Association Rules

Association Rule:

Association Rule Mining is one of the ways to find patterns (interesting associations and relationships) in data.

- It is sometimes referred to as "Market Basket Analysis", since that was the original application area of association mining.
- An association rule is an implication of the form

$$A \Rightarrow B,$$

where $A \subset I$, $B \subset I$, $A \neq \emptyset$, $B \neq \emptyset$, and $A \cap B \neq \emptyset$.

A is called **Antecedent** (Left Hand Side - LHS) and B is called **Consequent** (Right Hand Side - RHS).

B is the response (If A, then B).

Application of Association Rule Mining:

1. Market Basket Analysis.
2. Medical diagnosis.
3. Protein Sequences.
4. Fraud Detection in Credit Card Transactions.
5. Image classification.
6. Analyzing customer reviews.
7. Census Data.

Three Common Metrics to Measure Association:

1. **Support:** the percentage of transactions that contain all of the items in an itemset ($P(\mathrm{A} \cap \mathrm{B})$).

$$Sup(\mathrm{A}) = Support(\mathrm{A}) = \frac{frequency(A)}{N} = P(\mathrm{A})$$

$$Support(\mathrm{A} \Rightarrow B) = \frac{frequency(A, B)}{N} = P(\mathrm{A} \cap \mathrm{B})$$

Three Common Metrics to Measure Association:

2. **Confidence:** the percentage that a transaction that contains the items on the left hand side of the rule ($P(\mathrm{B}|\mathrm{A})$).

$$Confidence(\mathrm{A} \Rightarrow B) = \frac{sup(\mathrm{A} \cap \mathrm{B})}{sup(\mathrm{A})} = \frac{\frac{frequency(A,B)}{N}}{\frac{frequency(A)}{N}}$$

$$= \frac{frequency(A,B)}{frequency(A)} = \frac{P(\mathrm{A} \cap \mathrm{B})}{P(\mathrm{A})} = P(\mathrm{B}|\mathrm{A})$$

Note: *Support* and *Confidence* measure how interesting the rule is. It is set by the minimum support and minimum confidence thresholds.

Three Common Metrics to Measure Association:

3. **Lift:** the parentage of all of the items in a rule occurring together (a simple correlation measure).

$$Lift(\mathrm{A}, B) = \frac{sup(\mathrm{A} \cap \mathrm{B})}{sup(\mathrm{A})sup(\mathrm{B})} = \frac{P(\mathrm{A} \cap \mathrm{B})}{P(\mathrm{A})P(\mathrm{B})} = \frac{P(\mathrm{B}|\mathrm{A})}{P(\mathrm{B})}$$

if the two items are statistically independent, then the joint probability of the two items will be the same as the product of their probabilities. In other words, $P(\mathrm{A} \cap \mathrm{B}) = P(\mathrm{A})P(\mathrm{B})$.

- If the rule had a lift of 1, then A and B are independent and no rule can be derived from them.
- If the lift is > 1, then A and B are dependent on each other, and the degree of which is given by lift value.
- If the lift is < 1, then presence of A will have negative effect on B.

Example 1:

Find the support, confidence, and lift of the association rule $\{Bread\} \Rightarrow \{Milk\}$ for the following,

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

	Beer	Bread	Milk	Diaper	Eggs	Coke
T_1	0	1	1	0	0	0
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	1	1	0	0
T_5	0	1	1	1	0	1

$$Support\{Bread\} = \frac{4}{5} \quad and \quad Support\{Milk\} = \frac{4}{5}$$

Support says that 80% of customers purchased bread and milk.

$$Confidence(\{Bread\} \Rightarrow \{Milk\}) = \frac{P(\{B\} \cap \{M\})}{P(\{B\})} = \frac{3/5}{4/5} = 0.75$$

75% of the customers that bought bread also bought milk.

$$Lift(\{Bread\} \Rightarrow \{Milk\}) = \frac{P(\{B\} \cap \{M\})}{P(\{B\})P(\{M\})} = \frac{3/5}{(4/5)(4/5)} = 0.9$$

There is a negative association between the Bread and Milk.

