Association Rules

Rei Sanchez-Arias, Ph.D.

Introduction to Association Rules Mining

Pre-requisites

Checklist

☑ Load the `tidyverse` package

```
library(tidyverse)
```

☑ We will use the `arules` package to generate association rules and the `arulesViz` package to visualize them.

```
library(arules)
library(arulesViz)
```

☑ Check the documentation on the `arules` and `arulesViz` packages for additional information

Intro to Association Rules

As presented in:

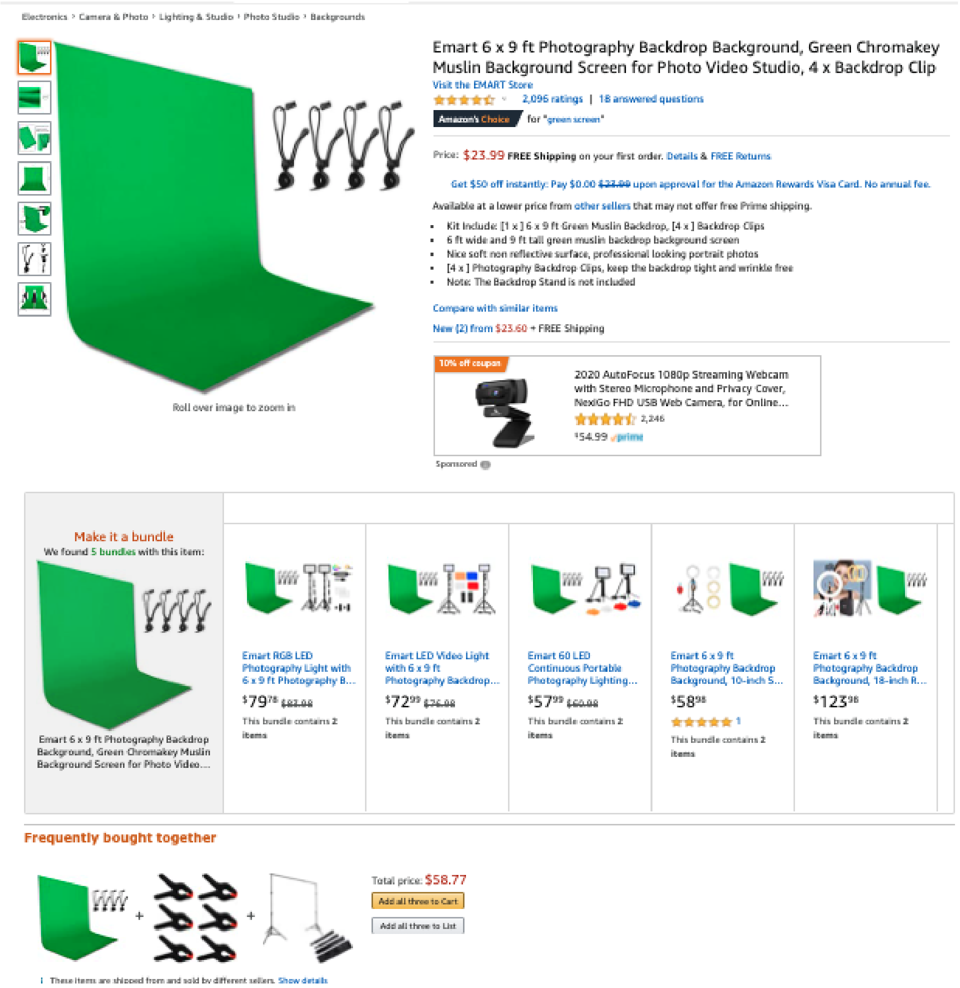
"Data Mining for Business Analytics"
by Schmueli et al.

and

"Data Mining and Machine Learning"
by Zaki and Meira

What are Association Rules?

- Study of “what goes with what”
- “Customers who bought X also bought Y”
- What symptoms go with what diagnosis
- Transaction-based or event-based



Also called **market basket analysis** and **affinity analysis**. Originated with study of customer **transactions** databases to determine **associations among items** purchased.

Common Terms

- "IF" part = **antecedent**
- "THEN" part = **consequent**
- **"itemset"** = the items (e.g., products) comprising the antecedent or consequent

Antecedent and consequent are **disjoint** (i.e., have no items in common)

Example: Phone Cases

Transaction	Case Color Purchased
1	{red, white, green}
2	{white, orange}
3	{white, blue}
4	{red, white, orange}
5	{red, blue}
6	{white, blue}
7	{white, orange}
8	{red, white, blue, green}
9	{red, white, blue}
10	{yellow}

Many rules are possible

For example: Transaction 1 supports several rules, such as

- “**If** red, **then** white” (“If a red faceplate is purchased, then so is a white one”)
- “**If** white, **then** red”
- “**If** red and white, **then** green”
- - several more

Frequent Itemsets

Ideally, we want to create all possible combinations of items

Problem: computation time grows exponentially as # items increases

Solution: consider only "**frequent** itemsets"

Criterion for frequent: **support**

Support for an itemset = # (or percent) of transactions that include an itemset

Support for a rule = # (or percent) of transactions that include **both** the antecedent and the consequent

Example: support for the itemset {red, white} is 4 out of 10 transactions, or 40%

Apriori Algorithm

Generating Frequent Itemsets

For k products...

