Market Basket Analysis

Langtao Chen

January 12, 2017

Contents

Note: In order to run this demo, the following R packages must be installed in your R environment:

- arules: mining association rules
- magrittr: forward pipe operator
- arulesViz: data visualization of association rules
- RColorBrewer: color palettes for plots

```r
# Clean the environment
rm(list = ls())
# Load the arules package for mining association rules
library(arules) # mining association rules
```

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

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

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

```r
library(magrittr) # forward pipe operator
```

1. Understand Dataset

Suppose we extract transaction-item relationship from a transactional database. The dataset is stored in a csv file. Now we use read.csv() method to read in the raw dataset.

```r
# Read in transaction dataset
df<- read.csv("groceries_raw.csv")
```

```r
# Show the head of the raw dataset
head(df)
```

```
##   TransactionID                Item        ItemCategory1
## 1             1        citrus fruit fruit and vegetables
## 2             1 semi-finished bread       fresh products
## 3             1           margarine       processed food
## 4             1         ready soups       processed food
## 5             2      tropical fruit fruit and vegetables
## 6             2              yogurt       fresh products
##            ItemCategory2
## 1                  fruit
## 2 bread and backed goods
## 3           vinegar/oils
## 4           soups/sauces
## 5                  fruit
## 6          dairy produce
```

The first column indicates the transaction ID. The second column is the item included in the transaction. The third and fourth columns are item categories ate different levels. As the above table shows, there are four items (citrus fruit, semi-finished bread, margarine, and ready soup) in the first transaction.

Let's check the structure of the dataset.

```r
# Show the structure of the dataset
str(df)
```

```
## 'data.frame':    43367 obs. of  4 variables:
##  $ TransactionID: int  1 1 1 1 2 2 2 3 4 4 ...
##  $ Item         : Factor w/ 169 levels "abrasive cleaner",..: 30 133 89 119 158 168 34 167 110 168 .
##  $ ItemCategory1: Factor w/ 10 levels "canned food",..: 5 4 9 9 5 4 3 4 5 4 ...
##  $ ItemCategory2: Factor w/ 55 levels "baby food","bags",..: 25 7 54 49 25 18 15 18 25 18 ...
```

Because the transaction ID should be a categorical variable, we change it as a factor.

```
# Show the structure of the dataset
df$TransactionID <- factor(df$TransactionID)
```

Let's check the structure of the dataset again.

```
# Show the structure of the dataset
str(df)
```

```
## 'data.frame':    43367 obs. of  4 variables:
##  $ TransactionID: Factor w/ 9835 levels "1","2","3","4",..: 1 1 1 1 2 2 2 3 4 4 ...
##  $ Item         : Factor w/ 169 levels "abrasive cleaner",..: 30 133 89 119 158 168 34 167 110 168 .
##  $ ItemCategory1: Factor w/ 10 levels "canned food",..: 5 4 9 9 5 4 3 4 5 4 ...
##  $ ItemCategory2: Factor w/ 55 levels "baby food","bags",..: 25 7 54 49 25 18 15 18 25 18 ...
```

As the data structure shows, the raw dataset contains 9835 transactions containing combination of 169 items. Those items belong to 10 categories at level 1, and 55 categoris at level 2.

```
# Show all level 1 categories
levels(df$ItemCategory1)
```

```
##  [1] "canned food"          "detergent"            "drinks"
##  [4] "fresh products"       "fruit and vegetables" "meat and sausage"
##  [7] "non-food"             "perfumery"            "processed food"
## [10] "snacks and candies"
```

```
# Show all level 2 categories
levels(df$ItemCategory2)
```

```
##  [1] "baby food"                       "bags"
##  [3] "bakery improver"                 "bathroom cleaner"
##  [5] "beef"                            "beer"
##  [7] "bread and backed goods"          "candy"
##  [9] "canned fish"                     "canned fruit/vegetables"
## [11] "cheese"                          "chewing gum"
## [13] "chocolate"                       "cleaner"
## [15] "coffee"                          "condiments"
## [17] "cosmetics"                       "dairy produce"
## [19] "delicatessen"                    "dental care"
## [21] "detergent/softener"              "eggs"
## [23] "fish"                            "frozen foods"
## [25] "fruit"                           "games/books/hobby"
## [27] "garden"                          "hair care"
## [29] "hard drinks"                     "health food"
## [31] "jam/sweet spreads"               "long-life bakery products"
## [33] "meat spreads"                    "non-alc. drinks"
## [35] "non-food house keeping products" "non-food kitchen"
## [37] "packaged fruit/vegetables"       "perfumery"
## [39] "personal hygiene"                "pet food/care"
## [41] "pork"                            "poultry"
## [43] "pudding powder"                  "sausage"
## [45] "seasonal products"               "shelf-stable dairy"
## [47] "snacks"                          "soap"
## [49] "soups/sauces"                    "staple foods"
## [51] "sweetener"                       "tea/cocoa drinks"
## [53] "vegetables"                      "vinegar/oils"
## [55] "wine"
```

2. Assocition Rules at Item Level

The data frame cannot be directly analyzed by the arules package. To do the association rules analysis, we need to convert the dataset to a transactions object. The arules package contains a read.transactions() method to read a transaction data file from disk and creates a transactions object.

```
trans_item <- read.transactions(file = "groceries_raw.csv",
                                format = "single",sep = ",",
                                cols = c(1,2), skip = 1)
```

```
# Check the type of the new dataset
class(trans_item)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```
# Show the structure of the dataset
str(trans_item)
```

```
## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:43367] 29 88 118 132 24 166 29 157 1 28 ...
##   .. .. ..@ p       : int [1:9836] 0 4 6 8 18 22 32 37 47 53 ...
##   .. .. ..@ Dim     : int [1:2] 169 9835
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  169 obs. of  1 variable:
##   .. ..$ labels: chr [1:169] "abrasive cleaner" "artif. sweetener" "baby cosmetics" "baby food" ...
##   ..@ itemsetInfo:'data.frame':  9835 obs. of  1 variable:
##   .. ..$ transactionID: chr [1:9835] "1" "10" "100" "1000" ...
```

```
# Show a summay of the transactions dataset
trans_item
```