Example 1:

For the same example, $\{\text{Egg}\} \Rightarrow \{Bread\}$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

	Beer	Bread	Milk	Diaper	Eggs	Coke
T_1	0	1	1	0	0	0
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	1	1	0	0
T_5	0	1	1	1	0	1

$$Support\{\text{Egg}\} = \frac{1}{5} \quad and \quad Support\{Bread\} = \frac{4}{5}$$

20% of customers purchased egg and 80% purchased bread.

$$Confidence(\{\text{Egg}\} \Rightarrow \{Bread\}) = \frac{P(\{E\} \cap \{B\})}{P(\{E\})} = \frac{1/5}{1/5} = 1$$

100% of the customers that bought egg also bought bread.

$$Lift(\{\text{Egg}\} \Rightarrow \{Bread\}) = \frac{P(\{E\} \cap \{E\})}{P(\{E\})P(\{B\})} = \frac{1/5}{(1/5)(4/5)} = 1.25$$

Lift represents the 25% increase in expectation that someone will buy bread, when we know that they bought egg.

Goal of Association Rule:

When you apply Association Rule Mining on a given set of transactions T your goal will be to find all rules with:

- Support greater than or equal to min_support
- Confidence greater than or equal to min_confidence

- We can limit the number of rules by tweaking a few parameters such as support, confidence and other parameters.
- If you want to get stronger rules, you have to increase the confidence. If you want lengthier rules increase the maxlen (the maximum number of items that can be present in the rule) parameter. If you want to eliminate shorter rules, decrease the minlen (minimum number of items required in the rule) parameter.

Example 2: (Groceries)

The *Groceries* dataset contains 1 month (30 days) of real-world point-of-sale transaction data from a typical local grocery outlet. The data set contains 9835 transactions and the items are aggregated to 169 categories (roughly 30 transactions per hour in a 12-hour business day).

1. The data come with **arules** package, so install it and install also **arulesviz** package which we will use it later.

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

```
install.packages("arulesviz")
library("arulesviz")
```

Example 2: (Groceries)

2. Since the dataset contains transactions, so it has a different format. Use data() function to read the data.

3. Use summary() function to show the most frequent items in the dataset.

```
> data(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           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

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

- The whole milk was the most often purchased item shown in 2513 transactions.

Example 2: (Groceries)

4. Since the example deals with transactions, the data has to be converted to one of class transactions. This is a necessary step because the apriori() function accepts transactions data of class transactions only.

The transactions class contains three slots:

- transactionInfo: A data frame with vectors of the same length as the number of transactions.
- itemInfo: A data frame to store item labels.
- data: a binary incidence matrix that indicates which item labels appear in every transaction.

```
> class(Groceries)
[1] "transactions"
attr(,"package")
[1] "arules"
```

Example 2: (Groceries)

5. Enter Groceries@itemInfo to display all 169 labels and their categories. Note: use @ not $.

Use Groceries@itemInfo$labels to display the labels only.

