```
library(tidyverse)

## ── Attaching packages ────────────────────────────────────────────────────────────────────────────────────────────────────────────────
tidyverse 1.2.1 ──

## ✓ ggplot2 3.2.0     ✓ purrr   0.3.2
## ✓ tibble  2.1.3     ✓ dplyr   0.8.3
## ✓ tidyr   0.8.3     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0

## ── Conflicts ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
─── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()

library(arules)  # has a big ecosystem of packages built around it

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

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

library(arulesViz)

## Loading required package: grid

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

Read in the groceries transactions This is in “long” format – every row is a basket with multiple items per row separated by commas. We separated the items in each basket by commas

```
groceries_raw = read.transactions("groceries.txt", sep=',')
```

Checking if our items are separated correctly

```
inspect(head(groceries_raw, 2))
```

```
##     items                 
## [1] {citrus fruit,        
##      margarine,           
##      ready soups,         
##      semi-finished bread} 
## [2] {coffee,              
##      tropical fruit,      
##      yogurt}
```

```
LIST(head(groceries_raw, 3))
```

```
## [[1]]
## [1] "citrus fruit"        "margarine"           "ready soups"        
## [4] "semi-finished bread"
## 
## [[2]]
## [1] "coffee"         "tropical fruit" "yogurt"        
## 
## [[3]]
## [1] "whole milk"
```

Checking if our items are separated correctly

```
inspect(head(groceries_raw, 2))
```

```
##     items                 
## [1] {citrus fruit,        
##      margarine,           
##      ready soups,         
##      semi-finished bread} 
## [2] {coffee,              
##      tropical fruit,      
##      yogurt}
```

```
LIST(head(groceries_raw, 3))
```

```
## [[1]]
## [1] "citrus fruit"        "margarine"           "ready soups"        
## [4] "semi-finished bread"
## 
## [[2]]
## [1] "coffee"         "tropical fruit" "yogurt"        
## 
## [[3]]
## [1] "whole milk"
```

```
str(groceries_raw)
```

```
## 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 33 157 167 166 38 91 ...
##   .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
##   .. .. ..@ 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':  0 obs. of  0 variables
```

```
summary(groceries_raw)
```

```
## 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
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   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
```

Counting the frequencies of each of the grocery items and plotting the top 20 items that appear most frequently.

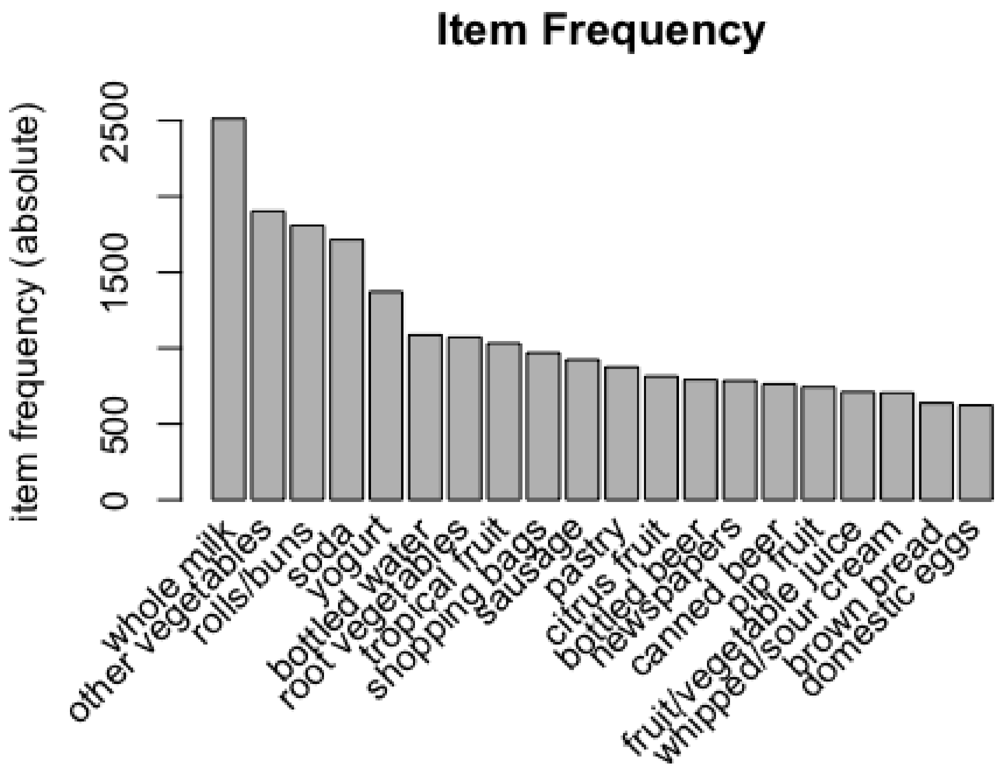
```
frequent = eclat(groceries_raw, parameter = list(supp = 0.07, maxlen = 15))
```