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)
```

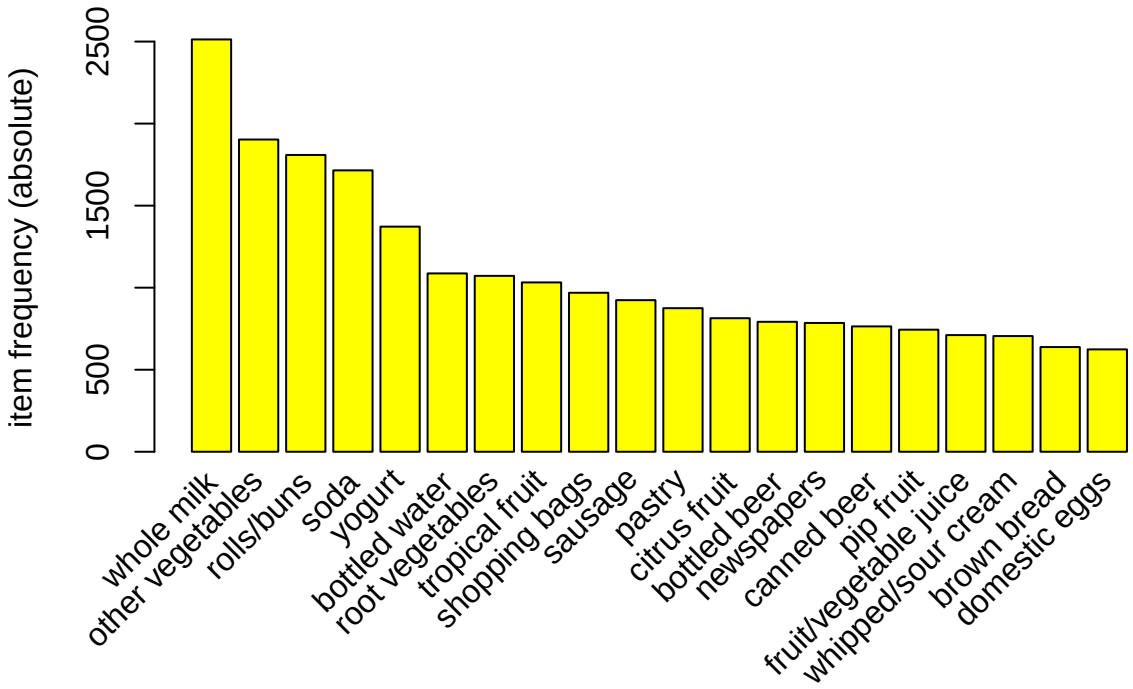
```
# Show the dimensions of the transactions object
dim(trans_item)
```

```
## [1] 9835  169
```

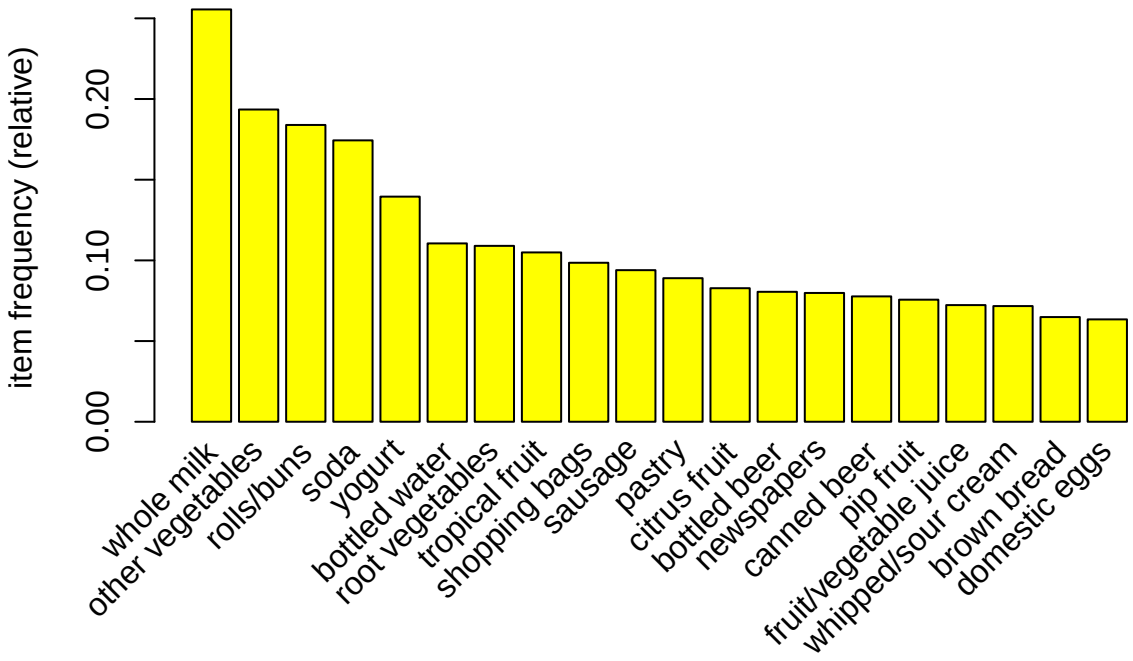
That is to say, the dataset contains 9835 market baskets of 169 SKUs.

Because there are 169 items in the dataset, some items are frequently bought and some are not. Now, let's check the top 20 items that are bought in those transactions.

```
# Drawitem frequency plot, using absolute frequency
itemFrequencyPlot(trans_item,topN=20,col="yellow",type="absolute")
```

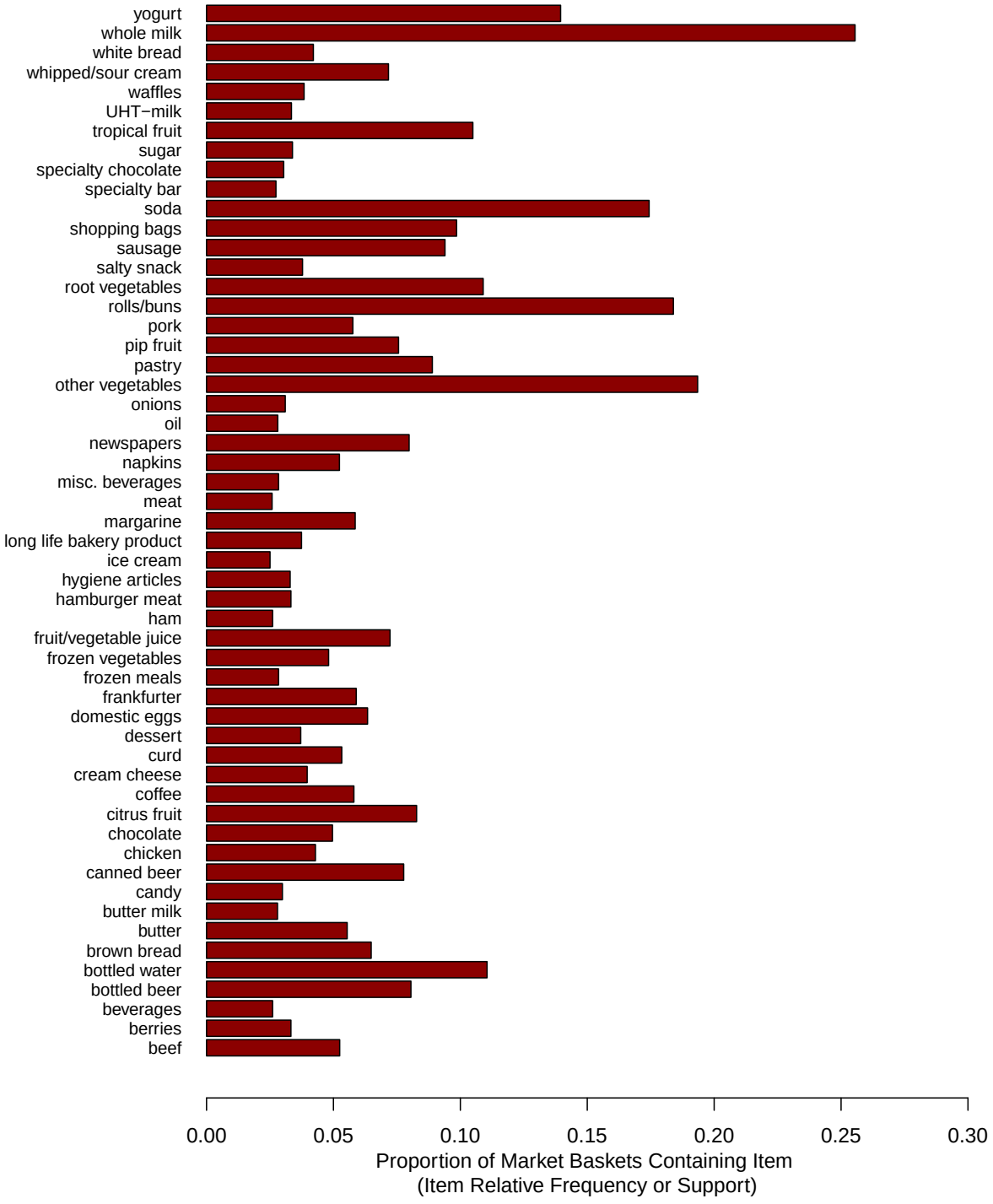