```
> 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
```

Example 2: (Groceries)

6. Look at the first 5 transactions.

We may use the head() function.

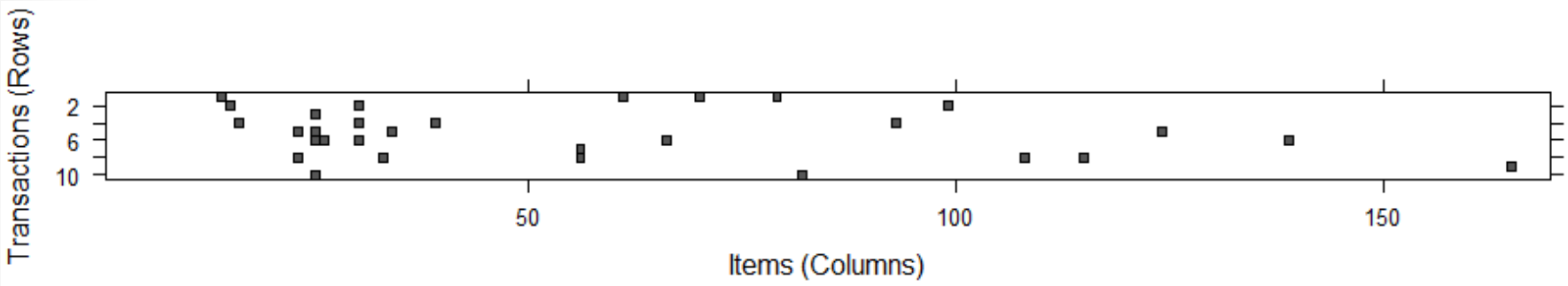
```
> inspect(Groceries[1:5])
    items
[1] {citrus fruit,semi-finished bread,margarine,ready soups}
[2] {tropical fruit,yogurt,coffee}
[3] {whole milk}
[4] {pip fruit,yogurt,cream cheese ,meat spreads}
[5] {other vegetables,whole milk,condensed milk,long life bakery product}
```

```
> inspect(head(Groceries))
    items
[1] {citrus fruit,semi-finished bread,margarine,ready soups}
[2] {tropical fruit,yogurt,coffee}
[3] {whole milk}
[4] {pip fruit,yogurt,cream cheese ,meat spreads}
[5] {other vegetables,whole milk,condensed milk,long life bakery product}
[6] {whole milk,butter,yogurt,rice,abrasive cleaner}
```

Example 2: (Groceries)

7. Display the sparse matrix for the first 10 transactions.

```
> image(Groceries[1:10])
```




```
> inspect(Groceries[1:10])
     items
[1]  {citrus fruit,semi-finished bread,margarine,ready soups}
[2]  {tropical fruit,yogurt,coffee}
[3]  {whole milk}
[4]  {pip fruit,yogurt,cream cheese ,meat spreads}
[5]  {other vegetables,whole milk,condensed milk,long life bakery product}
[6]  {whole milk,butter,yogurt,rice,abrasive cleaner}
[7]  {rolls/buns}
[8]  {other vegetables,UHT-milk,rolls/buns,bottled beer,liquor (appetizer)}
[9]  {pot plants}
[10] {whole milk,cereals}
```

Example 2: (Groceries)

8. Use the cross table to present the most frequent items in the transactions.

```
> ct = crossTable(Groceries, sort=TRUE)
> ct[1:5, 1:5]
                  whole milk other vegetables rolls/buns soda yogurt
whole milk              2513              736        557  394    551
other vegetables         736             1903        419  322    427
rolls/buns               557              419       1809  377    338
soda                     394              322        377 1715    269
yogurt                   551              427        338  269   1372
```

- The most often purchased item was whole milk (2513), then other vegetables (1903), etc.
- The whole milk and other vegetables were shown in 736 transactions.
- The whole milk and yogurt were shown in 551 transactions

Example 2: (Groceries)

9. Use which() function to find a specific label.

10. Find the percentage (relative frequency) of a specific label (labels).

