

Package ‘arulesSequences’

June 15, 2009

Version 0.1-8

Date 2009-06-15

Title Mining frequent sequences

Author Christian Buchta and Michael Hahsler

Maintainer Christian Buchta <christian.buchta@wu-wien.ac.at>

Description Add-on for arules to handle and mine frequent sequences. Provides interfaces to the C++ implementation of cSPADE by Mohammed J. Zaki.

Depends R (>= 2.7.0), methods, arules (>= 0.6-6)

License GPL-2

Imports arules

LazyLoad yes

OS_type unix

Repository CRAN

Date/Publication 2009-06-15 06:36:37

R topics documented:

c-methods	2
cspade	3
info-methods	5
inspect-methods	7
itemFrequency-methods	9
match-methods	11
read_baskets	13
ruleInduction-methods	14
sequencerules-class	16
sequences-class	18

sgCMatrix-class	20
similarity-methods	21
size-methods	23
SPcontrol-class	24
SPparameter-class	26
subset-methods	27
support-methods	29
timedsequences-class	31
timeFrequency-methods	33
times-methods	34
zaki	35

Index	37
--------------	-----------

c-methods

Combining Objects

Description

c combines a collection of (timed) sequences or sequence rules into a single object.

Usage

```
## S4 method for signature 'sequences':
c(x, ..., recursive = FALSE)

## S4 method for signature 'timedsequences':
c(x, ..., recursive = FALSE)

## S4 method for signature 'sequencerules':
c(x, ..., recursive = FALSE)
```

Arguments

x an object.

... (a list of) further objects of the same class as x

recursive a logical value specifying if the function should descend through lists.

Value

For c and unique an object of the same class as x.

Note

Method `c` is similar to `rbind` but with the added twist that objects are internally conformed matching their item labels. That is, an object based on the union of item labels is created.

For timed sequences event times are currently conformed as follows: if the union of all labels can be cast to integer the labels are sorted. Otherwise, labels not occurring in `x` are appended.

The default setting does not allow any object to be of a class other than `x`, i.e. the objects are not combined into a list.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), [sequencerules](#), method [match](#).

Examples

```
## continue example
example(ruleInduction, package = "arulesSequences")
s <- c(s1, s2)
s
match(unique(s), s1)

## combine rules
r <- c(r2, r2[1:2])
r
match(unique(r), r2)

## combine timed sequences
z <- as(zaki, "timedsequences")
match(z, c(z[1], z[-1]))
```

Description

Mining frequent sequential patterns with the cSPADE algorithm. This algorithm utilizes temporal joins along with efficient lattice search techniques and provides for timing constraints.

Usage

```
cspade(data, parameter = NULL, control = NULL, tmpdir = tempdir())
```

Arguments

<code>data</code>	an object of class <code>transactions</code> with temporal information.
<code>parameter</code>	an object of class <code>SPparameter</code> or a named list with corresponding components.
<code>control</code>	an object of class <code>SPcontrol</code> or a named list with corresponding components.
<code>tmpdir</code>	a non-empty character vector giving the directory name where temporary files are written.

Details

Interfaces the command-line tools for preprocessing and mining frequent sequences with the cSPADE algorithm by M. Zaki via a proper chain of system calls.

The temporal information is taken from components `sequenceID` (sequence or customer identifier) and `eventID` (event identifier) of slot `transactionInfo`. Both identifiers must be in (blockwise) ascending order.

The amount of disk space used by temporary files is reported in verbose mode (see class `SPcontrol`).

The utility function `read_baskets` provides for reading of text files with temporal transaction data.

Value

Returns an object of class `sequences`.

Note

Temporary files may not be deleted until the end of the R session if the call is interrupted.
`sequenceID` and `eventID` are coerced to factor if necessary.

Author(s)

Christian Buchta, Michael Hahsler

References

M. J. Zaki. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning Journal*, **42**, 31–60.

See Also

Class `transactions`, `sequences`, `SPparameter`, `SPcontrol`, method `ruleInduction`, function `read_baskets`.

Examples

```

## use example data from paper
data(zaki)
## mine frequent sequences
s1 <- cspade(zaki, parameter = list(support = 0.4),
            control = list(verbose = TRUE))

summary(s1)
as(s1, "data.frame")
## use timing constraint
s2 <- cspade(zaki, parameter = list(support = 0.4, maxwin = 5))
as(s2, "data.frame")

## replace timestamps
t <- zaki
transactionInfo(t)$eventID <-
  unlist(tapply(seq(t), transactionInfo(t)$sequenceID,
               function(x) x - min(x) + 1), use.names = FALSE)
as(t, "data.frame")
s0 <- cspade(t, parameter = list(support = 0.4))
s0
identical(as(s1, "data.frame"), as(s0, "data.frame"))

## Not run:
## use generated data
t <- read_baskets(con = system.file("misc", "test.txt", package =
                                "arulesSequences"),
                 info = c("sequenceID", "eventID", "SIZE"))

summary(t)
## use low support
s3 <- cspade(t, parameter = list(support=0.03),
            control = list(verbose=TRUE))

summary(s3)
## End(Not run)

```

info-methods

Get/Set Object Information

Description

`sequenceInfo` gets or sets information on the elements of a collection of sequences

`ruleInfo` gets or sets information on the elements of a collection of sequence rules.

`itemInfo` gets or sets information on the set of distinct items associated with a collection of sequences.

`timeInfo` gets or sets information on the event times of a collection of timed sequences.

Usage

```
## S4 method for signature 'sequences':
sequenceInfo(object)

## S4 method for signature 'sequences':
sequenceInfo(object) <- value

## S4 method for signature 'sequencerules':
ruleInfo(object)

## S4 method for signature 'sequencerules':
ruleInfo(object) <- value

## S4 method for signature 'sequences':
itemInfo(object)

## S4 method for signature 'sequences':
itemInfo(object) <- value

## S4 method for signature 'timedsequences':
timeInfo(object)

## S4 method for signature 'timedsequences':
timeInfo(object) <- value
```

Arguments

<code>object</code>	an object.
<code>value</code>	a data frame corresponding with the <i>elements</i> or <i>times</i> of <code>object</code> .

Value

For method `sequenceInfo` and method `ruleInfo` a data frame of information on and corresponding with the elements of `object`.

For method `itemInfo` a data frame of information on and corresponding with the distinct items of `object`.

For method `timeInfo` a data frame of information on and corresponding with the distinct event times of `object`.

Note

For reasons of efficiency the reference set of distinct itemsets may contain unreferenced *elements*, i.e. items that do not occur in any sequence.

Unique item identifiers must be provided in column `labels`.

Unique event time identifiers must be provided in columns `labels` and `eventID`. Note that the latter is used for computation of gaps, etc.

