install.packages("arules") **# install the package**

library("arules") **# invoke the package**

**#Visualizing Association Rules Package arulesViz supports visualization of**

**#association rules with scatter plot,**

**#graph, parallel coordinates plot, etc**

install.packages("arulesViz")

library("arulesViz") # for visualizing rules

**#importing data set**

Groceryy<-read.transactions(file.choose())

View(Groceryy)

head(Groceryy)

**## calculates support for frequent items**

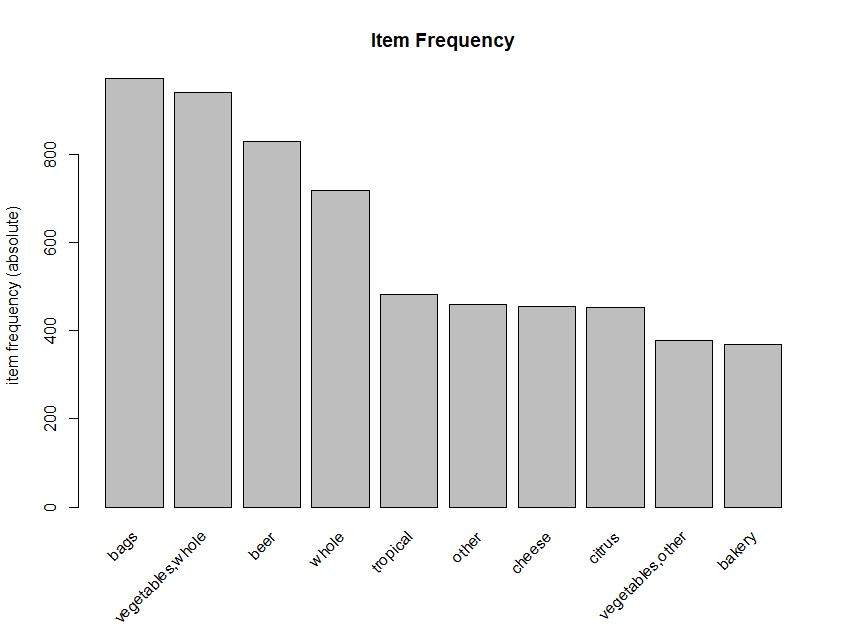
frequentItems <- eclat (Groceryy, parameter = list(supp = 0.07, maxlen = 15))

inspect(frequentItems)

|  |
| --- |
| items support transIdenticalToItemsets count  [1] {vegetables,whole} 0.09557702 940 940  [2] {bags} 0.09872903 971 971  [3] {beer} 0.08429080 829 829  [4] {whole} 0.07290290 717 717 |
|  |
| |  | | --- | | > | |

**# plot frequent items**

itemFrequencyPlot(Groceryy, topN=10, type="absolute", main="Item Frequency")



**# making rules using apriori algorithm**

**# Keep changing support and confidence values to obtain different rules**

**# Building rules using apriori algorithm**

Grocery\_Rule <- apriori(Groceryy, parameter = list(support=0.01,confidence=0.7))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext

0.7 0.1 1 none FALSE TRUE 5 0.01 1 10 rules TRUE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 98

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

set transactions ...[6928 item(s), 9835 transaction(s)] done [0.04s].

sorting and recoding items ... [46 item(s)] done [0.00s].

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

checking subsets of size 1 2 3 done [0.00s].

writing ... [7 rule(s)] done [0.00s].

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

Grocery\_Rule

set of 7 rules

**# to view the rules we use inspect**

inspect(head(sort(Grocery\_Rule,by="lift")))

**# here we got 7 rules, we have to increase the support value to get more rules**

**# to view the rules we use inspect**

inspect(head(sort(Grocery\_Rule,by="lift")))

lhs rhs support confidence coverage lift count

[1] {product} => {bakery} 0.01321810 1 0.01321810 26.72554 130

[2] {product} => {life} 0.01321810 1 0.01321810 26.72554 130

[3] {bakery} => {life} 0.03741739 1 0.03741739 26.72554 368

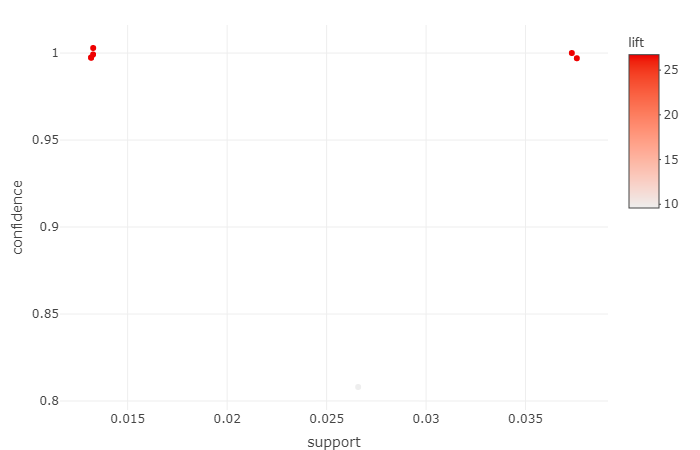
[4] {life} => {bakery} 0.03741739 1 0.03741739 26.72554 368

