Package 'lfl'

April 25, 2017

Type Package

Title Linguistic Fuzzy Logic

Version 1.4

Date 2017-04-25

Author Michal Burda

Maintainer Michal Burda <michal.burda@osu.cz>

Description
Various algorithms related to linguistic fuzzy logic: mining for linguistic fuzzy association rules, composition of fuzzy relations, performing perception-based logical deduction (PbLD), and forecasting time-series using fuzzy rule-based ensemble (FRBE).

License GPL (>= 3.0)

Suggests testthat, doMC

Depends R (>= 3.4.0)

Imports Rcpp (>= 0.11.0), foreach, forecast (>= 5.5), plyr, tseries, e1071, zoo, utils

LinkingTo Rcpp

NeedsCompilation yes

SystemRequirements C++11

Encoding UTF-8

Repository CRAN

Date/Publication 2017-04-25 21:44:22 UTC

R topics documented:

lfl-package 2
aggregateConsequents 4
algebra 6
antecedents 9
as.data.frame.farules 10
as.matrix.fsets 11

cbind.fsets . . . 12
compose . . . 13
consequents . . . 15
defuzz . . . 15
errors . . . 17
evalfrbe . . . 18
farules . . . 19
fcut . . . 20
fire . . . 23
frbe . . . 24
fsets . . . 26
head.farules . . . 28
head.fsets . . . 29
is.farules . . . 30
is.frbe . . . 30
is.fsets . . . 31
is.specific . . . 32
lcut . . . 34
mult . . . 38
pbld . . . 39
perceive . . . 41
plot.fsets . . . 43
print.farules . . . 44
print.frbe . . . 45
print.fsets . . . 46
rbcoverage . . . 47
reduce . . . 48
searchrules . . . 50
sel . . . 52
slices . . . 53
tail.farules . . . 54
tail.fsets . . . 55
triangle . . . 56

Index **58**

`lfl-package` *Linguistic Fuzzy Logic*

Description

This package provides functions to work with various algorithms related to linguistic fuzzy logic. Namely, the following functions are available: mining of linguistic fuzzy association rules, performing perception-based logical deduction (PbLD), and forecasting time-series using fuzzy rule-based ensemble (FRBE).

Details

Package:	lfl
Type:	Package
Version:	1.0
Date:	2015-01-01
License:	GNU/GPL version 3.0 or later

This packages provides functions related with linguistic fuzzy logic.

To convert data into membership degrees of fuzzy sets that model linguistic expressions, see `lcut`. To create fuzzy set partitions see `fcut`.

For fuzzy association rules mining, there is a function `searchrules` that searches for association rules and computes various statistics about them. There is also `reduce` to make small rule bases from mined rules by dropping rules that do not contribute singnificantly to the rule base coverage of source data.

To perform Perception-based Logical Deduction (PbLD), please use `pbld` function. Another ad-hoc inferences may be programmed with help of `fire`, `perceive`, `aggregateConsequents`, or `defuzz` functions.

Fuzzy Rule-based Ensemble (FRBE) is a tool for time-series prediction. Several existing time-series forecasting methods are combined based on features of given time-series to provide a robust forecast – see `frbe`. Also `evalfrbe` may be used to evaluate the performance of forecasting.

Author(s)

Michal Burda

Maintainer: Michal Burda <michal.burda@osu.cz>

Examples

```
# --- SEARCHING FOR RULES ---
# split data into training and testing set
testing <- CO2[1:5, ]
training <- CO2[-1 * 1:5, ]

# custom context of the RHS variable
uptakeContext <- c(7, 28.3, 46)

# convert training data into fuzzy sets
d <- lcut3(training, context=list(uptake=uptakeContext))

# search for rules
r <- searchrules(d, lhs=1:38, rhs=39:58)



# --- PBLD INFERENCE WITH FOUND RULES ---
# convert testing data info fuzzy sets
x <- lcut3(testing, context=list(uptake=uptakeContext))

# prepare values and partition
```

```
v <- slices(uptakeContext[1], uptakeContext[3], 1000)
p <- lcut3(v, name='uptake', context=uptakeContext)

# do the inference
pbld(x, r, p, v)



# --- FRBE TIME-SERIES FORECASTING ---
# prepare data (from the forecast package)
library(forecast)
horizon <- 10
train <- wineind[-1 * (length(wineind)-horizon+1):length(wineind)]
test <- wineind[(length(wineind)-horizon+1):length(wineind)]

# compute forecast
f <- frbe(ts(train, frequency=frequency(wineind)), h=horizon)

# display the forecast
f$mean

# evaluate the results
evalfrbe(f, test)
```

`aggregateConsequents` *Implicational aggregation of rules' consequents into a fuzzy set*

Description

Take a character vector of consequent names, a numeric vector of membership degrees and a matrix that models fuzzy sets corresponding to the consequent names and perform an aggregation of the consequents into a fuzzy set in an implicational way.

