ASDM Workshop Week 3: Association Rule Mining with R

- Charith Silva, Dr. Mo Saraee

Association rule mining is the data mining process of finding the rules that may govern associations between sets of items. The term market basket analysis refers to a specific implementation of association rules mining that many companies use for a variety of purposes, so in a given transaction with multiple items, it tries to find the rules that govern how or why such items are often bought together. Association rules are rules which surpass a user-specified minimum support and minimum Confidence threshold. For example, peanut butter and jelly are often bought together because a lot of people like to make sandwiches using these two items. Also, surprisingly, diapers and beer are bought together because, as it turns out, that dads are often tasked to do the shopping while the moms are left with the baby.

The main applications of association rule mining:

- Basket data analysis - is to analyse the association of purchased items in a single basket or single purchase.
- Cross marketing - is to work with other businesses that complement your own, not competitors.
- Catalogue design - the selection of items in a business' catalogue are often designed to complement each other so that buying one item will lead to buying of another. So, these items are often complements or very related.

Besides market basket analysis, association rules are commonly used for recommender systems and click stream analysis. Many online service providers such as Amazon and Netflix use recommender systems. Recommender systems can use association rules to discover related products or identify customers who have similar interests. For example, association rules may suggest that those customers who have bought product A have also bought product B, or those customers who have bought products A, B, and C are more similar to this customer. These findings provide opportunities for retailers to cross-sell their products. Click stream analysis refers to the analytics on data related to web browsing and user clicks, which is stored on the client or the server side. Web usage log files generated on web servers contain huge amounts of information, and association rules can potentially give useful knowledge to web usage data analysts.

transaction ID	items
1	milk, bread
2	bread, butter
3	beer
4	milk, bread, butter
5	bread, butter

Figure 1: An example supermarket database with five transactions

Support:

The support supp(X) of an item set X is defined as the proportion of transactions in the data set which contain the item set. For example, database in Figure 1, the item set {milk, bread} has a support of 2/5 = 0.4 since it occurs in 40% of all transactions(2 out of 5 transactions).

Confidence:

Confidence of a rule is defined conf(X ⇒ Y) = supp(X ∪ Y)/supp(X). Therefore, an association rule X ⇒ Y will satisfy, For example, the rule {milk, bread} ⇒ {butter} has a confidence of 0.2/0.4 = 0.5 in the database in Figure 1, which means that for 50% of the transactions containing milk and bread the rule is correct. Confidence can be interpreted as an estimate of the probability P(Y |X), the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.

Lift:

The lift of a rule is defined as lift(X ⇒ Y) = supp(X ∪ Y)/(supp(X)supp(Y))and can be interpreted as the deviation of the support of the whole rule from the support expected under independence given the supports of both sides of the rule. Greater lift values(>> 1) indicate stronger associations.

Part 1: Exercise

In this workshop we will practice how to implement Association Rule Mining using R programming. The dataset, **marketbasket.csv**, that we will use in the workshop can be downloaded from Blackboard. We will be using market basket data to find purchasing behaviours of customers using association rule mining. We want to know that what items are bought together using association rules.

Data Explanation

Item	Purchase
Apples	Yes=Purchased, No=Not purchased
banana	Yes=Purchased, No=Not purchased
Coke	Yes=Purchased, No=Not purchased
Turkey	Yes=Purchased, No=Not purchased
bourbon	Yes=Purchased, No=Not purchased
ice_cream	Yes=Purchased, No=Not purchased
Baguette	Yes=Purchased, No=Not purchased
Soda	Yes=Purchased, No=Not purchased
Choclate	Yes=Purchased, No=Not purchased
Cracker	Yes=Purchased, No=Not purchased
Cosmetics	Yes=Purchased, No=Not purchased
Avocado	Yes=Purchased, No=Not purchased
Artichoke	Yes=Purchased, No=Not purchased
Sardines	Yes=Purchased, No=Not purchased

Implementation:

In this workshop, we will be using "**arules"** package which use **Apriori Algorithm**. The Apriori Algorithm is an influential algorithm for mining frequent item sets for boolean association rules. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time.

1. Download **`marketbasket.csv`** dataset from Blackboard and save to a folder on your F: drive eg: F\ASDM\Week3. Open the CSV file using excel to get a rough idea about the dataset, e.g, attributes and their values.

2. Start RStudio.

3. Change working directory.

   `File → More → Go To Working Directory…`

In the Go To Working Directory dialogue, navigate to and select the folder where you saved your data file eg:`F:\ASDM\Week3`. Click OK.

or

using R commands as follow:

```
mypath = "F:\\ASDM\\Week3"   # you need to change the string to your
                             directory
setwd(mypath)                # set working directory
getwd()                      # check if the working directory has
                             changed correctly
```

4. Open a new R script window:

   `File → New File → R script`

5. Read the data file

```
#colClasses option is used to convert to factor
Marketbasket <-read.csv("marketbasket.csv",header=T,
                        colClasses="factor")
```

The data is read into the data frame named "`marketbasket`"

6. Inspect the dataset in R

   Once the file has been imported to R, we often want to do few things to explore the dataset:

```
names(marketbasket)
head(marketbasket)
tail(marketbasket)
summary(marketbasket)
str(marketbasket)
```

