**7.4 – Developing Association Rules in R** (arules/arulesViz)

The packages arules and arulesViz contain functions for constructing and visualizing association rules given a suitable “transactional” database. A suitable database will consist entirely or nominal/ordinal variables. If we have numeric variables they will need to converted to nominal/ordinal first. For example we might convert a continuous variable such as income ($) to an ordinal variable as: Low (income < $25000), Med ($25000 < income < $50000), High (income > $50000).

Example 7.3: Golf today?  
As a preliminary example of using the association rule packages in R consider the following small “transactional” database regarding weather and whether or not we play golf that day.

> Golf

outlook temperature humidity windy play

1 sunny hot high false no

2 sunny hot high true no

3 overcast hot high false yes

4 rainy mild high false yes

5 rainy cool normal false yes

6 rainy cool normal true no

7 overcast cool normal true yes

8 sunny mild high false no

9 sunny cool normal false yes

10 rainy mild normal false yes

11 sunny mild normal true yes

12 overcast mild high true yes

13 overcast hot normal false yes

14 rainy mild high true no

In its current format this dataset could be used to form association rules using the apriori   
function from the arules library. However there is a function for converting a data frame of this type to a transactional database. This conversion will enable us to use other functions within the arules library for inspecting items.

> golf.trans = as(Golf,"transactions")

> golf.trans

transactions in sparse format with

14 transactions (rows) and

12 items (columns)

> summary(golf.trans)

transactions as itemMatrix in sparse format with

14 rows (elements/itemsets/transactions) and

12 columns (items) and a density of 0.4166667

most frequent items:

play=yes windy=false humidity=high humidity=normal temperature=mild

9 8 7 7 6

(Other)

33

element (itemset/transaction) length distribution:

sizes

5

14

Min. 1st Qu. Median Mean 3rd Qu. Max.

5 5 5 5 5 5

includes extended item information - examples:

labels variables levels

1 outlook=overcast outlook overcast

2 outlook=rainy outlook rainy

3 outlook=sunny outlook sunny

includes extended transaction information - examples:

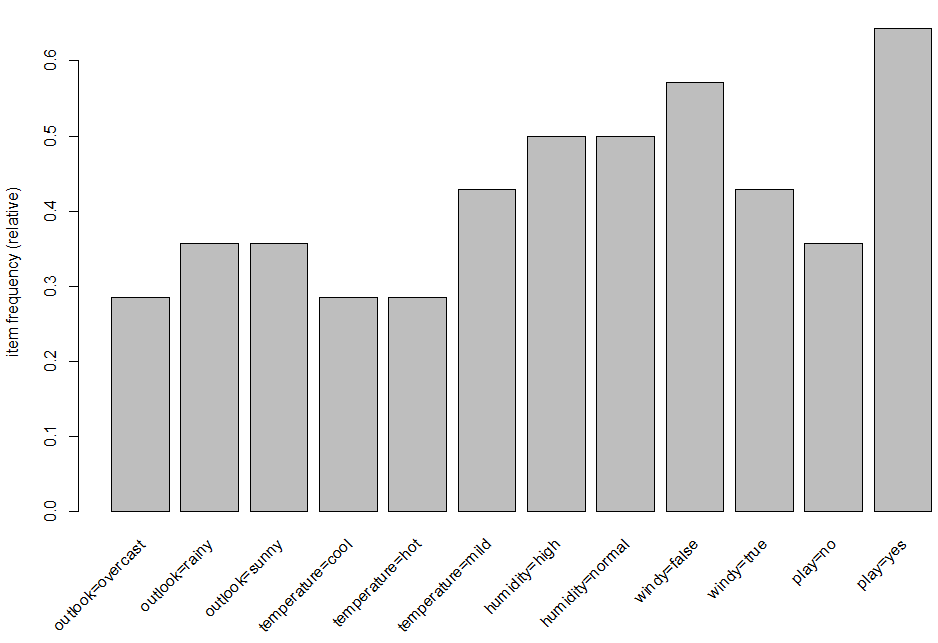
transactionID

1 1

2 2

3 3

> itemFrequencyPlot(golf.trans)



> golf.rules = apriori(golf.trans)

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.1 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Warning in apriori(golf.trans) :

You chose a very low absolute support count of 1. You might run out of memory! Increase minimum support.

apriori - find association rules with the apriori algorithm

version 4.21 (2004.05.09) (c) 1996-2004 Christian Borgelt

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

set transactions ...[12 item(s), 14 transaction(s)] done [0.00s].

sorting and recoding items ... [12 item(s)] done [0.00s].

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

checking subsets of size 1 2 3 4 done [0.00s].

writing ... [53 rule(s)] done [0.00s].

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

Even for small databases like this one, if we don’t filter out some based on support and/or confidence we will end up with a very large set of rules! The optional argument to do this has the form shown below.  
  
> arules(data, parameter = list(sup=minsup,conf=minconf,target=”rules”))

Below are some examples for the golf/weather database.

> golf.rules = apriori(golf.trans,parameter=list(supp=1/14,conf=.8,target="rules"))

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.07142857 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Warning in apriori(golf.trans, parameter = list(supp = 1/14, conf = 0.8, :

You chose a very low absolute support count of 1. You might run out of memory! Increase minimum support.

writing ... [246 rule(s)] done [0.00s].

> golf.rules = apriori(golf.trans,parameter=list(supp=3/14,conf=.9,target="rules"))

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.9 0.1 1 none FALSE TRUE 0.2142857 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

writing ... [8 rule(s)] done [0.00s].

> summary(golf.rules)

set of 8 rules

rule length distribution (lhs + rhs):sizes

2 3

2 6

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.00 2.75 3.00 2.75 3.00 3.00

summary of quality measures:

support confidence lift

Min. :0.2143 Min. :1 Min. :1.556

1st Qu.:0.2143 1st Qu.:1 1st Qu.:1.556

Median :0.2143 Median :1 Median :1.875

Mean :0.2411 Mean :1 Mean :1.902

3rd Qu.:0.2857 3rd Qu.:1 3rd Qu.:2.000

Max. :0.2857 Max. :1 Max. :2.800

mining info:

data ntransactions support confidence

golf.trans 14 0.2142857 0.9

> inspect(golf.rules)

lhs rhs support confidence lift

1 {outlook=overcast} => {play=yes} 0.2857143 1 1.555556

2 {temperature=cool} => {humidity=normal} 0.2857143 1 2.000000

3 {temperature=cool,play=yes} => {humidity=normal} 0.2142857 1 2.000000

4 {outlook=sunny,play=no} => {humidity=high} 0.2142857 1 2.000000

5 {outlook=sunny,humidity=high} => {play=no} 0.2142857 1 2.800000

6 {outlook=rainy,windy=false} => {play=yes} 0.2142857 1 1.555556

7 {outlook=rainy,play=yes} => {windy=false} 0.2142857 1 1.750000

8 {humidity=normal,windy=false} => {play=yes} 0.2857143 1 1.555556

We can use the sort() function to display the rules in order in terms of a quality measure, typically lift.

> golf.sort = sort(golf.rules,by=”lift”)

> inspect(golf.sort)

lhs rhs support confidence lift

5 {outlook=sunny,humidity=high} => {play=no} 0.2142857 1 2.800000

2 {temperature=cool} => {humidity=normal} 0.2857143 1 2.000000

3 {temperature=cool,play=yes} => {humidity=normal} 0.2142857 1 2.000000

4 {outlook=sunny,play=no} => {humidity=high} 0.2142857 1 2.000000

7 {outlook=rainy,play=yes} => {windy=false} 0.2142857 1 1.750000

1 {outlook=overcast} => {play=yes} 0.2857143 1 1.555556

6 {outlook=rainy,windy=false} => {play=yes} 0.2142857 1 1.555556

8 {humidity=normal,windy=false} => {play=yes} 0.2857143 1 1.555556

We can use the subset command to filter rules according to desired criterion. The syntax is complicated but two common/typical examples are shown below.

> golf.rules = apriori(golf.trans,parameter =list(supp=2/14,conf=.9,target="rules"))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.9 0.1 1 none FALSE TRUE 0.1428571 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 2

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

set transactions ...[12 item(s), 14 transaction(s)] done [0.00s].

sorting and recoding items ... [12 item(s)] done [0.00s].

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

checking subsets of size 1 2 3 4 done [0.00s].

writing ... [51 rule(s)] done [0.00s].

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

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

> golf.subset = subset(golf.rules,subset=lift>2.0)

> summary(golf.subset)

set of 16 rules

rule length distribution (lhs + rhs):sizes

3 4

9 7

Min. 1st Qu. Median Mean 3rd Qu. Max.

3.000 3.000 3.000 3.438 4.000 4.000

summary of quality measures:

support confidence lift

Min. :0.1429 Min. :1 Min. :2.333

1st Qu.:0.1429 1st Qu.:1 1st Qu.:2.800

Median :0.1429 Median :1 Median :2.800

Mean :0.1473 Mean :1 Mean :2.917

3rd Qu.:0.1429 3rd Qu.:1 3rd Qu.:2.975

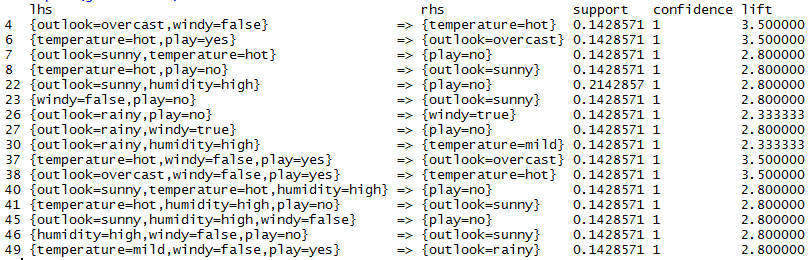
Max. :0.2143 Max. :1 Max. :3.500

mining info:

data ntransactions support confidence

golf.trans 14 0.1 0.8

> inspect(golf.subset)



> golf.subset2 = subset(golf.rules,subset=rhs %in% "play=yes")

> summary(golf.subset2)

set of 16 rules

rule length distribution (lhs + rhs):sizes

2 3 4

2 10 4

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.000 3.000 3.000 3.125 3.250 4.000

summary of quality measures:

support confidence lift

Min. :0.1429 Min. :0.8571 Min. :1.333

1st Qu.:0.1429 1st Qu.:1.0000 1st Qu.:1.556

Median :0.1429 Median :1.0000 Median :1.556

Mean :0.1830 Mean :0.9911 Mean :1.542

3rd Qu.:0.1607 3rd Qu.:1.0000 3rd Qu.:1.556

Max. :0.4286 Max. :1.0000 Max. :1.556

mining info:

data ntransactions support confidence

golf.trans 14 0.1 0.8

> inspect(golf.subset2)

lhs rhs support confidence lift

> inspect(golf.subset2)

lhs rhs support confidence lift

1 {outlook=overcast} => {play=yes} 0.2857143 1 1.555556

5 {outlook=overcast,temperature=hot} => {play=yes} 0.1428571 1 1.555556

12 {outlook=overcast,windy=true} => {play=yes} 0.1428571 1 1.555556

13 {outlook=overcast,humidity=high} => {play=yes} 0.1428571 1 1.555556

14 {outlook=overcast,humidity=normal} => {play=yes} 0.1428571 1 1.555556

15 {outlook=overcast,windy=false} => {play=yes} 0.1428571 1 1.555556

20 {temperature=cool,windy=false} => {play=yes} 0.1428571 1 1.555556

24 {outlook=sunny,humidity=normal} => {play=yes} 0.1428571 1 1.555556

31 {outlook=rainy,windy=false} => {play=yes} 0.2142857 1 1.555556

33 {temperature=mild,humidity=normal} => {play=yes} 0.1428571 1 1.555556

34 {humidity=normal,windy=false} => {play=yes} 0.2857143 1 1.555556

35 {outlook=overcast,temperature=hot,windy=false} => {play=yes} 0.1428571 1 1.555556

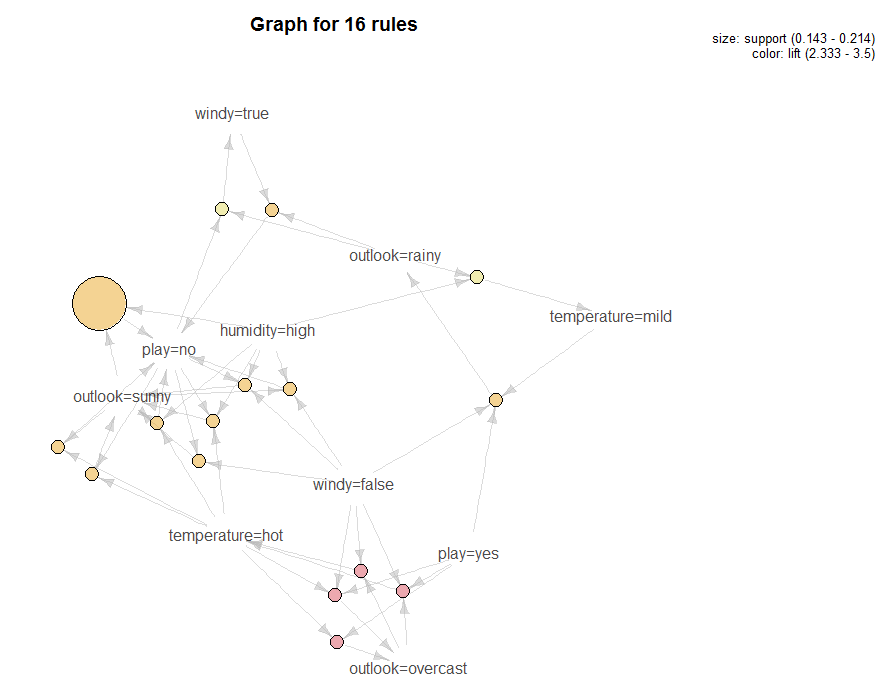
42 {temperature=cool,humidity=normal,windy=false} => {play=yes} 0.1428571 1 1.555556

