Regras de Associação Apriori

Heitor Gabriel S. Monteiro

08/11/2021

Contents

1 Prelúdio

Nosso objetivo é entender possíveis associações entre itens na cesta de consumo de uma base de dados de compras no supermercado. Para tal, usaremos o algoritmo *Apriori* que tem alguns conceitos chaves:

- **Support(A):**
$$= \frac{Freq(A)}{N} = P(A)$$
- **Support(A,B):**
$$= \frac{Freq(A \wedge B)}{N} \quad = \quad P(A \cap B)$$
- **Confidence(A $\rightarrow$ B):**
$$= \frac{Supp(A,B)}{Supp(A)} = \frac{Freq(A \wedge B)}{Freq(A)} = \frac{P(A \cap B)}{P(A)} = P(B|A)$$
- **Lift(A $\rightarrow$ B):**
$$= Lift(B \rightarrow A) = \quad \frac{Conf(A \rightarrow B)}{Supp(A)Supp(B)} \quad = \frac{P(A \cap B)}{P(A)P(B)}$$

Aplicando-os em nossa situação, *Support* é a probabilidade de encontrar o produto A numa cesta de compras, ou uma sub-cesta (A, B) na cesta. *Confidence* é a chance de encontrar o produto B se já encontramos o A, a frequência do B em todas as compras que contenham A. *Lift* mede a chance de um conjunto (A, B) aparecer juntos comparado à chance de suas frequências absolutas. Se, por exemplo, $Lift(A, B) > 1$, então há mais chances de ver os produtos A e B juntos do que encontrar A ou B, sozinho.

2 Importação e Descrição dos Dados

Vamos definir o diretório de trabalho, que contém os arquivos e guardará as saídas, e carregar os pacotes, com destaque para o `arules` e o `arulesViz`.

```
setwd('/home/heitor/Área de Trabalho/R Projects/Análise Macro/Labs/Lab 11')

library(tidyverse)
library(arules)
library(arulesViz)
library(RColorBrewer)
library(knitr)
library(kableExtra)
```

O tipo otimizador de lidar com esse banco é trabalhar como arquivo de transações, usaremos então o `read.transactions()`. Na visão geral, vemos 9.835 transações com 169 categorias de produtos; o tamanho médio das compras são 4,4 produtos.

```
dds <- read.transactions("groceries.csv", sep = ",")
summary(dds)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
```

```
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
inspect(dds[1:5])
```

```
##     items
## [1] {citrus fruit,
##      margarine,
##      ready soups,
##      semi-finished bread}
## [2] {coffee,
##      tropical fruit,
##      yogurt}
## [3] {whole milk}
## [4] {cream cheese,
##      meat spreads,
##      pip fruit,
##      yogurt}
## [5] {condensed milk,
##      long life bakery product,
##      other vegetables,
##      whole milk}
```

Podemos ver o *support* de cada item. Também fiz as seguintes operações para conseguir observar o *support* por pesquisa, no caso, pesquisei `ine`:

```
itemFrequency(dds[,1:3])
```

```
## abrasive cleaner artif. sweetener   baby cosmetics
##     0.0035587189     0.0032536858     0.0006100661
```

```
tst <- itemFrequency(dds) %>%
    as.data.frame() %>%
    rownames_to_column(var='product')
```