7. Check the dimension of the "marketbasket" dataset

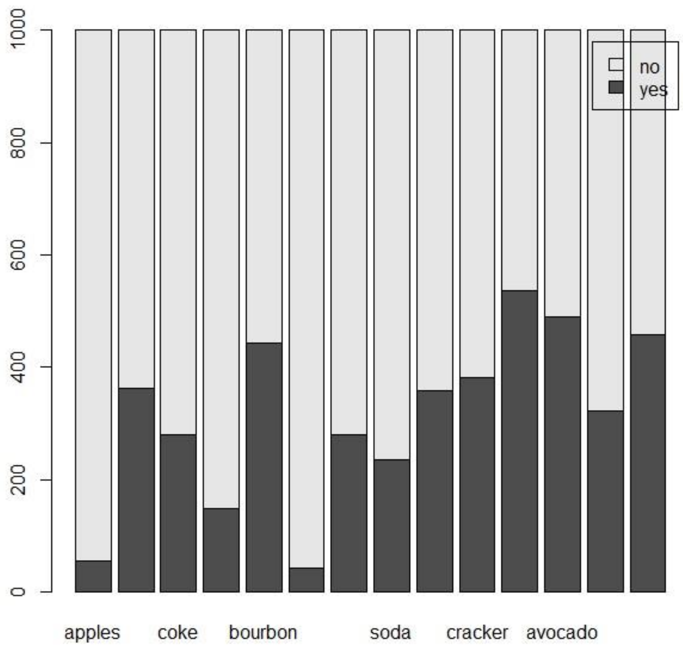
   `dim(marketbasket)`

8. Plot and explore the "marketbasket" dataset with barplot() function

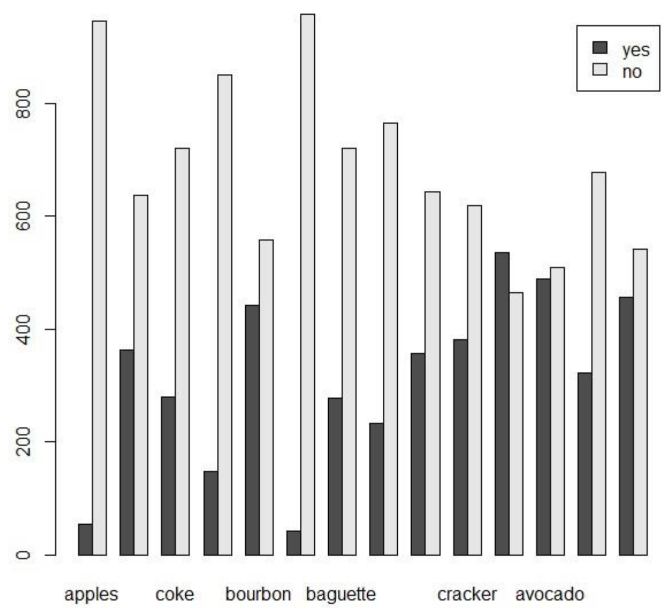
```
#colSums() function computes the sums of columns.
yes <- colSums(marketbasket == "Yes")
yes
```

```
no <-colSums(marketbasket=="No")
no
purchased <- rbind(yes,no)
purchased

barplot(purchased,legend=rownames(purchased)) #Plot 1
barplot(purchased, beside=T,legend=rownames(purchased))# Plot 2
```



Plot 1



Plot 2

9. Install and activate "arules" package.

#arules package is a powerful tool for mining associative rules in transactional databases. The most common use of arules package is market basket analysis in marketing and retail.

```
install.packages("arules")     # install "arules" package.
library(arules)                # activate "arules" package
```

10. Use the following code to create Association rules .

```
#The apriori () function from the arule package implements the
Apriori algorithm to create frequent itemsets.
#Note that, by default, the apriori () function executes all the
iterations at once.
```

```
#Usage of apriori () function
```

```
apriori(data, parameter = …, appearance = …)
```

```
Arguments
```

data - object of class transactions or any data structure which can be coerced into transactions (e.g., a binary matrix or data.frame).

parameter - named list. The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.8, maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).

appearance - named list. With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.

```
rules <- apriori(marketbasket)
```

Results : It shows that 68880 rules were generated using this line of code.

```
> rules<-apriori(marketbasket)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules
   ext
 FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 100

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[28 item(s), 1000 transaction(s)] done [0.00s].
sorting and recoding items ... [26 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.02s].
writing ... [68880 rule(s)] done [0.05s].
creating S4 object  ... done [0.03s].
Warning message:
In apriori(marketbasket) :
  Mining stopped (maxlen reached). Only patterns up to a length of 10 returned!
>
```

11. Get the summary of these rules

```
summary(rules)
```

```
> summary(rules)
set of 68880 rules

rule length distribution (lhs + rhs):sizes
    1     2     3     4     5     6     7     8     9    10
    3    85   942  4350 10739 17062 18066 11996  4665   972

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   6.000   7.000   6.542   8.000  10.000

summary of quality measures:
    support         confidence          lift            count
 Min.   :0.1000   Min.   :0.8000   Min.   :0.8781   Min.   :100.0
 1st Qu.:0.1150   1st Qu.:0.8667   1st Qu.:1.0389   1st Qu.:115.0
 Median :0.1370   Median :0.9453   Median :1.1565   Median :137.0
 Mean   :0.1583   Mean   :0.9259   Mean   :1.2019   Mean   :158.3
 3rd Qu.:0.1770   3rd Qu.:0.9821   3rd Qu.:1.2438   3rd Qu.:177.0
 Max.   :0.9580   Max.   :1.0000   Max.   :3.5714   Max.   :958.0

mining info:
         data ntransactions support confidence
 marketbasket          1000     0.1        0.8
>
```

```
#The result tells you that there was 3 rules with 1 item and 17062
rules with 6 items, etc….
```

12. To inspect the rules please use the following line of code:

```
inspect(rules)
#inspect function prints the internal representation of an R object
```

13. Since there are too many rules we need to reduce them into smaller number of rules hence we have to specify the parameters using the following code:

```
# When the max len parameter is not set, the algorithm continues
each iteration until it runs out of support or until k reaches the
default maxlen=10.

#set the  minlen=2,maxlen=3 and confident  = 0.95
```

```
rules <- apriori(marketbasket,
                 parameter =list(minlen=2,maxlen=3, conf = 0.95))
```

Results : #This has reduced the rules to 350

```
> rules <- apriori(marketbasket, parameter = list(minlen=2,maxlen=3, conf = 0.95))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
       0.95    0.1    1 none FALSE            TRUE       5     0.1      2      3  rules
   ext
 FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 100

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[28 item(s), 1000 transaction(s)] done [0.01s].
sorting and recoding items ... [26 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.02s].
writing ... [350 rule(s)] done [0.00s].
creating S4 object  ... done [0.03s].
```