```
## 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: 688
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating sparse bit matrix ... [18 row(s), 9835 column(s)] done [0.00s].
## writing  ... [19 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
inspect(frequent)
```

```
##      items                        support    count
## [1]  {other vegetables,whole milk} 0.07483477  736
## [2]  {whole milk}                 0.25551601 2513
## [3]  {other vegetables}           0.19349263 1903
## [4]  {rolls/buns}                 0.18393493 1809
## [5]  {yogurt}                     0.13950178 1372
## [6]  {soda}                       0.17437722 1715
## [7]  {root vegetables}            0.10899847 1072
## [8]  {tropical fruit}             0.10493137 1032
## [9]  {bottled water}              0.11052364 1087
## [10] {sausage}                    0.09395018  924
## [11] {shopping bags}              0.09852567  969
## [12] {citrus fruit}               0.08276563  814
## [13] {pastry}                     0.08896797  875
## [14] {pip fruit}                  0.07564820  744
## [15] {whipped/sour cream}         0.07168277  705
## [16] {fruit/vegetable juice}      0.07229283  711
## [17] {newspapers}                 0.07981698  785
## [18] {bottled beer}               0.08052872  792
## [19] {canned beer}                0.07768175  764
```

```
itemFrequencyPlot(groceries_raw, topN=20, type="absolute", main="Item
Frequency")
```



Creating a list of baskets: vectors of items by consumer Analagous to bags of words

Cast this variable as a special arules "transactions" class.

```
grotrans = as(groceries_raw, "transactions")
summary(grotrans)

## 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
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   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
```

Running the ‘apriori’ algorithm Looking at rules with support > .005 & confidence >.1 & length (# artists) <= 5

```
shoppingrules = apriori(grotrans,
                     parameter=list(support=.005, confidence=.1, maxlen=5))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  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: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [1582 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Looking at the output

```
#inspect(shoppingrules)
```

Choosing a subset For our thresholds for lift and confidence, we chose confidence to be greater than 0.5 and lift to be greater than 3. When we initially inspected when lift > 4, only four connections were returned. These returns had very low confidence ranging from 0.12 to 0.44, although the lifts were high. This means that these connections should not be correct most of the time. When we inspected for confidence > 0.6, 22 connections were returned. These returns had lifts that were below 3, which means that there is a greater chance of the connections being a coincidence compared to when lift is greater than 3. Thus, when we inspected for our threshold, 8 connections that were high in both confidence and lift were returned. These connections are therefore not just a confidence and they are correct most of the time.

Our choice of threshold is also supported by the Gephi diagram (using Force Atlas). The grocery items that have the most connections are found from the connections returned by applying the threshold.

