Association Rules

Problem statement

Prepare rules for the all the data sets

1) Try different values of support and confidence. Observe the change in number of rules for different support, confidence values

2) Change the minimum length in apriori algorithm

3) Visualize the obtained rules using different plots

Answer:

Rcode:

```
#Different set of rule values for Groceries Dataset using apriori algorithm.


groceries_data <- read.transactions(file.choose(),format = "basket")

View(groceries_data)

attach(groceries_data)

summary(groceries_data)

str(groceries_data)

class(groceries_data)


install.packages("arules")

install.packages("arulesViz")

library(arules)

library(arulesViz)


#to see most frequent items

FrequentItem <- eclat(groceries_data,parameter = list(support=0.07,maxlen=15))

inspect(FrequentItem)
```

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


rules <- apriori(groceries_data,parameter =
list(support=0.002,confidence=0.5,minlen=2,maxlen=5))

rules

inspect(head(sort(rules,by="lift"),n=15))

inspect(tail(sort(rules,by="lift"),n=15))

plot(rules)

quality(head(rules))

plot(rules, method = "grouped",control = list(cex=0.90))

plot(rules,method = "scatterplot",control = list(cex=0.90))

plot(rules,method = "graph",control = list(cex=0.90))


rules1 <- apriori(groceries_data,parameter =
list(support=0.001,confidence=0.5,minlen=3,maxlen=5))

rules1

rules_conf <- sort (rules1, by="confidence", decreasing=T)

inspect(head(rules_conf))

rules_lift <- sort(rules1,by="lift",decreasing = T)

inspect(head(rules_lift))

plot(rules1, method = "grouped",control = list(cex=0.90))

plot(rules1,method = "scatterplot",control = list(cex=0.90))

plot(rules1,method = "graph",control = list(cex=0.90))


#remove redundant rules

subsetRules <- which(colSums(is.subset(rules1, rules1)) > 1)

length(subsetRules)
```

rules1 <- rules1[-subsetRules]

rules1

subsetRules1 <- which(colSums(is.subset(rules, rules)) > 1)

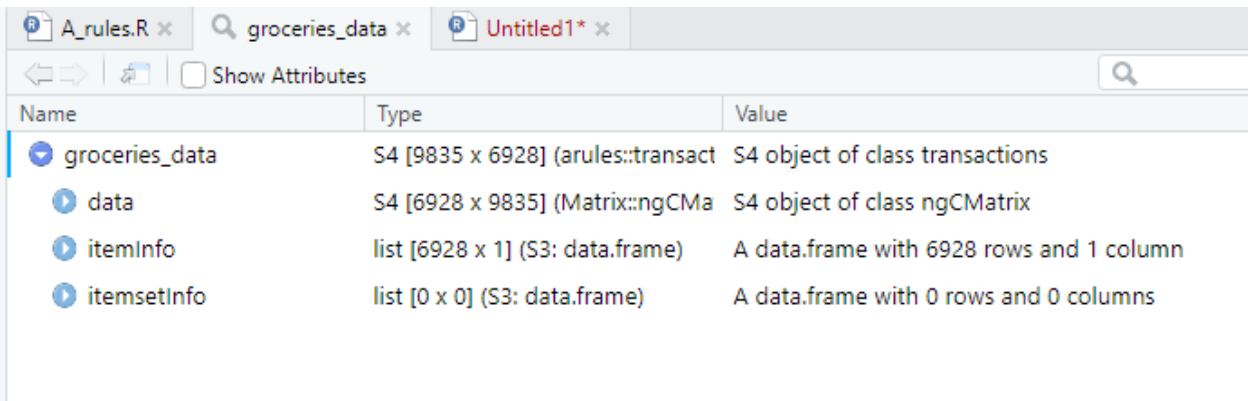
length(subsetRules1)

rules <- rules[-subsetRules1]

rules

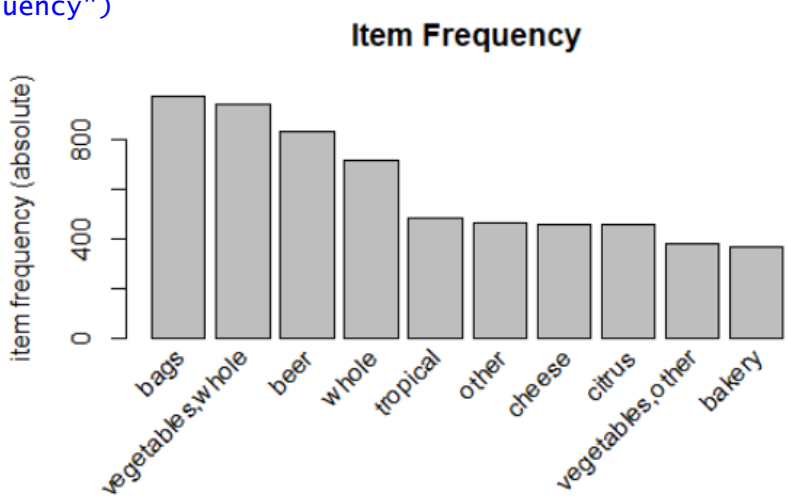
Console:

```
> groceries_data <- read.transactions(file.choose(),format = "basket")
> View(groceries_data)
```



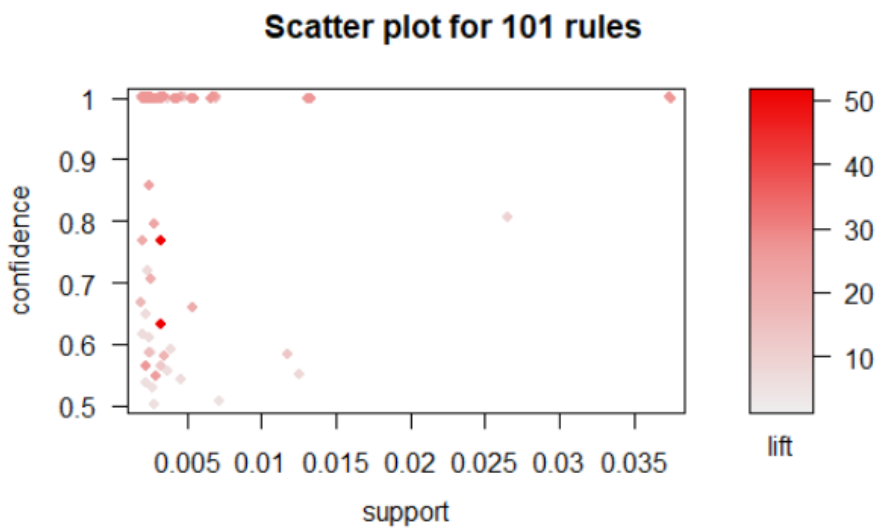

```
> summary(groceries_data)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 6928 columns (items) and a density of 0.0005240481

most frequent items:
            bags vegetables,whole             beer            whole
             971              940              829              717
        tropical          (Other)
             482            31768

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
16
1380 2743 1782 1246  907  599  413  294  166   94   76   43   39   20   10
9
  17   18   19   20   21   23
   2    3    3    3    1    2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.631   5.000  23.000

includes extended item information - examples:
                labels
1          (appetizer)
2 (appetizer),bathroom
3     (appetizer),cake
> str(groceries_data)
Formal class 'transactions' [package "arules"] with 3 slots
```

```
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:35707] 738 1531 3166 5794 3228 5913 3963 6757 96
1125 ...
  .. .. ..@ p       : int [1:9836] 0 4 6 8 13 20 23 24 28 30 ...
  .. .. ..@ Dim     : int [1:2] 6928 9835
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame':       6928 obs. of  1 variable:
  .. ..$ labels: chr [1:6928] "(appetizer)" "(appetizer),bathroom" "(appetize
r),cake" "(appetizer),candy,cling" ...
  ..@ itemsetInfo:'data.frame':       0 obs. of  0 variables
> class(groceries_data)
[1] "transactions"
attr(,"package")
[1] "arules"
> install.packages("arules")
> install.packages("arulesViz")
> library(arules)
> library(arulesViz)
> #to see most frequent items
> FrequentItem <- eclat(groceries_data,parameter = list(support=0.07,maxlen=1
5))
Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.07      1     15 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 688

create itemset ...
set transactions ...[6928 item(s), 9835 transaction(s)] done [0.06s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating sparse bit matrix ... [4 row(s), 9835 column(s)] done [0.00s].
writing  ... [4 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].
> inspect(FrequentItem)
    items              support    count
[1] {vegetables,whole} 0.09557702 940
[2] {bags}             0.09872903 971
[3] {beer}             0.08429080 829
[4] {whole}            0.07290290 717
> itemFrequencyPlot(groceries_data, topN=10, type="absolute", main="Item Freq
uency")
```

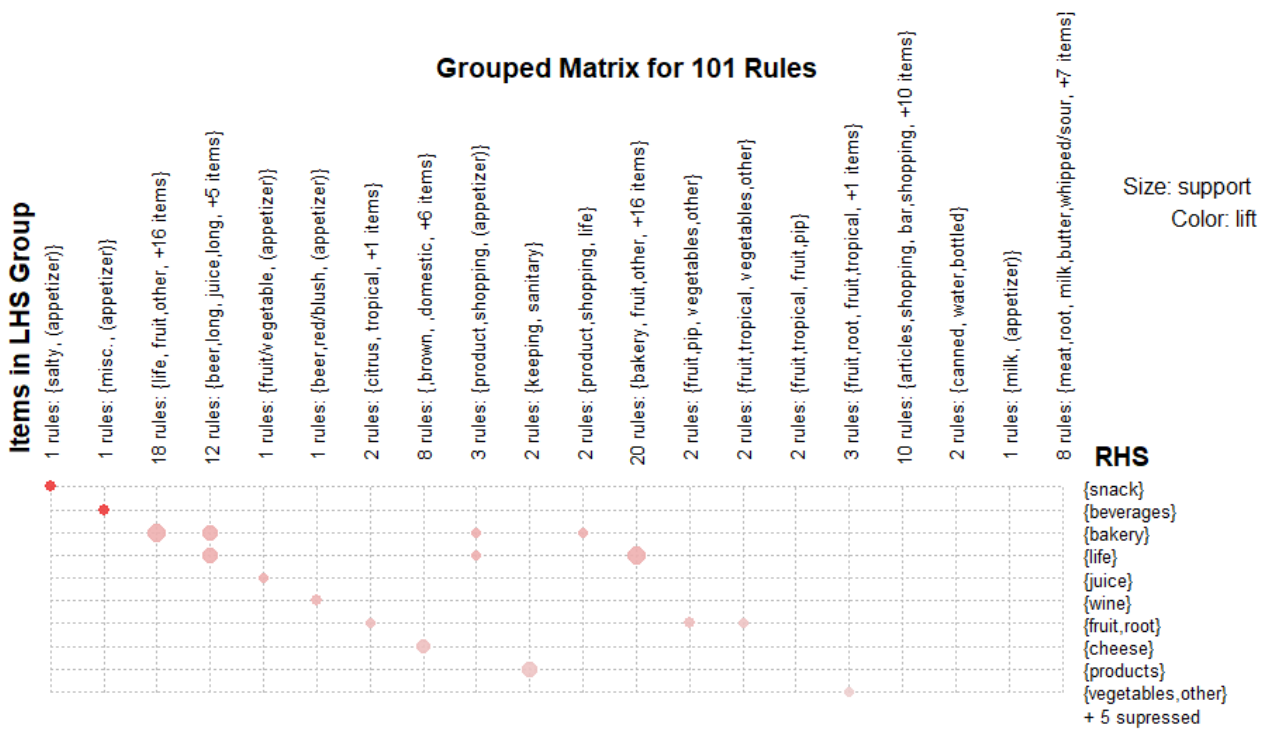


```
> rules <- apriori(groceries_data,parameter = list(support=0.002,confidence=0
.5,minlen=2,maxlen=5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen max
len
        0.5    0.1    1 none FALSE            TRUE       5   0.002      2
5
 target   ext
  rules FALSE

