C04

關聯性分析
Association Rules

吳漢銘
國立政治大學 統計學系

https://hmwu.idv.tw

Market Basket Analysis

Market Basket Example

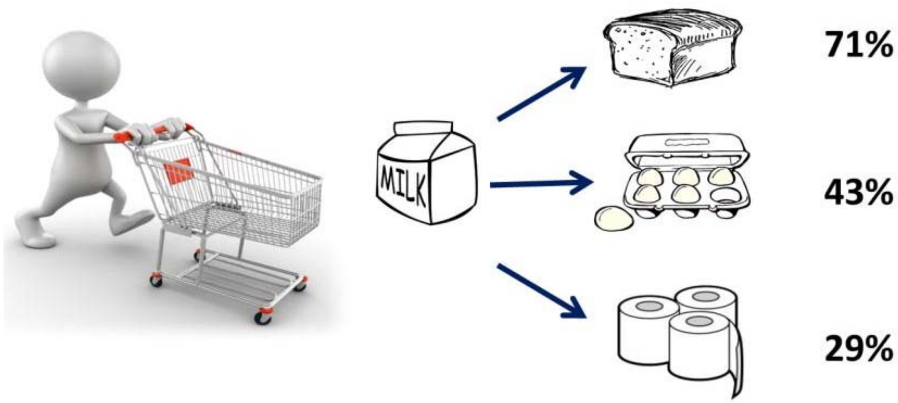
Where should detergents be placed in the Store to maximize their sales?

Are window cleaning products purchased when detergents and orange juice are bought together?

Is soda typically purchased with bananas? Does the brand of soda make a difference?

How are the demographics of the neighborhood affecting what customers are buying?

http://www.analyticsvidhya.com/blog/2014/08/effective-cross-selling-market-basket-analysis/

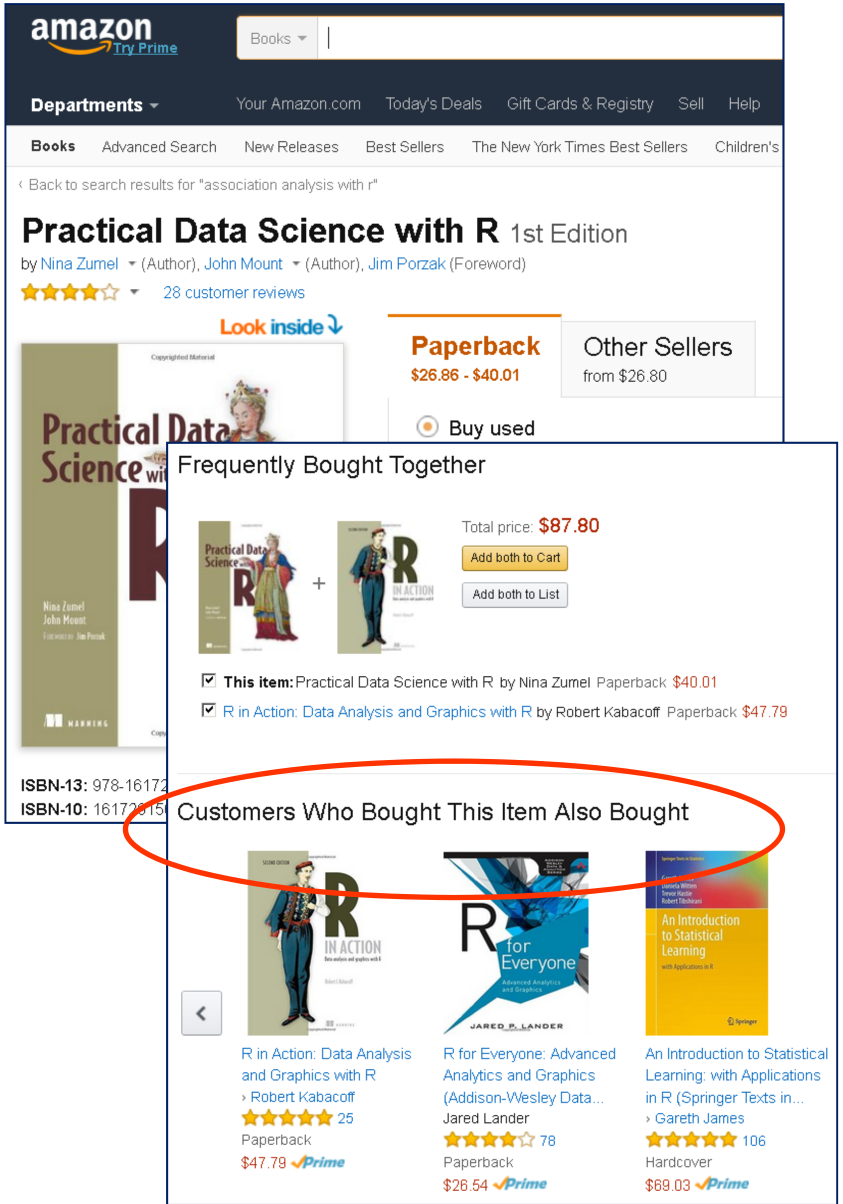


Of transactions that included milk:

- 71% included bread
- 43% included eggs
- 29% included toilet paper

https://blogs.adobe.com/digitalmarketing/wp-content/uploads/2013/08/pic1.jpg

應用實例


amazon
Try Prime
Books
Departments
Your Amazon.com
Today's Deals
Gift Cards & Registry
Sell
Help
Books
Advanced Search
New Releases
Best Sellers
The New York Times Best Sellers
Children's
Back to search results for "association analysis with r"
Practical Data Science with R 1st Edition
by Nina Zumel (Author), John Mount (Author), Jim Porzak (Foreword)
28 customer reviews
Look inside
Paperback
$26.86 - $40.01
Other Sellers
from $26.80
Buy used
Frequently Bought Together
Total price: $87.80
Add both to Cart
Add both to List
This item: Practical Data Science with R by Nina Zumel Paperback $40.01
R in Action: Data Analysis and Graphics with R by Robert Kabacoff Paperback $47.79
Customers Who Bought This Item Also Bought
R in Action: Data Analysis and Graphics with R
› Robert Kabacoff
25
Paperback
$47.79 Prime
R for Everyone: Advanced Analytics and Graphics (Addison-Wesley Data...
Jared Lander
78
Paperback
$26.54 Prime
An Introduction to Statistical Learning: with Applications in R (Springer Texts in...
› Gareth James
106
Hardcover
$69.03 Prime


PChome 24h購物
原廠・定焦鏡
鏡頭
SONY鏡頭
CANON 鏡頭
SONY E 24mm F1.8 ZA 蔡司鏡 公司貨
《大光圈廣角蔡司鏡
■瘋狂送UV保護鏡
■NEX首款蔡司鏡頭
■視角等同35-mm格式的36 mm
■內建對焦馬達
■輕巧方便攜帶
■絕美大光圈人像鏡
■SONY公司貨
網路價$34380
看此商品的人也看了...
SONY E 24mm F1.8 ZA 蔡司鏡 公司貨
網路價$30590
SONY SEL24F18Z 卡爾蔡司 T* E 24mm
網路價$33880
SONY 數位單眼相機 ILCE-6300L 單鏡組(公
網路價$37980

Market Basket Analysis

- **Market Basket Analysis** is one of the Data Mining approaches
  - to find associations and correlations between the different items that customers place in their shopping basket.
  - to help market owner to have much better opportunity to make a profit by controlling the order of products and marketing.

- **Retailers leverage Market Basket Analysis**
  - to provide a window into consumer shopping behavior, revealing how consumers select products, make spending tradeoffs, and group items in a shopping cart.
  - to understand how baskets are built. It can help retailers merchandise more effectively by leveraging market basket dynamics in pricing and promotion decisions.

R. Agrawal, T. Imieliński and A. Swami, "Mining Association Rule between Sets of Items in Large Databases," The ACM SIGMOD International Conference on Management of Data, pp. 207-216, May 1993.
(被引用 19551 次)

Association Rule Mining

- The ideas of Association Rule Learning (also called Association Rule Mining) come from the market basket analysis.

Mining association rules between sets of items in large databases
dl.acm.org/citation.cfm?id=170072 ▾
由 R Agrawal 著作 - 1993 - 被引用 17167 次 - 相關文章
Mining association rules between sets of items in large databases ... Proceedings of the 1993 ACM SIGMOD international conference on Management of data.

- **AR mining**:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Transaction ID (TID)	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

mining

Rule
{bread} ⇒ {milk}
{soda} ⇒ {chips}
{bread} ⇒ {jam}

Association Rule Mining

- **Formalizing the problem**:
  - **Transaction Database T**: a set of transactions **T** $= \{t_1, t_2, \ldots, t_n\}$.
  - Each transaction contains a set of items **I** (itemset)(項集).
  - An **itemset** is a collection of items **I** $= \{i_1, i_2, \ldots, i_m\}$.
  - ***k*-itemset**: an itemset that contains *k* items.

- Association rules are rules presenting association or correlation between itemsets.

- An association rule is in the form of $A \Rightarrow B$, where A and B are two disjoint itemsets, referred to respectively as the **lhs** (left-hand side) (先決條件) and **rhs** (right-hand side) (對應的連結結果) of the rule.

Definition: Frequent Itemset

$P(A)$ is the percentage (or probability) of cases containing A.

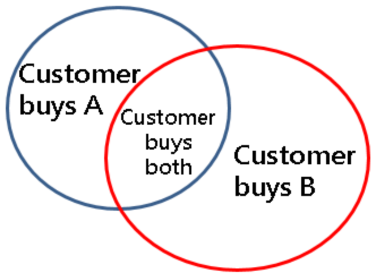
- Support count (σ)
  - Frequency of occurrence of an itemset.
  - σ({Milk, Bread}) = 3, σ({Soda, Chips}) = 4.
- Support (s) (支援度)
  - The occurring frequency of the rule.
  - The percentage of transactions that contains both itemsets A and B.

    means "and"
  - Support(A ⇒ B) = P(A ∩ B)
  - s({Milk, Bread}) = 3/8; s({Soda, Chips}) = 4/8
- Frequent itemset (頻繁項集):
  - s(itemset) ≧ *minsup* (minimum support) threshold.
  - Items that frequently appear together.
  - The strength of the association.

Transaction ID (TID)	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\text{ of transactions}} = 0.375$$

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$



Confidence and Lift

- **Confidence (c) (可靠度)**:
  - the percentage of cases containing A that also contain B.
  - confident(A ⇒ B) = P(B | A) = P(A ∩ B)/P(A)
  - confident(A ⇒ B) ≧ *mincon* (minimum confident)
- **Lift (提昇度)**:
  - the ratio of confidence to the percentage of cases containing B.
  - lift(A ⇒ B) = P(B | A)/P(B)
    = confident(A ⇒ B) / P(B)
    = P(A ∩ B)/P(A)P(B)
  - lift(A ⇒ B) = 1, A和B相互獨立，A對B出現的可能性沒有提昇作用。
  - lift(A ⇒ B) > 1，表示A對B的提昇程度愈大，連結性愈強。

Rule: $X \Rightarrow Y$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

NOTE: There are many other interestingness measures, such as chi-square, conviction, gini and leverage. An introduction to over 20 measures can be found in Tan, P.-N., Kumar, V., and Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. In KDD ’02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 32–41, New York, NY, USA. ACM Press.

Find a Rule

The following rules are binary partitions of the same itemset: {Milk, Bread, Jam}

- {Bread, Jam} ⇒ {Milk}, s=0.375, c=3/4=0.75
- {Milk, Jam} ⇒ {Bread}, s=0.375, c=0.75
- {Bread} ⇒ {Milk, Jam}, s=0.375, c=0.75
- {Jam} ⇒ {Bread, Milk}, s=0.375, c=0.6
- {Milk} ⇒ {Bread, Jam}, s=0.375, c=0.5

Transaction ID (TID)	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

- Rules originating from the same itemset have identical support but can have different confidence.
- Given a set of transactions **T**, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold.
  - confidence ≥ *minconf* threshold.

Mining Association Rules

- **Brute-force approach**:
  - List all possible association rules.
  - Compute the support and confidence for each rule.
  - Prune rules that fail the *minsup* and *minconf* thresholds.
  - Brute-force approach is computationally prohibitive!

- **Two step approach**:

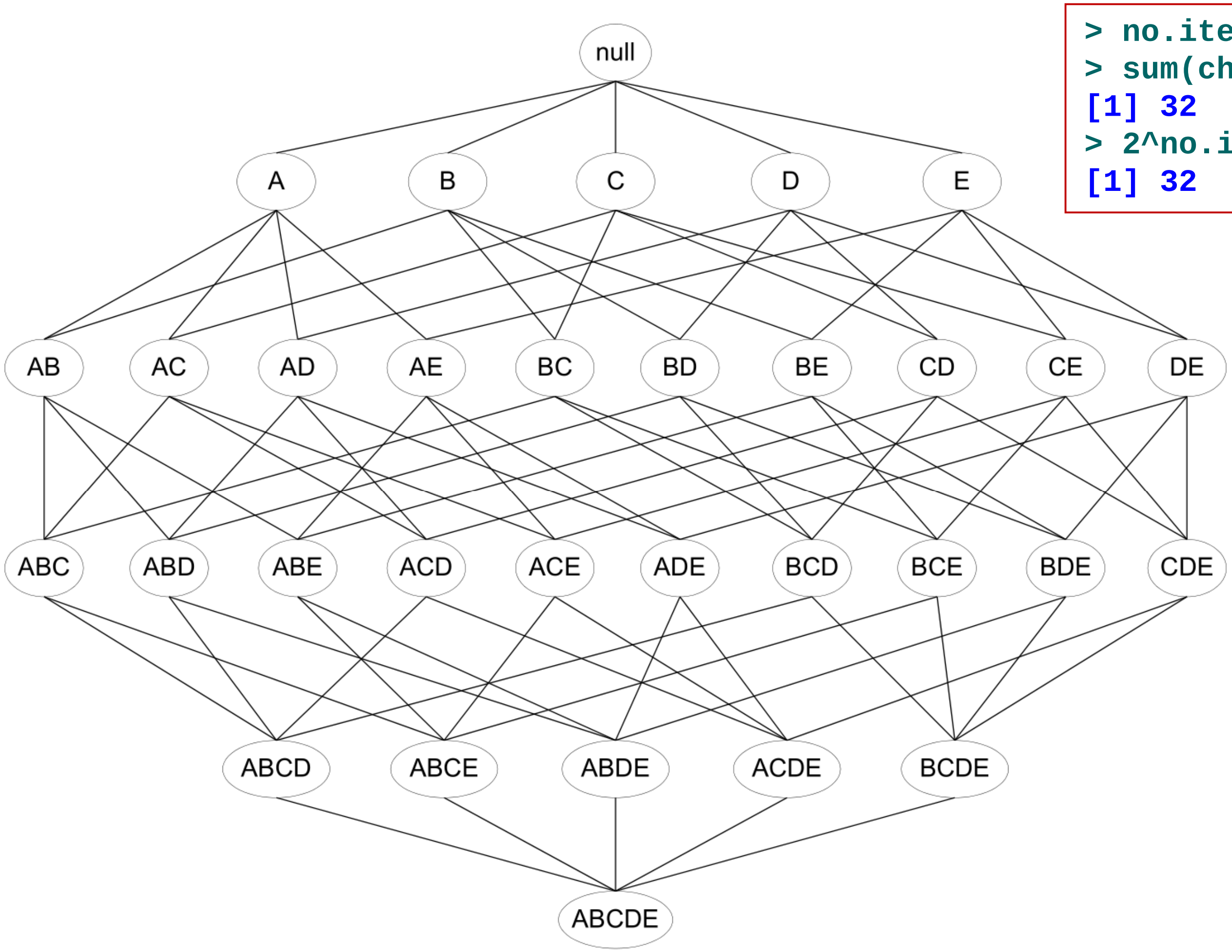
  Step (1): Frequent Itemset Generation:
  - Generate all itemsets whose support >= *minsup*

  Step (2): Rule Generation:
  - Generate high confidence rules from frequent itemset.
  - Each rule is a binary partitioning of a frequent itemset.

