Chapter 8 Code

Xi Liang

5/29/2017

Contents

Example 1. Identifying frequently purhcased groceries with association rules

Our market basket analysis wil utilize the purhcase data collected from one monht o operation at a real-word grocery store. The data contains 9,835 transactions or about 327 transactions per day.

Data Preparation - creating a sparse matrix for transaction data

```r
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

```r
groceries <- read.transactions("data/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
```

```
## 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
```

```r
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}
```

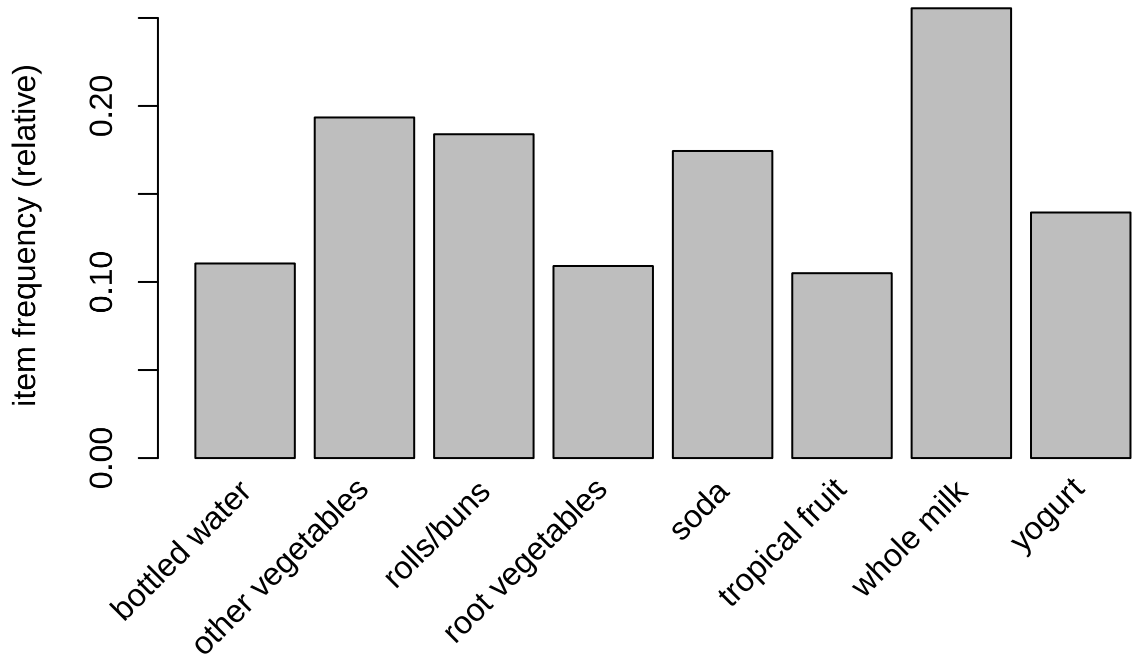
```r
itemFrequency(groceries[, 1:5])
```

```
## abrasive cleaner artif. sweetener   baby cosmetics        baby food
##     0.0035587189     0.0032536858     0.0006100661     0.0001016777
##             bags
##     0.0004067107
```

Visualizing item support - item frequency plots

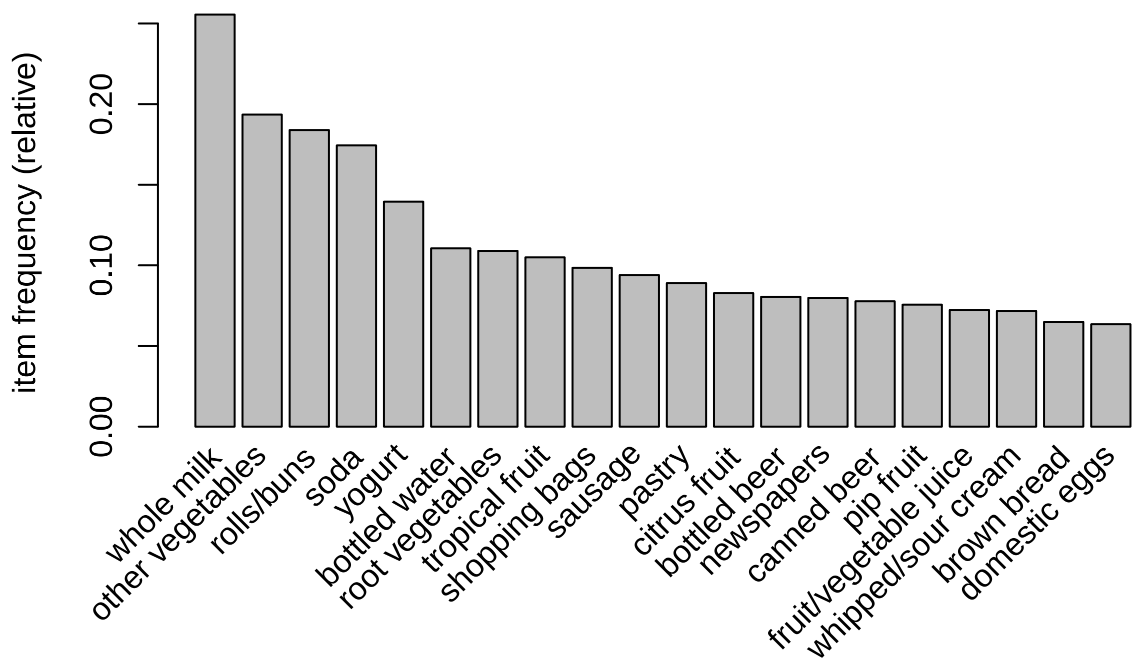
Eight items in the data with at least 10% support.