14. Get the summary of these rules:

```
summary(rules)
```

```
# The summary of the rules shows the number of rules and ranges of
  the support, confidence, and lift.
```

```
> summary(rules)
set of 350 rules

rule length distribution (lhs + rhs):sizes
  2   3 
 31 319 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   3.000   3.000   2.911   3.000   3.000 

summary of quality measures:
    support          confidence          lift            count      
 Min.   :0.1000   Min.   :0.9500   Min.   :0.9919   Min.   :100.0  
 1st Qu.:0.2402   1st Qu.:0.9589   1st Qu.:1.0068   1st Qu.:240.2  
 Median :0.3565   Median :0.9660   Median :1.0163   Median :356.5  
 Mean   :0.3710   Mean   :0.9687   Mean   :1.0664   Mean   :371.0  
 3rd Qu.:0.4820   3rd Qu.:0.9760   3rd Qu.:1.0250   3rd Qu.:482.0  
 Max.   :0.9090   Max.   :1.0000   Max.   :3.5714   Max.   :909.0  

mining info:
         data ntransactions support confidence
 marketbasket          1000     0.1       0.95
```

15. Inspect the rules:

```
inspect(rules)
```

Results

```
> inspect(rules) #inspect prints the internal representation of an R object (or the result of
an expression).
      lhs                             rhs            support confidence lift      count
[1]   {turkey=Yes}                 => {coke=Yes}     0.149   1.0000000  3.5714286 149
[2]   {soda=Yes}                   => {ice_cream=No} 0.224   0.9572650  0.9992327 224
[3]   {artichoke=Yes}              => {apples=No}    0.306   0.9503106  1.0045566 306
[4]   {bourbon=Yes}                => {ice_cream=No} 0.420   0.9502262  0.9918854 420
[5]   {sardines=Yes}               => {ice_cream=No} 0.438   0.9584245  1.0004431 438
[6]   {cosmetics=No}               => {apples=No}    0.441   0.9504310  1.0046840 441
[7]   {cosmetics=No}               => {ice_cream=No} 0.441   0.9504310  0.9920992 441
[8]   {avocado=Yes}                => {ice_cream=No} 0.474   0.9673469  1.0097567 474
[9]   {avocado=No}                 => {apples=No}    0.486   0.9529412  1.0073374 486
[10]  {cosmetics=Yes}              => {ice_cream=No} 0.517   0.9645522  1.0068395 517
[11]  {sardines=No}                => {apples=No}    0.518   0.9539595  1.0084138 518
[12]  {sardines=No}                => {ice_cream=No} 0.520   0.9576427  0.9996271 520
```

```
#Rules that have No values are not useful
```

16. Since we are more interested what customers are purchasing, hence we need rules with "Yes".

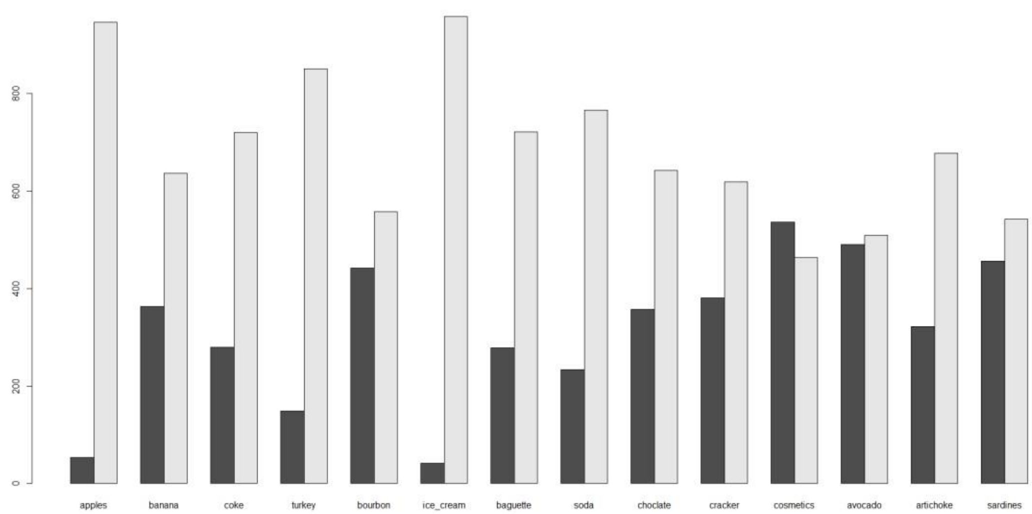
```
#First of all, we should find the most popular products based on sales dataset.

summary(marketbasket)
```

```
> summary(marketbasket)
 apples     banana     coke       turkey     bourbon    ice_cream baguette    soda
 No :946    No :637    No :720    No :851    No :558    No :958    No :721    No :766
 Yes: 54    Yes:363    Yes:280    Yes:149    Yes:442    Yes: 42    Yes:279    Yes:234
 choclate   cracker    cosmetics  avocado    artichoke  sardines
 No :643    No :619    No :464    No :510    No :678    No :543
 Yes:357    Yes:381    Yes:536    Yes:490    Yes:322    Yes:457
>
```

```
#Plotting could be the easiest way to find the most purchased item.

barplot(purchased, beside=T,legend=rownames(purchased))
```




```
#According to the plot cosmetics are the most popular items. Since we want to see rules where customers are buying more of the items, so use the following code to get those rules for cosmetics:

rules <- apriori(marketbasket,
            parameter = list(minlen=2, maxlen=3,conf = 0.95),
            appearance= list(rhs=c("cosmetics=Yes"),default="lhs"))
summary(rules)
```

```
inspect(rules)
```

Result: # There are no rules available for given parameter values

```
> rules <- apriori(marketbasket, parameter = list(minlen=2,maxlen=3, conf = 0.95),
+                  appearance =list(rhs=c("cosmetics=Yes"),default="lhs") )
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
       0.95    0.1    1 none FALSE            TRUE       5     0.1      2      3  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 100

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[28 item(s), 1000 transaction(s)] done [0.00s].
sorting and recoding items ... [26 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [0 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning message:
In apriori(marketbasket, parameter = list(minlen = 2, maxlen = 3,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
>
> summary(rules)
set of 0 rules

> inspect(rules)
```

```
#Change the confident parameter value to 70% (0.70), A lower
confidence threshold allows more rules to show up.

rules <- apriori(marketbasket,
                 parameter = list(minlen=2, maxlen=3,conf = 0.70),
                 appearance= list(rhs=c("cosmetics=Yes"),default="lhs"))
summary(rules)
inspect(rules)
```

