Installing the arules package

```
> ##install.packages("arules")
> library(arules)
```

Our market basket analysis will utilize the purchase data collected from one month of operation at a real-world grocery store. The data contains 9,835 transactions or about 327 transactions per day (roughly 30 transactions per hour in a 12-hour business day),suggesting that the retailer is not particularly large, nor is it particularly small.

Since we're loading the transactional data, we cannot simply use the read.csv() function used previously. Instead, arules provides a read.transactions() function that is similar to read.csv() with the exception that it results in a sparse matrix suitable for transactional data. The sep = "," parameter specifies that items in the input file are separated by a comma.

```
> ##As it is a transactional dataset read.csv won't be a useful
> groceries<-read.transactions("groceries.csv", sep=",")
```

```
> ##Summary of the sparse matrix created
> summary(groceries)
```

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146
```

The output 9835 rows refers to the number of transactions, and the output 169 columns refers to the 169 different items that might appear in someone's grocery basket.

The density value of 0.02609146 (2.6 percent) refers to the proportion of nonzero matrix cells.

```
most frequent items:
      whole milk other vegetables       rolls/buns             soda           yogurt
            2513             1903             1809             1715             1372
         (Other)
           34055
```

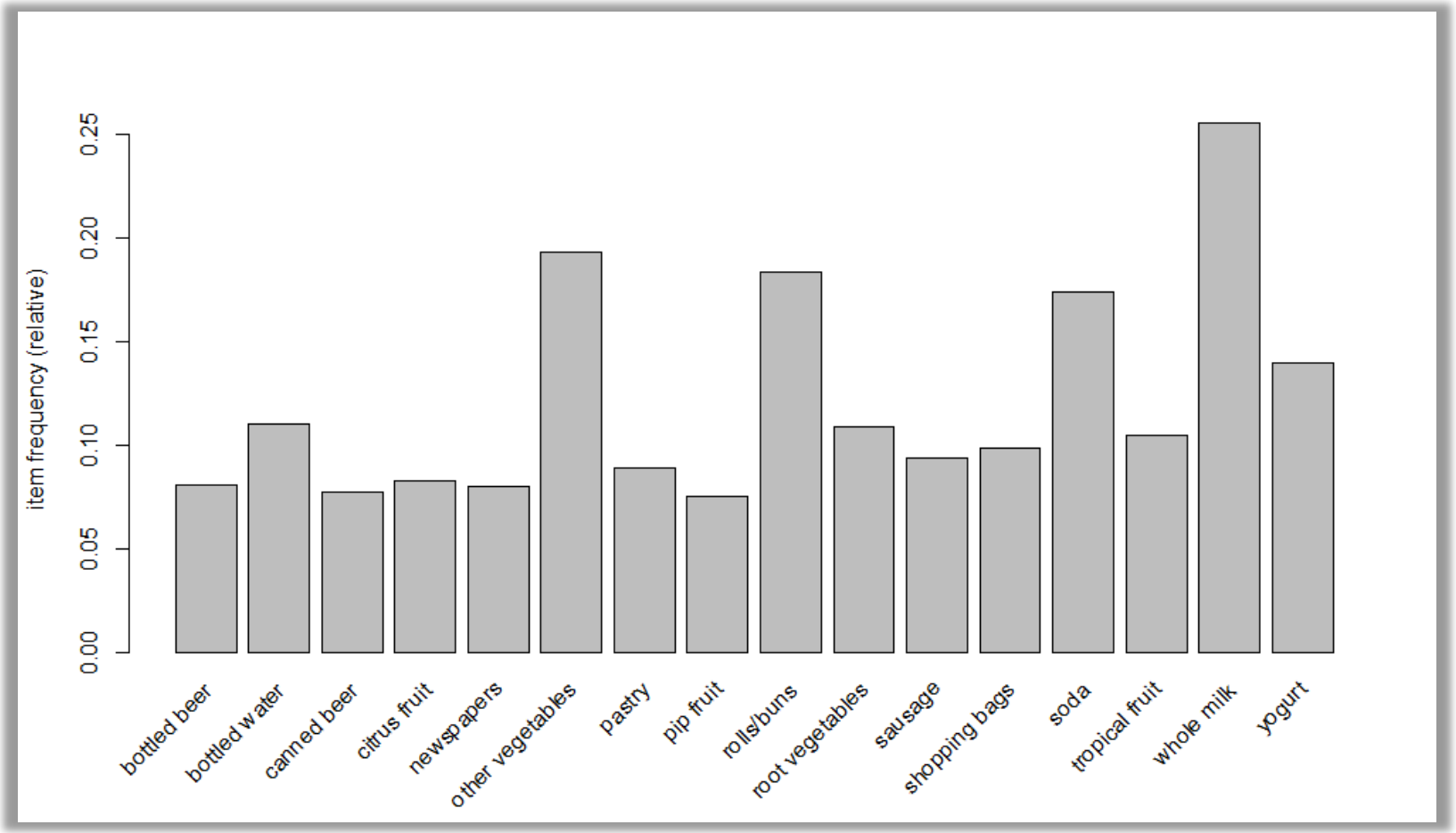
```
element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29
  18   19   20   21   22   23   24   26   27   28   29   32
  14   14    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
```