```
# Drawitem frequency plot, using relative frequency
itemFrequencyPlot(trans_item,topN=20,col="yellow",type="relative")
```



We can draw all frequently bought items whose support >= 0.025.

```
# Examine frequency for each item with support greater than 0.025
itemFrequencyPlot(trans_item, support = 0.025, cex.names=0.8, xlim = c(0,0.3),
  type = "relative", horiz = TRUE, col = "dark red", las = 1,
  xlab = paste("Proportion of Market Baskets Containing Item",
    "\n(Item Relative Frequency or Support)"))
```



There are similar items, such as “candy” and “specialy” bar, “chocolate” and “specialty chocolate”. To analyze association rules at the SKU level may loss some general meanings in terms what things are bought tegother by customer. In this case, we choose to analyze association at item category level.

3. Assocition Rules at Item Category Level

Now, we choose the item category 2 as the level of items. We need to construct another transactions object. Again, we use the read.transactions() method in arules package to read the transaction data file from disk and creates a transactions object at the item category 2 level. Notice that the item category is stored in the 4th column. We need to use "rm.duplicates = TRUE" to remove duplicates since the transactin raw dataset contains two or more items in a transaction that belong to the same category 2.

```
trans_cat2<- read.transactions(file = "groceries_raw.csv",
                               format = "single", sep = ",",rm.duplicates = TRUE,
                               cols = c(1,4), skip = 1)
```

```
## distribution of transactions with duplicates:
## items
##    1    2    3    4    5    6    7    8    9   10   11   12   15
## 1879  774  371  158   94   49   16    9    6    2    3    2    1
```

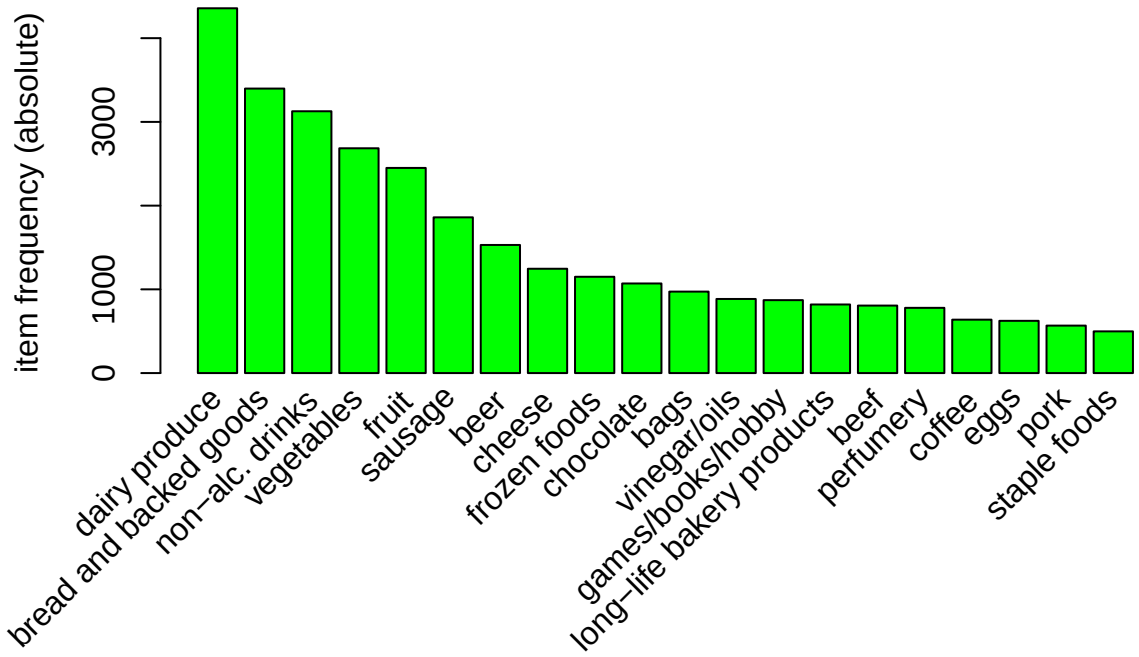
```
# Show a summay of the transactions dataset
trans_cat2
```

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  55 items (columns)
```

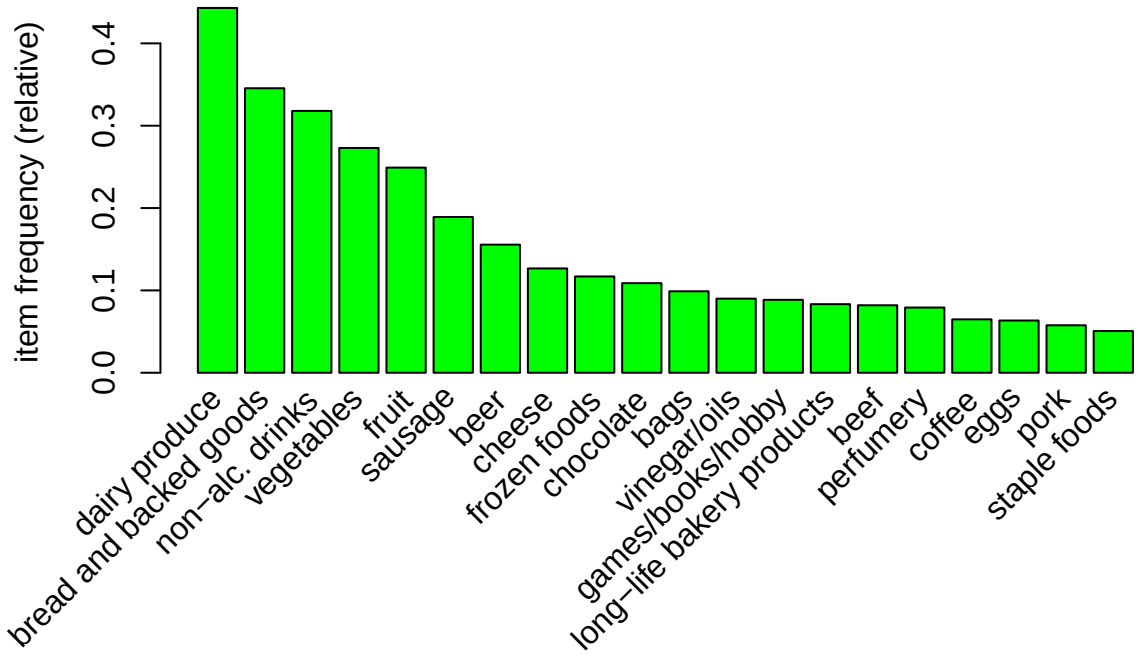
We can see that the transactions dataset contains 9835 transactions of 55 item categories.