- Frequent itemset generation is computationally expensive.

Step (1): Frequent Itemset Generation

- Given d items, there are 2^d possible candidate itemsets.



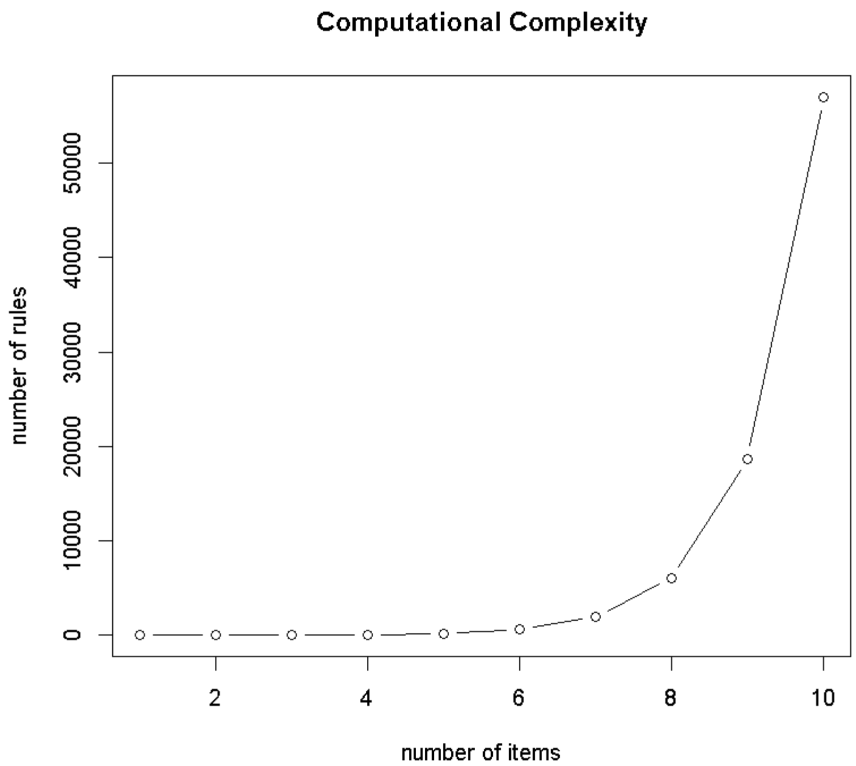

```
> no.item <- 5
> sum(choose(no.item, 0:no.item))
[1] 32
> 2^no.item
[1] 32
```

Source: (1) Prof. Pier Luca Lanzi, Association Rule Basics, Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)
(2) Tan,Steinbach, Kumar Introduction to Data Mining

Total Number of Possible Association Rules

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:

$$\sum_{k=1}^{d-1}\left[\binom{d}{k}\times\sum_{j=1}^{d-k}\binom{d-k}{j}\right]=3^d-2^{d+1}+1$$



Frequent Itemset Generation Strategies

- Reduce the number of candidates (M).
  - Complete search: $M=2^d$.
  - Use pruning techniques to reduce M.
- Reduce the number of transactions (N).
  - Reduce size of N as the size of itemset increases.
- Reduce the number of comparisons (NM).
  - Use efficient data structures to store the candidates or transactions.
  - No need to match every candidate against every transaction.

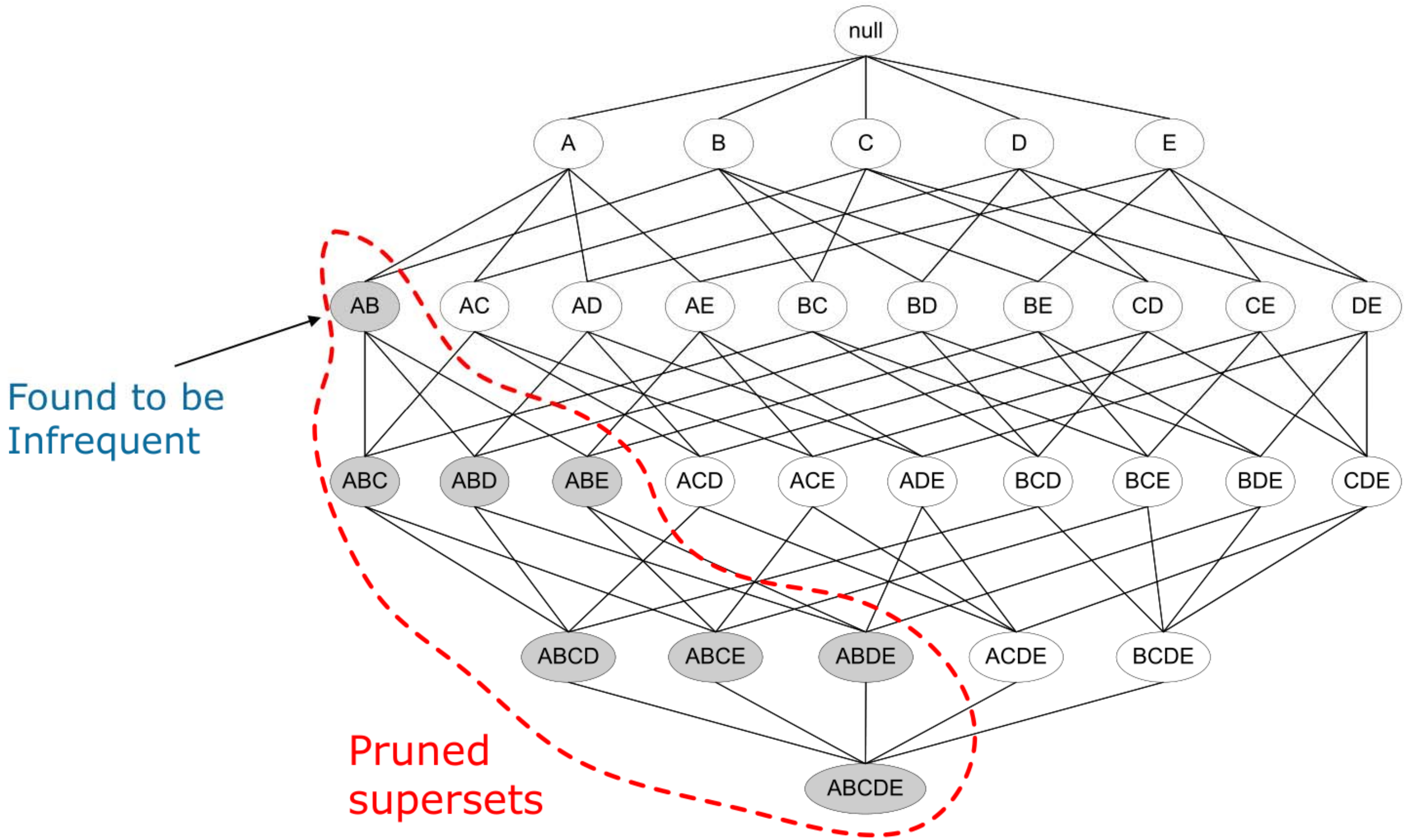
Reducing the Number of Candidates: Apriori Principle

- **Apriori principle**: If an itemset is frequent, then all of its subsets must also be frequent.

- Apriori principle holds due to the **anti-monotone** property of support measure: support of an itemset never exceeds the support of its subsets.

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Apriori Principle



Source: (1) Prof. Pier Luca Lanzi, Association Rule Basics, Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)
(2) Tan,Steinbach, Kumar Introduction to Data Mining

Illustrating Apriori Principle

Transaction ID (TID)	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Minimum Support = 4

1-itemsets

Item	Count
Bread	4
Peanuts	4
Milk	6
Fruit	6
Jam	5
Soda	6
Chips	4
Steak	1
Cheese	1
Yogurt	1

2-itemsets

Item	Count
Bread, Jam	4
Peanuts, Fruit	4
Milk, Fruit	5
Milk, Jam	4
Milk, Soda	5
Fruit, Soda	4
Jam, Soda	4
Soda, Chips	4

3-itemsets

Item	Count
Milk, Fruit, Soda	4

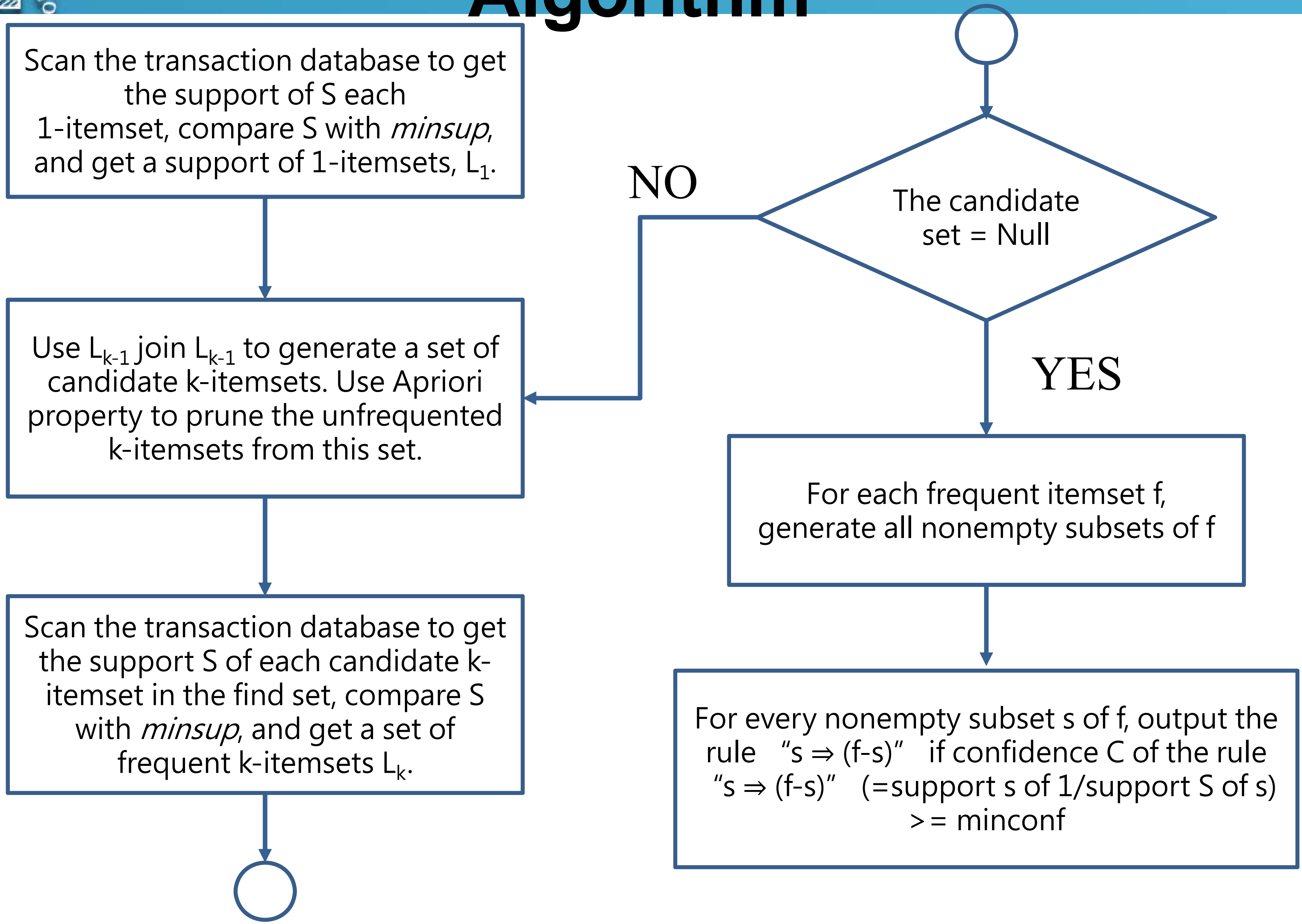
Definition of Apriori Algorithm

- The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

- Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as **candidate generation**), and groups of candidates are tested against the data.

- Apriori is designed to operate on database containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

Steps To Perform Apriori Algorithm



Apriori Algorithm

- Let k=1
- Generate frequent itemsets of length 1.
- Repeat until no new frequent itemsets are identified:
  - Generate length *(k+1)* candidate itemsets from length k frequent itemsets.
  - Prune candidate itemsets containing subsets of length k that are infrequent.
  - Count the support of each candidate by scanning the DB.
  - Eliminate candidates that are infrequent, leaving only those that are frequent.
- **Join Step**: C_k is generated by joining L_{k-1} with itself.
- **Prune Step**: any *(k-1)*-itemset that is not frequent cannot be a subset of a frequent k-itemset.

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

```
L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) do begin
    C_k+1 = candidates generated from L_k;
    for each transaction t in database do
        increment the count of all candidates in C_k+1
        that are contained in t
    L_k+1 = candidates in C_k+1 with min_support
    end
return ∪_k L_k;
```

Source: Prof. Pier Luca Lanzi, Association Rule Basics, Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

Example of Apriori Run

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

C_1 → 1st scan

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{D}	**1**
{E}	3

L_1 →

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{E}	3

C_2 →

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

→ 2nd scan

C_2

Itemset	Sup.
{A, B}	**1**
{A, C}	2
{A, E}	**1**
{B, C}	2
{B, E}	3
{C, E}	2

→

L_2

Itemset	Sup.
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

→

C_3

Itemset
{B, C, E}

→ 3rd scan

L_3

Itemset	Sup.
{B, C, E}	2

Step (2): Rule Generation

- Given a frequent itemset {L}, find all non-empty subsets $\{f\} \subset \{L\}$ such that the association rule $\{f\} \Rightarrow \{L - f\}$ satisfies the minimum confidence.

- Create the rule $\{f\} \Rightarrow \{L - f\}$.
  - If L={A,B,C,D} is a frequent itemset, candidate rules:
    {ABC} ⇒ {D}, {ABD} ⇒ {C}, {ACD} ⇒ {B}, {BCD} ⇒ {A},
    {A} ⇒ {BCD}, {B} ⇒ {ACD},{C} ⇒ {ABD}, {D} ⇒ {ABC},
    {AB} ⇒ {CD}, {AC} ⇒ {BD}, {AD} ⇒ {BC},
    {BC} ⇒ {AD}, {BD} ⇒ {AC}, {CD} ⇒ {AB}.

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $\{L\} \Rightarrow \{\varnothing\}$ and $\{\varnothing\} \Rightarrow \{L\}$).

