**Assignment No-03**

**Code:**

> library(arules)

> library(arulesViz)

> library(datasets)

> # Load the data set

> data(Groceries)

>

> #Lets explore the data before we make any rules:

> # Create an item frequency plot for the top 20 items

> itemFrequencyPlot(Groceries,topN=20,type="absolute")

>

> #You will always have to pass the minimum required support and confidence.

> # We set the minimum support to 0.001

> # We set the minimum confidence of 0.8

> # Get the rules

> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target

0.8 0.1 1 none FALSE TRUE 5 0.001 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: 9

set item appearances ...[0 item(s)] done [0.00s].

set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].

sorting and recoding items ... [157 item(s)] done [0.00s].

creating transaction tree ... done [0.02s].

checking subsets of size 1 2 3 4 5 6 done [0.01s].

writing ... [410 rule(s)] done [0.00s].

creating S4 object ... done [0.00s].

>

> # Show the top 5 rules, but only 2 digits

> options(digits=2)

> inspect(rules[1:5])

lhs rhs support confidence lift count

[1] {liquor,red/blush wine} => {bottled beer} 0.0019 0.90 11.2 19

[2] {curd,cereals} => {whole milk} 0.0010 0.91 3.6 10

[3] {yogurt,cereals} => {whole milk} 0.0017 0.81 3.2 17

[4] {butter,jam} => {whole milk} 0.0010 0.83 3.3 10

[5] {soups,bottled beer} => {whole milk} 0.0011 0.92 3.6 11

> #Sorting Rules by confidence

> rules<-sort(rules, by="confidence", decreasing=TRUE)

> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,maxlen=3))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target

0.8 0.1 1 none FALSE TRUE 5 0.001 1 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: 9

set item appearances ...[0 item(s)] done [0.00s].

set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].

sorting and recoding items ... [157 item(s)] done [0.00s].

creating transaction tree ... done [0.00s].

checking subsets of size 1 2 3 done [0.01s].

writing ... [29 rule(s)] done [0.00s].

creating S4 object ... done [0.00s].

Warning message:

In apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8, :

Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!

>

> # Redundancies

> subset.matrix <- is.subset(rules, rules)

> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA

> redundant <- colSums(subset.matrix, na.rm=T) >= 1

> rules.pruned <- rules[!redundant]

> rules<-rules.pruned

> #Targeting items

> rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),

+ appearance = list(default="lhs",rhs="whole milk"),

+ control = list(verbose=F))

> rules<-sort(rules, decreasing=TRUE,by="confidence")

> inspect(rules[1:5])

lhs rhs support confidence lift

[1] {rice,sugar} => {whole milk} 0.0012 1 3.9

[2] {canned fish,hygiene articles} => {whole milk} 0.0011 1 3.9

[3] {root vegetables,butter,rice} => {whole milk} 0.0010 1 3.9

[4] {root vegetables,whipped/sour cream,flour} => {whole milk} 0.0017 1 3.9

[5] {butter,soft cheese,domestic eggs} => {whole milk} 0.0010 1 3.9

count

[1] 12

[2] 11

[3] 10

[4] 17

[5] 10

>

> rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),

+ appearance = list(default="rhs",lhs="whole milk"),

+ control = list(verbose=F))

> rules<-sort(rules, decreasing=TRUE,by="confidence")

> inspect(rules[1:5])

lhs rhs support confidence lift count

[1] {whole milk} => {other vegetables} 0.075 0.29 1.5 736

[2] {whole milk} => {rolls/buns} 0.057 0.22 1.2 557

[3] {whole milk} => {yogurt} 0.056 0.22 1.6 551

[4] {whole milk} => {root vegetables} 0.049 0.19 1.8 481

[5] {whole milk} => {tropical fruit} 0.042 0.17 1.6 416

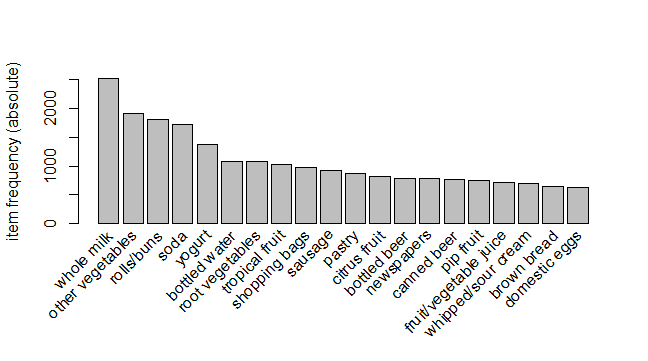
>

> #Visualization

> library(arulesViz)

> plot(rules,method="graph",interactive=TRUE,shading=NA)

**Output:**

****