Now, let's check the top 20 item categories that are bought in those transactions.

```
itemFrequencyPlot(trans_cat2,topN=20,col="green",type="absolute")
```

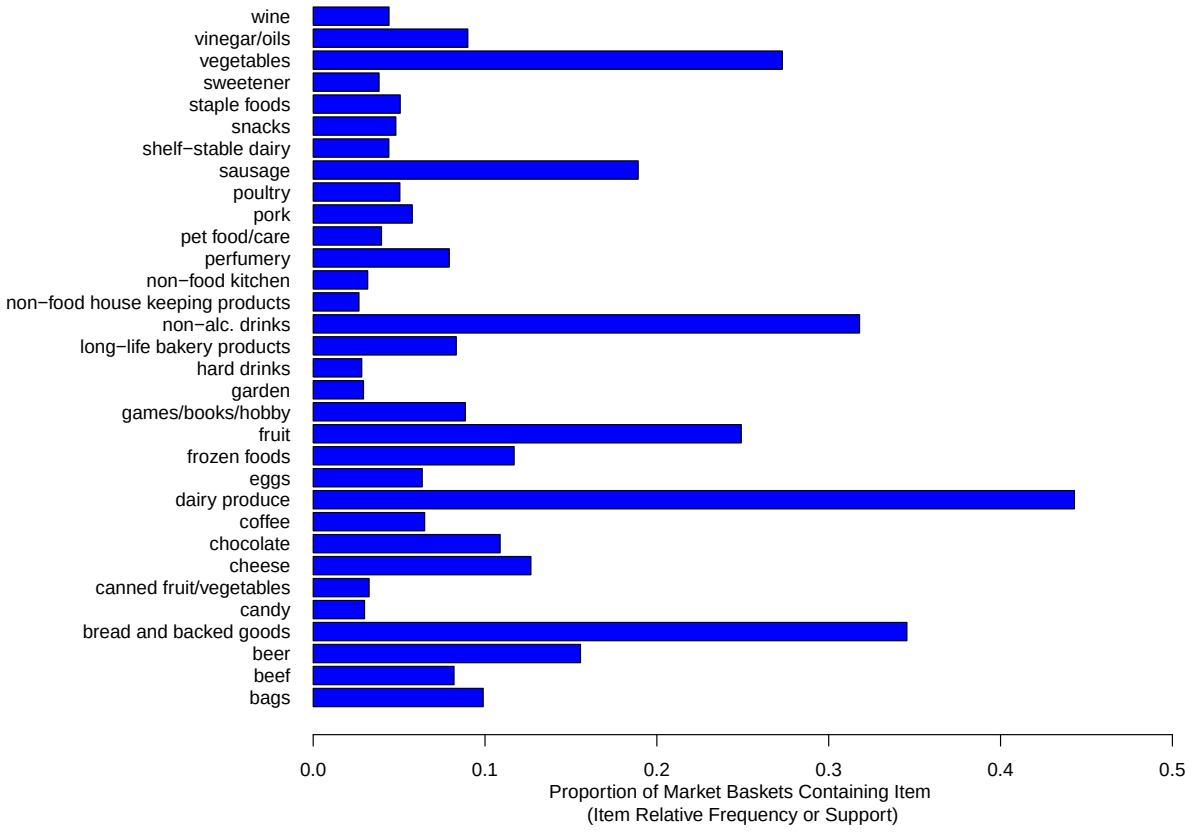


```
itemFrequencyPlot(trans_cat2,topN=20,col="green",type="relative")
```



Draw all frequently bought item categories whose support >= 0.025.

```
itemFrequencyPlot(trans_cat2, support = 0.025, cex.names=1, xlim = c(0,0.5),
  type = "relative", horiz = TRUE, col = "blue", las = 1,
  xlab = paste("Proportion of Market Baskets Containing Item",
    "\n(Item Relative Frequency or Support)"))
```



Now, let's call the apriori() method to generate association rules. We set the minimun support as 0.001 and the minimum confidence as 0.05.

```
# Mine frequent itemsets, association rules or association hyperedges using the Apriori algorithm.
first.rules <- apriori(trans_cat2,
                       parameter = list(support = 0.001, confidence = 0.05))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.001      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: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[55 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [54 item(s)] done [0.00s].
```

```
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.01s].
## writing ... [69921 rule(s)] done [0.02s].
## creating S4 object  ... done [0.01s].
```

Show summary of the 1st set of association rules.

```
summary(first.rules)
```

```
## set of 69921 rules
##
## rule length distribution (lhs + rhs):sizes
##     1     2     3     4     5     6     7     8
##    21  1205 10467 23895 22560  9888  1813    72
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   4.000   4.502   5.000   8.000
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.001017   Min.   :0.0500   Min.   : 0.4475
##  1st Qu.:0.001118   1st Qu.:0.2110   1st Qu.: 1.8315
##  Median :0.001525   Median :0.4231   Median : 2.2573
##  Mean   :0.002488   Mean   :0.4364   Mean   : 2.5382
##  3rd Qu.:0.002339   3rd Qu.:0.6269   3rd Qu.: 2.9662
##  Max.   :0.443010   Max.   :1.0000   Max.   :16.1760
##
## mining info:
##        data ntransactions support confidence
##  trans_cat2          9835   0.001       0.05
```

We notice that the Apriori algorithm detects 69,921 rules from the dataset by using the parameters (minimun support=0.001, minimum confidence=0.05). The rule set is too manay to analyze.