This function is typically used within an inference mechanism after a set of firing rules is determined and the membership degrees of their antecedents is computed to combine the consequents of the firing rules into a fuzzy set. The result of this function is then typically defuzzified to obtain crisp result of the inference.

Usage

```
aggregateConsequents(conseq,
          degrees,
          partition)
```

Arguments

`conseq`	A character vector of consequents. Each value in the vector must correspond to a name of some column of the `partition` matrix. The length of this vector must be the same as of the `degrees` argument.

degrees — A numeric vector of membership degrees at which the corresponding consequents (see argument conseq) are fired.

partition — A matrix of membership degrees that describes the meaning of the consequents in vector conseq: each column of the matrix corresponds to a fuzzy set that models a single consequent (of a name given by column names of the matrix), each row corresponds to a single crisp value (which is not important for this function), hence each cell corresponds to a membership degree in which the crisp value is a member of a fuzzy set modelling the consequent. Each consequent in conseq must correspond to some column of this matrix. Such matrix may be created e.g. by using the fcut or lcut functions.

Details

Function assumes a set of implicative rules with antecedents firing at degrees given in degrees and with consequents in conseq. The meaning of the consequents is modeled with fuzzy sets whose membership degree values are captured in the partition matrix.

The function computes a fuzzy set that results from a conjunction of all provided implicative rules. For implication and conjunction, the Lukasiewics implication and the minimum t-norm is used, respectively.

Value

A vector of membership degrees of fuzzy set elements that correspond to rows in the partition matrix. If empty vector of consequents is provided, a vector of 1's is returned. The length of the resulting vector equals to the number of rows of the partition matrix.

Author(s)

Michal Burda

See Also

fire, perceive, defuzz, fcut, lcut

Examples

```
# create a partition matrix
partition <- matrix(c(0:10/10, 10:0/10, rep(0, 5),
                      rep(0, 5), 0:10/10, 10:0/10,
                      0:12/12, 1, 12:0/12),
                    byrow=FALSE,
                    ncol=3)
colnames(partition) <- c('a', 'b', 'c')

# the result of aggregation is equal to:
# pmin(1, partition[, 1] + (1 - 0.5), partition[, 2] + (1 - 0.8))
aggregateConsequents(c('a', 'b'), c(0.5, 0.8), partition)
```

algebra *Algebra for Fuzzy Sets*

Description

Compute triangular norms (t-norms), triangular conorms (t-conorms), residua, bi-residua, and negations.

Usage

```
## a t-norm from concatenated arguments:
goedel.tnorm(..., na.rm=FALSE)
lukas.tnorm(..., na.rm=FALSE)
goguen.tnorm(..., na.rm=FALSE)

## a t-norm in a parallel (element-wise) manner:
pgoedel.tnorm(..., na.rm=FALSE)
plukas.tnorm(..., na.rm=FALSE)
pgoguen.tnorm(..., na.rm=FALSE)

## a t-conorm from concatenated arguments:
goedel.tconorm(..., na.rm=FALSE)
lukas.tconorm(..., na.rm=FALSE)
goguen.tconorm(..., na.rm=FALSE)

## a t-conorm in a parallel (element-wise) manner:
pgoedel.tconorm(..., na.rm=FALSE)
plukas.tconorm(..., na.rm=FALSE)
pgoguen.tconorm(..., na.rm=FALSE)

## compute a residuum (implication)
goedel.residuum(x, y)
lukas.residuum(x, y)
goguen.residuum(x, y)

## compute a bi-residuum (equivalence)
goedel.biresiduum(x, y)
lukas.biresiduum(x, y)
goguen.biresiduum(x, y)

## compute a negation
invol.neg(x)
strict.neg(x)

## algebra-related functions
algebra(name, stdneg=FALSE, ...)
is.algebra(a)
```