```
> which(Groceries@itemInfo$labels == 'whole milk')
[1] 25
```

```
> itemFrequency(Groceries[, 25])
whole milk
  0.255516
```

```
> itemFrequency(Groceries[, 25], type = "absolute")
whole milk
      2513
```

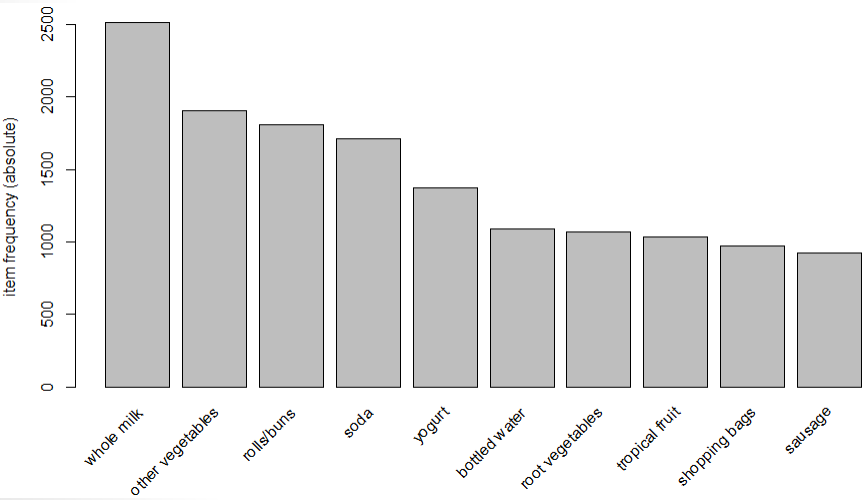
```
> itemFrequency(Groceries[,1:5])
frankfurter     sausage  liver loaf         ham        meat
0.058973055 0.093950178 0.005083884 0.026029487 0.025826131
```

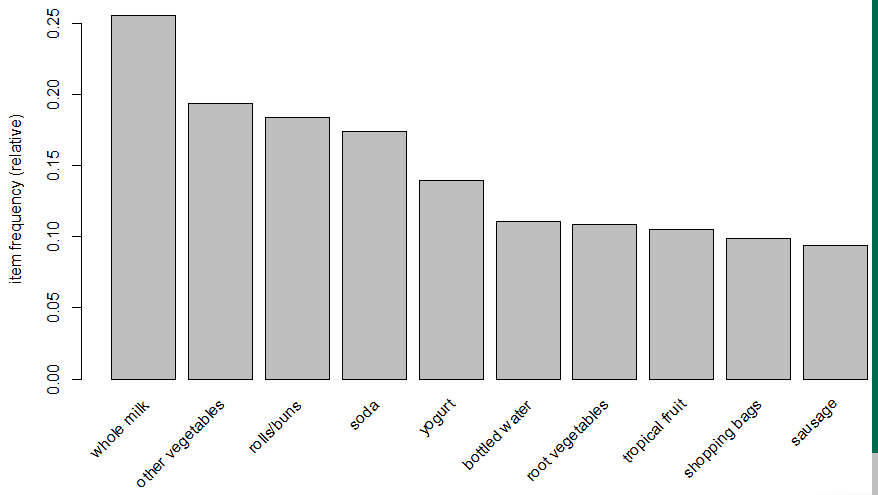
Example 2: (Groceries)

11. Plot the bar chart to display the frequency (and relative frequency) of the most 10 requent items.

Try adding "col=brewer.pal(8,'Pastel2')" to the function.

```
> itemFrequencyPlot(Groceries,topN=10, type="absolute") > itemFrequencyPlot(Groceries,topN=10, type="relative")
```





- The relative bar chart helps to determine the support for each item.

Example 2: (Groceries)

12. Now we have an idea of how the data looks.

We proceed to create rules. Rules are formed by defining the minimum support and confidence levels. Also the minlen option lets us to set the minimum number of items for both the LHS and RHS.

```
> apriori(Groceries)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1     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
```

Example 2: (Groceries)

13. Set a support threshold of 0.001 and confidence of 0.25.

```
> rules = apriori(Groceries, parameter = list(supp = 0.001, conf = 0.25))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
       0.25    0.1    1 none FALSE            TRUE       5   0.001      1     10  rules FALSE

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

Absolute minimum support count: 9
```

```
> rules
set of 17392 rules
```

- We can run the Apriori algorithm and obtain a set of 17,392 results.
- This number would reduce by tweaking either threshold.

Example 2: (Groceries)

14. Display the first 5 rules.

Round the numbers to the nearest two decimal places.

```
> options(digits=2)
> inspect(rules[1:5])
    lhs           rhs           support confidence lift count
[1] {}         => {whole milk} 0.2555  0.26       1.0  2513
[2] {honey}    => {whole milk} 0.0011  0.73       2.9    11
[3] {soap}     => {whole milk} 0.0011  0.42       1.7    11
[4] {tidbits}  => {soda}       0.0010  0.43       2.5    10
[5] {tidbits}  => {rolls/buns} 0.0012  0.52       2.8    12
```

- We can write the rules to a CSV file using

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

Example 2: (Groceries)

15. Set the default in which is mine rules with minimum support of 0.001, minimum confidence of 0.2, and maximum of 3 items (maxlen).