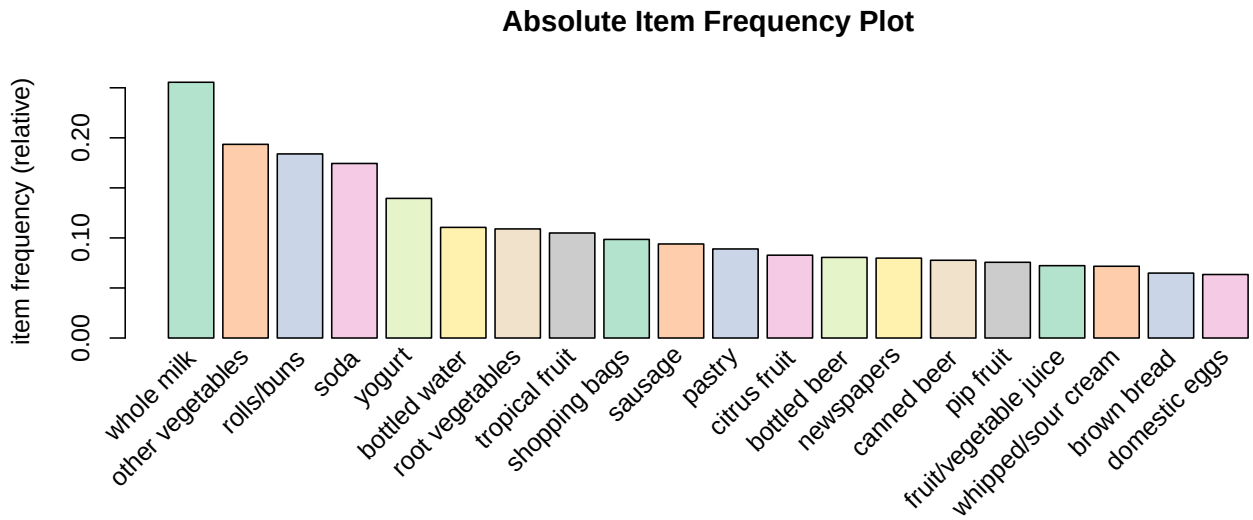
Table 1: Products that have 'ine' in name.

product	frq
margarine	0.0585663
red/blush wine	0.0192171
sparkling wine	0.0055923
vinegar	0.0065074
white wine	0.0190137

```
tst <- dplyr::rename(tst, 'frq' = `.`)
tst %>%
  dplyr::filter(str_detect(product, 'ine')) %>%
  kable(caption = "Products that have 'ine' in name.") %>%
  kable_styling(full_width =F,
                position = 'center',
                bootstrap_options =
                  c("striped", "hover",
                    "condensed", "responsive"))
```

Com o programa, podemos fazer o plot dos mais frequentes produtos comprados:

```
itemFrequencyPlot(dds, topN = 20,
                  type = "relative",
                  col  = brewer.pal(8,'Pastel2'),
                  main = "Absolute Item Frequency Plot")
```



3 O Modelo

Precisamos definir o modelo d acordo com os mínimos `support`, `confidence`, e `minlen`, que é o tamanho mínimo da cesta comprada. Considerando que as transações são de um período de um mês, determinei que a frequência seja de duas compras e meia por dia, daí o $(2.5 * 30)/9835$. Média de suporte e confidência não próximos do fixado é um indicativo que não escolhemos valores muito grandes ao ponto de prejudicar-nos.

```
rule1 <- apriori(dds, parameter=
                    list(support    = (2.5*30)/9835,
                         confidence = 0.25,
                         minlen     = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime     support minlen
##        0.25    0.1    1 none FALSE            TRUE       5 0.007625826      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 75
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [102 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [318 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rule1
```

```
## set of 318 rules
```

```
rule1 %>% summary()
```

```
## set of 318 rules
```

```
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 124 186   8
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.635   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage            lift
##  Min.   :0.007626   Min.   :0.2513   Min.   :0.01230   Min.   :0.9932
##  1st Qu.:0.008643   1st Qu.:0.2975   1st Qu.:0.02298   1st Qu.:1.6078
##  Median :0.010269   Median :0.3602   Median :0.03010   Median :1.8968
##  Mean   :0.013760   Mean   :0.3760   Mean   :0.03896   Mean   :1.9903
##  3rd Qu.:0.014718   3rd Qu.:0.4442   3rd Qu.:0.04342   3rd Qu.:2.3248
##  Max.   :0.074835   Max.   :0.6389   Max.   :0.25552   Max.   :3.7969
##      count
##  Min.   : 75.0
##  1st Qu.: 85.0
##  Median :101.0
##  Mean   :135.3
##  3rd Qu.:144.8
##  Max.   :736.0
##
## mining info:
##  data ntransactions     support confidence
##   dds          9835 0.007625826       0.25
```