Arguments

...	For t-norms and t-conorms, these arguments are numeric vectors of values to compute t-norms or t-conorms from. Values outside the $[0, 1]$ interval cause an error. NA values are also permitted. For the `algebra()` function, these arguments are passed to the factory functions that create the algebra. (Reserved for future use).
na.rm	whether to ignore NA values: `TRUE` means that NA's are ignored, i.e. the computation is performed as if such values were not present in the arguments; `FALSE` means that the NA's in arguments are taken into considerations, details below.
x	Numeric vector of values to compute a residuum or bi-residuum from. Values outside the $[0, 1]$ interval cause an error. NA values are also permitted.
y	Numeric vector of values to compute a residuum or bi-residuum from. Values outside the $[0, 1]$ interval cause an error. NA values are also permitted.
name	The name of the algebra to be created. Must be one of: "goedel", "lukasiewicz", "goguen" (or an unambiguous abbreviation).
stdneg	If `TRUE` the use of a "standard" negation (i.e. involutive negation) is forced. Otherwise, the appropriate negation is used in the algebra (e.g. strict negation in Goedel and Goguen algebra and involutive negation in Lukasiewicz algebra).
a	An object to be checked if it is a valid algebra (i.e. a list returned by the `algebra` function).

Details

`goedel.tnorm`, `lukas.tnorm`, and `goguen.tnorm` compute the Goedel, Lukasiewicz, and Goguen triangular norm (t-norm) from all values in the arguments. If the arguments are vectors they are combined together firstly so that a numeric vector of length 1 is returned.

`pgoedel.tnorm`, `plukas.tnorm`, and `pgoguen.tnorm` compute the same t-norms, but in a parallel manner (element-wisely). I.e. the values with indices 1 of all arguments are used to compute the t-norm, then the second values (while recycling the vectors if they do not have the same size) so that the result is a vector of values.

`goedel.tconorm`, `lukas.tconorm`, `goguen.tconorm`, are similar to the previously mentioned functions, exept that they compute triangular conorms (t-conorms). `pgoedel.tconorm`, `plukas.tconorm`, and `pgoguen.tconorm` are their parallel (i.e. element-wise) alternatives.

`goedel.residuum`, `lukas.residuum`, and `goguen.residuum` compute residua (i.e. implications) and `goedel.biresiduum`, `lukas.biresiduum`, and `goguen.biresiduum` compute bi-residua.

`invol.neg` and `strict.neg` compute the involutive and strict negation, respectively.

Let a, b be values from the interval $[0, 1]$. Here are mathematical definitions of the realized functions:

- Goedel t-norm: $min(a, b)$;
- Goguen t-norm: ab;
- Lukasiewicz t-norm: $max(0, a + b - 1)$;
- Goedel t-conorm: $max(a, b)$;
- Goguen t-conorm: $a + b - ab$;

- Lukasiewicz t-conorm: $min(1, a + b)$;
- Goedel residuum (standard Goedel implication): 1 if $a \leq b$ and b otherwise;
- Goguen residuum (implication): 1 if $a \leq b$ and b/a otherwise;
- Lukasiewicz residuum (standard Lukasiewicz implication): 1 if $a \leq b$ and $1 - a + b$ otherwise;
- Involutive negation: $1 - x$;
- Strict negation: 1 if $x = 0$ and 0 otherwise.

Bi-residuum B is derived from t-norm T and residuum R as follows:

$$B(a, b) = T(R(a, b), R(b, a)).$$

The arguments have to be numbers from the interval $[0, 1]$. Values outside that range cause an error. Also NaN causes an error.

If `na.rm=TRUE` then missing values (NA) are ignored. Otherwise, they are treated as unknown values accordingly to Kleene logic. See the examples below.

`algebra` returns a named list of functions that together form Goedel, Goguen, or Lukasiewicz algebra:

- `"goedel"`: strict negation and Goedel t-norm, t-conorm, residuum, and bi-residuum;
- `"goguen"`: strict negation and Goguen t-norm, t-conorm, residuum, and bi-residuum;
- `"lukasiewicz"`: involutive negation and Lukasiewicz t-norm, t-conorm, residuum, and bi-residuum.

`is.algebra` tests whether the given `a` argument is a valid algebra, i.e. a list returned by the `algebra` function.