Author(s)

Christian Buchta

See AlsoClass [sequences](#), [timedsequences](#), [sequencerules](#).**Examples**

```
## continue example
example(ruleInduction, package = "arulesSequences")

## empty
sequenceInfo(s2) <- sequenceInfo(s2)
ruleInfo(r2) <- ruleInfo(r2)

## item info
itemInfo(s2)

## time info
z <- as(zaki, "timedsequences")
timeInfo(z)
```

inspect-methods *Display Objects*

Description

`inspect` displays a collection of (timed) sequences or sequence rules and their associated quality measures formatted for online inspection.

`labels` retrieves the string representations of a collection of (timed) sequences or sequence rules.

`itemLabels` gets the string representations of the set of distinct items or itemsets (elements) associated with a collection of sequences, or sets item labels.

Usage

```
## S4 method for signature 'sequences':
inspect(x, setSep = ",", seqStart = "<", seqEnd = ">",
        decode = TRUE)

## S4 method for signature 'timedsequences':
inspect(x, setSep = ",", seqStart = "<", seqEnd = ">",
        decode = TRUE)

## S4 method for signature 'sequencerules':
inspect(x, setSep = ",", seqStart = "<", seqEnd = ">",
        ruleSep = "=>", decode = TRUE)
```

```
## S4 method for signature 'sequences':
labels(object, setSep = ",", seqStart = "<", seqEnd = ">",
        decode = TRUE, ...)

## S4 method for signature 'timedsequences':
labels(object, timeStart = "[", timeEnd = "]", setSep = ",",
        seqStart = "<", seqEnd = ">", decode = TRUE, ...)

## S4 method for signature 'sequencerules':
labels(object, setSep = ",", seqStart = "<", seqEnd = ">",
        ruleSep = " => ", decode = TRUE, ...)

## S4 method for signature 'sequences':
itemLabels(object, itemsets = FALSE, ...)

## S4 method for signature 'sequences, character':
itemLabels(object) <- value
```

Arguments

<code>x</code> , <code>object</code>	an object.
<code>setSep</code>	a string value specifying the itemset (element) separator.
<code>seqStart</code>	a string value specifying the left sequence delimiter.
<code>seqEnd</code>	a string value specifying the right sequence delimiter.
<code>ruleSep</code>	a string value specifying the separator of the left-hand (antecedent) and the right-hand side (consequent) sequence.
<code>timeStart</code>	a string value specifying the left event time delimiter.
<code>timeEnd</code>	a string value specifying the right event time delimiter.
<code>decode</code>	a logical value specifying if the item indexes should be replaced by item labels.
<code>itemsets</code>	a logical value specifying the type of labels.
<code>...</code>	arguments specifying the markup of itemsets: <code>itemSep = "</code> , <code>", setStart = "{", or <code>setEnd = "}</code>".</code>
<code>value</code>	a character vector of length the number of items of <code>object</code> .

Value

For method `inspect` returns `x` invisibly.

For method `labels` a character vector corresponding with the *elements* of `x`.

For method `itemLabels` a character vector corresponding with the distinct items or itemsets of `object`.

Note

For compatibility with package **arules** the markup of itemsets is not customizable in the inspect methods.

For reasons of efficiency the reference set of distinct itemsets may contain unreferenced *elements*, e.g. after subsetting.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), [sequencerules](#), method [subset](#).

Examples

```
## continue example
example(ruleInduction, package = "arulesSequences")

## stacked style
inspect(s2)
inspect(s2, setSep = "->", seqStart = "", seqEnd = "")

## economy style
labels(s2, setSep = "->", seqStart = "", seqEnd = "",
        itemSep = " ", setStart = "", setEnd = "")

## rules
inspect(r2)

## alternate style
labels(r2, ruleSep = " + ")

## itemset labels
itemLabels(s2, itemsets = TRUE)
itemLabels(s2[reduce = TRUE], itemsets = TRUE)

## item labels
itemLabels(s2) <- tolower(itemLabels(s2))
itemLabels(s2)

## timed
z <- as(zaki, "timedsequences")
labels(z)
inspect(z)
```

 itemFrequency-methods

Count Items or Itemsets

Description

itemFrequency counts the number of distinct occurrences of items or itemsets (elements) in a collection of sequences. That is, multiple occurrences within a sequence are ignored.

itemTable cross-tabulates the counts an item or itemset occurs in a sequence.

nitems computes the total number of distinct occurrences of items or itemsets in a collection of sequences.

dim retrieves the dimensions of an object of class `sequences` or `timedsequences`.

length retrieves the number of elements of a collection of sequences or sequence rules.

Usage

```
## S4 method for signature 'sequences':
itemFrequency(x, itemsets = FALSE, type = c("absolute", "relative"))

## S4 method for signature 'sequences':
itemTable(x, itemsets = FALSE)

## S4 method for signature 'sequences':
nitems(x, itemsets = FALSE)

## S4 method for signature 'sequences':
dim(x)

## S4 method for signature 'timedsequences':
dim(x)

## S4 method for signature 'sequences':
length(x)

## S4 method for signature 'sequencerules':
length(x)
```

Arguments

x	an object.
itemsets	a logical value specifying the type of count.
type	a string value specifying the scale of count.

Value

For `itemFrequency` returns a vector of counts corresponding with the reference set of distinct items or itemsets.

For `itemTable` returns a table with the rownames corresponding with the reference set of distinct items or itemsets.

For `nitmes` a scalar value.

For `dim` and class `sequences` a vector of length three containing the number of sequences and the dimension of the reference set of distinct itemsets. For class `timedsequences` the fourth element contains the number of distinct event times.

For `length` a scalar value.

Note

For efficiency reasons, the reference set of distinct itemsets can be larger than the set actually referenced by a collection of sequences. Thus, the counts of some items or itemsets may be zero.

Method `nitems` is provided for efficiency; method `dim` for technical information.

For analysis of a set of rules use the accessors `lhs` or `rhs`, or coerce to sequences.

Author(s)

Christian Buchta

See Also

Class `sequences`, `timedsequences`, method `size`, `subset`.

Examples

```
## continue example
example(cspade)

##
itemFrequency(s2)
itemFrequency(s2, itemsets = TRUE)

##
itemTable(s2)
itemTable(s2, itemsets = TRUE)

##
nitems(s2)
nitems(s2, itemsets = TRUE)

##
length(s2)
dim(s2)

##
z <- as(zaki, "timedsequences")
```

```
dim(z)
```

```
match-methods
```

```
Match Objects
```

Description

`match` finds the positions of first matches of a collection of sequences or sequence rules in an object of the same class.

`%in%` indicates matches of the left in the right operand. If the right operand is a vector of item labels indicates if a sequence contains any of the items given.

`%ain%` indicates if a sequence contains all the items given as the right operand.

`%pin%` indicates if a sequence contains any item matching the regular expression given as the right operand.

`%ein%` indicates if a sequence contains any itemset containing all the items given as the right operand.

`duplicated` indicates duplicate occurrences of sequences or sequence rules.