```r
itemFrequencyPlot(groceries, support = 0.1)
```



Top 20 items in the data

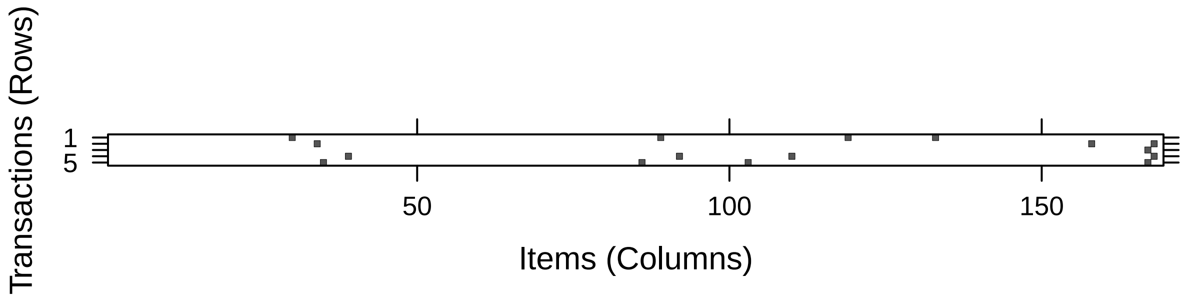
```r
itemFrequencyPlot(groceries, topN = 20)
```



Visualizing the transaction data - plotting the sparse matrix

To visualize the entire sparse matrix using `image()`

```r
image(groceries[1:5])
```

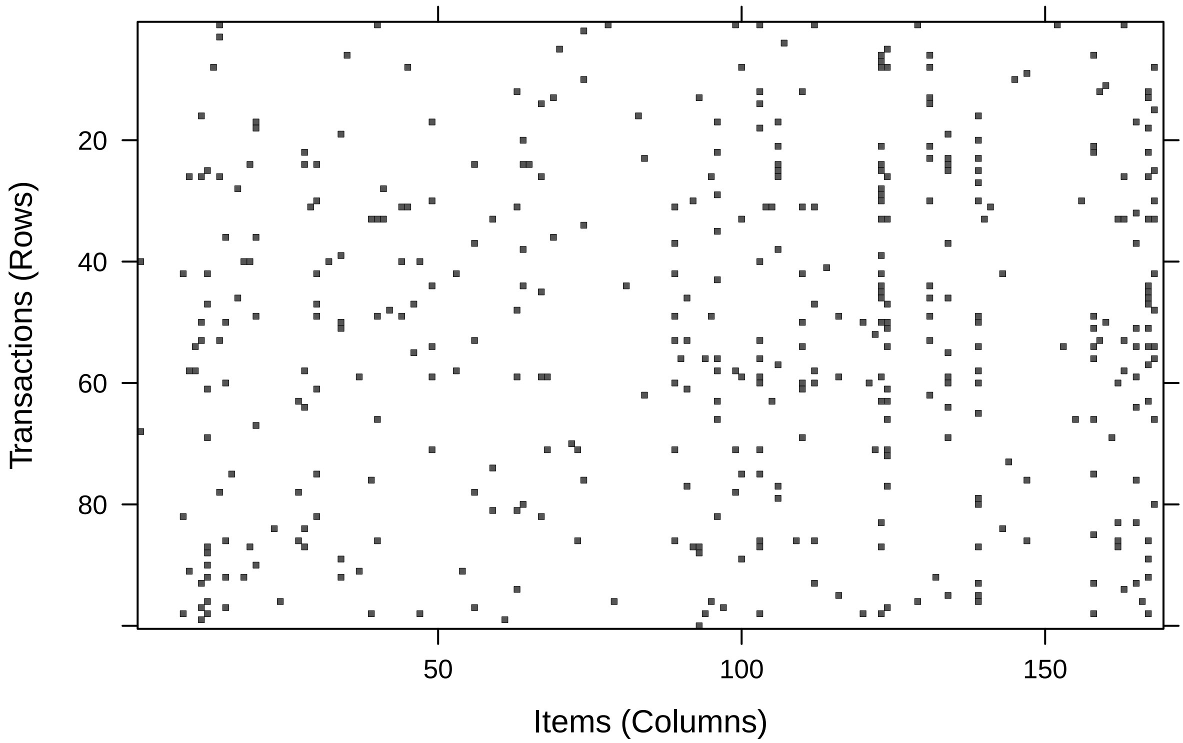


From the diagrame above, we observe that there are 5 rows and 169 columns, indicating 5 transactions and 169 possible items we requested.

We can also see that first, fourth, and fifth transactions contained four items each, and row three, five, two and four have an item in common.

Visualzing random selection of 100 transactions