1. User sets a minimum support criterion
2. Next, generate list of one-itemsets that meet the support criterion
3. Use the list of one-itemsets to generate list of two-itemsets that meet the support criterion
4. Use list of two-itemsets to generate list of three-itemsets
5. Continue up through k-itemsets

Measures of Rule Performance

Confidence: the % of antecedent transactions that also have the consequent itemset

Benchmark confidence = transactions with consequent as % of all transactions

Lift= confidence/(benchmark confidence)

Lift > 1 indicates a rule that is *useful* in finding consequent items sets (i.e., more useful than just selecting transactions randomly)

Alternate Data Format: Binary Matrix

Transaction	Red	White	Blue	Orange	Green	Yellow
1	1	1	0	0	1	0
2	0	1	0	1	0	0
3	0	1	1	0	0	0
4	1	1	0	1	0	0
5	1	0	1	0	0	0
6	0	1	1	0	0	0
7	0	1	0	1	0	0
8	1	1	1	0	1	0
9	1	1	1	0	0	0
10	0	0	0	0	0	1

Support for various itemsets

Transaction	Case Color Purchased
1	{red, white, green}
2	{white, orange}
3	{white, blue}
4	{red, white, orange}
5	{red, blue}
6	{white, blue}
7	{white, orange}
8	{red, white, blue, green}
9	{red, white, blue}
10	{yellow}

Itemset	Support (Count)
{red}	5
{white}	8
{blue}	5
{orange}	3
{green}	2
{red, white}	4
{red, blue}	3
{red, green}	2
{white, blue}	4
{white, orange}	3
{white, green}	2
{red, white, blue}	2
{red, white, green}	2

Process of rule selection

Generate all rules that meet specified support & confidence

- Find frequent itemsets (those with **sufficient support**)
- From these itemsets, generate rules with **sufficient confidence**

Example: rules from {red, white, green}

- {red, white} $\rightarrow$ {green} with confidence = 2/4 = 0.5

$$\frac{\text{support}(\{\text{red, white, green}\})}{\text{support }(\{\text{red, white}\})}$$

- {red, green} $\rightarrow$ {white} with confidence = 2/2 = 1.0

$$\frac{\text{support}(\{\text{red, white, green}\})}{\text{support }(\{\text{red, green}\})}$$

(An) Interpretation

- Lift ratio shows how effective the rule is in finding consequents (useful if finding particular consequents is important)
- Confidence shows the rate at which consequents will be found (useful in learning costs of promotion)
- Support measures overall impact

Caution: the role of chance

Random data can generate apparently interesting association rules. The more rules you produce, the greater this danger. Rules based on large numbers of records are less subject to this.

Checkpoint: quick recap

- Association rules (or affinity analysis, or market basket analysis) produce rules on associations between items from a database of transactions
- Sometimes used in recommender systems
- Most popular method is **Apriori algorithm**
- To reduce computation, we consider only "frequent" itemsets (=support)
- Performance of rules is measured by metrics such as confidence and lift

Some Technical Details

Motivation

Market basket analysis is an association rule method that identifies associations in *transactional data*. It is an **unsupervised machine learning** technique used for **knowledge discovery**. This analysis results in a set of association rules that **identify patterns of relationships among items**.

A rule can typically be expressed in the form:
{peanut butter, jelly} $\rightarrow$ {bread}

The above rule states that if both peanut butter and jelly are purchased, then bread is also likely to be purchased. **Transactional data** can be extremely large both in terms of the quantity of transactions and the number of items monitored. Given k items that can either appear or not appear in a set, there are 2^k possible itemsets that must be searched for rules.

Motivation (2)

Thus, even if a retailer only has 100 distinct items, he could have $2^{100} = 1.267651 \times 10^{30}$ itemsets to evaluate, which is quite an impossible task. However, a **smart rule learner** algorithm can take advantage of the fact that in reality, many of the potential item combinations are rarely found in practice.

For example, if a retailer sells both paints and dairy products, a set of {paint, butter} are *extremely unlikely to be common*. By ignoring these rare cases, it makes it possible to limit the scope of the search for rules to a much more manageable size.