Result: # 16 rules available for given parameters

```
> rules <- apriori(marketbasket, parameter = list(minlen=2,maxlen=3, conf = 0.70),
+                 appearance =list(rhs=c("cosmetics=Yes"),default="lhs") )
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
        0.7    0.1    1 none FALSE            TRUE       5     0.1      2      3  rules
   ext
 FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 100

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[28 item(s), 1000 transaction(s)] done [0.00s].
sorting and recoding items ... [26 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [16 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

17. Inspect these rules:

```
inspect(rules)
```

```
> inspect(rules)
     lhs                              rhs             support confidence lift     count
[1]  {avocado=Yes}                 => {cosmetics=Yes} 0.356   0.7265306  1.355468 356
[2]  {avocado=Yes,artichoke=Yes}   => {cosmetics=Yes} 0.116   0.7341772  1.369734 116
[3]  {choclate=Yes,avocado=Yes}    => {cosmetics=Yes} 0.130   0.7182320  1.339985 130
[4]  {cracker=Yes,avocado=Yes}     => {cosmetics=Yes} 0.146   0.7263682  1.355164 146
[5]  {avocado=Yes,sardines=No}     => {cosmetics=Yes} 0.200   0.7604563  1.418762 200
[6]  {bourbon=No,avocado=Yes}      => {cosmetics=Yes} 0.215   0.7904412  1.474704 215
[7]  {cracker=No,avocado=Yes}      => {cosmetics=Yes} 0.210   0.7266436  1.355678 210
[8]  {banana=No,avocado=Yes}       => {cosmetics=Yes} 0.237   0.7596154  1.417193 237
[9]  {choclate=No,avocado=Yes}     => {cosmetics=Yes} 0.226   0.7313916  1.364537 226
[10] {avocado=Yes,artichoke=No}    => {cosmetics=Yes} 0.240   0.7228916  1.348678 240
[11] {coke=No,avocado=Yes}         => {cosmetics=Yes} 0.267   0.7500000  1.399254 267
[12] {baguette=No,avocado=Yes}     => {cosmetics=Yes} 0.295   0.8452722  1.577000 295
[13] {soda=No,avocado=Yes}         => {cosmetics=Yes} 0.310   0.8288770  1.546412 310
[14] {turkey=No,avocado=Yes}       => {cosmetics=Yes} 0.313   0.7417062  1.383780 313
[15] {apples=No,avocado=Yes}       => {cosmetics=Yes} 0.335   0.7282609  1.358696 335
[16] {ice_cream=No,avocado=Yes}    => {cosmetics=Yes} 0.345   0.7278481  1.357926 345
```

```
#shows that when a customer buy avocado also buy cosmetics
```

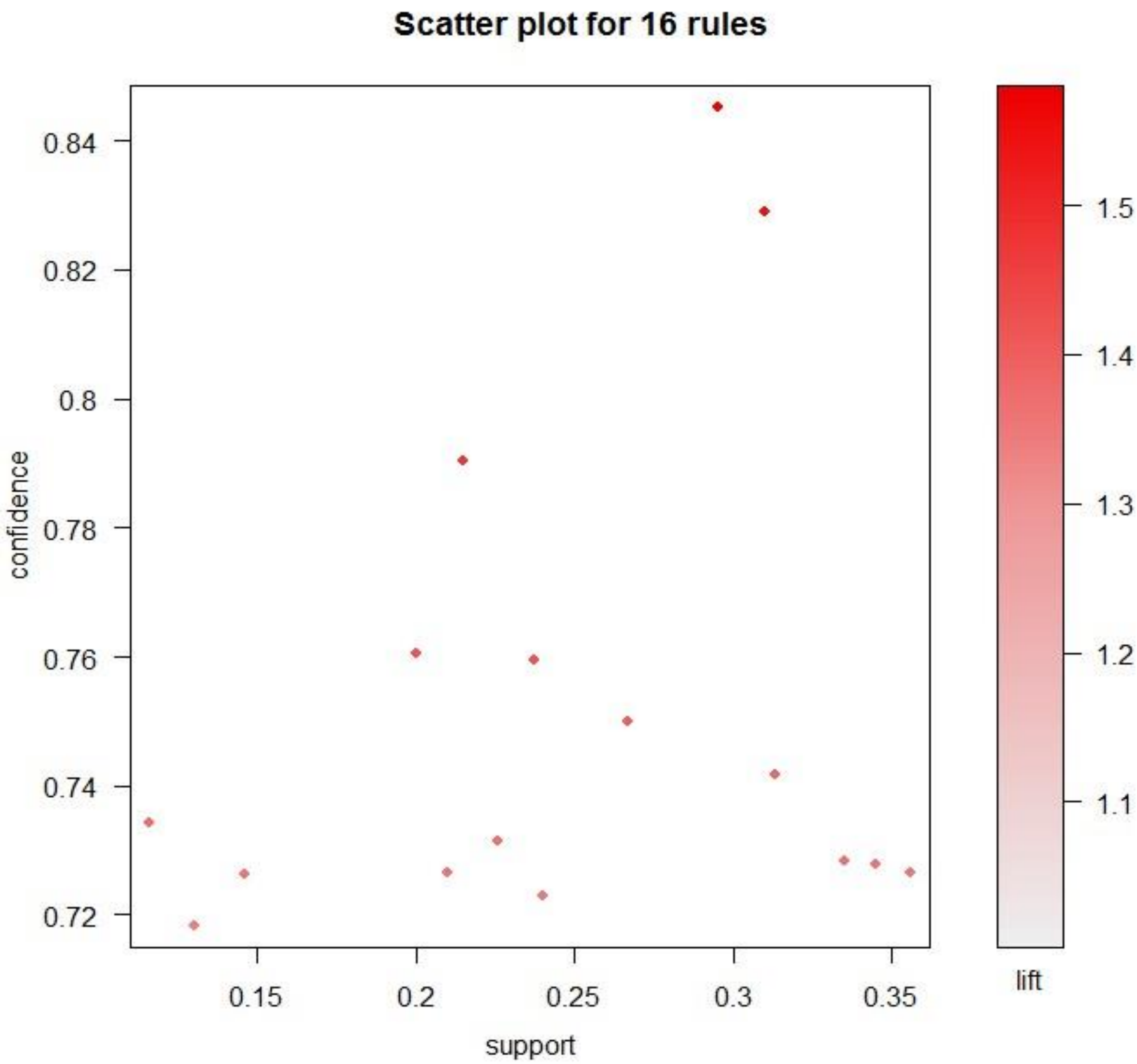
18. To Visualize these rules we can use a package called “**arulesViz**”

```
#arulesViz package provides various visualization techniques for
association rules and itemsets. The package also includes several
interactive visualizations for rule exploration.

install.packages(“arulesViz”)     # install “arulesViz”
library(arulesViz)                # activate “arules” package
```

19. Plot the rules.