[5] {bakery,product} => {life} 0.01321810 1 0.01321810 26.72554 130

[6] {life,product} => {bakery} 0.01321810 1 0.01321810 26.72554 130

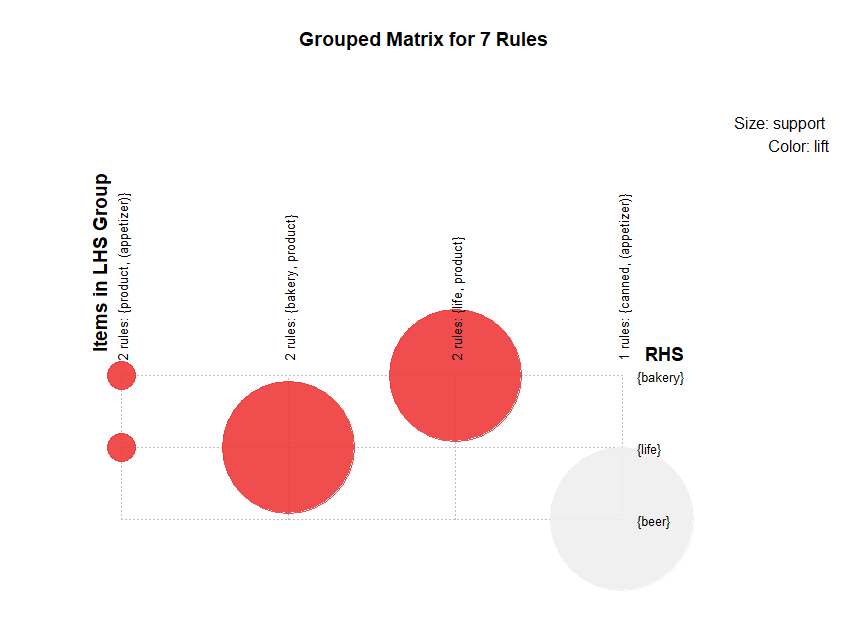
**#visualize the rules**

plotly\_arules(Grocery\_Rule)



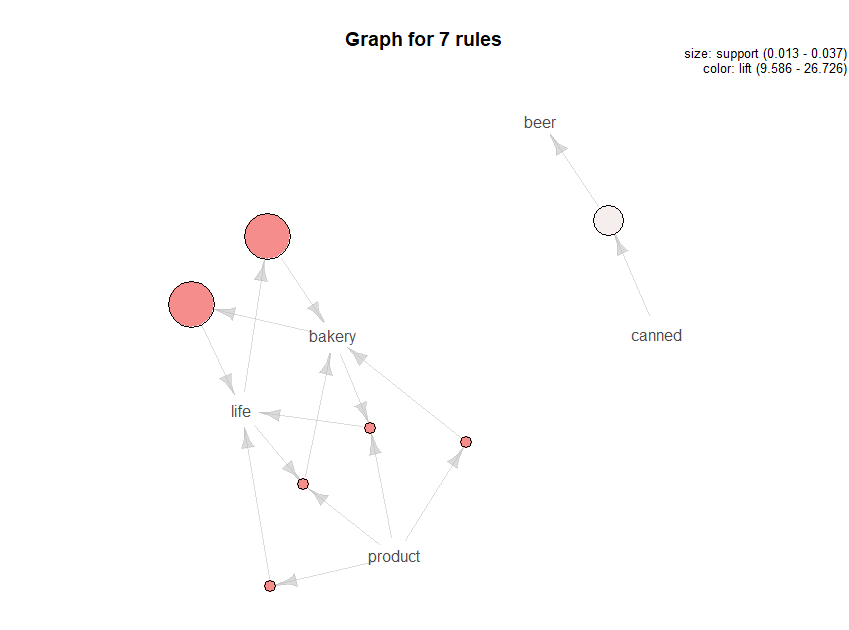
**##grouped matrix for association rules**

plot(Grocery\_Rule,method="grouped")