Podemos ver quais são as regras de associação descobertas, organizadas por pesquisa ou por ranqueamentos:

```
inspect(rule1[1:3])
```

```
##     lhs            rhs                support     confidence coverage
## [1] {herbs}     => {other vegetables} 0.007727504 0.4750000  0.01626843
## [2] {herbs}     => {whole milk}       0.007727504 0.4750000  0.01626843
## [3] {detergent} => {whole milk}       0.008947636 0.4656085  0.01921708
##     lift     count
## [1] 2.454874 76
## [2] 1.858983 76
## [3] 1.822228 88
```

```r
# procurando por regras que contenham "yogurt"
inspect(
    subset(rule1, items %pin% "yogurt")[1:10]) %>%
    as.data.frame() %>%
    kable(caption = "Rules that have 'yogurt' in name.") %>%
  kable_styling(full_width =F,
              position = 'center',
              bootstrap_options =
                c("striped", "hover",
                  "condensed", "responsive"))
```

```
##      lhs                          rhs      support     confidence coverage
## [1]  {butter milk}           => {yogurt} 0.008540925 0.3054545  0.02796136
## [2]  {sliced cheese}         => {yogurt} 0.008032537 0.3278008  0.02450432
## [3]  {berries}               => {yogurt} 0.010574479 0.3180428  0.03324860
## [4]  {dessert}               => {yogurt} 0.009862735 0.2657534  0.03711235
## [5]  {cream cheese}          => {yogurt} 0.012404677 0.3128205  0.03965430
## [6]  {frozen vegetables}     => {yogurt} 0.012404677 0.2579281  0.04809354
## [7]  {curd}                  => {yogurt} 0.017285206 0.3244275  0.05327911
## [8]  {butter}                => {yogurt} 0.014641586 0.2642202  0.05541434
## [9]  {fruit/vegetable juice} => {yogurt} 0.018708693 0.2587904  0.07229283
## [10] {whipped/sour cream}    => {yogurt} 0.020742247 0.2893617  0.07168277
##      lift     count
## [1]  2.189610  84
## [2]  2.349797  79
## [3]  2.279848 104
## [4]  1.905018  97
## [5]  2.242412 122
## [6]  1.848924 122
## [7]  2.325615 170
## [8]  1.894027 144
## [9]  1.855105 184
## [10] 2.074251 204
```

```r
# organizando por uma característica específica
inspect(sort(rule1, by = "count")[1:5])
```

```
##     lhs                    rhs                support    confidence coverage
## [1] {other vegetables} => {whole milk}       0.07483477 0.3867578  0.1934926
## [2] {whole milk}       => {other vegetables} 0.07483477 0.2928770  0.2555160
## [3] {rolls/buns}       => {whole milk}       0.05663447 0.3079049  0.1839349
## [4] {yogurt}           => {whole milk}       0.05602440 0.4016035  0.1395018
```

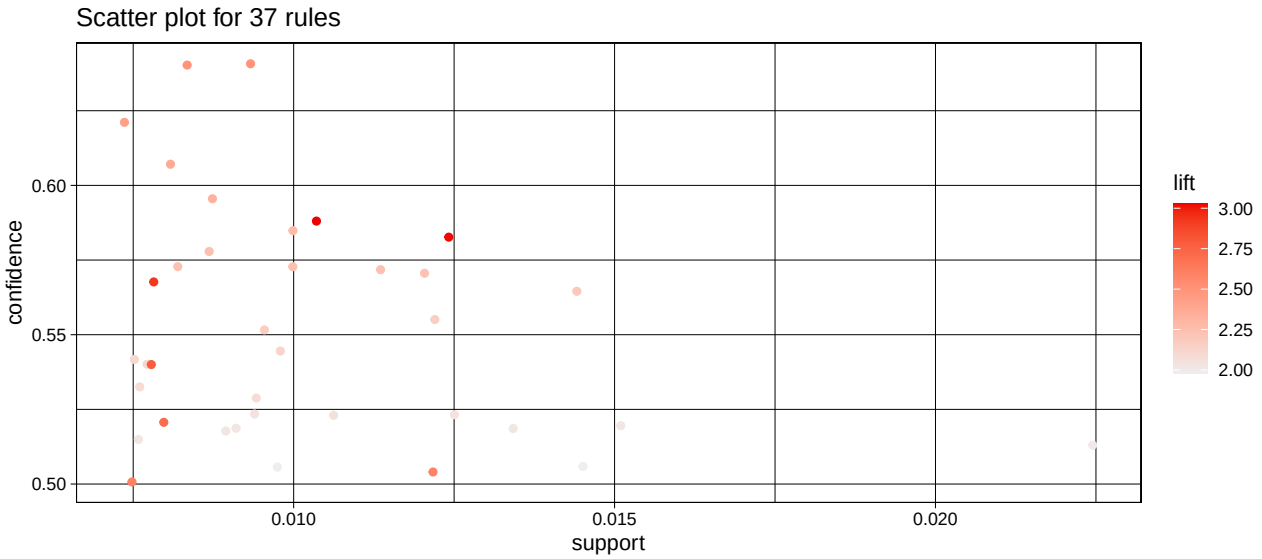
Table 2: Rules that have 'yogurt' in name.