R. Agrawal and R. Srikant

R. Agrawal and R. Srikant introduced the **apriori algorithm**: it utilizes a simple prior belief (hence the name a priori) about the properties of frequent items. Using this a priori belief, **all subsets of frequent items must also be frequent**. This makes it possible to limit the number of rules to search for.

Fast Algorithms for Mining Association Rules

Rakesh Agrawal   Ramakrishnan Srikant*

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Abstract

We consider the problem of discovering association rules between items in a large database of sales transactions. We present two new algorithms for solving this problem that are fundamentally different from the known algorithms. Experiments with synthetic as well as real-life data show that these algorithms outperform the known algorithms by factors ranging from three for small problems to more than an order of magnitude for large problems. We also show how the best features of the two proposed algorithms can be combined into a hybrid algorithm, called AprioriHybrid. Scale-up experiments show that AprioriHybrid scales linearly with the number of transactions. AprioriHybrid also has excellent scale-up properties with respect to the transaction size and the number of items in the database.

Presented at the 20th int. conf. very large databases, VLDB, 1994

For example, the set {paint, butter} can only be frequent if {paint} and {butter} both occur frequently. Conversely, if neither {paint} nor {butter} are frequent, then any set containing these two items can be excluded from the search.

Support and Confidence

Let $I = \{i_1, i_2, \ldots, i_d\}$ be the set of all items in a market basket data and $T = \{t_1, t_2, \ldots, t_N\}$ be the set of all transactions. Each transaction t_i contains a subset of items chosen from I. In association analysis, a collection of zero or more items is termed an **itemset**. If an itemset contains k items, is called a k-itemset.

A transaction t_j is said to contain an itemset X if X is a subset of t_j. An important property of an itemset is its *support count*, which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, $\sigma(X)$, for an itemset X can be stated as follows:

$$\sigma(X) = |t_i\colon X \subseteq t_i, \quad t_i \in T|$$

where the symbol $|\cdot|$ denotes the number of elements in a set.

Support and Confidence (2)

Support : This measures how frequently an itemset occurs in the data:

$$\text{Support}(X) = \frac{\text{Count}(X)}{N} = \frac{\sigma(X)}{N}$$

where X represents an item and N represents the total number of transactions.

An itemset X is called *frequent* if the support is greater than some user-defined threshold (sometimes referred to as *minsup*)

Confidence : This measures the algorithm's predictive power or accuracy. It is calculated as the support of item X and Y divided by the support of item X.

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

Support and Confidence (3)

Confidence measures how frequently items in Y appear in transactions containing X

$$\mathrm{Confidence}(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Important to notice that $\mathrm{Confidence}(X \to Y) \neq \mathrm{Confidence}(Y \to X)$

Lift: the lift of a rule is defined as

$$\mathrm{Lift}(X \to Y) = \frac{\mathrm{Support}(X \cup Y)}{\mathrm{Support}(X) \cdot \mathrm{Support}(Y)}$$

Greater lift values ($\gg 1$) indicate stronger associations.

Small Example

Transaction	Purchases
1	{flowers, get well card, soda}
2	{toy bear, flowers, balloons, candy}
3	{get well card, candy, flowers}
4	{toy bear, balloons, soda}
5	{flowers, get well card, soda}

$$\text{Confidence}(\text{get well card} \rightarrow \text{flowers}) = \frac{\text{Support}(\text{get well card} \cup \text{flowers})}{\text{Support}(\text{get well card})} = \frac{0.6}{0.6} = 1.0$$

$$\text{Confidence}(\text{flowers} \rightarrow \text{get well card}) = \frac{\text{Support}(\text{flowers} \cup \text{get well card})}{\text{Support}(\text{flowers})} = \frac{0.6}{0.8} = 0.75$$

This means that a purchase of a "get well card" results in a purchase of flowers 100% of the time, while a purchase of flowers results in a purchase of a get well card 75% of the time.

Apriori Algorithm: how it works

(1) Identify all itemsets that meet a minimum support threshold

This process occurs in multiple iterations. Each successive iteration evaluates the support of storing a set of **increasingly large items**. The first iteration involves evaluating the set of of 1-itemsets. The second iteration involves evaluating the set of 2-itemsets, and so on. The result of each iteration k is a set of k-itemsets that meet the minimum threshold. All itemsets from iteration k are combined in order to **generate candidate itemsets** for evaluation in iteration $k+1$.

The apriori principle can eliminate some of the items before the next iteration begins. For example, if {A}, {B}, and {C} are frequent in iteration 1, but {D} is not, then the second iteration will only consider the itemsets {A, B}, {A, C}, and {B, C}.

(2) Create rules from these items that meet a minimum confidence threshold.