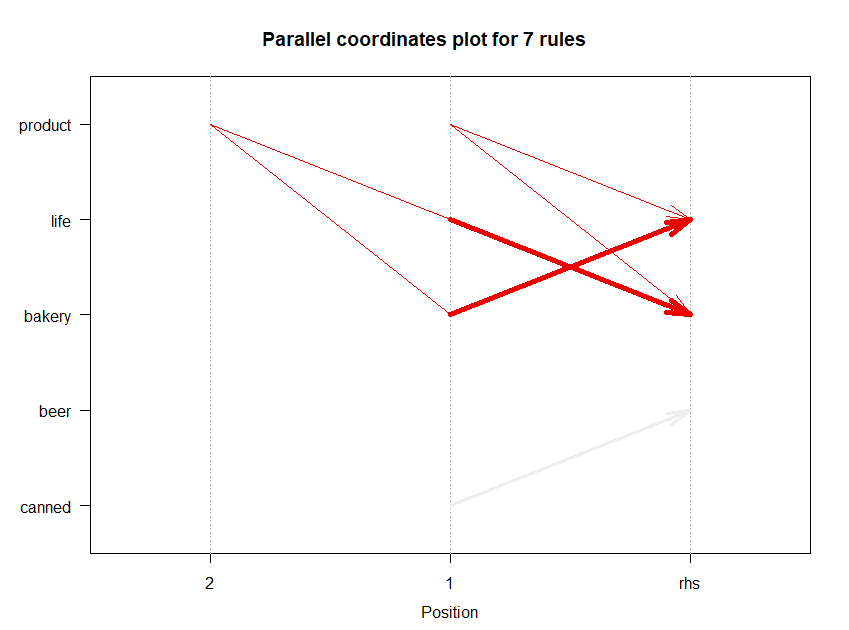
**#graph**

plot(Grocery\_Rule , method="graph", control=list(type="items"))



#**parallel coordinates plot**

plot(Grocery\_Rule, method="paracoord", control=list(reorder=TRUE))



#####################################################################

**# decrease support=0.001, confidence=0**.8

Grocery\_Rule1 <- apriori(Groceryy, parameter = list(support=0.001,confidence=0.8))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 5 0.001 1 10 rules TRUE

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 ...[6928 item(s), 9835 transaction(s)] done [0.04s].

sorting and recoding items ... [483 item(s)] done [0.00s].

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

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

writing ... [196 rule(s)] done [0.00s].

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

> Grocery\_Rule1

set of 196 rules

Grocery\_Rule1

**#display first few rules by highest lift ratios**

inspect(head(sort(Grocery\_Rule1,by="lift")))

lhs rhs support confidence coverage lift count

[1] {bread,Instant} => {food} 0.001016777 1.0000000 0.001016777 74.50758 10

[2] {products,canned} => {food} 0.001220132 0.9230769 0.001321810 68.77622 12

[3] {pot} => {plants} 0.001016777 0.9090909 0.001118454 59.21132 10

[4] {sparkling} => {wine} 0.001016777 0.9090909 0.001118454 41.39310 10

[5] {vegetables,roll} => {products} 0.001220132 1.0000000 0.001220132 31.52244 12

[6] {bread,roll} => {products} 0.001423488 1.0000000 0.001423488 31.52244 14

**#display fisr 10 rules with highest lift ratio**

inspect(sort(Grocery\_Rule1[1:10],by="lift"))

lhs rhs support confidence coverage lift count

[1] {pot} => {plants} 0.001016777 0.9090909 0.001118454 59.21132 10

[2] {sparkling} => {wine} 0.001016777 0.9090909 0.001118454 41.39310 10

[3] {,rolls/buns} => {cheese} 0.001118454 1.0000000 0.001118454 21.66300 11

[4] {,spread} => {cheese} 0.001118454 1.0000000 0.001118454 21.66300 11

[5] {vegetables,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

[6] {beverages,shopping} => {bags} 0.001016777 1.0000000 0.001016777 10.12873 10

[7] {food,shopping} => {bags} 0.001016777 1.0000000 0.001016777 10.12873 10

[8] {film/bags,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

[9] {care,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

[10] {bread,shopping} => {bags} 0.001423488 1.0000000 0.001423488 10.12873 14

**#lets check presence of redundant rules**

Grocery\_Rule1

set of 196 rules

redundant\_rules<-is.redundant(Grocery\_Rule1)

redundant\_rules

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[15] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[29] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[43] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[57] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

[71] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[85] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[99] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[113] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE

[127] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE

[141] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[155] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[169] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[183] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE

summary(redundant\_rules)

summary(redundant\_rules)

Mode FALSE TRUE

logical 76 120

**# here we can see that TRUE value 120, means that 120 values are redundant**

**# we have to remove all redundant values**

Grocery\_Rule1<-Grocery\_Rule1[!redundant\_rules]

Grocery\_Rule1

set of 76 rules

**# we can see that there we have 76 rules reamins**

inspect(sort(Grocery\_Rule1[1:10],by="lift"))

lhs rhs support confidence coverage lift count

[1] {pot} => {plants} 0.001016777 0.9090909 0.001118454 59.21132 10

[2] {sparkling} => {wine} 0.001016777 0.9090909 0.001118454 41.39310 10

[3] {,rolls/buns} => {cheese} 0.001118454 1.0000000 0.001118454 21.66300 11

[4] {,spread} => {cheese} 0.001118454 1.0000000 0.001118454 21.66300 11

[5] {vegetables,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

[6] {beverages,shopping} => {bags} 0.001016777 1.0000000 0.001016777 10.12873 10

[7] {food,shopping} => {bags} 0.001016777 1.0000000 0.001016777 10.12873 10

[8] {film/bags,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

[9] {care,shopping} => {bags} 0.001118454 1.0000000 0.001118454 10.12873 11

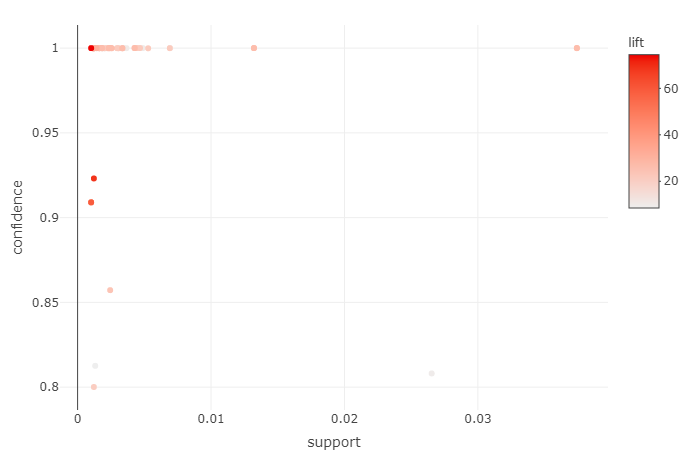
[10] {bread,shopping} => {bags} 0.001423488 1.0000000 0.001423488 10.12873 14

**# lhs(left hand side) means the product that is customer going to buy,**

**#rhs(right hand side) indicates the product to reccommend to customer**

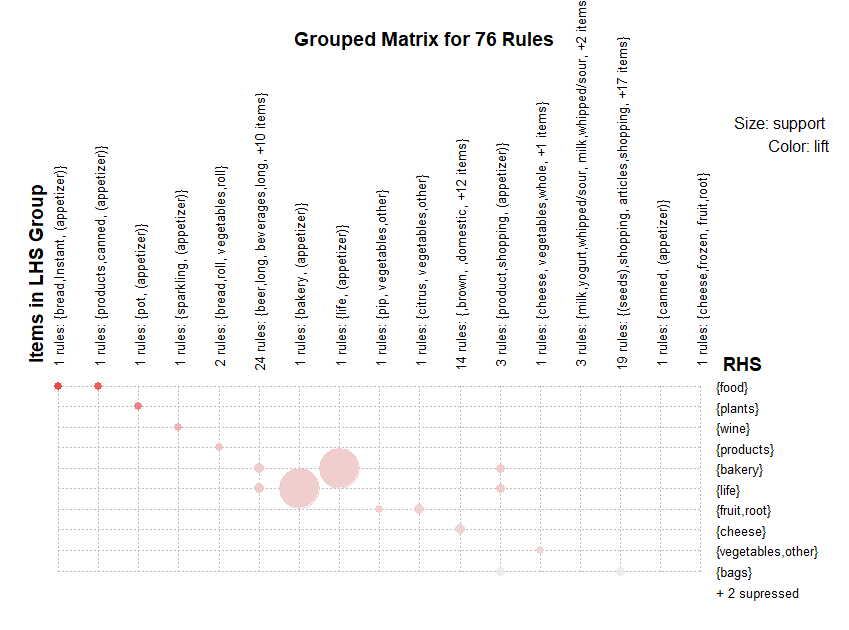
**#visualize the rules**

plotly\_arules(Grocery\_Rule1)