Value

Functions for t-norms and t-conorms (such as `goedel.tnorm`) return a numeric vector of size 1 that is the result of the appropriate t-norm or t-conorm applied on all values of all arguments.

Parallel versions of t-norms and t-conorms (such as `pgoedel.tnorm`) return a vector of results after applying the appropriate t-norm or t-conorm on argument in an element-wise (i.e. parallel, by indices) way. The resulting vector is of length of the longest argument (shorter arguments are recycled).

Residua and bi-residua functions return a numeric vector of length of the longest argument (shorter argument is recycled).

`strict.neg` and `invol.neg` compute negations and return a numeric vector of the same size as the argument `x`.

`algebra` returns a list of functions of the requested algebra: `"n"` (negation), `"t"` (t-norm), `"pt"` (parallel, i.e. element-wise, t-norm), `"c"` (t-conorm), `"pc"` (parallel t-conorm), `"r"` (residuum), and `"b"` (bi-residuum).

Author(s)

Michal Burda

Examples

```
# direct and parallel version of functions
goedel.tnorm(c(0.3, 0.2, 0.5), c(0.8, 0.1, 0.5))  # 0.1
pgoedel.tnorm(c(0.3, 0.2, 0.5), c(0.8, 0.1, 0.5)) # c(0.3, 0.1, 0.5)

# handling of missing values
goedel.tnorm(c(0.3, 0, NA), na.rm=TRUE)    # 0
goedel.tnorm(c(0.3, 0.7, NA), na.rm=TRUE)  # 0.3

goedel.tnorm(c(0.3, 0, NA), na.rm=FALSE)   # 0
goedel.tnorm(c(0.3, 0.7, NA), na.rm=FALSE) # NA

goedel.tconorm(c(0.3, 1, NA), na.rm=TRUE)    # 1
goedel.tconorm(c(0.3, 0.7, NA), na.rm=TRUE)  # 0.7

goedel.tconorm(c(0.3, 1, NA), na.rm=FALSE)   # 1
goedel.tconorm(c(0.3, 0.7, NA), na.rm=FALSE) # NA

# algebras
x <- runif(10)
y <- runif(10)
a <- algebra('goedel')
a$n(x)     # negation
a$t(x, y)  # t-norm
a$pt(x, y) # parallel t-norm
a$c(x, y)  # t-conorm
a$pc(x, y) # parallel t-conorm
a$r(x, y)  # residuum
a$b(x, y)  # bi-residuum

is.algebra(a) # TRUE
```

`antecedents` *Extract antecedent-part (LHS) of the rules in a list*

Description

Given a list of rules, the function returns a list of antecedents (i.e. left hand side) of the rules.

Usage

```
antecedents(rules)
```

Arguments

`rules`	Either a list of rules or an object of class `farules`.

Details

This function assumes `rules` to be a list of character vectors where the first element of each vector is a consequent part of the rule and the rest is the antecedent part of the rule. Function returns a list of antecedents.

Value

A list of character vectors.

Author(s)

Michal Burda

See Also

`farules`

`as.data.frame.farules` *Convert the 'farules' object into a data frame*

Description

This function converts an instance of class `farules` into a data frame.

Usage

```
## S3 method for class 'farules'
as.data.frame(x, ...)
```

Arguments

`x`	An instance of class `farules` to be transformed.
`...`	Unused.

Details

This function converts an instance of class `farules` into a data frame. Empty `farules` object is converted into an empty `data.frame` object.

Value

A data frame of statistics of the rules that are stored in the given `farules` object. Row names of the resulting data frame are in the form: `A1 & A2 & ... & An => C`, where `Ai` are antecedent predicates and `C` is a consequent. Empty `farules` object is converted into an empty `data.frame` object.

Author(s)

Michal Burda

See Also

farules

---

as.matrix.fsets *Convert a 'fsets' object into matrix*

Description

This function converts an instance of class fsets into matrix.

Usage

```
## S3 method for class 'fsets'
as.matrix(x, ...)
```

Arguments

x	An instance of class fsets to be transformed.
...	Unused.

Details

This function converts an instance of class fsets into matrix. Also the vars and specs attributes are deleted.

Value

A numeric matrix of membership degrees.

Author(s)

Michal Burda

See Also

fsets, fcut, lcut

Examples

```
ff <- fcut(runif(10), breaks=c(0, 0.5, 1), name='age')
as.matrix(ff)
```

`cbind.fsets` *Combine several 'fsets' objects into a single one*

Description

Take a sequence of objects of class 'fsets' and combine them by columns. This version of cbind takes care of the `vars` and `specs` attributes of the arguments and merges them to the result.