Usage

```
## S4 method for signature 'sequences, sequences':
match(x, table, nomatch = NA_integer_, incomparables = NULL)
```

```
## S4 method for signature 'sequencerules, sequencerules':
match(x, table, nomatch = NA_integer_, incomparables = NULL)
```

```
## S4 methods for signature 'sequences, character':
```

```
x %in% table
```

```
x %ain% table
```

```
x %pin% table
```

```
x %ein% table
```

```
## S4 method for signature 'sequences':
```

```
duplicated(x, incomparables = FALSE)
```

```
## S4 method for signature 'sequencerules':
```

```
duplicated(x, incomparables = FALSE)
```

Arguments

<code>x</code>	an object.
<code>table</code>	an object (of the same class as <code>x</code>).
<code>nomatch</code>	the value to be returned in the case of no match.
<code>incomparables</code>	not used.

Value

For `match` returns an integer vector of the same length as `x` containing the position in `table` of the first match, or if there is no match the value of `nomatch`.

For `%in%`, `%ain%`, and `%pin%` returns a logical vector indicating for each *element* of `x` if a match was found in the right operand.

For `duplicate` a logical vector corresponding with the *elements* of `x`.

Note

For practical reasons, the item labels given in the right operand must match the item labels associated with `x` exactly.

Currently, an operator for matching against the labels of a set of sequences is not provided. For example, it could be defined as

```
"%lin%" <- function(l, r) match(r, labels(l)) > 0
```

with the caveat of being too general.

FIXME currently matching of timed sequences does not take event times into consideration.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [sequencerules](#), method [labels](#), [itemLabels](#).

Examples

```
## continue example
example(cspade)

## match
labels(s1[match(s2, s1)])
labels(s1[s1 %in% s2])           # the same

## match items
labels(s2[s2 %in% c("B", "F")])
labels(s2[s2 %ain% c("B", "F")])
labels(s2[s2 %pin% "F"])

## match itemsets
labels(s1[s1 %ein% c("F", "B")])
```

read_baskets	<i>Read Transaction Data</i>
--------------	------------------------------

Description

Read transaction data in basket format (with additional temporal or other information) and create an object of class `transactions`.

Usage

```
read_baskets(con, sep = "[ \\t]+", info = NULL, iteminfo = NULL)
```

Arguments

<code>con</code>	an object of class <code>connection</code> or file name.
<code>sep</code>	a regular expression specifying how fields are separated in the data file.
<code>info</code>	a character vector specifying the header for columns with additional transaction information.
<code>iteminfo</code>	a data frame specifying (additional) item information.

Details

Each line of text represents a transaction where items are separated by a pattern matching the regular expression specified by `sep`.

Columns with additional information such as customer or time (event) identifiers are required to come before any item identifiers and must be specified by `info`.

Sequential data are identified by the presence of the column identifiers `sequenceID` (sequence or customer identifier) and `eventID` (time or event identifier) of slot `transactionInfo`.

The row names of `iteminfo` must match the item identifiers present in the data. However, `iteminfo` need not contain a labels column.

Value

An object of class `transactions`.

Note

Currently, it is not checked if column `eventID` defines a temporal order. `sequenceID` and `eventID` will be coerced to factor if necessary.

For efficiency, the item labels are not sorted and, thus, are in the order they appear in the data.

Author(s)

Christian Buchta

See Also

Class [timedsequences](#), [transactions](#), function [cspade](#).

Examples

```
## read example data
x <- read_baskets(con = system.file("misc", "zaki.txt", package =
                                "arulesSequences"),
                 info = c("sequenceID", "eventID", "SIZE"))
as(x, "data.frame")
```

ruleInduction-methods

Induce Sequence Rules

Description

Induce a set of strong sequence rules from a set of frequent sequences, i.e. which (1) satisfy the minimum confidence threshold and (2) which contain the last element of the generating sequence as the right-hand side (consequent) sequence.

Usage

```
## S4 method for signature 'sequences':
ruleInduction(x, transactions, confidence = 0.8, control = NULL)
```

Arguments

`x` an object.

`transactions` currently not used.

`confidence` a numeric value specifying the minimum confidence threshold.

`control` a list with logical components `maximal` specifying if rules should be induced from maximally frequent sequences only, and `verbose` if progress and runtime information should be displayed.

Value

Returns an object of class [sequencerules](#).

Note

Currently, the collection of sequences supplied must be closed with respect to the rules to be induced. That is, the left- and the right-hand side sequence of each candidate rule must be contained in the collection of sequences. However, using timing constraints in the mining step the set of frequent sequences may not be closed under rule induction.

Author(s)

Christian Buchta

See AlsoClass [sequences](#), [sequencerules](#), function [cspade](#).**Examples**

```
## continue example
example(cspade)

## mine rules
r2 <- ruleInduction(s2, confidence = 0.5,
                   control      = list(verbose = TRUE))

summary(r2)
as(r2, "data.frame")
```

 sequencerules-class

Class "sequencerules" — Collections of Sequential Rules

Description

Represents a collection of sequential rules and their associated quality measure. That is, the elements in the consequent occur at a later time than the elements of the antecedent.

Objects from the Class

Typically objects are created by a sequence rule mining algorithm as the result value, e.g. method [ruleInduction](#).

Objects can be created by calls of the form `new("sequencerules", ...)`.

Slots

elements: an object of class [itemsets](#) containing a sparse representation of the unique elements of a sequence.

lhs: an object of class [sgCMatrix](#) containing a sparse representation of the left-hand sides of the rules (antecedent sequences).

rhs: an object of class [sgCMatrix](#) containing a sparse representation of the right-hand sides of the rules (consequent sequences).

ruleInfo: a data.frame which may contain additional information on a sequence rule.

quality: a data.frame containing the quality measures of a sequence rule.

Extends

Class "[associations](#)", directly.

Methods

```

coerce signature(from = "sequencerules", to = "list")
coerce signature(from = "sequencerules", to = "data.frame")
coerce signature(from = "sequencerules", to = "sequences"); coerce a collection of sequence rules to a collection of sequences by appending to each left-hand (antecedent) sequence its right-hand (consequent) sequence.
c signature(x = "sequencerules")
coverage signature(x = "sequencerules"); returns the support values of the left-hand side (antecedent) sequences.
duplicated signature(x = "sequencerules")
labels signature(x = "sequencerules")
ruleInfo signature(object = "sequencerules")
ruleInfo<- signature(object = "sequencerules")
inspect signature(x = "sequencerules")
labels signature(object = "sequencerules")
length signature(x = "sequencerules")
lhs signature(x = "sequencerules")
match signature(x = "sequencerules")
rhs signature(x = "sequencerules")
show signature(object = "sequencerules")
size signature(x = "sequencerules")
subset signature(x = "sequencerules")
summary signature(object = "sequencerules")
unique signature(x = "sequencerules")

```

Note

Some of the methods for sequences are not implemented as objects of this class can be coerced to sequences.

