	f	Germany
13	1	john mayer	f	Germany
14	1	the who	f	Germany
15	1	guano apes	f	Germany
16	1	the rolling stones	f	Germany
17	3	devendra banhart	m	United States
18	3	boards of canada	m	United States
19	3	cocorosie	m	United States
20	3	aphex twin	m	United States
21	3	animal collective	m	United States
22	3	atmosphere	m	United States
23	3	joanna newsom	m	United States
24	3	air	m	United States
25	3	portishead	m	United States
26	3	massive attack	m	United States
27	3	broken social scene	m	United States

Showing 1 to 27 of 289,955 entries

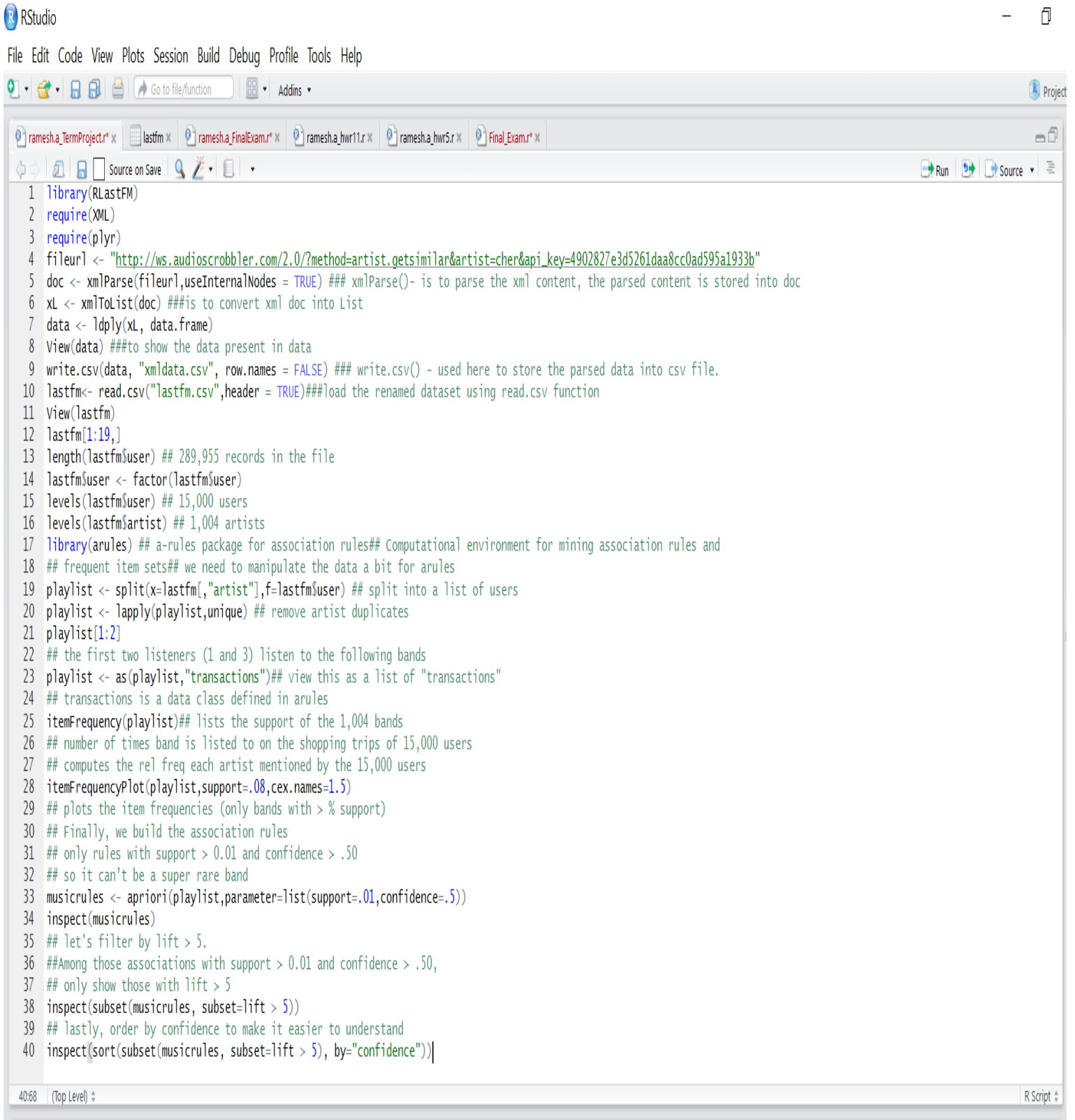
Console

The suitable approach is to count the number of incidences across all the network members. After this, we need to divide those frequencies with the number of members. Using the support value the values of confidence and lift can be calculated. This has resulted in the first 16 rows of the data matrix.

