

title: "AssociationVersion1" output: html_document

Apriori algorithm is given by R. Agrawal and R. Srikant for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find itemsets. To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space. Apriori Property – All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Parameters 1 Support This measure gives an idea of how frequent an itemset is in all the transactions. 2. Confidence This measure defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents. 3. Lift Lift controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}.

...

```
library(arulesViz)
```

```
## Loading required package: arules
```

```
## Loading required package: Matrix
```

```
##  
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':  
##  
## abbreviate, write
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':  
## method from  
## reorder.hclust gclus
```

1. Load the Groceries dataset and apply apriori algorithm with support as 0.005 and confidence as 0.5 the data.

```
data(Groceries)  
rules <- apriori(Groceries, parameter=list(support=0.005, confidence=0.5))
```

```
## Apriori  
##  
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.5 0.1 1 none FALSE TRUE 5 0.005 1  
## maxlen target ext  
## 10 rules FALSE  
##  
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##  
## Absolute minimum support count: 49  
##  
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [120 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## writing ... [120 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].
```

```
rules
```

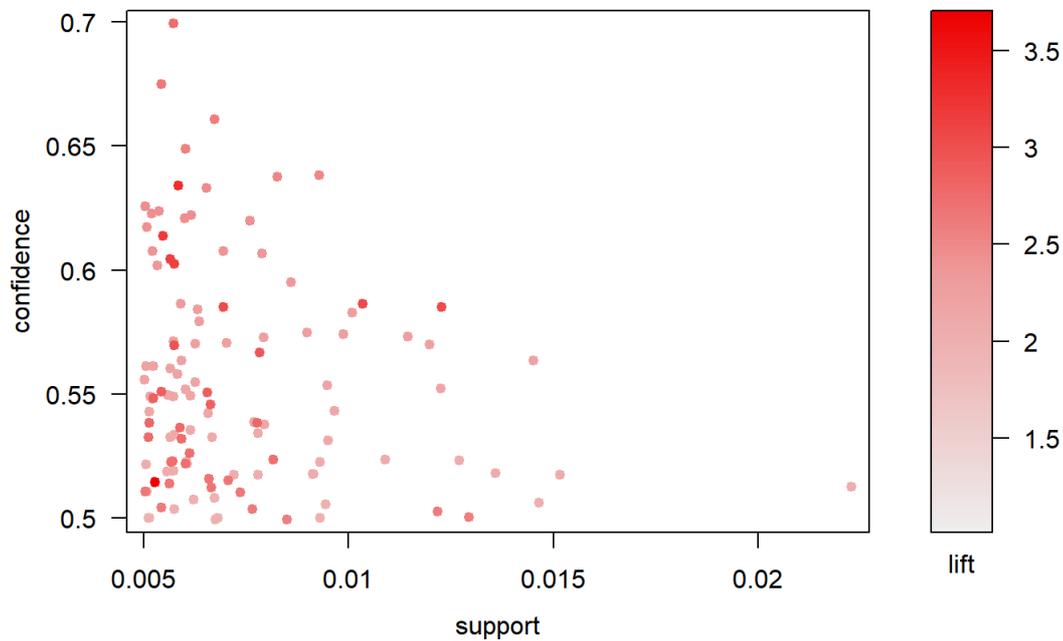
```
## set of 120 rules
```

2. Plot the rules generated by the apriori algorithm

```
# Scatterplot
plot(rules)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

Scatter plot for 120 rules



```
# try sel <- plot(rules, interactive=TRUE)
```

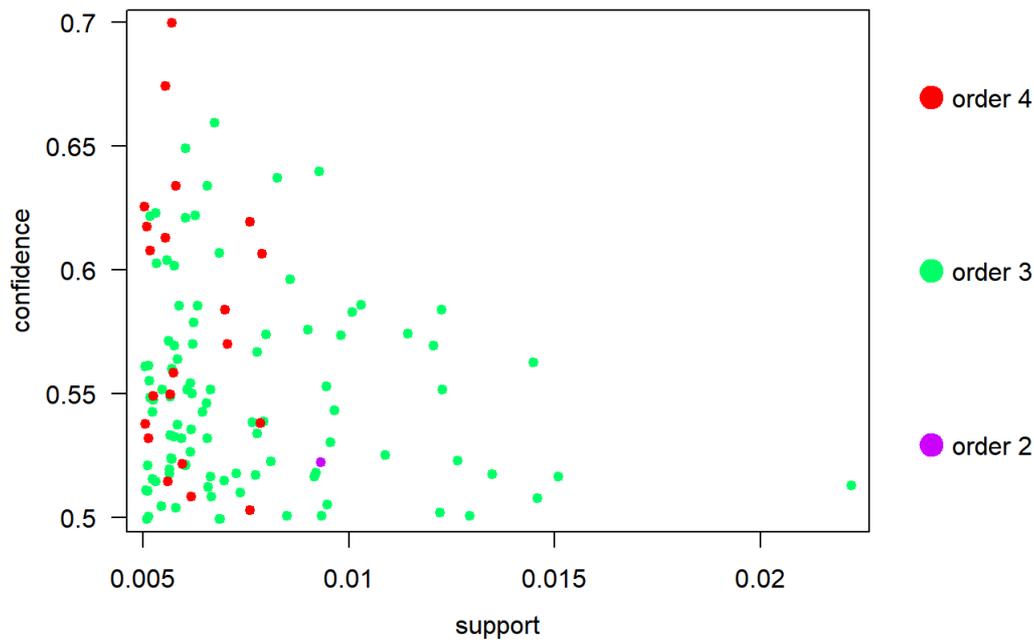
Above scatter plot shows the 120 rules generated by the apriori algorithm for the Groceries dataset with support=0.005, confidence=0.5

3. Group the rules based on order

```
## Two-key plot is a scatterplot with shading = "order"
plot(rules, shading="order", control=list(main = "Two-key plot",
col=rainbow(5)))
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

Two-key plot



This visualization method draws a two dimensional scatterplot with different measures of interestingness (support and confidence) on the axes and a third measure (parameter "shading") is represented by the points color. There is a special value for shading called "order". With this value the color of the points represents the length (order) of the rule. This is used for two-key plots.