Author(s)

Christian Buchta

See Also

Class [sgCMatrix](#), [itemsets](#), [associations](#), [sequences](#), method [ruleInduction](#), function [cspade](#)

Examples

```
## continue example
example(ruleInduction, package = "arulesSequences")
as(r2, "data.frame")

## coerce to sequences
as(as(r2, "sequences"), "data.frame")
```

sequences-class *Class "sequences" — Collections of Sequences*

Description

Represents a collection of sequences and the associated quality measures.

Objects from the Class

Most frequently, objects are created by a sequence mining algorithm such as cSPADE as the return value.

Objects can also be created by calls of the form `new("sequences", ...)`.

Slots

elements: an object of class `itemsets` containing a sparse representation of the unique elements of a sequence.

data: an object of class `sgCMatrix` containing a sparse representation of ordered *lists* (collections of) indexes into the unique elements.

sequenceInfo: a data frame which may contain additional information on a sequence.

quality: a data.frame containing the quality measures of a sequence.

Extends

Class "`associations`", directly.

Methods

```
coerce signature(from = "sequences", to = "list")
coerce signature(from = "sequences", to = "data.frame")
coerce signature(from = "list", to = "sequences")
%in% signature(x = "sequences", table = "character")
%ain% signature(x = "sequences", table = "character")
%pin% signature(x = "sequences", table = "character")
%ein% signature(x = "sequences", table = "character")
c signature(x = "sequences")
```

```

dim signature(x = "sequences")
duplicated signature(x = "sequences")
labels signature(object = "sequences")
length signature(x = "sequences")
LIST signature(x = "sequences")
match signature(x = "sequences")
nitems signature(x = "sequences")
sequenceInfo signature(object = "sequences")
sequenceInfo<- signature(object = "sequences")
inspect signature(x = "sequences")
is.maximal signature(x = "sequences"); returns a logical vector indicating if a sequence is not a subsequence of any other sequence in x.
is.subset signature(x = "sequences")
is.superset signature(x = "sequences")
itemFrequency signature(x = "sequences")
itemInfo signature(object = "sequences")
itemInfo<- signature(object = "sequences")
itemLabels signature(object = "sequences")
itemLabels<- signature(object = "sequences")
itemTable signature(x = "sequences")
itemsets signature(x = "sequences"); returns the reference set of distinct itemsets (elements).
ruleInduction signature(x = "sequences")
show signature(object = "sequences")
size signature(x = "sequences")
subset signature(x = "sequences")
summary signature(object = "sequences")
support signature(x = "sequences")
unique signature(x = "sequences")

```

Note

Coercion from an object of class [transactions](#) with temporal information to an object of class [sequences](#) is not provided as this information would be lost. Use class [timedsequences](#) instead.

Currently, a general method for concatenation of sequences similar to `cbind`, is not provided.

Author(s)

Christian Buchta

See Also

Class `sgCMatrix`, `timedsequences`, `itemsets`, `associations`, method `ruleInduction`, `FIXME`, function `cspade`, data `zaki`.

Examples

```
## 3 example sequences
x <- list("01" = list(c("A", "B"), "C"),
         "02" = list("C"),
         "03" = list("B", "B"))

## coerce
s <- as(x, "sequences")
as(s, "data.frame")

## get reference set
as(itemsets(s), "data.frame")
```

sgCMatrix-class *Class "sgCMatrix" – Sparse Ordered Lists of Symbols*

Description

Sparse pseudo matrices in column-compressed form for storing ordered *lists* of symbols.

Objects from the Class

Most frequently, an object is created upon creation of an object of class `sequences` or `sequencerules`.

Objects can also be created by calls of the form `new("sgCMatrix", ...)`.

Slots

p: an integer vector of length the number of columns in the matrix plus one. These are *zero-based pointers* into `i`, i.e. to the first element of a list. However, note that the last element contains the number of elements of `i`.

i: an integer vector of length the number of non-zero elements in the matrix. These are *zero-based* symbol indexes, i.e. pointers into the row names if such exist.

Dim: an integer vector representing the number of symbols and the number of lists.

Dimnames: a list with components for symbol and *list* labels.

factors: unused, for compatibility with package **Matrix** only.

Methods

```

coerce signature(from = "sgCMatrx", to = "list")
coerce signature(from = "list", to = "sgCMatrx")
coerce signature(from = "ngCMatrx", to = "sgCMatrx")
dim signature(x = "sgCMatrx")
dimnames signature(x = "sgCMatrx")
dimnames<- signature(x = "sgCMatrx", value = "ANY")
show signature(x = "sgCMatrx")

```

Note

The number of rows can be larger than the number of symbols actually occurring. Thus *i* need not be recoded upon subsetting or two collections of lists with the same index base can be easily combined (column or *row*-wise).

Many of the *methods* of this class implemented in C are currently not interfaced as R methods.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), [sequencerules](#).

Examples

```

## 3 example sequences
x <- list("01" = list(c("A", "B"), "C"),
         "02" = list("C"),
         "03" = list("B", "B"))

## uses paste
s <- as(x, "sgCMatrx")
s

##
dim(s)
dimnames(s)

```

 similarity-methods *Compute Similarities*

Description

Provides the generic function `similarity` and the S4 method to compute similarities among a collection of sequences.

`is.subset`, `is.superset` find subsequence or supersequence relationships among a collection of sequences.

Usage

```
similarity(x, y = NULL, ...)

## S4 method for signature 'sequences':
similarity(x, y = NULL,
           method = c("jaccard", "dice", "cosine", "subset"),
           strict = FALSE)

## S4 method for signature 'sequences':
is.subset(x, y = NULL, proper = FALSE)
## S4 method for signature 'sequences':
is.superset(x, y = NULL, proper = FALSE)
```

Arguments

<code>x</code> , <code>y</code>	an object.
<code>...</code>	further (unused) arguments.
<code>method</code>	a string specifying the similarity measure to use (see details).
<code>strict</code>	a logical value specifying if strict itemset matching should be used.
<code>proper</code>	a logical value specifying if only strict relationships (omitting equality) should be indicated.

Details

Let the number of *common* elements of two sequences refer to those that occur in a longest common subsequence. The following similarity measures are implemented:

jaccard: The number of common elements divided by the total number of elements (the sum of the lengths of the sequences minus the length of the longest common subsequence).

dice: Uses two times the number of common elements.

cosine: Uses the square root of the product of the sequence lengths for the denominator.

subset: Zero if the first sequence is not a subsequence of the second. Otherwise the number of common elements divided by the number of elements in the first sequence.

If `strict = TRUE` the elements (itemsets) of the sequences must be equal to be matched. Otherwise matches are quantified by the similarity of the itemsets (as specified by `method`) thresholded at 0.5, and the common sequence by the sum of the similarities.

Value

For `similarity`, returns an object of class `dsCMatrix` if the result is symmetric (or `method = "subset"`) and an object of class `dgCMatrix` otherwise.

For `is.subset`, `is.superset` returns an object of class `lgCMatrix`.

Note

Computation of the longest common subsequence of two sequences of length n , m takes $O(n*m)$ time.