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

Absolute minimum support count: 19

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6928 item(s), 9835 transaction(s)] done [0.05s].
sorting and recoding items ... [257 item(s)] done [0.01s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [101 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules
set of 101 rules
> inspect(head(sort(rules,by="lift"),n=15))
     lhs                    rhs         support     confidence lift     count
[1]  {salty}                => {snack}     0.003050330 0.7692308  51.46520  30
[2]  {misc.}                => {beverages} 0.003152008 0.6326531  51.42267  31
[3]  {product,shopping}     => {bakery}    0.002338587 1.0000000  26.72554  23
[4]  {product,shopping}     => {life}      0.002338587 1.0000000  26.72554  23
[5]  {beer,long}            => {bakery}    0.002440264 1.0000000  26.72554  24
[6]  {beer,long}            => {life}      0.002440264 1.0000000  26.72554  24
[7]  {product,specialty}    => {bakery}    0.002541942 1.0000000  26.72554  25
[8]  {product,specialty}    => {life}      0.002541942 1.0000000  26.72554  25
[9]  {snack,long}           => {bakery}    0.003355363 1.0000000  26.72554  33
[10] {snack,long}           => {life}      0.003355363 1.0000000  26.72554  33
[11] {juice,long}           => {bakery}    0.004270463 1.0000000  26.72554  42
[12] {juice,long}           => {life}      0.004270463 1.0000000  26.72554  42
[13] {product}              => {bakery}    0.013218099 1.0000000  26.72554 130
[14] {product}              => {life}      0.013218099 1.0000000  26.72554 130
[15] {bakery}               => {life}      0.037417387 1.0000000  26.72554 368
> inspect(tail(sort(rules,by="lift"),n=15))
     lhs                                  rhs                 support
[1]  {bakery,life,product,shopping}    => {bags}              0.002338587
[2]  {canned}                          => {beer}              0.026537875
[3]  {milk}                            => {whole}             0.012709710
[4]  {milk,butter,whipped/sour}        => {vegetables,whole} 0.002338587
[5]  {fruit,tropical,vegetables,other} => {vegetables,whole} 0.002236909
[6]  {fruit,pip,vegetables,other}      => {vegetables,whole} 0.002135231
[7]  {water,bottled}                   => {beer}              0.004677173
[8]  {fruit,root,fruit,tropical}       => {vegetables,whole} 0.002541942
[9]  {milk,yogurt,whipped/sour}        => {vegetables,whole} 0.003965430
[10] {meat,root}                       => {vegetables,whole} 0.002440264
[11] {fruit,pip,fruit,root}            => {vegetables,whole} 0.003558719
[12] {cheese,vegetables,other}         => {vegetables,whole} 0.002236909
[13] {vegetables,onions,other}         => {vegetables,whole} 0.002745297
[14] {fruit,root,vegetables,other}     => {vegetables,whole} 0.007320793
[15] {fruit,root,tropical}             => {vegetables,whole} 0.002745297
     confidence lift      count
[1]  1.0000000  10.128733  23
[2]  0.8080495   9.586450 261
[3]  0.5530973   7.586768 125
[4]  0.7187500   7.520113  23
[5]  0.6470588   6.770025  22
[6]  0.6176471   6.462297  21
```

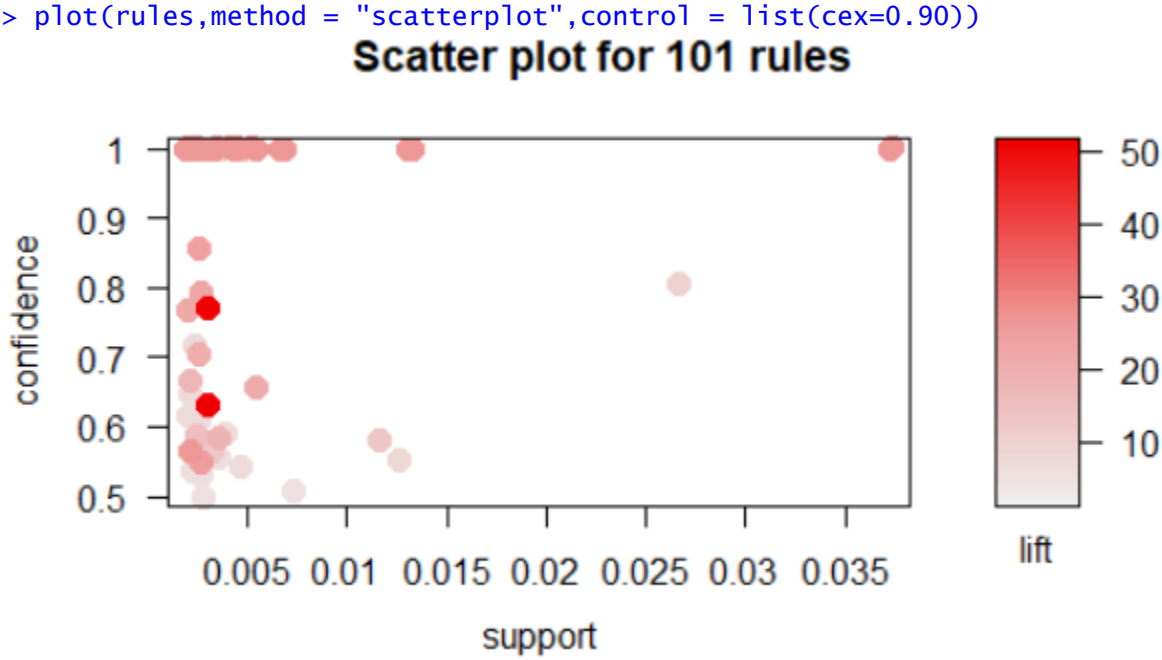
```
[7]  0.5411765   6.420351  46
[8]  0.6097561   6.379735  25
[9]  0.5909091   6.182544  39
[10] 0.5853659   6.124546  24
[11] 0.5555556   5.812648  35
[12] 0.5365854   5.614167  22
[13] 0.5294118   5.539111  27
[14] 0.5070423   5.305064  72
[15] 0.5000000   5.231383  27
> plot(rules)
```



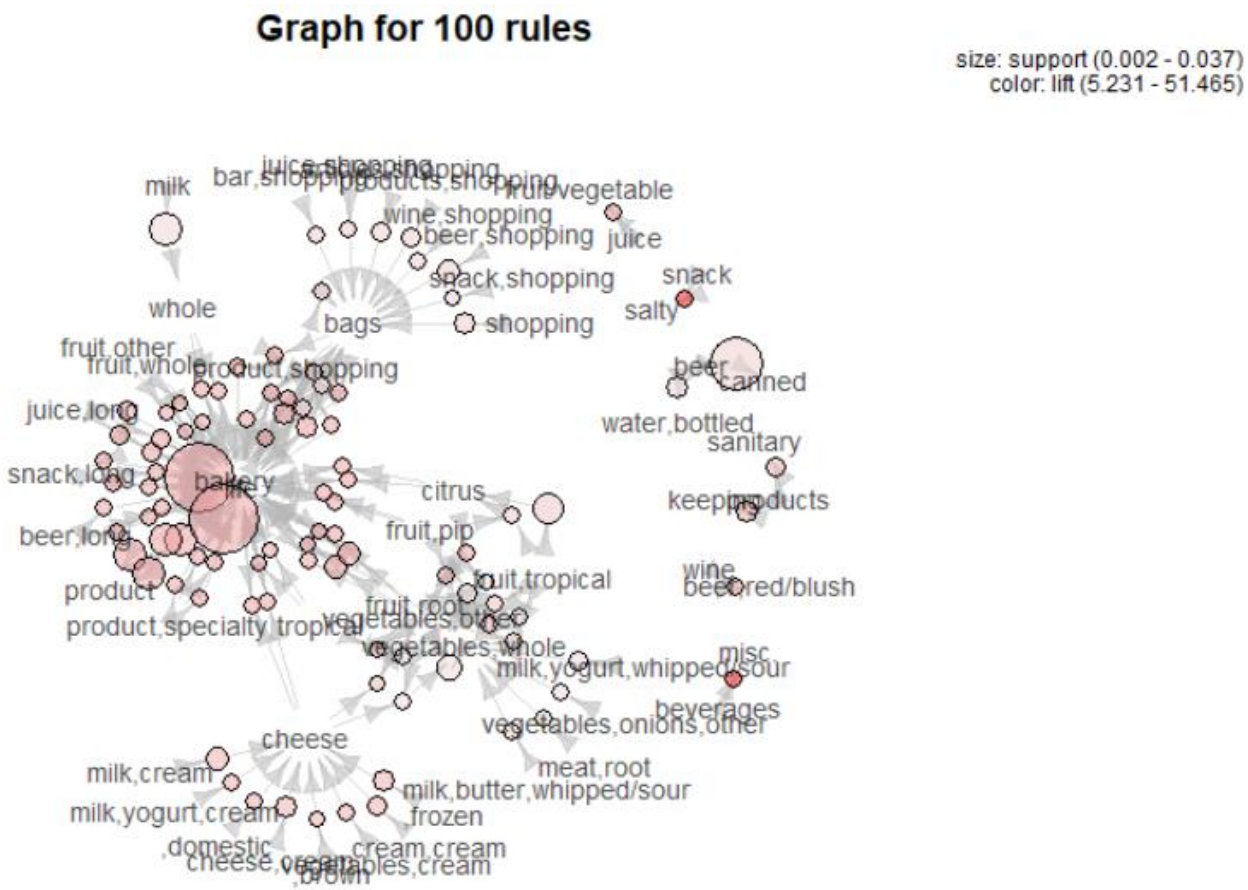

```
> quality(head(rules))
      support confidence     lift count
1 0.003050330  0.7692308 51.46520    30
2 0.004880529  1.0000000 10.12873    48
3 0.002236909  0.5641026 26.54521    22
4 0.002033554  1.0000000 10.12873    20
5 0.003152008  0.6326531 51.42267    31
6 0.003050330  1.0000000 10.12873    30
> plot(rules, method = "grouped",control = list(cex=0.90))
```



```
Available control parameters (with default values):
main     =  Grouped Matrix for 101 Rules
k        =  20
rhs_max  =  10
lhs_items        =  2
aggr.fun         =  function (x, ...)  UseMethod("mean")
col      =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF",
"#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF",
"#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF",
"#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF",
"#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF",
"#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF",
"#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF"
, "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF
", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DF
F", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7
FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B
1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBC
BCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC
6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#E
ED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#
EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "
#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
reverse  =  TRUE
xlab     =  NULL
ylab     =  NULL
legend   =  Size: support  Color: lift
spacing  =  -1
panel.function  =  function (row, size, shading, spacing)  {     size[size ==
0] <- NA     shading[is.na(shading)] <- 1     grid.circle(x = c(1:length(size
)), y = row, r = size/2 * (1 - spacing), default.units = "native", gp = gpar(
fill = shading, col = shading, alpha = 0.9)) }
gp_main  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_labels        =  list(cex = 0.8)
gp_labs  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_lines         =  list(col = "gray", lty = 3)
newpage  =  TRUE
max.shading      =  NA
engine   =  default
verbose  =  FALSE
> plot(rules,method = "scatterplot",control = list(cex=0.90))
```

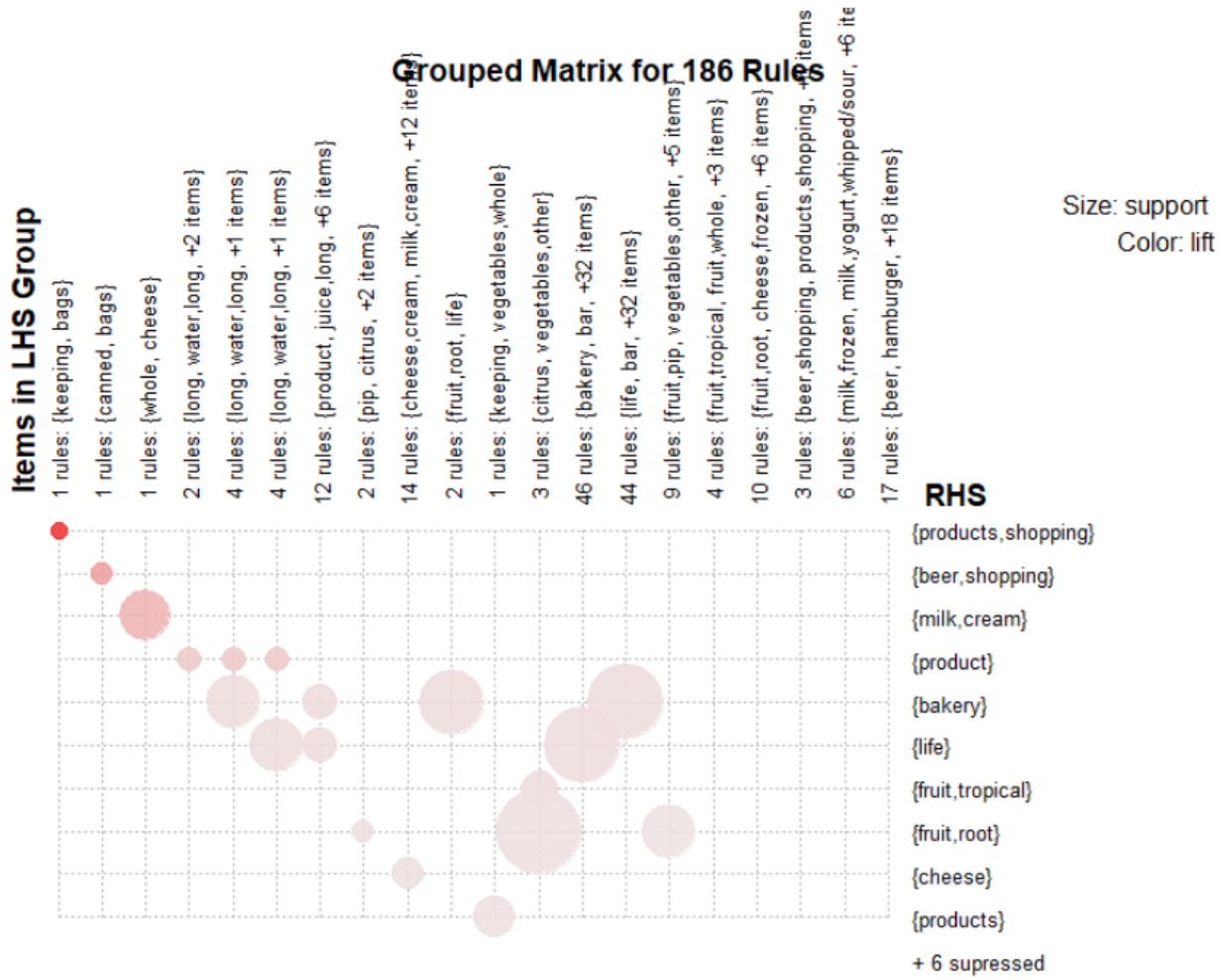



```
> plot(rules,method = "graph",control = list(cex=0.90))
```



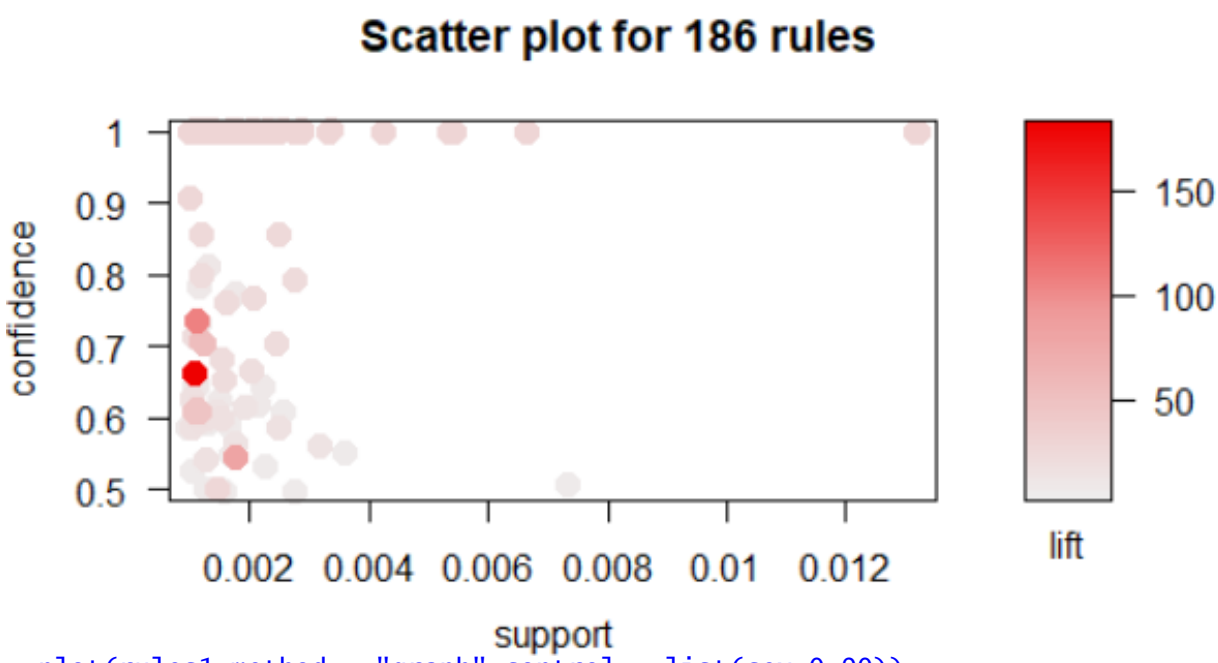