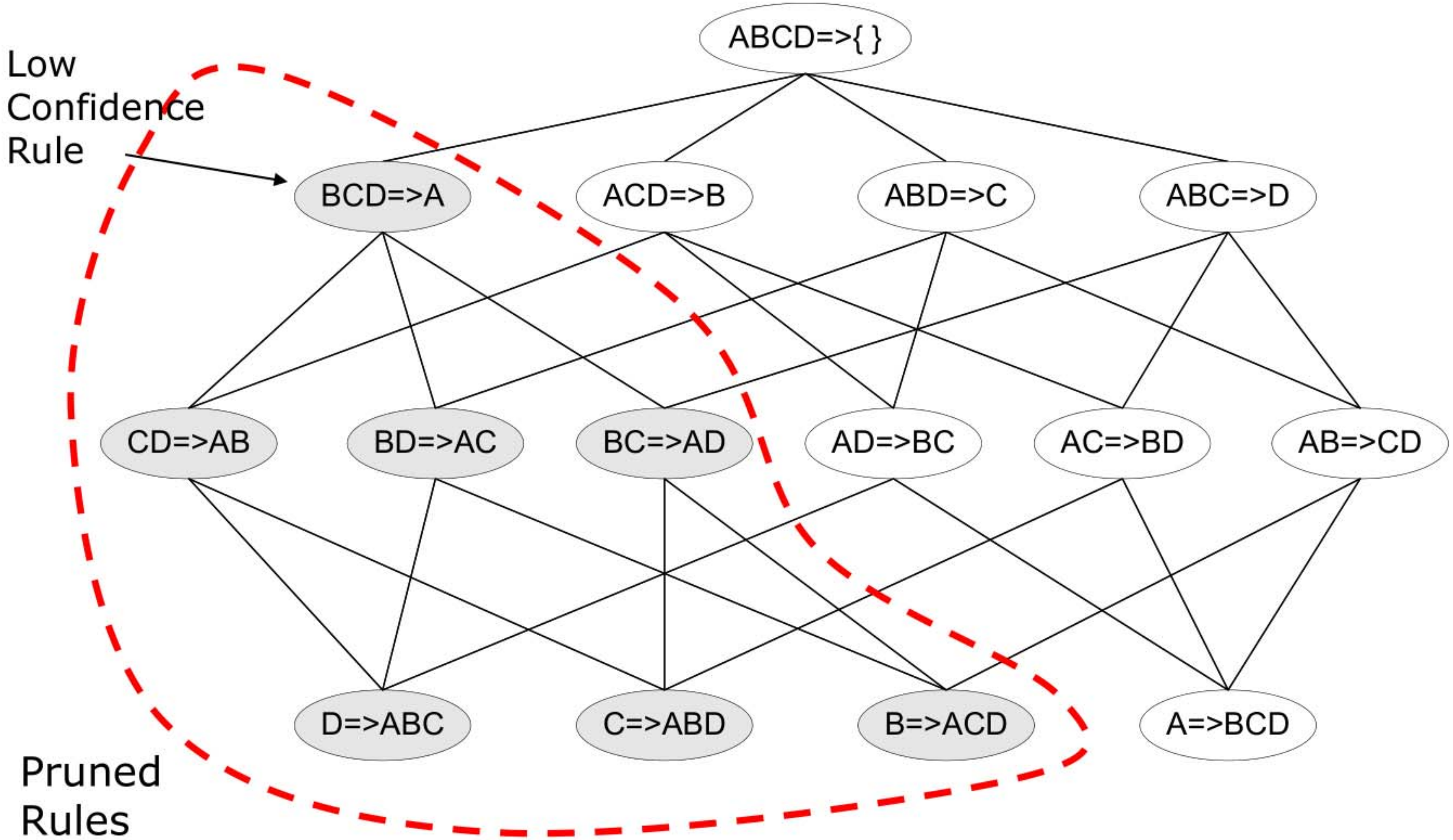
Generate Rules from Frequent Itemsets

- Confidence does not have an anti-monotone property
  - c({ABC} ⇒ {D}) can be larger or smaller than c({AB} ⇒ {D})

- But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:
    c({ABC} ⇒ {D}) >= c({AB} ⇒ {CD}) >= c({A} ⇒ {BCD})
  - Confidence is anti-monotone with respect to the number of items on the right hand side of the rule.
  - We can apply this property to prune the rule generation.

$$\text{confident}(A \Rightarrow B) = P(B \mid A) = P(A \cap B)/P(A)$$

Rule Generation for Apriori Algorithm



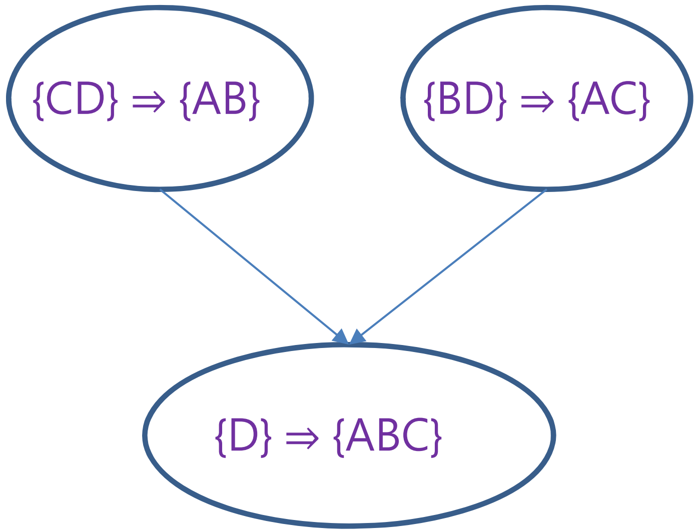
Source: (1) Prof. Pier Luca Lanzi, Association Rule Basics, Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)
(2) Tan,Steinbach, Kumar Introduction to Data Mining

Rule Generation for Apriori Algorithm

- Candidate rule is generated by **merging two rules** that share the same prefix in the rule consequent.

- Join($\{CD\} \Rightarrow \{AB\}$, $\{BD\} \Rightarrow \{AC\}$) would produce the candidate rule $\{D\} \Rightarrow \{ABC\}$

- Prune rule $\{D\} \Rightarrow \{ABC\}$ if its subset $\{AD\} \Rightarrow \{BC\}$ does not have high confidence.



Apriori Advantages/Disadvantages

- Advantages
  - Uses large itemset property.
  - Easily parallelized.
  - Easy to implement.
- Disadvantages
  - Assumes transaction database is memory resident.
  - Requires many database scans.
- Challenges in AR Mining
  - Apriori scans the data base multiple times.
  - Most often, there is a high number of candidates.
  - Support counting for candidates can be time expensive.
- Several methods try to improve this points by
  - Reduce the number of scans of the data base.
  - Shrink the number of candidates.
  - Counting the support of candidates more efficiently.

Choose an Appropriate *minsup* and Pattern Evaluation

- Choose an Appropriate *minsup*
  - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
  - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large
  - A single minimum support threshold may not be effective.

- Pattern Evaluation
  - Association rule algorithms tend to produce too many rules
    - many of them are uninteresting or redundant.

    (Redundant if $\{A,B,C\} \Rightarrow \{D\}$ and $\{A,B\} \Rightarrow \{D\}$ have same support & confidence.)
  - Interestingness measures can be used to prune/rank the derived patterns.
  - In the original formulation of association rules, support & confidence are the only measures used.

R Package: arules

- **arules**: Mining Association Rules and Frequent Itemsets
  - Provides the infrastructure for representing, manipulating and analyzing transaction data and patterns (frequent itemsets and association rules).
  - Also provides interfaces to C implementations of the association mining algorithms Apriori and Eclat by C. Borgelt.
- **apriori{arules}**:
  - The Apriori algorithm employs level-wise search for frequent itemsets.
  - The defaults: (1) supp=0.1, the minimum support of rules; (2) conf=0.8, the minimum confidence of rules; and (3) maxlen=10, which is the maximum length of rules.
- **eclat{arules}**:
  - The ECLAT algorithm finds frequent itemsets with equivalence classes, depth-first search and set intersection instead of counting.
- **interestMeasure{arules}**: more than twenty measures for selecting interesting association rules can be calculated.
- Other R packages:
  - **arulesViz**: A package for visualizing association rules based on package **arules**.
  - **arulesSequences**: provides functions for mining sequential patterns.
  - **arulesNBMiner**: implements an algorithm for mining negative binomial (NB) frequent itemsets and NB-precise rules.

http://lyle.smu.edu/IDA/arules/

https://cran.r-project.org/web/packages/arules/index.html

http://michael.hahsler.net/research/arules_RUG_2015/demo/

(**arules**: Association Rule Mining with R — A Tutorial, Michael Hahsler, Mon Sep 21 10:51:59 2015)

apriori{arules}: Mining Associations with Apriori

- Description
  - Mine frequent itemsets, association rules or association hyperedges using the Apriori algorithm. The Apriori algorithm employs level-wise search for frequent itemsets. The implementation of Apriori used includes some improvements (e.g., a prefix tree and item sorting).
- Usage

```
apriori(data, parameter = NULL, appearance = NULL, control = NULL)
```

- Arguments
  - `data`: object of class transactions or any data structure which can be coerced into transactions (e.g., a binary matrix or data.frame).
  - `parameter`: object of class APparameter or named list. The default behavior is to mine rules with support 0.1, confidence 0.8, and maxlen 10.
  - `appearance`: object of class APappearance or named list. With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.
  - `control`: object of class APcontrol or named list. Controls the algorithmic performance of the mining algorithm (item sorting, etc.)
- **Note**: Apriori only creates rules with one item in the RHS!

eclat{arules}: Mining Associations with Eclat

- Description
  - Mine frequent itemsets with the Eclat algorithm. This algorithm uses simple intersection operations for equivalence class clustering along with bottom-up lattice traversal.
- Usage

```
eclat(data, parameter = NULL, control = NULL)
```

- Arguments
  - `data`: object of class transactions or any data structure which can be coerced into transactions (e.g., binary matrix, data.frame).
  - `parameter`: object of class ECparameter or named list (default values are: support 0.1 and maxlen 5)
  - `control`: object of class ECcontrol or named list for algorithmic controls.

Case Study 0: Adult Data Set

```
> # The AdultUCI datset contains the questionnaire data of the "Adult" database (originally
called the "Census Income" Database) with 48842 observations on the 15 variables.
> library(arules)
> data(AdultUCI)
> head(AdultUCI)
  age        workclass fnlwgt education education-num     marital-status        occupation
1  39        State-gov  77516 Bachelors            13      Never-married      Adm-clerical
2  50 Self-emp-not-inc  83311 Bachelors            13 Married-civ-spouse   Exec-managerial
3  38          Private 215646   HS-grad             9           Divorced Handlers-cleaners
4  53          Private 234721      11th             7 Married-civ-spouse Handlers-cleaners
5  28          Private 338409 Bachelors            13 Married-civ-spouse    Prof-specialty
6  37          Private 284582   Masters            14 Married-civ-spouse   Exec-managerial
   relationship  race    sex capital-gain capital-loss hours-per-week native-country income
1 Not-in-family White   Male         2174            0             40  United-States  small
2       Husband White   Male            0            0             13  United-States  small
3 Not-in-family White   Male            0            0             40  United-States  small
4       Husband Black   Male            0            0             40  United-States  small
5          Wife Black Female            0            0             40           Cuba  small
6          Wife White Female            0            0             40  United-States  small
> data(Adult)
> ?Adult #see how to create transactions from AdultUCI
> Adult
transactions in sparse format with
 48842 transactions (rows) and
 115 items (columns)
> class(Adult)
[1] "transactions"
attr(,"package")
[1] "arules"
> ?transactions
```

See Also

APparameter-class, APcontrol-class, APappearance-class, transactions-cl

Examples

```
data("Adult")
## Mine association rules.
rules <- apriori(Adult,
        parameter = list(supp = 0.5, conf = 0.9, target = "rules"))
summary(rules)
```

Adult Data Set (transactions form)


```
> str(Adult)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:612200] 1 10 25 32 35 50 59 61 63 65 ...
  .. .. ..@ p       : int [1:48843] 0 13 26 39 52 65 78 91 104 117 ...
  .. .. ..@ Dim     : int [1:2] 115 48842
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 115 obs. of  3 variables:
  .. ..$ labels   : chr [1:115] "age=Young" "age=Middle-aged" "age=Senior" "age=Old" ...
  .. ..$ variables: Factor w/ 13 levels "age","capital-gain",..: 1 1 1 1 13 13 13 13 13 13 ...
  .. ..$ levels   : Factor w/ 112 levels "10th","11th",..: 111 63 92 69 30 54 65 82 90 91 ...
  ..@ itemsetInfo:'data.frame': 48842 obs. of  1 variable:
  .. ..$ transactionID: chr [1:48842] "1" "2" "3" "4" ...
```

```
> inspect(Adult[1:2])
  items                              transactionID
1 {age=Middle-aged,
   workclass=State-gov,
   education=Bachelors,
   marital-status=Never-married,
   occupation=Adm-clerical,
   relationship=Not-in-family,
   race=White,
   sex=Male,
   capital-gain=Low,
   capital-loss=None,
   hours-per-week=Full-time,
   native-country=United-States,
   income=small}                                 1
```

```
2 {age=Senior,
   workclass=Self-emp-not-inc,
   education=Bachelors,
   marital-status=Married-civ-spouse,
   occupation=Exec-managerial,
   relationship=Husband,
   race=White,
   sex=Male,
   capital-gain=None,
   capital-loss=None,
   hours-per-week=Part-time,
   native-country=United-States,
   income=small}
```

Adult Data Set

```
> summary(Adult)
transactions as itemMatrix in sparse format with
 48842 rows (elements/itemsets/transactions) and
 115 columns (items) and a density of 0.1089939

most frequent items:
     capital-loss=None    capital-gain=None    native-country=United-States     race=White
                 46560                44807                           43832          41762
     workclass=Private              (Other)
                 33906               401333

element (itemset/transaction) length distribution:
sizes
    9    10    11    12    13
   19   971  2067 15623 30162

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   12.00   13.00   12.53   13.00   13.00

includes extended item information - examples:
           labels variables      levels
1       age=Young       age       Young
2 age=Middle-aged       age Middle-aged
3      age=Senior       age      Senior

includes extended transaction information - examples:
  transactionID
1             1
2             2
3             3
```

How to Create Transactions Data

- Transactions can be created by coercion from lists containing transactions, but also from matrix and data.frames. However, you will need to prepare your data first. Association rule mining can only use items and does not work with continuous variables.

```
> # creating transactions form a list
> a.list <- list(
+       c("a","b","c"),
+       c("a","b"),
+       c("a","b","d"),
+       c("c","e"),
+       c("a","b","d","e")
+       )
> names(a.list) <- paste0("Customer", c(1:5))
```

```
> a.list
$Customer1
[1] "a" "b" "c"

$Customer2
[1] "a" "b"

$Customer3
[1] "a" "b" "d"

$Customer4
[1] "c" "e"

$Customer5
[1] "a" "b" "d" "e"
```

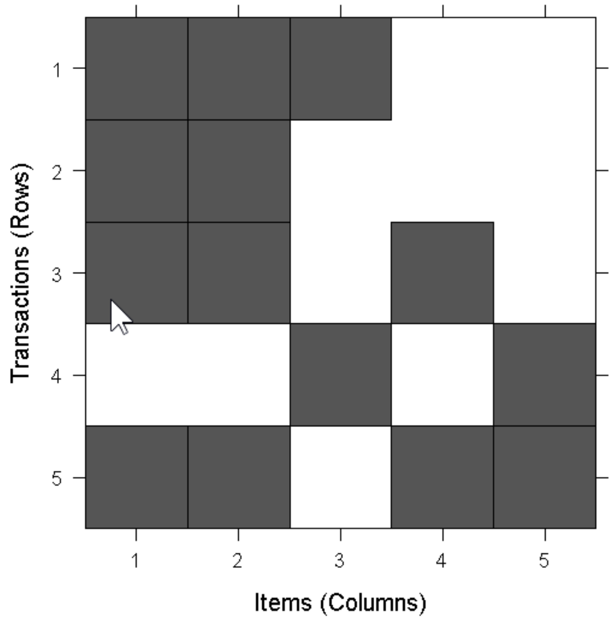
```
# avoid "no method or default for coercing"
library(Matrix)
productTD <- as(product_by_user$Product, "transactions")
inspect(productTD[1:5])
```

http://127.0.0.1:18470/library/arules/html/transactions-class.html

Coerce a List Into Transactions

```
> alist.trans <- as(a.list, "transactions")
> summary(alist.trans) # analyze transactions
transactions as itemMatrix in sparse format with
 5 rows (elements/itemsets/transactions) and
 5 columns (items) and a density of 0.56

most frequent items:
      a       b       c       d       e (Other)
      4       4       2       2       2       0

element (itemset/transaction) length distribution:
sizes
2 3 4
2 2 1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    2.0     2.0     3.0     2.8     3.0     4.0

includes extended item information - examples:
  labels
1      a
2      b
3      c

includes extended transaction information - examples:
  transactionID
1     Customer1
2     Customer2
3     Customer3
```

```
> image(alist.trans)
```



