Class 7 - Association Rule

A -> B, or B given A

1. Support $P(A \cap B)$
2. Confidence $P(B|A) = \frac{P(A \cap B)}{P(A)}$
3. Lift $P(A|B) = \frac{P(A \cap B)}{P(A) * P(B)}$

Apriori

1. Frequent Itemsets: These are sets of items that appear together in a transaction more frequently than a specified threshold (called support).
2. Association Rules: These are rules of the form {A} → {B}, where:
   - A is an item or itemset (e.g., "Bread"),
   - B is another item or itemset (e.g., "Butter").
   - The rule suggests that if A is bought, Bis likely to be bought as well.

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

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

```
library(arulesViz)
```

```
data(Groceries)
head(Groceries)
```

```
## transactions in sparse format with
##  6 transactions (rows) and
##  169 items (columns)
```

```
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
```

```
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}
```

```
rules = apriori(Groceries, parameter = list(support = 0.05, confidence = 0.1))
```

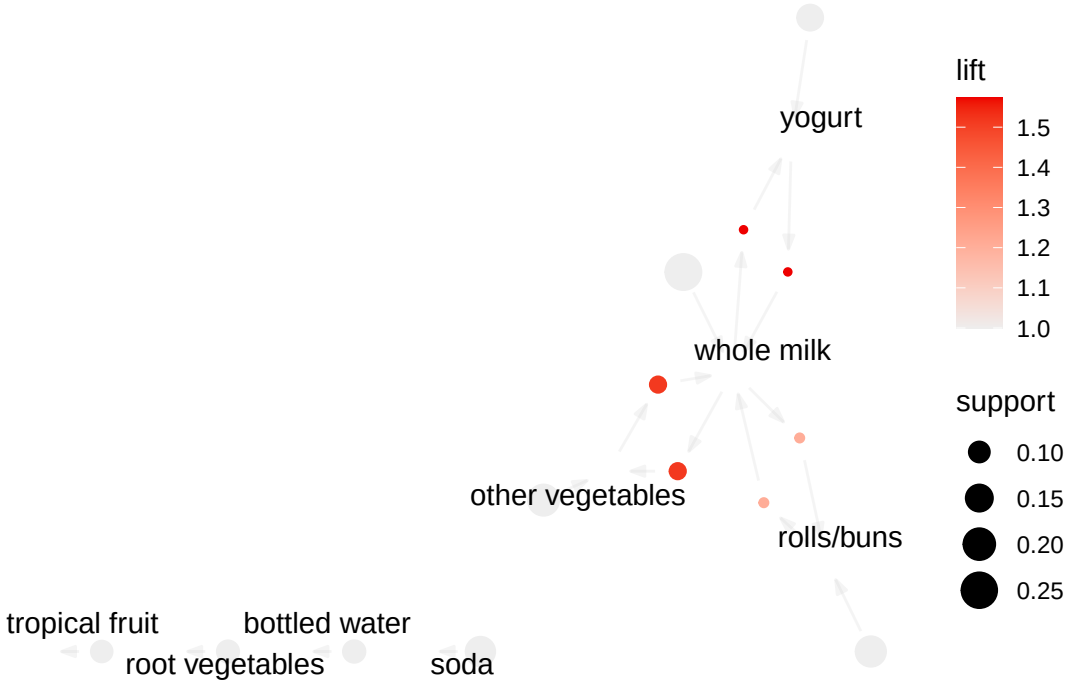
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.05      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 491
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [28 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [14 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
sorted_rules = sort(rules, by='lift', decreasing=T)
inspect(sorted_rules)
```

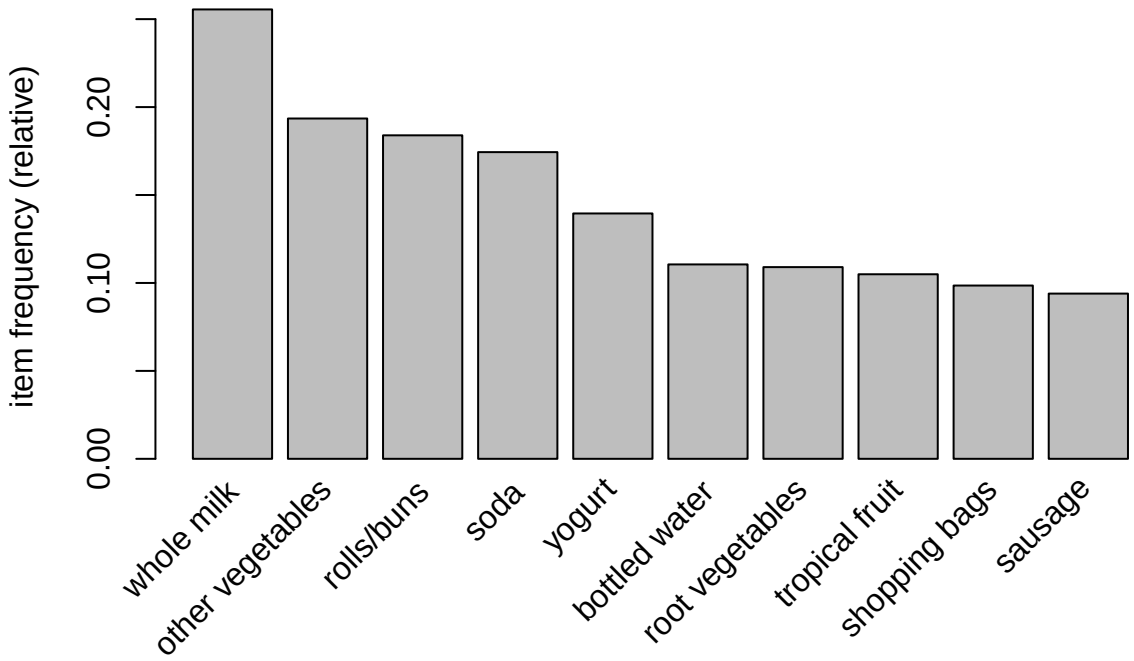
```
##      lhs                   rhs                support    confidence coverage
## [1]  {yogurt}           => {whole milk}       0.05602440 0.4016035  0.1395018
## [2]  {whole milk}       => {yogurt}           0.05602440 0.2192598  0.2555160
## [3]  {other vegetables} => {whole milk}       0.07483477 0.3867578  0.1934926
## [4]  {whole milk}       => {other vegetables} 0.07483477 0.2928770  0.2555160
## [5]  {rolls/buns}       => {whole milk}       0.05663447 0.3079049  