Apriori Algorithm Example using the `arules` package

Groceries

We use the `Groceries` dataset from the R `arules` package. The `Groceries` dataset is collected from 30 days of real-world point-of-sale transactions of a grocery store.

The data contain 9835 transactions, or about 328 transactions per day. If we remove brands and just consider product type, it will give total 169 items.

- Any guesses about which types of items might be purchased together?
- Will wine and cheese be a common pairing? Bread and butter? Milk and eggs?

```
data(Groceries)
Groceries
```

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)
```

```r
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
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9
##   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
```

Things to notice

- Density, 0.026 means 2.6% are non zero matrix cells
- Matrix has 9835 times 169, i.e. 1662115 cells. Hence 9835 times 169 times 0.02609146, i.e. 43367, items were purchased
- Whole milk appeared 2513 times out of 9835 transactions, means 0.26 percent of transactions.
- Average transaction contained 43367/9835 = 4.409456 items
- A total of 2159 transactions contained only a single item, while one transaction had 32 items.
- The first quartile and median purchase size are 2 and 3 items respectively, implying that 25 percent of transactions contained two or fewer items and about half contained around three items.
- The mean of 4.409 matches the value we calculated manually.

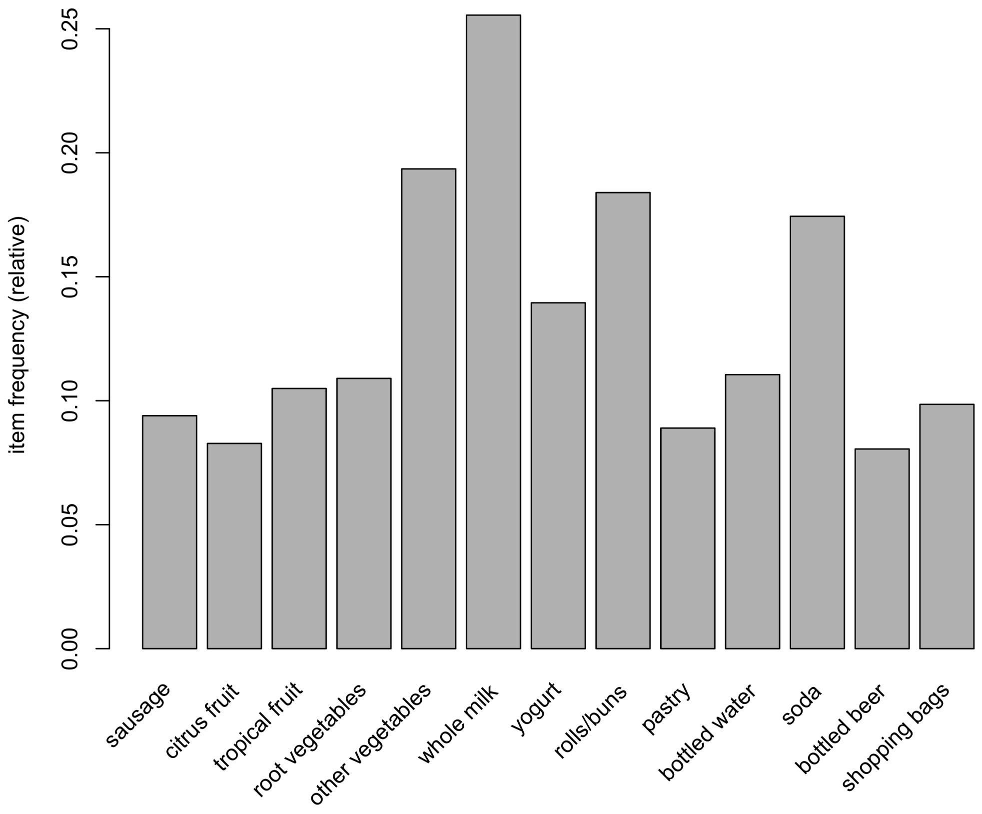
Find item frequency