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{butter milk}	=>	{yogurt}	0.0085409	0.3054545	0.0279614	2.189610	84
[2]	{sliced cheese}	=>	{yogurt}	0.0080325	0.3278008	0.0245043	2.349797	79
[3]	{berries}	=>	{yogurt}	0.0105745	0.3180428	0.0332486	2.279848	104
[4]	{dessert}	=>	{yogurt}	0.0098627	0.2657534	0.0371124	1.905018	97
[5]	{cream cheese}	=>	{yogurt}	0.0124047	0.3128205	0.0396543	2.242412	122
[6]	{frozen vegetables}	=>	{yogurt}	0.0124047	0.2579281	0.0480935	1.848923	122
[7]	{curd}	=>	{yogurt}	0.0172852	0.3244275	0.0532791	2.325615	170
[8]	{butter}	=>	{yogurt}	0.0146416	0.2642202	0.0554143	1.894027	144
[9]	{fruit/vegetable juice}	=>	{yogurt}	0.0187087	0.2587904	0.0722928	1.855105	184
[10]	{whipped/sour cream}	=>	{yogurt}	0.0207422	0.2893617	0.0716828	2.074251	204

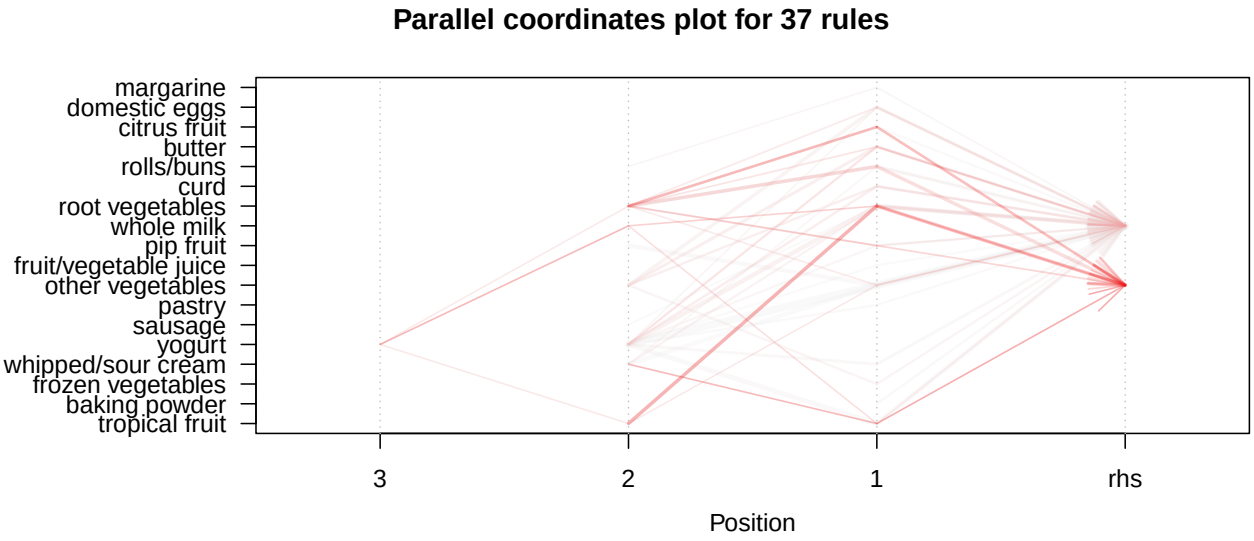
```
## [5] {root vegetables}  => {whole milk}       0.04890696 0.4486940  0.1089985
##     lift     count
## [1] 1.513634 736
## [2] 1.513634 736
## [3] 1.205032 557
## [4] 1.571735 551
## [5] 1.756031 481
```

Para vizualização, podemos fazer um refinamento maior ainda, para ficar com as maiores *confidence*, acima de 0.5. A visualização mostra que