```
plot(rules)
```

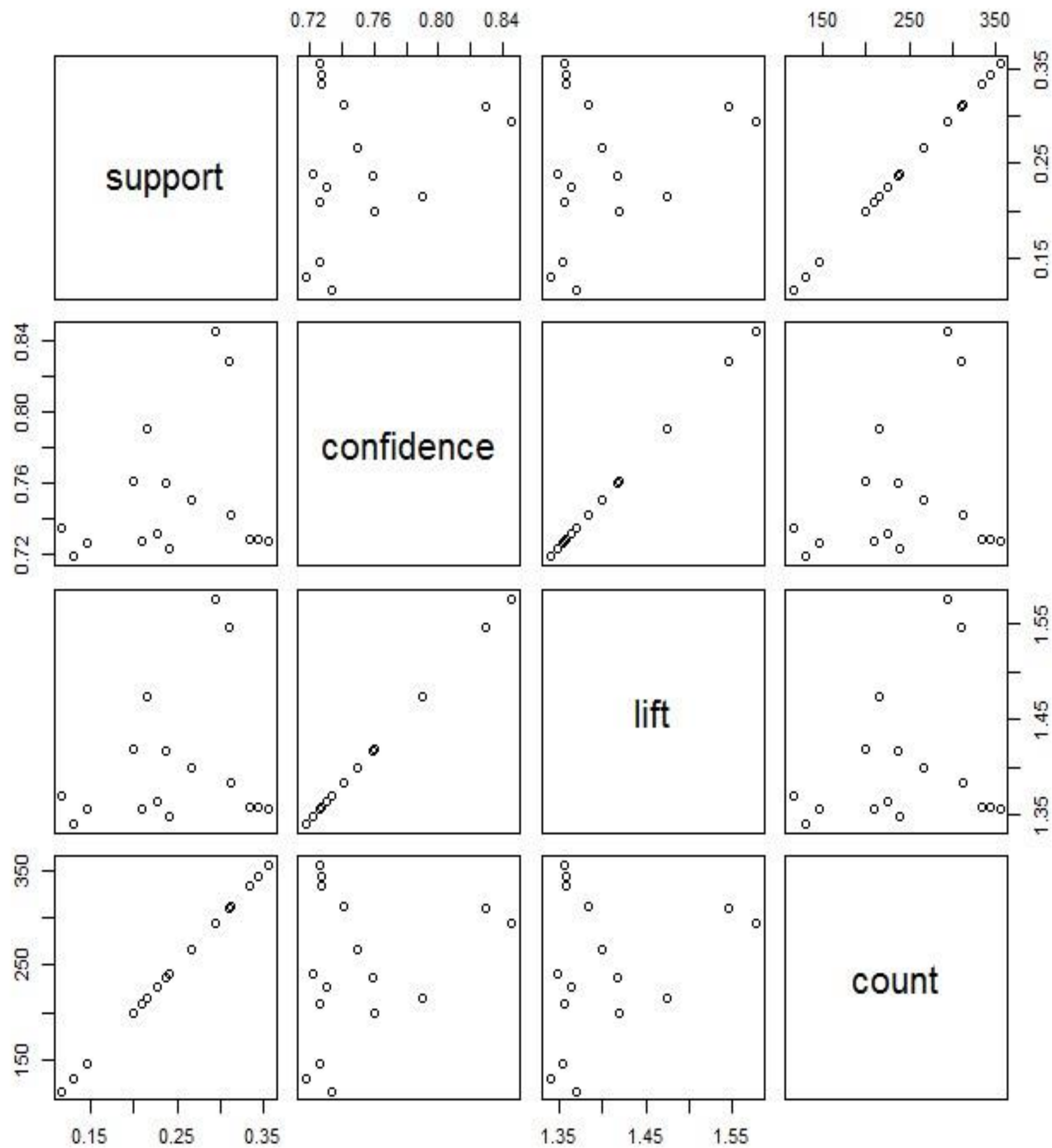


20.Use the following code to Plot the rules in groups:

```
plot(rules, method="grouped")
```

21. Below code displays a scatterplot matrix to compare the support, confidence, and lift