4. 2D matrix with shading. The following techniques work better with fewer rules for better visualization

```
subrules <- subset(rules, lift>2.5)
subrules
```

```
## set of 41 rules
```

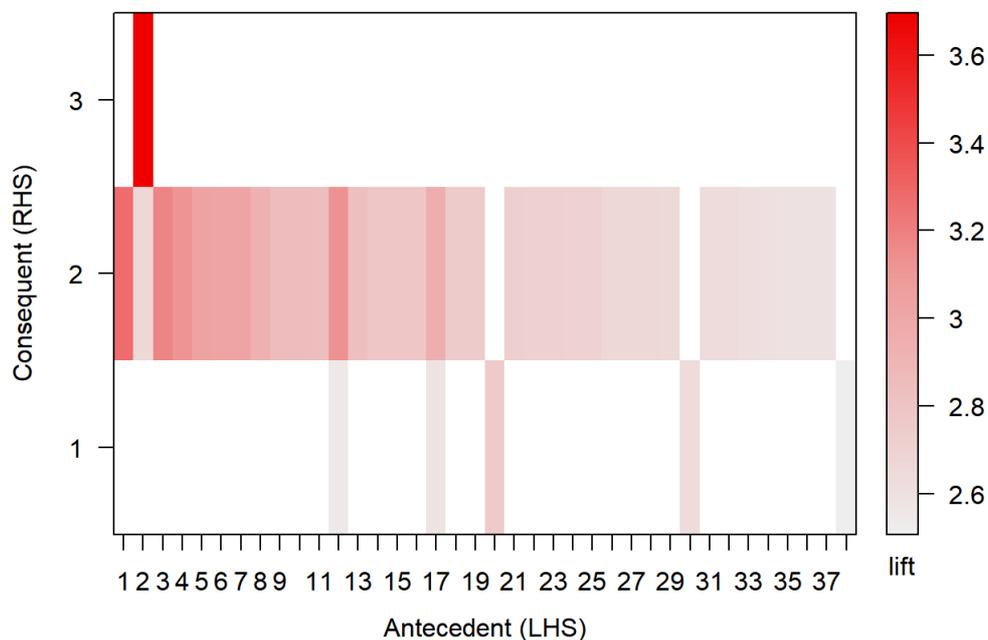
```
plot(subrules, method="matrix", measure="lift")
```

```

## Itemsets in Antecedent (LHS)
## [1] "{citrus fruit,root vegetables,whole milk}"
## [2] "{tropical fruit,curd}"
## [3] "{pip fruit,root vegetables,whole milk}"
## [4] "{root vegetables,onions}"
## [5] "{citrus fruit,root vegetables}"
## [6] "{tropical fruit,root vegetables,whole milk}"
## [7] "{tropical fruit,root vegetables}"
## [8] "{tropical fruit,whipped/sour cream}"
## [9] "{tropical fruit,butter}"
## [10] "{root vegetables,fruit/vegetable juice}"
## [11] "{root vegetables,whole milk,whipped/sour cream}"
## [12] "{pip fruit,whipped/sour cream}"
## [13] "{onions,whole milk}"
## [14] "{root vegetables,whole milk,yogurt}"
## [15] "{whole milk,yogurt,fruit/vegetable juice}"
## [16] "{root vegetables,pastry}"
## [17] "{butter,whipped/sour cream}"
## [18] "{root vegetables,margarine}"
## [19] "{pip fruit,whole milk,yogurt}"
## [20] "{tropical fruit,root vegetables,yogurt}"
## [21] "{root vegetables,frozen vegetables}"
## [22] "{chicken,root vegetables}"
## [23] "{citrus fruit,whipped/sour cream}"
## [24] "{pip fruit,root vegetables}"
## [25] "{root vegetables,newspapers}"
## [26] "{root vegetables,shopping bags}"
## [27] "{pork,root vegetables}"
## [28] "{whole milk,yogurt,whipped/sour cream}"
## [29] "{root vegetables,butter}"
## [30] "{pip fruit,root vegetables,other vegetables}"
## [31] "{root vegetables,domestic eggs}"
## [32] "{whipped/sour cream,domestic eggs}"
## [33] "{root vegetables,curd}"
## [34] "{tropical fruit,whole milk,yogurt}"
## [35] "{root vegetables,rolls/buns}"
## [36] "{root vegetables,whipped/sour cream}"
## [37] "{root vegetables,yogurt}"
## [38] "{butter,yogurt}"
## Itemsets in Consequent (RHS)
## [1] "{whole milk}"      "{other vegetables}" "{yogurt}"

```

Matrix with 41 rules



It arranges the association rules as a matrix with the itemsets in the antecedents on one axis and the itemsets in the consequent on the other.

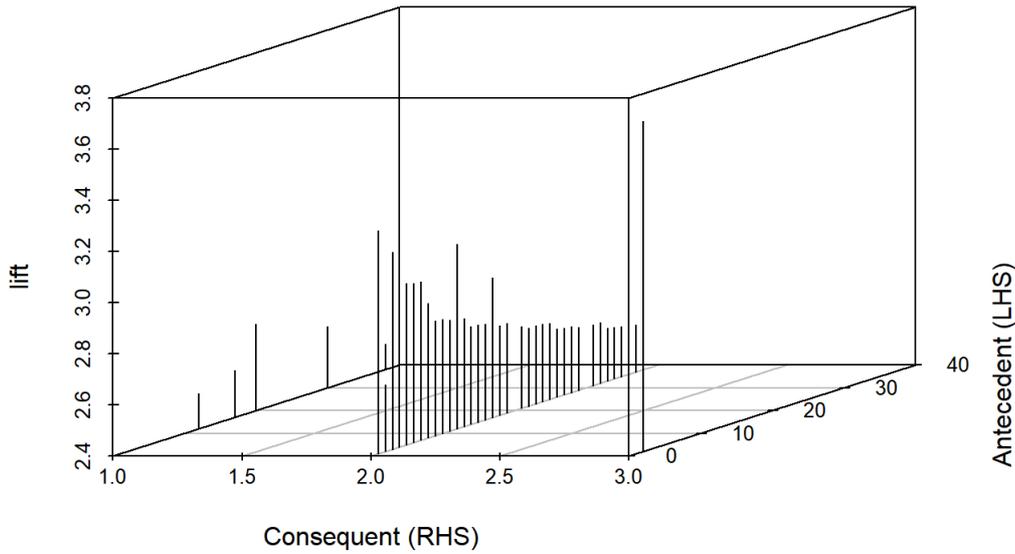
5. 3D Matrix visualization

```
plot(subrules, method="matrix3D", measure="lift")
```

```
## Warning in plot.rules(subrules, method = "matrix3D", measure = "lift"): method
## 'matrix3D' is deprecated use method 'matrix' with engine '3d'
```

```
## Itemsets in Antecedent (LHS)
## [1] "{citrus fruit,root vegetables,whole milk}"
## [2] "{tropical fruit,curd}"
## [3] "{pip fruit,root vegetables,whole milk}"
## [4] "{root vegetables,onions}"
## [5] "{citrus fruit,root vegetables}"
## [6] "{tropical fruit,root vegetables,whole milk}"
## [7] "{tropical fruit,root vegetables}"
## [8] "{tropical fruit,whipped/sour cream}"
## [9] "{tropical fruit,butter}"
## [10] "{root vegetables,fruit/vegetable juice}"
## [11] "{root vegetables,whole milk,whipped/sour cream}"
## [12] "{pip fruit,whipped/sour cream}"
## [13] "{onions,whole milk}"
## [14] "{root vegetables,whole milk,yogurt}"
## [15] "{whole milk,yogurt,fruit/vegetable juice}"
## [16] "{root vegetables,pastry}"
## [17] "{butter,whipped/sour cream}"
## [18] "{root vegetables,margarine}"
## [19] "{pip fruit,whole milk,yogurt}"
## [20] "{tropical fruit,root vegetables,yogurt}"
## [21] "{root vegetables,frozen vegetables}"
## [22] "{chicken,root vegetables}"
## [23] "{citrus fruit,whipped/sour cream}"
## [24] "{pip fruit,root vegetables}"
## [25] "{root vegetables,newspapers}"
## [26] "{root vegetables,shopping bags}"
## [27] "{pork,root vegetables}"
## [28] "{whole milk,yogurt,whipped/sour cream}"
## [29] "{root vegetables,butter}"
## [30] "{pip fruit,root vegetables,other vegetables}"
## [31] "{root vegetables,domestic eggs}"
## [32] "{whipped/sour cream,domestic eggs}"
## [33] "{root vegetables,curd}"
## [34] "{tropical fruit,whole milk,yogurt}"
## [35] "{root vegetables,rolls/buns}"
## [36] "{root vegetables,whipped/sour cream}"
## [37] "{root vegetables,yogurt}"
## [38] "{butter,yogurt}"
## Itemsets in Consequent (RHS)
## [1] "{whole milk}"      "{other vegetables}" "{yogurt}"
```