0.1839349
## [6]  {whole milk}       => {rolls/buns}       0.05663447 0.2216474  0.2555160
## [7]  {}                 => {yogurt}           0.13950178 0.1395018  1.0000000
## [8]  {}                 => {rolls/buns}       0.18393493 0.1839349  1.0000000
## [9]  {}                 => {bottled water}    0.11052364 0.1105236  1.0000000
## [10] {}                 => {tropical fruit}   0.10493137 0.1049314  1.0000000
## [11] {}                 => {root vegetables}  0.10899847 0.1089985  1.0000000
## [12] {}                 => {soda}             0.17437722 0.1743772  1.0000000
## [13] {}                 => {other vegetables} 0.19349263 0.1934926  1.0000000
## [14] {}                 => {whole milk}       0.25551601 0.2555160  1.0000000
##      lift     count
## [1]  1.571735  551
## [2]  1.571735  551
## [3]  1.513634  736
## [4]  1.513634  736
## [5]  1.205032  557
## [6]  1.205032  557
## [7]  1.000000 1372
## [8]  1.000000 1809
## [9]  1.000000 1087
## [10] 1.000000 1032
## [11] 1.000000 1072
## [12] 1.000000 1715
## [13] 1.000000 1903
```

```
## [14] 1.000000 2513
```

```
plot(sorted_rules, method='graph')
```




```
itemFrequencyPlot(Groceries, topN=10)
```




```
library(ggplot2)
plot(sorted_rules,
  method = "graph",
  control = list(
    edges = ggraph::geom_edge_link(
      end_cap = ggraph::circle(4, "mm"),
      start_cap = ggraph::circle(4, "mm"),
      color = "green",
      arrow = arrow(length = unit(2, "mm"), angle = 20, type = "closed"),
      alpha = .8
    ),
    nodes = ggraph::geom_node_point(aes(size = support, color = lift)),
    nodetext = ggraph::geom_node_label(aes(label = label), alpha = .8, repel = TRUE)
  ),
  limit = 10
) +
  scale_color_gradient(low = "blue", high = "red") +
  scale_size(range = c(2, 10))
```

```
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.

## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 10 rows containing missing values or values outside the scale range
```

```
## (`geom_label_repel()`).
```