ASSOCIATION ANALYSIS

The dataset of interest here involves transactions at a grocery store (groceries.csv). This example is adapted from the book *Machine Learning in R* by Brett Lantz [1] Each line in the txt file contains a comma separated list of items purchased in a single transaction. There is no other identifying information given about the purchaser of the items. We'll start by reading the data into a transaction dataset for use in the arules package.

```
> library(arules)
> groceries = read.transactions("groceries.csv", sep = ",")
> summary(groceries)
```

```
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
```

The arules library is *very* specific about how the data is read in to R and formatted. The easiest way is to start with a text file contains transactions in a list like this. We can examine the first five transactions as follows:

[1]Machine Learning with R, by Brett Lantz. Packt publishing 2015. Second Edition.

```
> inspect(groceries[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}
```

We can also examine and visualize the frequency of the items.

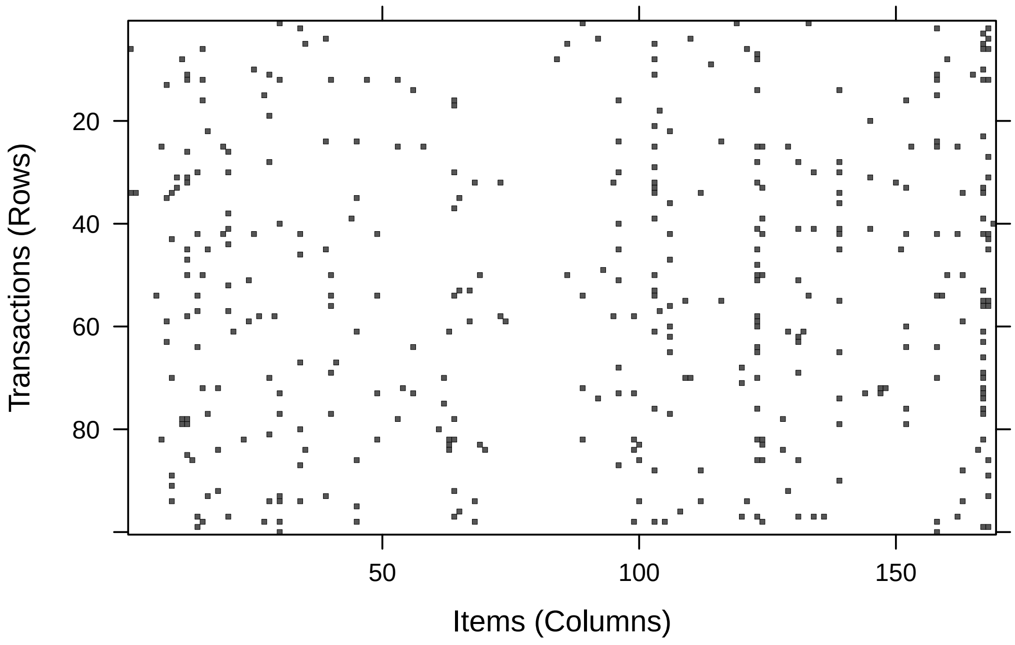
```
> itemFrequency(groceries[, 1:3])
```

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

```
> itemFrequencyPlot(groceries, support = 0.1)
> itemFrequencyPlot(groceries, topN = 20)
```

Can also create this visualization of the sparse matrix for the first hundred transactions.

```
> image(groceries[1:100])
```



The apriori function learns association rules from the transaction data set. Here, the default settings result in zero rules learned:

```
> apriori(groceries)
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.8    0.1    1 none FALSE            TRUE     0.1      1     10  rules
   ext
 FALSE

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

Absolute minimum support count: 983

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
```

```
checking subsets of size 1 2 done [0.00s].
writing ... [0 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
set of 0 rules
```

We'll have to set support and confidence levels to learn more rules:

```
> rules <- apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
       0.25    0.1    1 none FALSE            TRUE   0.006      2     10  rules
   ext
 FALSE

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

Absolute minimum support count: 59

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [109 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [463 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> rules
```

```
set of 463 rules
```

```
> summary(rules)
```

```
set of 463 rules

rule length distribution (lhs + rhs):sizes
  2   3   4
150 297  16

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   3.000   2.711   3.000   4.000

summary of quality measures:
    support           confidence          lift
 Min.   :0.006101   Min.   :0.2500   Min.   :0.9932
 1st Qu.:0.007117   1st Qu.:0.2971   1st Qu.:1.6229
 Median :0.008744   Median :0.3554   Median :1.9332
 Mean   :0.011539   Mean   :0.3786   Mean   :2.0351
 3rd Qu.:0.012303   3rd Qu.:0.4495   3rd Qu.:2.3565
 Max.   :0.074835   Max.   :0.6600   Max.   :3.9565
```

```
mining info:
      data ntransactions support confidence
 groceries          9835   0.006       0.25
```

Look at the first three rules in the output. How are they sorted?

```
> inspect(rules[1:3])
```

```
  lhs                 rhs                support     confidence lift
1 {potted plants} => {whole milk}      0.006914082 0.4000000  1.565460
2 {pasta}         => {whole milk}      0.006100661 0.4054054  1.586614
3 {herbs}         => {root vegetables} 0.007015760 0.4312500  3.956477
```

Sort the rules by lift:

```
> inspect(sort(rules, by = "lift")[1:5])
```

```
  lhs                    rhs                        support  confidence     lift
1 {herbs}             => {root vegetables}    0.007015760  0.4312500 3.956477
2 {berries}           => {whipped/sour cream} 0.009049314  0.2721713 3.796886
3 {other vegetables,
   tropical fruit,
   whole milk}        => {root vegetables}    0.007015760  0.4107143 3.768074
4 {beef,
   other vegetables} => {root vegetables}    0.007930859  0.4020619 3.688692
5 {other vegetables,
   tropical fruit}    => {pip fruit}          0.009456024  0.2634561 3.482649
```

Perhaps you’re interested in finding subsets of rules containing any certain items, for example rules that contain the items “chocolate" or “ice cream":

```
> sweetrules <- subset(rules, items %in% c("chocolate","ice cream"))
> inspect(sweetrules)
```

```
   lhs             rhs                 support    confidence lift
87 {chocolate} => {soda}              0.01352313 0.2725410  1.562939
88 {chocolate} => {other vegetables} 0.01270971 0.2561475  1.323810
89 {chocolate} => {whole milk}       0.01667514 0.3360656  1.315243
```