```
plot(rules@quality)
```

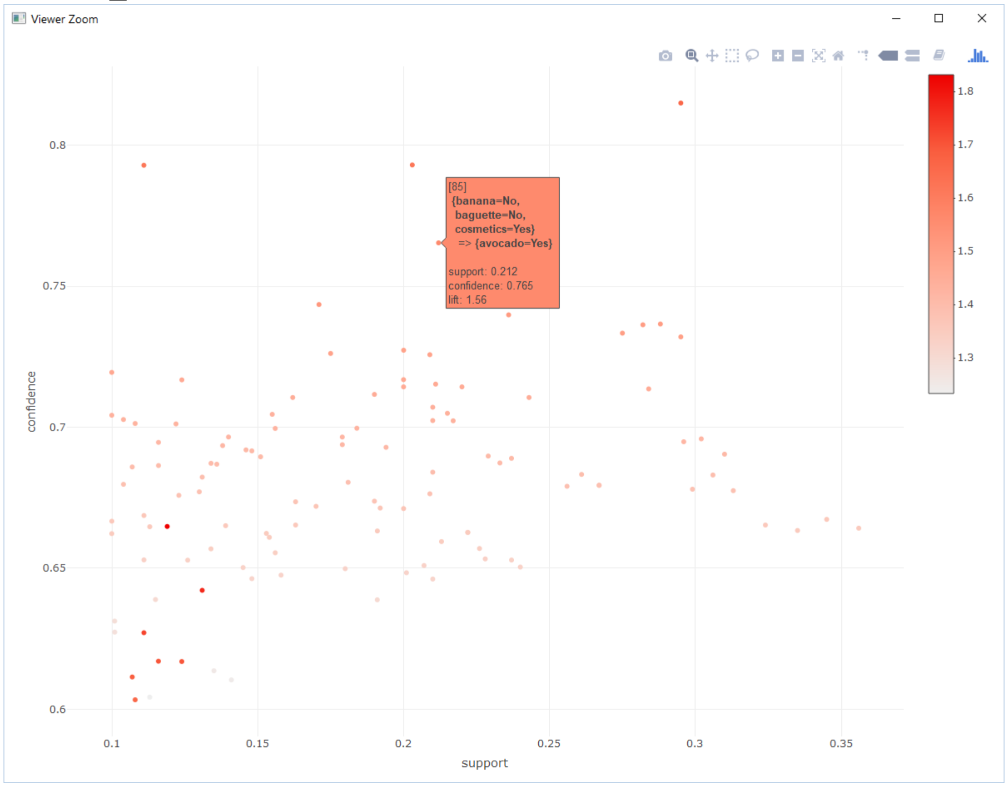


22. **plotly_arules()** function can be used to create Interactive Scatter Plot for Association Rules. You can hover over each rule and view all quality measures (support, confidence and lift).

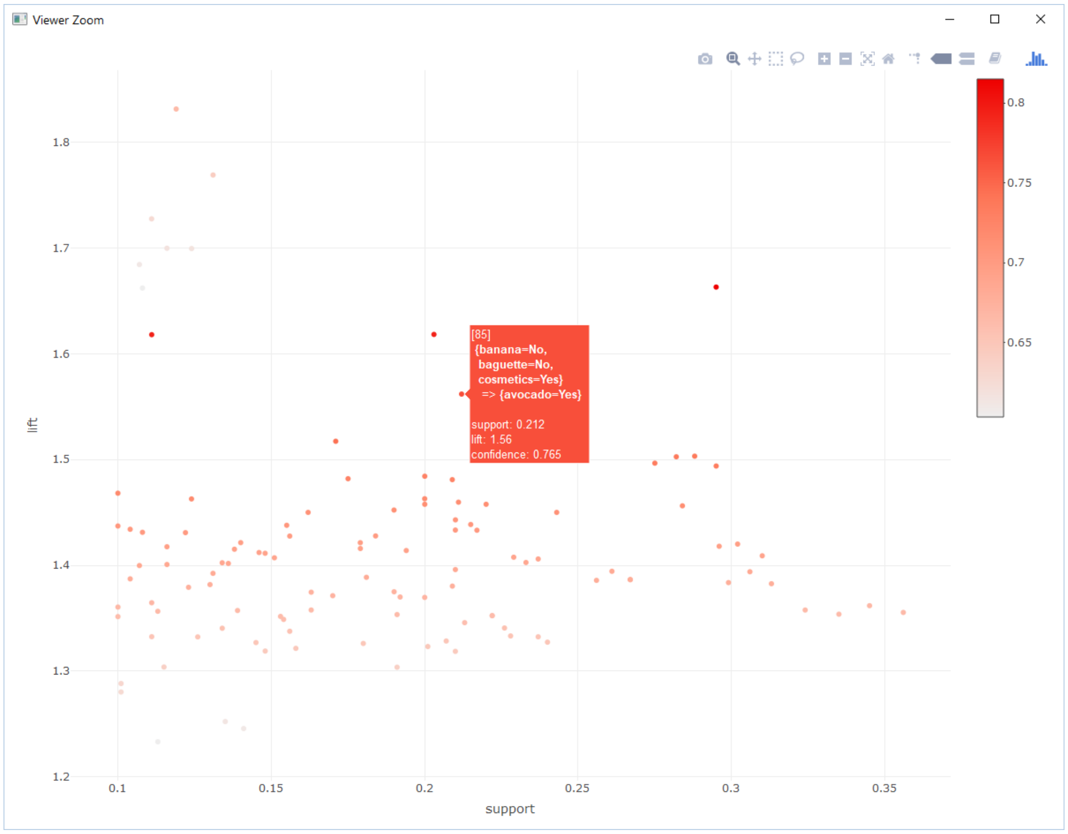
```
?plotly_arules

rules3 <-  apriori(marketbasket,
           parameter = list(minlen=2,maxlen=4, conf = 0.60),
           appearance =list(rhs=c("banana=Yes","apples=Yes","avocado=Yes")
                         ,default="lhs") )
```

```
plotly_arules(rules3)
```




```
plotly_arules(rules3, measure = c("support", "lift"), shading = "confidence")
```



23. In most cases we don’t need to see any rules which contain purchased items “No”. The fore use following line of code to get the rules with only purchased items “Yes” on left hand side and right-hand side:

```
rules2 <- apriori(marketbasket,
                  parameter = list(minlen=2, maxlen=3,conf = 0.7),
                  appearance =list(rhs=c("cosmetics=Yes"),
                                   lhs=c("apples=Yes",
                                         "banana=Yes",
                                         "coke=Yes",
                                         "turkey=Yes",
                                         "bourbon=Yes",
                                         "ice_cream=Yes",
                                         "baguette=Yes",
                                         "soda=Yes",
                                         "choclate=Yes",
                                         "cracker=Yes",
                                         "avocado=Yes",
                                         "sardines=Yes"),
                                   default="none"))
```

Result: 3 rules available for given parameters

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target
        0.7    0.1    1 none FALSE            TRUE       5     0.1      2      3  rules
   ext
 FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 100

set item appearances ...[13 item(s)] done [0.00s].
set transactions ...[13 item(s), 1000 transaction(s)] done [0.06s].
sorting and recoding items ... [11 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

24. Inspect the rules:

```
inspect(rules2)
```

Result:

```
> inspect(rules2)
    lhs                            rhs              support confidence lift  count
[1] {avocado=Yes}               => {cosmetics=Yes} 0.356   0.727      1.355 356
[2] {choclate=Yes,avocado=Yes} => {cosmetics=Yes} 0.130   0.718      1.340 130
[3] {cracker=Yes,avocado=Yes}  => {cosmetics=Yes} 0.146   0.726      1.355 146
```

25. Change the confident parameter value to 50% to see more rules. A lower confidence threshold allows more rules to show up.

26. You can do variety of practice on other purchased items as you wish with different confidence thresholds.

27. You can interactively explore Association Rules using **ruleExplorer()** function.