The results from the table returned and the gephi diagram(uploaded as another file on github) makes sense. For example, it is reasonable to claim that customers purchasing onions and root vegetables also purchase other vegetables. Also, it makes sense for customers purchasing curd and tropical fruits to purchase yogurt (a combination of ingredients people usually have).

```
inspect(subset(shoppingrules, subset=lift > 4))

##     lhs                      rhs                     support confidence
lift count
## [1] {ham}                 => {white bread}        0.005083884  0.1953125
4.639851    50
## [2] {white bread}         => {ham}                0.005083884  0.1207729
4.639851    50
## [3] {butter,
##      other vegetables} => {whipped/sour cream} 0.005795628  0.2893401
4.036397    57
## [4] {citrus fruit,
##      other vegetables,
##      whole milk}          => {root vegetables}    0.005795628  0.4453125
4.085493    57
```

```
inspect(subset(shoppingrules, subset=confidence > 0.6))

##      lhs                           rhs                    support confidence
lift count
## [1]  {onions,
##       root vegetables}          => {other vegetables} 0.005693950  0.6021505
3.112008    56
## [2]  {curd,
##       tropical fruit}           => {whole milk}       0.006507372  0.6336634
2.479936    64
## [3]  {domestic eggs,
##       margarine}                => {whole milk}       0.005185562  0.6219512
2.434099    51
## [4]  {butter,
##       domestic eggs}            => {whole milk}       0.005998983  0.6210526
2.430582    59
## [5]  {butter,
##       whipped/sour cream}       => {whole milk}       0.006710727  0.6600000
2.583008    66
## [6]  {bottled water,
##       butter}                   => {whole milk}       0.005388917  0.6022727
```

```
2.357084    53
## [7]  {butter,
##       tropical fruit}        => {whole milk}       0.006202339  0.6224490
2.436047    61
## [8]  {butter,
##       root vegetables}       => {whole milk}       0.008235892  0.6377953
2.496107    81
## [9]  {butter,
##       yogurt}                => {whole milk}       0.009354347  0.6388889
2.500387    92
## [10] {domestic eggs,
##       pip fruit}             => {whole milk}       0.005388917  0.6235294
2.440275    53
## [11] {domestic eggs,
##       tropical fruit}        => {whole milk}       0.006914082  0.6071429
2.376144    68
## [12] {pip fruit,
##       whipped/sour cream}    => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [13] {pip fruit,
##       whipped/sour cream}    => {whole milk}       0.005998983  0.6483516
2.537421    59
## [14] {fruit/vegetable juice,
##       other vegetables,
##       yogurt}                => {whole milk}       0.005083884  0.6172840
2.415833    50
## [15] {other vegetables,
##       root vegetables,
##       whipped/sour cream}    => {whole milk}       0.005185562  0.6071429
2.376144    51
## [16] {other vegetables,
##       pip fruit,
##       root vegetables}       => {whole milk}       0.005490595  0.6750000
2.641713    54
## [17] {pip fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005490595  0.6136364
3.171368    54
## [18] {other vegetables,
##       pip fruit,
##       yogurt}                => {whole milk}       0.005083884  0.6250000
2.446031    50
## [19] {citrus fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005795628  0.6333333
3.273165    57
## [20] {root vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.005693950  0.7000000
2.739554    56
```

```
## [21] {other vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.007625826  0.6198347
2.425816    75
## [22] {other vegetables,
##       root vegetables,
##       yogurt}                => {whole milk}       0.007829181  0.6062992
2.372842    77
```