In order to reduce the number of association rules generated, we can enlarge the minimum support and confidence setting. Now, let's set minimun support=0.025 and keep minimum confidence=0.05 and call call apriori() method again.

```
# Mine frequent itemsets, association rules or association hyperedges using the Apriori algorithm.
second.rules <- apriori(trans_cat2,
                        parameter = list(support = 0.025, confidence = 0.05))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.025      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: 245
##
## set item appearances ...[0 item(s)] done [0.00s].
```

```
## set transactions ...[55 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [32 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [344 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Show summary of the association rule
summary(second.rules)
```

```
## set of 344 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4
##  21 162 129  32
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.0     2.0     2.5     3.0     4.0
##
## summary of quality measures:
##     support          confidence          lift
##  Min.   :0.02542   Min.   :0.05043   Min.   :0.6669
##  1st Qu.:0.03030   1st Qu.:0.18202   1st Qu.:1.2498
##  Median :0.03854   Median :0.39522   Median :1.4770
##  Mean   :0.05276   Mean   :0.37658   Mean   :1.4831
##  3rd Qu.:0.05236   3rd Qu.:0.51271   3rd Qu.:1.7094
##  Max.   :0.44301   Max.   :0.79841   Max.   :2.4073
##
## mining info:
##        data ntransactions support confidence
##  trans_cat2          9835   0.025       0.05
```

Now, we get 344 rules, much less than the 1st set of 69,921 rules.

A picture is worth a thousand words. We can visualize the association rules. To do that, we first need to load two packages: "arulesViz"" for association rules plot, and "RColorBrewer" for generating color palettes for graphs.

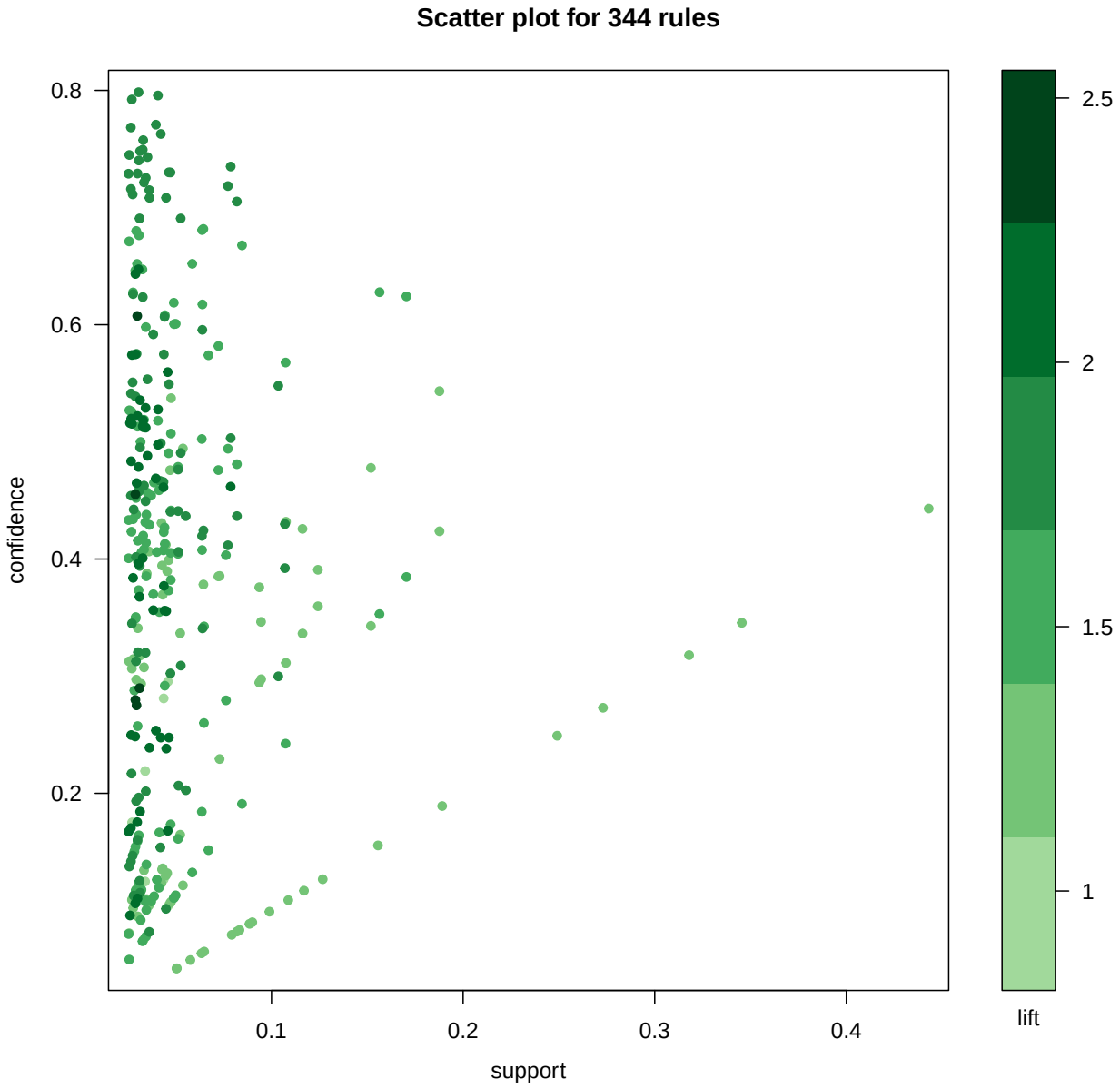
```
library(arulesViz)  # data visualization of association rules
```

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

```
library(RColorBrewer)  # color palettes for plots
```