Creating Transactions from a Matrix


```
> a.matrix <- matrix(c(
+         1,1,1,0,0,
+         1,1,0,0,0,
+         1,1,0,1,0,
+         0,0,1,0,1), ncol = 5)
> dimnames(a.matrix) <- list(
    paste("Customer", letters[1:4]),
    paste0("Item", c(1:5)))
> a.matrix
           Item1 Item2 Item3 Item4 Item5
Customer a     1     0     0     0     0
Customer b     1     1     0     1     1
Customer c     1     1     1     0     0
Customer d     0     0     1     0     1
>
> amatirx.trans <- as(a.matrix, "transactions")
> amatirx.trans
transactions in sparse format with
 4 transactions (rows) and
 5 items (columns)
> inspect(amatirx.trans)
    items                     transactionID
[1] {Item1}                   Customer a
[2] {Item1,Item2,Item4,Item5} Customer b
[3] {Item1,Item2,Item3}       Customer c
[4] {Item3,Item5}             Customer d
```

```
> summary(amatirx.trans)
transactions as itemMatrix in sparse format with
 4 rows (elements/itemsets/transactions) and
 5 columns (items) and a density of 0.5

most frequent items:
  Item1   Item2   Item3   Item5   Item4 (Other)
      3       2       2       2       1       0

element (itemset/transaction) length distribution:
sizes
1 2 3 4
1 1 1 1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.75    2.50    2.50    3.25    4.00

includes extended item information - examples:
  labels
1  Item1
2  Item2
3  Item3

includes extended transaction information -
examples:
  transactionID
1    Customer a
2    Customer b
3    Customer c
```

More Examples

```
> # creating transactions from data.frame
> a.df <- data.frame(
+   age   = as.factor(c(6, 8, NA, 9, 16)),
+   grade = as.factor(c("A", "C", "F", NA, "C")),
+   pass  = c(TRUE, TRUE, FALSE, TRUE, TRUE))
> # note: factors are translated to
> # logicals and NAs are ignored
> a.df
   age grade  pass
1    6     A  TRUE
2    8     C  TRUE
3 <NA>     F FALSE
4    9  <NA>  TRUE
5   16     C  TRUE
> adf.trans <- as(a.df, "transactions")
> inspect(adf.trans)
    items                      transactionID
[1] {age=6,grade=A,pass}  1
[2] {age=8,grade=C,pass}  2
[3] {grade=F}             3
[4] {age=9,pass}          4
[5] {age=16,grade=C,pass} 5
> as(adf.trans, "data.frame")
                  items transactionID
1  {age=6,grade=A,pass}             1
2  {age=8,grade=C,pass}             2
3             {grade=F}             3
4          {age=9,pass}             4
5 {age=16,grade=C,pass}             5
```

```
> # creating transactions from (IDs, items)
> a.df2 <- data.frame(
+   TID = c(1, 1, 2, 2, 2, 3),
+   item = c("a", "b", "a", "b", "c", "b"))
> a.df2
  TID item
1   1    a
2   1    b
3   2    a
4   2    b
5   2    c
6   3    b
> a.df2.s <- split(a.df2[, "item"],
a.df2[,"TID"])
> a.df2.s
$`1`
[1] a b
Levels: a b c
$`2`
[1] a b c
Levels: a b c
$`3`
[1] b
Levels: a b c
> adf2.trans <- as(a.df2.s, "transactions")
> inspect(adf2.trans)
    items   transactionID
[1] {a,b}   1
[2] {a,b,c} 2
[3] {b}     3
```

Example: Create Transactions Data

```
> data(AdultUCI)
> summary(AdultUCI)
> # remove attributes
> AdultUCI[["fnlwgt"]] <- NULL
> AdultUCI[["education-num"]] <- NULL
```

```
> summary(AdultUCI)
      age                   workclass          fnlwgt               education       education-num
 Min.   :17.00   Private         :33906   Min.   :  12285   HS-grad     :15784   Min.   : 1.00
 1st Qu.:28.00   Self-emp-not-inc: 3862   1st Qu.: 117551   Some-college:10878   1st Qu.: 9.00
 Median :37.00   Local-gov       : 3136   Median : 178145   Bachelors   : 8025   Median :10.00
 Mean   :38.64   State-gov       : 1981   Mean   : 189664   Masters     : 2657   Mean   :10.08
 3rd Qu.:48.00   Self-emp-inc    : 1695   3rd Qu.: 237642   Assoc-voc   : 2061   3rd Qu.:12.00
 Max.   :90.00   (Other)         : 1463   Max.   :1490400   11th        : 1812   Max.   :16.00
                 NA's            : 2799                     (Other)     : 7625
               marital-status              occupation            relationship                     race
 Divorced             : 6633   Prof-specialty : 6172   Husband       :19716   Amer-Indian-Eskimo:  470
 Married-AF-spouse    :   37   Craft-repair   : 6112   Not-in-family :12583   Asian-Pac-Islander: 1519
 Married-civ-spouse   :22379   Exec-managerial: 6086   Other-relative: 1506   Black             : 4685
 Married-spouse-absent:  628   Adm-clerical   : 5611   Own-child     : 7581   Other             :  406
 Never-married        :16117   Sales          : 5504   Unmarried     : 5125   White             :41762
 Separated            : 1530   (Other)        :16548   Wife          : 2331
 Widowed              : 1518   NA's           : 2809
     sex          capital-gain     capital-loss     hours-per-week         native-country      income
 Female:16192   Min.   :    0   Min.   :   0.0   Min.   : 1.00   United-States:43832   small:24720
 Male  :32650   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00   Mexico       :  951   large: 7841
                Median :    0   Median :   0.0   Median :40.00   Philippines  :  295   NA's :16281
                Mean   : 1079   Mean   :  87.5   Mean   :40.42   Germany      :  206
                3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00   Puerto-Rico  :  184
                Max.   :99999   Max.   :4356.0   Max.   :99.00   (Other)      : 2517
                                                                 NA's         :  857
```

http://127.0.0.1:18470/library/arules/html/Adult.html

Example: Create Transactions Data


```
> # map metric attributes
> AdultUCI[["age"]] <- ordered(cut(AdultUCI[[ "age"]], c(15, 25, 45, 65, 100)),
+   labels = c("Young", "Middle-aged", "Senior", "Old"))
> AdultUCI[["hours-per-week"]] <- ordered(cut(AdultUCI[["hours-per-week"]],
+   c(0, 25, 40, 60, 168)),
+   labels = c("Part-time", "Full-time", "Over-time", "Workaholic"))
> AdultUCI[["capital-gain"]] <- ordered(cut(AdultUCI[["capital-gain"]],
+   c(-Inf, 0, median(AdultUCI[["capital-gain"]][AdultUCI[["capital-gain"]] > 0]), Inf)),
+   labels = c("None", "Low", "High"))
> AdultUCI[["capital-loss"]] <- ordered(cut(AdultUCI[["capital-loss"]],
+   c(-Inf, 0, median(AdultUCI[["capital-loss"]][AdultUCI[["capital-loss"]] > 0]), Inf)),
+   labels = c("None", "Low", "High"))
>
> summary(AdultUCI[c("age", "hours-per-week", "capital-gain", "capital-loss")])
          age              hours-per-week  capital-gain capital-loss
 Young      : 9627   Part-time : 5913   None:44807   None:46560
 Middle-aged:24671   Full-time :28577   Low : 2345   Low : 1166
 Senior     :12741   Over-time :12676   High: 1690   High: 1116
 Old        : 1803   Workaholic: 1676
>
> # create transactions
> MyAdult <- as(AdultUCI, "transactions")
> MyAdult
transactions in sparse format with
 48842 transactions (rows) and
 115 items (columns)
```

Example: Create Transactions Data

```
> summary(MyAdult)
transactions as itemMatrix in sparse format with
 48842 rows (elements/itemsets/transactions) and
 115 columns (items) and a density of 0.1089939

most frequent items:
           capital-loss=None            capital-gain=None native-country=United-States
                       46560                        44807                        43832
                  race=White            workclass=Private                      (Other)
                       41762                        33906                       401333

element (itemset/transaction) length distribution:
sizes
    9    10    11    12    13
   19   971  2067 15623 30162

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   12.00   13.00   12.53   13.00   13.00

includes extended item information - examples:
           labels variables      levels
1       age=Young       age       Young
2 age=Middle-aged       age Middle-aged
3      age=Senior       age      Senior

includes extended transaction information - examples
  transactionID
1             1
2             2
3             3
```

```
> inspect(MyAdult[1:2])
    items                       transactionID
[1] {age=Middle-aged,
     workclass=State-gov,
     education=Bachelors,
     marital-status=Never-married,
     occupation=Adm-clerical,
     relationship=Not-in-family,
     race=White,
     sex=Male,
     capital-gain=Low,
     capital-loss=None,
     hours-per-week=Full-time,
     native-country=United-States,
     income=small}                          1
```

Case Study 1: Groceries Data Set

- **Description**: The Groceries data set 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.
- **Format**: Object of class transactions.

```
> library(arules)
> data(Groceries)
> ?Groceries
> str(Groceries)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:43367] 13 60 69 78 14 29 98 24 15 29 ...
  .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
  .. .. ..@ Dim     : int [1:2] 169 9835
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 169 obs. of  3 variables:
  .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42 42 41
...
  .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  ..@ itemsetInfo:'data.frame': 0 obs. of  0 variables
```

```
Groceries@itemInfo
```

summary, inspect

```
> 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
            2513             1903             1809             1715             1372
         (Other)
           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
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14
  19   20   21   22   23   24   26   27   28   29   32
  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(Groceries[1:4])
  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}
```

Apply apriori

```
> rule0 <- apriori(Groceries)
Apriori

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

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].
```

The default behavior is to mine rules with
minimum support of 0.1,
minimum confidence of 0.8,
maximum of 10 items (maxlen), and
a maximal time for subset checking of 5 seconds (maxtime).

Apply apriori With Different Arguments

```
> rule1 <- apriori(Groceries, parameter=list(support=0.005, confidence=0.64))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
       0.64    0.1    1 none FALSE            TRUE   0.005      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: 49

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [120 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [4 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(rule1)
  lhs                                         rhs          support     confidence lift
1 {butter,whipped/sour cream}              => {whole milk} 0.006710727 0.6600000  2.583008
2 {pip fruit,whipped/sour cream}           => {whole milk} 0.005998983 0.6483516  2.537421
3 {pip fruit,root vegetables,other vegetables} => {whole milk} 0.005490595 0.6750000  2.641713
4 {tropical fruit,root vegetables,yogurt}      => {whole milk} 0.005693950 0.7000000  2.739554
```

Class 'rules'

```
> str(rule1)
Formal class 'rules' [package "arules"] with 4 slots
  ..@ lhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
  .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. .. .. ..@ i        : int [1:10] 25 30 15 30 15 19 22 14 19 29
  .. .. .. .. ..@ p        : int [1:5] 0 2 4 7 10
  .. .. .. .. ..@ Dim      : int [1:2] 169 4
  .. .. .. .. ..@ Dimnames:List of 2
  .. .. .. .. .. ..$ : NULL
  .. .. .. .. .. ..$ : NULL
  .. .. .. .. ..@ factors : list()
  .. .. ..@ itemInfo   :'data.frame':   169 obs. of  3 variables:
  .. .. .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. .. .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42 42 41 ...
  .. .. .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  .. .. ..@ itemsetInfo:'data.frame':   0 obs. of  0 variables
  ..@ rhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
  .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. .. .. ..@ i        : int [1:4] 24 24 24 24
  .. .. .. .. ..@ p        : int [1:5] 0 1 2 3 4
  .. .. .. .. ..@ Dim      : int [1:2] 169 4
  .. .. .. .. ..@ Dimnames:List of 2
  .. .. .. .. .. ..$ : NULL
  .. .. .. .. .. ..$ : NULL
  .. .. .. .. ..@ factors : list()
  .. .. ..@ itemInfo   :'data.frame':   169 obs. of  3 variables:
  .. .. .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. .. .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42 42 41 ...
  .. .. .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  .. .. ..@ itemsetInfo:'data.frame':   0 obs. of  0 variables
  ..@ quality:'data.frame':     4 obs. of  3 variables:
  .. ..$ support   : num [1:4] 0.00671 0.006 0.00549 0.00569
  .. ..$ confidence: num [1:4] 0.66 0.648 0.675 0.7
  .. ..$ lift      : num [1:4] 2.58 2.54 2.64 2.74
  ..@ info   :List of 4
  .. ..$ data         : symbol Groceries
  .. ..$ ntransactions: int 9835
  .. ..$ support      : num 0.005
  .. ..$ confidence   : num 0.64
```

```
> rule1@quality
      support confidence     lift
1 0.006710727  0.6600000 2.583008
2 0.005998983  0.6483516 2.537421
3 0.005490595  0.6750000 2.641713
4 0.005693950  0.7000000 2.739554
```

Select Top AR by Support

```
> rule2 <- apriori(Groceries, parameter=list(support=0.001, confidence=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE   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

set item appearances ...[0 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 3 4 5 6 done [0.01s].
writing ... [5668 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
> rule2.sorted_sup <- sort(rule2, by="support")
> inspect(rule2.sorted_sup[1:5])
     lhs                                      rhs          support    confidence lift
1472 {other vegetables,yogurt}             => {whole milk} 0.02226741 0.5128806  2.007235
1467 {tropical fruit,yogurt}               => {whole milk} 0.01514997 0.5173611  2.024770
1449 {other vegetables,whipped/sour cream} => {whole milk} 0.01464159 0.5070423  1.984385
1469 {root vegetables,yogurt}              => {whole milk} 0.01453991 0.5629921  2.203354
1454 {pip fruit,other vegetables}          => {whole milk} 0.01352313 0.5175097  2.025351
```

Select a Subset of Rules

```
> # Select a subset of rules using partial matching on the items
> # in the right-hand-side and a quality measure
> rule2.sub <- subset(rule2, subset = rhs %pin% "whole milk" & lift > 1.3)
> rule2.sub
set of 2679 rules
>
> # Display the top 3 support rules
> inspect(head(rule2.sub, n = 3, by = "support"))
     lhs                                       rhs          support    confidence lift
1472 {other vegetables,yogurt}              => {whole milk} 0.02226741 0.5128806  2.007235
1467 {tropical fruit,yogurt}                => {whole milk} 0.01514997 0.5173611  2.024770
1449 {other vegetables,whipped/sour cream} => {whole milk} 0.01464159 0.5070423  1.984385
>
> # Display the first 3 rules
> inspect(rule2.sub[1:3])
  lhs                rhs          support     confidence lift
1 {honey}         => {whole milk} 0.001118454 0.7333333  2.870009
3 {cocoa drinks}  => {whole milk} 0.001321810 0.5909091  2.312611
4 {pudding powder} => {whole milk} 0.001321810 0.5652174  2.212062
>
> # Get labels for the first 3 rules
> labels(rule2.sub[1:3])
[1] "{honey} => {whole milk}"          "{cocoa drinks} => {whole milk}"
[3] "{pudding powder} => {whole milk}"
> labels(rule2.sub[1:3], itemSep = " + ", setStart = "", setEnd="",  ruleSep = " ---> ")
[1] "honey ---> whole milk"          "cocoa drinks ---> whole milk"
[3] "pudding powder ---> whole milk"
```