```
> ##Inspecting first five transaction
> 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}
```

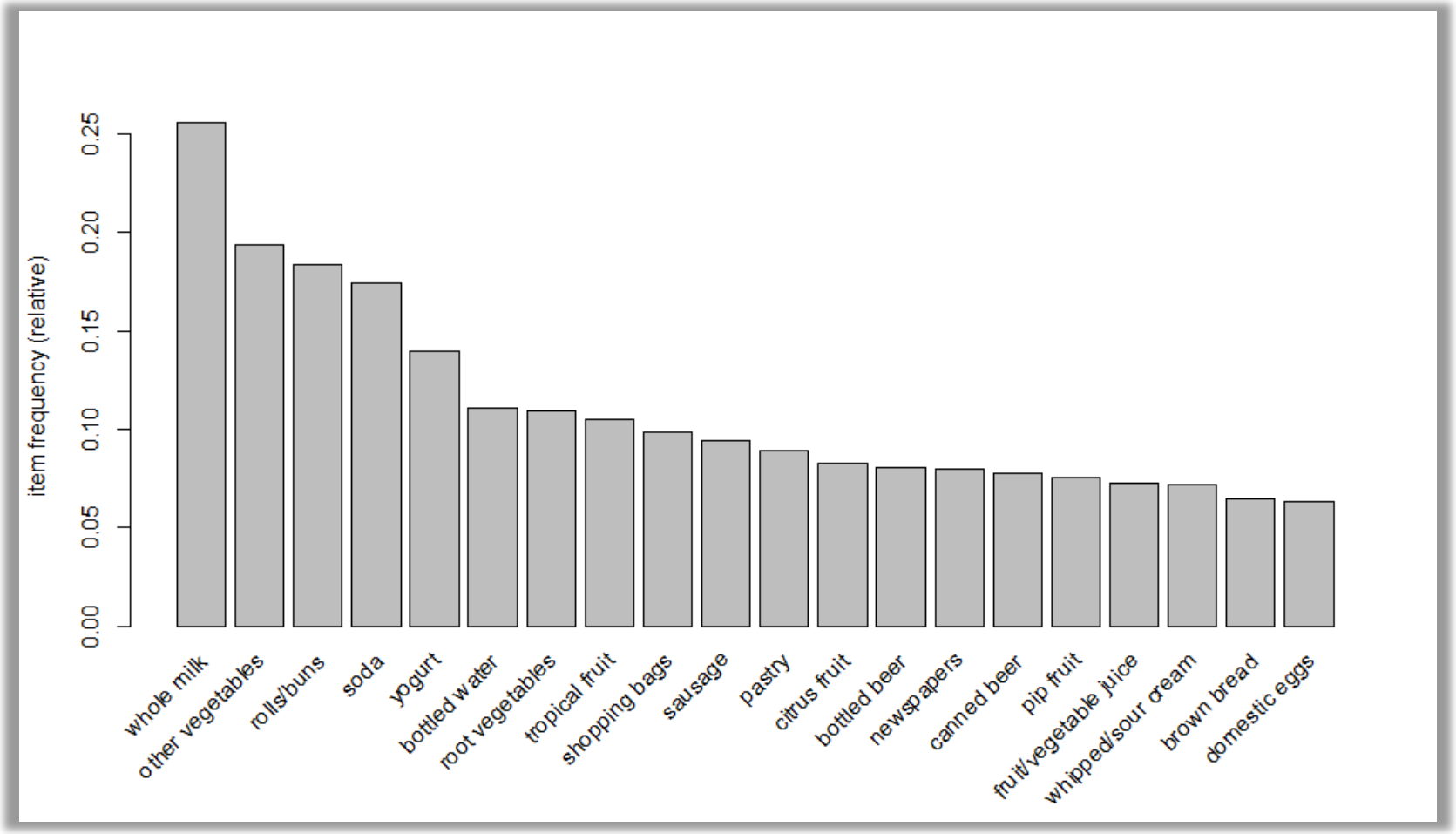
```
> ##Proportion of transaction for first few items
> itemFrequency(groceries[, 1:5])
abrasive cleaner artif. sweetener   baby cosmetics        baby food
    0.0035587189     0.0032536858     0.0006100661     0.0001016777
            bags
    0.0004067107
```

The items in the sparse matrix are sorted in columns by alphabetical order. Abrasive cleaner and artificial sweeteners are found in about 0.3 percent of the transactions, while baby cosmetics are found in about 0.06 percent of the transactions.