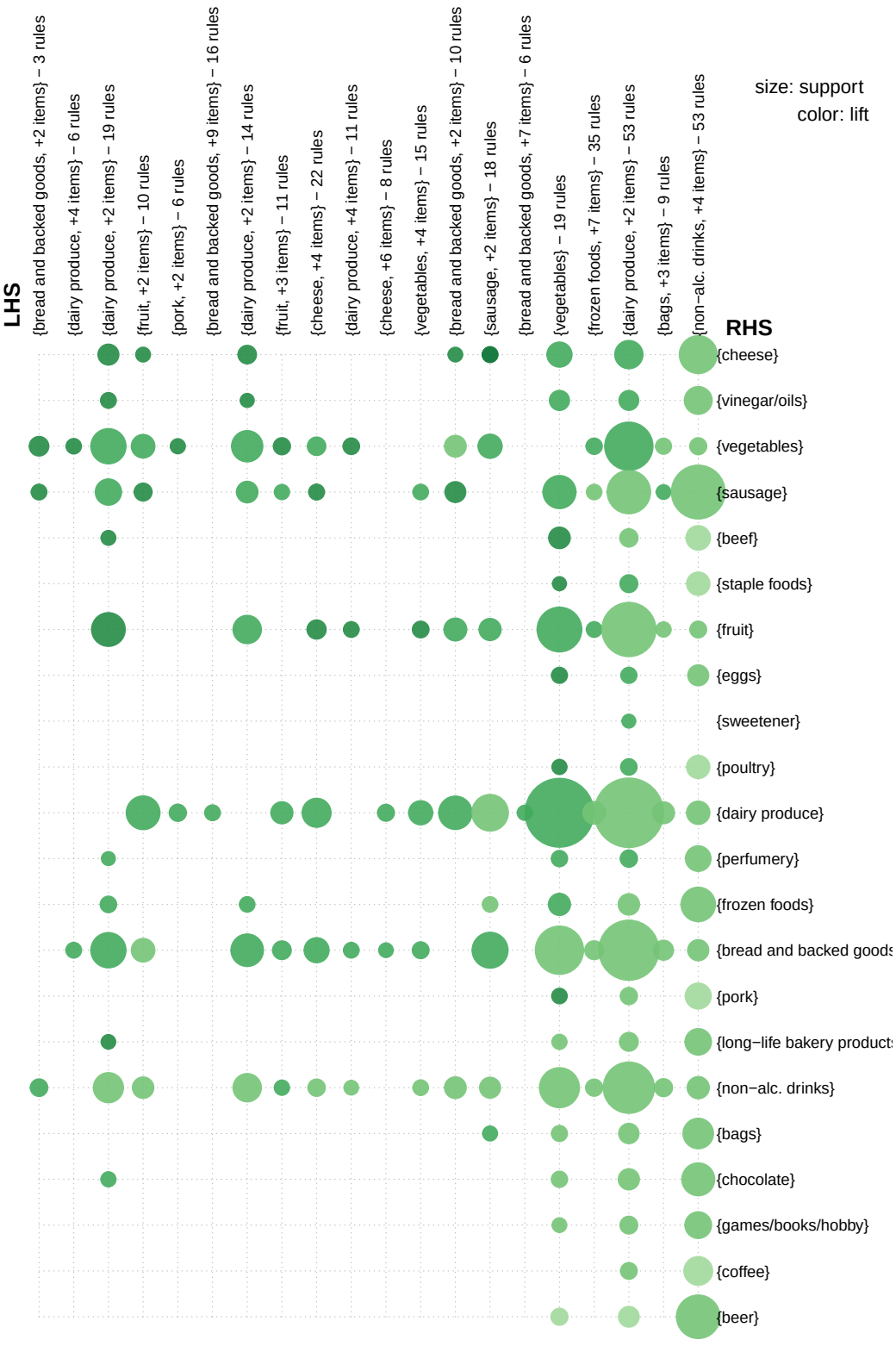
Draw all 344 rules in a scatter plot.

```
# Data visualization of association rules in scatter plot
plot(second.rules,
     control=list(jitter=2, col = rev(brewer.pal(9, "Greens")[4:9])),
     shading = "lift")
```



Grouped matrix-based visualization of all association rules.

```
# grouped matrix of rules
plot(second.rules,
     method="grouped",
     control=list(col = rev(brewer.pal(9, "Greens")[4:9]), main = ""))
```


LHS
{bread and backed goods, +2 items} – 3 rules
{dairy produce, +4 items} – 6 rules
{dairy produce, +2 items} – 19 rules
{fruit, +2 items} – 10 rules
{pork, +2 items} – 6 rules
{bread and backed goods, +9 items} – 16 rules
{dairy produce, +2 items} – 14 rules
{fruit, +3 items} – 11 rules
{cheese, +4 items} – 22 rules
{dairy produce, +4 items} – 11 rules
{cheese, +6 items} – 8 rules
{vegetables, +4 items} – 15 rules
{bread and backed goods, +2 items} – 10 rules
{sausage, +2 items} – 18 rules
{bread and backed goods, +7 items} – 6 rules
{vegetables} – 19 rules
{frozen foods, +7 items} – 35 rules
{dairy produce, +2 items} – 53 rules
{bags, +3 items} – 9 rules
{non−alc. drinks, +4 items} – 53 rules
size: support
color: lift
RHS
{cheese}
{vinegar/oils}
{vegetables}
{sausage}
{beef}
{staple foods}
{fruit}
{eggs}
{sweetener}
{poultry}
{dairy produce}
{perfumery}
{frozen foods}
{bread and backed goods
{pork}
{long−life bakery product
{non−alc. drinks}
{bags}
{chocolate}
{games/books/hobby}
{coffee}
{beer}

Inspect the first 40 rules of the 344 rules.

```
inspect(second.rules[1:40])
```

```
##      lhs                rhs                             support    confidence
## [1]  {}              => {poultry}                       0.05043213 0.05043213
## [2]  {}              => {pork}                          0.05765125 0.05765125
## [3]  {}              => {staple foods}                  0.05063549 0.05063549
## [4]  {}              => {coffee}                        0.06487036 0.06487036
## [5]  {}              => {eggs}                          0.06344687 0.06344687
## [6]  {}              => {games/books/hobby}             0.08856126 0.08856126
## [7]  {}              => {long-life bakery products}     0.08327402 0.08327402
## [8]  {}              => {perfumery}                     0.07920691 0.07920691
## [9]  {}              => {beef}                          0.08195221 0.08195221
## [10] {}              => {bags}                          0.09893238 0.09893238
## [11] {}              => {vinegar/oils}                  0.08998475 0.08998475
## [12] {}              => {chocolate}                     0.10879512 0.10879512
## [13] {}              => {beer}                          0.15556685 0.15556685
## [14] {}              => {frozen foods}                  0.11692933 0.11692933
## [15] {}              => {cheese}                        0.12669039 0.12669039
## [16] {}              => {sausage}                       0.18912049 0.18912049
## [17] {}              => {fruit}                         0.24911032 0.24911032
## [18] {}              => {non-alc. drinks}               0.31794611 0.31794611
## [19] {}              => {vegetables}                    0.27300458 0.27300458
## [20] {}              => {bread and backed goods}        0.34550076 0.34550076
## [21] {}              => {dairy produce}                 0.44300966 0.44300966
## [22] {sweetener}     => {dairy produce}                 0.02572445 0.67108753
## [23] {dairy produce} => {sweetener}                     0.02572445 0.05806748
## [24] {poultry}       => {vegetables}                    0.02897814 0.57459677
## [25] {vegetables}    => {poultry}                       0.02897814 0.10614525
## [26] {poultry}       => {dairy produce}                 0.03263854 0.64717742
## [27] {dairy produce} => {poultry}                       0.03263854 0.07367455
## [28] {pork}          => {vegetables}                    0.03009659 0.52204586
## [29] {vegetables}    => {pork}                          0.03009659 0.11024209
## [30] {pork}          => {dairy produce}                 0.03446873 0.59788360
## [31] {dairy produce} => {pork}                          0.03446873 0.07780583
## [32] {staple foods}  => {vegetables}                    0.02613116 0.51606426
## [33] {vegetables}    => {staple foods}                  0.02613116 0.09571695
## [34] {staple foods}  => {dairy produce}                 0.03619725 0.71485944
## [35] {dairy produce} => {staple foods}                  0.03619725 0.08170760
## [36] {coffee}        => {dairy produce}                 0.03324860 0.51253918
## [37] {dairy produce} => {coffee}                        0.03324860 0.07505164
## [38] {eggs}          => {fruit}                         0.02806304 0.44230769
## [39] {fruit}         => {eggs}                          0.02806304 0.11265306
## [40] {eggs}          => {non-alc. drinks}               0.02541942 0.40064103
##      lift
## [1]  1.000000
## [2]  1.000000
## [3]  1.000000
## [4]  1.000000
## [5]  1.000000
## [6]  1.000000
## [7]  1.000000
## [8]  1.000000
## [9]  1.000000
```

```
## [10] 1.000000
## [11] 1.000000
## [12] 1.000000
## [13] 1.000000
## [14] 1.000000
## [15] 1.000000
## [16] 1.000000
## [17] 1.000000
## [18] 1.000000
## [19] 1.000000
## [20] 1.000000
## [21] 1.000000
## [22] 1.514837
## [23] 1.514837
## [24] 2.104715
## [25] 2.104715
## [26] 1.460865
## [27] 1.460865
## [28] 1.912224
## [29] 1.912224
## [30] 1.349595
## [31] 1.349595
## [32] 1.890314
## [33] 1.890314
## [34] 1.613643
## [35] 1.613643
## [36] 1.156948
## [37] 1.156948
## [38] 1.775549
## [39] 1.775549
## [40] 1.260091
```