```
?ruleExplorer
```

Usage : `ruleExplorer(x, parameter = ….)`

Arguments :

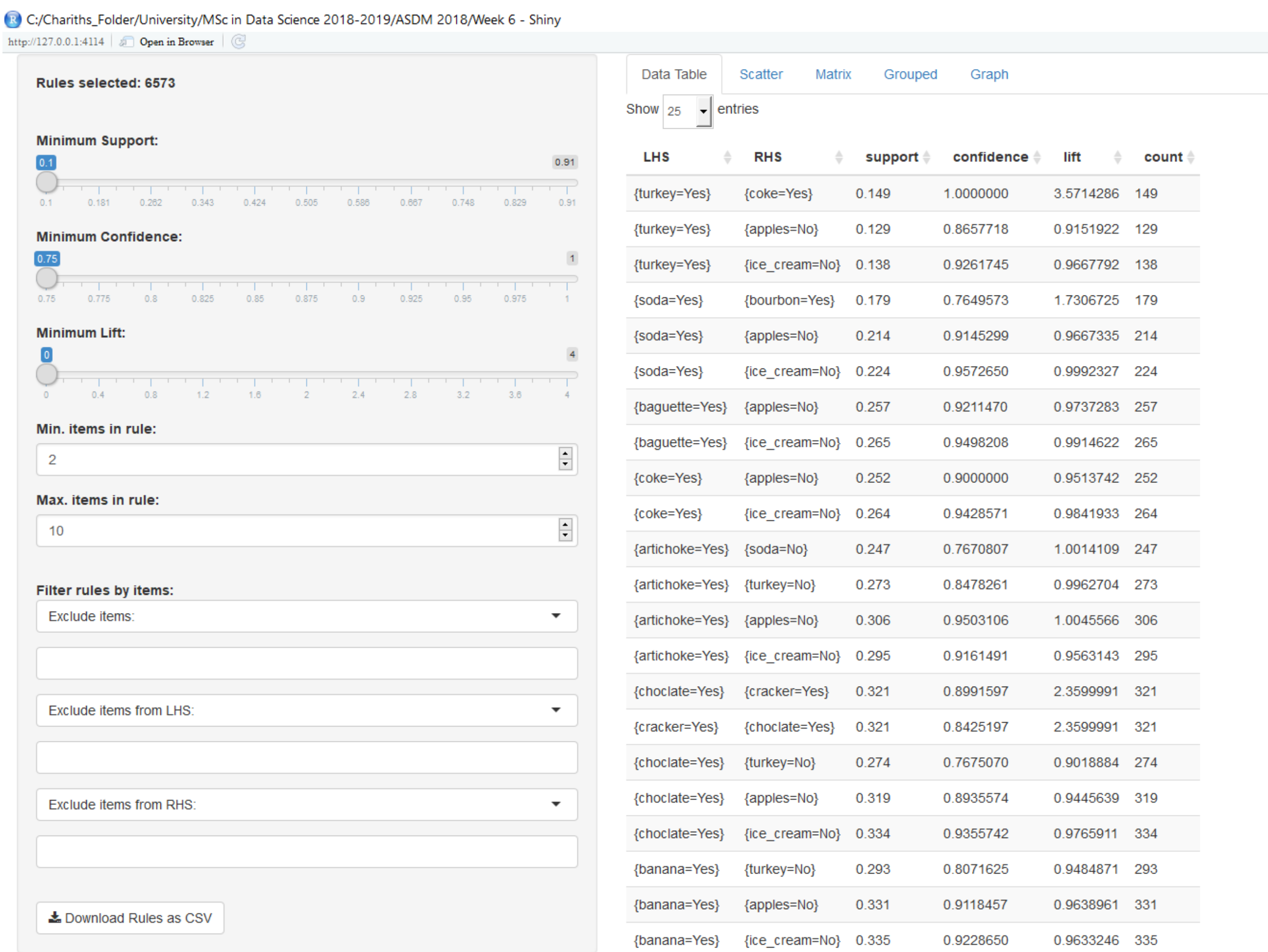
`x` a set of rules, a transactions object or a data.frame.

parameter a list with parameters passed on to apriori. the list can be used to set the initial support and confidence thresholds. Values are ignored if x contains a set of rules.

```
rules_ex <-apriori(marketbasket,
                   parameter =list(minlen=2,maxlen=4,conf=0.75))

#Explore association rules using interactive manipulations and visualization using shiny.