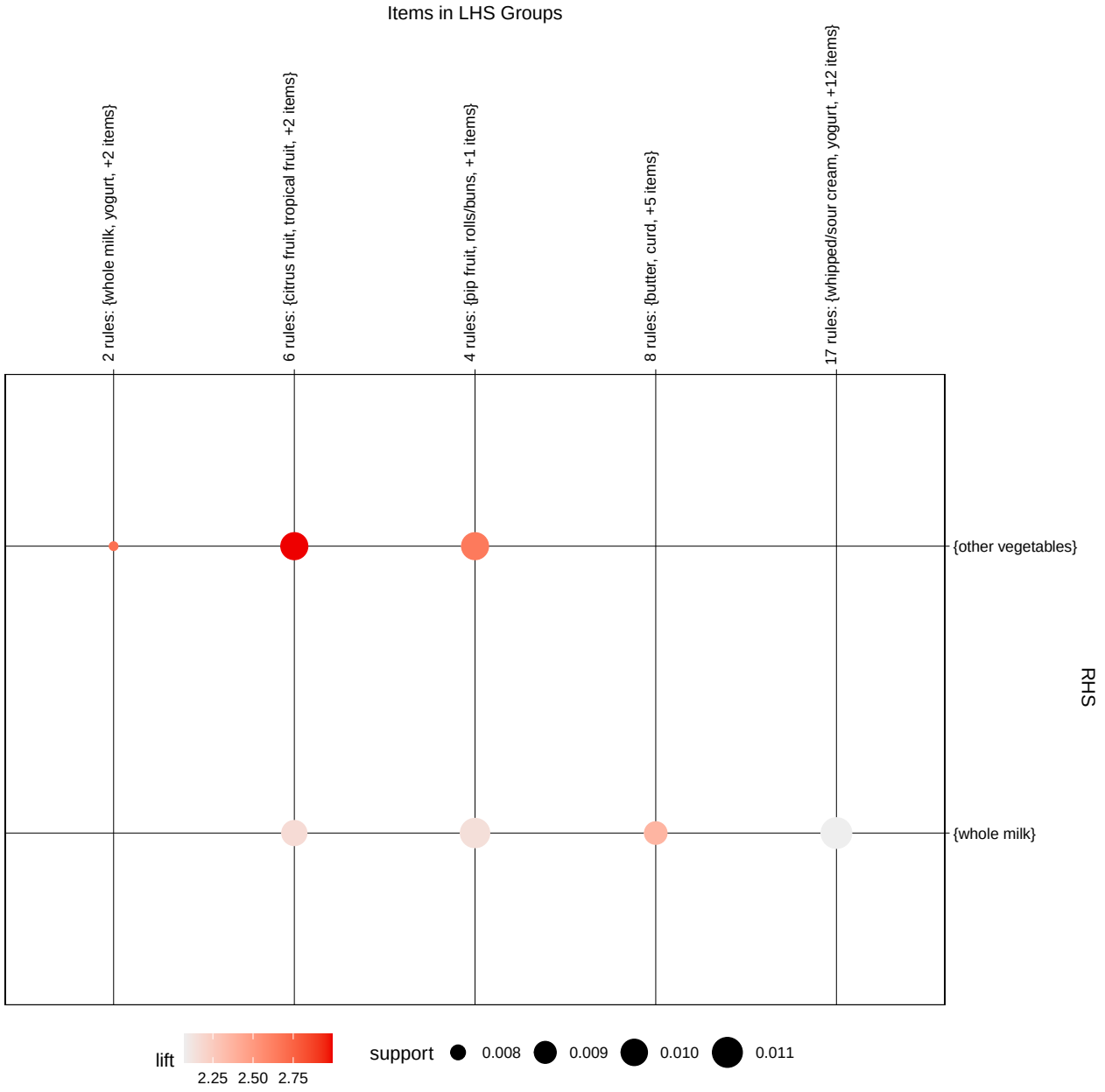
```
sub_rule1 <- rule1[quality(rule1)$confidence>0.5]

plot(sub_rule1)
```



```
plot(sub_rule1,
     method="paracoord",
     control=list(alpha=.5, reorder=TRUE))
```



