Market Basket Analysis

Jayasri

2023-02-14

## Market Basket Analysis

Install necessary packages

#arpack <- c("arules","arulesViz")  
#install.packages(arpack) # install if necessary  
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

library(arulesViz)

## Groceries Dataset

Load the Groceries dataset

data("Groceries")  
##str(Groceries)  
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 level2 level1  
## 1 frankfurter sausage meat and sausage  
## 2 sausage sausage meat and sausage  
## 3 liver loaf sausage meat and sausage

**Observations:** The summary shows that the most frequent items in the dataset include items such as whole milk, other vegetables, rolls/buns, soda, and yogurt. These items are purchased more often than the others.

The class of the dataset is transactions, as defined by the arules package. The transactions class contains three slots:

● itemsetInfo: Not being used currently ● itemInfo: A data frame to store itemlabels ● data: A binary incidence matrix that indicates which item labels appear in every transaction

Groceries@itemInfo[1:20,]

## labels level2 level1  
## 1 frankfurter sausage meat and sausage  
## 2 sausage sausage meat and sausage  
## 3 liver loaf sausage meat and sausage  
## 4 ham sausage meat and sausage  
## 5 meat sausage meat and sausage  
## 6 finished products sausage meat and sausage  
## 7 organic sausage sausage meat and sausage  
## 8 chicken poultry meat and sausage  
## 9 turkey poultry meat and sausage  
## 10 pork pork meat and sausage  
## 11 beef beef meat and sausage  
## 12 hamburger meat beef meat and sausage  
## 13 fish fish meat and sausage  
## 14 citrus fruit fruit fruit and vegetables  
## 15 tropical fruit fruit fruit and vegetables  
## 16 pip fruit fruit fruit and vegetables  
## 17 grapes fruit fruit and vegetables  
## 18 berries fruit fruit and vegetables  
## 19 nuts/prunes fruit fruit and vegetables  
## 20 root vegetables vegetables fruit and vegetables

Each grocery label is mapped to two levels of categories— level2 and level1 — where level1 is a superset of level2.For example, grocery label sausage belongs to the sausage category in level2, and it is part of the meat and sausage category in level1.

The following code displays the 10th to 20th transactions of the Groceries dataset. The [10:20] can be changed to [1:9835] to display all the transactions.

apply(Groceries@data[,10:20], 2,  
function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")  
)

## [1] "whole milk, cereals"   
## [2] "tropical fruit, other vegetables, white bread, bottled water, chocolate"   
## [3] "citrus fruit, tropical fruit, whole milk, butter, curd, yogurt, flour, bottled water, dishes"  
## [4] "beef"   
## [5] "frankfurter, rolls/buns, soda"   
## [6] "chicken, tropical fruit"   
## [7] "butter, sugar, fruit/vegetable juice, newspapers"   
## [8] "fruit/vegetable juice"   
## [9] "packaged fruit/vegetables"   
## [10] "chocolate"   
## [11] "specialty bar"

Each row in the output shows a transaction that includes one or more products, and each transaction corresponds to everything in a customer’s shopping cart. For example, in the first transaction, a customer has purchased whole milk and cereals. \*\*\* ## Frequent Itemset Generation We will use the apriori() function from the arule package to create frequent itemsets. By default, apriori() executes all iterations at once. But, for illustration purposes, we will execute apriori() manually from iteration to iteration.

### Iteration 1-itemsets

The following code identifies 59 frequent 1-itemsets that satisfy the minimum support. The parameters of apriori() specify the minimum and maximum lengths of the itemsets, the minimum support threshold, and the target indicating the type of association mined.

itemsets <- apriori(Groceries, parameter=list(minlen=1, maxlen=1,  
support=0.02, target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.02 1  
## maxlen target ext  
## 1 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 196   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [59 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1

## Warning in apriori(Groceries, parameter = list(minlen = 1, maxlen = 1, support  
## = 0.02, : Mining stopped (maxlen reached). Only patterns up to a length of 1  
## returned!

## done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [59 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(itemsets)

## set of 59 itemsets  
##   
## most frequent items:  
## frankfurter sausage ham meat chicken (Other)   
## 1 1 1 1 1 54   
##   
## element (itemset/transaction) length distribution:sizes  
## 1   
## 59   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1 1 1 1 1 1   
##   
## summary of quality measures:  
## support transIdenticalToItemsets count   
## Min. :0.02105 Min. :0.000305 Min. : 207.0   
## 1st Qu.:0.03015 1st Qu.:0.001373 1st Qu.: 296.5   
## Median :0.04809 Median :0.002440 Median : 473.0   
## Mean :0.06200 Mean :0.004031 Mean : 609.8   
## 3rd Qu.:0.07666 3rd Qu.:0.003711 3rd Qu.: 754.0   
## Max. :0.25552 Max. :0.028266 Max. :2513.0   
##   
## includes transaction ID lists: FALSE   
##   
## mining info:  
## data ntransactions support confidence  
## Groceries 9835 0.02 1

The summary of the itemsets shows that the support of 1-itemsets ranges from 0.02105 to 0.25552. Because the maximum support of the 1-itemsets in the dataset is only 0.25552, to enable the discovery of interesting rules, the minimum support threshold should not be set too close to that number.