```r
itemFrequency(Groceries[ , 1:5])
```

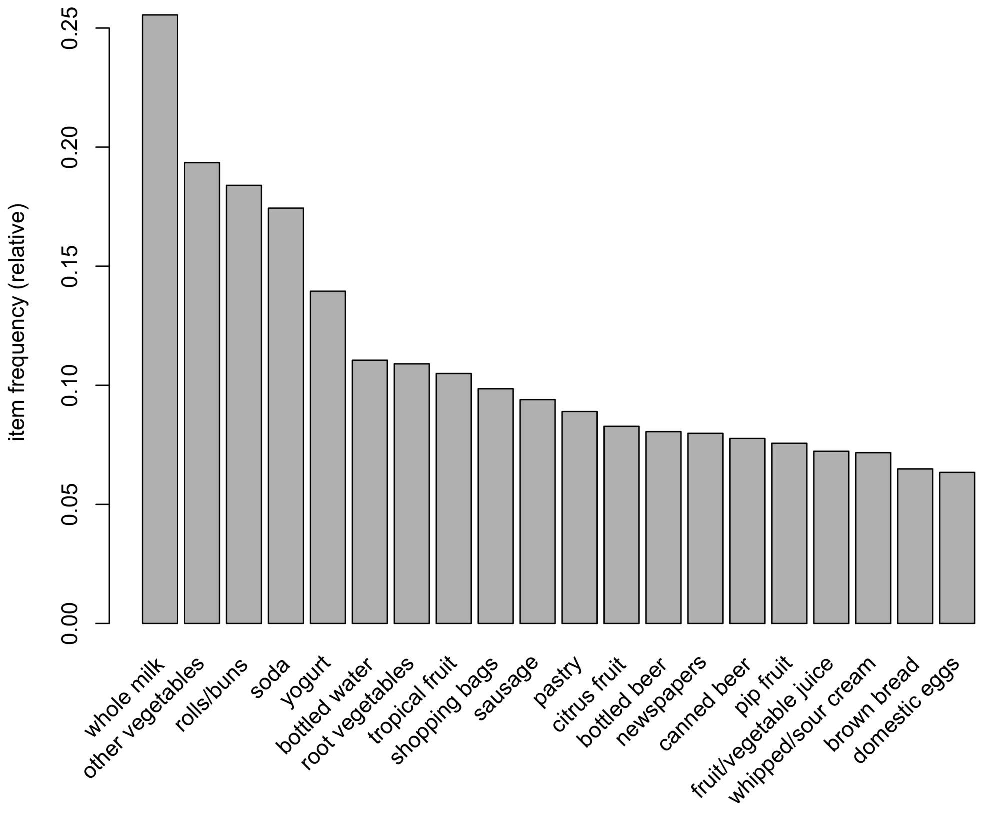
```
## frankfurter     sausage  liver loaf         ham        meat
## 0.058973055 0.093950178 0.005083884 0.026029487 0.025826131
```

Item frequency plots

```r
itemFrequencyPlot(Groceries, support = 0.08)
```




```r
itemFrequencyPlot(Groceries, topN = 20)
```



```r
apriori(Groceries)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ex
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules TRU
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 983
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [8 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## set of 0 rules
```

Why 0 rules?

With the default support value of 0.1, an item must have appeared in at least $0.1 \times 9385 = 938$ transactions. Only 8 items appeared those many times, so no rules were generated

Support: We might set a support by thinking of the minimum number of transactions we would need. For example, if an item is purchased three times a day (about 90 times) then it may be worth taking a look at. In such case, the support will be 90 out of 9835 transactions, i.e. 0.009

Confidence: We will set a confidence threshold of 0.25, which means that in order to be included in the results, the rule has to be correct at least 25 percent of the time. This will eliminate the most unreliable rules while allowing some room for us to modify behavior with targeted promotions.

In addition, we set `minlen = 2` to eliminate rules that contain fewer than two items.

```r
grules <-  apriori(Groceries,
                   parameter = list(support = 0.009,
                                    confidence = 0.25,
                                    minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ex
##        0.25    0.1    1 none FALSE            TRUE       5   0.009      2     10  rules TRU
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 88
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [93 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [224 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Evaluating performance

```r
summary(grules)
```

```
## set of 224 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3
## 111 113
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.504   3.000   3.000
##
## summary of quality measures:
##     support           confidence        coverage             lift            count
##  Min.   :0.009049   Min.   :0.2513   Min.   :0.01464   Min.   :0.9932   Min.   : 89.0
##  1st Qu.:0.010066   1st Qu.:0.2974   1st Qu.:0.02725   1st Qu.:1.5767   1st Qu.: 99.0
##  Median :0.012303   Median :0.3603   Median :0.03711   Median :1.8592   Median :121.0
##  Mean   :0.016111   Mean   :0.3730   Mean   :0.04574   Mean   :1.9402   Mean   :158.5
##  3rd Qu.:0.018480   3rd Qu.:0.4349   3rd Qu.:0.05541   3rd Qu.:2.2038   3rd Qu.:181.8
##  Max.   :0.074835   Max.   :0.6389   Max.   :0.25552   Max.   :3.7969   Max.   :736.0
##
## mining info:
##       data ntransactions support confidence
##  Groceries          9835   0.009       0.25
```