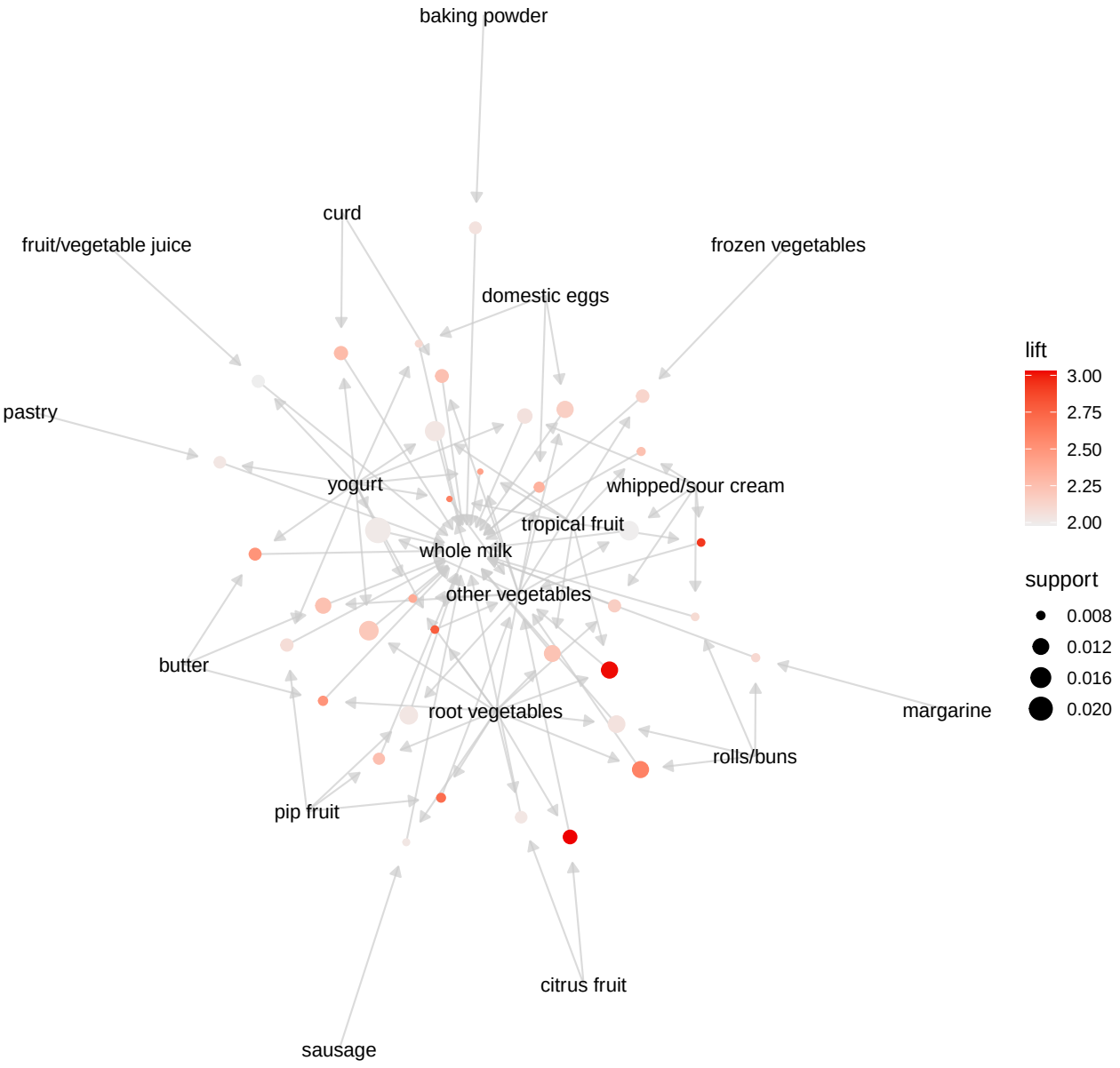

```
plot(sub_rule1,
     method = "grouped", control = list(k = 5))
```




```
plot(sub_rule1,
     method="graph", control=list(type="items"))
```

```
## Available control parameters (with default values):
## layout    =  list(fun = function (graph, dim = 2, ...)  {     if ("layout" %in% graph
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
```

```
## verbose    =  FALSE
```




```
#plot(sub_rule1,
#    measure=c("support","lift"),
#    shading="confidence",interactive=T)

top10subRules <- head(sub_rule1,
                      n = 10, by = "confidence")
plot(top10subRules, method = "graph",  engine = "htmlwidget")
```