Matrix with 41 rules

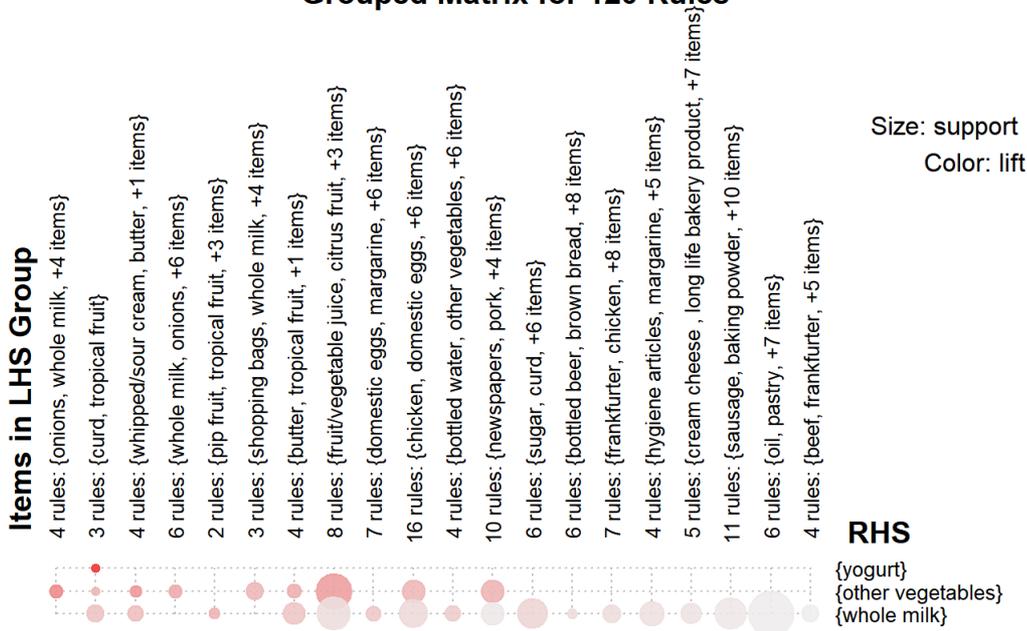


Matrix3D Arranges the association rules as a matrix with the itemsets in the antecedents on one axis, the itemsets in the consequent on the other and lift in the third dimension

6. grouped matrix plot

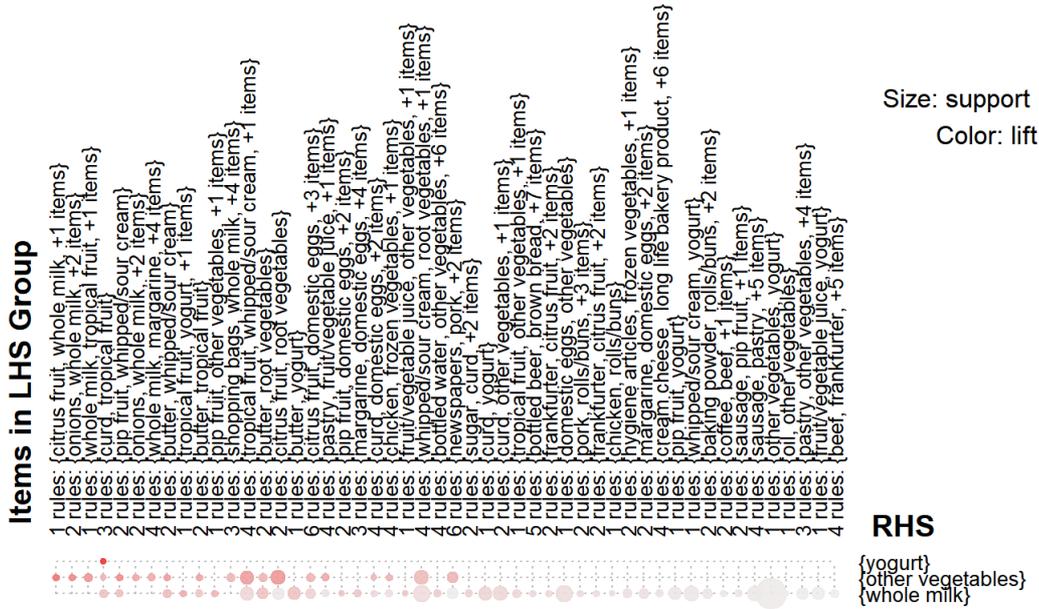
```
plot(rules, method="grouped")
```

Grouped Matrix for 120 Rules



```
plot(rules, method = "grouped", control = list(k = 50))
```