Usage

```
cbind.fsets(..., deparse.level = 1)
```

Arguments

`...`	A sequence of objects of class 'fsets' to be merged by columns.
`deparse.level`	This argument has currently no function and is added here only for compatibility with generic `cbind` function.

Details

Take a sequence of objects of class 'fsets' and combine them by columns. This version of cbind takes care of the `vars` and `specs` attributes of the arguments and merges them to the result. If some argument does not inherit from class 'fsets' an error is thrown.

The `vars` attribute is merged by concatenating the `vars` attributes of each argument. Also the `specs` attributes of the arguments are merged together.

Value

An object of class 'fsets' that is created by merging the arguments by columns. Also the arguments' attributes `vars` and `specs` are merged together.

Author(s)

Michal Burda

See Also

`vars`, `specs`, `fcut`, `lcut`, `farules`

Examples

```
d1 <- lcut3(CO2[, 1:2])
d2 <- lcut3(CO2[, 3:4])
r <- cbind(d1, d2)

print(colnames(d1))
print(colnames(d2))
print(colnames(r))
```

```
print(vars(d1))
print(vars(d2))
print(vars(r))

print(specs(d1))
print(specs(d2))
print(specs(r))
```

`compose` *Composition of Fuzzy Relations*

Description

Composition of Fuzzy Relations

Usage

```
compose(x,
        y,
        e=NULL,
        alg=c('goedel', 'goguen', 'lukasiewicz'),
        type=c('basic', 'sub', 'super', 'square'),
        quantifier=NULL)
```

Arguments

`x`	A first fuzzy relation to be composed. It must be a numeric matrix with values within the $[0, 1]$ interval. The number of columns must match with the number of rows of the `y` matrix.
`y`	A second fuzzy relation to be composed. It must be a numeric matrix with values within the $[0, 1]$ interval. The number of columns must match with the number of rows of the `x` matrix.
`e`	An excluding fuzzy relation. (EXPERIMENTAL FEATURE!) If not NULL, it must be a numeric matrix with dimensions equal to the `y` matrix.
`alg`	An algebra to be used for composition. It must be one of `'goedel'` (default), `'goguen'`, or `'lukasiewicz'`.
`type`	A type of a composition to be performed. It must be one of `'basic'` (default), `'sub'`, `'super'`, or `'square'`.
`quantifier`	If not NULL, it must be a function taking a single argument, a vector of relative cardinalities, that would be translated into membership degrees.

Details

Function composes a fuzzy relation `x` (i.e. a numeric matrix of size (u, v)) with a fuzzy relation `y` (i.e. a numeric matrix of size (v, w)) and possibly with the use of an exclusion fuzzy relation `e` (i.e. a numeric matrix of size (v, w)).

The style of composition is determined by the algebra `alg`, the composition type `type`, and possibly also by a `quantifier`.

Value

A matrix with v rows and w columns, where v is the number of rows of `x` and w is the number of columns of `y`.

Author(s)

Michal Burda

See Also

`algebra`, `mult`

Examples

```
R <- matrix(c(0.1, 0.6, 1, 0, 0, 0,
              0, 0.3, 0.7, 0.9, 1, 1,
              0, 0, 0.6, 0.8, 1, 0,
              0, 1, 0.5, 0, 0, 0,
              0, 0, 1, 1, 0, 0), byrow=TRUE, nrow=5)

S <- matrix(c(0.9, 1, 0.9, 1,
              1, 1, 1, 1,
              0.1, 0.2, 0, 0.2,
              0, 0, 0, 0,
              0.7, 0.6, 0.5, 0.4,
              1, 0.9, 0.7, 0.6), byrow=TRUE, nrow=6)

RS <- matrix(c(0.6, 0.6, 0.6, 0.6,
               1, 0.9, 0.7, 0.6,
               0.7, 0.6, 0.5, 0.4,
               1, 1, 1, 1,
               0.1, 0.2, 0, 0.2), byrow=TRUE, nrow=5)

compose(R, S, alg='goedel', type='basic') # should equal to RS
```

consequents *Extract consequent-part (RHS) of the rules in a list*

Description

Given a list of rules, the function returns a vector of consequents (i.e. right hand side) of the rules.