The supported set of operations for the above matrix classes depends on package **Matrix**. In case of problems, expand to full storage representation using `as(x, "matrix")` or `as.matrix(x)`.

For efficiency use `as(x, "dist")` to convert a symmetric result matrix for clustering.

Author(s)

Christian Buchta

See Also

Class `sequences`, method `dissimilarity`.

Examples

```
## use example data
data(zaki)
z <- as(zaki, "timedsequences")
similarity(z)

# require equality
similarity(z, strict = TRUE)

## emphasize common
similarity(z, method = "dice")

##
is.subset(z)
is.subset(z, proper = TRUE)
```

Description

`size` computes the size of a sequence. This can be either the number of (distinct) itemsets (elements) or items occurring in a sequence.

`ritems` compute the minimum (maximum) number an item or itemset (element) is repeatedly occurring in a sequence.

Usage

```
## S4 method for signature 'sequences':  
size(x, type = c("size", "itemsets", "length", "items"))  
  
## S4 method for signature 'sequences':  
ritems(x, type = c("min", "max"), itemsets = FALSE)
```

Arguments

`x` an object.
`type, itemsets` as string (logical) value specifying the type of count to be computed.

Value

Returns a vector of counts corresponding with the *elements* of object `x`.

Note

The total number of items occurring in a sequence is often referred to as the *length* of the sequence. Similarly, we refer to the total number of itemsets as the `size` of the sequence. Note that we follow this terminology in the summary methods.

For use with a collection of rules use the accessors `lhs` or `rhs`, or coerce to sequences.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#).

Examples

```

## continue example
example(cspade)

## default size
size(s2)
size(s2, "itemsets")
size(s2, "length")
size(s2, "items")

## crosstab
table(length = size(s1, "length"),
       items = size(s1, "items"))

## repetitions
ritems(s1)
ritems(s1, "max")
ritems(s1, "max", TRUE)

```

SPcontrol-class *Class "SPcontrol" — cSPADE Control Parameters*

Description

Provides control parameters for the cSPADE algorithm for mining frequent sequences.

Objects from the Class

A suitable default parameter object will be automatically created by a call to `cspade`. However, the values can be replaced by specifying a named list with the names (partially) matching the slot names of the `SPparameter` class.

Objects can be created by calls of the form `new("SPcontrol", ...)`.

Slots

memsize: an integer value specifying the maximum amount of memory to use (default none [32 MB], range ≥ 16).

numpart: an integer value specifying the number of database partitions to use (default auto, range > 1).

bfstype: a logical value specifying if a breadth-first type of search should be performed (default FALSE [DFS]).

verbose: a logical value specifying if progress and runtime information should be displayed (default FALSE).

summary: a logical value specifying if summary information should be logged (default FALSE).

Methods

```
coerce signature(from = "NULL", to = "SPcontrol")
coerce signature(from = "list", to = "SPcontrol")
coerce signature(from = "SPcontrol", to = "character")
coerce signature(from = "SPcontrol", to = "data.frame")
coerce signature(from = "SPcontrol", to = "list")
coerce signature(from = "SPcontrol", to = "vector")
format signature(x = "SPcontrol")
```

Note

User-supplied values are silently coerced to the target class, e.g. `integer`.

Parameters with no (default) value are not supplied to the mining algorithm, i.e., take the default values implemented there. A default can be unset using `NULL`.

The value of `memsize` implicitly determines the number of database partitions used unless overridden by `numpart`. Usually, the more partitions the less the runtime in the mining stage. However, there may be a trade-off with preprocessing time.

If `summary = TRUE` output on the console from the system calls in the preprocessing and mining steps are written to the file `summary.out` in the current working directory.

Author(s)

Christian Buchta

See Also

Class [SPparameter](#), function [cspade](#).

Examples

```
## coerce from list
p <- as(list(verbose = TRUE), "SPcontrol")
p

## coerce to
as(p, "vector")
as(p, "data.frame")
```

SPparameter-class Class "SPparameter" — cSPADE Mining Parameters

Description

Provides the constraint parameters for the cSPADE algorithm for mining frequent sequences.

Objects from the Class

A suitable default parameter object will be automatically created by a call to `cspade`. However, the values can be replaced by specifying a named list with the names (partially) matching the slot names of the `SPparameter` class.

Objects can be created by calls of the form `new("SPparameter", support, ...)`.

Slots

support: a numeric value specifying the minimum support of a sequence (default 0.1, range [0,1]).

maxsize: an integer value specifying the maximum number of items of an element of a sequence (default 10, range > 0).

maxlen: an integer value specifying the maximum number of elements of a sequence (default 10, range > 0).

mingap: an integer value specifying the minimum time difference between consecutive elements of a sequence (default none, range >= 0).

maxgap: an integer value specifying the maximum time difference between consecutive elements of a sequence (default none, range >= 0).

maxwin: an integer value specifying the maximum time difference between any two elements of a sequence (default none, range >= 0).

Methods

```

coerce signature(from = "NULL", to = "SPparameter")
coerce signature(from = "list", to = "SPparameter")
coerce signature(from = "SPparameter", to = "character")
coerce signature(from = "SPparameter", to = "data.frame")
coerce signature(from = "SPparameter", to = "list")
coerce signature(from = "SPparameter", to = "vector")
format signature(x = "SPparameter")

```

Note

User-supplied values are silently coerced to the target class, e.g. `integer`.

Parameters with no (default) value are not supplied to the mining algorithm, i.e., take the default values implemented there. A value can be unset using `NULL`.

Author(s)

Christian Buchta

See AlsoClass `SPcontrol`, function `cspade`.**Examples**

```
## coerce from list
p <- as(list(maxsize = NULL, maxwin = 5), "SPparameter")
p

## coerce to
as(p, "vector")
as(p, "data.frame")
```

subset-methods

*Subset Objects***Description**

`subset` extracts a subset of a collection of sequences or sequence rules which meet conditions specified with respect to their associated (or derived) quality measures, additional information, or patterns of items or itemsets.

[extracts subsets from a collection of (timed) sequences or sequence rules.

`unique` extracts the unique set of sequences or sequence rules from a collection of sequences or sequence rules.

`lhs`, `rhs` extract the left-hand (antecedent) or right-hand side (consequent) sequences from a collection of sequence rules.

Usage

```
## S4 method for signature 'sequences':
subset(x, subset)

## S4 method for signature 'sequencerules':
subset(x, subset)

## S4 method for signature 'sequences, ANY, ANY, ANY':
x[i, j, ..., reduce = FALSE, drop = FALSE]

## S4 method for signature 'timedsequences, ANY, ANY, ANY':
x[i, j, k, ..., reduce = FALSE, drop = FALSE]

## S4 method for signature 'sequencerules, ANY, missing, ANY':
```

```
x[i, j, ..., drop = FALSE]

## S4 method for signature 'sequences':
unique(x, incomparables = FALSE)

## S4 method for signature 'sequencerules':
unique(x, incomparables = FALSE)

## S4 method for signature 'sequencerules':
lhs(x)

## S4 method for signature 'sequencerules':
rhs(x)
```

Arguments

<code>x</code>	an object.
<code>subset</code>	an expression specifying the conditions where the columns in quality and info must be referenced by their names, and the object itself as <code>x</code> .
<code>i</code>	a vector specifying the subset of <i>elements</i> to be extracted.
<code>k</code>	a vector specifying the subset of event times to be extracted.
<code>reduce</code>	a logical value specifying if the reference set of distinct itemsets should be reduced if possible.
<code>j, ..., drop</code>	unused arguments (for compatibility with package Matrix only).
<code>incomparables</code>	not used.