Grouped Matrix for 120 Rules



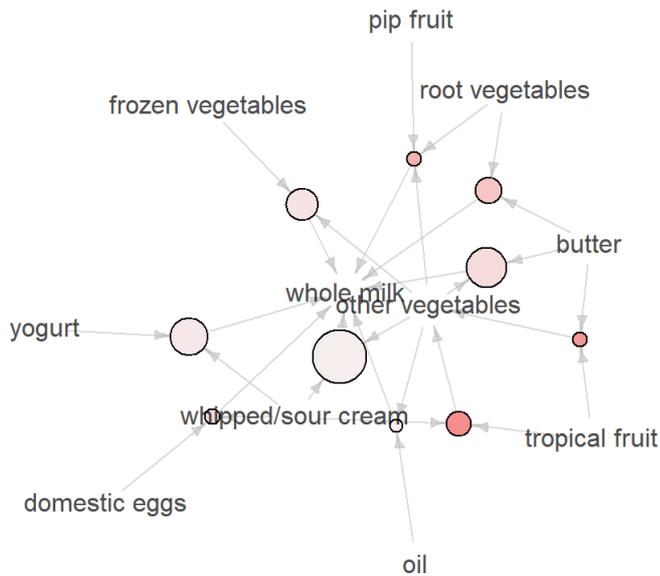
Grouped matrix-based visualization Antecedents (columns) in the matrix are grouped using clustering. Groups are represented by the most interesting item (highest ratio of support in the group to support in all rules) in the group. Balloons in the matrix are used to represent with what consequent the antecedents are connected

7. Visualization of rules using Graphs

```
# graphs only work well with very few rules
subrules2 <- sample(rules, 10)
plot(subrules2, method="graph")
```

Graph for 10 rules

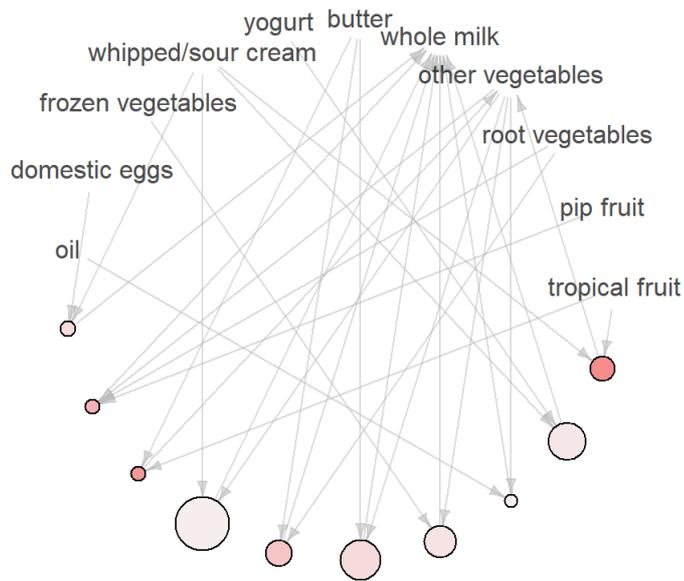
size: support (0.005 - 0.015)
color: lift (1.984 - 2.926)



```
plot(subrules2, method="graph", control=list(layout=igraph::in_circle()))
```

Graph for 10 rules

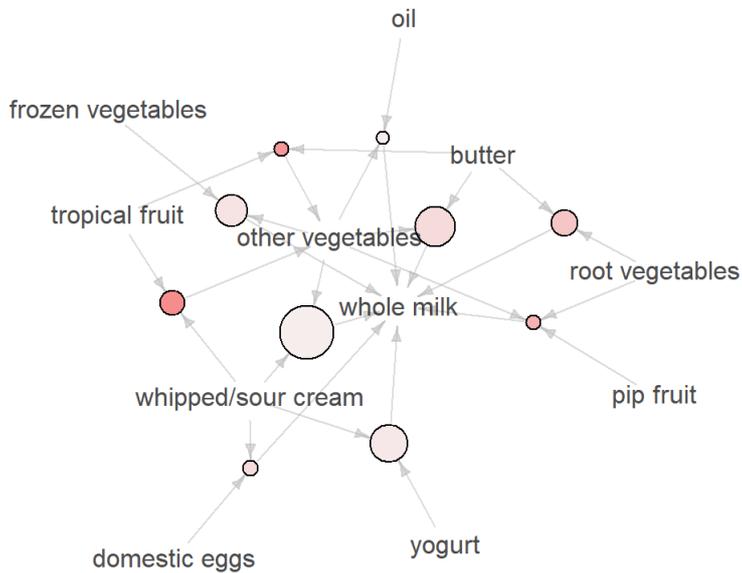
size: support (0.005 - 0.015)
color: lift (1.984 - 2.926)



```
plot(subrules2, method="graph", control=list(  
  layout=igraph::with_graphopt(spring.const=5, mass=50)))
```

Graph for 10 rules

size: support (0.005 - 0.015)
color: lift (1.984 - 2.926)



```
plot(subrules2, method="graph", control=list(type="itemsets"))
```

```
## Warning: Unknown control parameters: type
```