Let us now inspect the top 10 frequent 1-itemsets

inspect(head(sort(itemsets, by = "support"), 10))

## items support transIdenticalToItemsets count  
## [1] {whole milk} 0.25551601 0.015149975 2513   
## [2] {other vegetables} 0.19349263 0.008439248 1903   
## [3] {rolls/buns} 0.18393493 0.015048297 1809   
## [4] {soda} 0.17437722 0.019318760 1715   
## [5] {yogurt} 0.13950178 0.005998983 1372   
## [6] {bottled water} 0.11052364 0.008845958 1087   
## [7] {root vegetables} 0.10899847 0.003355363 1072   
## [8] {tropical fruit} 0.10493137 0.003050330 1032   
## [9] {shopping bags} 0.09852567 0.006100661 969   
## [10] {sausage} 0.09395018 0.003050330 924

Of all the transaction records, the 59 1-itemsets such as {whole milk}, {other vegetables}, {rolls/buns}, {soda}, and {yogurt} all satisfy the minimum support. Therefore, they are called frequent 1-itemsets.

### Iteration 2-itemsets

In this iteration, the list of frequent 1-itemsets is joined onto itself to form all possible candidate2-itemsets.

itemsets <- apriori(Groceries, parameter=list(minlen=2, maxlen=2,  
support=0.02, target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.02 2  
## maxlen target ext  
## 2 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 196   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [59 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2

## Warning in apriori(Groceries, parameter = list(minlen = 2, maxlen = 2, support  
## = 0.02, : Mining stopped (maxlen reached). Only patterns up to a length of 2  
## returned!

## done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [61 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(itemsets)

## set of 61 itemsets  
##   
## most frequent items:  
## whole milk other vegetables yogurt rolls/buns   
## 25 17 9 9   
## soda (Other)   
## 9 53   
##   
## element (itemset/transaction) length distribution:sizes  
## 2   
## 61   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2 2 2 2 2 2   
##   
## summary of quality measures:  
## support transIdenticalToItemsets count   
## Min. :0.02003 Min. :0.0000000 Min. :197.0   
## 1st Qu.:0.02227 1st Qu.:0.0003050 1st Qu.:219.0   
## Median :0.02613 Median :0.0005084 Median :257.0   
## Mean :0.02951 Mean :0.0007151 Mean :290.3   
## 3rd Qu.:0.03223 3rd Qu.:0.0009151 3rd Qu.:317.0   
## Max. :0.07483 Max. :0.0034570 Max. :736.0   
##   
## includes transaction ID lists: FALSE   
##   
## mining info:  
## data ntransactions support confidence  
## Groceries 9835 0.02 1

We now display the top 10 2-itemsets

inspect(head(sort(itemsets, by ="support"),10))

## items support transIdenticalToItemsets  
## [1] {other vegetables,whole milk} 0.07483477 0.0009150991   
## [2] {whole milk,rolls/buns} 0.05663447 0.0020335536   
## [3] {whole milk,yogurt} 0.05602440 0.0010167768   
## [4] {root vegetables,whole milk} 0.04890696 0.0007117438   
## [5] {root vegetables,other vegetables} 0.04738180 0.0009150991   
## [6] {other vegetables,yogurt} 0.04341637 0.0004067107   
## [7] {other vegetables,rolls/buns} 0.04260295 0.0012201322   
## [8] {tropical fruit,whole milk} 0.04229792 0.0009150991   
## [9] {whole milk,soda} 0.04006101 0.0014234875   
## [10] {rolls/buns,soda} 0.03833249 0.0034570412   
## count  
## [1] 736   
## [2] 557   
## [3] 551   
## [4] 481   
## [5] 466   
## [6] 427   
## [7] 419   
## [8] 416   
## [9] 394   
## [10] 377

### Iteration 3-itemsets

Let us now combine the frequent 2-itemsets to get candidate 3-itemsets

itemsets <- apriori(Groceries, parameter=list(minlen=3, maxlen=3,  
support=0.02, target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.02 3  
## maxlen target ext  
## 3 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 196   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [59 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3

## Warning in apriori(Groceries, parameter = list(minlen = 3, maxlen = 3, support  
## = 0.02, : Mining stopped (maxlen reached). Only patterns up to a length of 3  
## returned!

## done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [2 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(sort(itemsets, by ="support"))

## items support   
## [1] {root vegetables,other vegetables,whole milk} 0.02318251  
## [2] {other vegetables,whole milk,yogurt} 0.02226741  
## transIdenticalToItemsets count  
## [1] 0.000305033 228   
## [2] 0.000305033 219

There are only two itemsets here.

### Iteration 4-itemsets

Let us run the algorithm one more time

itemsets <- apriori(Groceries, parameter=list(minlen=4, maxlen=4,  
support=0.02, target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.02 4  
## maxlen target ext  
## 4 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 196   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [59 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [0 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(sort(itemsets, by ="support"))

The previous steps simulate the Apriori algorithm at each iteration. For the Groceries dataset, the iterations run out of support when k = 4. Therefore, the frequent itemsets contain 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

### Iteration Complete

Let us now run the apriori algorithm with no restriction on the maxlen

itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,  
target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.02 1  
## maxlen target ext  
## 10 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 196   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [59 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [122 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

As shown in the code output that follows, 122 frequent itemsets have been identified. This matches the total number of 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.