47 {outlook=rainy,temperature=mild,windy=false} => {play=yes} 0.1428571 1 1.555556

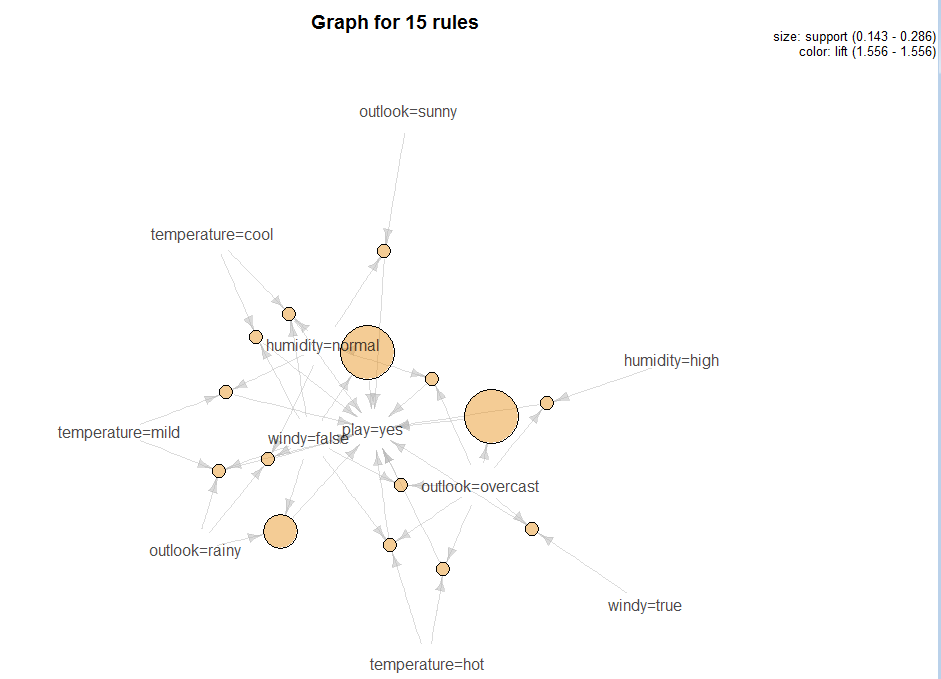
50 {outlook=rainy,humidity=normal,windy=false} => {play=yes} 0.1428571 1 1.555556

If you have a fairly small subset of rules you wish to display graphically, the plot function in the aruleViz library has some useful options for doing this.

> plot(golf.subset,method=”graph”)



> plot(golf.subset2,method=”graph”)



Example 7.4: Association Rules for the Mushroom Data

In this example we look for association rules within the mushroom data set we have looked at previously. We can treat the characteristics of the mushrooms as transactional data and look for characteristics that most associated. For example a rule might be:  
 {cap-surface=f,cap-color=c} => {veil-type=u}.

The mushroom attributes are defined below:

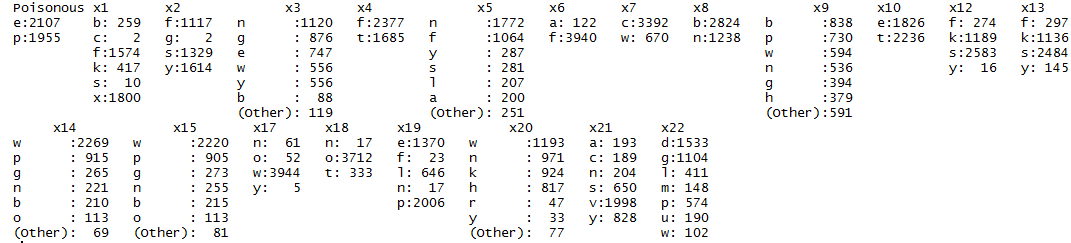
Variable Levels \*  
     Poisonous edible=e,poisonous=p

X1 = cap-shape: bell=b,conical=c,convex=x,flat=f,  
 knobbed=k,sunken=s  
 X2 = cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s  
 X3 = cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,  
 pink=p,purple=u,red=e,white=w,yellow=y  
 X4 = bruises?: true=t,false=f  
 X5 = odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,  
 musty=m,none=n,pungent=p,spicy=s  
 X6 = gill-attachment: attached=a, free=f  
 X7 = gill-spacing: close=c,crowded=w  
 X8 = gill-size: broad=b,narrow=n  
 X9 = gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,  
 green=r,orange=o,pink=p,purple=u,red=e,  
 white=w,yellow=y  
 X10 = stalk-shape: enlarging=e,tapering=t  
 X11 = stalk-root: bulbous=b,club=c,cup=u,equal=e,  
 rhizomorphs=z,rooted=r,missing=? 🡨 has lots of missing values!  
 X12 = stalk-surface-above-ring: ibrous=f,scaly=y,silky=k,smooth=s  
 X13 = stalk-surface-below-ring: ibrous=f,scaly=y,silky=k,smooth=s 🡨 has lots of missing values!  
 X14 = stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X15 = stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X16 = veil-type: partial=p,universal=u 🡨 All partial (no universal)!  
 X17 = veil-color: brown=n,orange=o,white=w,yellow=y  
 X18 = ring-number: none=n,one=o,two=t  
 X19 = ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,  
 none=n,pendant=p,sheathing=s,zone=z 🡨 some levels not represented  
 X20 = spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,  
 orange=o,purple=u,white=w,yellow=y  
 X21 = population: abundant=a,clustered=c,numerous=n,  
 scattered=s,several=v,solitary=y  
 X22 = habitat: grasses=g,leaves=l,meadows=m,paths=p,  
 urban=u,waste=w,woods=d

Note: We have used these data previously Chapters 2 and 3.

> mushrooms = Mushrooms.train[,-c(12,17)]

> summary(mushsubset)



> mush.trans = as(mushsubset,"transactions")

> mush.trans

transactions in sparse format with

4062 transactions (rows) and

113 items (columns)

> summary(mush.trans)

transactions as itemMatrix in sparse format with

4062 rows (elements/itemsets/transactions) and

113 columns (items) and a density of 0.1858407

most frequent items:

x17=w x6=f x18=o x7=c x8=b (Other)

3944 3940 3712 3392 2824 67490

element (itemset/transaction) length distribution:

sizes

21

4062

Min. 1st Qu. Median Mean 3rd Qu. Max.

21 21 21 21 21 21

includes extended item information - examples:

labels variables levels

1 Poisonous=e Poisonous e

2 Poisonous=p Poisonous p

3 x1=b x1 b

includes extended transaction information - examples:

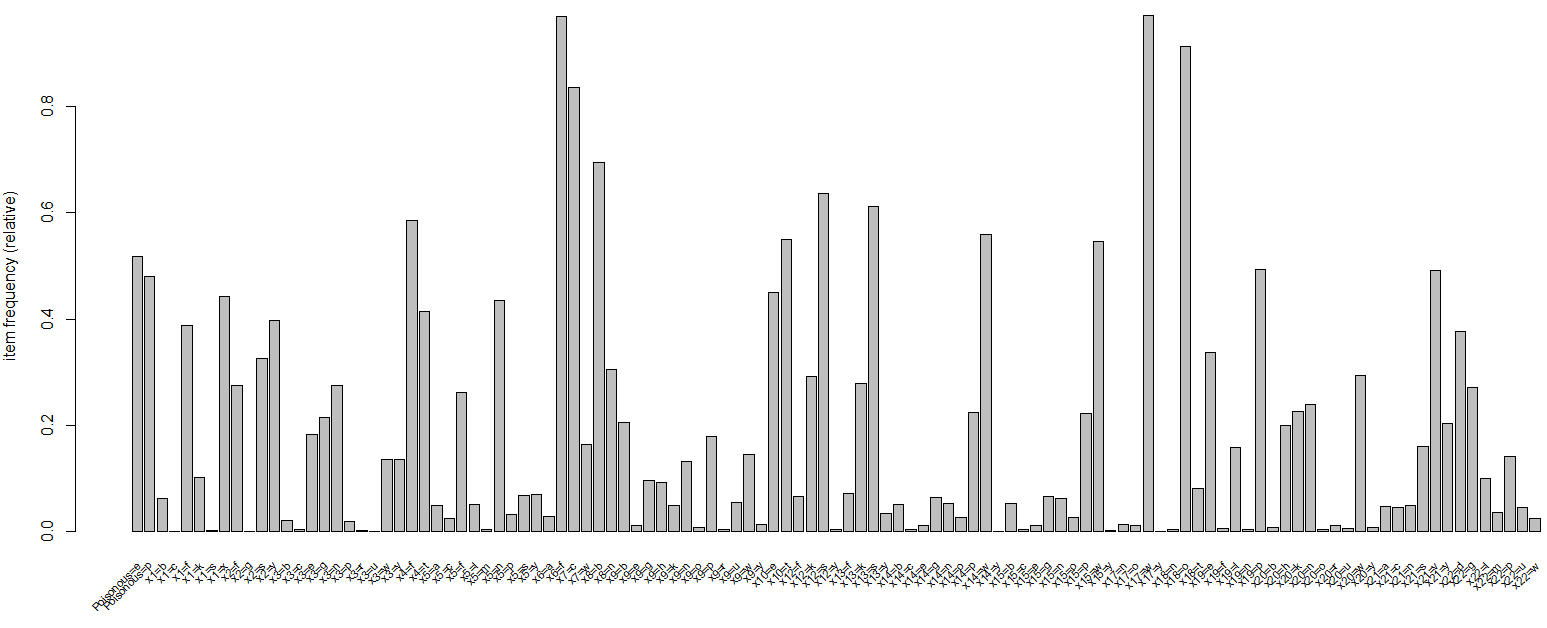
transactionID

1 7251

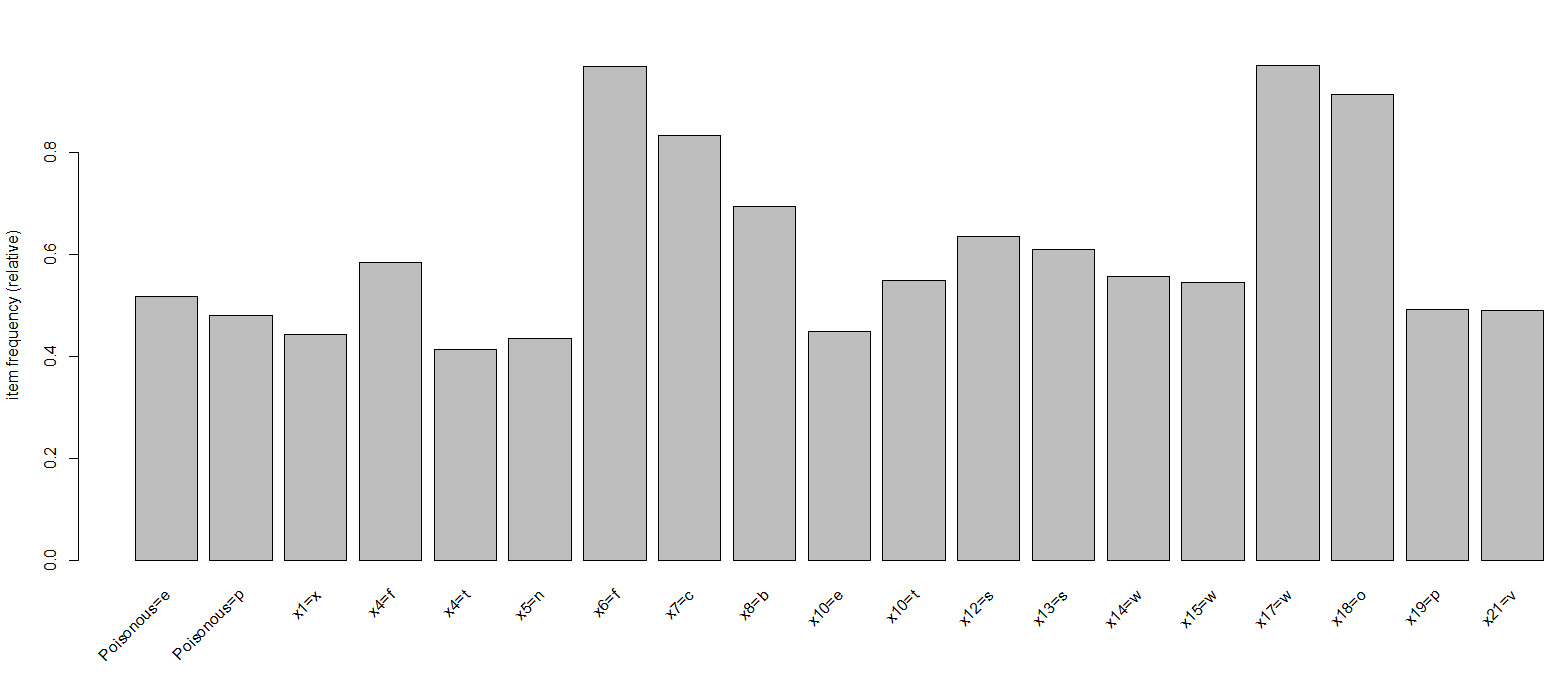
2 724

3 2997

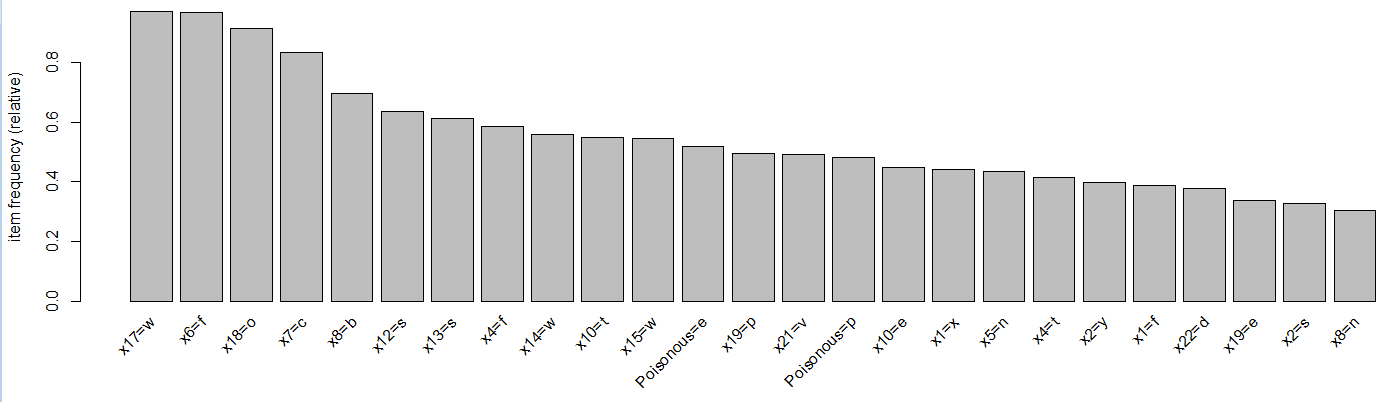
> itemFrequencyPlot(mush.trans,cex=.6)



> itemFrequencyPlot(mush.trans,support=.4)



> itemFrequencyPlot(mush.trans,topN=25)



> mush.rules = apriori(mush.trans)

> mush.rules = apriori(mush.trans)

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.1 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 406

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

set transactions ...[113 item(s), 4062 transaction(s)] done [0.00s].

sorting and recoding items ... [52 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 [0.64s].

writing ... [686273 rule(s)] done [0.14s].

creating S4 object ... done [0.29s].

This is way too many rules to plot or interact with! We have a few options to reduce or filter this rule set:

1. filter out some of these rules by choosing only those with a high support, confidence/lift
2. sort and inspect only those rules with the highest “quality”
3. put more stringent requirements when calling the apriori function
4. inspect only those with a specific RHS/consequent

> summary(mush.rules)

set of 686273 rules

rule length distribution (lhs + rhs):sizes

1 2 3 4 5 6 7 8 9 10

4 272 3805 20547 60973 116429 156321 154157 112927 60838

Min. 1st Qu. Median Mean 3rd Qu. Max.

1.000 6.000 7.000 7.358 9.000 10.000

summary of quality measures:

support confidence lift

Min. :0.1002 Min. :0.8000 Min. :0.8243

1st Qu.:0.1022 1st Qu.:1.0000 1st Qu.:1.0943

Median :0.1041 Median :1.0000 Median :1.7089

Mean :0.1241 Mean :0.9859 Mean :1.8902

3rd Qu.:0.1297 3rd Qu.:1.0000 3rd Qu.:2.0849

Max. :0.9710 Max. :1.0000 Max. :6.2879

mining info:

data ntransactions support confidence

mush.trans 4062 0.1 0.8

We will first use option (1). We can save the quality measures to an object and then use the quality object to filter our rules.

> mush.quality = quality(mush.rules)

Here we filter out rules with and , or grab only those with lift > 2.50 and support > 0.20.

> mush.subset = mush.rules[mush.quality$lift>2.5 & mush.quality$supp>.2]

> summary(mush.subset)

set of 9142 rules

rule length distribution (lhs + rhs):sizes

2 3 4 5 6 7 8 9 10

3 74 413 1140 1938 2242 1839 1068 425

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.000 6.000 7.000 6.943 8.000 10.000

summary of quality measures:

support confidence lift

Min. :0.2026 Min. :0.8096 Min. :2.509

1st Qu.:0.2063 1st Qu.:1.0000 1st Qu.:2.965

Median :0.2063 Median :1.0000 Median :3.281

Mean :0.2065 Mean :0.9902 Mean :3.386

3rd Qu.:0.2063 3rd Qu.:1.0000 3rd Qu.:3.405

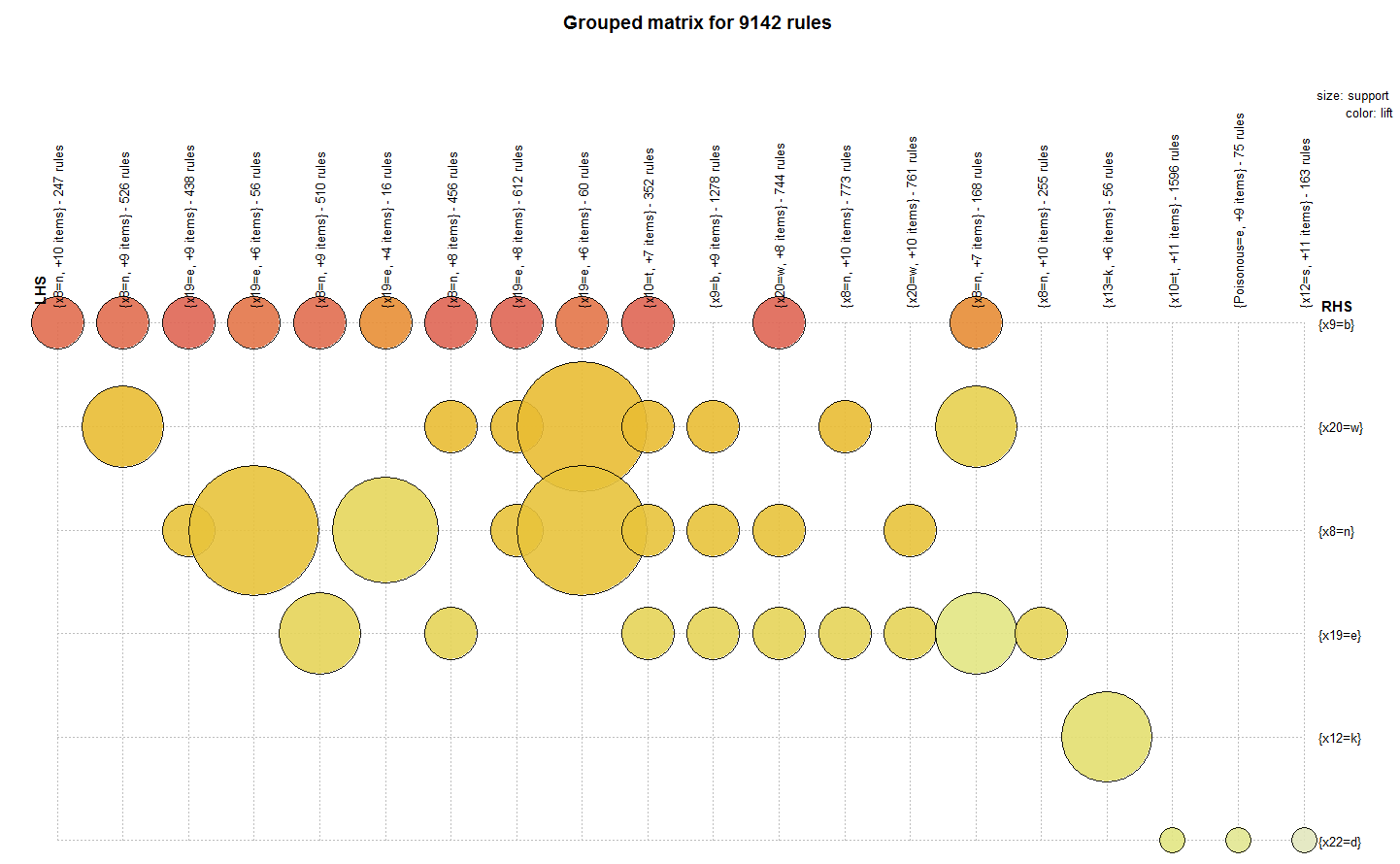
Max. :0.3003 Max. :1.0000 Max. :4.847

mining info:

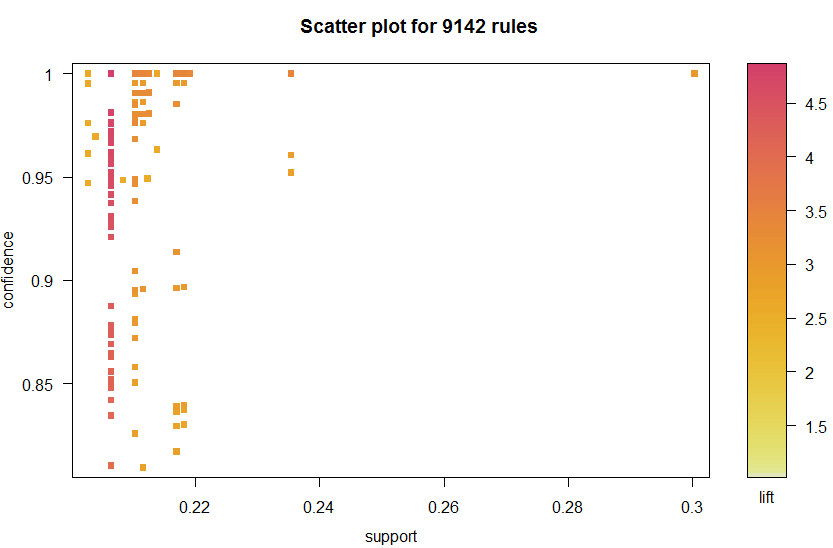
data ntransactions support confidence

mush.trans 4062 0.1 0.8

> plot(mush.subset,method=”grouped”)



> plot(mush.subset)

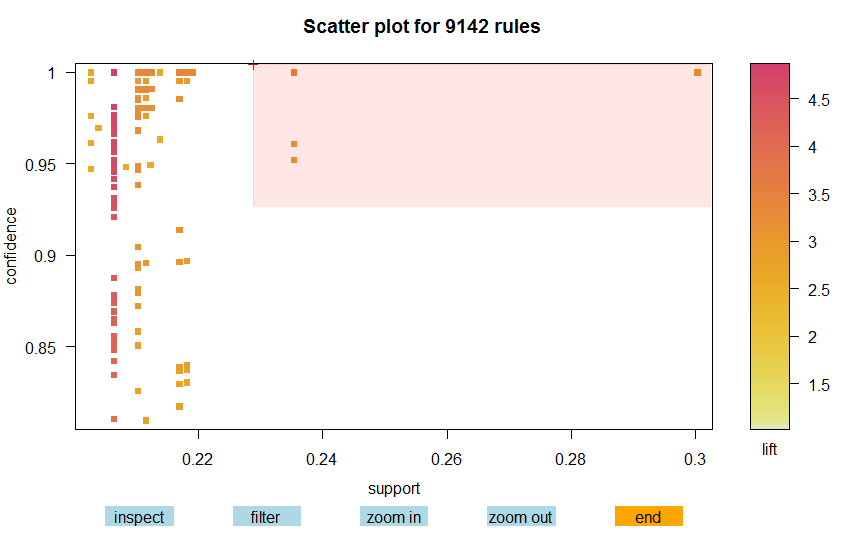


We can inspect and further filter these rules interactively by using the interactive=T option to the plot function.

> sel = plot(mush.subset,interactive=T)

Interactive mode.

Select a region with two clicks!