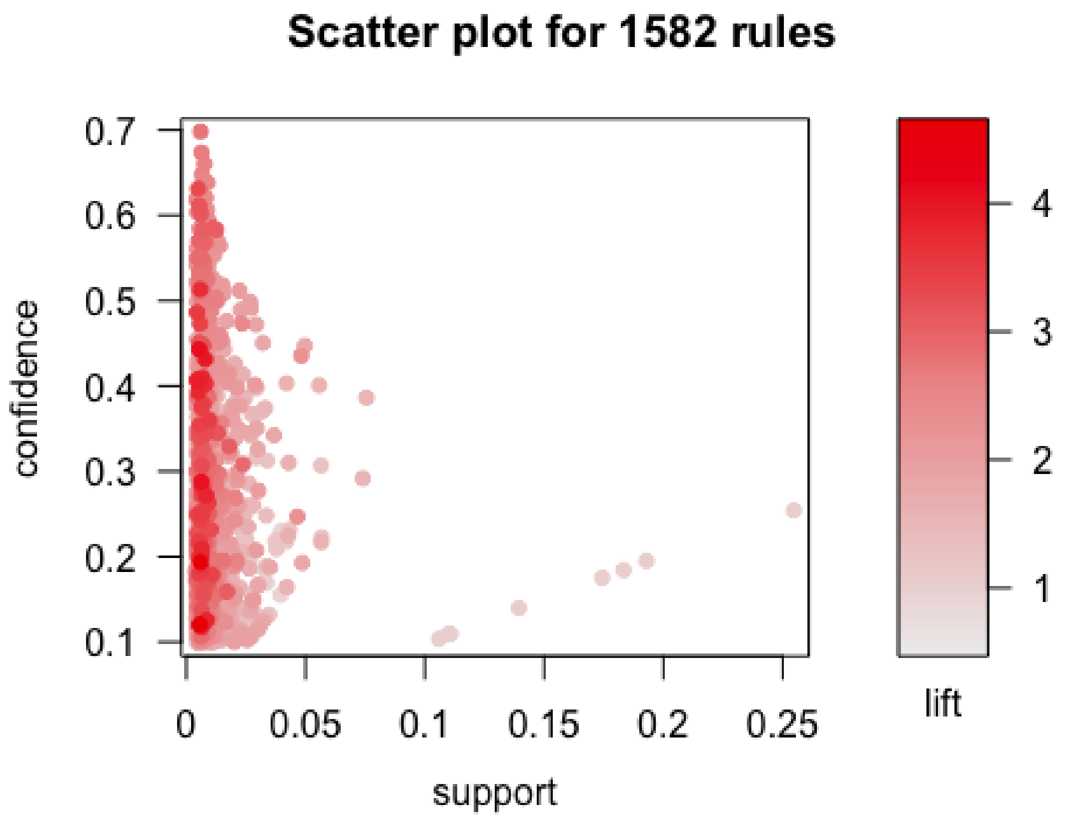
```
inspect(subset(shoppingrules, subset=lift > 3 & confidence > 0.5))
```

```
##     lhs                        rhs                     support confidence
lift count
## [1] {onions,
##      root vegetables}    => {other vegetables} 0.005693950  0.6021505
3.112008    56
## [2] {curd,
##      tropical fruit}     => {yogurt}           0.005287239  0.5148515
3.690645    52
## [3] {pip fruit,
##      whipped/sour cream} => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [4] {citrus fruit,
##      root vegetables}    => {other vegetables} 0.010371124  0.5862069
3.029608   102
## [5] {root vegetables,
##      tropical fruit}     => {other vegetables} 0.012302999  0.5845411
3.020999   121
## [6] {pip fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005490595  0.6136364
3.171368    54
## [7] {citrus fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005795628  0.6333333
3.273165    57
## [8] {root vegetables,
##      tropical fruit,
##      whole milk}         => {other vegetables} 0.007015760  0.5847458
3.022057    69
```

ploting all the rules in the (support, confidence) space Higher lift rules tend to have lower support