```
> rules = apriori(Groceries, parameter = list(supp = 0.001, conf = 0.2, maxlen=3))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.2    0.1    1 none FALSE            TRUE       5   0.001      1      3  rules FALSE

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

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [9958 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning message:
In apriori(Groceries, parameter = list(supp = 0.001, conf = 0.2,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
> rules
set of 9958 rules
```

Example 2: (Groceries)

16. Sort the results according to confidence, and show only the first 5 rules.

```
> inspect(sort(rules, by="confidence", decreasing=TRUE)[1:5])
    lhs                                                rhs          support     confidence lift     count
[1] {rice,sugar}                                    => {whole milk} 0.001220132 1.0000000  3.913649 12
[2] {canned fish,hygiene articles}                  => {whole milk} 0.001118454 1.0000000  3.913649 11
[3] {whipped/sour cream,house keeping products}     => {whole milk} 0.001220132 0.9230769  3.612599 12
[4] {rice,bottled water}                            => {whole milk} 0.001220132 0.9230769  3.612599 12
[5] {soups,bottled beer}                            => {whole milk} 0.001118454 0.9166667  3.587512 11
```

- Looking at the first rule, there is a high likelihood of whole milk being purchased along with root rice and sugar purchases.

Example 2: (Groceries)

16. Sort the results according to lift, and show only the first 5 rules.

```
> inspect(sort(rules, by="lift", decreasing=TRUE)[1:5])
    lhs                                rhs                          support     confidence lift     count
[1] {bottled beer,red/blush wine} => {liquor}                   0.001931876 0.3958333  35.71579 19
[2] {hamburger meat,soda}         => {Instant food products}    0.001220132 0.2105263  26.20919 12
[3] {ham,white bread}             => {processed cheese}         0.001931876 0.3800000  22.92822 19
[4] {bottled beer,liquor}         => {red/blush wine}           0.001931876 0.4130435  21.49356 19
[5] {Instant food products,soda}  => {hamburger meat}           0.001220132 0.6315789  18.99565 12
```

- A lift value > 1 indicates that items in RHS are more likely to be purchases with items on LHS.
- A lift value < 1 indicates that items in RHS are unlikely to be purchased with items in LHS.
- In the above first rule, we can conclude that liquor are purchased 35 time more with bottle beer, red/blush wine than it being purchased alone.

Example 2: (Groceries)

17. Product recommendations, let's look at the top product recommendations for people who purchased whole milk.

```
> rules2 = apriori(Groceries, parameter = list(supp = 0.001, conf = 0.2, maxlen=3),
+                  appearance = list(default="rhs",lhs="whole milk"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.2    0.1    1 none FALSE            TRUE       5   0.001      1      3  rules FALSE

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

Absolute minimum support count: 9

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules2
set of 3 rules
```

Example 2: (Groceries)

18. Sort the rules in part (17), and interpret the results.

```
> inspect(sort(rules2, by="lift", decreasing=TRUE))
    lhs                  rhs                    support    confidence lift     count
[1] {whole milk} => {yogurt}           0.05602440 0.2192598  1.571735 551
[2] {whole milk} => {other vegetables} 0.07483477 0.2928770  1.513634 736
[3] {whole milk} => {rolls/buns}       0.05663447 0.2216474  1.205032 557
```

- The top product recommendation for people who purchased whole milk are yogurt followed by other vegetables.

Example 2: (Groceries)

19. Visually, map out the rules in a graph.

We need another library called “arulesViz”.

```
> plot(rules2,method="graph",interactive=TRUE,shading=NA)
Error in plot.associations(rules2, method = "graph", interactive = TRUE,  :
  Needed package 'arulesViz' not installed or loaded!
```

- It didn’t work with me, please try it. You should get a map similar to the following