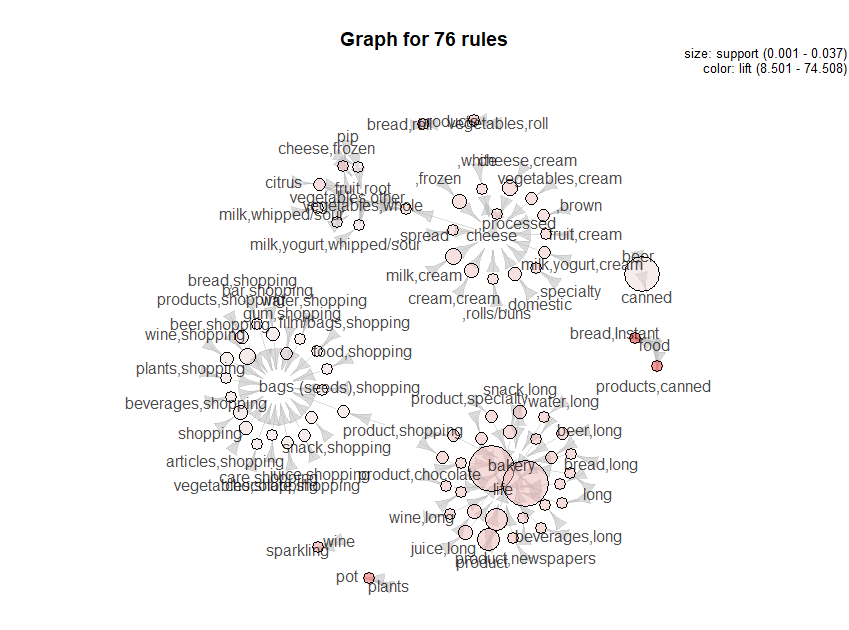
**##grouped matrix for association rules**

plot(Grocery\_Rule1,method="grouped")



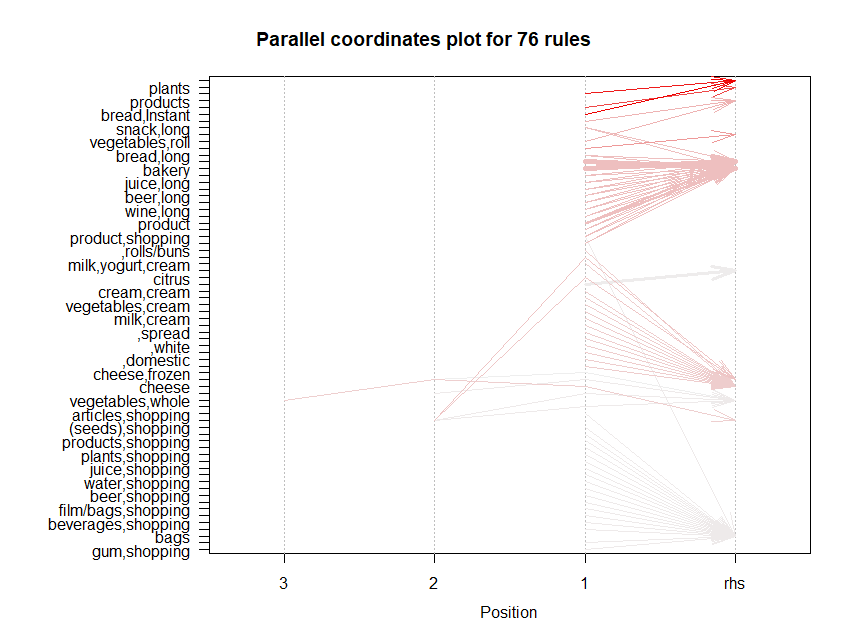
**#graph**

plot(Grocery\_Rule1 , method="graph", control=list(type="items"))



**#parallel coordinates plot**

plot(Grocery\_Rule1, method="paracoord", control=list(reorder=TRUE))



###################################################################

**#lets change support value .0001, and confidence=.85**

Grocery\_Rule2 <- apriori(Groceryy, parameter = list(support=0.003,confidence=0.85))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext

0.85 0.1 1 none FALSE TRUE 5 0.003 1 10 rules TRUE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 29

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

set transactions ...[6928 item(s), 9835 transaction(s)] done [0.03s].

sorting and recoding items ... [173 item(s)] done [0.00s].

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

checking subsets of size 1 2 3 done [0.00s].