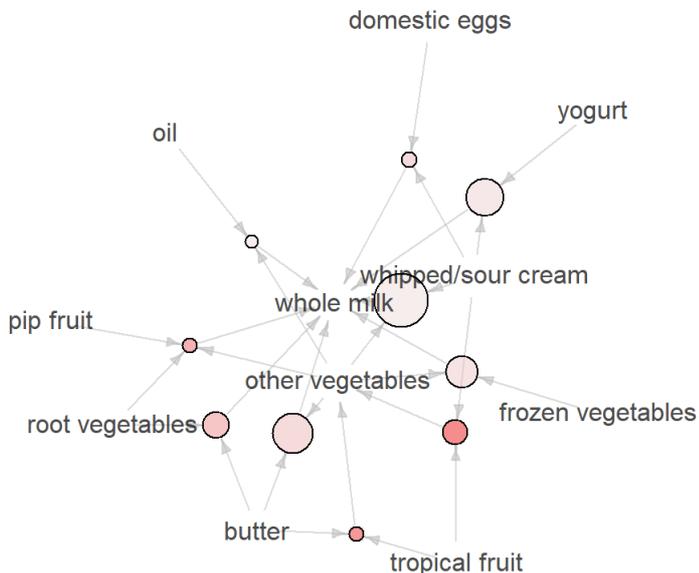
```

## Available control parameters (with default values):
## main = Graph for 10 rules
## nodeColors = c("#66CC6680", "#9999CC80")
## nodeCol = c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF",
"#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE
2E2EFF", "#EE3131FF", "#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747
FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5FFF",
"#EE6262FF", "#EE6666FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF", "#EE7878FF", "#E
E7B7BFF", "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE949
4FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF"
, "#EEA5A5FF", "#EEA7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1FF", "#
EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#EEBDBDFF", "#EEBFBF", "#EEC1
C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC6C6FF", "#EEC8C8FF", "#EEC9C9FF", "#EECBCBFF", "#EECD CDFF", "#EECF CF"
, "#EED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCD CFF",
"#EED EDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAEFF", "#EE
ECECFF", "#EEEEEEFF")
## edgeCol = c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF",
"#555555FF", "#575757FF", "#595959FF", "#5B5B5BFF", "#5E5E5EFF", "#606060FF", "#626262FF", "#646464FF", "#66
6666FF", "#686868FF", "#6A6A6AFF", "#6C6C6CFF", "#6E6E6EFF", "#707070FF", "#727272FF", "#747474FF", "#767676
FF", "#787878FF", "#7A7A7AFF", "#7C7C7CFF", "#7E7E7EFF", "#808080FF", "#828282FF", "#848484FF", "#868686FF",
"#888888FF", "#8A8A8AFF", "#8C8C8CFF", "#8D8D8DFF", "#8F8F8FFF", "#919191FF", "#939393FF", "#959595FF", "#9
79797FF", "#999999FF", "#9A9A9AFF", "#9C9C9CFF", "#9E9E9EFF", "#A0A0A0FF", "#A2A2A2FF", "#A3A3A3FF", "#A5A5A
5FF", "#A7A7A7FF", "#A9A9A9FF", "#AAAAAAFF", "#ACACACFF", "#AEAEAEFF", "#AF AF AF FF", "#B1B1B1FF", "#B3B3B3FF"
, "#B4B4B4FF", "#B6B6B6FF", "#B7B7B7FF", "#B9B9B9FF", "#BBBBBBFF", "#BCBCBCFF", "#BEBEBEFF", "#BFBFBFFF", "#
C1C1C1FF", "#C2C2C2FF", "#C3C3C4FF", "#C5C5C5FF", "#C6C6C6FF", "#C8C8C8FF", "#C9C9C9FF", "#CACACAFF", "#CCCC
CCFF", "#CDCDCDFF", "#CECECEFF", "#CFCFCFFF", "#D1D1D1FF", "#D2D2D2FF", "#D3D3D3FF", "#D4D4D4FF", "#D5D5D5F
F", "#D6D6D6FF", "#D7D7D7FF", "#D8D8D8FF", "#D9D9D9FF", "#DADADAFF", "#DBDBDBFF", "#DCDCDCFF", "#DDDDDDFF",
"#DEDEDEFF", "#DEDEDEFF", "#DFDFDFFF", "#E0E0E0FF", "#E0E0E0FF", "#E1E1E1FF", "#E1E1E1FF", "#E2E2E2FF", "#E2
E2E2FF", "#E2E2E2FF")
## alpha = 0.5
## cex = 1
## itemLabels = TRUE
## labelCol = #000000B3
## measureLabels = FALSE
## precision = 3
## layout = NULL
## layoutParams = list()
## arrowSize = 0.5
## engine = igraph
## plot = TRUE
## plot_options = list()
## max = 100
## verbose = FALSE

```

Graph for 10 rules

size: support (0.005 - 0.015)
color: lift (1.984 - 2.926)

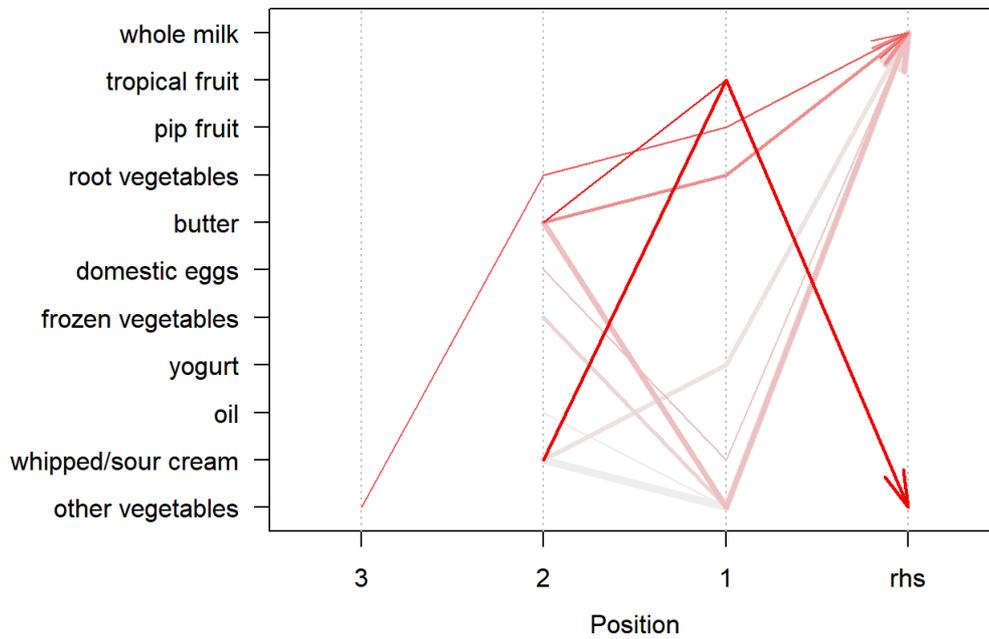


It represents the rules (or itemsets) as a graph with items as labeled vertices, and rules (or itemsets) represented as vertices connected to items using arrows. For rules, the LHS items are connected with arrows pointing to the vertex representing the rule and the RHS has an arrow pointing to the item.

8. Visualization of rules using parallel coordinates

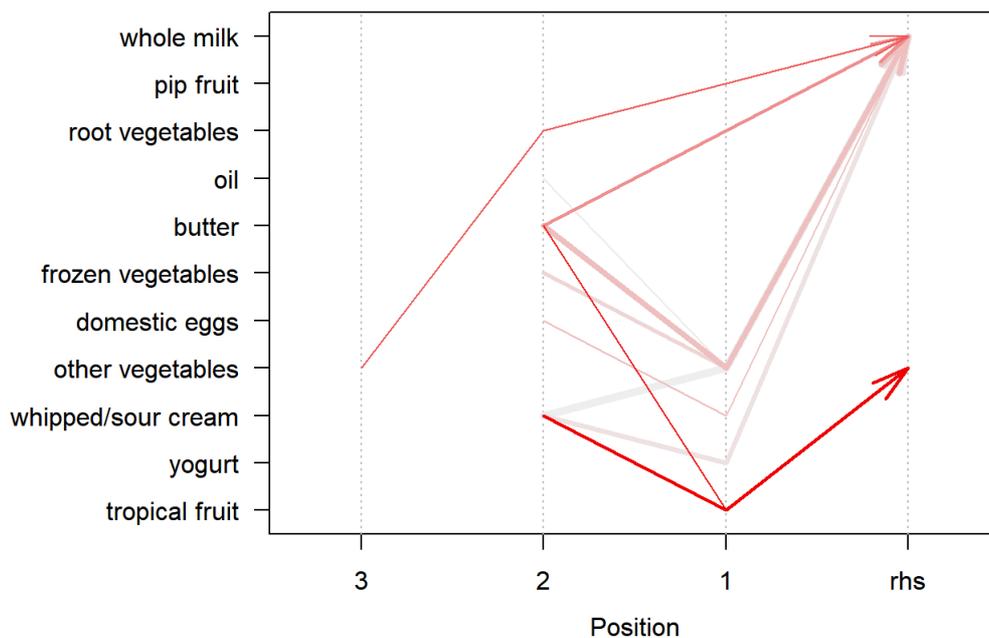
```
plot(subrules2, method = "paracoord")
```

Parallel coordinates plot for 10 rules



```
plot(subrules2, method = "paracoord", control = list(reorder = TRUE))
```

Parallel coordinates plot for 10 rules



It represents the rules as a parallel coordinate plot.