Value

For `subset`, `[]`, and `unique` returns an object of the same class as `x`.

For `lhs` and `rhs` returns an object of class `sequences`.

Note

In package **arules**, somewhat confusingly, the object itself has to be referenced as `items`. We do not provide this, as well as any of the references `items`, `lhs`, or `rhs`.

After extraction the reference set of distinct itemsets may be larger than the set actually referred to unless reduction to this set is explicitly requested. However, this may increase memory consumption.

Event time indexes of mode character are matched against the time labels. Any duplicate indexes are ignored and their order does not matter, i.e. reordering of a sequence is not possible.

The accessors `lhs` and `rhs` impute the support of a sequence from the support and confidence of a rule. This may lead to numerical inaccuracies over back-to-back derivations.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), [sequencerules](#), method [lhs](#), [rhs](#), [match](#), [nitems](#), [c](#).

Examples

```
## continue example
example(ruleInduction, package = "arulesSequences")

## matching a pattern
as(subset(s2, size(x) > 1), "data.frame")
as(subset(s2, x %ain% c("B", "F")), "data.frame")

## as well as a measure
as(subset(s2, x %ain% c("B", "F") & support == 1), "data.frame")

## matching a pattern in the left-hand side
as(subset(r2, lhs(x) %ain% c("B", "F")), "data.frame")

## matching a derived measure
as(subset(r2, coverage(x) == 1), "data.frame")

## reduce
s <- s2[11, reduce = TRUE]
itemLabels(s)
itemLabels(s2)

## drop initial events
z <- as(zaki, "timedsequences")
summary(z[1,,-1])
```

support-methods *Support Counting for Sequences*

Description

Compute the relative or absolute support of an arbitrary collection of sequences among a set of transactions with additional sequence and temporal information.

Usage

```
## S4 method for signature 'sequences':
support(x, transactions, type= c("relative", "absolute"),
        control = NULL)
```

Arguments

`x` an object.
`transactions` an object of class `transactions`.
`type` a character value specifying the scale of support (relative or absolute).
`control` a named list with logical component `verbose` specifying if progress and run-time information should be displayed.

Value

Returns a numeric vector the elements of which correspond with the elements of `x`.

Note

Currently, only prefix-tree counting is implemented. This approach uses the ordering information of the elements of a sequence only. Therefore, the counts might be higher than those computed by `cspade`.

Author(s)

Christian Buchta

See Also

Class `sequences`, method `ruleInduction`, function `cspade`, `read_baskets`.

Examples

```
## continue example
example(cspade)

## recompute support
s <- support(s2, zaki, control = list(verbose = TRUE))
cbind(as(s2, "data.frame"), upper = s)
```

timedsequences-class

Class "timedsequences" — Collections of Sequences with Timing Information

Description

Represents a collection of (observed) sequences and the associated timing information.

Objects from the Class

Typically, objects are created by coercion from an object of class `transactions`.
 Objects can also be created by calls of the form `new("timedsequences", ...)`.

Slots

time: an object of class `ngCMatrix` containing a sparse representation of the event times of the elements of the sequences. note that the storage layout is the same as for slot data.

timeInfo: a data frame containing the set of time identifiers (column `eventID`) and possibly distinct labels.

elements: inherited from class `sequences`.

data: inherited from class `sequences`.

sequenceInfo: inherited from class `sequences`.

quality: inherited from class `sequences`, usually empty.

Extends

Class "`sequences`", directly. Class "`associations`", by class "`sequences`", distance 2.

Methods

```

coerce signature(from = "transactions", to = "timedsequences")
coerce signature(from = "timedsequences", to = "transactions")
c signature(x = "timedsequences")
dim signature(x = "timedsequences")
labels signature(object = "timedsequences")
LIST signature(x = "timedsequences")
inspect signature(x = "timedsequences")
show signature(object = "timedsequences")
summary signature(object = "timedsequences")
timeFrequency signature(x = "timedsequences")
timeInfo<- signature(object = "timedsequences")
timeInfo signature(object = "timedsequences")
timesets signature(object = "timedsequences")
times signature(x = "timedsequences")
timesets signature(x = "timedsequences"); returns a collection of sequences of
  event times as an object of class itemMatrix.
timeTable signature(x = "timedsequences")

```

Note

The temporal information is taken from components `sequenceID` and `eventID` of slot `transactionInfo`. It may be either on an ordinal or metric scale. The former is always assumed if column `eventID` is a factor.

Note that a sequence must not contain two or more events with the same `eventID`.

Coercion from an object of class `sequences` is not provided as this class does not contain timing information.

FIXME currently objects of this class cannot be used directly in sequence mining with `cspade`.

Author(s)

Christian Buchta

See AlsoClass `itemMatrix`, `transactions`, `sequences`.**Examples**

```
## use example data
data(zaki)

## coerce
z <- as(zaki, "timedsequences")
z

## get time sequences
summary(timesets(z))

## coerce back
as(z, "transactions")
```

timeFrequency-methods

Count Event Times

Description

`timeFrequency` counts the number of occurrences of event times, of the time gaps between the events of a sequence, the minimum or maximum gap of a sequence, or the span of a sequence.

`timeTable` cross-tabulates the above statistics for items or itemsets. For items the sequences are reduced to the events containing the item.

`firstOrder` computes a first order model, i.e. a table of counts of state changes among a collection of timed sequences, where the elements or the times can be the states.

Usage

```
## S4 method for signature 'timedsequences':
timeFrequency(x, type = c("times", "gaps", "mingap", "maxgap",
                          "span"))

## S4 method for signature 'timedsequences':
timeTable(x, type = c("times", "gaps", "mingap", "maxgap", "span"),
          itemsets = FALSE)

## S4 method for signature 'timedsequences':
firstOrder(x, times = FALSE)
```

Arguments

`x` an object.
`type, itemsets, times`
 a string (logical) value specifying the type of count.

Value

For `timeFrequency` returns a vector of counts corresponding with the set of distinct event times, the set of gaps or spans as indicated by the names attribute.

For `timeTable` returns a table of counts with the rownames corresponding with the reference set of distinct items or itemsets.

For `firstOrder` a matrix of counts corresponding with the set of distinct itemsets or event times.

Note

Undefined values are not included in the counts, e.g. the `mingap` of a sequence with one element only. Thus, except for `times` and `gaps` the counts (per item or itemset) always add up to less than or equal the number of sequences, i.e. `length(x)`.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), method [size](#), [times](#), [itemFrequency](#).