Suppose we want to identify products that are commonly purchased along with vegetables. We can use subset() method to selecting rules that satisfy certain criteria.

```
# Select rules with vegetables in consequent (right-hand-side) item subsets
vegie.rules <- subset(second.rules, subset = rhs %pin% "vegetables")
inspect(vegie.rules)  # 41 rules
```

```
##      lhs                            rhs             support confidence      lift
## [1]  {}                          => {vegetables} 0.27300458  0.2730046 1.0000000
## [2]  {poultry}                   => {vegetables} 0.02897814  0.5745968 2.1047148
## [3]  {pork}                      => {vegetables} 0.03009659  0.5220459 1.9122238
## [4]  {staple foods}              => {vegetables} 0.02613116  0.5160643 1.8903136
## [5]  {eggs}                      => {vegetables} 0.03141840  0.4951923 1.8138608
## [6]  {games/books/hobby}         => {vegetables} 0.02785968  0.3145809 1.1522918
## [7]  {long-life bakery products} => {vegetables} 0.02907982  0.3492063 1.2791227
## [8]  {perfumery}                 => {vegetables} 0.03213015  0.4056483 1.4858662
## [9]  {beef}                      => {vegetables} 0.04585663  0.5595533 2.0496116
## [10] {bags}                      => {vegetables} 0.03141840  0.3175745 1.1632571
## [11] {vinegar/oils}              => {vegetables} 0.04199288  0.4666667 1.7093731
## [12] {chocolate}                 => {vegetables} 0.03192679  0.2934579 1.0749195
## [13] {beer}                      => {vegetables} 0.03406202  0.2189542 0.8020168
## [14] {frozen foods}              => {vegetables} 0.04738180  0.4052174 1.4842879
## [15] {cheese}                    => {vegetables} 0.05531266  0.4365971 1.5992300
```

```
## [16] {sausage}                  => {vegetables} 0.07625826  0.4032258 1.4769929
## [17] {fruit}                    => {vegetables} 0.10706660  0.4297959 1.5743176
## [18] {non-alc. drinks}          => {vegetables} 0.09456024  0.2974097 1.0893944
## [19] {bread and backed goods}   => {vegetables} 0.11621759  0.3363743 1.2321198
## [20] {dairy produce}            => {vegetables} 0.17041179  0.3846683 1.4090180
## [21] {beef,
##       dairy produce}            => {vegetables} 0.02989324  0.6074380 2.2250104
## [22] {dairy produce,
##       vinegar/oils}             => {vegetables} 0.03141840  0.5355286 1.9616103
## [23] {dairy produce,
##       frozen foods}             => {vegetables} 0.03436706  0.5121212 1.8758704
## [24] {cheese,
##       fruit}                    => {vegetables} 0.02674123  0.5197628 1.9038613
## [25] {bread and backed goods,
##       cheese}                   => {vegetables} 0.02887646  0.4536741 1.6617821
## [26] {cheese,
##       dairy produce}            => {vegetables} 0.04219624  0.4987981 1.8270686
## [27] {fruit,
##       sausage}                  => {vegetables} 0.03426538  0.5290424 1.9378517
## [28] {non-alc. drinks,
##       sausage}                  => {vegetables} 0.03029995  0.4156206 1.5223944
## [29] {bread and backed goods,
##       sausage}                  => {vegetables} 0.04382308  0.4229637 1.5492916
## [30] {dairy produce,
##       sausage}                  => {vegetables} 0.05266904  0.4905303 1.7967842
## [31] {fruit,
##       non-alc. drinks}          => {vegetables} 0.04361973  0.4657980 1.7061914
## [32] {bread and backed goods,
##       fruit}                    => {vegetables} 0.05124555  0.4763705 1.7449177
## [33] {dairy produce,
##       fruit}                    => {vegetables} 0.07869853  0.5032510 1.8433793
## [34] {bread and backed goods,
##       non-alc. drinks}          => {vegetables} 0.04636502  0.3731588 1.3668590
## [35] {dairy produce,
##       non-alc. drinks}          => {vegetables} 0.06446365  0.4243641 1.5544213
## [36] {bread and backed goods,
##       dairy produce}            => {vegetables} 0.08195221  0.4366197 1.5993128
## [37] {dairy produce,
##       fruit,
##       sausage}                  => {vegetables} 0.02714794  0.5741935 2.1032378
## [38] {bread and backed goods,
##       dairy produce,
##       sausage}                  => {vegetables} 0.03284189  0.5135135 1.8809704
## [39] {dairy produce,
##       fruit,
##       non-alc. drinks}          => {vegetables} 0.03304525  0.5183413 1.8986543
## [40] {bread and backed goods,
##       dairy produce,
##       fruit}                    => {vegetables} 0.04077275  0.5276316 1.9326840
## [41] {bread and backed goods,
##       dairy produce,
##       non-alc. drinks}          => {vegetables} 0.03345196  0.4627286 1.6949480
```

We get 41 rules whose right hand side is “vegetables”. The 41 rules are still too many to analyze. We choose