To accomplish the above-mentioned approach first, we need to transform the imported data set into an incidence matrix, where each listener represents a row, with 0s and 1s across the columns. This indicates if he or she has played a certain artist or not. Then, the support for each of the listed 1004 artists needs to be calculated by displaying the support for all artists with support larger than 0.08. Then, the association rules using the function `Apriori` in the R package `arules` are constructed. Then we need look for artists (or groups of artists) who have support that is larger than 0.01 (1%). After the calculation is checked, another music collection of an artist turns out to be larger than 0.50 (50%).

The screenshots of the R code used to achieve this and the outputs obtained can be found in the subsequent pages.

R-code:



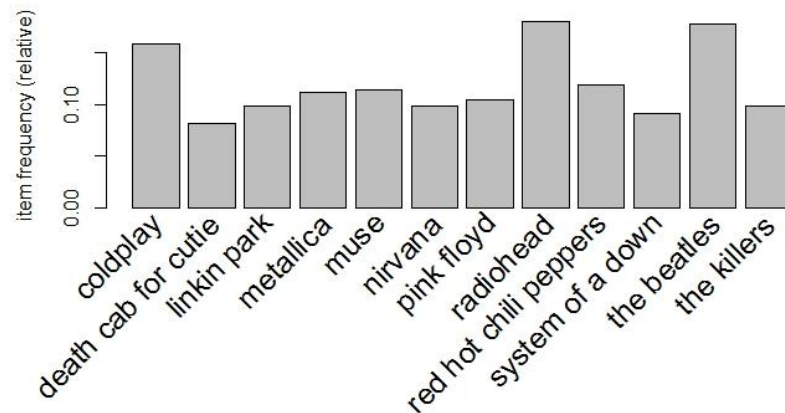
The screenshot shows the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and a 'Go to file/function' search bar. The source editor displays the following R code:

```
1 library(RLSTM)
2 require(XML)
3 require(plyr)
4 fileurl <- "http://ws.audioscrobbler.com/2.0/?method=artist.getsimilar&artist=cher&api_key=4902827e3d5261daa8cc0ad595a1933b"
5 doc <- xmlParse(fileurl,useInternalNodes = TRUE) ## xmlParse() - is to parse the xml content, the parsed content is stored into doc
6 xL <- xmlToList(doc) ##is to convert xml doc into List
7 data <- ldply(xL, data.frame)
8 View(data) ##to show the data present in data
9 write.csv(data, "xmldata.csv", row.names = FALSE) ## write.csv() - used here to store the parsed data into csv file.
10 lastfm<- read.csv("lastfm.csv",header = TRUE)##load the renamed dataset using read.csv function
11 View(lastfm)
12 lastfm[1:19,]
13 length(lastfm$user) ## 289,955 records in the file
14 lastfm$user <- factor(lastfm$user)
15 levels(lastfm$user) ## 15,000 users
16 levels(lastfm$artist) ## 1,004 artists
17 library(arules) ## a-rules package for association rules## Computational environment for mining association rules and
18 ## frequent item sets## we need to manipulate the data a bit for arules
19 playlist <- split(x=lastfm[, "artist"],f=lastfm$user) ## split into a list of users
20 playlist <- lapply(playlist,unique) ## remove artist duplicates
21 playlist[1:2]
22 ## the first two listeners (1 and 3) listen to the following bands
23 playlist <- as(playlist,"transactions")## view this as a list of "transactions"
24 ## transactions is a data class defined in arules
25 itemFrequency(playlist)## lists the support of the 1,004 bands
26 ## number of times band is listed to on the shopping trips of 15,000 users
27 ## computes the rel freq each artist mentioned by the 15,000 users
28 itemFrequencyPlot(playlist,support=.08,cex.names=1.5)
29 ## plots the item frequencies (only bands with > % support)
30 ## Finally, we build the association rules
31 ## only rules with support > 0.01 and confidence > .50
32 ## so it can't be a super rare band
33 musicrules <- apriori(playlist,parameter=list(support=.01,confidence=.5))
34 inspect(musicrules)
35 ## let's filter by lift > 5.
36 ##Among those associations with support > 0.01 and confidence > .50,
37 ## only show those with lift > 5
38 inspect(subset(musicrules, subset=lift > 5))
39 ## lastly, order by confidence to make it easier to understand
40 inspect(sort(subset(musicrules, subset=lift > 5), by="confidence"))
```

The status bar at the bottom shows '40:68 (Top Level)' and 'R Script'.