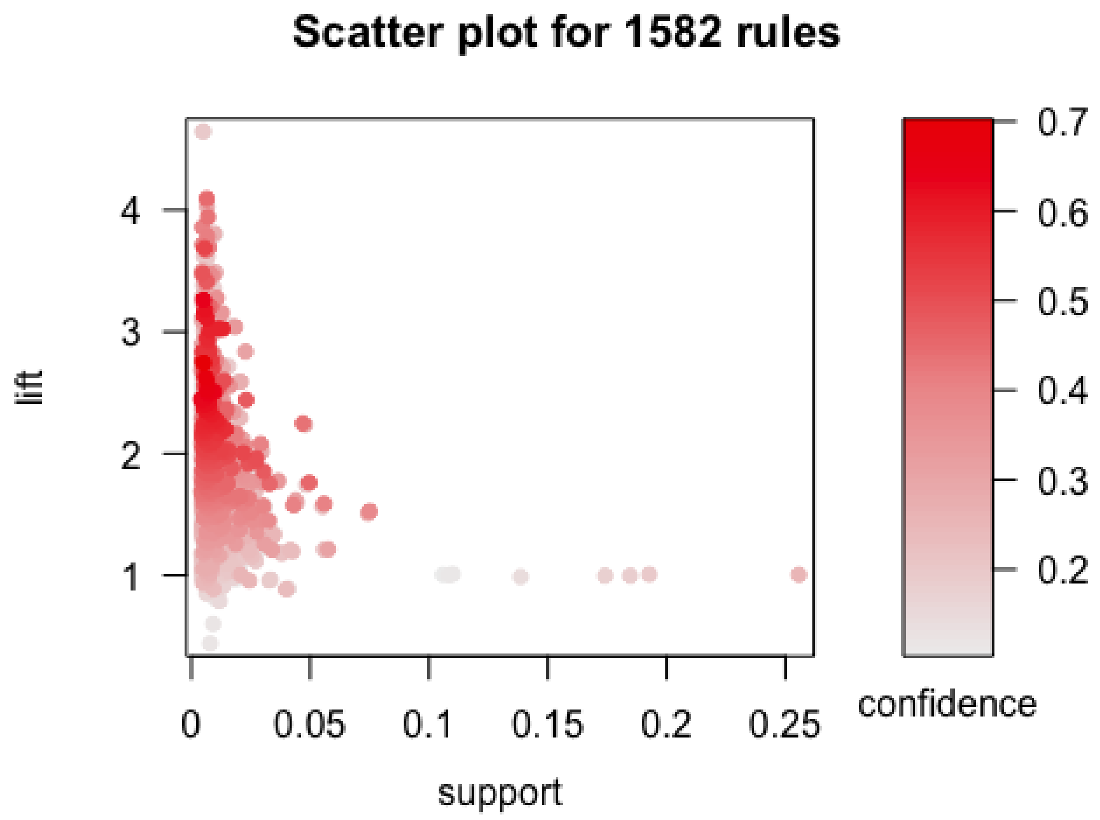
```
plot(shoppingrules)
```

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



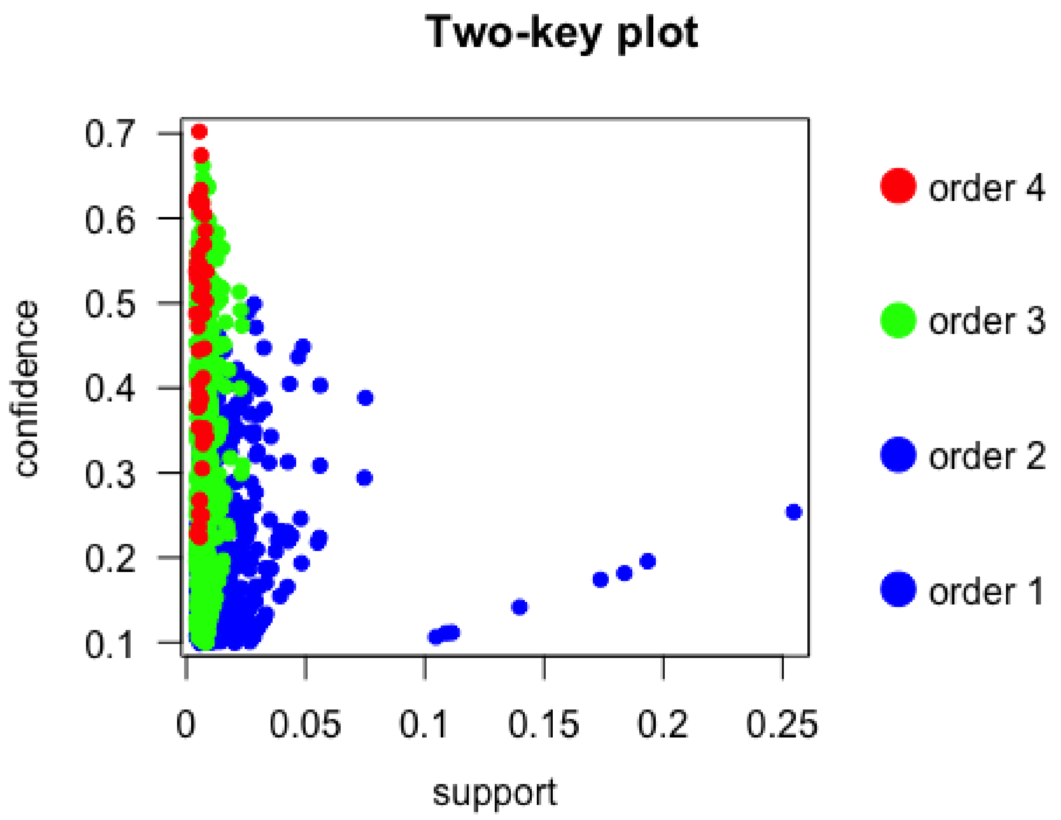
Swapping the axes and color scales

```
plot(shoppingrules, measure = c("support", "lift"), shading = "confidence")

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



“two key” plot: coloring is by size (order) of item set