```
> rules1 <- apriori(groceries_data,parameter = list(support=0.001,confidence=
0.5,minlen=3,maxlen=5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.5    0.1    1 none FALSE            TRUE       5   0.001      3
 maxlen target   ext
      5  rules 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 ...[6928 item(s), 9835 transaction(s)] done [0.05s].
sorting and recoding items ... [483 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [186 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> rules1
set of 186 rules
> rules_conf <- sort (rules1, by="confidence", decreasing=T)
> inspect(head(rules_conf))
    lhs                            rhs      support     confidence lift
[1] {long,product}              => {bakery} 0.001118454 1          26.72554
[2] {long,product}              => {life}   0.001118454 1          26.72554
[3] {bakery,long}               => {life}   0.001830198 1          26.72554
[4] {life,long}                 => {bakery} 0.001830198 1          26.72554
[5] {bakery,product,newspapers} => {life}   0.001220132 1          26.72554
```

```
[6] {life,product,newspapers}   => {bakery} 0.001220132 1          26.72554
    count
[1] 11
[2] 11
[3] 18
[4] 18
[5] 12
[6] 12
> rules_lift <- sort(rules1,by="lift",decreasing = T)
> inspect(head(rules_lift))
    lhs                          rhs                 support     confidence
[1] {bags,keeping}           => {products,shopping} 0.001016777 0.6666667
[2] {bags,canned}            => {beer,shopping}     0.001118454 0.7333333
[3] {cheese,whole}           => {milk,cream}        0.001728521 0.5483871
[4] {bakery,water,long}      => {product}           0.001220132 0.7058824
[5] {life,water,long}        => {product}           0.001220132 0.7058824
[6] {bakery,life,water,long} => {product}           0.001220132 0.7058824
    lift      count
[1] 182.12963 10
[2] 106.06373 11
[3]  79.31452 17
[4]  53.40271 12
[5]  53.40271 12
[6]  53.40271 12
> plot(rules1, method = "grouped",control = list(cex=0.90))
```

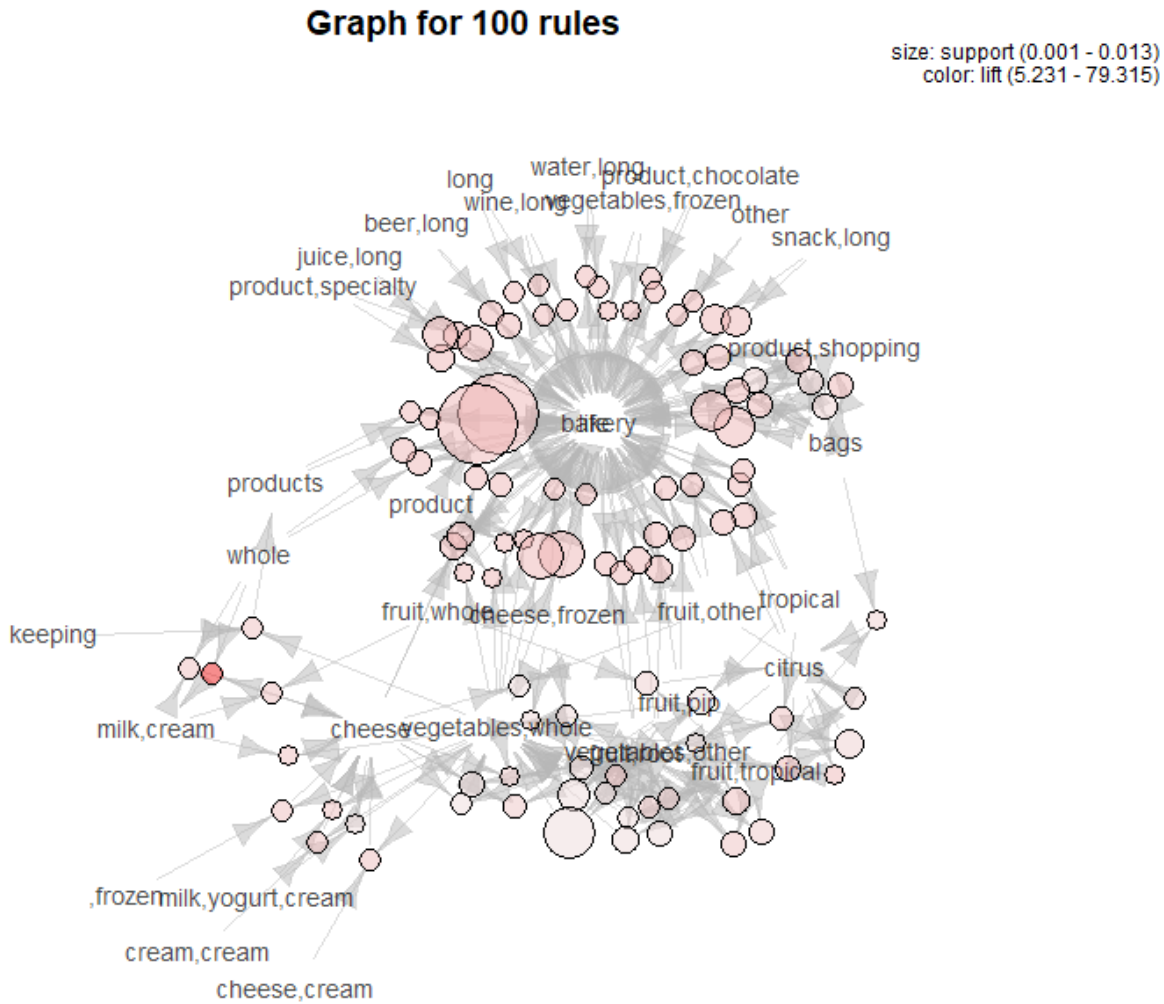



```
Available control parameters (with default values):
main     =  Grouped Matrix for 186 Rules
k        =  20
rhs_max  =  10
lhs_items       =  2
aggr.fun        =  function (x, ...)  UseMethod("mean")
```

```
col      =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF",
"#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF",
"#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF",
"#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF",
"#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF",
"#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF",
"#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF"
, "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF
", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DF
F", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7
FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B
1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBC
BCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC
6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#E
ED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#
EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "
#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
reverse  =  TRUE
xlab     =  NULL
ylab     =  NULL
legend   =  Size: support  Color: lift
spacing  =  -1
panel.function  =  function (row, size, shading, spacing)  {     size[size ==
0] <- NA     shading[is.na(shading)] <- 1     grid.circle(x = c(1:length(size
)), y = row, r = size/2 * (1 - spacing), default.units = "native", gp = gpar(
fill = shading, col = shading, alpha = 0.9)) }
gp_main  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_labels        =  list(cex = 0.8)
gp_labs  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_lines         =  list(col = "gray", lty = 3)
newpage  =  TRUE
max.shading      =  NA
engine   =  default
verbose  =  FALSE
> plot(rules1,method = "scatterplot",control = list(cex=0.90))
```




```
> plot(rules1,method = "graph",control = list(cex=0.90))
```




```
> #remove redundant rules
> subsetRules <- which(colSums(is.subset(rules1, rules1)) > 1)
> length(subsetRules)
[1] 128
> rules1 <- rules1[-subsetRules]
> rules1
set of 58 rules
>
> subsetRules1 <- which(colSums(is.subset(rules, rules)) > 1)
> length(subsetRules1)
[1] 46
> rules <- rules[-subsetRules1]
> rules
set of 55 rules
```

Rcode:

#Different set of rule values for my_movies Dataset using apriori algorithm.

mymovies <- read.transactions(file.choose(),format = "basket")

View(mymovies)

summary(mymovies)

```
str(mymovies)
class(mymovies)

#to see most frequent items
FrequentItem <- eclat(mymovies,parameter = list(support=0.07,maxlen=15))
inspect(FrequentItem)
itemFrequencyPlot(mymovies, topN=5, type="absolute", main="Item Frequency")
dev.off()
rules3 <- apriori(mymovies,parameter =
list(support=0.002,confidence=0.5,minlen=2,maxlen=5))
rules3
inspect(head(sort(rules3,by="lift"),n=15))
inspect(tail(sort(rules3,by="lift"),n=15))
plot(rules3)
quality(head(rules3))
plot(rules3, method = "grouped",control = list(cex=0.90))
plot(rules3,method = "scatterplot",control = list(cex=0.90))
plot(rules3,method = "graph",control = list(cex=0.90))

mymovies1 <- read.csv(file.choose())
names(mymovies1)
summary(mymovies1)
attach(mymovies1)
sd(Sixth.Sense)
sd(Gladiator)
sd(LOTR1)
sd(Harry.Potter1)
```

```
sd(Patriot)
sd(LOTR2)
sd(Harry.Potter2)
sd(LOTR)
sd(Braveheart)
sd(Green.Mile)
var(Sixth.Sense)
var(Gladiator)
var(LOTR1)
var(Harry.Potter1)
var(Patriot)
var(LOTR2)
var(Harry.Potter2)
var(LOTR)
var(Braveheart)
var(Green.Mile)
library(moments)
skewness(Sixth.Sense)
skewness(Gladiator)
skewness(LOTR1)
skewness(Harry.Potter1)
skewness(Patriot)
skewness(LOTR2)
skewness(Harry.Potter2)
skewness(LOTR)
skewness(Braveheart)
skewness(Green.Mile)
```

```
kurtosis(Sixth.Sense)
kurtosis(Gladiator)
kurtosis(LOTR1)
kurtosis(Harry.Potter1)
kurtosis(Patriot)
kurtosis(LOTR2)
kurtosis(Harry.Potter2)
kurtosis(LOTR)
kurtosis(Braveheart)
kurtosis(Green.Mile)
rules2 <- apriori(as.matrix(mymovies1[,6:15],parameter=list(support=0.2,
confidence = 0.5,minlen=5)))
rules2
rules_conf <- sort (rules2, by="confidence", decreasing=T)
inspect(head(rules_conf))
rules_lift <- sort(rules2,by="lift",decreasing = T)
inspect(head(rules_lift))
plot(rules2, method = "grouped",control = list(cex=0.90))
plot(rules2,method = "scatterplot",control = list(cex=0.90))
plot(rules2,method = "graph",control = list(cex=0.90))

#remove redundant rules
subsetRules <- which(colSums(is.subset(rules2, rules2)) > 1)
length(subsetRules)
rules2 <- rules2[-subsetRules]
rules2
```

subsetRules1 <- which(colSums(is.subset(rules3, rules)) > 1)

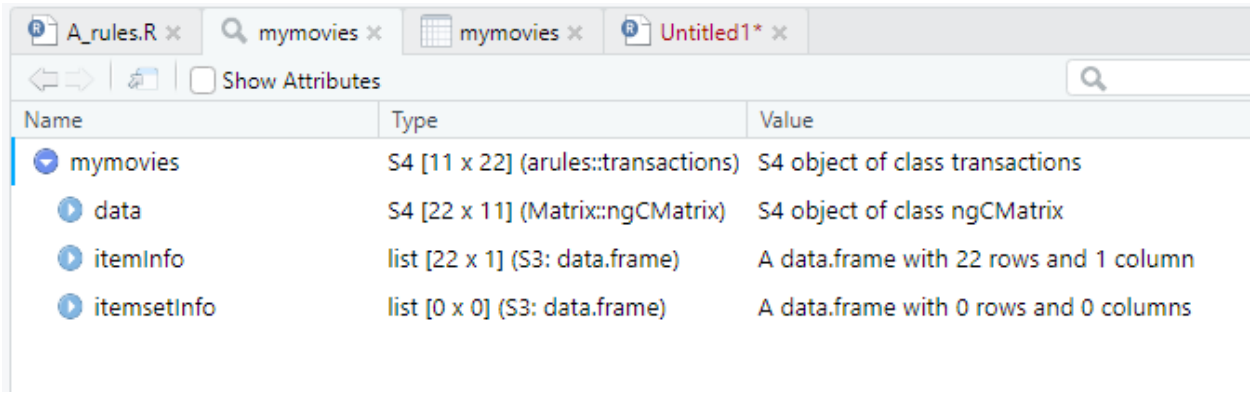
length(subsetRules1)

rules3 <- rules3[-subsetRules1]

rules3

Console:

```
> #Different set of rule values for my_movies Dataset using apriori algorithm
.
> mymovies <- read.transactions(file.choose(),format = "basket")
> View(mymovies)
```




```
> summary(mymovies)
transactions as itemMatrix in sparse format with
 11 rows (elements/itemsets/transactions) and
 22 columns (items) and a density of 0.1404959

most frequent items:
                       Gladiator                ,"Patriot","Sixth
                               6                                 4
Sense","","",1,1,0,0,1,0,0,0,0,0                      Sixth Sense
                               4                                 2
                         ,"Harry                          (Other)
                               1                                17

element (itemset/transaction) length distribution:
sizes
2 3 4 6
3 6 1 1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.500   3.000   3.091   3.000   6.000