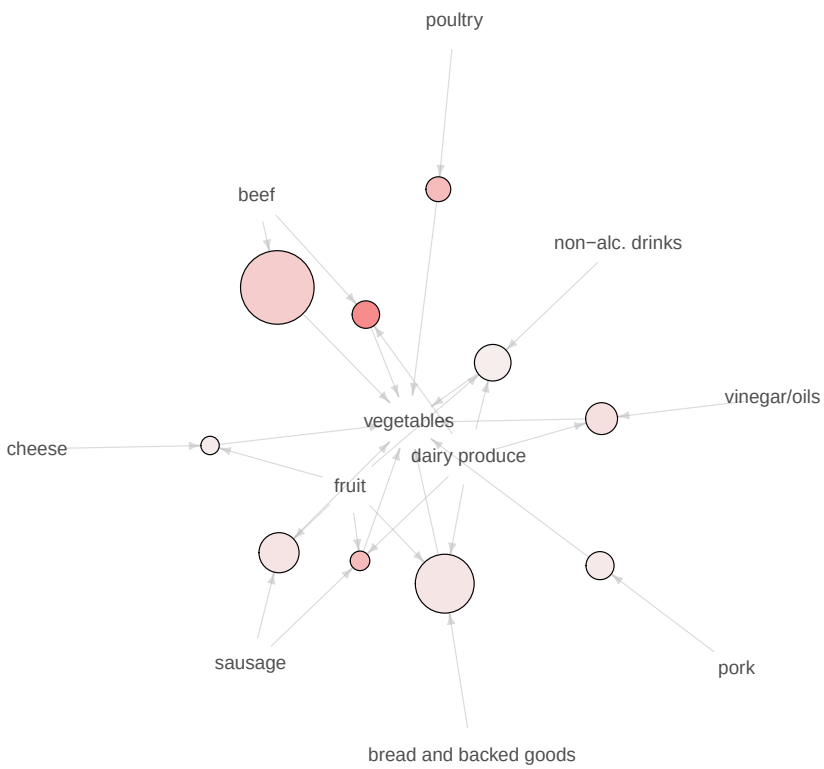
to get the top 10 rules with highest lift from the 41 set.

```
# Sort by lift and identify the top 10 rules
top.vegie.rules <- vegie.rules %>% sort(decreasing = TRUE, by = "lift") %>% head(10)
inspect(top.vegie.rules)
```

```
##      lhs                          rhs             support confidence     lift
## [1]  {beef,
##       dairy produce}          => {vegetables} 0.02989324  0.6074380 2.225010
## [2]  {poultry}               => {vegetables} 0.02897814  0.5745968 2.104715
## [3]  {dairy produce,
##       fruit,
##       sausage}               => {vegetables} 0.02714794  0.5741935 2.103238
## [4]  {beef}                  => {vegetables} 0.04585663  0.5595533 2.049612
## [5]  {dairy produce,
##       vinegar/oils}          => {vegetables} 0.03141840  0.5355286 1.961610
## [6]  {fruit,
##       sausage}               => {vegetables} 0.03426538  0.5290424 1.937852
## [7]  {bread and backed goods,
##       dairy produce,
##       fruit}                 => {vegetables} 0.04077275  0.5276316 1.932684
## [8]  {pork}                  => {vegetables} 0.03009659  0.5220459 1.912224
## [9]  {cheese,
##       fruit}                 => {vegetables} 0.02674123  0.5197628 1.903861
## [10] {dairy produce,
##       fruit,
##       non-alc. drinks}       => {vegetables} 0.03304525  0.5183413 1.898654
```

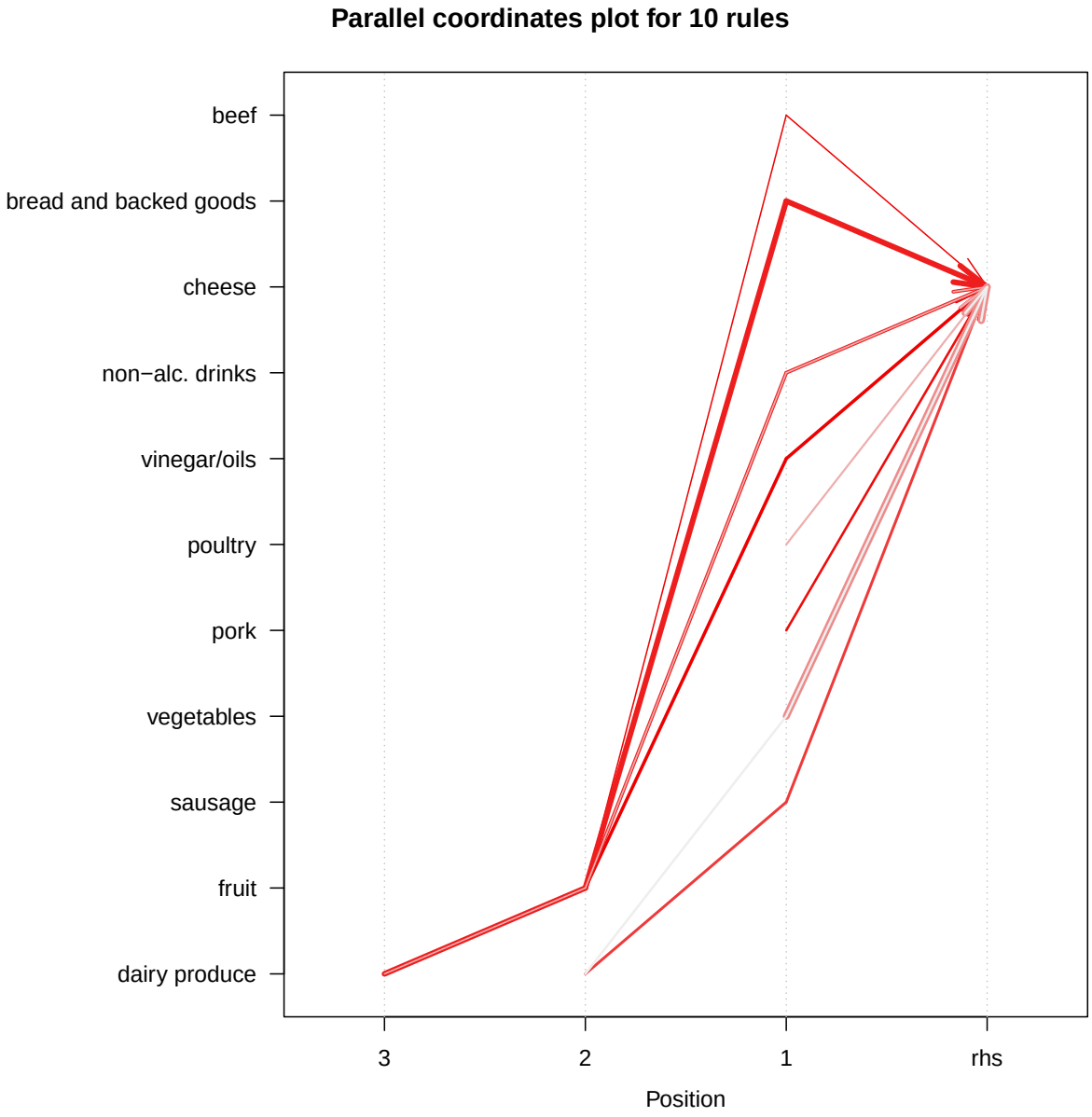
Draw the graph of the 10 association rules.

```
plot(top.vegie.rules, method="graph",
  control=list(type="items"),
  shading = "lift")
```



Represents the rules (or itemsets) as a parallel coordinate plot.

```
plot(top.vegie.rules, method="paracoord", control=list(reorder=TRUE))
```


Parallel coordinates plot for 10 rules
beef
bread and backed goods
cheese
non−alc. drinks
vinegar/oils
poultry
pork
vegetables
sausage
fruit
dairy produce
3
2
1
rhs
Position