HW2_wenting_xu

Q1

a.

Suppose ith dimension has

$$Li$$

levels, the total number of cuboid is

$$\prod (Li + 1)$$

There are 10 dimensions with no hierarchy, so the number of cuboids are

```
2^10
```

```
## [1] 1024
```

```
1024 cuboids are there in the full data cube
```

b.

First, we consider at least one of the first three elements is not aggregated.

```
(2^3 - 1)*2^7
```

```
## [1] 896
```

There are 3 base cell.

```
3 * 896
```

```
## [1] 2688
```

Then we consider that if the first three dimensions are all aggregated

```
2^7
```

```
## [1] 128
```

Since there are 3 base cells. So the number of cell is

```
2688 + 128 - 3
```

```
## [1] 2813
```

```
The complete cube will contain 2813 distinct aggregated (i.e., non-base) cells.
```

c.

when count > 2, it means we only consider the last 7 dimensions.

```
2^7
```

```
## [1] 128
```

```
An iceberg cube will contain 128 distinct aggregated cells, if the condition of the iceberg cube is cou
```

c.

only the last 7 dimensions can be counted 3

```
The closed cell with count = 3  has 7 non-star dimensions.
```

Q2

a

Suppose ith dimension has

$$Li$$

levels, the total number of cuboid is

$$\prod(Li+1)$$

```r
3*2*2*2
```

```
## [1] 24
```

b

```r
data = read.csv("Q2data.csv",header = FALSE)
name = c("id","state","city","category","price","rating")
colnames(data) = name
```

```r
## sort
data = data[order(data[,3],data[,4],data[,5],data[,6]),]
```

```r
library(plyr)
nrow(count(data,vars = c("city","category","price","rating")))
```

```
## [1] 48
```

```
There are 48 cells in the cuboid (Location(city), Category, Rating, Price).
```

c

```r
nrow(count(data,vars = c("state","category","price","rating")))
```

```
## [1] 34
```

```
There are 34 cells in the cuboid (Location(State), Category, Rating, Price).
```

d

```r
nrow(count(data,vars = c("category","price","rating")))
```

```
## [1] 23
```

```
There are 23 cells in the cuboid (* , Category , Rating , Price).
```

e

```r
sum(data$state == "Illinois" & data$rating == 3 & data$price == "moderate")
```

```
## [1] 2
```

The count for the cell (Location(state) = 'Illinois' , * , rating = 3 , Price = 'Moderate') is 2.

f

```r
sum(data$city == "Chicago" & data$category == "food")
```

```
## [1] 2
```

The count for the cell (Location(city) = 'Chicago' , Category='food' , * , *) is 2

Q3

a. support = 20

1.

```r
library("arules")
```

```
## Warning: package 'arules' was built under R version 3.4.2
## Loading required package: Matrix
##
## Attaching package: 'arules'
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
tr = read.transactions("Q3data",format="basket",sep=" ")
```

```r
frequentItems = eclat (tr, parameter = list(supp = 0.2))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE     0.2      1     10 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 20
##
## create itemset ...
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
```

```
## creating bit matrix ... [7 row(s), 100 column(s)] done [0.00s].
## writing  ... [30 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```r
length(frequentItems)
```

```
## [1] 30
```

The number of frequent patterns is 30

2.

```r
frequentItems <- eclat (tr, parameter = list(supp = 0.2,maxlen = 3, minlen = 3))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE     0.2      3      3 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 20
##
## create itemset ...
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating bit matrix ... [7 row(s), 100 column(s)] done [0.00s].
## writing  ... [8 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```r
length(frequentItems)
```

```
## [1] 8
```

The number of frequent patterns with length 3 is 8

3

```r
rules <- apriori(tr,
                 parameter = list(supp = 0.2, target = "rules"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.2      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: 20
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [17 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
maximal = is.maximal(rules)
length(rules[maximal])
```

```
## [1] 7
```

The number of max patterns is 7

b. support = 10

1

```
frequentItems_10 = eclat (tr, parameter = list(supp = 0.1))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE     0.1      1     10 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 10
##
## create itemset ...
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating bit matrix ... [7 row(s), 100 column(s)] done [0.00s].
## writing  ... [55 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
length(frequentItems_10)
```

```
## [1] 55
```

The number of frequent pattern is 55.

2

```
frequentItems_10_3 = eclat (tr, parameter = list(supp = 0.1,maxlen = 3, minlen = 3))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE     0.1      3      3 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 10
##
## create itemset ...
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating bit matrix ... [7 row(s), 100 column(s)] done [0.00s].
## writing  ... [20 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```r
length(frequentItems_10_3)
```

```
## [1] 20
```

The number of frequent patterns with length 3 is 20

3

Calculate the number of maximal patterns

```r
maximal = is.maximal(frequentItems_10)
length(frequentItems_10[maximal])
```

```
## [1] 6
```

4

```r
frequentItems = apriori(tr, parameter = list(supp = 0.1,conf = 0,maxlen = 3,minlen = 3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##           0    0.1    1 none FALSE            TRUE       5     0.1      3
##  maxlen target   ext
##       3  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 10
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
```

```
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3
```

```
## Warning in apriori(tr, parameter = list(supp = 0.1, conf = 0, maxlen =
## 3, : Mining stopped (maxlen reached). Only patterns up to a length of 3
## returned!
```

```
##  done [0.00s].
## writing ... [60 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
round(quality(frequentItems[60])$confidence,3)
```

```
## [1] 0.679
```

According to the formula:

$$P(A|C \cap E) = \frac{P(A \cap C \cap E)}{P(C \cap E))}$$

`the confidence measure of the association rule (C, E) -> A is 0.679`

5

```
frequentItems = apriori (tr, parameter = list(supp = 0.1,conf = 0,maxlen = 4, minlen = 4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##           0    0.1    1 none FALSE            TRUE       5     0.1      4
##  maxlen target   ext
##       4  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 10
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[7 item(s), 100 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4
```

```
## Warning in apriori(tr, parameter = list(supp = 0.1, conf = 0, maxlen =
## 4, : Mining stopped (maxlen reached). Only patterns up to a length of 4
## returned!
```

```
##  done [0.00s].
## writing ... [36 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
round(quality(frequentItems[34])$confidence,3)
```

```
## [1] 0.742
```

According to the formula:

$$P(E|A \cap B \cap C) = \frac{P(A \cap B \cap C \cap E)}{P(A \cap B \cap C))}$$

```
the confidence measure of the association rule (A, B, C) -> E is 0.742
```