includes extended item information - examples:
                      labels
1                    ,"Harry
2 ,"LOTR","Gladiator","Green
3            ,"LOTR1","Harry
> str(mymovies)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:34] 7 11 15 17 19 21 2 13 14 20 ...
  .. .. ..@ p       : int [1:12] 0 6 10 12 14 17 20 23 26 28 ...
  .. .. ..@ Dim     : int [1:2] 22 11
```

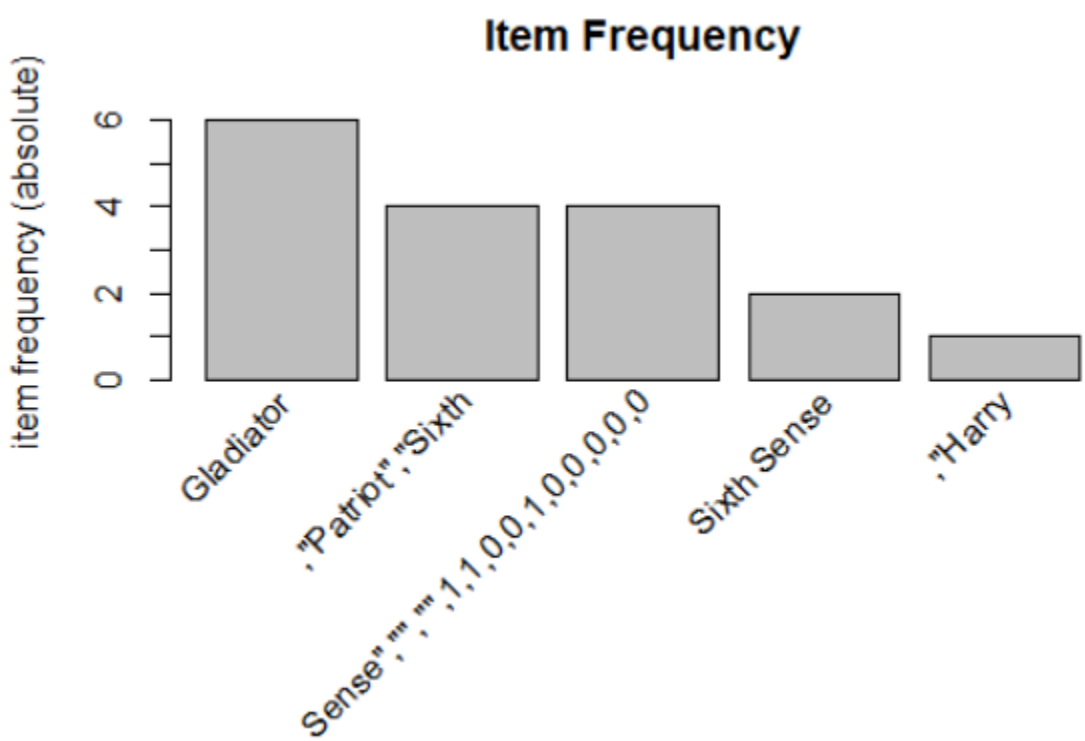
```
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame':       22 obs. of  1 variable:
  .. ..$ labels: chr [1:22] ",\"Harry" ",\"LOTR\",\"Gladiator\",\"Green" ",\"
LOTR1\",\"Harry" ",\"LOTR2\",\"\",\"\",\"\",0,0,1,0,0,1,0,0,0,0" ...
  ..@ itemsetInfo:'data.frame':       0 obs. of  0 variables
> class(mymovies)
[1] "transactions"
attr(,"package")
[1] "arules"
> #to see most frequent items
> FrequentItem <- eclat(mymovies,parameter = list(support=0.07,maxlen=15))
Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.07      1     15 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 0

create itemset ...
set transactions ...[22 item(s), 11 transaction(s)] done [0.00s].
sorting and recoding items ... [22 item(s)] done [0.00s].
creating sparse bit matrix ... [22 row(s), 11 column(s)] done [0.00s].
writing  ... [105 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].
> inspect(FrequentItem)
      items                                                  support count
[1]   {,"Patriot","","","",0,1,0,0,1,0,0,0,0,0,
       Gladiator}                                          0.09090909     1
[2]   {,"LOTR2","","","",0,0,1,0,0,1,0,0,0,0,
       LOTR1}                                              0.09090909     1
[3]   {,"Patriot","Braveheart","","",0,1,0,0,1,0,0,0,1,0,
       Gladiator}                                          0.09090909     1
[4]   {,"LOTR","Gladiator","Green,
       Mile","",1,1,0,0,0,0,0,1,0,1,
       Sixth Sense}                                        0.09090909     1
[5]   {Mile","",1,1,0,0,0,0,0,1,0,1,
       Sixth Sense}                                        0.09090909     1
[6]   {,"LOTR","Gladiator","Green,
       Mile","",1,1,0,0,0,0,0,1,0,1}                       0.09090909     1
[7]   {,"LOTR","Gladiator","Green,
       Sixth Sense}                                        0.09090909     1
[8]   {,"Harry,
       Harry Potter1,
       Potter2","","","",0,0,0,1,0,0,1,0,0,0}              0.09090909     1
[9]   {,"Harry,
       Potter2","","","",0,0,0,1,0,0,1,0,0,0}              0.09090909     1
[10]  {Harry Potter1,
       Potter2","","","",0,0,0,1,0,0,1,0,0,0}              0.09090909     1
[11]  {,"Harry,
       Harry Potter1}                                      0.09090909     1
[12]  {,"LOTR1","Harry,
       Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Potter1","Green,
       Sixth Sense}                                        0.09090909     1
[13]  {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Potter1","Green,
       Sixth Sense}                                        0.09090909     1
[14]  {,"LOTR1","Harry,
       Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Potter1","Green}                                    0.09090909     1
```

```
[15]  {,"LOTR1","Harry,
       Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Sixth Sense}                                    0.09090909     1
[16]  {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Sixth Sense}                                    0.09090909     1
[17]  {,"LOTR1","Harry,
       Mile","LOTR2",1,0,1,1,0,1,0,0,0,1}              0.09090909     1
[18]  {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
       Potter1","Green}                                0.09090909     1
[19]  {,"LOTR1","Harry,
       Potter1","Green,
       Sixth Sense}                                    0.09090909     1
[20]  {Potter1","Green,
       Sixth Sense}                                    0.09090909     1
[21]  {,"LOTR1","Harry,
       Potter1","Green}                                0.09090909     1
[22]  {,"LOTR1","Harry,
       Sixth Sense}                                    0.09090909     1
[23]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[24]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}              0.09090909     1
[25]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[26]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[27]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}              0.09090909     1
[28]  {Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[29]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}              0.09090909     1
[30]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[31]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}              0.09090909     1
[32]  {Mile",
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
[33]  {,"V2","V3","V4","V5","Sixth,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                             0.09090909     1
```

```
[34]  {,"V2","V3","V4","V5","Sixth,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}                0.09090909     1
[35]  {Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                               0.09090909     1
[36]  {Mile",
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}                0.09090909     1
[37]  {Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}                0.09090909     1
[38]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[39]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[40]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[41]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[42]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[43]  {Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[44]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[45]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[46]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[47]  {Mile",
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[48]  {,"V2","V3","V4","V5","Sixth,
       Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[49]  {,"V2","V3","V4","V5","Sixth,
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[50]  {Potter2","LOTR","Braveheart","Green,
       V1}                                               0.09090909     1
[51]  {Mile",
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[52]  {Potter1","Patriot","LOTR2","Harry,
       Potter2","LOTR","Braveheart","Green}              0.09090909     1
[53]  {Potter2","LOTR","Braveheart","Green,
       Sense","Gladiator","LOTR1","Harry}                0.09090909     1
[54]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                               0.09090909     1
[55]  {,"V2","V3","V4","V5","Sixth,
       Mile",
```

```
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[56]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[57]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[58]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[59]  {Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[60]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[61]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[62]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[63]  {Mile",
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[64]  {,"V2","V3","V4","V5","Sixth,
       Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[65]  {,"V2","V3","V4","V5","Sixth,
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[66]  {Sense","Gladiator","LOTR1","Harry,
       V1}                                                   0.09090909     1
[67]  {Mile",
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[68]  {Potter1","Patriot","LOTR2","Harry,
       Sense","Gladiator","LOTR1","Harry}                    0.09090909     1
[69]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry,
       V1}                                                   0.09090909     1
[70]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       Potter1","Patriot","LOTR2","Harry}                    0.09090909     1
[71]  {Mile",
       Potter1","Patriot","LOTR2","Harry,
       V1}                                                   0.09090909     1
[72]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry,
       V1}                                                   0.09090909     1
[73]  {,"V2","V3","V4","V5","Sixth,
       Potter1","Patriot","LOTR2","Harry}                    0.09090909     1
[74]  {Potter1","Patriot","LOTR2","Harry,
       V1}                                                   0.09090909     1
[75]  {Mile",
       Potter1","Patriot","LOTR2","Harry}                    0.09090909     1
[76]  {,"V2","V3","V4","V5","Sixth,
       Mile",
       V1}                                                   0.09090909     1
[77]  {,"V2","V3","V4","V5","Sixth,
       Mile"}                                                0.09090909     1
[78]  {Mile",
       V1}                                                   0.09090909     1
[79]  {,"V2","V3","V4","V5","Sixth,
       V1}                                                   0.09090909     1
```

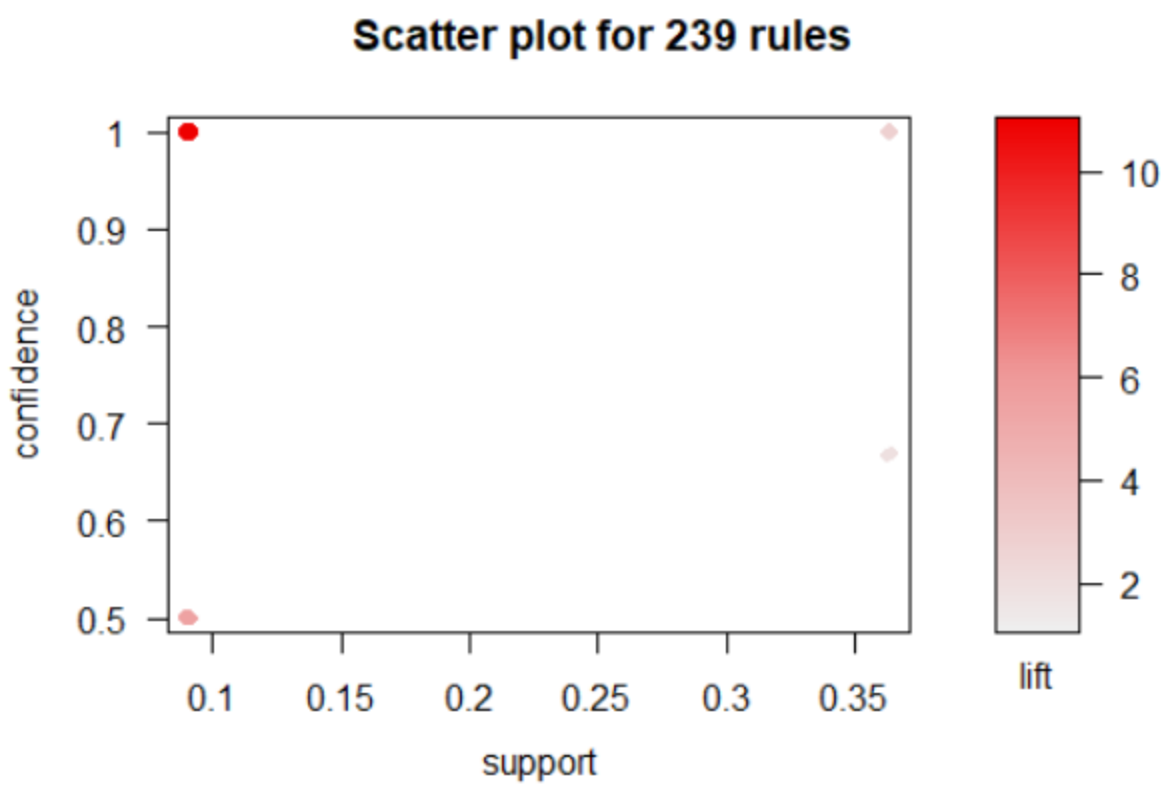
```
[80]  {,"Patriot","Sixth,
       Gladiator,
       Sense","","",1,1,0,0,1,0,0,0,0,0}                  0.36363636     4
[81]  {,"Patriot","Sixth,
       Gladiator}                                        0.36363636     4
[82]  {,"Patriot","Sixth,
       Sense","","",1,1,0,0,1,0,0,0,0,0}                  0.36363636     4
[83]  {Gladiator,
       Sense","","",1,1,0,0,1,0,0,0,0,0}                  0.36363636     4
[84]  {Gladiator}                                        0.54545455     6
[85]  {Sense","","",1,1,0,0,1,0,0,0,0,0}                  0.36363636     4
[86]  {,"Patriot","Sixth}                                0.36363636     4
[87]  {Sixth Sense}                                      0.18181818     2
[88]  {,"V2","V3","V4","V5","Sixth}                      0.09090909     1
[89]  {V1}                                               0.09090909     1
[90]  {Mile"}                                            0.09090909     1
[91]  {Potter1","Patriot","LOTR2","Harry}                0.09090909     1
[92]  {Sense","Gladiator","LOTR1","Harry}                0.09090909     1
[93]  {Potter2","LOTR","Braveheart","Green}              0.09090909     1
[94]  {,"LOTR1","Harry}                                  0.09090909     1
[95]  {Potter1","Green}                                  0.09090909     1
[96]  {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1}                0.09090909     1
[97]  {,"Harry}                                          0.09090909     1
[98]  {Harry Potter1}                                    0.09090909     1
[99]  {Potter2","","","",0,0,0,1,0,0,1,0,0,0}            0.09090909     1
[100] {,"LOTR","Gladiator","Green}                       0.09090909     1
[101] {Mile","",1,1,0,0,0,0,0,1,0,1}                     0.09090909     1
[102] {,"Patriot","Braveheart","","",0,1,0,0,1,0,0,0,1,0} 0.09090909     1
[103] {,"LOTR2","","","",0,0,1,0,0,1,0,0,0,0}            0.09090909     1
[104] {LOTR1}                                            0.09090909     1
[105] {,"Patriot","","","",0,1,0,0,1,0,0,0,0,0}          0.09090909     1
> itemFrequencyPlot(mymovies, topN=5, type="absolute", main="Item Frequency")
```



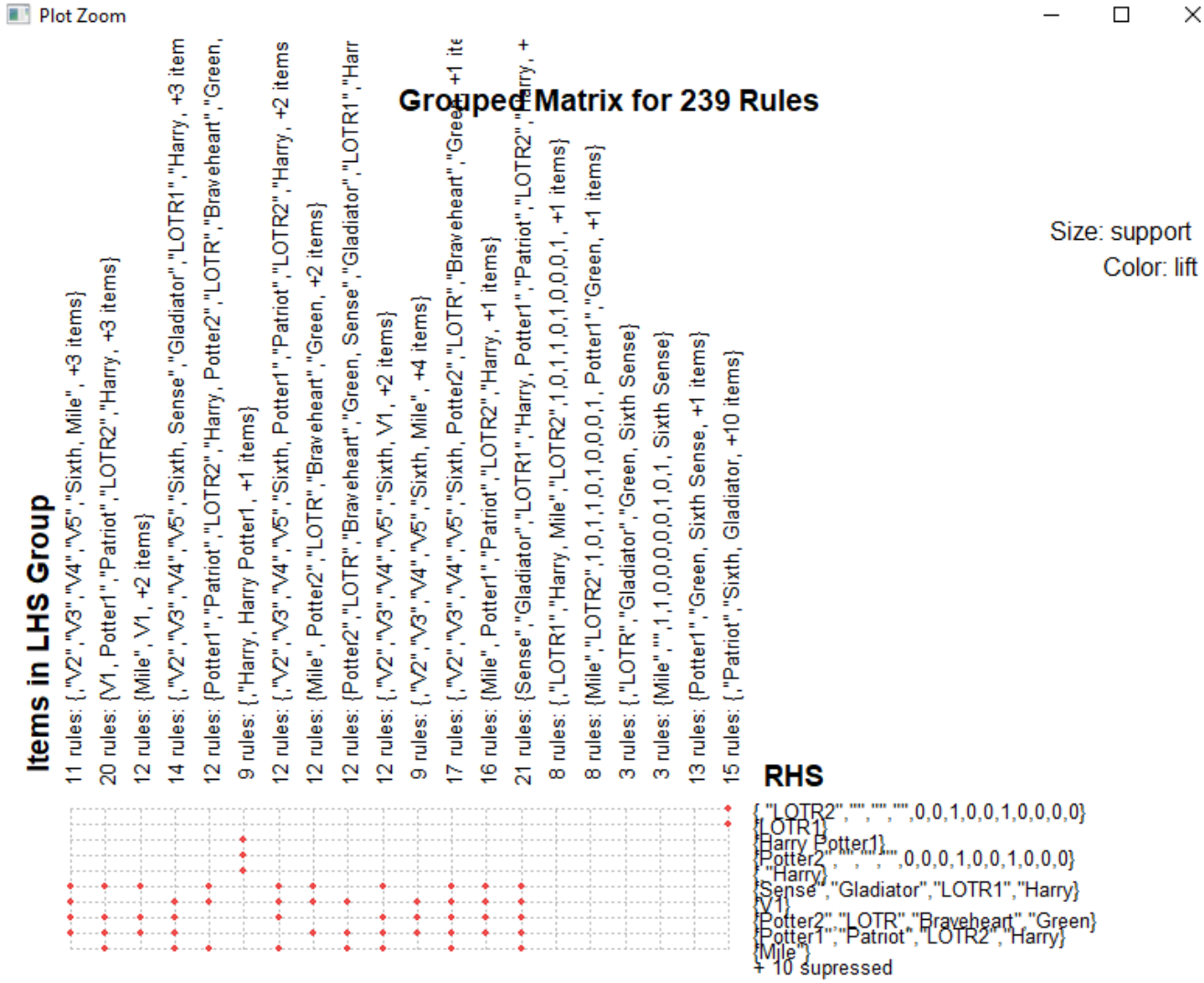