Select Top AR by Confidence, Lift

```
> rule2.sorted_con <- sort(rule2, by="confidence")
> inspect(rule2.sorted_con[1:5])
     lhs                                          rhs          support     confidence lift
113  {rice,sugar}                              => {whole milk} 0.001220132 1          3.913649
258  {canned fish,hygiene articles}            => {whole milk} 0.001118454 1          3.913649
1487 {root vegetables,butter,rice}             => {whole milk} 0.001016777 1          3.913649
1646 {root vegetables,whipped/sour cream,flour} => {whole milk} 0.001728521 1          3.913649
1670 {butter,soft cheese,domestic eggs}        => {whole milk} 0.001016777 1          3.913649
>
> rule2.sorted_lift <- sort(rule2, by="lift")
> inspect(rule2.sorted_lift[1:5])
    lhs                                     rhs               support     confidence lift
53  {Instant food products,soda}         => {hamburger meat} 0.001220132 0.6315789  18.99565
37  {soda,popcorn}                       => {salty snack}    0.001220132 0.6315789  16.69779
444 {flour,baking powder}                => {sugar}          0.001016777 0.5555556  16.40807
327 {ham,processed cheese}               => {white bread}    0.001931876 0.6333333  15.04549
55  {whole milk,Instant food products}   => {hamburger meat} 0.001525165 0.5000000  15.03823
```

```
sort(x, decreasing = TRUE, na.last = NA,  by = "support", order = FALSE, ...)
```

S4 method for signature 'associations'

```
head(x, n = 6L, by = NULL, decreasing = TRUE, ...)
```

S4 method for signature 'associations'

```
tail(x, n = 6L, by = NULL, decreasing = TRUE, ...)
```

Select Top Frequent Itemsets

```
> rule.freq_item <- apriori(Groceries, parameter=list(support=0.001, target="frequent
itemsets"), control=list(sort=-1))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen            target   ext
         NA    0.1    1 none FALSE            TRUE   0.001      1     10 frequent itemsets FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE   -1    TRUE

Absolute minimum support count: 9

...
checking subsets of size 1 2 3 4 5 6 done [0.02s].
writing ... [13492 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rule.freq_item
set of 13492 itemsets
> inspect(rule.freq_item[1:5])
  items                support
1 {whole milk}       0.2555160
2 {other vegetables} 0.1934926
3 {rolls/buns}       0.1839349
4 {soda}             0.1743772
5 {yogurt}           0.1395018
```

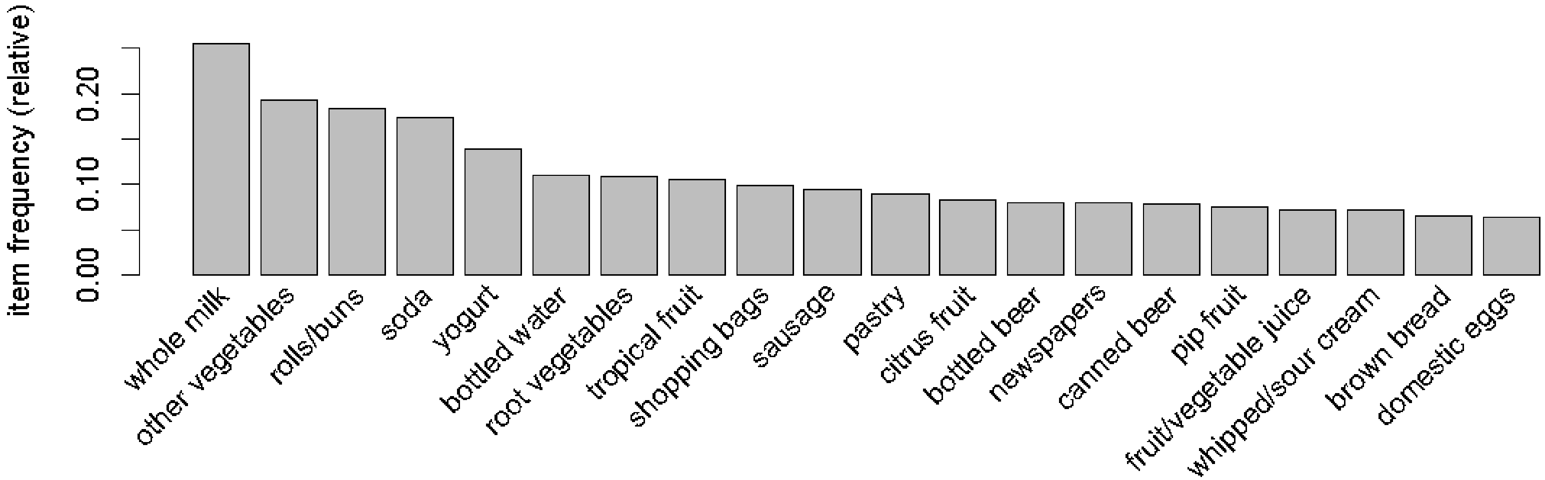
Frequent k-itemsets

```
> rule.fi_eclat <- eclat(Groceries, parameter=list(minlen=1, maxlen=3, support=0.001,
target="frequent itemsets"), control=list(sort=-1))
Eclat
...
> rule.fi_eclat
set of 9969 itemsets
> inspect(rule.fi_eclat[1:5])
  items                        support
1 {whole milk,honey}           0.001118454
2 {whole milk,cocoa drinks}    0.001321810
3 {whole milk,pudding powder}  0.001321810
4 {tidbits,rolls/buns}         0.001220132
5 {tidbits,soda}               0.001016777

> rule.fi_eclat <- eclat(Groceries, parameter=list(minlen=3, maxlen=5, support=0.001,
target="frequent itemsets"), control=list(sort=-1))
Eclat
...
> rule.fi_eclat
set of 10344 itemsets
> inspect(rule.fi_eclat[1:5])
  items                                          support
1 {liver loaf,whole milk,yogurt}                 0.001016777
2 {tropical fruit,other vegetables,curd cheese}  0.001016777
3 {whole milk,curd cheese,rolls/buns}            0.001016777
4 {other vegetables,whole milk,curd cheese}      0.001220132
5 {other vegetables,whole milk,cleaner}          0.001016777
```

itemFrequencyPlot{arules}: Creating a Item Frequencies/Support Bar Plot

```
itemFrequencyPlot(x, type = c("relative", "absolute"),
    weighted = FALSE, support = NULL, topN = NULL,
    population = NULL, popCol = "black", popLwd = 1,
    lift = FALSE, horiz = FALSE,
    names = TRUE, cex.names =  graphics::par("cex.axis"),
    xlab = NULL, ylab = NULL, mai = NULL, ...)
```

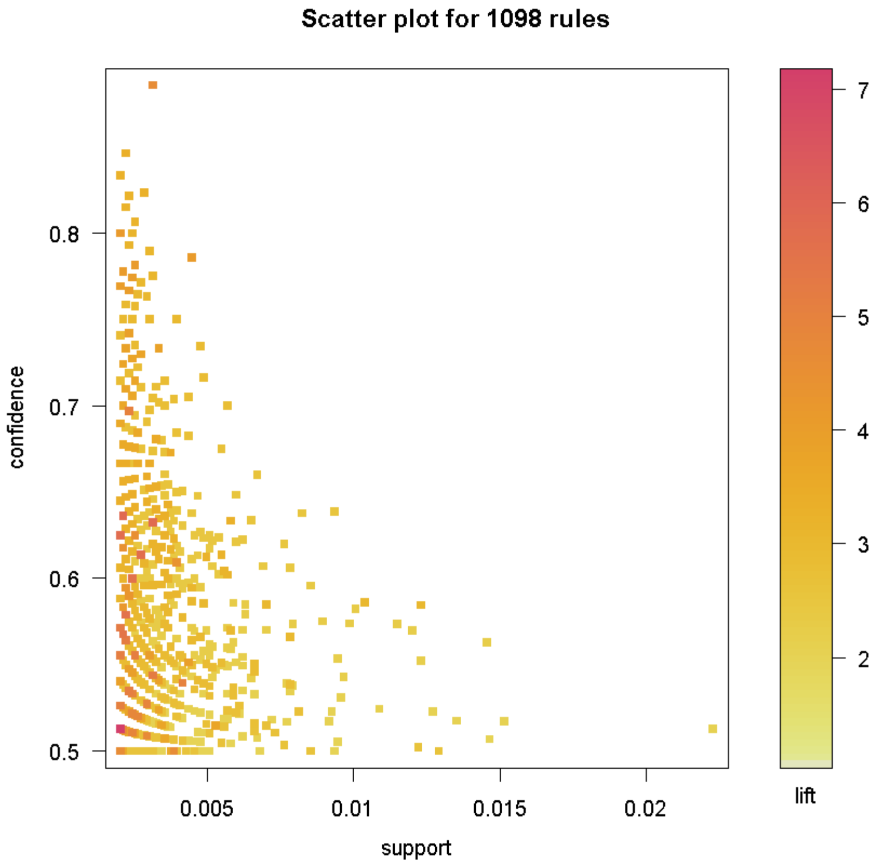
`itemFrequencyPlot(Groceries, topN=20)`

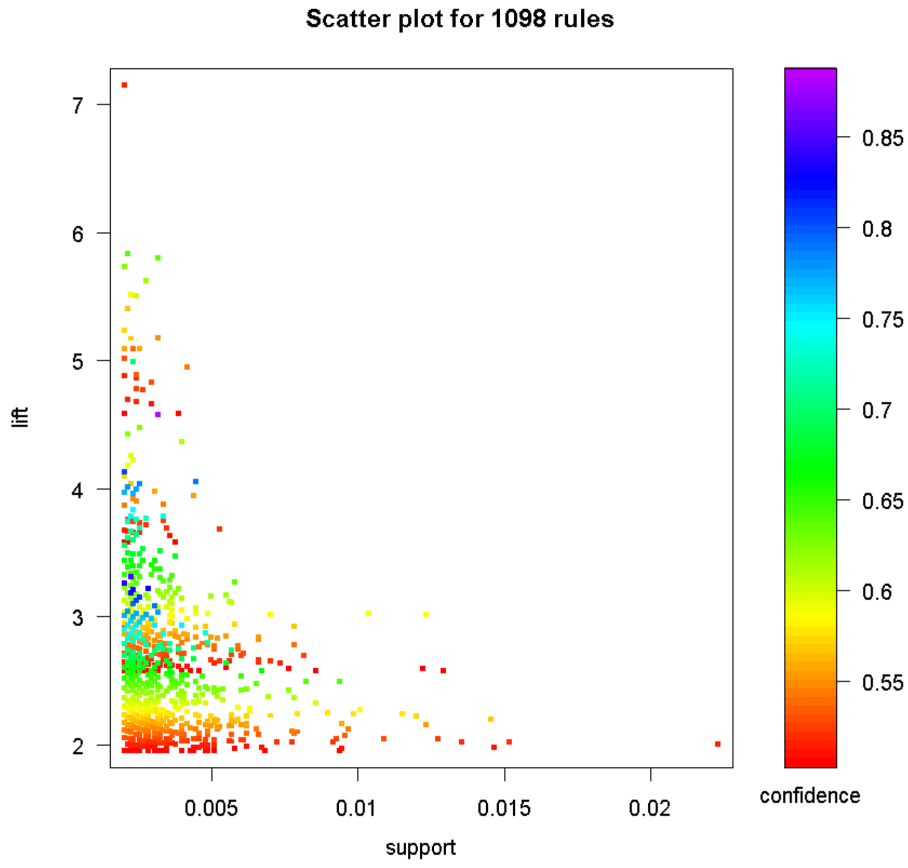


arulesViz

Visualizing Association Rules and Frequent Itemsets

```
> library(arulesViz)
> rule.a <- apriori(Groceries, parameter=list(support=0.002, confidence=0.5))
> plot(rule.a)
```

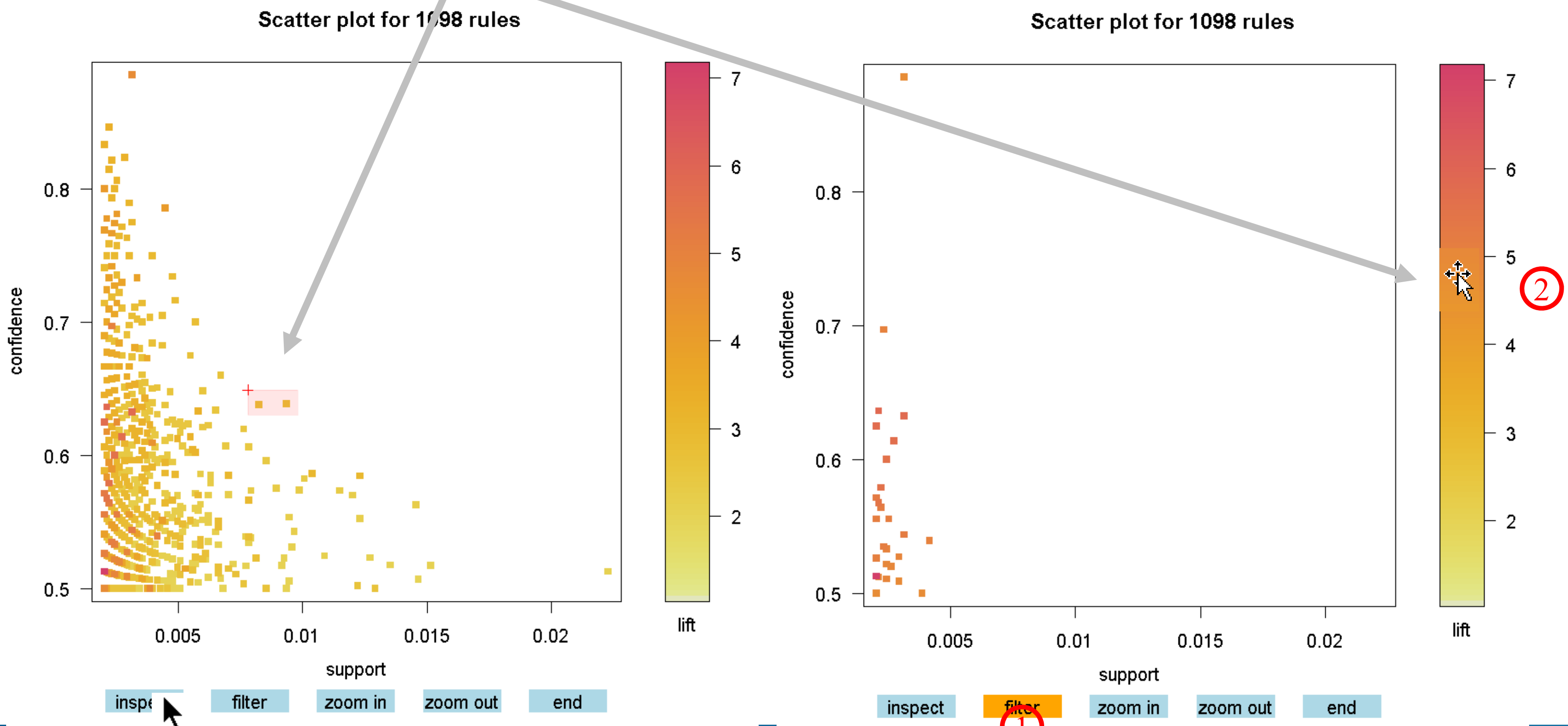





```
> plot(rule.a, measure=c("support", "lift"), shading="confidence",
       col=rainbow(100)[80:1], cex=0.3)
```