```
plot(shoppingrules, method='two-key plot')

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



looking at subsets driven by the plot

```
inspect(subset(shoppingrules, support > 0.035))
```

```
##      lhs                    rhs                  support    confidence
## [1]  {}                  => {bottled water}      0.11052364 0.1105236
## [2]  {}                  => {tropical fruit}     0.10493137 0.1049314
## [3]  {}                  => {root vegetables}    0.10899847 0.1089985
## [4]  {}                  => {soda}               0.17437722 0.1743772
## [5]  {}                  => {yogurt}             0.13950178 0.1395018
## [6]  {}                  => {rolls/buns}         0.18393493 0.1839349
## [7]  {}                  => {other vegetables}   0.19349263 0.1934926
## [8]  {}                  => {whole milk}         0.25551601 0.2555160
## [9]  {tropical fruit}    => {other vegetables}   0.03589222 0.3420543
## [10] {other vegetables}  => {tropical fruit}     0.03589222 0.1854966
## [11] {tropical fruit}    => {whole milk}         0.04229792 0.4031008
## [12] {whole milk}        => {tropical fruit}     0.04229792 0.1655392
## [13] {root vegetables}   => {other vegetables}   0.04738180 0.4347015
## [14] {other vegetables}  => {root vegetables}    0.04738180 0.2448765
## [15] {root vegetables}   => {whole milk}         0.04890696 0.4486940
## [16] {whole milk}        => {root vegetables}    0.04890696 0.1914047
## [17] {soda}              => {rolls/buns}         0.03833249 0.2198251
## [18] {rolls/buns}        => {soda}               0.03833249 0.2084024
## [19] {soda}             => {whole milk}       0.04006101 0.2297376
## [20] {whole milk}       => {soda}             0.04006101 0.1567847
## [21] {yogurt}           => {other vegetables} 0.04341637 0.3112245
## [22] {other vegetables} => {yogurt}           0.04341637 0.2243826
## [23] {yogurt}           => {whole milk}       0.05602440 0.4016035
## [24] {whole milk}       => {yogurt}           0.05602440 0.2192598
## [25] {rolls/buns}       => {other vegetables} 0.04260295 0.2316197
## [26] {other vegetables} => {rolls/buns}       0.04260295 0.2201787
## [27] {rolls/buns}       => {whole milk}       0.05663447 0.3079049
## [28] {whole milk}       => {rolls/buns}       0.05663447 0.2216474
## [29] {other vegetables} => {whole milk}       0.07483477 0.3867578
## [30] {whole milk}       => {other vegetables} 0.07483477 0.2928770
##      lift      count
## [1]  1.0000000 1087
## [2]  1.0000000 1032
## [3]  1.0000000 1072
## [4]  1.0000000 1715
## [5]  1.0000000 1372
## [6]  1.0000000 1809
## [7]  1.0000000 1903
## [8]  1.0000000 2513
## [9]  1.7677896  353
## [10] 1.7677896  353
## [11] 1.5775950  416
## [12] 1.5775950  416
## [13] 2.2466049  466
## [14] 2.2466049  466
## [15] 1.7560310  481
## [16] 1.7560310  481
## [17] 1.1951242  377
## [18] 1.1951242  377
## [19] 0.8991124  394
## [20] 0.8991124  394
## [21] 1.6084566  427
## [22] 1.6084566  427
## [23] 1.5717351  551
## [24] 1.5717351  551
## [25] 1.1970465  419
## [26] 1.1970465  419
## [27] 1.2050318  557
## [28] 1.2050318  557
## [29] 1.5136341  736
## [30] 1.5136341  736
```