```
> rules3 <- apriori(mymovies,parameter = list(support=0.002,confidence=0.5,mi
nlen=2,maxlen=5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.5    0.1    1 none FALSE            TRUE       5   0.002      2
 maxlen target   ext
```

```
      5  rules FALSE

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

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[22 item(s), 11 transaction(s)] done [0.00s].
sorting and recoding items ... [22 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [239 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules3
set of 239 rules
> inspect(head(sort(rules3,by="lift"),n=15))
     lhs                                          rhs
support confidence lift count
[1]  {LOTR1}                                   => {,"LOTR2","","","",0,0,1,0,0,
1,0,0,0,0} 0.09090909          1   11     1
[2]  {,"LOTR2","","","",0,0,1,0,0,1,0,0,0,0} => {LOTR1}
0.09090909          1   11     1
[3]  {,"LOTR","Gladiator","Green}              => {Mile","",1,1,0,0,0,0,0,1,0,1
}          0.09090909          1   11     1
[4]  {Mile","",1,1,0,0,0,0,0,1,0,1}            => {,"LOTR","Gladiator","Green}
0.09090909          1   11     1
[5]  {Potter2","","","",0,0,0,1,0,0,1,0,0,0} => {Harry Potter1}
0.09090909          1   11     1
[6]  {Harry Potter1}                           => {Potter2","","","",0,0,0,1,0,
0,1,0,0,0} 0.09090909          1   11     1
[7]  {Potter2","","","",0,0,0,1,0,0,1,0,0,0} => {,"Harry}
0.09090909          1   11     1
[8]  {,"Harry}                                 => {Potter2","","","",0,0,0,1,0,
0,1,0,0,0} 0.09090909          1   11     1
[9]  {Harry Potter1}                           => {,"Harry}
0.09090909          1   11     1
[10] {,"Harry}                                 => {Harry Potter1}
0.09090909          1   11     1
[11] {,"LOTR1","Harry}                         => {Mile","LOTR2",1,0,1,1,0,1,0,
0,0,1}     0.09090909          1   11     1
[12] {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1}       => {,"LOTR1","Harry}
0.09090909          1   11     1
[13] {,"LOTR1","Harry}                         => {Potter1","Green}
0.09090909          1   11     1
[14] {Potter1","Green}                         => {,"LOTR1","Harry}
0.09090909          1   11     1
[15] {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1}       => {Potter1","Green}
0.09090909          1   11     1
> inspect(tail(sort(rules3,by="lift"),n=15))
     lhs                                                        rhs
support confidence     lift count
[1]  {,"LOTR1","Harry,
      Mile","LOTR2",1,0,1,1,0,1,0,0,0,1}                   => {Sixth Sense}
0.09090909  1.0000000 5.500000     1
[2]  {,"LOTR1","Harry,
      Potter1","Green}                                     => {Sixth Sense}
0.09090909  1.0000000 5.500000     1
[3]  {Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
      Potter1","Green}                                     => {Sixth Sense}
0.09090909  1.0000000 5.500000     1
[4]  {,"LOTR1","Harry,
      Mile","LOTR2",1,0,1,1,0,1,0,0,0,1,
      Potter1","Green}                                     => {Sixth Sense}
0.09090909  1.0000000 5.500000     1
[5]  {,"Patriot","Sixth}                                   => {Sense","","",1,1
,0,0,1,0,0,0,0,0} 0.36363636  1.0000000 2.750000     4
```

```
[6]  {Sense","","",1,1,0,0,1,0,0,0,0,0}                    => {,"Patriot","Sixt
h}                0.36363636  1.0000000 2.750000     4
[7]  {,"Patriot","Sixth,
      Gladiator}                                          => {Sense","","",1,1
,0,0,1,0,0,0,0,0} 0.36363636  1.0000000 2.750000     4
[8]  {Gladiator,
      Sense","","",1,1,0,0,1,0,0,0,0,0}                    => {,"Patriot","Sixt
h}                0.36363636  1.0000000 2.750000     4
[9]  {,"Patriot","","","",0,1,0,0,1,0,0,0,0,0}             => {Gladiator}
0.09090909  1.0000000 1.833333     1
[10] {,"Patriot","Braveheart","","",0,1,0,0,1,0,0,0,1,0} => {Gladiator}
0.09090909  1.0000000 1.833333     1
[11] {,"Patriot","Sixth}                                  => {Gladiator}
0.36363636  1.0000000 1.833333     4
[12] {Gladiator}                                          => {,"Patriot","Sixt
h}                0.36363636  0.6666667 1.833333     4
[13] {Sense","","",1,1,0,0,1,0,0,0,0,0}                    => {Gladiator}
0.36363636  1.0000000 1.833333     4
[14] {Gladiator}                                          => {Sense","","",1,1
,0,0,1,0,0,0,0,0} 0.36363636  0.6666667 1.833333     4
[15] {,"Patriot","Sixth,
      Sense","","",1,1,0,0,1,0,0,0,0,0}                    => {Gladiator}
0.36363636  1.0000000 1.833333     4
> plot(rules3)
```

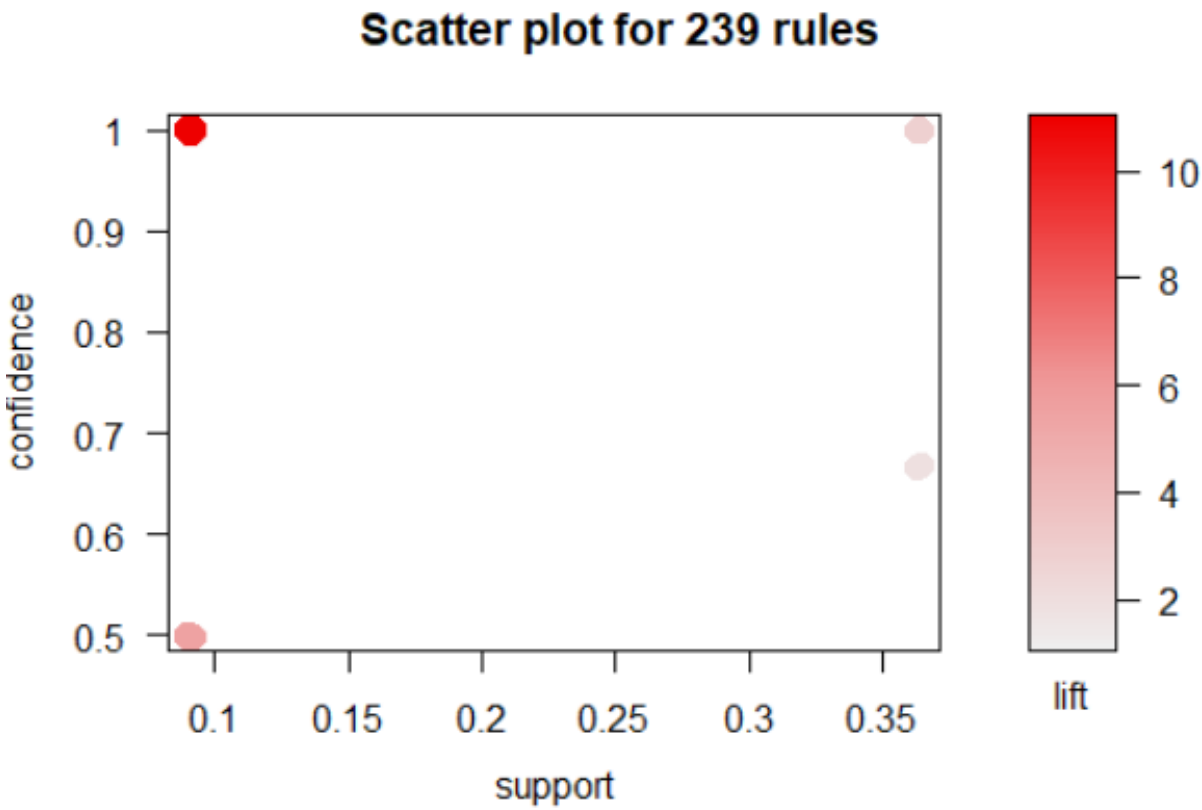



```
> quality(head(rules3))
     support confidence      lift count
1 0.09090909          1  1.833333     1
2 0.09090909          1 11.000000     1
3 0.09090909          1 11.000000     1
4 0.09090909          1  1.833333     1
5 0.09090909          1 11.000000     1
6 0.09090909          1 11.000000     1
> plot(rules3, method = "grouped",control = list(cex=0.90))
```

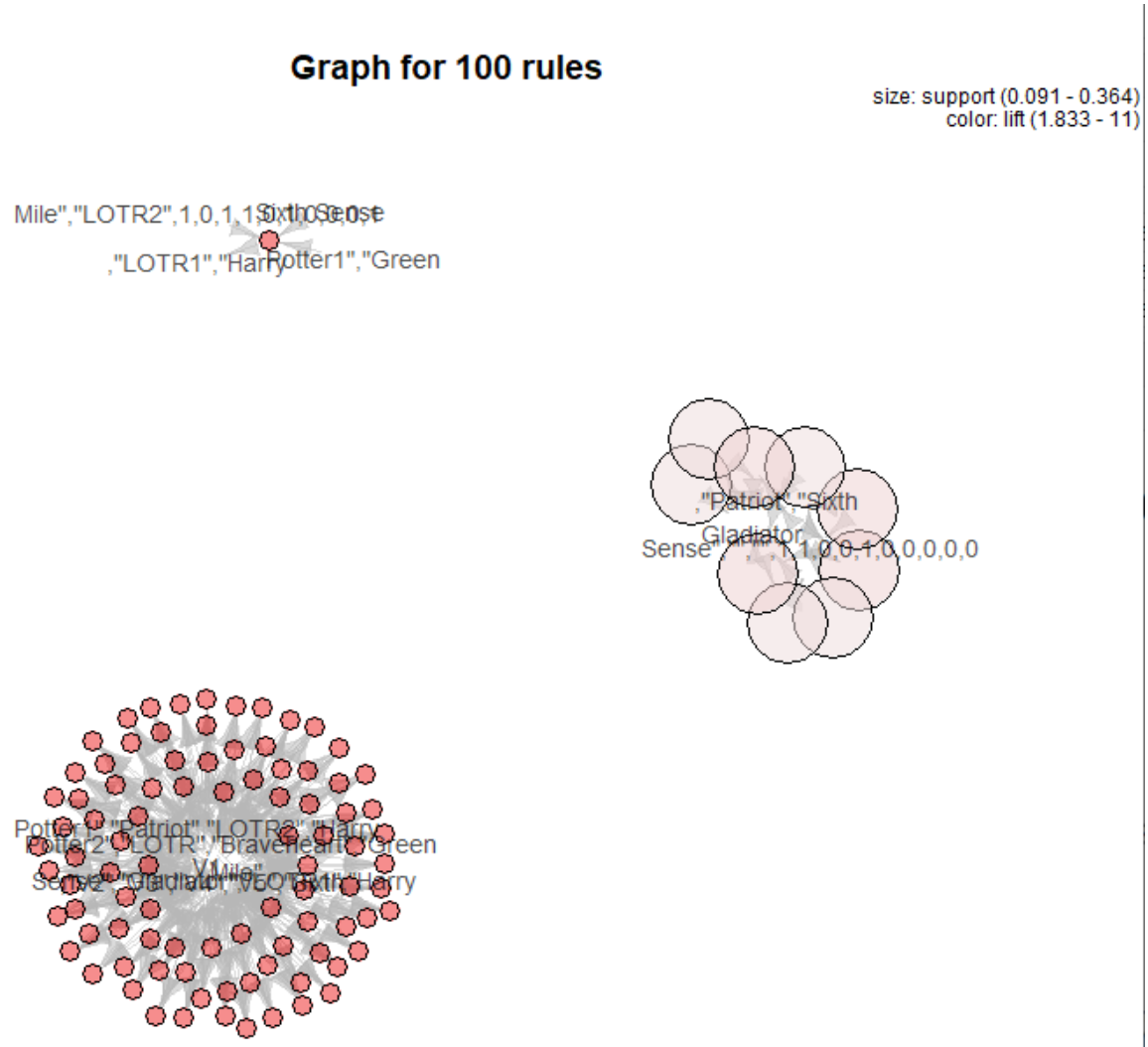



```
Available control parameters (with default values):
main     =  Grouped Matrix for 239 Rules
k        =  20
rhs_max  =  10
lhs_items        =  2
aggr.fun         =  function (x, ...)  UseMethod("mean")
col      =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF",
"#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF",
"#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF",
"#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF",
"#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF",
"#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF",
"#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF"
, "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF
", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DF
F", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7
FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B
1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBC
BCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC
6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#E
ED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#
EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "
#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
reverse  =  TRUE
xlab     =  NULL
ylab     =  NULL
legend   =  Size: support  Color: lift
```