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

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

Grocery\_Rule2

set of 29 rules

**#display the rules**

inspect(Grocery\_Rule2)

lhs rhs support confidence coverage lift count

[1] {shopping} => {bags} 0.004880529 1 0.004880529 10.12873 48

[2] {wine,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[3] {bar,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[4] {products,shopping} => {bags} 0.003660397 1 0.003660397 10.12873 36

[5] {articles,shopping} => {bags} 0.003457041 1 0.003457041 10.12873 34

[6] {beer,shopping} => {bags} 0.006914082 1 0.006914082 10.12873 68

[7] {,domestic} => {cheese} 0.003050330 1 0.003050330 21.66300 30

[8] {snack,long} => {bakery} 0.003355363 1 0.003355363 26.72554 33

[9] {snack,long} => {life} 0.003355363 1 0.003355363 26.72554 33

[10] {juice,long} => {bakery} 0.004270463 1 0.004270463 26.72554 42

[11] {juice,long} => {life} 0.004270463 1 0.004270463 26.72554 42

[12] {cream,cream} => {cheese} 0.004473818 1 0.004473818 21.66300 44

[13] {milk,cream} => {cheese} 0.006914082 1 0.006914082 21.66300 68

[14] {,frozen} => {cheese} 0.004677173 1 0.004677173 21.66300 46

[15] {cheese,cream} => {cheese} 0.005287239 1 0.005287239 21.66300 52

[16] {product} => {bakery} 0.013218099 1 0.013218099 26.72554 130

[17] {product} => {life} 0.013218099 1 0.013218099 26.72554 130

[18] {bakery} => {life} 0.037417387 1 0.037417387 26.72554 368

[19] {life} => {bakery} 0.037417387 1 0.037417387 26.72554 368

[20] {bakery,snack,long} => {life} 0.003355363 1 0.003355363 26.72554 33

[21] {life,snack,long} => {bakery} 0.003355363 1 0.003355363 26.72554 33

[22] {bakery,juice,long} => {life} 0.004270463 1 0.004270463 26.72554 42

[23] {juice,long,life} => {bakery} 0.004270463 1 0.004270463 26.72554 42

[24] {bakery,product} => {life} 0.013218099 1 0.013218099 26.72554 130

[25] {life,product} => {bakery} 0.013218099 1 0.013218099 26.72554 130

[26] {bags,bakery} => {life} 0.005388917 1 0.005388917 26.72554 53

[27] {bags,life} => {bakery} 0.005388917 1 0.005388917 26.72554 53

[28] {bakery,vegetables,whole} => {life} 0.006609049 1 0.006609049 26.72554 65

[29] {life,vegetables,whole} => {bakery} 0.006609049 1 0.006609049 26.72554 65

**#display fisr 10 rules with highest lift ratio**

inspect(sort(Grocery\_Rule[1:10],by="lift"))

lhs rhs support confidence coverage lift count

[1] {snack,long} => {bakery} 0.003355363 1 0.003355363 26.72554 33

[2] {snack,long} => {life} 0.003355363 1 0.003355363 26.72554 33

[3] {juice,long} => {bakery} 0.004270463 1 0.004270463 26.72554 42

[4] {,domestic} => {cheese} 0.003050330 1 0.003050330 21.66300 30

[5] {shopping} => {bags} 0.004880529 1 0.004880529 10.12873 48

[6] {wine,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[7] {bar,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[8] {products,shopping} => {bags} 0.003660397 1 0.003660397 10.12873 36

[9] {articles,shopping} => {bags} 0.003457041 1 0.003457041 10.12873 34

[10] {beer,shopping} => {bags} 0.006914082 1 0.006914082 10.12873 68

**#lets check presence of redundant rules**

Grocery\_Rule2

set of 29 rules

redundant\_rules2<-is.redundant(Grocery\_Rule2)

redundant\_rules2

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[15] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[29] TRUE

summary(redundant\_rules2)

summary(redundant\_rules2)

Mode FALSE TRUE

logical 19 10

**# here we can see that TRUE vale 10, means that 10 values are redundant**

**# we have to remove all redundant values**

Grocery\_Rule2<-Grocery\_Rule2[!redundant\_rules2]

Grocery\_Rule2

set of 19 rules

**# we can see that there we have 76 rules remains**

inspect(sort(Grocery\_Rule2[1:10],by="lift"))

lhs rhs support confidence coverage lift count

[1] {snack,long} => {bakery} 0.003355363 1 0.003355363 26.72554 33

[2] {snack,long} => {life} 0.003355363 1 0.003355363 26.72554 33

[3] {juice,long} => {bakery} 0.004270463 1 0.004270463 26.72554 42

[4] {,domestic} => {cheese} 0.003050330 1 0.003050330 21.66300 30

[5] {shopping} => {bags} 0.004880529 1 0.004880529 10.12873 48

[6] {wine,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[7] {bar,shopping} => {bags} 0.003050330 1 0.003050330 10.12873 30