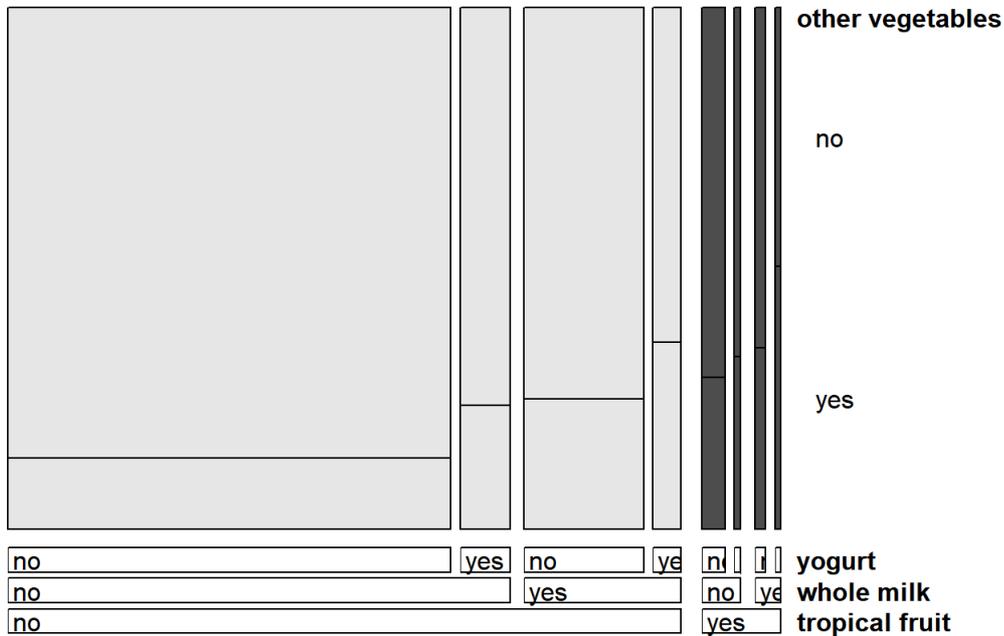
9. Visualization of one rule using doubledecker plot

```
oneRule <- sample(rules, 1)
inspect(oneRule)
```

```
##      lhs                rhs                support confidence    lift count
## [1] {tropical fruit,
##      whole milk,
##      yogurt}          => {other vegetables} 0.007625826 0.5033557 2.601421    75
```

```
plot(oneRule, method = "doubledecker", data = Groceries)
```

Doubledecker plot for 1 rule



It represents a single rule as a doubledecker or mosaic plot. Parameter data has to be specified to compute the needed contingency table.

10. Itemset Visualization as Graph

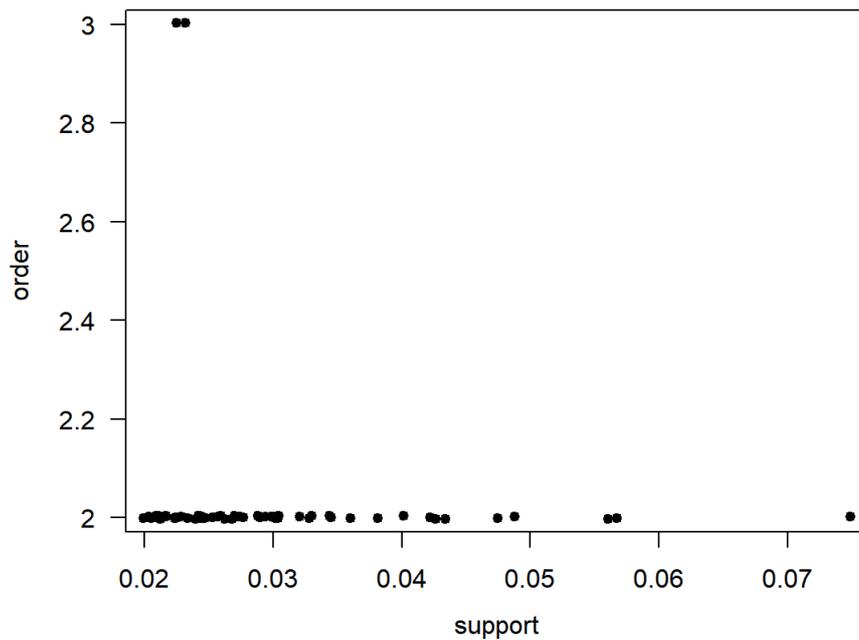
```
itemsets <- eclat(Groceries, parameter = list(support = 0.02, minlen=2))
```

```
## Eclat
##
## parameter specification:
## tidLists support minlen maxlen          target  ext
##   FALSE   0.02     2     10 frequent itemsets FALSE
##
## algorithmic control:
## sparse sort verbose
##     7  -2   TRUE
##
## Absolute minimum support count: 196
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [59 item(s)] done [0.00s].
## creating sparse bit matrix ... [59 row(s), 9835 column(s)] done [0.00s].
## writing ... [63 set(s)] done [0.01s].
## Creating S4 object ... done [0.00s].
```

```
plot(itemsets)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

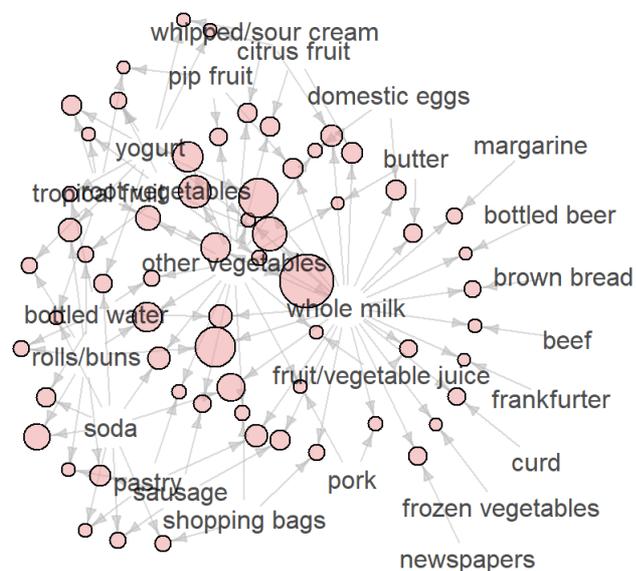
Scatter plot for 63 itemsets