```
spacing  =  -1
panel.function  =  function (row, size, shading, spacing)  {     size[size ==
0] <- NA     shading[is.na(shading)] <- 1     grid.circle(x = c(1:length(size
)), y = row, r = size/2 * (1 - spacing), default.units = "native", gp = gpar(
fill = shading, col = shading, alpha = 0.9)) }
gp_main  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_labels       =  list(cex = 0.8)
gp_labs  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_lines        =  list(col = "gray", lty = 3)
newpage  =  TRUE
max.shading     =  NA
engine   =  default
verbose  =  FALSE
> plot(rules3,method = "scatterplot",control = list(cex=0.90))
```




```
> plot(rules3,method = "graph",control = list(cex=0.90))
```

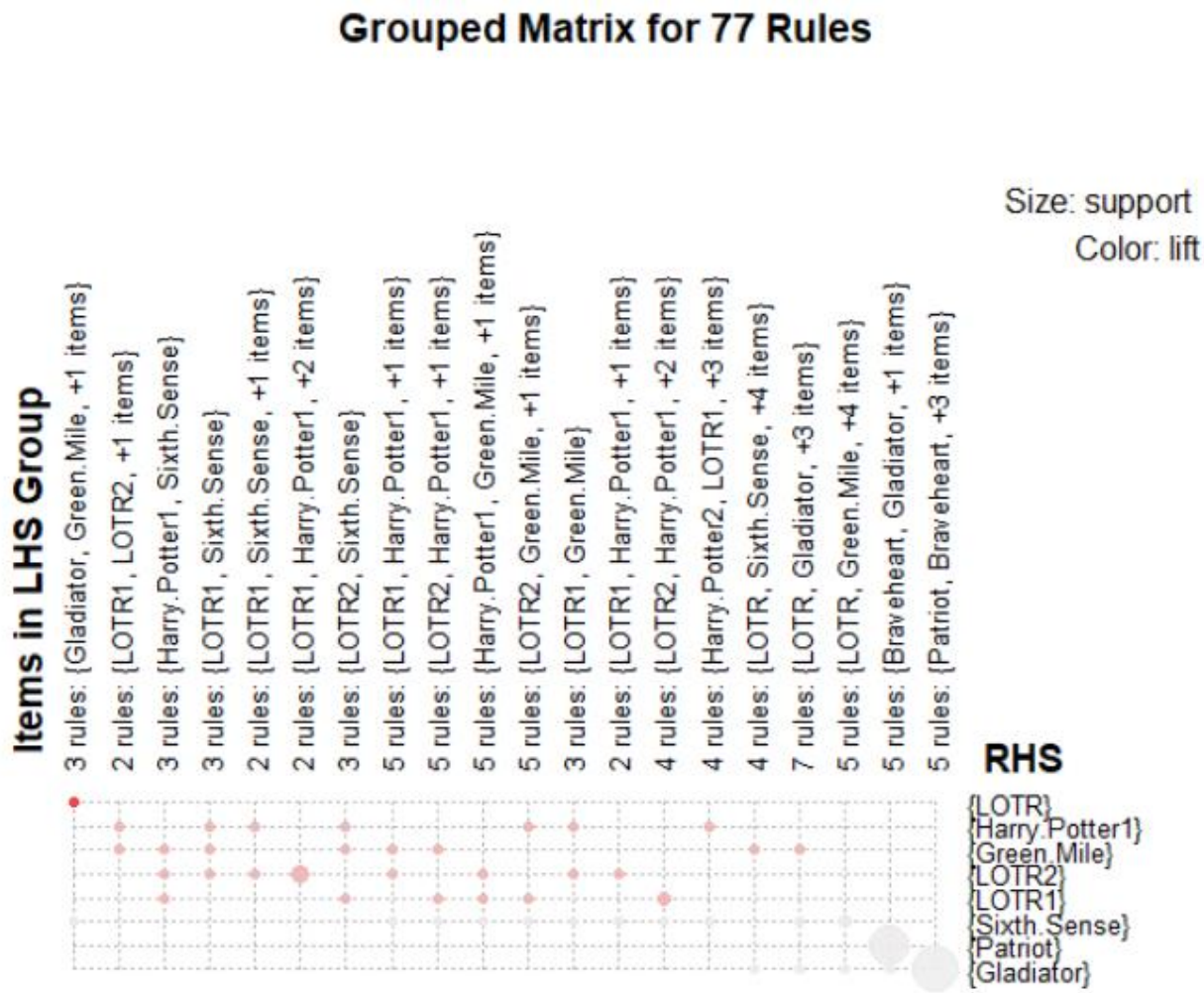



```
> mymovies1 <- read.csv(file.choose())
> names(mymovies1)
 [1] "V1"            "V2"            "V3"            "V4"
 [5] "V5"            "Sixth.Sense"   "Gladiator"     "LOTR1"
 [9] "Harry.Potter1" "Patriot"       "LOTR2"         "Harry.Potter2"
[13] "LOTR"          "Braveheart"    "Green.Mile"
> summary(mymovies1)
             V1                 V2                 V3              V4
 Gladiator    :6   Harry Potter2:1                 :3              :8
 Harry Potter1:1   LOTR         :1   Braveheart   :1   Green Mile:2
 LOTR1        :1   LOTR1        :1   Gladiator    :1
 Sixth Sense  :2   LOTR2        :1   Harry Potter1:1
                   Patriot      :6   Sixth Sense  :4

     V5      Sixth.Sense    Gladiator        LOTR1      Harry.Potter1
      :9   Min.   :0.0   Min.   :0.00   Min.   :0.0   Min.   :0.0
 LOTR2:1   1st Qu.:0.0   1st Qu.:0.25   1st Qu.:0.0   1st Qu.:0.0
           Median :1.0   Median :1.00   Median :0.0   Median :0.0
           Mean   :0.6   Mean   :0.70   Mean   :0.2   Mean   :0.2
           3rd Qu.:1.0   3rd Qu.:1.00   3rd Qu.:0.0   3rd Qu.:0.0
           Max.   :1.0   Max.   :1.00   Max.   :1.0   Max.   :1.0
    Patriot        LOTR2      Harry.Potter2      LOTR       Braveheart
 Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0
 1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0
 Median :1.0   Median :0.0   Median :0.0   Median :0.0   Median :0.0
 Mean   :0.6   Mean   :0.2   Mean   :0.1   Mean   :0.1   Mean   :0.1
 3rd Qu.:1.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0
```

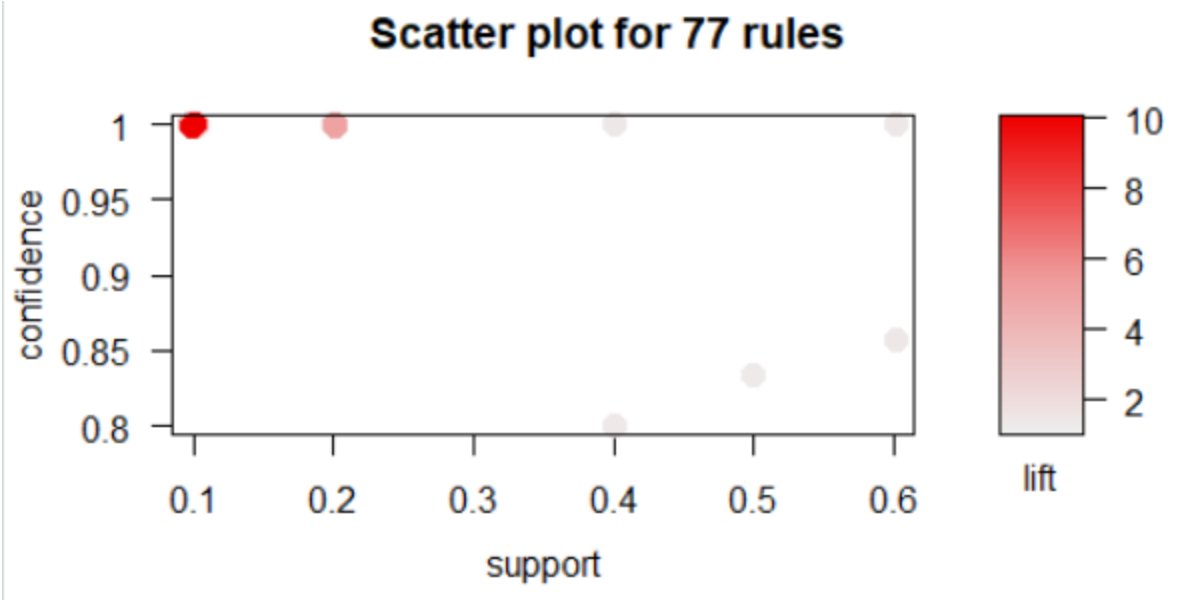
```
 Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0
   Green.Mile
 Min.   :0.0
 1st Qu.:0.0
 Median :0.0
 Mean   :0.2
 3rd Qu.:0.0
 Max.   :1.0
> attach(mymovies1)

> sd(Sixth.Sense)
[1] 0.5163978
> sd(Gladiator)
[1] 0.4830459
> sd(LOTR1)
[1] 0.421637
> sd(Harry.Potter1)
[1] 0.421637
> sd(Patriot)
[1] 0.5163978
> sd(LOTR2)
[1] 0.421637
> sd(Harry.Potter2)
[1] 0.3162278
> sd(LOTR)
[1] 0.3162278
> sd(Braveheart)
[1] 0.3162278
> sd(Green.Mile)
[1] 0.421637
> var(Sixth.Sense)
[1] 0.2666667
> var(Gladiator)
[1] 0.2333333
> var(LOTR1)
[1] 0.1777778
> var(Harry.Potter1)
[1] 0.1777778
> var(Patriot)
[1] 0.2666667
> var(LOTR2)
[1] 0.1777778
> var(Harry.Potter2)
[1] 0.1
> var(LOTR)
[1] 0.1
> var(Braveheart)
[1] 0.1
> var(Green.Mile)
[1] 0.1777778
> library(moments)
> skewness(Sixth.Sense)
[1] -0.4082483
> skewness(Gladiator)
[1] -0.8728716
> skewness(LOTR1)
[1] 1.5
> skewness(Harry.Potter1)
[1] 1.5
> skewness(Patriot)
[1] -0.4082483
> skewness(LOTR2)
[1] 1.5
> skewness(Harry.Potter2)
[1] 2.666667
> skewness(LOTR)
[1] 2.666667
> skewness(Braveheart)
```

```
[1] 2.666667
> skewness(Green.Mile)
[1] 1.5
>
> kurtosis(Sixth.Sense)
[1] 1.166667
> kurtosis(Gladiator)
[1] 1.761905
> kurtosis(LOTR1)
[1] 3.25
> kurtosis(Harry.Potter1)
[1] 3.25
> kurtosis(Patriot)
[1] 1.166667
> kurtosis(LOTR2)
[1] 3.25
> kurtosis(Harry.Potter2)
[1] 8.111111
> kurtosis(LOTR)
[1] 8.111111
> kurtosis(Braveheart)
[1] 8.111111
> kurtosis(Green.Mile)
[1] 3.25

> rules2 <- apriori(as.matrix(mymovies1[,6:15],parameter=list(support=0.2, co
nfidence = 0.5,minlen=5)))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 1

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 10 transaction(s)] done [0.00s].
sorting and recoding items ... [10 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [77 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules2
set of 77 rules
> rules_conf <- sort (rules2, by="confidence", decreasing=T)
> inspect(head(rules_conf))
    lhs                 rhs              support confidence lift     count
[1] {Harry.Potter2} => {Harry.Potter1} 0.1     1          5.000000 1
[2] {Braveheart}    => {Patriot}       0.1     1          1.666667 1
[3] {Braveheart}    => {Gladiator}     0.1     1          1.428571 1
[4] {LOTR}          => {Green.Mile}    0.1     1          5.000000 1
[5] {LOTR}          => {Gladiator}     0.1     1          1.428571 1
[6] {LOTR}          => {Sixth.Sense}   0.1     1          1.666667 1
> rules_lift <- sort(rules2,by="lift",decreasing = T)
> inspect(head(rules_lift))
    lhs                                   rhs              support confidence
[1] {Gladiator,Green.Mile}             => {LOTR}           0.1     1
[2] {Sixth.Sense,Gladiator,Green.Mile} => {LOTR}           0.1     1
[3] {Harry.Potter2}                    => {Harry.Potter1} 0.1     1
[4] {LOTR}                             => {Green.Mile}    0.1     1
[5] {LOTR1}                            => {LOTR2}         0.2     1
[6] {LOTR2}                            => {LOTR1}         0.2     1
```

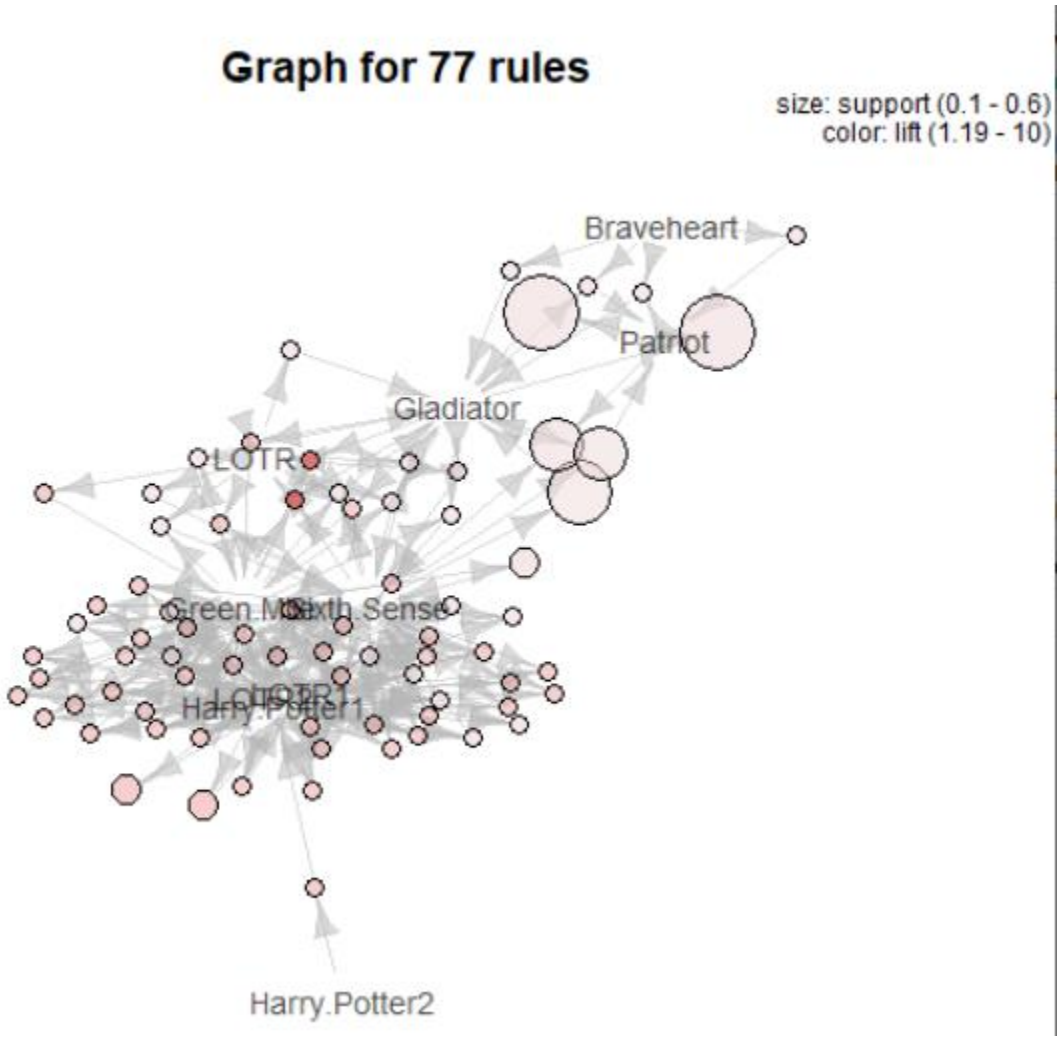
```
    lift count
[1] 10    1
[2] 10    1
[3]  5    1
[4]  5    1
[5]  5    2
[6]  5    2
> plot(rules2, method = "grouped",control = list(cex=0.90))
```