Examples

```
## continue example
example("timedsequences-class")

## totals
timeFrequency(z)
timeFrequency(z, "gaps")
timeFrequency(z, "span")

## default items
timeTable(z)
timeTable(z, "gaps")
timeTable(z, "span")

## beware of large data sets
timeTable(z, itemsets = TRUE)

## first order models
firstOrder(z)
firstOrder(z, times = TRUE)
```

Description

Computes the gaps, the minimum or maximum gap, or the span of sequence.

Usage

```
## S4 method for signature 'timedsequences':  
times(x, type = c("times", "gaps", "mingap", "maxgap", "span"))
```

Arguments

`x` an object.
`type` a string value specifying the type of statistic.

Value

If `type = "items"` returns a list of vectors of events times corresponding with the elements of a sequence.

If `type = "gaps"` returns a list of vectors of time differences between consecutive elements of a sequence.

Otherwise, a vector corresponding with the *elements* of `x`.

Note

Gap statistics are not defined for sequences of size one, i.e. which contain a single element. NA is used for undefined values.

FIXME lists are silently reduced to vector if possible.

Author(s)

Christian Buchta

See Also

Class [sequences](#), [timedsequences](#), method [size](#), [itemFrequency](#), [timeFrequency](#).

Examples

```
## continue example  
example("timedsequences-class")  
  
##  
times(z)  
times(z, "gaps")
```

```
## all defined
times(z, "span")

## crosstab
table(size = size(z), span = times(z, "span"))
```

zaki

Zaki Data Set

Description

A small example database for sequence mining provided as an object of class `transactions` and as a text file.

Usage

```
data(zaki)
```

Details

The data set contains the sequential database described in the paper by M. J. Zaki for illustration of the concepts of sequence mining. `sequenceID` and `eventID` denote the sequence and event (time) identifiers of the transactions.

Source

M. J. Zaki. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning Journal*, **42**, 31–60.

See Also

Class `transactions`, `sequences`, function `cspade`.

Examples

```
data(zaki)
summary(zaki)
as(zaki, "data.frame")
```

Index

- *Topic **attribute**
 - c-methods, 2
 - info-methods, 5
- *Topic **classes**
 - sequencerules-class, 15
 - sequences-class, 17
 - sgCMatrix-class, 19
 - SPcontrol-class, 24
 - SPparameter-class, 26
 - timedsequences-class, 30
- *Topic **datasets**
 - zaki, 35
- *Topic **file**
 - read_baskets, 13
- *Topic **manip**
 - c-methods, 2
 - info-methods, 5
 - match-methods, 11
 - similarity-methods, 21
 - subset-methods, 27
- *Topic **methods**
 - size-methods, 23
 - times-methods, 34
- *Topic **models**
 - cspade, 3
 - itemFrequency-methods, 9
 - ruleInduction-methods, 14
 - support-methods, 29
 - timeFrequency-methods, 32
- *Topic **print**
 - inspect-methods, 7
- [(*subset-methods*), 27
- [, sequencerules, ANY, missing, ANY-method (*subset-methods*), 27
- [, sequences, ANY, ANY, ANY-method (*subset-methods*), 27
- [, sgCMatrix, ANY, ANY, ANY-method (*sgCMatrix-class*), 19
- [, timedsequences, ANY, ANY, ANY-method (*subset-methods*), 27
- %ain% (*match-methods*), 11
- %ain%, sequences, character-method (*match-methods*), 11
- %ein% (*match-methods*), 11
- %ein%, sequences, character-method (*match-methods*), 11
- %in% (*match-methods*), 11
- %in%, sequences, character-method (*match-methods*), 11
- %pin% (*match-methods*), 11
- %pin%, sequences, character-method (*match-methods*), 11
- associations, 16, 17, 19, 31
- c, 29
- c (*c-methods*), 2
- c, sequencerules-method (*c-methods*), 2
- c, sequences-method (*c-methods*), 2
- c, timedsequences-method (*c-methods*), 2
- c-methods, 2
- coerce, dsCMatrix, dist-method (*similarity-methods*), 21
- coerce, list, sequences-method (*sequences-class*), 17
- coerce, list, sgCMatrix-method (*sgCMatrix-class*), 19
- coerce, list, SPcontrol-method (*SPcontrol-class*), 24
- coerce, list, SPparameter-method (*SPparameter-class*), 26
- coerce, ngCMatrix, sgCMatrix-method (*sgCMatrix-class*), 19
- coerce, NULL, SPcontrol-method (*SPcontrol-class*), 24
- coerce, NULL, SPparameter-method (*SPparameter-class*), 26

- coerce, sequencerules, data.frame-method
 - (sequencerules-class), 15
- coerce, sequencerules, list-method
 - (sequencerules-class), 15
- coerce, sequencerules, sequences-method
 - (sequencerules-class), 15
- coerce, sequences, data.frame-method
 - (sequences-class), 17
- coerce, sequences, list-method
 - (sequences-class), 17
- coerce, sgCMatrix, list-method
 - (sgCMatrix-class), 19
- coerce, SPcontrol, character-method
 - (SPcontrol-class), 24
- coerce, SPcontrol, data.frame-method
 - (SPcontrol-class), 24
- coerce, SPcontrol, list-method
 - (SPcontrol-class), 24
- coerce, SPcontrol, vector-method
 - (SPcontrol-class), 24
- coerce, SPparameter, character-method
 - (SPparameter-class), 26
- coerce, SPparameter, data.frame-method
 - (SPparameter-class), 26
- coerce, SPparameter, list-method
 - (SPparameter-class), 26
- coerce, SPparameter, vector-method
 - (SPparameter-class), 26
- coerce, timedsequences, transactions-method
 - (timedsequences-class), 30
- coerce, transactions, timedsequences-method
 - (timedsequences-class), 30
- coverage, sequencerules-method
 - (sequencerules-class), 15
- cspade, 3, 14, 15, 17, 19, 24–27, 30, 31, 35
- dgCMatrix, 22
- dim(itemFrequency-methods), 9
- dim, sequences-method
 - (itemFrequency-methods), 9
- dim, sgCMatrix-method
 - (sgCMatrix-class), 19
- dim, timedsequences-method
 - (itemFrequency-methods), 9
- dimnames, sgCMatrix-method
 - (sgCMatrix-class), 19
- dimnames<-, sgCMatrix, ANY-method
 - (sgCMatrix-class), 19
- dissimilarity, 22
- dgCMatrix, 22
 - duplicated(match-methods), 11
 - duplicated, sequencerules-method
 - (match-methods), 11
 - duplicated, sequences-method
 - (match-methods), 11
 - firstOrder
 - (timeFrequency-methods), 32
 - firstOrder, timedsequences-method
 - (timeFrequency-methods), 32
 - format, SPcontrol-method
 - (SPcontrol-class), 24
 - format, SPparameter-method
 - (SPparameter-class), 26
 - generatingItemsets
 - (sequencerules-class), 15
 - info-methods, 5
 - initialize, SPcontrol-method
 - (SPcontrol-class), 24
 - initialize, SPparameter-method
 - (SPparameter-class), 26
 - inspect(inspect-methods), 7
 - inspect, sequencerules-method
 - (inspect-methods), 7
 - inspect, sequences-method
 - (inspect-methods), 7
 - inspect, timedsequences-method
 - (inspect-methods), 7
 - inspect-methods, 7
 - is.maximal, sequences-method
 - (sequences-class), 17
 - is.subset(similarity-methods), 21
 - is.subset, sequences-method
 - (similarity-methods), 21
 - is.superset(similarity-methods), 21
 - is.superset, sequences-method
 - (similarity-methods), 21
 - itemFrequency, 33, 34
 - itemFrequency
 - (itemFrequency-methods), 9
 - itemFrequency, sequences-method
 - (itemFrequency-methods), 9
 - itemFrequency-methods, 9
 - itemInfo(info-methods), 5