Number of rules selected: 16

lhs rhs support confidence lift order

2583 {x7=c,x19=e} => {x20=w} 0.2353520 1.0000000 3.404862 3

16178 {x6=f,x7=c,x19=e} => {x20=w} 0.2353520 1.0000000 3.404862 4

16182 {x7=c,x17=w,x19=e} => {x20=w} 0.2353520 1.0000000 3.404862 4

60402 {x6=f,x7=c,x17=w,x19=e} => {x20=w} 0.2353520 1.0000000 3.404862 5

2919 {x4=f,x10=t} => {x19=e} 0.3003447 1.0000000 2.964964 3

18123 {x4=f,x10=t,x18=o} => {x19=e} 0.3003447 1.0000000 2.964964 4

18127 {x4=f,x6=f,x10=t} => {x19=e} 0.3003447 1.0000000 2.964964 4

18131 {x4=f,x10=t,x17=w} => {x19=e} 0.3003447 1.0000000 2.964964 4

65654 {x4=f,x6=f,x10=t,x18=o} => {x19=e} 0.3003447 1.0000000 2.964964 5

65659 {x4=f,x10=t,x17=w,x18=o} => {x19=e} 0.3003447 1.0000000 2.964964 5

65664 {x4=f,x6=f,x10=t,x17=w} => {x19=e} 0.3003447 1.0000000 2.964964 5

163588 {x4=f,x6=f,x10=t,x17=w,x18=o} => {x19=e} 0.3003447 1.0000000 2.964964 6

16177 {x6=f,x7=c,x20=w} => {x19=e} 0.2353520 0.9608040 2.848749 4

60401 {x6=f,x7=c,x17=w,x20=w} => {x19=e} 0.2353520 0.9608040 2.848749 5

2582 {x7=c,x20=w} => {x19=e} 0.2353520 0.9521912 2.823212 3

16181 {x7=c,x17=w,x20=w} => {x19=e} 0.2353520 0.9521912 2.823212 4

Using option (2) we can sort the original large rule set by some criterion, e.g. by lift.

> inspect(sort(mush.rules,by="lift")[1:10]) 🡨 show the top 10 rules by lift

lhs rhs support confidence lift

490 {x13=k,x20=h} => {x19=l} 0.159035 1 6.287926

493 {x12=k,x20=h} => {x19=l} 0.159035 1 6.287926

522 {x5=f,x10=e} => {x19=l} 0.159035 1 6.287926

4480 {x5=f,x13=k,x20=h} => {x19=l} 0.159035 1 6.287926

4484 {x5=f,x12=k,x20=h} => {x19=l} 0.159035 1 6.287926

4488 {x5=f,x10=e,x20=h} => {x19=l} 0.159035 1 6.287926

4496 {x4=f,x5=f,x20=h} => {x19=l} 0.159035 1 6.287926

4520 {x12=k,x13=k,x20=h} => {x19=l} 0.159035 1 6.287926

4524 {x10=e,x13=k,x20=h} => {x19=l} 0.159035 1 6.287926

4528 {Poisonous=p,x13=k,x20=h} => {x19=l} 0.159035 1 6.287926

We can set rule filters based on quality measures directly into the apriori function call as shown below (Option 3).

> highconfrules = apriori(mush.trans,parameter=list(support=0.2,conf=.98))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.98 0.1 1 none FALSE TRUE 0.2 1 10 rules FALSE

Algorithmic control:

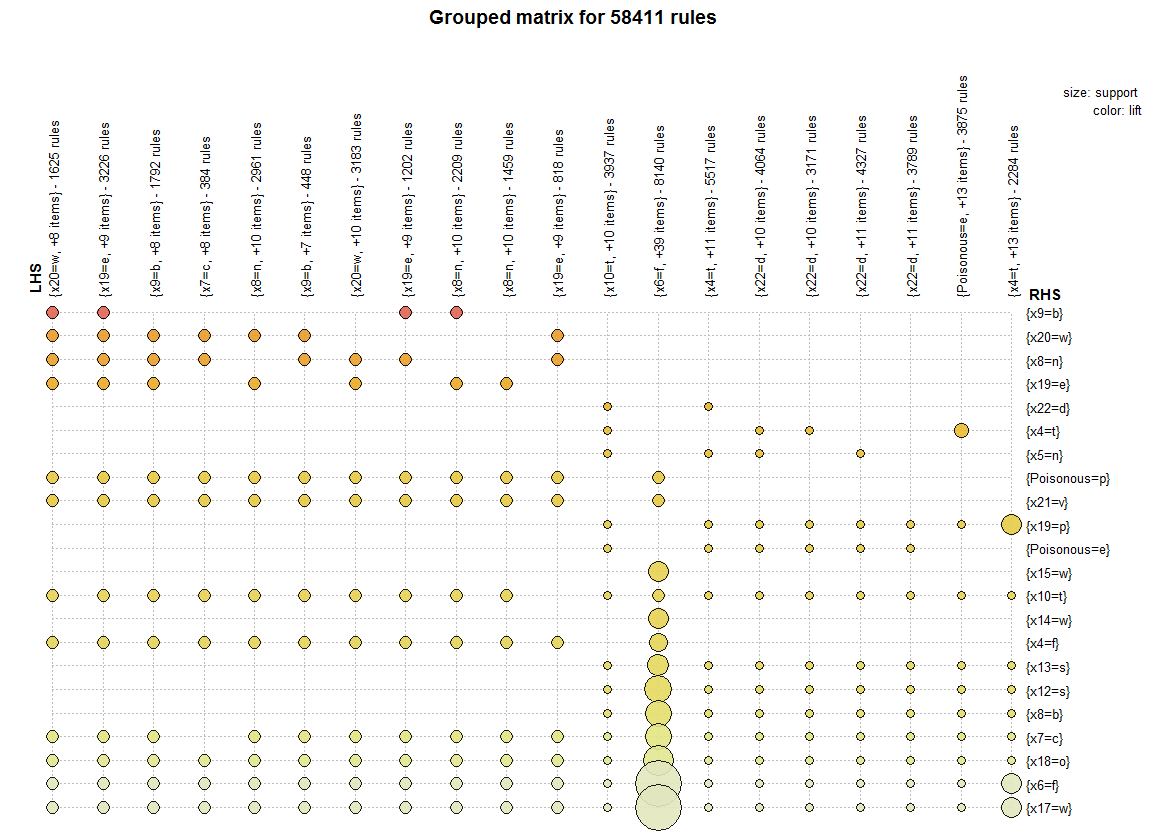
filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

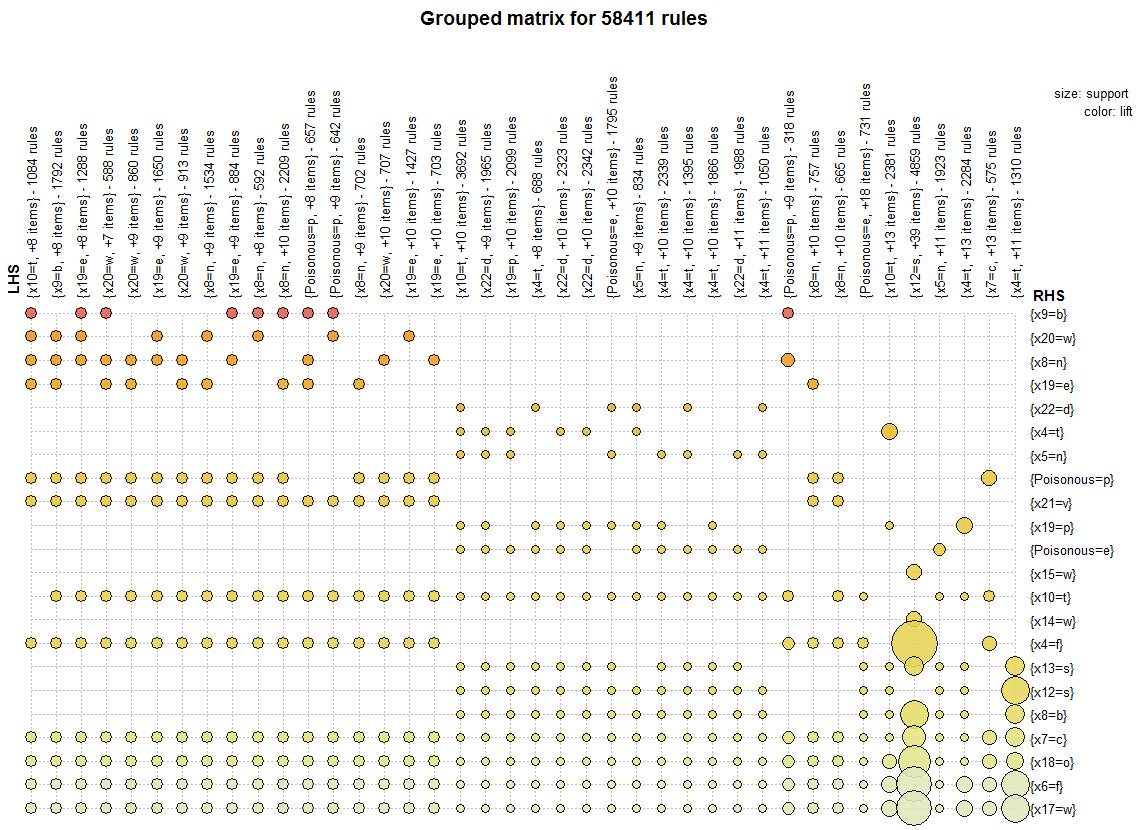
Absolute minimum support count: 812

writing ... [58411 rule(s)] done [0.01s].

> plot(highconfrules,method="grouped")



The rules in the upper left corner of this plot for most part have the highest lift. We can change the number of groups (columns) which has *k = 20* by default.

> plot(highconfrules,method="grouped",control=list(k=40))  
  
  
> confonerules = apriori(mush.trans,parameter=list(support=.4,conf=.98))

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.98 0.1 1 none FALSE TRUE 0.25 1 10 rules FALSE

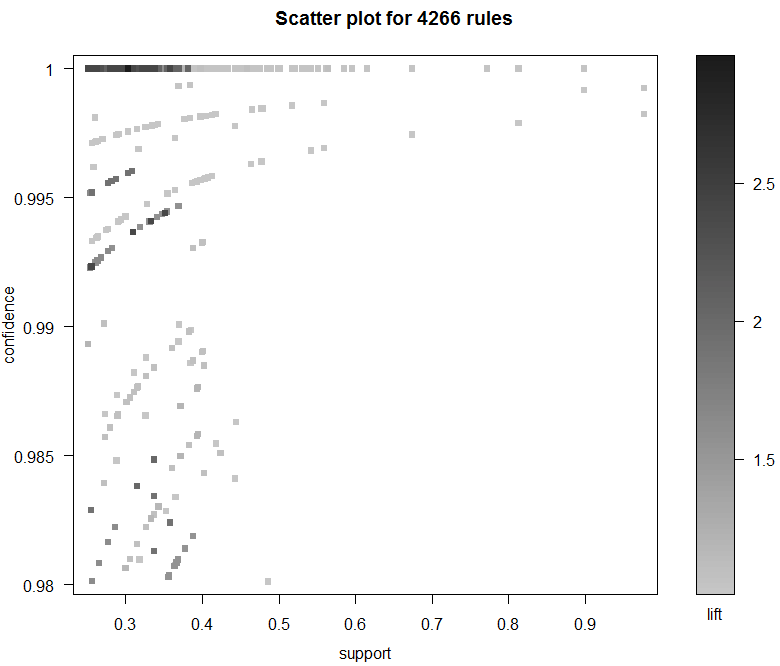
Algorithmic control:

filter tree heap memopt load sort verbose

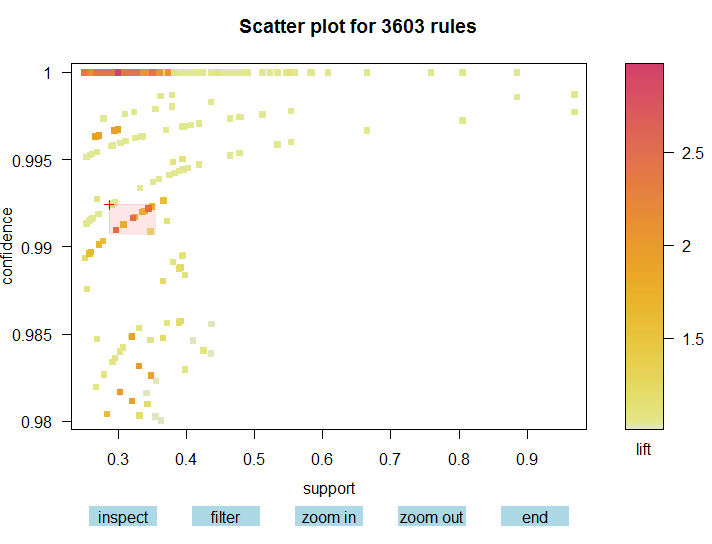
0.1 TRUE TRUE FALSE TRUE 2 TRUE

writing ... [4266 rule(s)] done [0.00s].