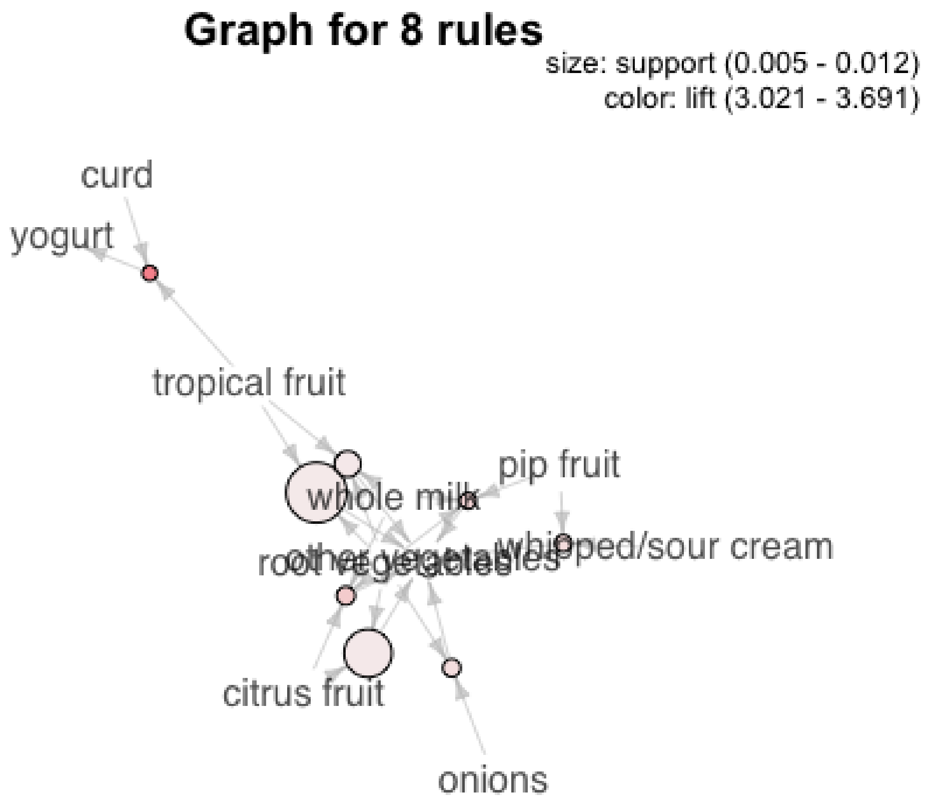
```
inspect(subset(shoppingrules, confidence > 0.5 & lift > 3))
```

```
##     lhs                      rhs                     support confidence
lift count
## [1] {onions,
##      root vegetables}    => {other vegetables} 0.005693950  0.6021505
```