```
> plot(rules2,method = "scatterplot",control = list(cex=0.90))
```




```
> plot(rules2,method = "graph",control = list(cex=0.90))
```



Rcode:

```
#Different set of rule values for book Dataset using apriori algorithm.
book <- read.csv(file.choose())
View(book)
summary(book)
str(book)
class(book)
names(book)
attach(book)
sd(ChildBks)
sd(YouthBks)
sd(CookBks)
sd(DoItYBks)
sd(RefBks)
sd(ArtBks)
sd(GeogBks)
sd(ItalCook)
sd(ItalAtlas)
sd(ItalArt)
sd(Florence)
var(book)

skewness(ChildBks)
skewness(YouthBks)
skewness(CookBks)
skewness(DoItYBks)
skewness(RefBks)
```

```
skewness(ArtBks)
skewness(GeogBks)
skewness(ItalCook)
skewness(ItalAtlas)
skewness(ItalArt)
skewness(Florence)

kurtosis(ChildBks)
kurtosis(YouthBks)
kurtosis(CookBks)
kurtosis(DoItYBks)
kurtosis(RefBks)
kurtosis(ArtBks)
kurtosis(GeogBks)
kurtosis(ItalCook)
kurtosis(ItalAtlas)
kurtosis(ItalArt)
kurtosis(Florence)
rules4 <- apriori(as.matrix(book,parameter=list(support=0.02, confidence =
0.5,minlen=5)))
rules4
rules_confidence <- sort (rules4, by="confidence", decreasing=T)
inspect(head(rules_confidence))
rules_lift <- sort(rules4,by="lift",decreasing = T)
inspect(head(rules_lift))
plot(rules4, method = "grouped",control = list(cex=0.90))
plot(rules4,method = "scatterplot",control = list(cex=0.90))
```

```
plot(rules4,method = "graph",control = list(cex=0.90))


rules5 <- apriori(as.matrix(book,parameter=list(support=0.05, confidence =
0.7,minlen=6)))

rules5

rules_confidence <- sort (rules5, by="confidence", decreasing=T)

inspect(head(rules_confidence))

rules_lift <- sort(rules5,by="lift",decreasing = T)

inspect(head(rules_lift))

plot(rules5, method = "grouped",control = list(cex=0.90))

plot(rules5,method = "scatterplot",control = list(cex=0.90))

plot(rules5,method = "graph",control = list(cex=0.90))


#remove redundant rules

subsetRules <- which(colSums(is.subset(rules4, rules4)) > 1)

length(subsetRules)

rules4 <- rules4[-subsetRules]

rules4


subsetRules <- which(colSums(is.subset(rules5, rules5)) > 1)

length(subsetRules)

rules5 <- rules5[-subsetRules]

rules5
```

Console:

```
> #Different set of rule values for book Dataset using apriori algorithm.
> book <- read.csv(file.choose())
> View(book)
```

A_rules.R × | book × | Untitled1* ×

Filter

	ChildBks	YouthBks	CookBks	DoItYBks	RefBks	ArtBks	GeogBks	ItalCook	ItalAtla
1	0	1	0	1	0	0	1	0	0
2	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	1	1	1	0	1	0	1	0	0
5	0	0	1	0	0	0	1	0	0
6	1	0	0	0	0	1	0	0	0
7	0	1	0	0	0	0	0	0	0
8	0	1	0	0	1	0	0	0	0
9	1	0	0	1	0	0	0	0	0
10	1	1	1	0	0	0	1	0	0

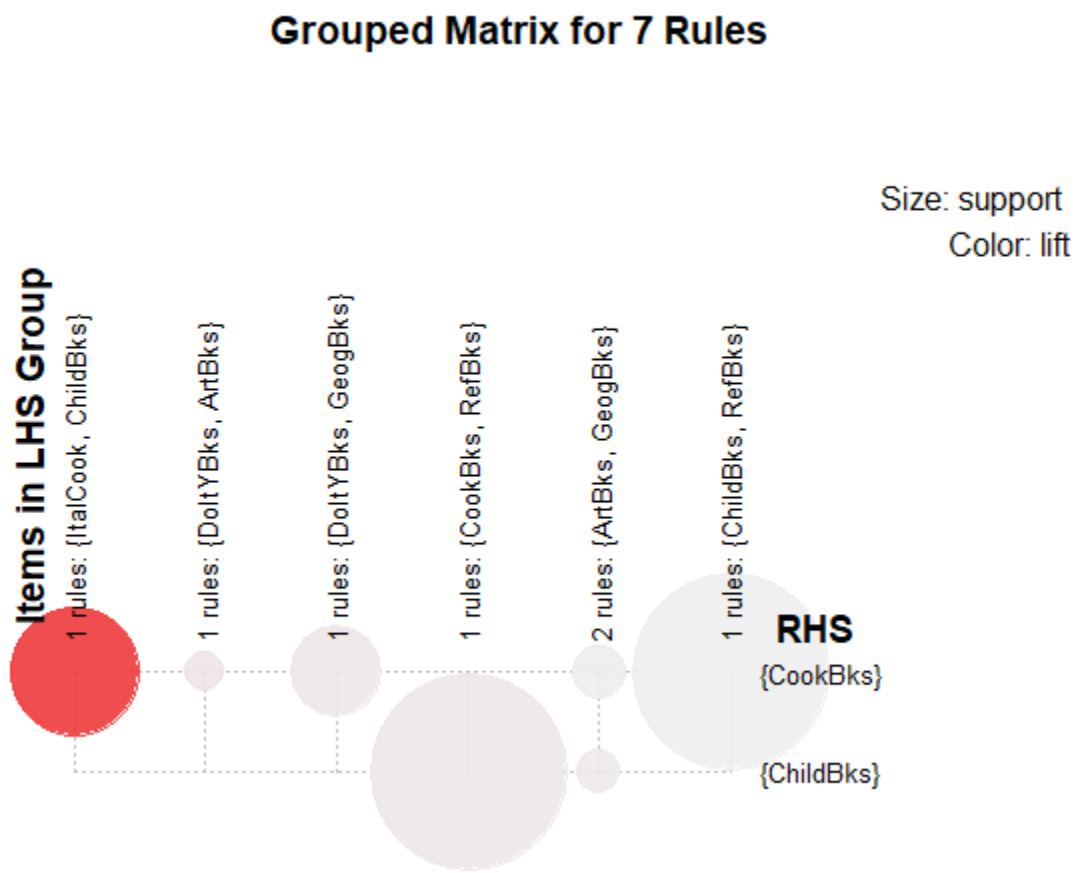
Showing 1 to 10 of 2,000 entries, 11 total columns

```
> summary(book)
    ChildBks        YouthBks         CookBks         DoItYBks
 Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   Median :0.0000   Median :0.000   Median :0.000
 Mean   :0.423   Mean   :0.2475   Mean   :0.431   Mean   :0.282
 3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:1.000   3rd Qu.:1.000
 Max.   :1.000   Max.   :1.0000   Max.   :1.000   Max.   :1.000
     RefBks           ArtBks          GeogBks         ItalCook
 Min.   :0.0000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
 Median :0.0000   Median :0.000   Median :0.000   Median :0.0000
 Mean   :0.2145   Mean   :0.241   Mean   :0.276   Mean   :0.1135
 3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.000   Max.   :1.000   Max.   :1.0000
   ItalAtlas        ItalArt          Florence
 Min.   :0.000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.000   Median :0.0000   Median :0.0000
 Mean   :0.037   Mean   :0.0485   Mean   :0.1085
 3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.000   Max.   :1.0000   Max.   :1.0000
> str(book)
'data.frame':	2000 obs. of  11 variables:
 $ ChildBks : int  0 1 0 1 0 1 0 0 1 1 ...
 $ YouthBks : int  1 0 0 1 0 0 1 1 0 1 ...
 $ CookBks  : int  0 0 0 1 1 0 0 0 0 1 ...
 $ DoItYBks : int  1 0 0 0 0 0 0 0 1 0 ...
 $ RefBks   : int  0 0 0 1 0 0 0 1 0 0 ...
 $ ArtBks   : int  0 0 0 0 0 1 0 0 0 0 ...
 $ GeogBks  : int  1 0 0 1 1 0 0 0 0 1 ...
 $ ItalCook : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ItalAtlas: int  0 0 0 0 0 0 0 0 0 0 ...
 $ ItalArt  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Florence : int  0 0 0 0 0 1 0 0 0 0 ...
> class(book)
[1] "data.frame"
> names(book)
 [1] "ChildBks"  "YouthBks"  "CookBks"   "DoItYBks"  "RefBks"    "ArtBks"
 [7] "GeogBks"   "ItalCook"  "ItalAtlas" "ItalArt"   "Florence"
> attach(book)
> sd(ChildBks)
[1] 0.494159
> sd(YouthBks)
[1] 0.4316676
```

```
> sd(CookBks)
[1] 0.49534
> sd(DoItYBks)
[1] 0.4500859
> sd(RefBks)
[1] 0.4105777
> sd(ArtBks)
[1] 0.4277973
> sd(GeogBks)
[1] 0.4471286
> sd(ItalCook)
[1] 0.3172823
> sd(ItalAtlas)
[1] 0.188809
> sd(ItalArt)
[1] 0.214874
> sd(Florence)
[1] 0.3110886
> var(book)
             ChildBks     YouthBks      CookBks     DoItYBks      RefBks
ChildBks  0.244193097  0.060337669 0.0737238619 0.0647463732 0.060796898
YouthBks  0.060337669  0.186336918 0.0553551776 0.0457278639 0.043432966
CookBks   0.073723862  0.055355178 0.2453616808 0.0659909955 0.060080540
DoItYBks  0.064746373  0.045727864 0.0659909955 0.2025772886 0.045033517
RefBks    0.060796898  0.043432966 0.0600805403 0.0450335168 0.168574037
ArtBks    0.060587294  0.041373187 0.0631605803 0.0555657829 0.037824412
GeogBks   0.078291146  0.052216108 0.0735807904 0.0546953477 0.051323662
ItalCook  0.037008004  0.030924212 0.0646138069 0.0265062531 0.022165333
ItalAtlas 0.012855428  0.008346673 0.0125592796 0.0085702851 0.029078039
ItalArt   0.015492246  0.011001751 0.0201065533 0.0163311656 0.009601551
Florence  0.002605803 -0.001354427 0.0007368684 0.0009034517 0.007730615
               ArtBks    GeogBks    ItalCook   ItalAtlas     ItalArt
ChildBks  0.060587294 0.07829115 0.037008004 0.012855428 0.015492246
YouthBks  0.041373187 0.05221611 0.030924212 0.008346673 0.011001751
CookBks   0.063160580 0.07358079 0.064613807 0.012559280 0.020106553
DoItYBks  0.055565783 0.05469535 0.026506253 0.008570285 0.016331166
RefBks    0.037824412 0.05132366 0.022165333 0.029078039 0.009601551
ArtBks    0.183010505 0.06101451 0.029161081 0.009087544 0.036829915
GeogBks   0.061014507 0.19992396 0.032690345 0.010293147 0.016122061
ItalCook  0.029161081 0.03269035 0.100668084 0.018809905 0.032011256
ItalAtlas 0.009087544 0.01029315 0.018809905 0.035648824 0.014712856
ItalArt   0.036829915 0.01612206 0.032011256 0.014712856 0.046170835
Florence  0.022362681 0.01256028 0.005187844 0.002486743 0.007241371
               Florence
ChildBks   0.0026058029
YouthBks  -0.0013544272
CookBks    0.0007368684
DoItYBks   0.0009034517
RefBks     0.0077306153
ArtBks     0.0223626813
GeogBks    0.0125602801
ItalCook   0.0051878439
ItalAtlas  0.0024867434
ItalArt    0.0072413707
Florence   0.0967761381
>
> skewness(ChildBks)
[1] 0.3117185
> skewness(YouthBks)
[1] 1.170174
> skewness(CookBks)
[1] 0.2786662
> skewness(DoItYBks)
[1] 0.9689463
> skewness(RefBks)
[1] 1.391071
> skewness(ArtBks)
[1] 1.211157
```

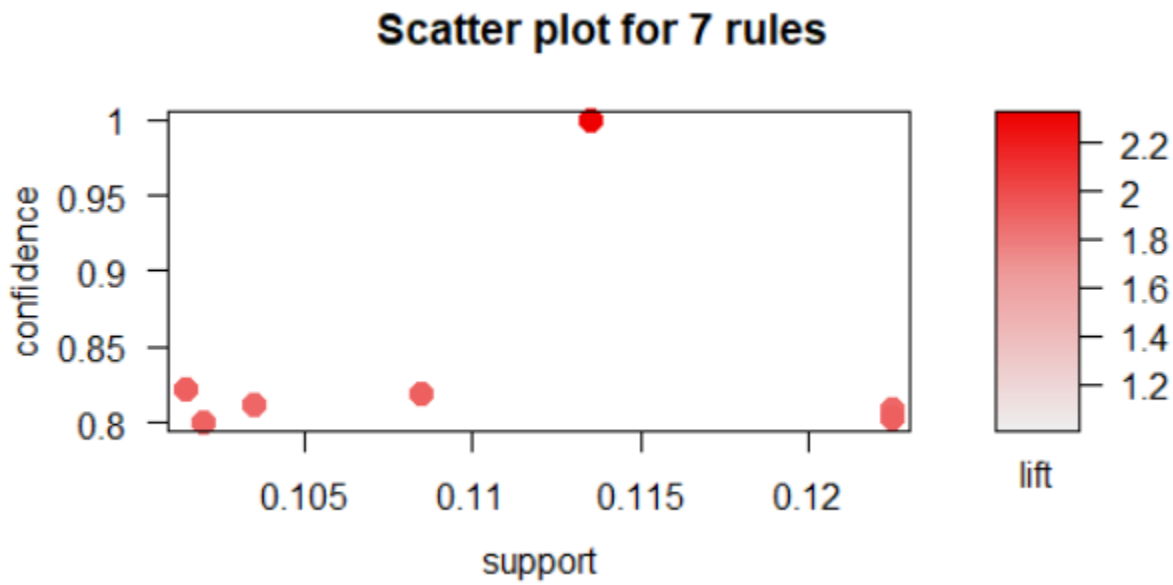
```
> skewness(GeogBks)
[1] 1.0022
> skewness(ItalCook)
[1] 2.436925
> skewness(ItalAtlas)
[1] 4.905655
> skewness(ItalArt)
[1] 4.203514
> skewness(Florence)
[1] 2.517597