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

> plot(confonerules)  


We can again interactively select rules from this plot.



Number of rules selected: 26

lhs rhs support confidence lift order

1418 {x6=f,x7=c,x8=b,x19=p} => {x4=t} 0.3446578 0.9922041 2.391889 5

1421 {x7=c,x8=b,x17=w,x19=p} => {x4=t} 0.3446578 0.9922041 2.391889 5

2324 {x6=f,x7=c,x8=b,x17=w,x19=p} => {x4=t} 0.3446578 0.9922041 2.391889 6

1481 {Poisonous=e,x6=f,x7=c,x8=b} => {x4=t} 0.3217627 0.9916540 2.390563 5

1485 {Poisonous=e,x7=c,x8=b,x17=w} => {x4=t} 0.3217627 0.9916540 2.390563 5

2414 {Poisonous=e,x6=f,x7=c,x8=b,x17=w} => {x4=t} 0.3217627 0.9916540 2.390563 6

2218 {Poisonous=e,x6=f,x7=c,x8=b,x19=p} => {x4=t} 0.2966519 0.9909539 2.388875 6

2222 {Poisonous=e,x7=c,x8=b,x17=w,x19=p} => {x4=t} 0.2966519 0.9909539 2.388875 6

2961 {Poisonous=e,x6=f,x7=c,x8=b,x17=w,x19=p} => {x4=t} 0.2966519 0.9909539 2.388875 7

940 {Poisonous=e,x7=c,x8=b} => {x12=s} 0.3495815 0.9923131 1.560502 4

942 {Poisonous=e,x6=f,x7=c} => {x12=s} 0.3387494 0.9920692 1.560118 4

943 {Poisonous=e,x7=c,x17=w} => {x12=s} 0.3387494 0.9920692 1.560118 4

1912 {Poisonous=e,x6=f,x7=c,x17=w} => {x12=s} 0.3387494 0.9920692 1.560118 5

858 {Poisonous=e,x7=c,x19=p} => {x12=s} 0.3357952 0.9920000 1.560009 4

1762 {Poisonous=e,x7=c,x8=b,x19=p} => {x12=s} 0.3244707 0.9917231 1.559574 5

1904 {Poisonous=e,x6=f,x7=c,x8=b} => {x12=s} 0.3217627 0.9916540 1.559465 5

1905 {Poisonous=e,x7=c,x8=b,x17=w} => {x12=s} 0.3217627 0.9916540 1.559465 5

2764 {Poisonous=e,x6=f,x7=c,x8=b,x17=w} => {x12=s} 0.3217627 0.9916540 1.559465 6

1767 {Poisonous=e,x6=f,x7=c,x19=p} => {x12=s} 0.3079764 0.9912837 1.558883 5

1768 {Poisonous=e,x7=c,x17=w,x19=p} => {x12=s} 0.3079764 0.9912837 1.558883 5

2657 {Poisonous=e,x6=f,x7=c,x17=w,x19=p} => {x12=s} 0.3079764 0.9912837 1.558883 6

2643 {Poisonous=e,x6=f,x7=c,x8=b,x19=p} => {x12=s} 0.2966519 0.9909539 1.558364 6

2644 {Poisonous=e,x7=c,x8=b,x17=w,x19=p} => {x12=s} 0.2966519 0.9909539 1.558364 6

3251 {Poisonous=e,x6=f,x7=c,x8=b,x17=w,x19=p} => {x12=s} 0.2966519 0.9909539 1.558364 7

580 {x6=f,x7=c,x22=d} => {x18=o} 0.3476120 0.9908772 1.084306 4

1323 {x6=f,x7=c,x17=w,x22=d} => {x18=o} 0.3476120 0.9908772 1.084306 5

We can also specify restrictions for the LHS (antecedent) and RHS (consequent) in the association rule search. For example, suppose we wish to find the rules that imply the mushroom is poisonous. We would then use the following apriori function call.

> Poison.rules = apriori(mush.trans,appearance=list(default="lhs",rhs="Poisonous=p"))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.1 1 10 rules FALSE

writing ... [27744 rule(s)] done [0.01s].

> summary(Poison.rules)

set of 27744 rules

rule length distribution (lhs + rhs):sizes

2 3 4 5 6 7 8 9 10

8 145 840 2582 4960 6566 6251 4301 2091

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.000 6.000 7.000 7.283 8.000 10.000

summary of quality measures:

support confidence lift

Min. :0.1002 Min. :0.8017 Min. :1.666

1st Qu.:0.1022 1st Qu.:1.0000 1st Qu.:2.078

Median :0.1041 Median :1.0000 Median :2.078

Mean :0.1206 Mean :0.9947 Mean :2.067

3rd Qu.:0.1118 3rd Qu.:1.0000 3rd Qu.:2.078

Max. :0.3848 Max. :1.0000 Max. :2.078

> Poison.rules = sort(Poison.rules,by="lift")

> inspect(Poison.rules[1:20])

lhs rhs support confidence lift

2 {x19=l} => {Poisonous=p} 0.1590350 1 2.077749

4 {x9=b} => {Poisonous=p} 0.2063023 1 2.077749

5 {x5=f} => {Poisonous=p} 0.2619399 1 2.077749

14 {x19=l,x20=h} => {Poisonous=p} 0.1590350 1 2.077749

15 {x5=f,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

16 {x13=k,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

17 {x12=k,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

18 {x10=e,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

19 {x4=f,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

20 {x8=b,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

21 {x7=c,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

22 {x18=o,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

23 {x6=f,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

24 {x17=w,x19=l} => {Poisonous=p} 0.1590350 1 2.077749

25 {x3=e,x9=b} => {Poisonous=p} 0.1058592 1 2.077749

27 {x3=e,x8=n} => {Poisonous=p} 0.1058592 1 2.077749

29 {x3=e,x4=f} => {Poisonous=p} 0.1070901 1 2.077749

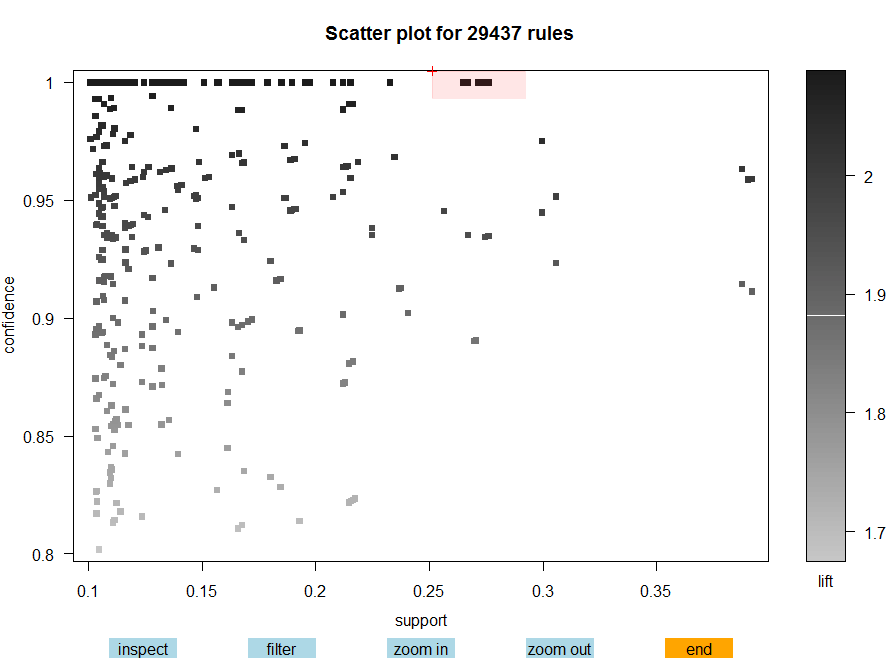
30 {x5=f,x20=h} => {Poisonous=p} 0.1954702 1 2.077749

31 {x13=k,x20=h} => {Poisonous=p} 0.1590350 1 2.077749

32 {x12=k,x20=h} => {Poisonous=p} 0.1590350 1 2.077749

Again we use plots to interactive select a subset of higher quality rules.

> sel2 = plot(Poison.rules,interactive=T)



Sel2 – max lift with support of 25%

Sel 3 – lift = 2.0 and max support.

Number of rules selected: 24

lhs rhs support confidence lift order

117 {x7=c,x12=k} => {Poisonous=p} 0.2735106 1 2.077749 3

118 {x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 3

758 {x4=f,x7=c,x12=k} => {Poisonous=p} 0.2735106 1 2.077749 4

759 {x4=f,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 4

766 {x7=c,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 4

767 {x6=f,x7=c,x12=k} => {Poisonous=p} 0.2712949 1 2.077749 4

768 {x7=c,x12=k,x17=w} => {Poisonous=p} 0.2735106 1 2.077749 4

769 {x6=f,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 4

770 {x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 4

2854 {x4=f,x7=c,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

2855 {x4=f,x6=f,x7=c,x12=k} => {Poisonous=p} 0.2712949 1 2.077749 5

2856 {x4=f,x7=c,x12=k,x17=w} => {Poisonous=p} 0.2735106 1 2.077749 5

2857 {x4=f,x6=f,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

2858 {x4=f,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

2866 {x6=f,x7=c,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

2867 {x7=c,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

2868 {x6=f,x7=c,x12=k,x17=w} => {Poisonous=p} 0.2712949 1 2.077749 5

2869 {x6=f,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 5

7262 {x4=f,x6=f,x7=c,x12=k,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 6

7263 {x4=f,x7=c,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 6

7264 {x4=f,x6=f,x7=c,x12=k,x17=w} => {Poisonous=p} 0.2712949 1 2.077749 6

7265 {x4=f,x6=f,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 6

7270 {x6=f,x7=c,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 6

13631 {x4=f,x6=f,x7=c,x12=k,x17=w,x18=o} => {Poisonous=p} 0.2693255 1 2.077749 7

The second selection highlighted “sel3”.

> inspect(sel3)

lhs rhs support confidence lift order

3573 {x4=f,x6=f,x7=c,x18=o} => {Poisonous=p} 0.3806007 0.9572755 1.988979 5

3574 {x4=f,x7=c,x17=w,x18=o} => {Poisonous=p} 0.3806007 0.9572755 1.988979 5

8535 {x4=f,x6=f,x7=c,x17=w,x18=o} => {Poisonous=p} 0.3806007 0.9572755 1.988979 6

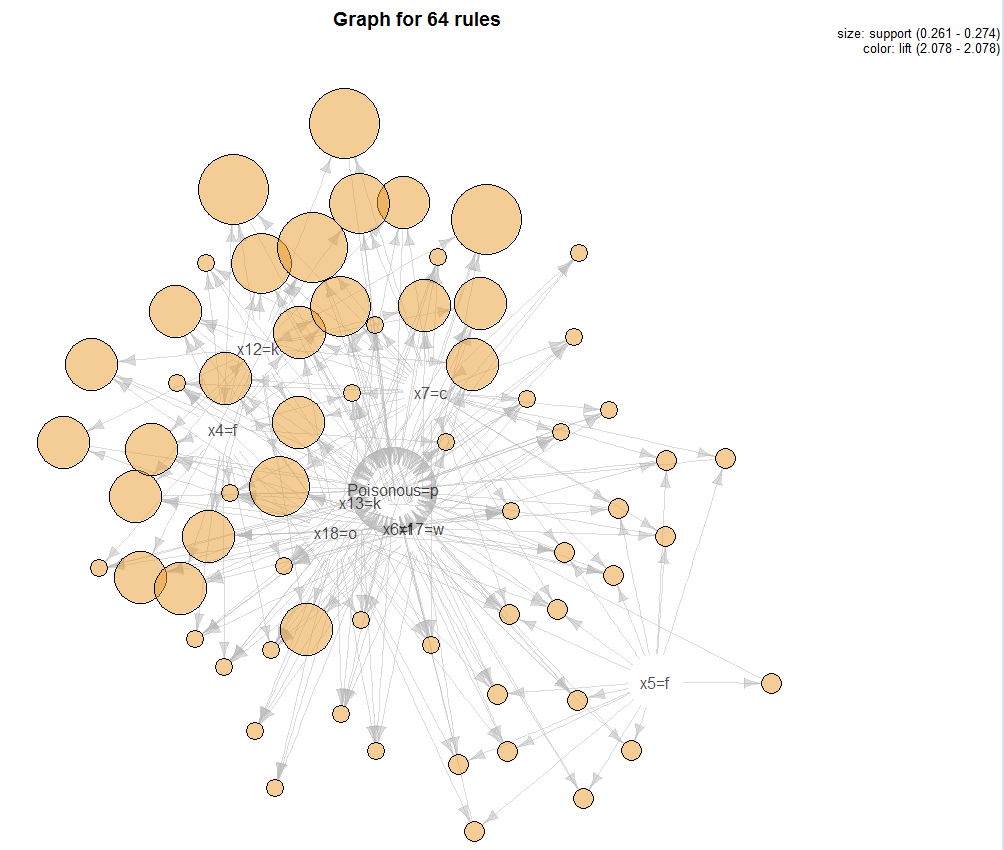
993 {x4=f,x7=c,x17=w} => {Poisonous=p} 0.3847858 0.9513086 1.976581 4