Inspecting rules by support

Use the `inspect()` function to display the top N frequent itemsets sorted by support

```r
inspect(head(sort(grules, by = "support"), 10))
```

```
##      lhs                   rhs                  support    confidence coverage  lift     count
## [1]  {other vegetables} => {whole milk}         0.07483477 0.3867578  0.1934926 1.513634 736
## [2]  {whole milk}       => {other vegetables}   0.07483477 0.2928770  0.2555160 1.513634 736
## [3]  {rolls/buns}       => {whole milk}         0.05663447 0.3079049  0.1839349 1.205032 557
## [4]  {yogurt}           => {whole milk}         0.05602440 0.4016035  0.1395018 1.571735 551
## [5]  {root vegetables}  => {whole milk}         0.04890696 0.4486940  0.1089985 1.756031 481
## [6]  {root vegetables}  => {other vegetables}   0.04738180 0.4347015  0.1089985 2.246605 466
## [7]  {yogurt}           => {other vegetables}   0.04341637 0.3112245  0.1395018 1.608457 427
## [8]  {tropical fruit}   => {whole milk}         0.04229792 0.4031008  0.1049314 1.577595 416
## [9]  {tropical fruit}   => {other vegetables}   0.03589222 0.3420543  0.1049314 1.767790 353
## [10] {bottled water}    => {whole milk}         0.03436706 0.3109476  0.1105236 1.216940 338
```

Type of rules

A common approach is to take the result of learning association rules and divide them into three categories:

- **Actionable** : The goal of a market basket analysis is to find actionable associations, or rules that provide a clear and useful insight. Some rules are clear, others are useful; it is less common to find a combination of both of these factors.
- **Trivial** : Any rules that are so obvious that they are not worth mentioning, they are clear, but not useful.
- **Inexplicable** : If the connection between the items is so unclear that figuring out how to use the information for action would require additional research.

Inspect rules by lift

We can also inspect the rules by lift.

```r
inspect(head(sort(grules, by = "lift")))
```

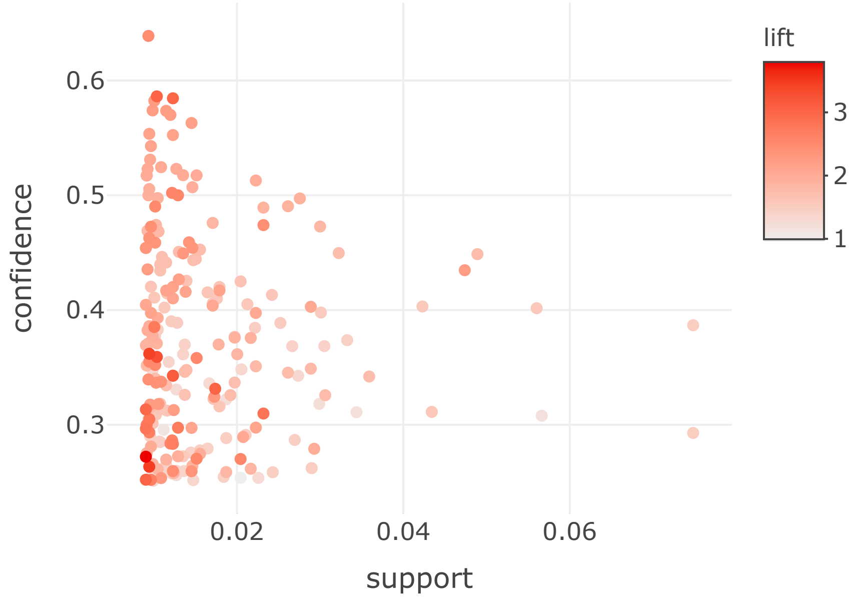
```
##                                  lhs                    rhs     support confidence     lift count
## [1]                        {berries} {whipped/sour cream} 0.009049314  0.2721713 3.796886    89
## [2] {tropical fruit,other vegetables}          {pip fruit} 0.009456024  0.2634561 3.482649    93
## [3]      {pip fruit,other vegetables}     {tropical fruit} 0.009456024  0.3618677 3.448613    93
## [4]   {citrus fruit,other vegetables}    {root vegetables} 0.010371124  0.3591549 3.295045   102
## [5] {tropical fruit,other vegetables}    {root vegetables} 0.012302999  0.3427762 3.144780   121
## [6] {tropical fruit,other vegetables}       {citrus fruit} 0.009049314  0.2521246 3.046248    89
```

The first rule, with a lift of 3.796886, implies that people who buy berries are nearly four times more likely to buy whipped/sour cream than the typical customer

Rules generation and visualization

Use `plot(grules)` to display the scatterplot of the rules, where the horizontal axis is the support, the vertical axis is the confidence, and the shading is the lift.

```
plot(grules)
```



The scatterplot shows that, of the rules generated from the `Groceries` dataset, the highest lift occurs at a low support and a low confidence.