```
> ##Plot of most frequent items (Image 1)
> itemFrequencyPlot(groceries, support = 0.075)
```



```
> ##Top N items (Image 2)
> itemFrequencyPlot(groceries, topN = 20)
```



```
> ##Implementation of Apriori algorithm
> rules<-apriori(groceries, parameter=list(support=0.006, confidence=0.25,
+                                          minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
       0.25    0.1    1 none FALSE            TRUE       5   0.006      2     10
 target   ext
  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].
```

One way to approach the problem of setting a minimum support threshold is to think about the smallest number of transactions we would need before we would consider a pattern interesting. For instance, we could argue that if an item is purchased twice a day (about 60 times in a month of data), it may be an interesting pattern. From there, it is possible to calculate the support level needed to find only the rules matching at least that many transactions. Since 60 out of 9,835 equals 0.006, we'll try setting the support there first.

Setting the minimum confidence involves a delicate balance. On one hand, if confidence is too low, we might be overwhelmed with a large number of unreliable rules—such as dozens of rules indicating the items commonly purchased. How would we know where to target our advertising budget then? On the other hand, if we set confidence too high, we will be limited to the rules that are obvious or inevitable. The appropriate minimum confidence level depends a great deal on the goals of your analysis. If you start with a conservative value, you can always reduce it to broaden the search if you aren't finding actionable intelligence.

minlen = 2 to eliminate rules that contain fewer than two items.

```
> ##Rules
> rules
set of 463 rules
```

```
> ##Summary of rules created
> 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
```

We might be alarmed if most or all of the rules had support and confidence very near the minimum thresholds, as this would mean that we may have set the bar too high. This is not the case here, as there are many rules with much higher values of each.

```
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
```

Inspecting the rules generated.

```
> inspect(rules[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
```

Depending upon the objectives of the market basket analysis, the most useful rules might be the ones with the highest support, confidence, or lift. The arules package includes a sort() function that can be used to reorder the list of rules so that the ones with the highest or lowest values of the quality measure come first.

```
> 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
```

The subset() function provides a method to search for subsets of transactions, items, or rules. To use it to find any rules with berries appearing in the rule, use the following command. It will store the rules in a new object titled berryrules:

```
> berryrules<-subset(rules, 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
```