- itemInfo, sequences-method
(*info-methods*), 5
- itemInfo<- (*info-methods*), 5
- itemInfo<-, sequences-method
(*info-methods*), 5
- itemLabels, 12
- itemLabels (*inspect-methods*), 7
- itemLabels, sequences-method
(*inspect-methods*), 7
- itemLabels<- (*inspect-methods*), 7
- itemLabels<-, sequences-method
(*inspect-methods*), 7
- itemMatrix, 31, 32
- itemsets, 16–19
- itemsets (*sequences-class*), 17
- itemsets, sequences-method
(*sequences-class*), 17
- itemTable
(*itemFrequency-methods*), 9
- itemTable, sequences-method
(*itemFrequency-methods*), 9

- labels, 12
- labels (*inspect-methods*), 7
- labels, sequencerules-method
(*inspect-methods*), 7
- labels, sequences-method
(*inspect-methods*), 7
- labels, timedsequences-method
(*inspect-methods*), 7
- length (*itemFrequency-methods*), 9
- length, sequencerules-method
(*itemFrequency-methods*), 9
- length, sequences-method
(*itemFrequency-methods*), 9
- lgCMatrix, 22
- lhs, 10, 23, 29
- lhs (*subset-methods*), 27
- lhs, sequencerules-method
(*subset-methods*), 27
- LIST, sequences-method
(*sequences-class*), 17
- LIST, timedsequences-method
(*timedsequences-class*), 30

- match, 2, 29
- match (*match-methods*), 11
- match, sequencerules, sequencerules-method
(*match-methods*), 11
- match, sequences, sequences-method
(*match-methods*), 11
- match-methods, 11

- nitems, 29
- nitems (*itemFrequency-methods*), 9
- nitems, sequences-method
(*itemFrequency-methods*), 9

- read_baskets, 3, 4, 13, 30
- rhs, 10, 23, 29
- rhs (*subset-methods*), 27
- rhs, sequencerules-method
(*subset-methods*), 27
- ritems (*size-methods*), 23
- ritems, sequences-method
(*size-methods*), 23
- ruleInduction, 4, 15, 17, 19, 30
- ruleInduction
(*ruleInduction-methods*), 14
- ruleInduction, sequences-method
(*ruleInduction-methods*), 14
- ruleInduction-methods, 14
- ruleInfo (*info-methods*), 5
- ruleInfo, sequencerules-method
(*info-methods*), 5
- ruleInfo<- (*info-methods*), 5
- ruleInfo<-, sequencerules-method
(*info-methods*), 5

- sequenceInfo (*info-methods*), 5
- sequenceInfo, sequences-method
(*info-methods*), 5
- sequenceInfo<- (*info-methods*), 5
- sequenceInfo<-, sequences-method
(*info-methods*), 5
- sequencerules, 2, 6, 8, 12, 15, 19, 20, 29
- sequencerules-class, 15
- sequences, 2, 4, 6, 8–10, 12, 15, 17, 19, 20,
22, 23, 28–35
- sequences-class, 17
- sgCMatrix, 16, 17, 19
- sgCMatrix-class, 19
- show, sequencerules-method
(*sequencerules-class*), 15
- show, sequences-method
(*sequences-class*), 17
- show, sgCMatrix-method
(*sgCMatrix-class*), 19

- show, SPcontrol-method
(*SPcontrol-class*), 24
- show, SPparameter-method
(*SPparameter-class*), 26
- show, summary.sequencerules-method
(*sequencerules-class*), 15
- show, summary.sequences-method
(*sequences-class*), 17
- show, summary.timedsequences-method
(*timedsequences-class*), 30
- show, timedsequences-method
(*timedsequences-class*), 30
- similarity (similarity-methods),
21
- similarity, sequences-method
(*similarity-methods*), 21
- similarity-methods, 21
- size, 10, 33, 34
- size (*size-methods*), 23
- size, sequences-method
(*size-methods*), 23
- size-methods, 23
- SPcontrol, 3, 4, 27
- SPcontrol-class, 24
- SPparameter, 3, 4, 24–26
- SPparameter-class, 26
- subset, 8, 10
- subset (*subset-methods*), 27
- subset, sequencerules-method
(*subset-methods*), 27
- subset, sequences-method
(*subset-methods*), 27
- subset-methods, 27
- summary, sequencerules-method
(*sequencerules-class*), 15
- summary, sequences-method
(*sequences-class*), 17
- summary, timedsequences-method
(*timedsequences-class*), 30
- summary.sequencerules-class
(*sequencerules-class*), 15
- summary.sequences-class
(*sequences-class*), 17
- summary.timedsequences-class
(*timedsequences-class*), 30
- support (*support-methods*), 29
- support, sequences-method
(*support-methods*), 29
- support-methods, 29
- timedsequences, 2, 6, 8–10, 14, 19, 20,
23, 29, 33, 34
- timedsequences-class, 30
- timeFrequency, 34
- timeFrequency
(*timeFrequency-methods*), 32
- timeFrequency, timedsequences-method
(*timeFrequency-methods*), 32
- timeFrequency-methods, 32
- timeInfo (*info-methods*), 5
- timeInfo, timedsequences-method
(*info-methods*), 5
- timeInfo<- (*info-methods*), 5
- timeInfo<-, timedsequences-method
(*info-methods*), 5
- times, 33
- times (*times-methods*), 34
- times, timedsequences-method
(*times-methods*), 34
- times-methods, 34
- timesets (*timedsequences-class*),
30
- timesets, timedsequences-method
(*timedsequences-class*), 30
- timeTable
(*timeFrequency-methods*), 32
- timeTable, timedsequences-method
(*timeFrequency-methods*), 32
- transactions, 3, 4, 13, 14, 19, 30, 32, 35
- unique (*subset-methods*), 27
- unique, sequencerules-method
(*subset-methods*), 27
- unique, sequences-method
(*subset-methods*), 27
- zaki, 19, 35