992 {x4=f,x6=f,x7=c} => {Poisonous=p} 0.3825702 0.9510404 1.976024 4

3575 {x4=f,x6=f,x7=c,x17=w} => {Poisonous=p} 0.3825702 0.9510404 1.976024 5

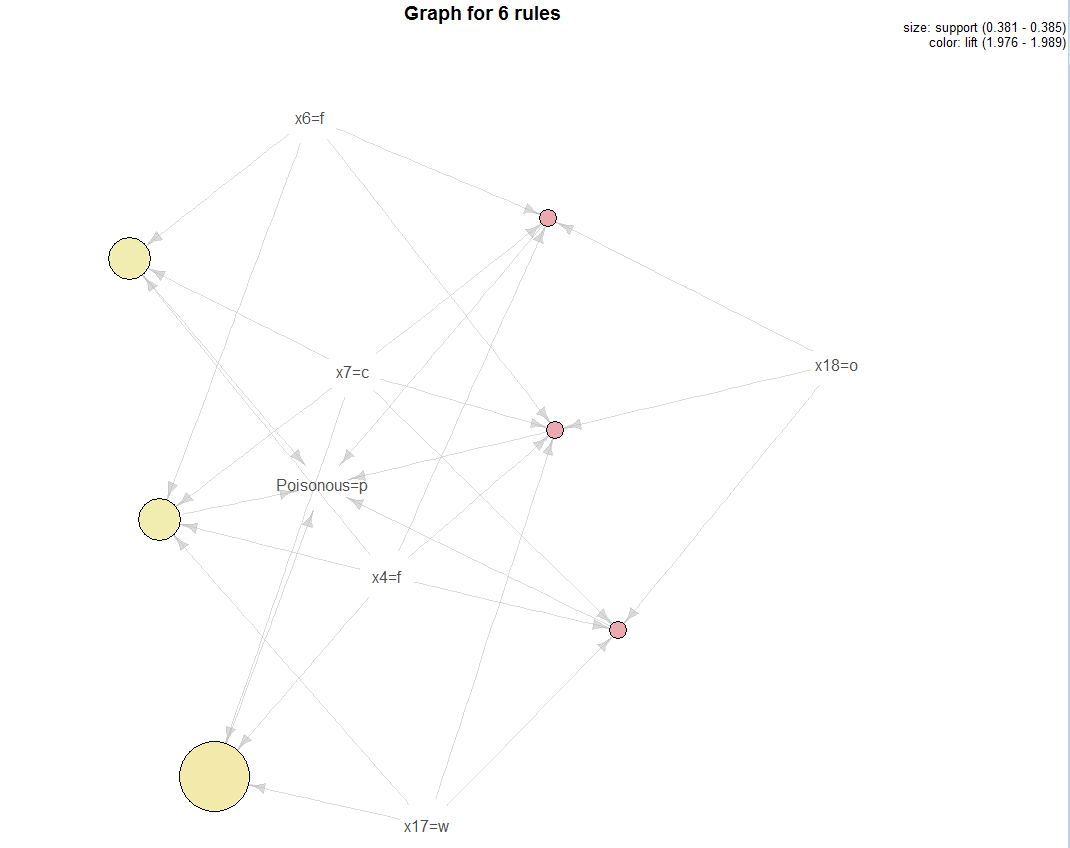
Plot of the rules in the sel2 region.

> plot(sel2,method=”graph”)



Plot of the rules in the sel3 region.

> plot(sel3,method=”graph”)



We can specify that we want to see shorter length rules using the maxlen option. There is also a minlen option if you want to see longer rules.

> Poison.rules = apriori(mush.trans,parameter=list(supp=.25,conf=.9,maxlen=3),

appearance=list(default="lhs",rhs="Poisonous=p"))

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.9 0.1 1 none FALSE TRUE 0.25 1 3 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 1015

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

writing ... [19 rule(s)] done [0.00s].

> summary(Poison.rules)

set of 19 rules

rule length distribution (lhs + rhs):sizes

2 3

3 16

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.000 3.000 3.000 2.842 3.000 3.000

summary of quality measures:

support confidence lift

Min. :0.2546 Min. :0.9156 Min. :1.902

1st Qu.:0.2612 1st Qu.:0.9340 1st Qu.:1.941

Median :0.2619 Median :0.9374 Median :1.948

Mean :0.2669 Mean :0.9645 Mean :2.004

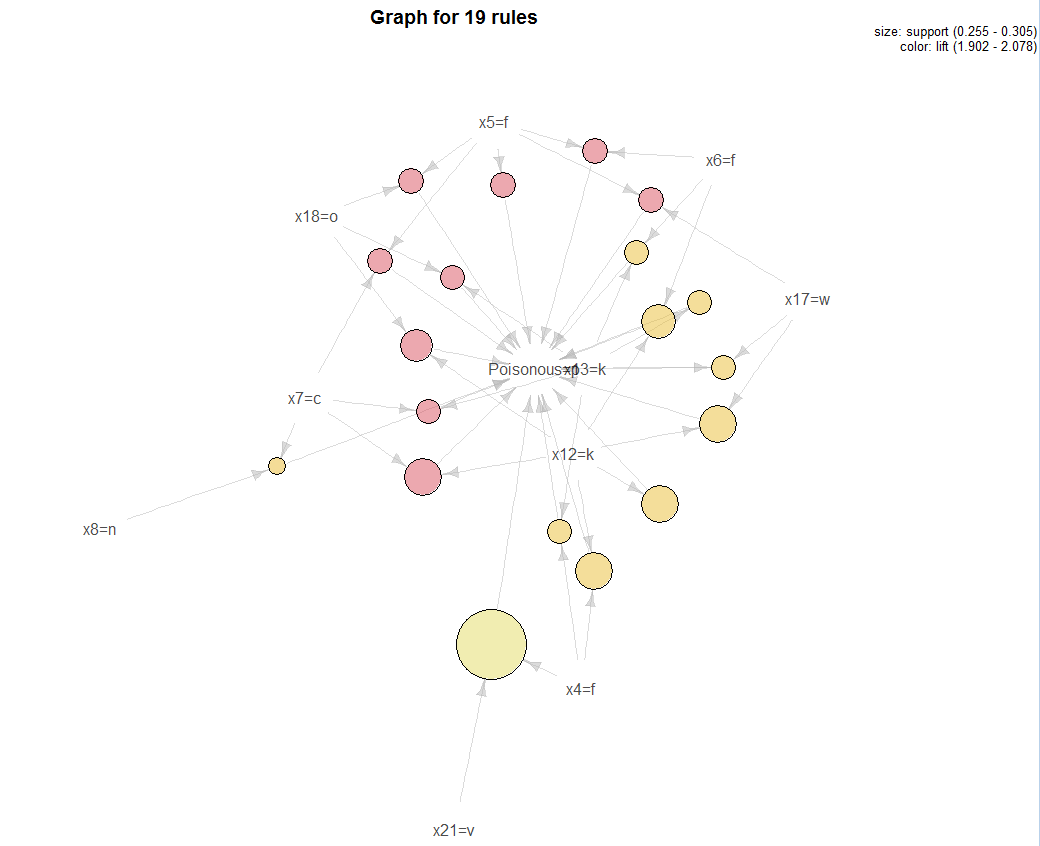
3rd Qu.:0.2724 3rd Qu.:1.0000 3rd Qu.:2.078

Max. :0.3045 Max. :1.0000 Max. :2.078

mining info:

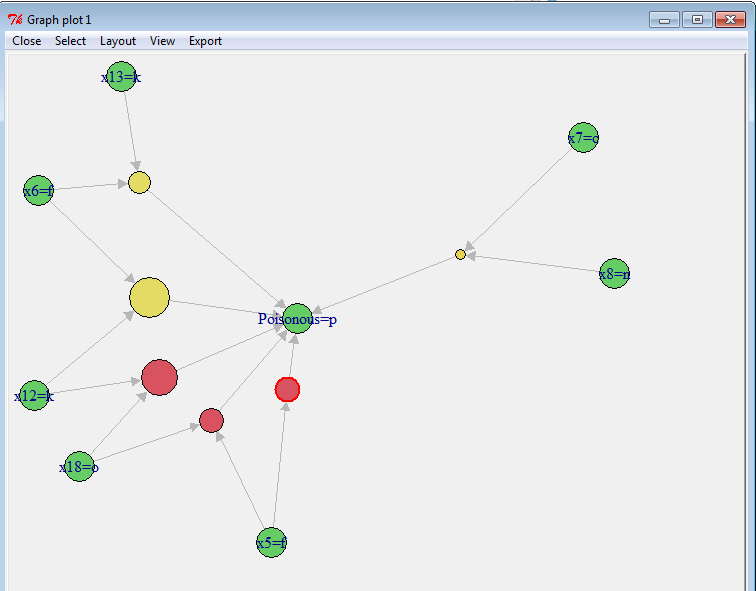
data ntransactions support confidence

mush.trans 4062 0.25 0.9



> sixrules = sample(Poison.rules,6)

> plot(sixrules,method="graph",interactive=T) 🡨 allows you to move nodes around!



In the latest version of arulesViz (version 1.3-3) there is a new function that makes exploring, inspecting, filtering, and graphing rules MUCH easier than the methods shown above. The function ruleExplorer allows users to explore association rules interactively.

> mush.rules = apriori(mush.trans)

Apriori

Parameter specification:

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

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

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 406

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

set transactions ...[113 item(s), 4062 transaction(s)] done [0.00s].

sorting and recoding items ... [52 item(s)] done [0.01s].

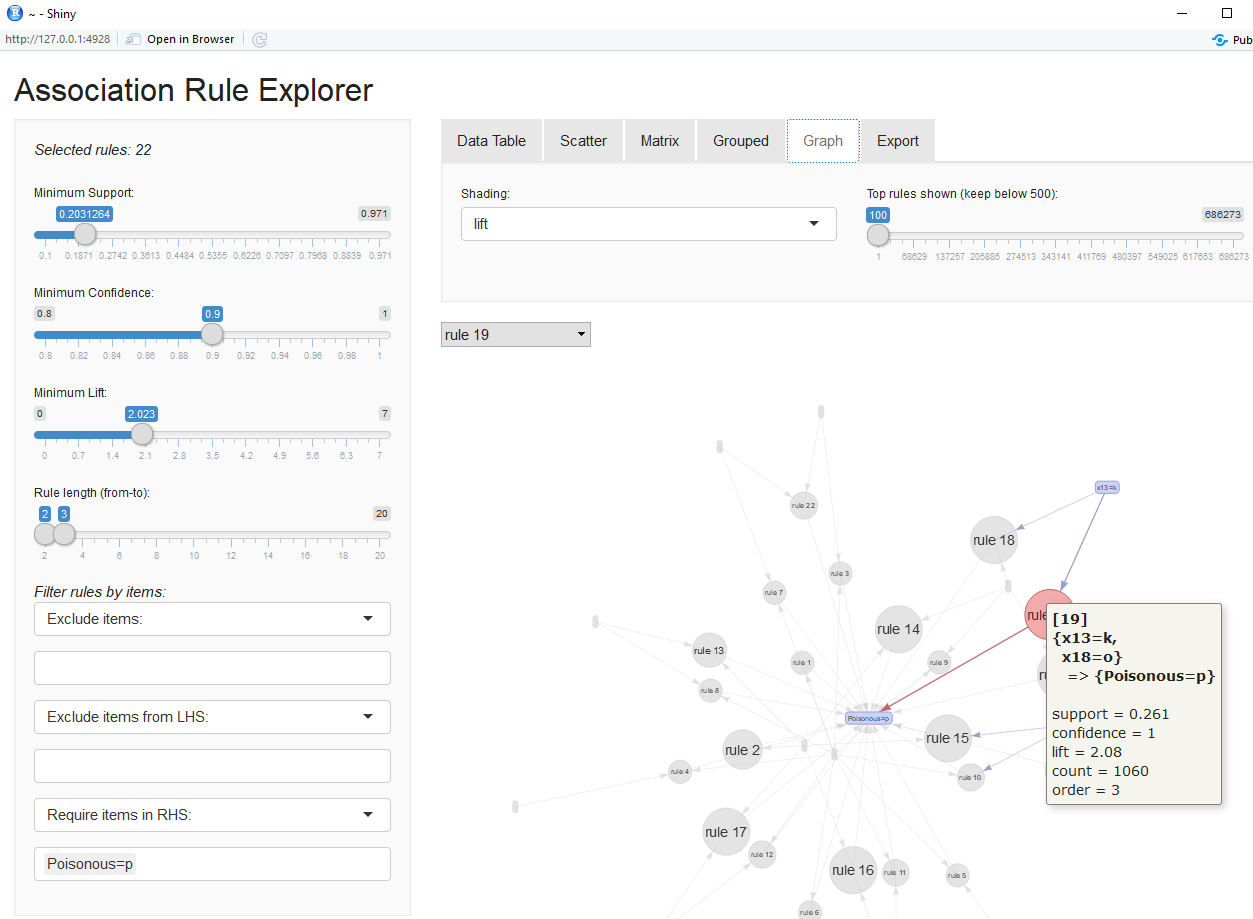
creating transaction tree ... done [0.00s].

checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.47s].

writing ... [686273 rule(s)] done [0.47s].

creating S4 object ... done [0.24s].