Usage

```
consequents(rules)
```

Arguments

rules	Either a list of rules or an object of class farules.

Details

This function assumes rules to be a list of character vectors where the first element of each vector is a consequent part of the rule and the rest is the consequent part of the rule. Function returns a vector of consequents.

Value

A character vectors.

Author(s)

Michal Burda

See Also

farules

defuzz *Convert fuzzy set into a crisp numeric value*

Description

Take a fuzzy set in the form of a vector of membership degrees and a vector of numeric values that correspond to that degrees and perform a selected type of defuzzification, i.e. conversion of the fuzzy set into a single crisp value.

Usage

```
defuzz(degrees,
       values,
       type=c('mom', 'fom', 'lom', 'dee'))
```

Arguments

degrees	A fuzzy set in the form of a numeric vector of membership degrees. Membership degrees must correspond to crisp values in the values argument.
values	Crisp values that correspond to memberhsip degrees in the degrees vector. Function assumes that the values are sorted in the ascending order.
type	Type of requested defuzzification method. The possibilities are: • 'mom'Mean of Maxima - maximum membership degrees are found and a mean of values that correspond to that degrees is returned; • 'fom'First of Maxima - first value with maximum membership degree is returned; • 'lom'Last of Maxima - last value with maximum membership degree is returned; • 'dee'Defuzzification of Evaluative Expressions - method used by the pbld inference mechanism that combines the former three approaches accordingly to the shape of the degrees vector: If degrees is non-increasing then 'lom' type is used, if it is non-decreasing then 'fom' is applied, else 'mom' is selected.

Details

Function converts input fuzzy set into a crisp value. The definition of input fuzzy set is provided by the arguments degrees and values. These arguments should be numeric vectors of the same length, the former containing memberhsip degrees in the interval $[0, 1]$ and the latter containing the corresponding crisp values; the fuzzy set is interpreted as values[i] to have the memberhsip degree degrees[i]. The values vector is assumed to be sorted in ascending order.

Value

A crisp value computed from values with respect to degrees and a type of defuzzification.

Author(s)

Michal Burda

See Also

fire, aggregateConsequents, perceive, pbld, fcut, lcut

Examples

```
defuzz(c(0, 0, 0, 0.1, 0.3, 0.9, 0.9, 0.9, 0.2, 0), 1:10, type='mom')
```

errors *Compute forecast errors*

Description

Compute Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE), and Root Mean Squared Error (RMSE) from forecasted and validation data.

Usage

```
smape(forecast, validation)
mase(forecast, validation)
rmse(forecast, validation)
```

Arguments

`forecast`	A numeric vector of predicted or forecasted values. Its length must be the same as the length of the `validation` argument.
`validation`	A numeric vector of actual (real) values being forecasted. Its length must be the same as the length of the `forecast` argument.

Details

The function compute various error measures of the forecasts. Let v_i, f_i be the i-th elements of `validation` or `forecast`, respectively, and n be the length of `validation`. Then:

- $SMAPE = 1/n \sum_{i=1}^{n} (2|f_i - v_i|)/(|f_i| + |v_i|)$
- $MASE = (\sum_{i=1}^{n} |v_i - f_i|)/(n/(n-1) * \sum_{i=2}^{n} |v_i - v_{i-1}|)$
- $RMSE = sqrt(1/n * \sum_{i=1}^{n} (v_i - f_i)^2)$

Value

A numeric value.

Author(s)

Michal Burda

See Also

`evalfrbe`, `frbe`

`evalfrbe` *Evaluate the performance of the FRBE forecast*

Description

Take a FRBE forecast and compare it with real values using arbitrary error function.

Usage

```
evalfrbe(fit,
         real,
         error=c('smape', 'mase', 'rmse'))
```

Arguments

`fit`	A FRBE model of class `frbe` as returned by the `frbe` function.
`real`	A numeric vector of real (known) values. The vector must correspond to the values being forecasted, i.e. the length must be the same as the horizon forecasted by `frbe`.
`error`	Error measure to be computed. It can be either Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE), or Root Mean Squared Error (RMSE). See `smape`, `mase`, and `rmse`, for more details.

Details

Take a FRBE forecast and compare it with real values by evaluating a given error measure. FRBE forecast should be made for a horizon of the same value as length of the vector of real values.