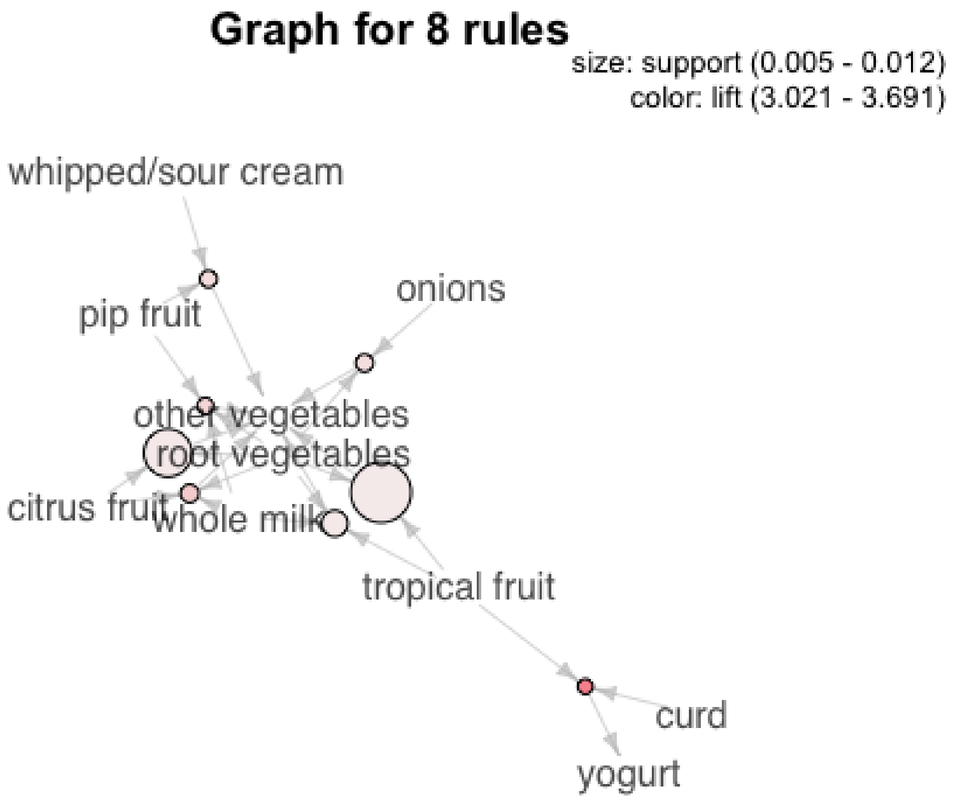
```
3.112008    56
## [2] {curd,
##      tropical fruit}     => {yogurt}           0.005287239  0.5148515
3.690645    52
## [3] {pip fruit,
##      whipped/sour cream} => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [4] {citrus fruit,
##      root vegetables}    => {other vegetables} 0.010371124  0.5862069
3.029608   102
## [5] {root vegetables,
##      tropical fruit}     => {other vegetables} 0.012302999  0.5845411
3.020999   121
## [6] {pip fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005490595  0.6136364
3.171368    54
## [7] {citrus fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005795628  0.6333333
3.273165    57
## [8] {root vegetables,
##      tropical fruit,
##      whole milk}         => {other vegetables} 0.007015760  0.5847458
3.022057    69
```

graph-based visualization For the visualization, we used the threshold that we chose earlier (confidence > 0.5 & lift > 3). After plotting the subset, we are able to clearly see strong connections between the grocery items. We tried using a confidence that was higher, which did not show as many interesting and insightful connections.

```
sub1 = subset(shoppingrules, subset=confidence > 0.5 & lift > 3)
summary(sub1)

## set of 8 rules
##
## rule length distribution (lhs + rhs):sizes
## 3 4
## 5 3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   3.375   4.000   4.000
##
## summary of quality measures:
##     support           confidence          lift           count
##  Min.   :0.005287   Min.   :0.5149   Min.   :3.021   Min.   : 52.00
##  1st Qu.:0.005567   1st Qu.:0.5847   1st Qu.:3.028   1st Qu.: 54.75
##  Median :0.005745   Median :0.5942   Median :3.118   Median : 56.50
##  Mean   :0.007194   Mean   :0.5905   Mean   :3.180   Mean   : 70.75
##  3rd Qu.:0.007855   3rd Qu.:0.6067   3rd Qu.:3.197   3rd Qu.: 77.25
##  Max.   :0.012303   Max.   :0.6333   Max.   :3.691   Max.   :121.00
##
## mining info:
##      data ntransactions support confidence
##  grotrans          9835   0.005        0.1

plot(sub1, method='graph')
```



```
?plot.rules

plot(head(sub1, 100, by='lift'), method='graph')
```



export There is a file on github showing the connections after applying ‘Force Alas’

```
saveAsGraph(head(shoppingrules, n = 1000, by = "lift"), file =
"shoppingrules.graphml")
```