```
image(sample(groceries, 100))
```



Training a model on the data

We will attempt to use the default settings of support = 0.1 and confidence = 0.8.

```
apriori(data = groceries)
```

```
## 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: 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 ended with 0 rules returned when we used the default settings, which is not surprising. Because support = 0.1 by default, in order to genderate a rule, an item must have appeared in at least 0.1 * 9,385 = 938.5 transactions. Since only eight items appeared this frequently in our data, it's no wonder that we did not find any rules.

One way to approach the above problem of setting a min support threshold is to think about the smalles number of transactions you would need before you would consider a pattern interesting. For example, you could argue that if an item is purhcased twice a day (about 60 times in a month of data), it may be an intersting pattern. From there, it is possible to calculate the support level needed to find only the rules mathcing a least that many transactions. Since 60/9835 = 0.006, we'll try setting the support there first.

We will start with confidence threhold of 0.25, which means that in order to be included in the reuslts,the rule has be correct at least 25 perent of the time. We'll also set minlen = 2 to eliminate rules that contain fewer than two items.

```
groceryrules <- apriori(groceries, parameter = list(support = 0.006, 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   0.006      2
##  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: 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].
```

```
groceryrules
```

```
## set of 463 rules
```

Our `groceryrules` object contains a set of 463 association rules. To determine whether any of them are usefule, we will have to dig deeper.

Evaluating model performance

```
summary(groceryrules)
```

```
## 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
```

In our rule set, 150 rules have only two items, while 297 have three, and 16 have four.

We will inspect the first 3 rules in the `groceryrules` object:

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

```
##     lhs             rhs                support     confidence lift
## [1] {pot 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
```

Improving model performance

Sorting the set of association rules

Depending upon the objects of the market basket analysis, the most uesful rules might be the ones with the highest support, confidence, or lift. The best five rules according to the lift statistic can be examined using the following command:

```
inspect(sort(groceryrules, 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
```

Taking subetsets of association rules

Suppose that given the preceding rule, the marketing team is excited about the possibilities of creating an advertisment to promote berries, which are now in season. Before finalizing the campagin, however, they ask you to investigate whether berries are often purchased with other items. To answer this question, we will need to find all the rules that include berries in some form.

```r
berryrules <- subset(groceryrules, items %in% "berries")
inspect(berryrules)
```

```
##     lhs          rhs                  support     confidence lift
## [1] {berries} => {whipped/sour cream} 0.009049314 0.2721713  3.796886
## [2] {berries} => {yogurt}             0.010574479 0.3180428  2.279848
## [3] {berries} => {other vegetables}   0.010269446 0.3088685  1.596280
## [4] {berries} => {whole milk}         0.011794611 0.3547401  1.388328
```

Saving association rules to a file or data frame

```r
write(groceryrules, file = "groceryrules.csv",
      sep = ",", quote = TRUE, row.names = FALSE)
```

```r
groceryrules_df <- as(groceryrules, "data.frame")
```

```r
str(groceryrules_df)
```

```
## 'data.frame':    463 obs. of  4 variables:
##  $ rules     : Factor w/ 463 levels "{baking powder} => {other vegetables}",..: 340 302 207 206 208 3
##  $ support   : num  0.00691 0.0061 0.00702 0.00773 0.00773 ...
##  $ confidence: num  0.4 0.405 0.431 0.475 0.475 ...
##  $ lift      : num  1.57 1.59 3.96 2.45 1.86 ...
```