Value

Function returns a data.frame with single row and columns corresponding to the error of the individual forecasting methods that the FRBE is computed from. Additionally to this, a column "avg" is added with error of simple average of the individual forecasting methods and a column "frbe" with error of the FRBE forecasts.

Author(s)

Michal Burda

References

Štěpnička, M., Burda, M., Štěpničková, L. Fuzzy Rule Base Ensemble Generated from Data by Linguistic Associations Mining. FUZZY SET SYST. 2015.

See Also

`frbe`, `smape`, `mase`, `rmse`

Examples

```
# prepare data (from the forecast package)
library(forecast)
horizon <- 10
train <- wineind[-1 * (length(wineind)-horizon+1):length(wineind)]
test <- wineind[(length(wineind)-horizon+1):length(wineind)]
f <- frbe(ts(train, frequency=frequency(wineind)), h=horizon)
evalfrbe(f, test)
```

`farules` *A class of rules with statistical characteristics.*

Description

The aim of the `farules` S3 class is to store a list of rules (a rule-base) together with some statistical characteristics. To search for fuzzy association rules, refer to `searchrules` function.

Usage

```
farules(rules, statistics)
```

Arguments

`rules`	A list of character vectors, where each vector represents a rule and each value of the vector represents a predicate. The first value of the vector is assumed to be a rule's consequent, the rest is antecedent.
`statistics`	A numeric matrix of various statistical characteristics of the rules. Each column of that matrix corresponds to some statistic (such as support, confidence, etc.). Rows correspond to the rules in the list of `rules`.

Details

The `farules` function is a constructor for an instance of the `farules` class.

Value

Returns an object of class `farules`.

Author(s)

Michal Burda

See Also

`searchrules`, `reduce`

`fcut` *Transform data into a set of fuzzy attributes using triangular or raised cosine shapes of the fuzzy sets*

Description

This function creates a set of fuzzy attributes from crisp data. Factors, numeric vectors, matrix or data frame columns are transformed into a set of fuzzy attributes, i.e. columns with membership degrees. Unlike `lcut`, for transformation is not used the linguistic linguistic approach, but partitioning using regular shapes of the fuzzy sets (such as triangle, raised cosine).

Usage

```
fcut(x, ...)
## S3 method for class 'data.frame'
fcut(x,
     breaks,
     name=NULL,
     type=c('triangle', 'raisedcos'),
     merge=1,
     parallel=FALSE,
     ...)
## S3 method for class 'numeric'
fcut(x,
     breaks,
     name=deparse(substitute(x)),
     type=c('triangle', 'raisedcos'),
     merge=1,
     parallel=FALSE,
     ...)
```

Arguments

`x`	Data to be transformed: a vector, matrix, or data frame. Non-numeric data are allowed.
`breaks`	This argument determines the break-points of the positions of the fuzzy sets. It should be an ordered vector of numbers such that the i-th index specifies the beginning, $(i+1)$-th the center, and $(i+2)$-th the ending of the i-th fuzzy set. I.e. the minimum number of breaks-points is 3; $n-2$ elementary fuzzy sets would be created for n break-points. If considering an i-th fuzzy set (of `type='triangle'`), `x` values lower than i-th break (and greater than $(i+2)$-th break) would result in zero memberhsip degree, values equal to $(i+1)$-th break would have membership degree equal 1 and values between them the appropriate membership degree between 0 and 1. The resulting fuzzy sets would be named after the original data by adding dot (".") and a number i of fuzzy set.

Unlike cut, x values, that are lower or greater than the given break-points, will have all memberhsip degrees equal to zero.

For non-numeric data, this argument is ignored. For x being a numeric vector, it must be a vector of numeric values. For x being a numeric matrix or data frame, it must be a named list containing a numeric vector for each column - if not, the values are repeated for each column.

`name` A name to be added as a suffix to the created fuzzy attribute names. This parameter can be used only if x is a vector. If x is a matrix or data frame, name should be NULL because the fuzzy attribute names are taken from column names of the argument x.

`type` The type of fuzzy sets to create Currently, 'triangle' or 'raisedcos' may be used. The type argument may be also a function of 4 arguments that from the value of the first argument, and considering the boundaries given by the next 3 arguments, computes a membership degree. See e.g. triangle or raisedcos for details on how such function should look like.