Interactive Plot

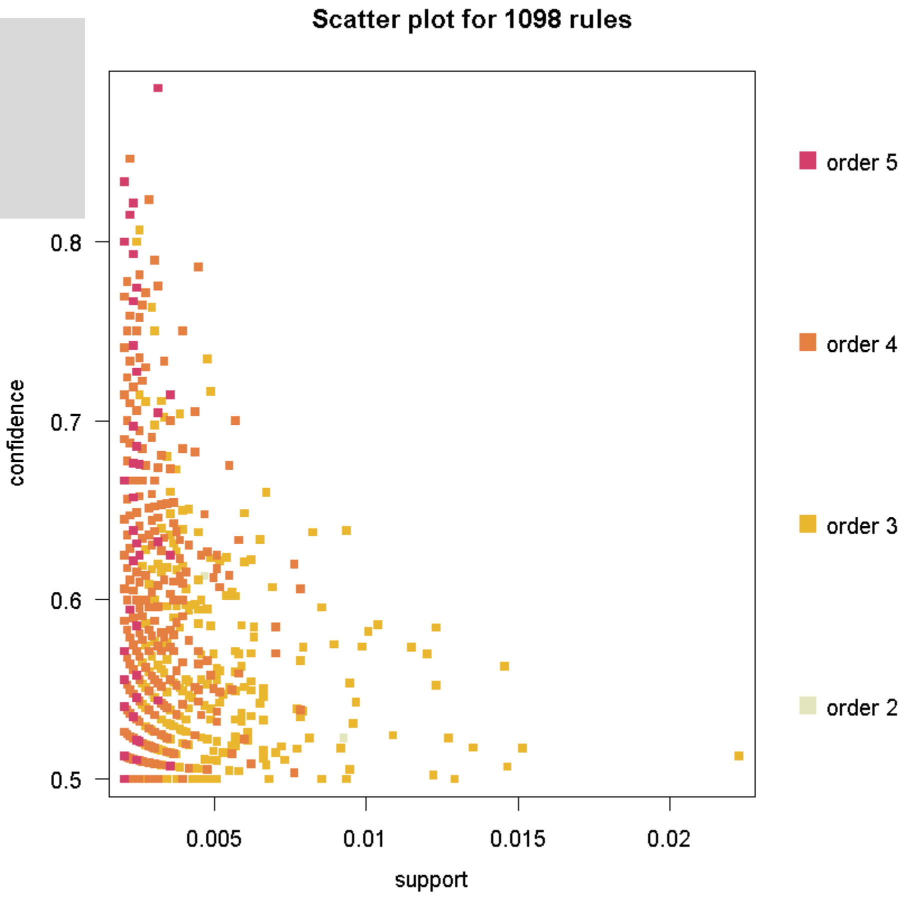
```
> plot(rule.a, interactive=TRUE)
Interactive mode.
Select a region with two clicks!
Number of rules selected: 2
    lhs                          rhs            support     confidence lift     order
524 {butter,yogurt}           => {whole milk} 0.009354347 0.6388889  2.500387 3
523 {root vegetables,butter} => {whole milk} 0.008235892 0.6377953  2.496107 3

Select minimum lift in colorkey.
```



Plot Methods

```
plot(rule.a, method="two-key plot")
```

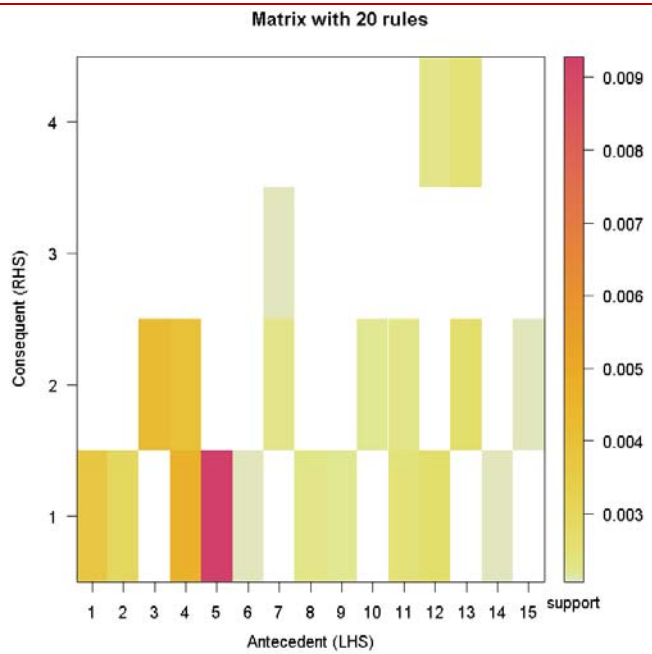
```
plot(x, method = NULL, measure =
"support", shading = "lift",
    interactive = FALSE, data = NULL,
control = NULL, ...)
```

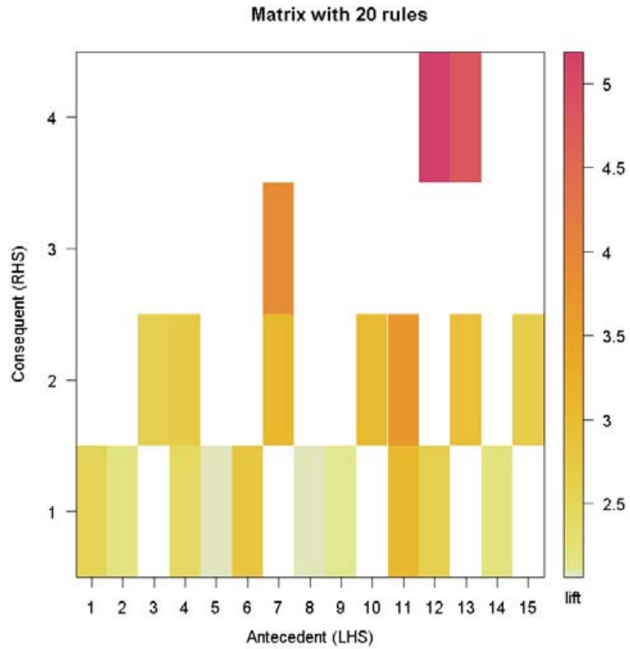


`method`: a string with value "scatterplot", "two-key plot", "matrix", "matrix3D", "mosaic", "doubledecker", "graph", "paracoord" or "grouped", "iplots" selecting the visualization method (see Details).

Plot Methods: matrix

```
> plot(rule.a[1:20], method="matrix")
> plot(rule.a[1:20], method="matrix", measure="lift")
Itemsets in Antecedent (LHS)
 [1] "{cereals}"                          "{jam}"
 [3] "{specialty cheese}"                 "{rice}"
 [5] "{baking powder}"                    "{yogurt,specialty cheese}"
 [7] "{whole milk,specialty cheese}"      "{other vegetables,specialty cheese}"
 [9] "{turkey,other vegetables}"          "{turkey,whole milk}"
[11] "{root vegetables,rice}"             "{other vegetables,rice}"
[13] "{whole milk,rice}"                  "{other vegetables,frozen dessert}"
[15] "{whole milk,frozen dessert}"
Itemsets in Consequent (RHS)
[1] "{whole milk}"       "{other vegetables}" "{yogurt}"           "{root vegetables}"
```

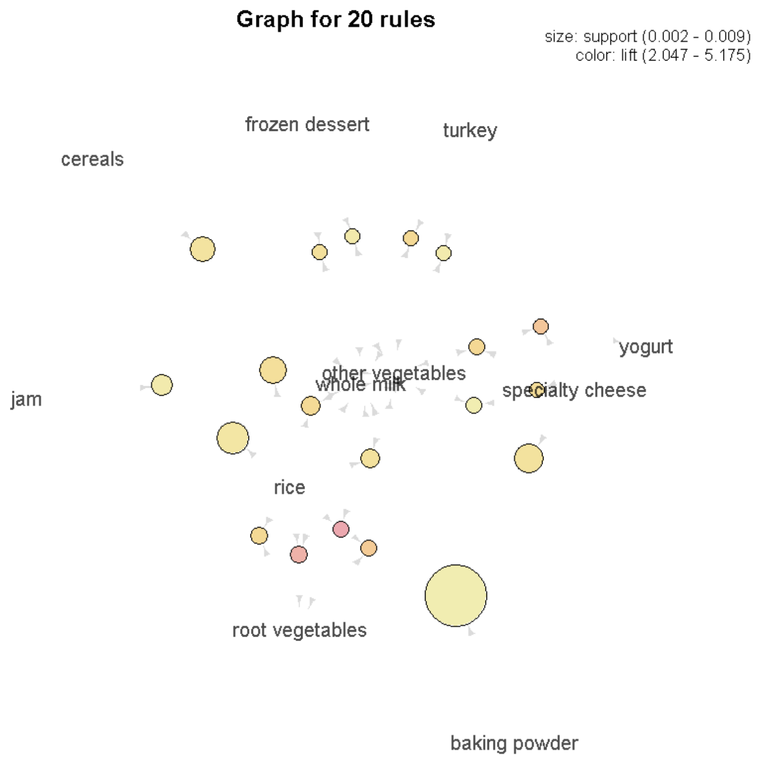


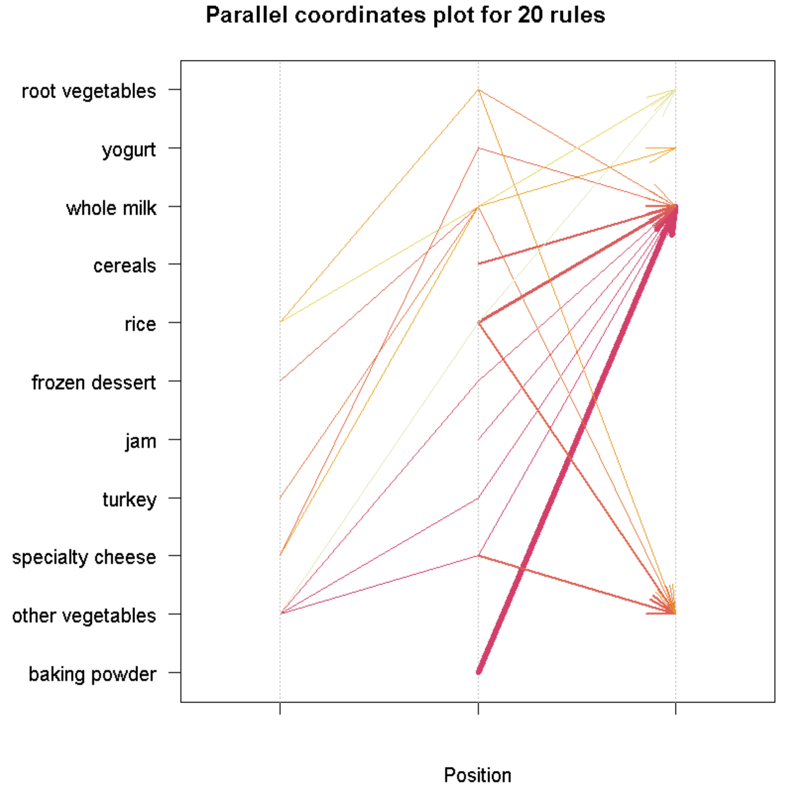


Other Plot Methods

```
plot(rule.a[1:20], method="graph")
```

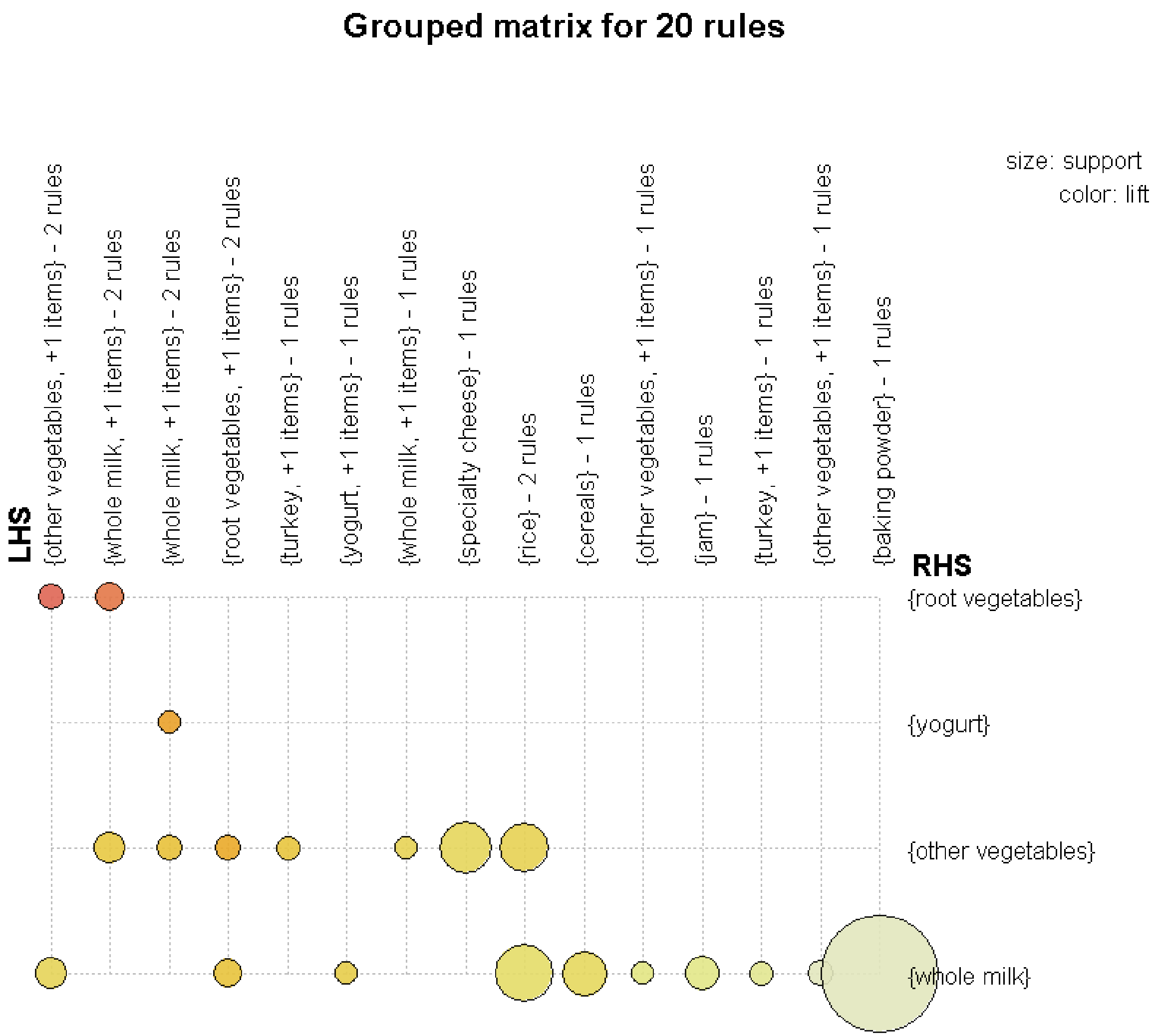
```
plot(rule.a[1:20], method="paracoord")
```