> kurtosis(ChildBks)
[1] 1.097168
> kurtosis(YouthBks)
[1] 2.369308
> kurtosis(CookBks)
[1] 1.077655
> kurtosis(DoItYBks)
[1] 1.938857
> kurtosis(RefBks)
[1] 2.935079
> kurtosis(ArtBks)
[1] 2.466901
> kurtosis(GeogBks)
[1] 2.004404
> kurtosis(ItalCook)
[1] 6.938604
> kurtosis(ItalAtlas)
[1] 25.06545
> kurtosis(ItalArt)
[1] 18.66953
> kurtosis(Florence)
[1] 7.338295

> rules4 <- apriori(as.matrix(book,parameter=list(support=0.02, confidence =
0.5,minlen=5)))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 200

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 2000 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules4
set of 7 rules
> rules_confidence <- sort (rules4, by="confidence", decreasing=T)
> inspect(head(rules_confidence))
    lhs                    rhs        support confidence lift     count
[1] {ItalCook}         => {CookBks}  0.1135  1.0000000  2.320186 227
[2] {DoItYBks,ArtBks}  => {CookBks}  0.1015  0.8218623  1.906873 203
[3] {DoItYBks,GeogBks} => {CookBks}  0.1085  0.8188679  1.899926 217
[4] {ArtBks,GeogBks}   => {CookBks}  0.1035  0.8117647  1.883445 207
[5] {ChildBks,RefBks}  => {CookBks}  0.1225  0.8085809  1.876058 245
[6] {CookBks,RefBks}   => {ChildBks} 0.1225  0.8032787  1.899004 245
```

```
> rules_lift <- sort(rules4,by="lift",decreasing = T)
> inspect(head(rules_lift))
    lhs                    rhs        support confidence lift     count
[1] {ItalCook}          => {CookBks}  0.1135  1.0000000  2.320186 227
[2] {DoItYBks,ArtBks}   => {CookBks}  0.1015  0.8218623  1.906873 203
[3] {DoItYBks,GeogBks}  => {CookBks}  0.1085  0.8188679  1.899926 217
[4] {CookBks,RefBks}    => {ChildBks} 0.1225  0.8032787  1.899004 245
[5] {ArtBks,GeogBks}    => {ChildBks} 0.1020  0.8000000  1.891253 204
[6] {ArtBks,GeogBks}    => {CookBks}  0.1035  0.8117647  1.883445 207
> plot(rules4, method = "grouped",control = list(cex=0.90))
```

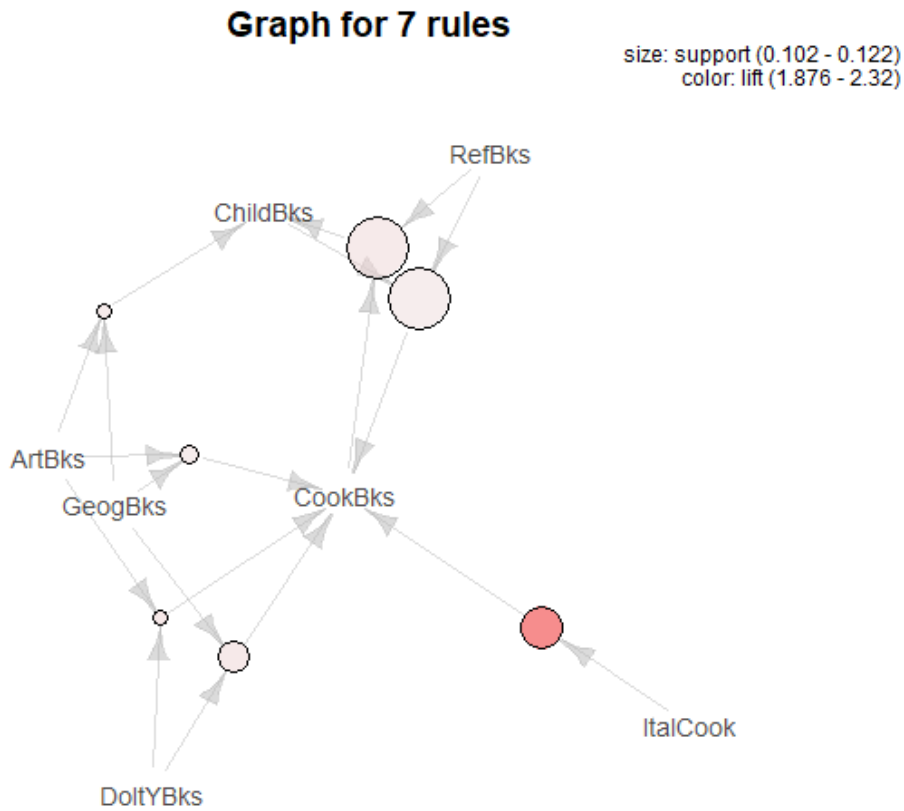



```
Available control parameters (with default values):
main     =  Grouped Matrix for 7 Rules
k        =  20
rhs_max  =  10
lhs_items        =  2
aggr.fun         =  function (x, ...)  UseMethod("mean")
col      =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF",
"#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF",
"#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF",
"#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF",
"#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF",
"#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF",
"#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF"
, "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF
", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DF
F", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7
FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B
1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBC
BCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC
6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#E
ED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#
EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "
#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
reverse  =  TRUE
```

```
xlab     =  NULL
ylab     =  NULL
legend   =  Size: support  Color: lift
spacing  =  -1
panel.function  =  function (row, size, shading, spacing)  {     size[size ==
0] <- NA     shading[is.na(shading)] <- 1     grid.circle(x = c(1:length(size
)), y = row, r = size/2 * (1 - spacing), default.units = "native", gp = gpar(
fill = shading, col = shading, alpha = 0.9)) }
gp_main  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_labels        =  list(cex = 0.8)
gp_labs  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_lines         =  list(col = "gray", lty = 3)
newpage  =  TRUE
max.shading      =  NA
engine   =  default
verbose  =  FALSE
> plot(rules4,method = "scatterplot",control = list(cex=0.90))
```

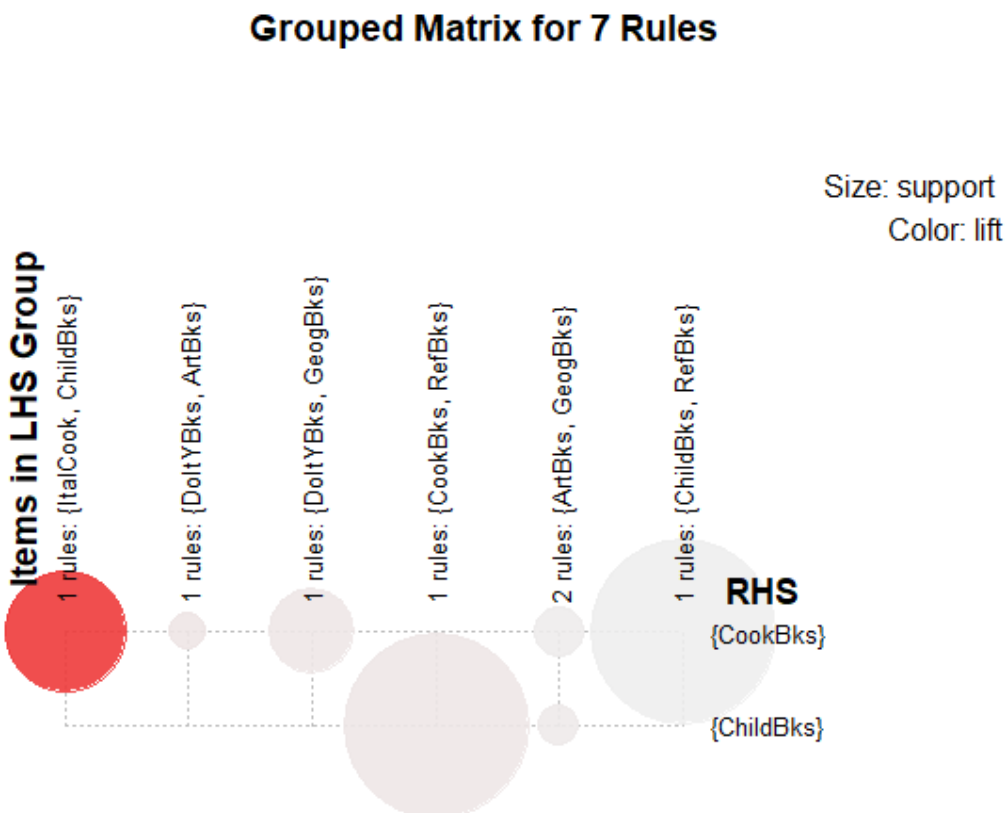



```
> plot(rules4,method = "graph",control = list(cex=0.90))
```



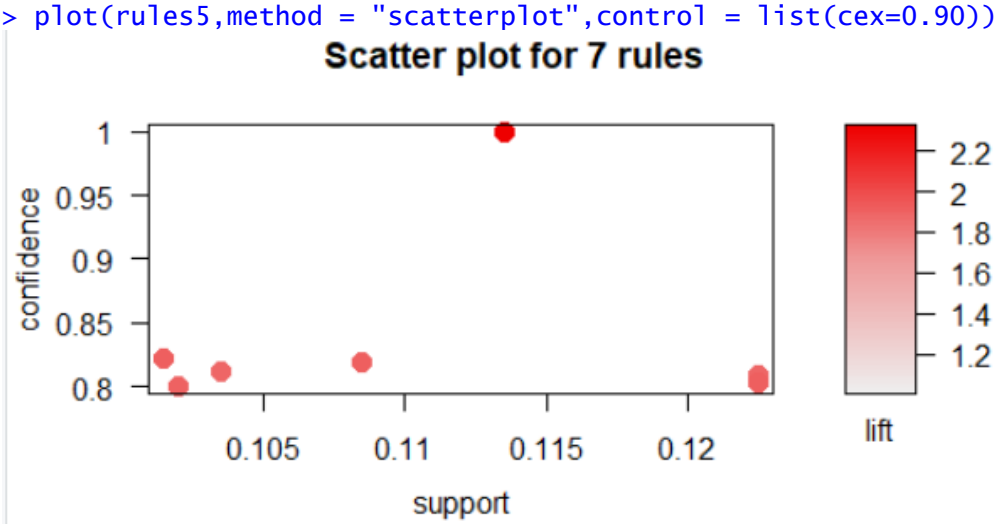

```
> rules5 <- apriori(as.matrix(book,parameter=list(support=0.05, confidence =
0.7,minlen=6)))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 200

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 2000 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules5
set of 7 rules
> rules_confidence <- sort (rules5, by="confidence", decreasing=T)
> inspect(head(rules_confidence))
    lhs                    rhs        support confidence lift     count
[1] {ItalCook}          => {CookBks}  0.1135  1.0000000  2.320186 227
[2] {DoItYBks,ArtBks}   => {CookBks}  0.1015  0.8218623  1.906873 203
[3] {DoItYBks,GeogBks}  => {CookBks}  0.1085  0.8188679  1.899926 217
[4] {ArtBks,GeogBks}    => {CookBks}  0.1035  0.8117647  1.883445 207
[5] {ChildBks,RefBks}   => {CookBks}  0.1225  0.8085809  1.876058 245
[6] {CookBks,RefBks}    => {ChildBks} 0.1225  0.8032787  1.899004 245
> rules_lift <- sort(rules5,by="lift",decreasing = T)
> inspect(head(rules_lift))
```

```
    lhs                    rhs        support confidence lift     count
[1] {ItalCook}         => {CookBks}  0.1135  1.0000000  2.320186 227
[2] {DoItYBks,ArtBks}  => {CookBks}  0.1015  0.8218623  1.906873 203
[3] {DoItYBks,GeogBks} => {CookBks}  0.1085  0.8188679  1.899926 217
[4] {CookBks,RefBks}   => {ChildBks} 0.1225  0.8032787  1.899004 245
[5] {ArtBks,GeogBks}   => {ChildBks} 0.1020  0.8000000  1.891253 204
[6] {ArtBks,GeogBks}   => {CookBks}  0.1035  0.8117647  1.883445 207
> plot(rules5, method = "grouped",control = list(cex=0.90))
```




```
Available control parameters (with default values):
main     =  Grouped Matrix for 7 Rules
k        =  20
rhs_max  =  10
lhs_items       =  2
aggr.fun        =  function (x, ...)  UseMethod("mean")
col      =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF",
"#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF",
"#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF",
"#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF",
"#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF",
"#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF",
"#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF"
,  "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF
", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DF
F", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7
FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B
1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBC
BCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC
6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#E
ED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#
EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "
#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
reverse  =  TRUE
xlab     =  NULL
ylab     =  NULL
legend   =  Size: support  Color: lift
spacing  =  -1
```

```
panel.function  =  function (row, size, shading, spacing)  {     size[size ==
0] <- NA     shading[is.na(shading)] <- 1     grid.circle(x = c(1:length(size
)), y = row, r = size/2 * (1 - spacing), default.units = "native", gp = gpar(
fill = shading, col = shading, alpha = 0.9)) }
gp_main  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_labels        =  list(cex = 0.8)
gp_labs  =  list(cex = 1.2, fontface = "bold", font = c(bold = 2))
gp_lines         =  list(col = "gray", lty = 3)
newpage  =  TRUE
max.shading      =  NA
engine   =  default
verbose  =  FALSE
> plot(rules5,method = "scatterplot",control = list(cex=0.90))
```




```
> plot(rules5,method = "graph",control = list(cex=0.90))
```