```
plot(itemsets, method="graph")
```

Graph for 63 itemsets

size: support (0.02 - 0.075)



The `eclat()` takes in a transactions object and gives the most frequent items in the data based the support argument. The `maxlen` defines the maximum number of items in each itemset of frequent items. This plot used to represent the most frequent items in the dataset as graph with support value as 0.02 and length as 2

11. Itemset Graph visualization with various support values

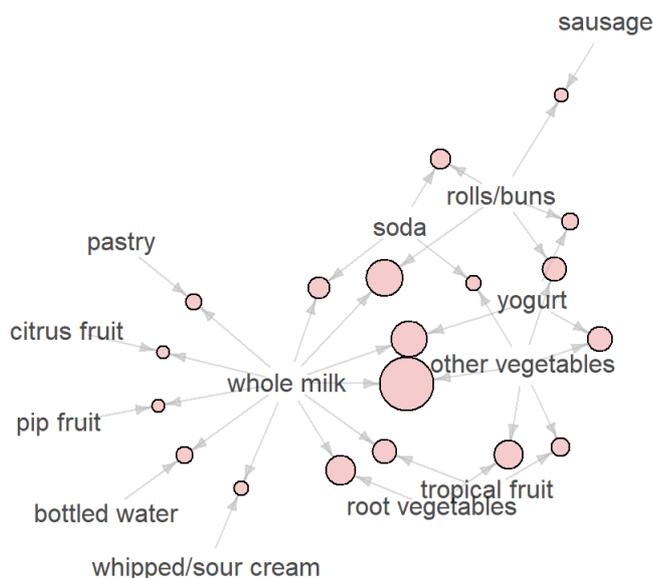
```
itemsets <- eclat(Groceries, parameter = list(support = 0.03, minlen=2))
```

```
## Eclat
##
## parameter specification:
## tidLists support minlen maxlen          target  ext
## FALSE 0.03 2 10 frequent itemsets FALSE
##
## algorithmic control:
## sparse sort verbose
## 7 -2 TRUE
##
## Absolute minimum support count: 295
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [44 item(s)] done [0.00s].
## creating sparse bit matrix ... [44 row(s), 9835 column(s)] done [0.00s].
## writing ... [19 set(s)] done [0.01s].
## Creating S4 object ... done [0.00s].
```

```
plot(itemsets, method="graph")
```

Graph for 19 itemsets

size: support (0.03 - 0.075)



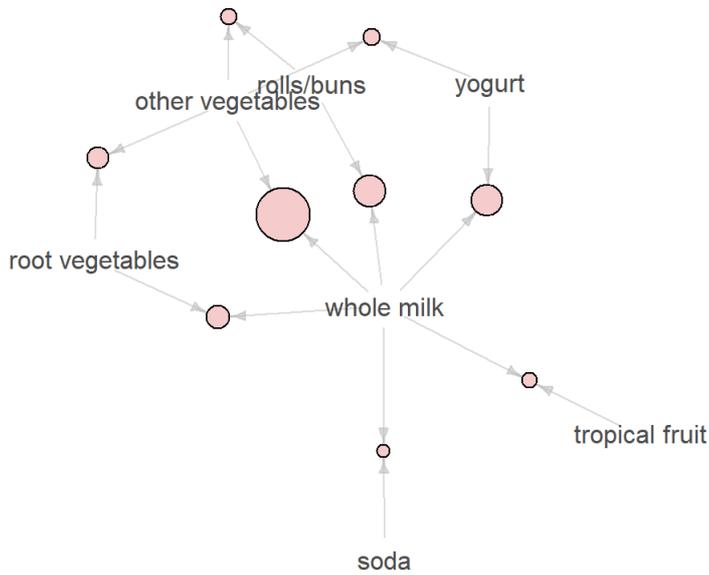
```
itemsets <- eclat(Groceries, parameter = list(support = 0.04, minlen=2))
```

```
## Eclat
##
## parameter specification:
## tidLists support minlen maxlen          target  ext
## FALSE 0.04 2 10 frequent itemsets FALSE
##
## algorithmic control:
## sparse sort verbose
## 7 -2 TRUE
##
## Absolute minimum support count: 393
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [32 item(s)] done [0.00s].
## creating sparse bit matrix ... [32 row(s), 9835 column(s)] done [0.00s].
## writing ... [9 set(s)] done [0.00s].
## Creating S4 object ... done [0.00s].
```

```
plot(itemsets, method="graph")
```

Graph for 9 itemsets

size: support (0.04 - 0.075)

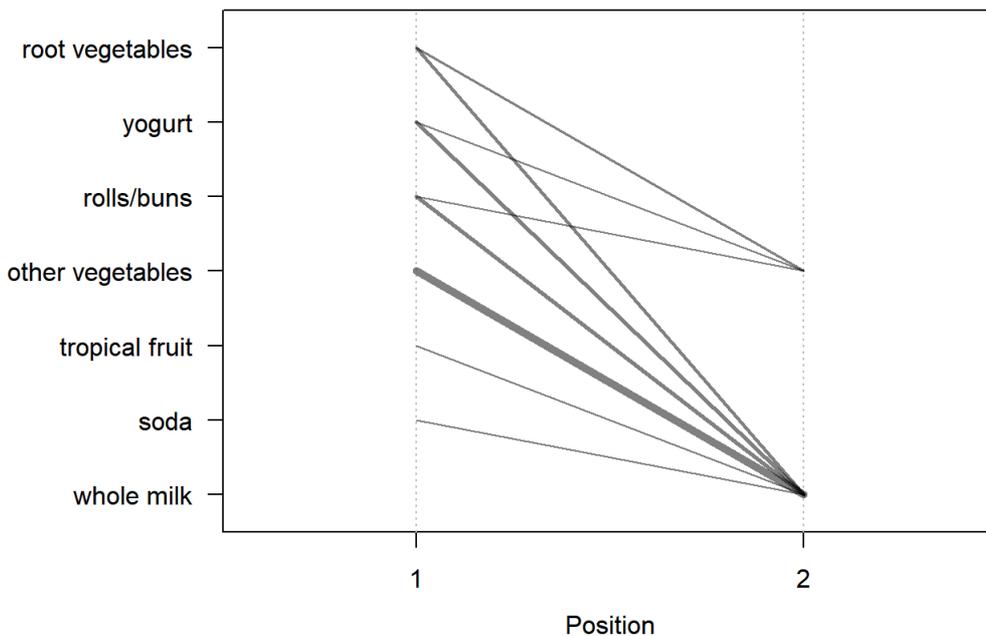


The eclat() takes in a transactions object and gives the most frequent items in the data based the support argument. The maxlen defines the maximum number of items in each itemset of frequent items. This plot used to represent the most frequent items in the dataset as graph with support value as 0.03,0.04 respectively and length as 2

12. Itemset visualization using parallel coordinates