Outputs Obtained:

plots of the item frequencies (only bands with > % support)



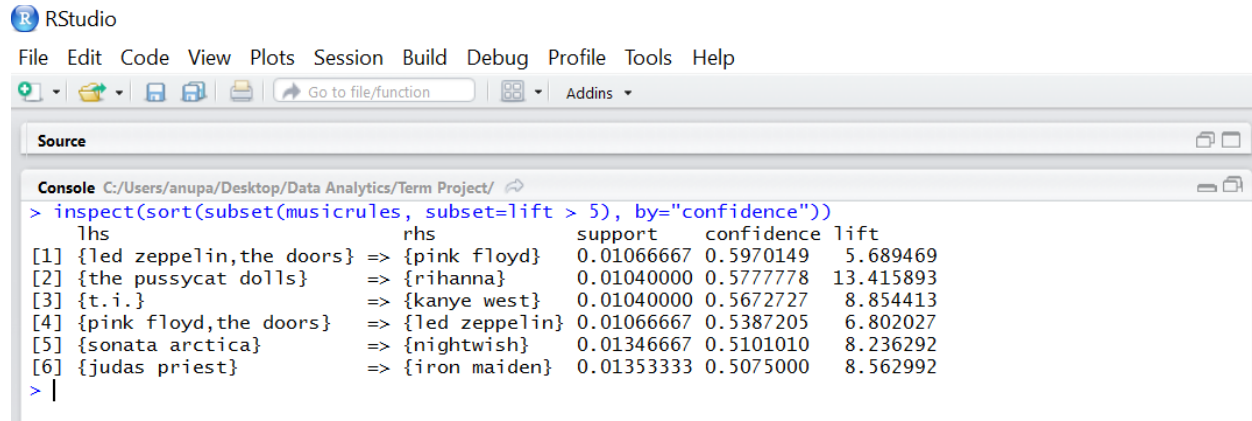
Finally, the association rules only rules with support > 0.01 and confidence > .50 are built. So it can't be a super rare band. This is accomplished by using the following command.

```
musicrules <- apriori(playlist,parameter=list(support=.01,confidence=.5))  
inspect(musicrules)
```

Then we filter by lift > 5, among those associations with support > 0.01 and confidence > .50. So we show only those with lift > 5

```
RStudio  
File Edit Code View Plots Session Build Debug Profile Tools Help  
Go to file/function Addins  
Source  
Console C:/Users/anupa/Desktop/Data Analytics/Term Project/  
> inspect(subset(musicrules, subset=lift > 5))  
lhs rhs support confidence lift  
[1] {t.i.} => {kanye west} 0.01040000 0.5672727 8.854413  
[2] {the pussycat dolls} => {rihanna} 0.01040000 0.5777778 13.415893  
[3] {sonata arctica} => {nightwish} 0.01346667 0.5101010 8.236292  
[4] {judas priest} => {iron maiden} 0.01353333 0.5075000 8.562992  
[5] {led zeppelin,the doors} => {pink floyd} 0.01066667 0.5970149 5.689469  
[6] {pink floyd,the doors} => {led zeppelin} 0.01066667 0.5387205 6.802027
```


Lastly, we sort the above results in an order by confidence to make it easier to understand.



The image shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main window is divided into two panes: 'Source' and 'Console'. The 'Console' pane shows the following R code and its output:

```
> inspect(sort(subset(musicrules, subset=lift > 5), by="confidence"))
```

	lhs	rhs	support	confidence	lift
[1]	{led zeppelin,the doors}	=> {pink floyd}	0.01066667	0.5970149	5.689469
[2]	{the pussycat dolls}	=> {rihanna}	0.01040000	0.5777778	13.415893
[3]	{t.i.}	=> {kanye west}	0.01040000	0.5672727	8.854413
[4]	{pink floyd,the doors}	=> {led zeppelin}	0.01066667	0.5387205	6.802027
[5]	{sonata arctica}	=> {nightwish}	0.01346667	0.5101010	8.236292
[6]	{judas priest}	=> {iron maiden}	0.01353333	0.5075000	8.562992

```
> |
```