ruleExplorer(rules_ex)
```

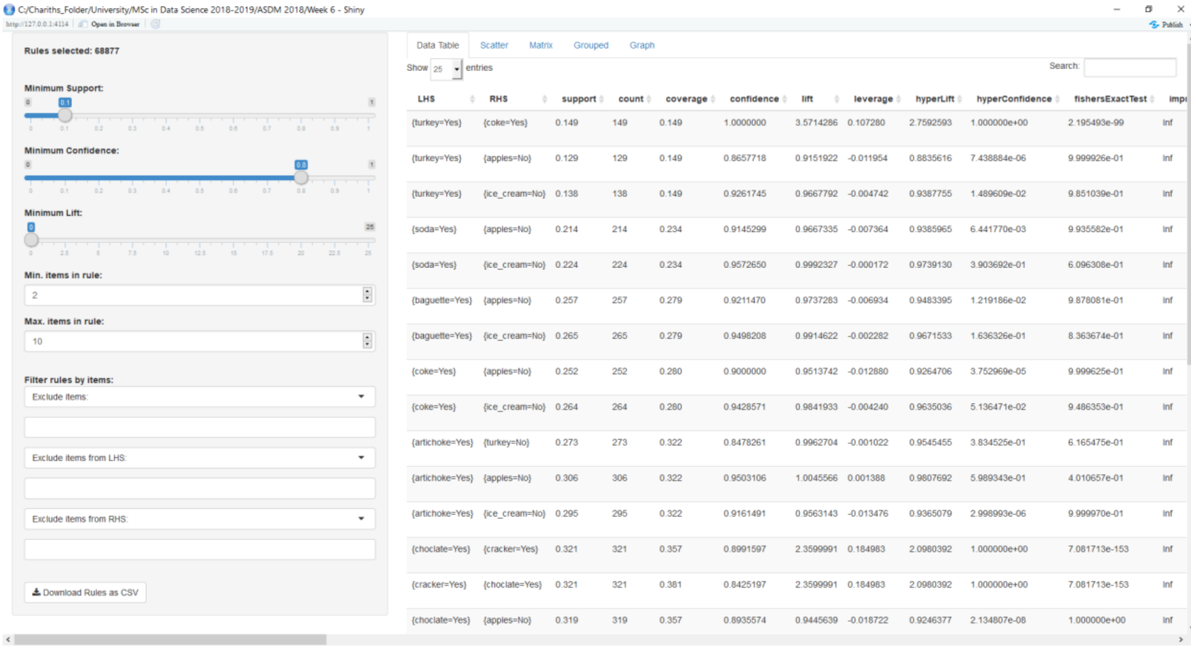


LHS	RHS	support	confidence	lift	count
{turkey=Yes}	{coke=Yes}	0.149	1.0000000	3.5714286	149
{turkey=Yes}	{apples=No}	0.129	0.8657718	0.9151922	129
{turkey=Yes}	{ice_cream=No}	0.138	0.9261745	0.9667792	138
{soda=Yes}	{bourbon=Yes}	0.179	0.7649573	1.7306725	179
{soda=Yes}	{apples=No}	0.214	0.9145299	0.9667335	214
{soda=Yes}	{ice_cream=No}	0.224	0.9572650	0.9992327	224
{baguette=Yes}	{apples=No}	0.257	0.9211470	0.9737283	257
{baguette=Yes}	{ice_cream=No}	0.265	0.9498208	0.9914622	265
{coke=Yes}	{apples=No}	0.252	0.9000000	0.9513742	252
{coke=Yes}	{ice_cream=No}	0.264	0.9428571	0.9841933	264
{artichoke=Yes}	{soda=No}	0.247	0.7670807	1.0014109	247
{artichoke=Yes}	{turkey=No}	0.273	0.8478261	0.9962704	273
{artichoke=Yes}	{apples=No}	0.306	0.9503106	1.0045566	306
{artichoke=Yes}	{ice_cream=No}	0.295	0.9161491	0.9563143	295
{choclate=Yes}	{cracker=Yes}	0.321	0.8991597	2.3599991	321
{cracker=Yes}	{choclate=Yes}	0.321	0.8425197	2.3599991	321
{choclate=Yes}	{turkey=No}	0.274	0.7675070	0.9018884	274
{choclate=Yes}	{apples=No}	0.319	0.8935574	0.9445639	319
{choclate=Yes}	{ice_cream=No}	0.334	0.9355742	0.9765911	334
{banana=Yes}	{turkey=No}	0.293	0.8071625	0.9484871	293
{banana=Yes}	{apples=No}	0.331	0.9118457	0.9638961	331
{banana=Yes}	{ice_cream=No}	0.335	0.9228650	0.9633246	335

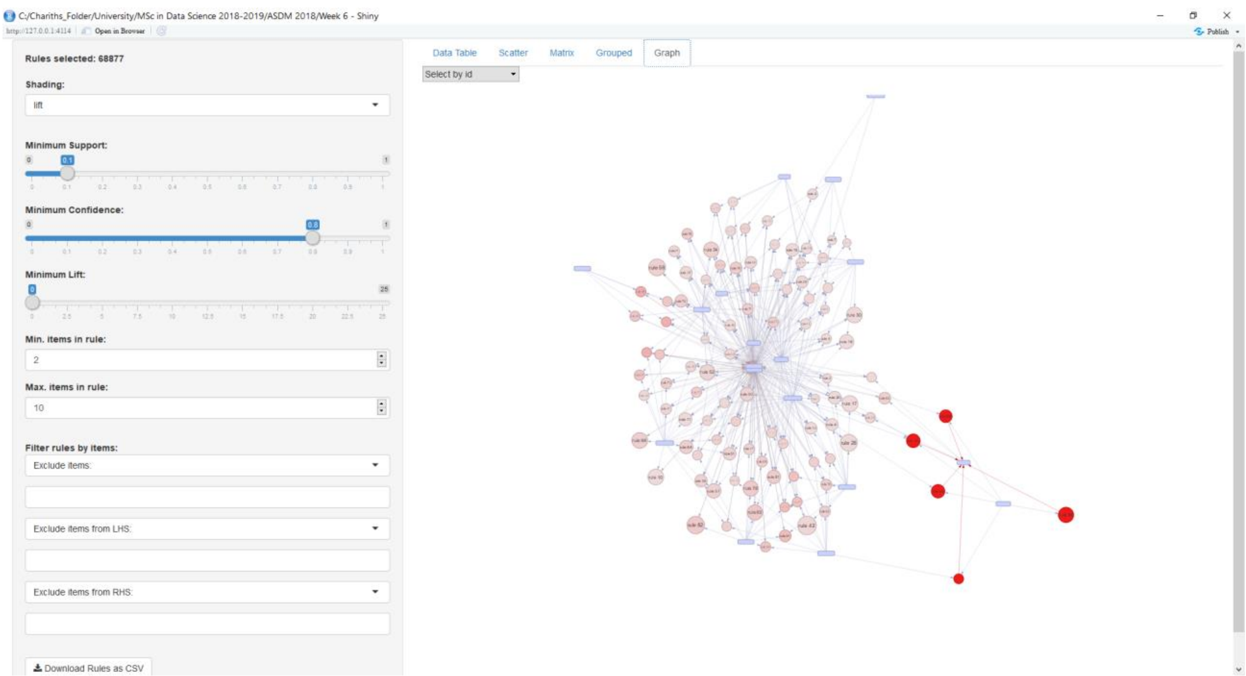
28. Mine and explore rules in given data sources on the fly using **ruleExplorer()** function

```
ruleExplorer(marketbasket)
```

Data Table tab



Graph Tab



Part 2: Exercise

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this exercise, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the Association Rule mining to predict which passengers survived from the tragedy.

titanic.csv data set can be downloaded from the Blackboard.

COLUMN DESCRIPTION:

Class (0 = crew, 1 = first, 2 = second, 3 = third)
Age (1 = adult, 0 = child)
Sex (1 = male, 0 = female)
Survived (1 = yes, 0 = no)

source : https://www.kaggle.com/c/titanic