A screenshot of the results of running ruleExplorer on the full set of mushroom rules is shown on the following page. I have applied several filters to shorten the list of rules displayed. Specifically, I have chosen to focus on rules that imply that mushrooms are poisonous, have a support > 0.20, a confidence > 0.90, have a lift > 2.023 and are at most length 3.

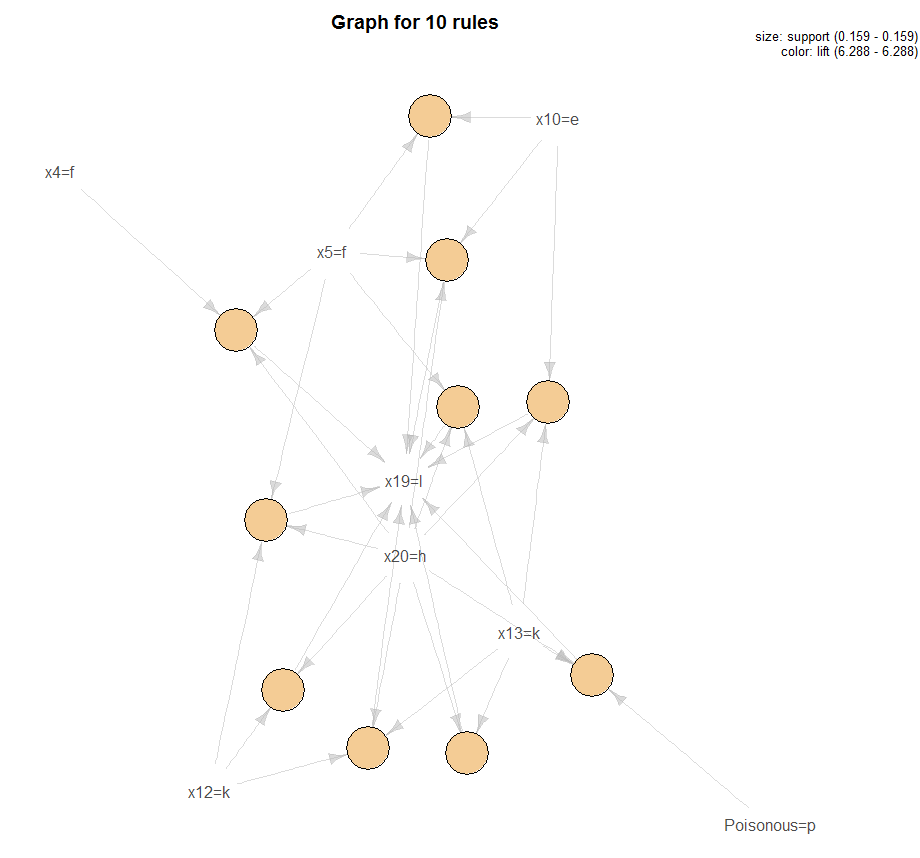
> ruleExplorer(mush.rules)  


**More examples of visualizing rules**

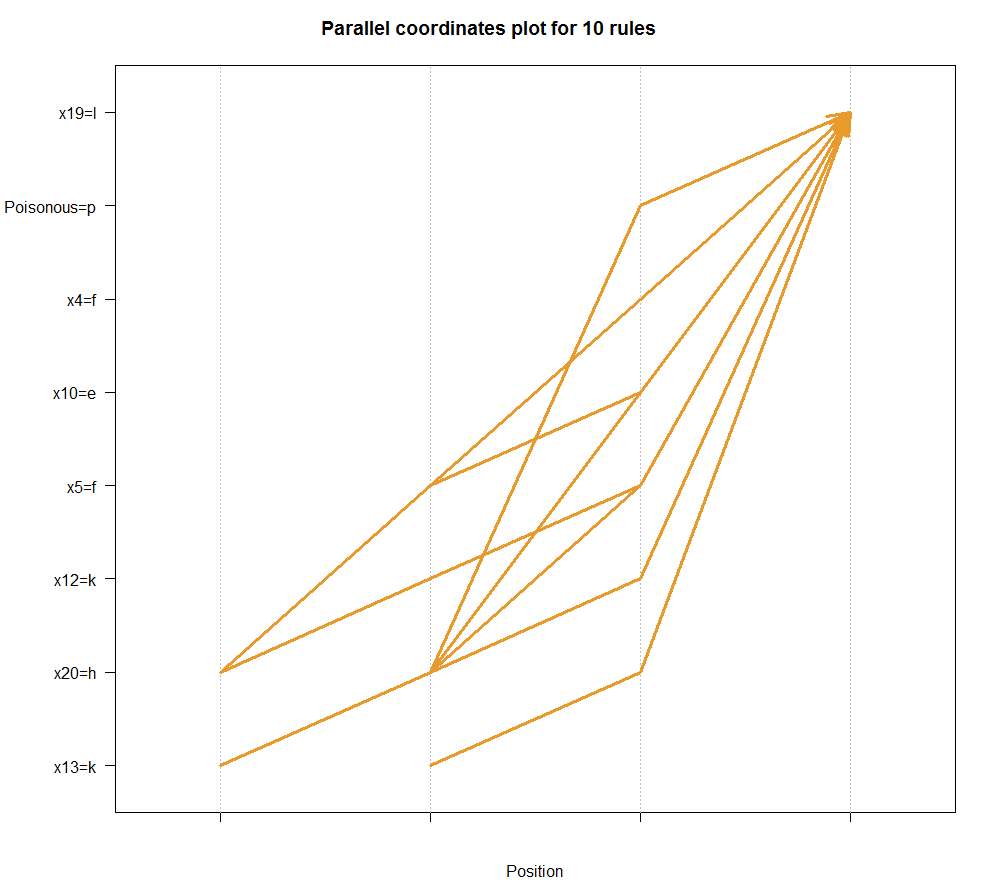
> mush.rules = apriori(mush.trans)

> subrules2 = head(sort(mush.rules,by="lift"),10) 🡨 save the 10 rules with highest lift

> plot(subrules2,method="graph")



> plot(subrules2,method=”paracoord”)



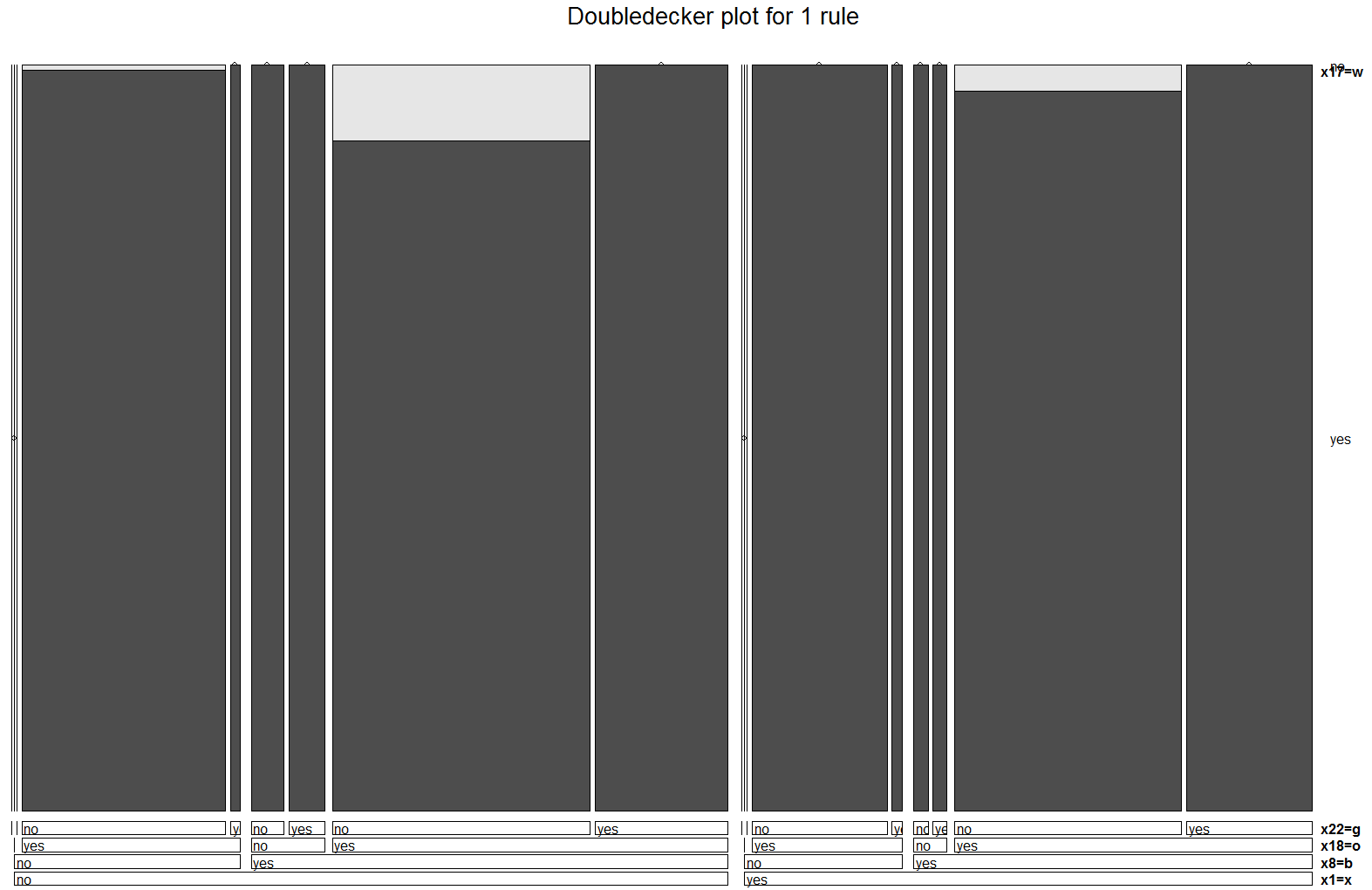
> oneRule = sample(mush.rules,1)

> inspect(oneRule)

lhs rhs support confidence lift

51231 {x1=x,x8=b,x18=o,x22=g} => {x17=w} 0.1033973 1 1.029919

> plot(oneRule,method="doubledecker",data=mush.trans)

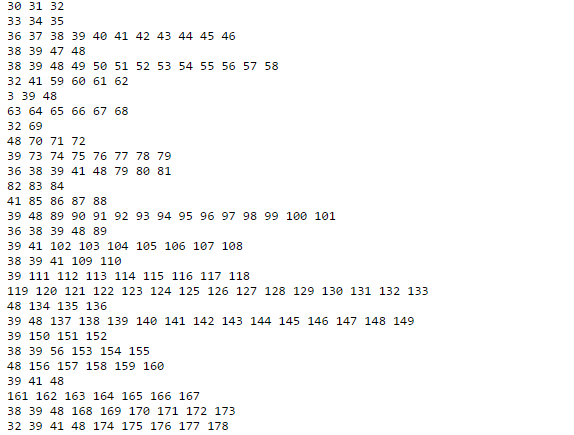


This a mosaic plot attempting to show the rule above graphically. These plots are often times difficult to read, particularly for rules with a long length.

Example 7.5: Retail Sales in a Belgian Supermarket

These data contain the market baskets from 88161 transactions where there are total of 16470 different items sold. The data is contained a text file where each row corresponds to the set of items bought by one of the 88162 customers. We wish to identify association rules within in this large transactional data base.

> BelgRetail = readLines("http://fimi.ua.ac.be/data/retail.dat")

🡨 note each line corresponds to a transaction with items purchased listed

> head(BelgRetail)

[1] "0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 " 🡨 This transaction does not look legitimate so we will remove it.

[2] "30 31 32 "

[3] "33 34 35 "

[4] "36 37 38 39 40 41 42 43 44 45 46 "

[5] "38 39 47 48 "

[6] "38 39 48 49 50 51 52 53 54 55 56 57 58 "

> BelgRetail = BelgRetail[-1] 🡨 Remove suspicious transaction

> BelgRetail = strsplit(BelgRetail," ") 🡨creates an array where each element is   
 transaction list for each customer.