Plot Methods: grouped

```
plot(rule.a[1:20], method="grouped")
```



Case Study 2: The Titanic Dataset

- The Titanic dataset (a 4-dimensional table) gives the values of four categorical attributes for each of the 2201 people on board the Titanic when it struck an iceberg and sank.
- The attributes are social class (first class, second class, third class, crewmember), age (adult or child), sex, and whether or not the person survived.
- To make it suitable for association rule mining, we reconstruct the raw data as `titanic.raw`, where each row represents a person.

```
> Titanic
, , Age = Child, Survived = No
      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No
      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes
      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes
      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```

Chapter 9: Association Rules, R and Data Mining: Examples and Case Studies.
http://www.rdatamining.com/docs/RDataMining.pdf

Reform of The Titanic Dataset

```
> str(Titanic)
 table [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
 - attr(*, "dimnames")=List of 4
  ..$ Class   : chr [1:4] "1st" "2nd" "3rd" "Crew"
  ..$ Sex     : chr [1:2] "Male" "Female"
  ..$ Age     : chr [1:2] "Child" "Adult"
  ..$ Survived: chr [1:2] "No" "Yes"
```

```
> Titanic.df <- as.data.frame(Titanic)
> Titanic.df
   Class    Sex   Age Survived Freq
1    1st   Male Child       No    0
2    2nd   Male Child       No    0
3    3rd   Male Child       No   35
4   Crew   Male Child       No    0
5    1st Female Child       No    0
6    2nd Female Child       No    0
7    3rd Female Child       No   17
8   Crew Female Child       No    0
9    1st   Male Adult       No  118
10   2nd   Male Adult       No  154
11   3rd   Male Adult       No  387
12  Crew   Male Adult       No  670
13   1st Female Adult       No    4
...
31   3rd Female Adult      Yes   76
32  Crew Female Adult      Yes   20
```

```
> # Make a dataset where each row stands for a person,
> # and it can be used for association rule.
> Titanic.raw <- NULL
> for(i in 1:4) {
+     Titanic.raw <- cbind(Titanic.raw,
+          rep(as.character(Titanic.df[,i]),
+          Titanic.df$Freq))
+ }
>
> Titanic.raw <- as.data.frame(Titanic.raw)
> names(Titanic.raw) <- names(Titanic.df)[1:4]
> dim(Titanic.raw)
[1] 2201    4
```

The raw Titanic dataset can also be downloaded from http://www.cs.toronto.edu/~delve/data/titanic/desc.html. The data is file “Dataset.data” in the compressed archive "titanic.tar.gz".

Titanic.raw

```
> str(Titanic.raw)
'data.frame':   2201 obs. of  4 variables:
 $ Class   : Factor w/ 4 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
 $ Age     : Factor w/ 2 levels "Adult","Child": 2 2 2 2 2 2 2 2 2 2 ...
 $ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
> head(Titanic.raw)
  Class  Sex   Age Survived
1   3rd Male Child       No
2   3rd Male Child       No
3   3rd Male Child       No
4   3rd Male Child       No
5   3rd Male Child       No
6   3rd Male Child       No
> summary(Titanic.raw)
  Class          Sex            Age        Survived
 1st :325   Female: 470   Adult:2092   No :1490
 2nd :285   Male  :1731   Child: 109   Yes: 711
 3rd :706
 Crew:885
```

Use apriori{arules}

```
> library(arules)
> # find association rules with default settings
> rules.all <- apriori(Titanic.raw)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
ext
        0.8    0.1    1 none FALSE            TRUE     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: 220

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [27 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> quality(rules.all) <- round(quality(rules.all), digits=3)
> rules.all
set of 27 rules
```

inspect(rules.all)

```
> inspect(rules.all) # or use > arules::inspect(rules.all)
   lhs                                  rhs            support confidence lift
1  {}                                => {Age=Adult}    0.950   0.950      1.000
2  {Class=2nd}                       => {Age=Adult}    0.119   0.916      0.964
3  {Class=1st}                       => {Age=Adult}    0.145   0.982      1.033
4  {Sex=Female}                      => {Age=Adult}    0.193   0.904      0.951
5  {Class=3rd}                       => {Age=Adult}    0.285   0.888      0.934
6  {Survived=Yes}                    => {Age=Adult}    0.297   0.920      0.968
7  {Class=Crew}                      => {Sex=Male}     0.392   0.974      1.238
8  {Class=Crew}                      => {Age=Adult}    0.402   1.000      1.052
9  {Survived=No}                     => {Sex=Male}     0.620   0.915      1.164
10 {Survived=No}                     => {Age=Adult}    0.653   0.965      1.015
11 {Sex=Male}                        => {Age=Adult}    0.757   0.963      1.013
12 {Sex=Female,Survived=Yes}         => {Age=Adult}    0.144   0.919      0.966
13 {Class=3rd,Sex=Male}              => {Survived=No}  0.192   0.827      1.222
14 {Class=3rd,Survived=No}           => {Age=Adult}    0.216   0.902      0.948
15 {Class=3rd,Sex=Male}              => {Age=Adult}    0.210   0.906      0.953
16 {Sex=Male,Survived=Yes}           => {Age=Adult}    0.154   0.921      0.969
17 {Class=Crew,Survived=No}          => {Sex=Male}     0.304   0.996      1.266
18 {Class=Crew,Survived=No}          => {Age=Adult}    0.306   1.000      1.052
19 {Class=Crew,Sex=Male}             => {Age=Adult}    0.392   1.000      1.052
20 {Class=Crew,Age=Adult}            => {Sex=Male}     0.392   0.974      1.238
21 {Sex=Male,Survived=No}            => {Age=Adult}    0.604   0.974      1.025
22 {Age=Adult,Survived=No}           => {Sex=Male}     0.604   0.924      1.175
23 {Class=3rd,Sex=Male,Survived=No}  => {Age=Adult}    0.176   0.917      0.965
24 {Class=3rd,Age=Adult,Survived=No} => {Sex=Male}     0.176   0.813      1.034
25 {Class=3rd,Sex=Male,Age=Adult}    => {Survived=No}  0.176   0.838      1.237
26 {Class=Crew,Sex=Male,Survived=No} => {Age=Adult}    0.304   1.000      1.052
27 {Class=Crew,Age=Adult,Survived=No} => {Sex=Male}    0.304   0.996      1.266
```

Rules with rhs containing "Survived" only


```
> # All other items can appear in the lhs, as set with default="lhs".
> # set minlen to 2 to exclude empty at the left-hand side (lhs) of the first rule
> rules <- apriori(Titanic.raw, control = list(verbose=F),
+     parameter = list(minlen=2, supp=0.005, conf=0.8),
+     appearance = list(rhs=c("Survived=No", "Survived=Yes"),
+     default="lhs"))
>
> quality(rules) <- round(quality(rules), digits=3)
> # Rules are sorted by lift to make high-lift rules appear first
> rules.sorted <- sort(rules, by="lift")
> inspect(rules.sorted)
   lhs                                  rhs            support confidence lift
1  {Class=2nd,Age=Child}             => {Survived=Yes} 0.011   1.000      3.096
7  {Class=2nd,Sex=Female,Age=Child}  => {Survived=Yes} 0.006   1.000      3.096
4  {Class=1st,Sex=Female}            => {Survived=Yes} 0.064   0.972      3.010
10 {Class=1st,Sex=Female,Age=Adult}  => {Survived=Yes} 0.064   0.972      3.010
2  {Class=2nd,Sex=Female}            => {Survived=Yes} 0.042   0.877      2.716
5  {Class=Crew,Sex=Female}           => {Survived=Yes} 0.009   0.870      2.692
11 {Class=Crew,Sex=Female,Age=Adult} => {Survived=Yes} 0.009   0.870      2.692
8  {Class=2nd,Sex=Female,Age=Adult}  => {Survived=Yes} 0.036   0.860      2.663
9  {Class=2nd,Sex=Male,Age=Adult}    => {Survived=No}  0.070   0.917      1.354
3  {Class=2nd,Sex=Male}              => {Survived=No}  0.070   0.860      1.271
12 {Class=3rd,Sex=Male,Age=Adult}    => {Survived=No}  0.176   0.838      1.237
6  {Class=3rd,Sex=Male}              => {Survived=No}  0.192   0.827      1.222
```

the details of progress are suppressed with verbose=F.

NOTE: the minimum support is set to 0.005, so each rule is supported at least by 12 (=ceiling(0.005 *2201)) cases, which is acceptable for a population of 2201.

Removing Redundancy

- When other settings are unchanged, with a lower minimum support, more rules will be produced, and the associations between itemsets shown in the rules will be more likely to be by chance.

- Some rules in `rules.sorted` provide little or no extra information when some other rules are in the result.
  - For example, the above rule #2 provides no extra knowledge in addition to rule #1, since rules #1 tells us that all 2nd-class children survived.

- Generally speaking, when a rule (such as rule #2) is a super rule of another rule (such as rule #1) and the former has the same or a lower lift, the former rule (rule #2) is considered to be redundant.

- Other redundant rules in the above result are rules #4, #7 and #8, compared respectively with rules #3, #6 and #5.

Remove Redundant Rules

```
> # find redundant rules
> # finding subsets in associations and itemMatrix objects
> subset.matrix <- is.subset(rules.sorted, rules.sorted)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
> which(redundant)
 {Class=2nd,Sex=Female,Age=Child,Survived=Yes}
                                             2
 {Class=1st,Sex=Female,Age=Adult,Survived=Yes}
                                             4
{Class=Crew,Sex=Female,Age=Adult,Survived=Yes}
                                             7
 {Class=2nd,Sex=Female,Age=Adult,Survived=Yes}
                                             8
```

- **`is.subset(r1, r2)`**: checks whether r1 is a subset of r2 (i.e., whether r2 is a superset of r1).
- **`lower.tri()`**: returns a logical matrix with TURE in lower triangle.

Interpreting Remaining Rules

```
> # remove redundant rules
> rules.pruned <- rules.sorted[!redundant]
> inspect(rules.pruned)
   lhs                                rhs            support confidence lift
1  {Class=2nd,Age=Child}           => {Survived=Yes} 0.011   1.000      3.096
4  {Class=1st,Sex=Female}          => {Survived=Yes} 0.064   0.972      3.010
2  {Class=2nd,Sex=Female}          => {Survived=Yes} 0.042   0.877      2.716
5  {Class=Crew,Sex=Female}         => {Survived=Yes} 0.009   0.870      2.692
9  {Class=2nd,Sex=Male,Age=Adult}  => {Survived=No}  0.070   0.917      1.354
3  {Class=2nd,Sex=Male}            => {Survived=No}  0.070   0.860      1.271
12 {Class=3rd,Sex=Male,Age=Adult}  => {Survived=No}  0.176   0.838      1.237
6  {Class=3rd,Sex=Male}            => {Survived=No}  0.192   0.827      1.222
```

- It is not uncommon that the association rules are misinterpreted to find their business meanings.
  - For instance, the first rule in **`rules.pruned`** "{Class=2nd, Age=Child} => {Survived=Yes}" has a confidence of one and a lift of three and there are no rules on children of the 1st or 3rd classes. Therefore, it might be interpreted by users as children of the 2nd class had a higher survival rate than other children. This is wrong!
  - The rule states only that all children of class 2 survived, but provides no information at all to compare the survival rates of different classes.

Rules About Children

- To investigate the above issue, we run the code below to find rules whose rhs is "Survived=Yes" and lhs contains "Class=1st", "Class=2nd", "Class=3rd", "Age=Child" and "Age=Adult" only, and which contains no other items (default="none").
- We use lower thresholds for both support and confidence than before to find all rules for children of different classes.

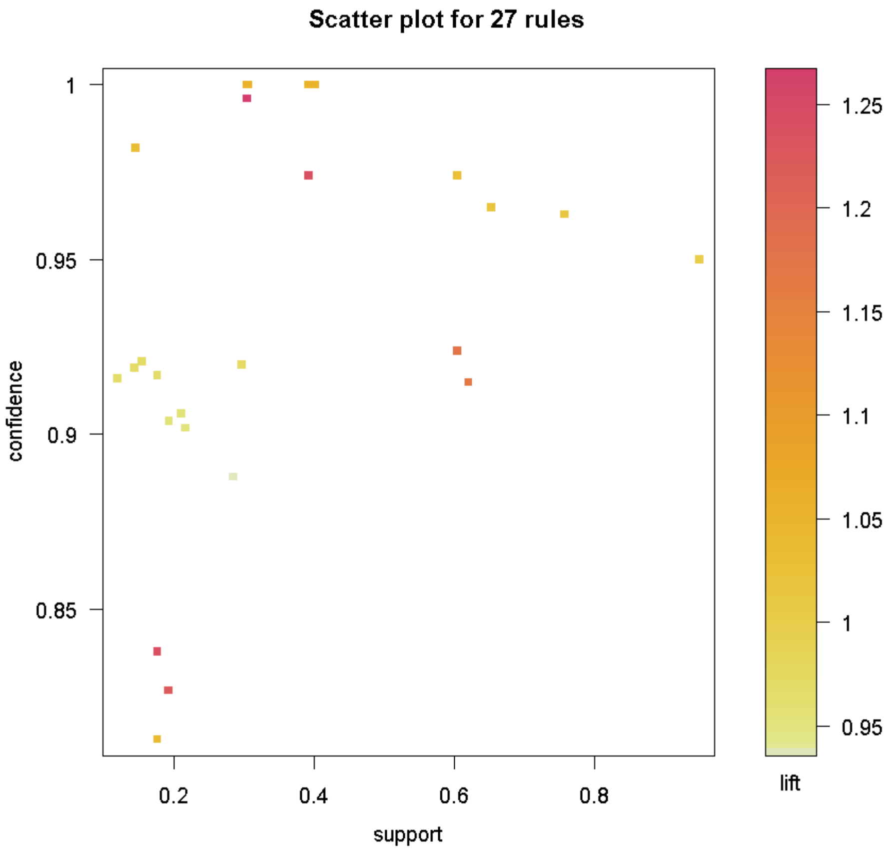
```
> rules <- apriori(Titanic.raw,
+     parameter = list(minlen=3, supp=0.002, conf=0.2),
+     appearance = list(rhs=c("Survived=Yes"),
+     lhs=c("Class=1st", "Class=2nd", "Class=3rd",
+     "Age=Child", "Age=Adult"),
+     default="none"),
+     control = list(verbose=F))
> rules.sorted <- sort(rules, by="confidence")
> inspect(rules.sorted)
  lhs                          rhs            support     confidence lift
1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134 1.0000000  3.0956399
2 {Class=1st,Age=Child} => {Survived=Yes} 0.002726034 1.0000000  3.0956399
5 {Class=1st,Age=Adult} => {Survived=Yes} 0.089504771 0.6175549  1.9117275
4 {Class=2nd,Age=Adult} => {Survived=Yes} 0.042707860 0.3601533  1.1149048
3 {Class=3rd,Age=Child} => {Survived=Yes} 0.012267151 0.3417722  1.0580035
6 {Class=3rd,Age=Adult} => {Survived=Yes} 0.068605179 0.2408293  0.7455209
```

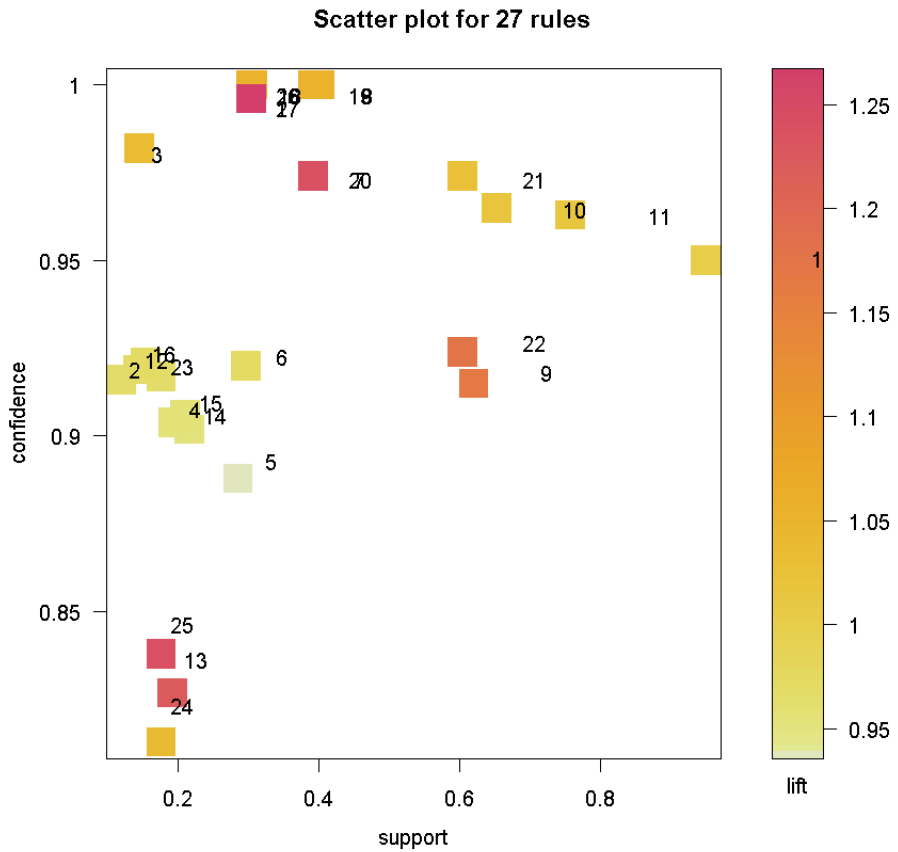
- The first two rules show that children of the 1st class are of the same survival rate as children of the 2nd class and that all of them survived.
- The rule of 1st-class children didn't appear before, simply because of its support was below the threshold specified in Section 5 presents a sad fact that children of class 3 had a low survival rate of 34%, which is comparable with that of 2nd-class adults (see rule 4) and much lower than 1st-class adults (see rule #3).

Visualizing Association Rules