Another set of rules

Let us now assume that the minimum support threshold is now set to a lower value 0.001, and the minimum confidence threshold is set to 0.6. A lower minimum support threshold allows more rules to show up.

The following code creates 2918 rules from all the transactions in the `Groceries` dataset that satisfy both the minimum support and the minimum confidence.

```
rules <- apriori(Groceries,
                 parameter = list(support = 0.001,
                                  confidence = 0.6,
                                  target = "rules"))
```

```r
summary(rules)
```

```
## set of 2918 rules
##
## rule length distribution (lhs + rhs):sizes
##    2    3    4    5    6
##    3  490 1765  626   34
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   4.000   4.068   4.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift            count
##  Min.   :0.001017   Min.   :0.6000   Min.   :0.001017   Min.   : 2.348   Min.   :10.00
##  1st Qu.:0.001118   1st Qu.:0.6316   1st Qu.:0.001525   1st Qu.: 2.668   1st Qu.:11.00
##  Median :0.001220   Median :0.6818   Median :0.001830   Median : 3.168   Median :12.00
##  Mean   :0.001480   Mean   :0.7028   Mean   :0.002157   Mean   : 3.450   Mean   :14.55
##  3rd Qu.:0.001525   3rd Qu.:0.7500   3rd Qu.:0.002339   3rd Qu.: 3.692   3rd Qu.:15.00
##  Max.   :0.009354   Max.   :1.0000   Max.   :0.014642   Max.   :18.996   Max.   :92.00
##
## mining info:
##       data ntransactions support confidence
##  Groceries          9835   0.001        0.6
```

Higher lift

Let us create a smaller set of rules with the top 5 rules sorted by their value of "lift"

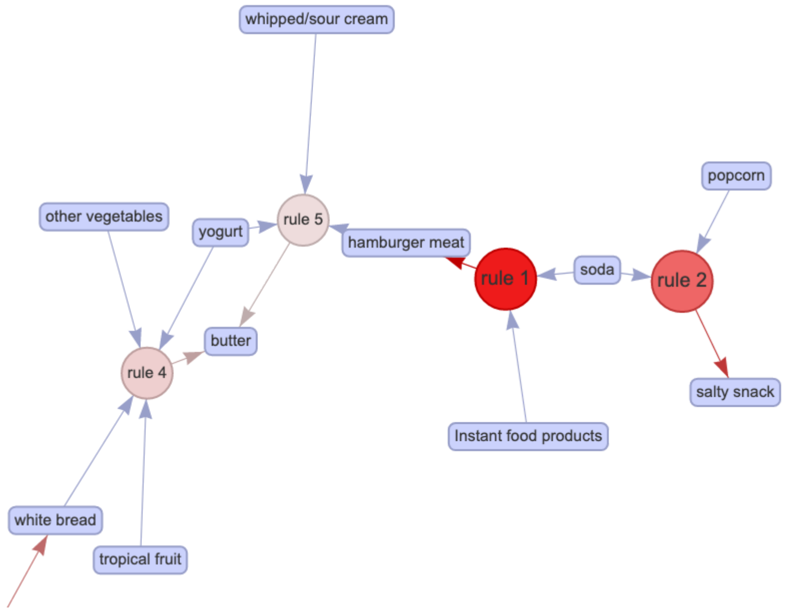
```
high_lift <- head(sort(rules, by="lift"), 5)
```

```r
inspect(head(sort(rules, by="lift"), 10))
```

```
##      lhs                          rhs                   support confidence    coverage      lift count
## [1]  {Instant food products,                                                                          
##       soda}                    => {hamburger meat}     0.001220132  0.6315789 0.001931876 18.995654    12
## [2]  {soda,                                                                                           
##       popcorn}                 => {salty snack}        0.001220132  0.6315789 0.001931876 16.697793    12
## [3]  {ham,                                                                                            
##       processed cheese}        => {white bread}        0.001931876  0.6333333 0.003050330 15.045491    19
## [4]  {tropical fruit,                                                                                 
##       other vegetables,                                                                               
##       yogurt,                                                                                         
##       white bread}             => {butter}             0.001016777  0.6666667 0.001525165 12.030581    10
## [5]  {hamburger meat,                                                                                 
##       yogurt,                                                                                         
##       whipped/sour cream}      => {butter}             0.001016777  0.6250000 0.001626843 11.278670    10
## [6]  {tropical fruit,                                                                                 
##       other vegetables,                                                                               
##       whole milk,                                                                                     
##       yogurt,                                                                                         
##       domestic eggs}           => {butter}             0.001016777  0.6250000 0.001626843 11.278670    10
## [7]  {liquor,                                                                                         
##       red/blush wine}          => {bottled beer}       0.001931876  0.9047619 0.002135231 11.235269    19
## [8]  {other vegetables,                                                                               
##       butter,                                                                                         
##       sugar}                   => {whipped/sour cream} 0.001016777  0.7142857 0.001423488  9.964539    10
## [9]  {whole milk,                                                                                     
##       butter,                                                                                         
##       hard cheese}             => {whipped/sour cream} 0.001423488  0.6666667 0.002135231  9.300236    14
## [10] {tropical fruit,                                                                                 
##       other vegetables,                                                                               
##       butter,                                                                                         
##       fruit/vegetable juice}   => {whipped/sour cream} 0.001016777  0.6666667 0.001525165  9.300236    10
```