[8] {products,shopping} => {bags} 0.003660397 1 0.003660397 10.12873 36

[9] {articles,shopping} => {bags} 0.003457041 1 0.003457041 10.12873 34

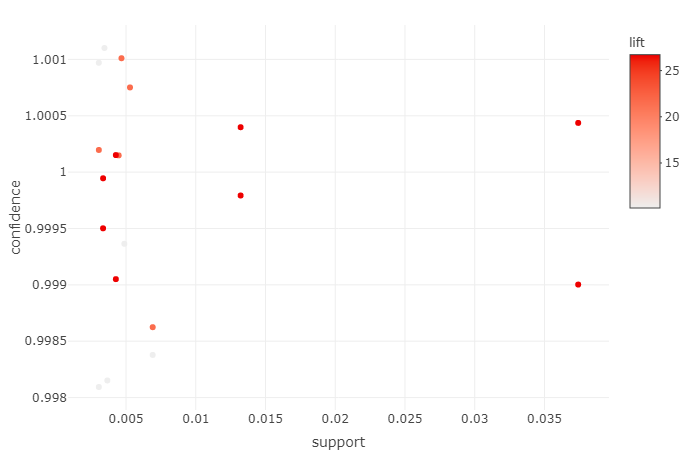
[10] {beer,shopping} => {bags} 0.006914082 1 0.006914082 10.12873 68

**# lhs(left hand side) means the product that is customer going to buy,**

**#rhs(right hand side) indicates the product to recommend to customer**

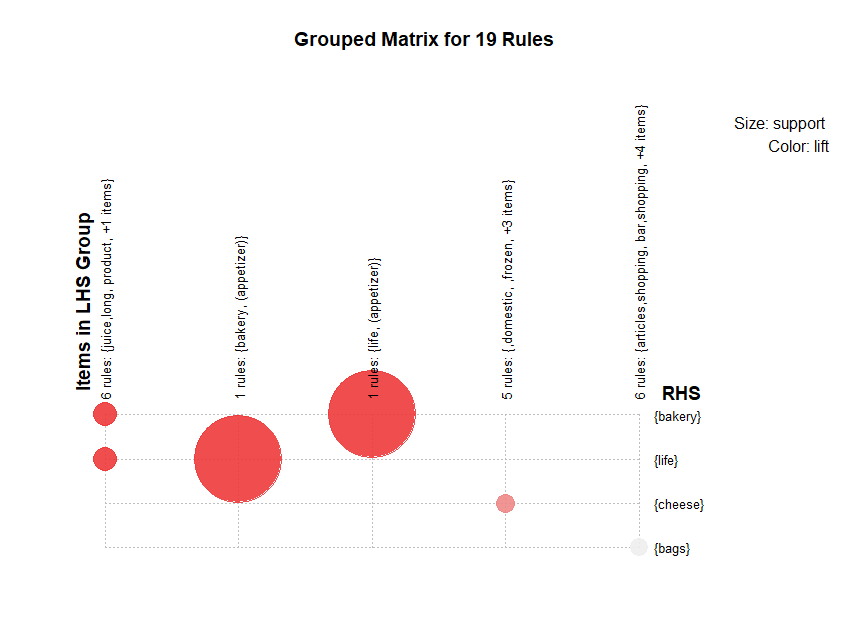
**#visualize the rules**

plotly\_arules(Grocery\_Rule2)



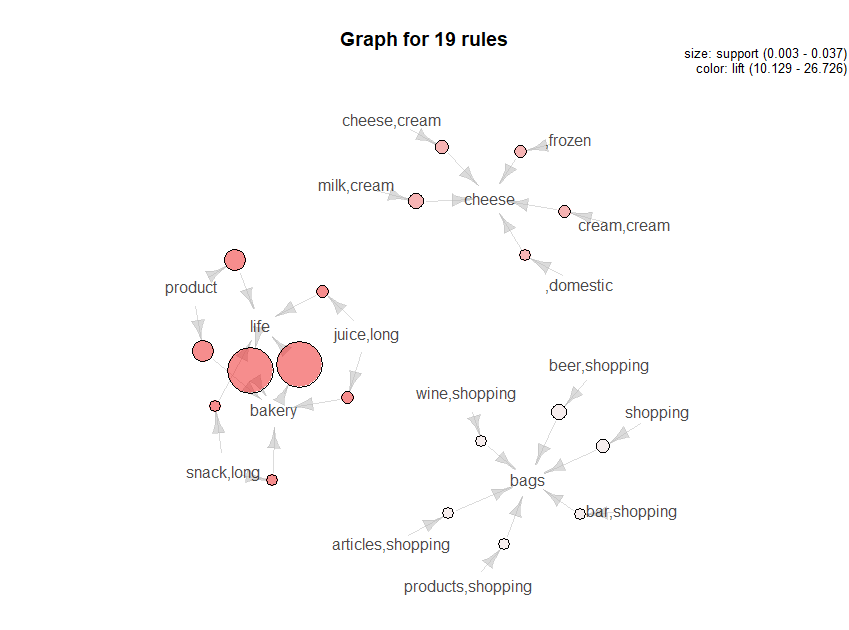
**##grouped matrix for association rules**