```
plot(rules.all)
```

```
> plot(rules.all, cex=2)
> x <- rules.all@quality$support
> y <- rules.all@quality$confidence
> text(x, y, rownames(rules.all@quality))
```

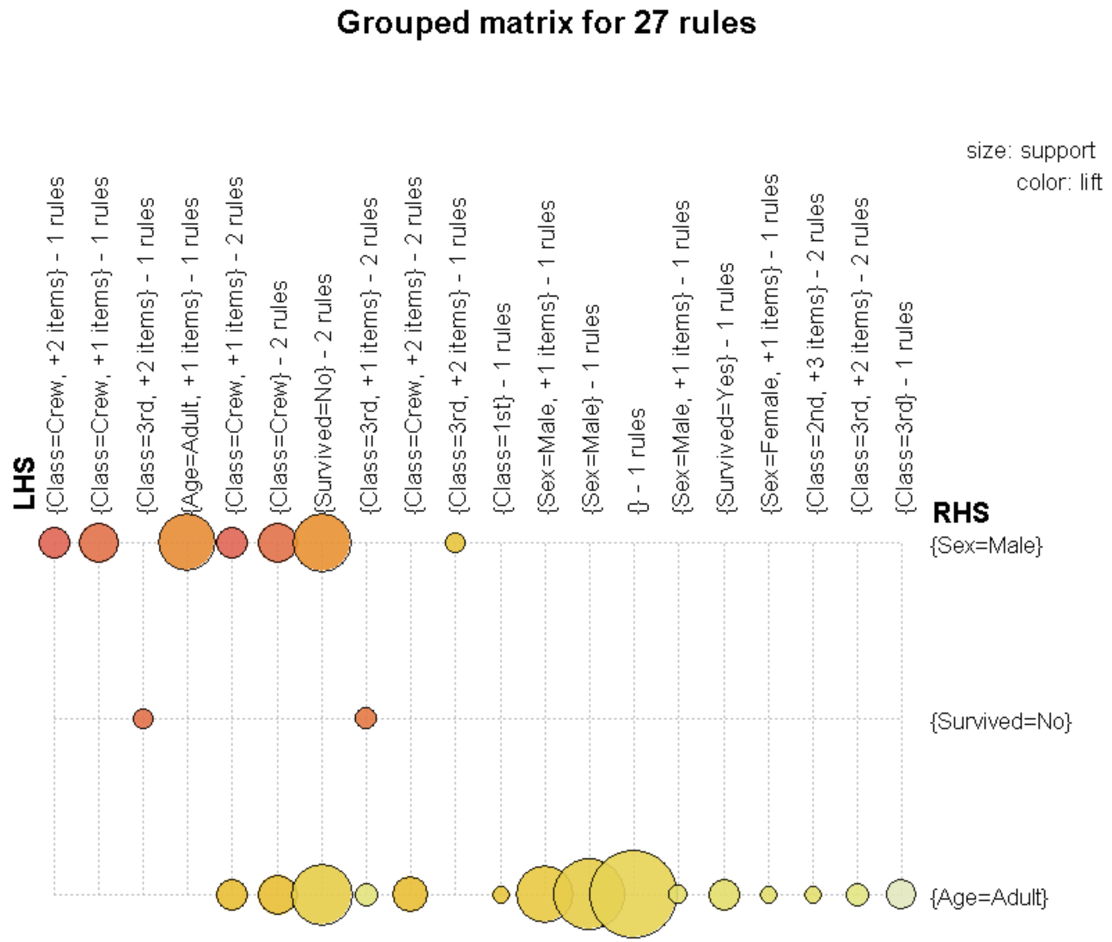




http://cran.r-project.org/web/packages/arulesViz

A Balloon Plot of Association Rules

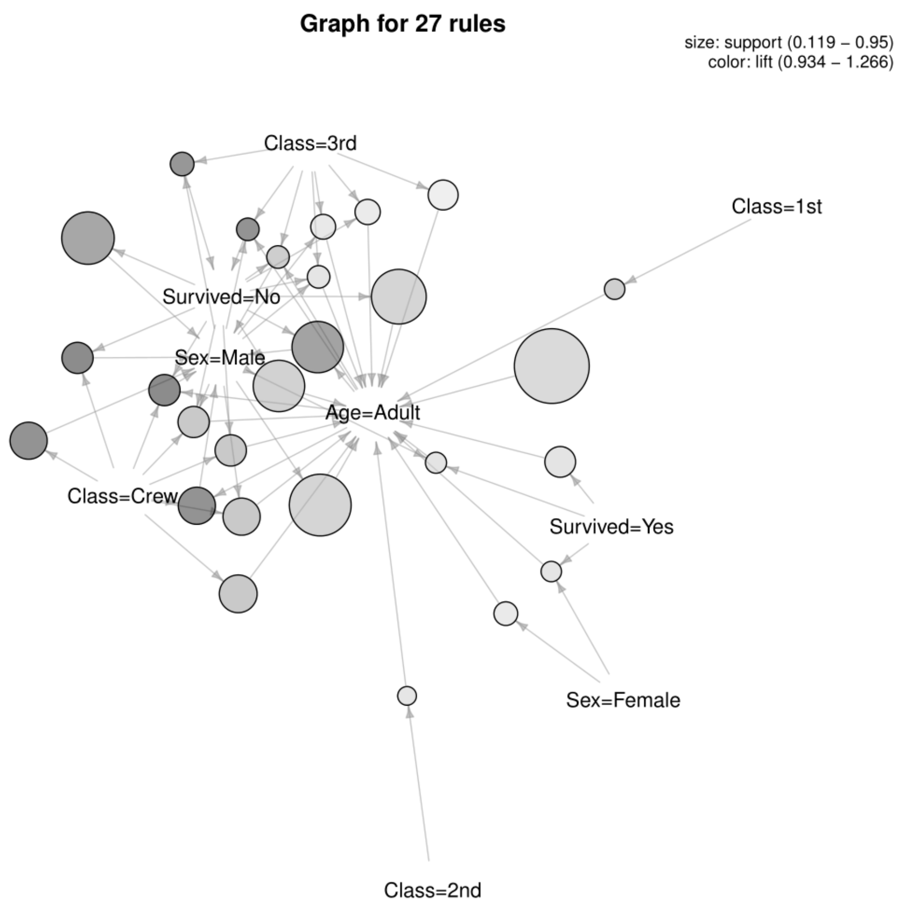
```
plot(rules.all, method="grouped")
```

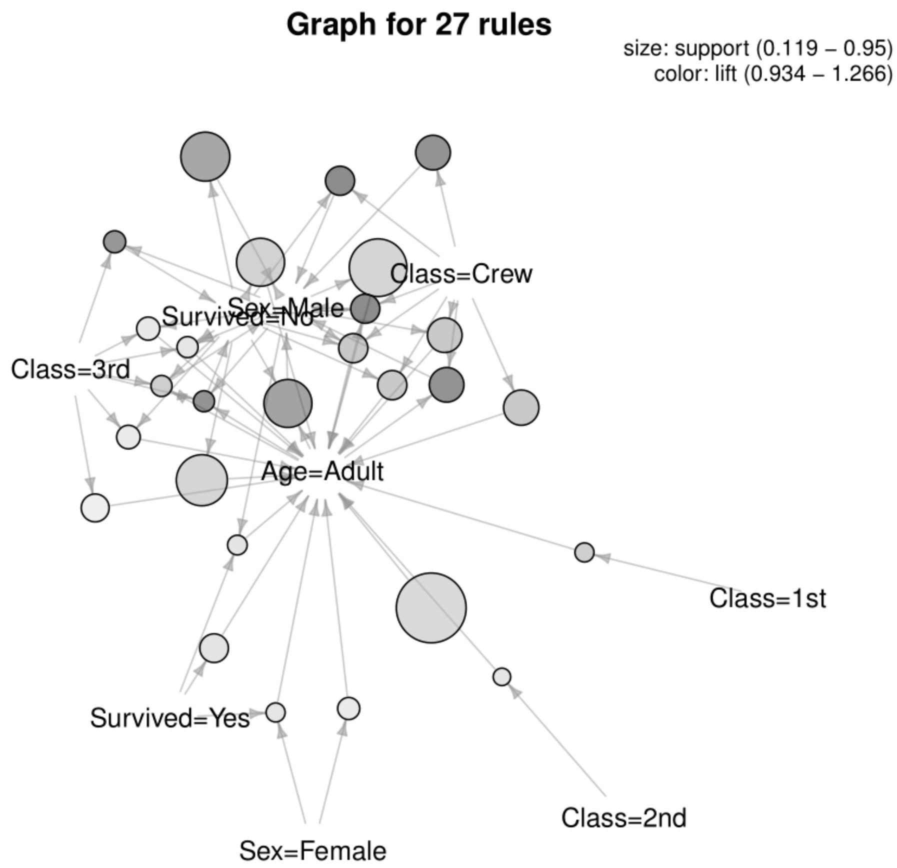


A Graph of Association Rules

```
plot(rules.all, method="graph")
```

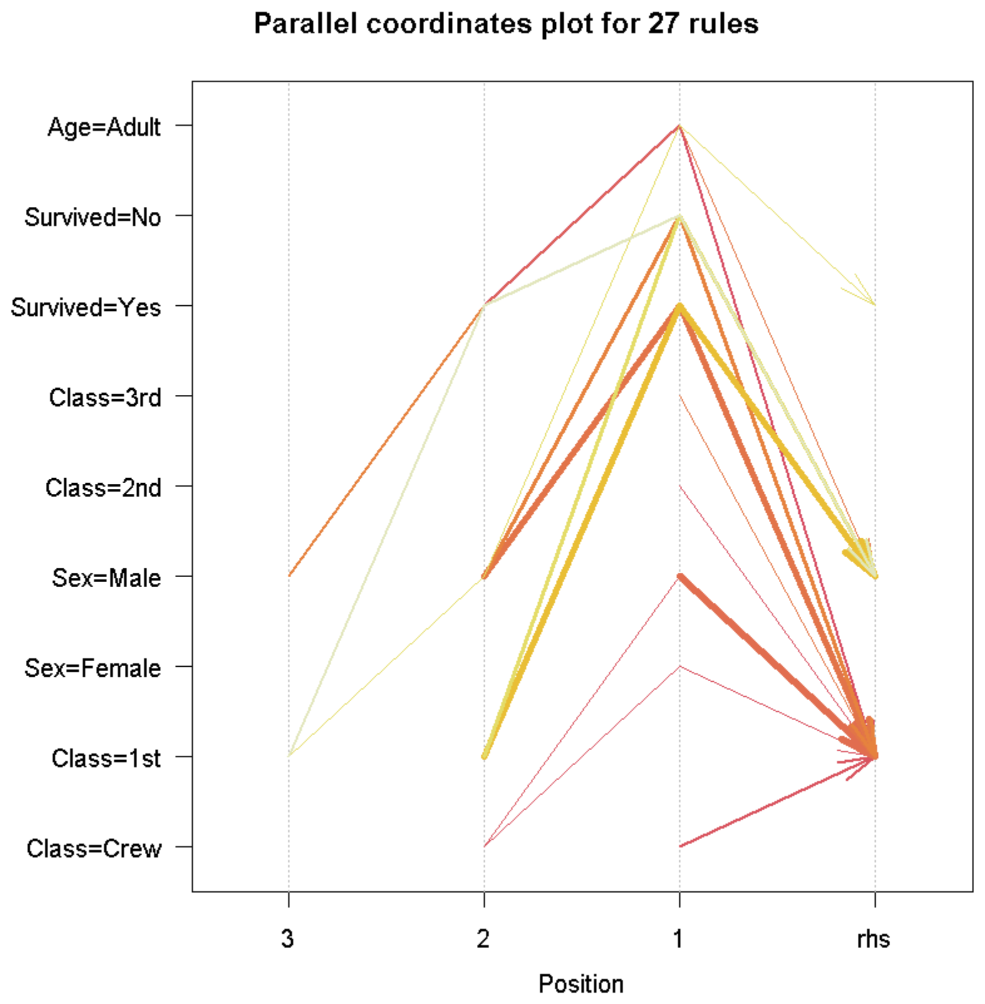
```
plot(rules.all, method="graph",
     control=list(type="items"))
```





A Parallel Coordinates Plot of Association Rules

```
plot(rules.all, method="paracoord", control=list(reorder=TRUE))
```



- How to apply association analysis formulation to non-asymmetric binary variables?

- Example of Association Rule:
  - {Number of Pages ∈[5,10) ∧ (Browser=Mozilla)} ⇒ {Buy = No}
  - Age∈[21,35) ∧ Salary∈[70k,120k) ⇒ Buy
  - Salary∈[70k,120k) ∧ Buy ⇒ Age: $\mu=28$, $\sigma=4$

- **Some Review Papers:**

  - Jyoti Arora, Nidhi Bhalla and Sanjeev Rao, 2013, A Review On Association Rule Mining Algorithms, International Journal of Innovative Research in Computer and Communication Engineering, 1(5), 1246-1251.
  - Xu Chi, and Zhang Wen Fang, 2011, Review of association rule mining algorithm in data mining, Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, pp512 - 516.
  - Rahul B. Diwate and Amit Sahu , 2014,Data Mining Techniques in Association Rule: A Review, International Journal of Computer Science and Information Technologies, 5(1), 2014, 227-229.
  - Pooja Rajendra Harne and Deshpande, 2015, Mining of Association Rules: A Review Paper, International Journal of Environmental Engineering Science and Technology Research, 3(1), pp. 1-8.