> head(BelgRetail)

[[1]]

[1] "30" "31" "32"

[[2]]

[1] "33" "34" "35"

[[3]]

[1] "36" "37" "38" "39" "40" "41" "42" "43" "44" "45" "46"

[[4]]

[1] "38" "39" "47" "48"

[[5]]

[1] "38" "39" "48" "49" "50" "51" "52" "53" "54" "55" "56" "57" "58"

[[6]]

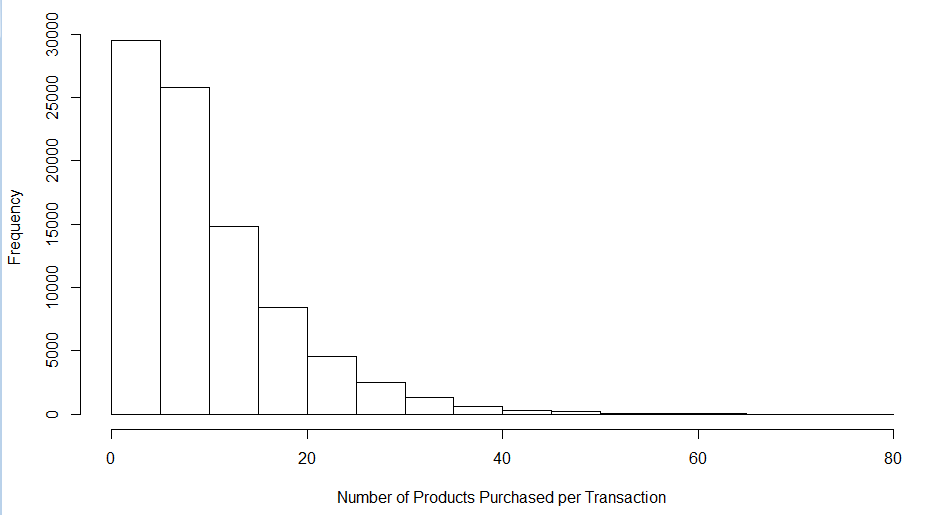
[1] "32" "41" "59" "60" "61" "62"

> numCustomers = length(BelgRetail)

> numCustomers

[1] 88161  
> prodCount = sapply(BelgRetail,length)

> hist(prodCount,xlab="Number of Products Purchased per Transaction")



> maxCount = max(prodCount)

> maxCount

[1] 76 🡨 maximum number of products bought by a single customer

> BelgRetail.trans = as(BelgRetail,"transactions") 🡨turn this into a transactional database

> Belg.rules = apriori(BelgRetail.trans)

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.1 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 8816

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

set transactions ...[16470 item(s), 88161 transaction(s)] done [0.11s].

sorting and recoding items ... [5 item(s)] done [0.00s].

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

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

writing ... [0 rule(s)] done [0.00s]. 🡨 NO RULES RETURNED!!! WHY DID THIS HAPPEN??

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

> summary(BelgRetail.trans)

transactions as itemMatrix in sparse format with

88161 rows (elements/itemsets/transactions) and

16470 columns (items) and a density of 0.0006257154

most frequent items:

39 48 38 32 41 (Other)

50675 42135 15596 15167 14945 770028

element (itemset/transaction) length distribution:

sizes

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

3016 5516 6919 7210 6814 6163 5746 5143 4660 4086 3751 3285 2866 2620 2310 2115 1874 1645 1469

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38

1290 1205 981 887 819 684 586 582 472 480 354 310 303 272 234 194 136 153 123

39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

115 112 76 66 71 60 50 44 37 37 33 22 24 21 21 10 11 10 9

58 59 60 61 62 63 64 65 66 67 68 71 73 74 76

11 4 9 7 4 5 2 2 5 3 3 1 1 1 1

Min. 1st Qu. Median Mean 3rd Qu. Max.

1.00 4.00 8.00 10.31 14.00 76.00

includes extended item information - examples:

labels

1 0

2 1

3 10  
  
> Belg.rules = apriori(BelgRetail.trans,parameter=list(supp=.005))  
Apriori

Parameter specification:

confidence minval smax arem aval originalSupport support minlen maxlen target ext

0.8 0.1 1 none FALSE TRUE 0.005 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 440

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

set transactions ...[16470 item(s), 88161 transaction(s)] done [0.12s].

sorting and recoding items ... [221 item(s)] done [0.01s].

creating transaction tree ... done [0.03s].

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

writing ... [62 rule(s)] done [0.00s].

> summary(Belg.rules)

set of 62 rules

rule length distribution (lhs + rhs):sizes

2 3 4 5

13 31 17 1

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.0 3.0 3.0 3.1 4.0 5.0

summary of quality measures:

support confidence lift

Min. :0.005 Min. :0.80 Min. : 1

1st Qu.:0.006 1st Qu.:0.84 1st Qu.: 2

Median :0.007 Median :0.96 Median : 5

Mean :0.011 Mean :0.92 Mean : 5

3rd Qu.:0.012 3rd Qu.:0.98 3rd Qu.: 6

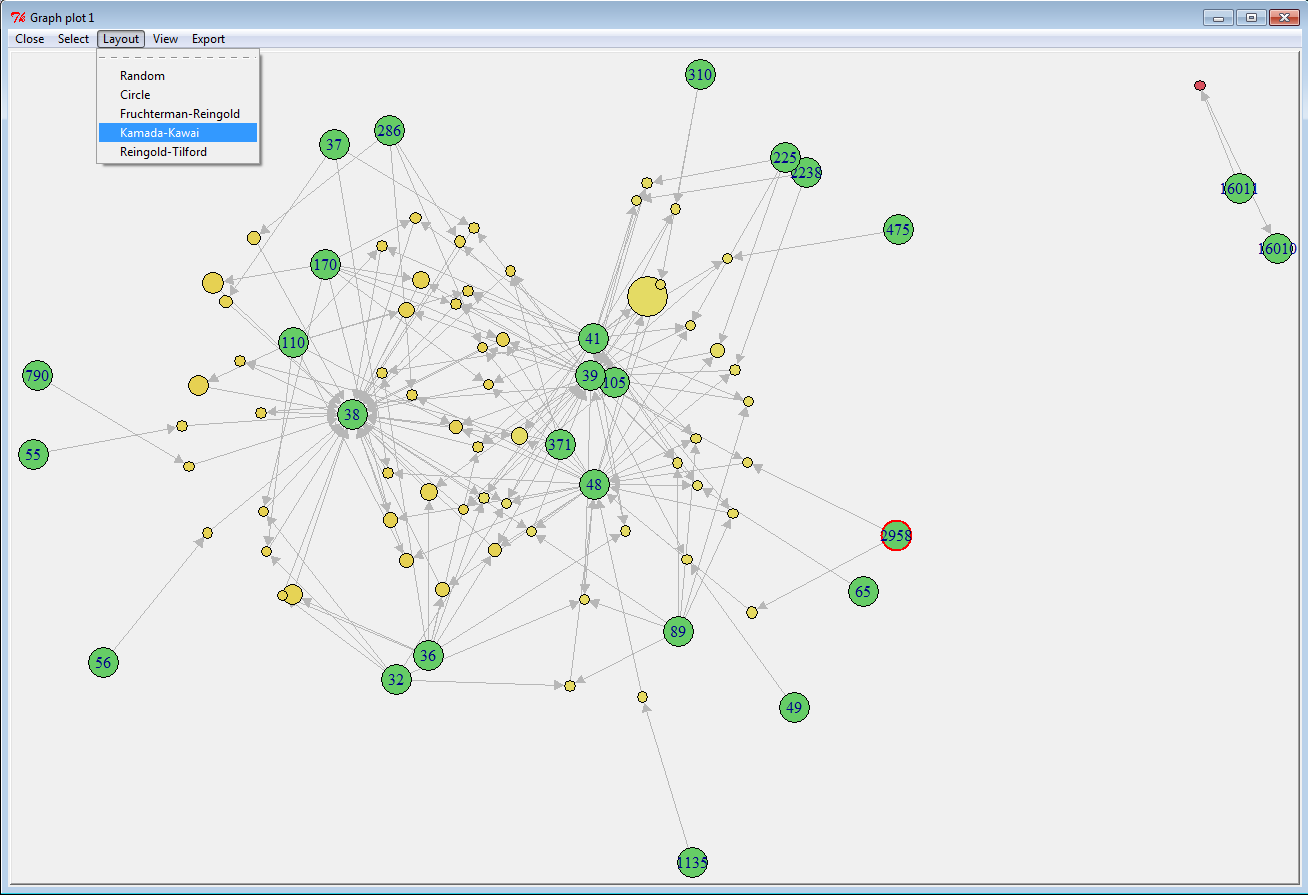
Max. :0.084 Max. :0.99 Max. :65

mining info:

data ntransactions support confidence

BelgRetail.trans 88161 0.005 0.8

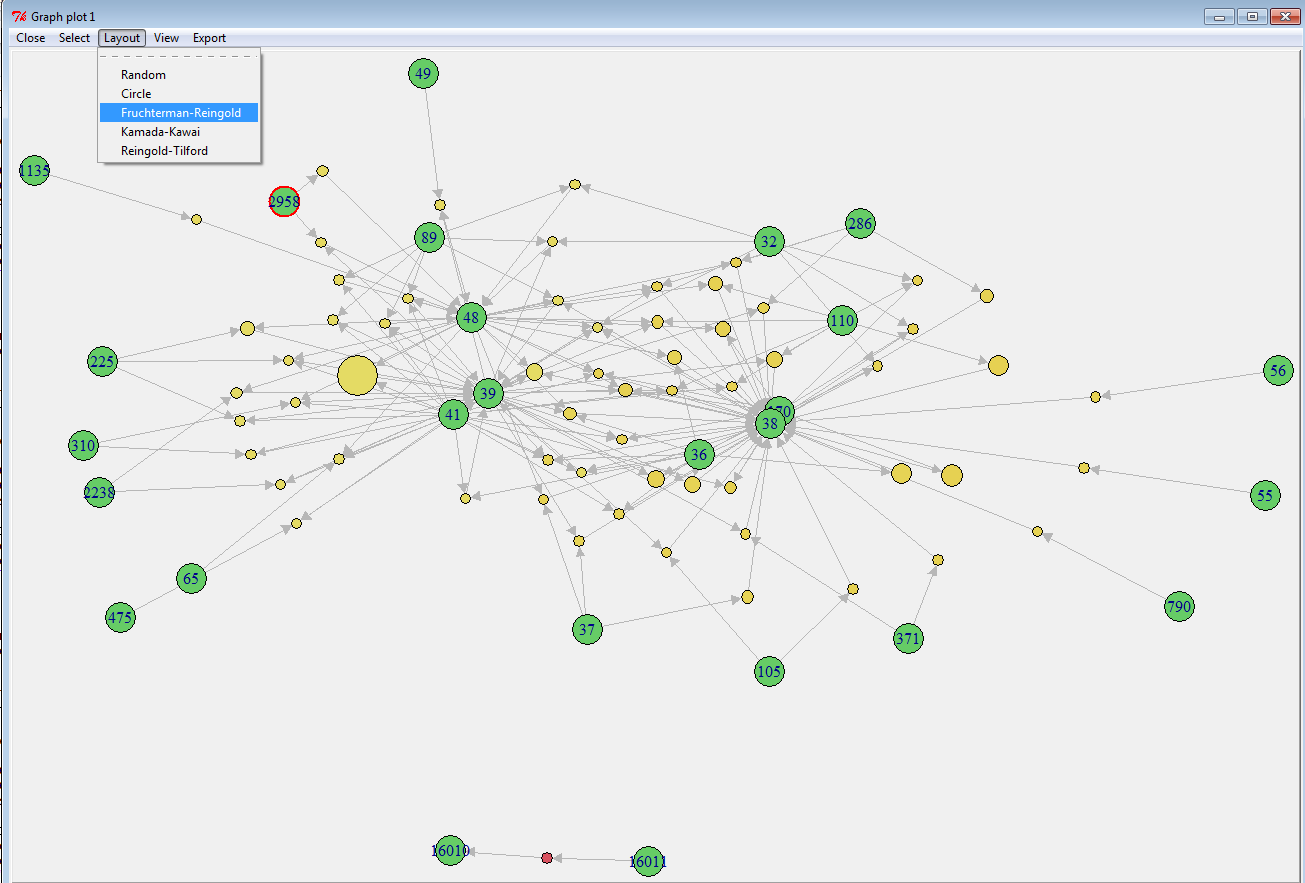
> plot(Belg.rules,method=”graph”,interactive=T)🡨 allows you to move nodes around



There are optimal node layout algorithms called *Forced Directed Graph Drawing* techniques that allow you to more clearly see the association rules. The Kamada-Kawai and the Fruchterman-Reingold both produce nice displays of the rules for these data.

Because of the large number of potential items in the retail store and the fairly small numbers of items per cart, we need to lower the support substantially in order find some “quality” association rules. Most of the rules have RHS resulting in items 32, 36, 38, 39, 48 and there is one very strong rule associating items 16010 & 16011. The lift values for some of the rules found are very large, indicating strong associations between some of the items in this fairly sparse transactional database.

> plot(Belg.rules,method=”graph”,interactive=T) 🡨 allows you to move nodes around



Again, we can accomplish much of what is shown in the example above using ruleExplorer.

> ruleExplorer(Belg.rules)