plot(Grocery\_Rule2,method="grouped")



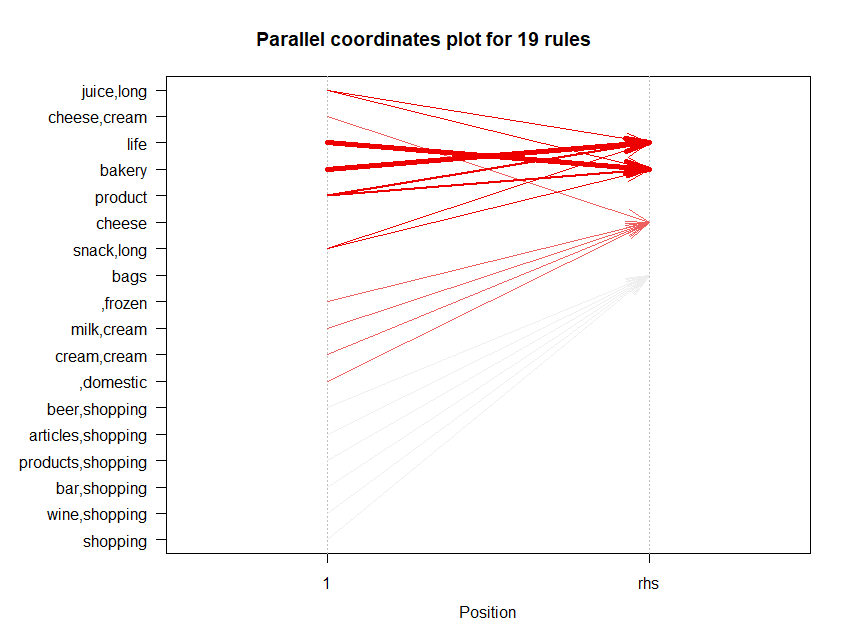
**#graph**

plot(Grocery\_Rule2 , method="graph", control=list(type="items"))



**#parallel coordinates plot**

plot(Grocery\_Rule2, method="paracoord", control=list(reorder=TRUE))



#######################################################################

**#write the top 10 rules as csv files, here I am taking Grocery\_Rule1**

write(sort(Grocery\_Rule1[1:10],by="lift"), file="Grocery\_rules.csv",sep=",")

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column1** | **rules** | **support** | **confidence** | **coverage** | **lift** | **count** |
| 2 | {pot} => {plants} | 0.001017 | 0.909090909 | 0.0011185 | 59.21132 | 10 |
| 1 | {sparkling} => {wine} | 0.001017 | 0.909090909 | 0.0011185 | 41.3931 | 10 |
| 3 | {,rolls/buns} => {cheese} | 0.001118 | 1 | 0.0011185 | 21.663 | 11 |
| 9 | {,spread} => {cheese} | 0.001118 | 1 | 0.0011185 | 21.663 | 11 |
| 4 | {vegetables,shopping} => {bags} | 0.001118 | 1 | 0.0011185 | 10.12873 | 11 |
| 5 | {beverages,shopping} => {bags} | 0.001017 | 1 | 0.0010168 | 10.12873 | 10 |
| 6 | {food,shopping} => {bags} | 0.001017 | 1 | 0.0010168 | 10.12873 | 10 |
| 7 | {film/bags,shopping} => {bags} | 0.001118 | 1 | 0.0011185 | 10.12873 | 11 |
| 8 | {care,shopping} => {bags} | 0.001118 | 1 | 0.0011185 | 10.12873 | 11 |
| 10 | {bread,shopping} => {bags} | 0.001423 | 1 | 0.0014235 | 10.12873 | 14 |