Graph Visualization: top 5 rules

Select by id



Graph Visualization (cont.)

The following code provides a visualization of the **top five rules** with the highest lift. In the graph, the arrow always points from an item on the LHS to an item on the RHS.

```
plot(high_lift, method="graph", control=list(type="items",  engine = "htmlwidget"))
```

For example, the arrows that connect ham, processed cheese, and white bread suggest:

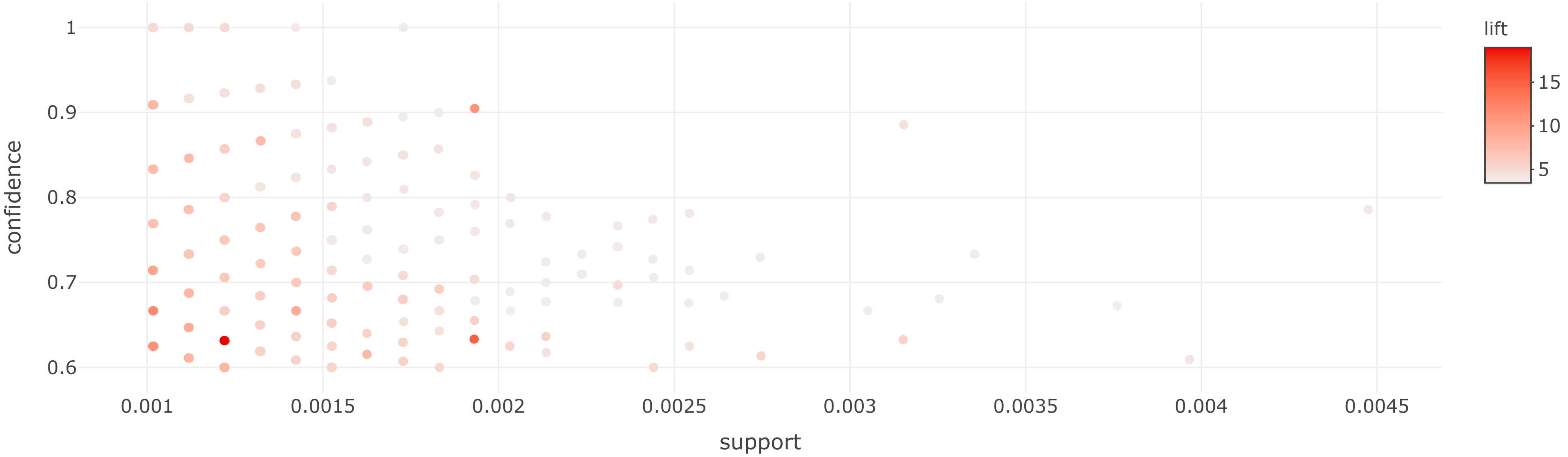
$$\{\text{ham, processed cheese}\} \longrightarrow \{\text{white bread}\}.$$

The size of a circle indicates the support of the rules ranging from 0.001 to 0.002. The color (or shade) represents the lift, which in this case ranges from 11.279 to 18.996. The rule with the highest lift is

$$\{\text{Instant food products,soda}\} \longrightarrow \{\text{hamburger meat}\}.$$

Confidence vs Support plot

```
plot(rules)
```



Inspect rules

	LHS	RHS	support	confidence	coverage	lift	count
	All	All	All	All	All	All	All
[1]	{honey}	{whole milk}	0.001	0.733	0.002	2.870	11.000
[2]	{cereals}	{whole milk}	0.004	0.643	0.006	2.516	36.000
[3]	{rice}	{whole milk}	0.005	0.613	0.008	2.400	46.000
[4]	{liver loaf,yogurt}	{whole milk}	0.001	0.667	0.002	2.609	10.000
[5]	{tropical fruit,curd cheese}	{other vegetables}	0.001	0.667	0.002	3.445	10.000
[6]	{curd cheese,rolls/buns}	{whole milk}	0.001	0.625	0.002	2.446	10.000

Previous 1 2 3 4 5 ... 487 Next