```
plot(itemsets, method="paracoord", control=list(alpha=.5, reorder=TRUE))
```

Parallel coordinates plot for 9 itemsets



It represents the itemset as a parallel coordinate plot.

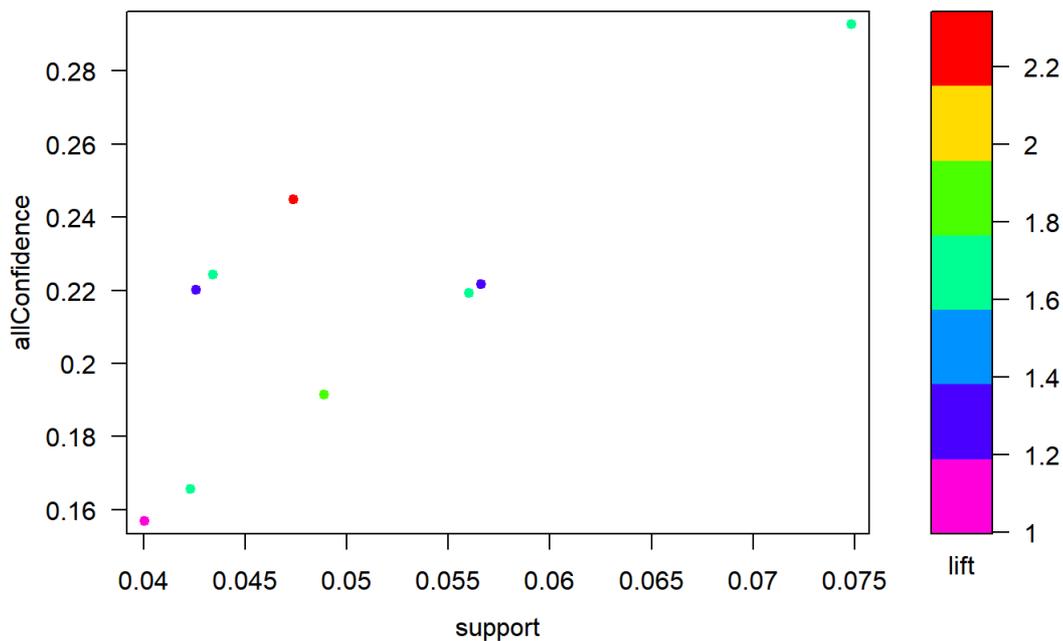
13. Add more quality measures to the scatterplot

```
quality(itemsets) <- interestMeasure(itemsets, trans=Groceries)
head(quality(itemsets))
```

```
##      support allConfidence crossSupportRatio    lift
## 1 0.04229792    0.1655392      0.4106645 1.5775950
## 2 0.04890696    0.1914047      0.4265818 1.7560310
## 3 0.04738180    0.2448765      0.5633211 2.2466049
## 4 0.04006101    0.1567847      0.6824513 0.8991124
## 5 0.05602440    0.2192598      0.5459610 1.5717351
## 6 0.04341637    0.2243826      0.7209669 1.6084566
```

```
plot(itemsets, measure=c("support", "allConfidence"), shading="lift", control = list(col=rainbow(7)))
```

Scatter plot for 9 itemsets



It represents the itemsets with various support and confidence values and with rainbow color using the lift parameter

14. Visualization of rules with Left hand side value as whole milk

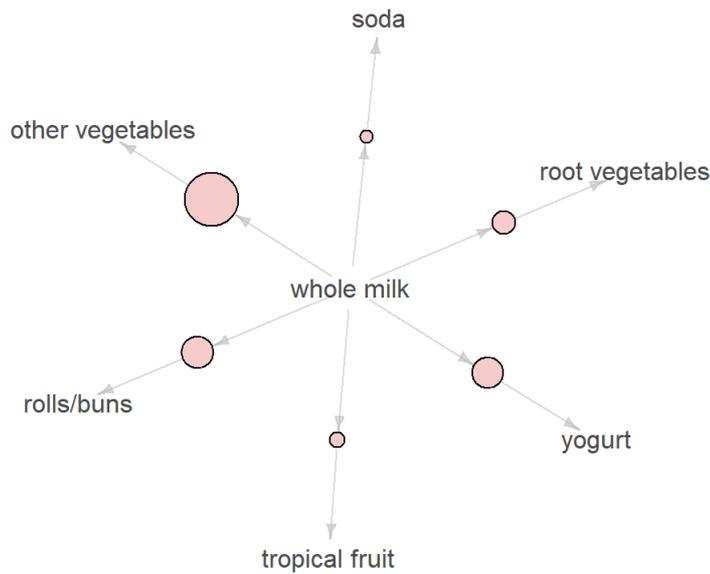
```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
  appearance = list(default="rhs",lhs="whole milk"),
  control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##      lhs      rhs      support  confidence lift    count
## [1] {whole milk} => {other vegetables} 0.07483477 0.2928770 1.513634 736
## [2] {whole milk} => {rolls/buns}      0.05663447 0.2216474 1.205032 557
## [3] {whole milk} => {yogurt}          0.05602440 0.2192598 1.571735 551
## [4] {whole milk} => {root vegetables} 0.04890696 0.1914047 1.756031 481
## [5] {whole milk} => {tropical fruit}  0.04229792 0.1655392 1.577595 416
```

```
library(arulesViz)
plot(rules,method="graph",engine = 'default',shading=NA)
```

Graph for 6 rules

size: support (0.04 - 0.075)



It is used find out the Customers who bought 'Whole Milk' also bought other items

15. Visualization of rules with Right Hand Side value as Butter

```
rules<-apriori(data=Groceries, parameter=list(supp=0.005,conf = 0.08),
  appearance = list(default="lhs",rhs="butter"),
  control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##      lhs                                rhs      support   confidence
## [1] {whole milk,whipped/sour cream} => {butter} 0.006710727 0.2082019
## [2] {other vegetables,whipped/sour cream} => {butter} 0.005795628 0.2007042
## [3] {whole milk,domestic eggs} => {butter} 0.005998983 0.2000000
## [4] {root vegetables,whole milk} => {butter} 0.008235892 0.1683992
## [5] {whole milk,yogurt} => {butter} 0.009354347 0.1669691
##      lift      count
## [1] 3.757185 66
## [2] 3.621883 57
## [3] 3.609174 59
## [4] 3.038910 81
## [5] 3.013104 92
```

```
library(arulesViz)
plot(rules,method="graph",engine = 'default',shading=NA)
```