`merge` This argument determines whether to derive additional fuzzy sets by merging the elementary fuzzy sets (whose position is determined with the breaks argument) into super-sets. The argument is ignored for non-numeric data in x.

merge may contain any integer number from 1 to length(breaks) - 2. Value 1 means that the elementary fuzzy sets should be present in the output. Value 2 means that the two consecutive elementary fuzzy sets should be combined by using the Lukasiewic t-conorm, value 3 causes combining three consecutive elementary fuzzy sets etc.

The names of the derived (merged) fuzzy sets is derived from the names of the original elementary fuzzy sets by concatenating them with the "|" (pipe) separator.

`parallel` Whether the processing should be run in parallel or not. Parallelization is implemented using the foreach package. The parallel environment must be set properly in advance, e.g. with the registerDoMC function. Currently this argument is applied only if x is a matrix or data frame.

`...` Other parameters to some methods.

Details

The aim of this function is to transform numeric data into a set of fuzzy attributes. The result is in the form of the object of class "fsets", i.e. a numeric matrix whose columns represent fuzzy sets (fuzzy attributes) with values being the membership degrees.

The function behaves diffently to the type of input x.

If x is a factor or a logical vector (or other non-numeric data) then for each distinct value of an input, a fuzzy set is created, and data would be transformed into crisp membership degrees 0 or 1 only.

If x is a numeric vector then fuzzy sets are created accordingly to break-points specified in the breaks argument with 1st, 2nd and 3rd break-point specifying the first fuzzy set, 2nd, 3rd and 4th break-point specifying th second fuzzy set etc. The shape of the fuzzy set is determined by the type argument that may be equal either to a string 'triangle' or 'raisedcos' or it could be a function that computes the membership degrees for itself (see triangle or raisedcos functions for details).

Additionally, super-sets of these elementary sets may be created by specifying the `merge` argument. Values of this argument specify how many consecutive fuzzy sets should be combined (by using the Lukasiewic's t-conorm) to produce super-sets - see the description of `merge` above.

If a matrix (resp. data frame) is provided to this function instead of single vector, all columns are processed separately as described above and the result is combined with the `cbind.fsets` function.

The function sets up properly the `vars` and `specs` properties of the result.

Value

An object of class "fsets" is returned, which is a numeric matrix with columns representing the fuzzy attributes. Each source columm of the `x` argument corresponds to multiple columns in the resulting matrix. Columns have names that indicate the name of the source as well as a index i of fuzzy set(s) – see the description of arguments `breaks` and `merge` above.

The resulting object would also have set the `vars` and `specs` properties with the former being created from original column names (if `x` is a matrix or data frame) or the `name` argument (if `x` is a numeric vector). The `specs` incidency matrix would be created to reflect the superset-hood of the merged fuzzy sets.

Author(s)

Michal Burda

See Also

`lcut`, `farules`, `pbld vars`, `specs`, `cbind.fsets`

Examples

```
# fcut on non-numeric data
ff <- factor(substring("statistics", 1:10, 1:10), levels = letters)
fcut(ff)

# transform a single vector into a single fuzzy set
x <- runif(10)
fcut(x, breaks=c(0, 0.5, 1), name='age')

# transform single vector into a partition of the interval 0-1
# (the boundary triangles are right-angled)
fcut(x, breaks=c(0, 0, 0.5, 1, 1), name='age')

# also create supersets
fcut(x, breaks=c(0, 0, 0.5, 1, 1), name='age', merge=c(1, 2))

# transform all columns of a data frame
# with different breakpoints
data <- CO2[, c('conc', 'uptake')]
fcut(data, breaks=list(conc=c(95, 95, 350, 1000, 1000),
                       uptake=c(7, 7, 28.3, 46, 46)))
```

`fire` *Compute truth-degrees of rules on data*

Description

Given data in the form of membership degrees to fuzzy sets, compute the truth value of given list of rules.

Usage

```
fire(x,
     rules,
     tnorm=c("goedel", "goguen", "lukasiewicz"),
     onlyAnte=TRUE,
     parallel=FALSE)
```

Arguments

`x`	Data for the rules to be evaluated on. Could be either a numeric matrix or numeric vector. If matrix is given then the rules are evaluated on rows. Each value of the vector or column of the matrix represents a predicate - it's numeric value represents the truth values (values in the interval [0, 1]).
`rules`	Either an object of class "farules" or list of character vectors where each vector is a rule with consequent being the first element of the vector. Elements of the vectors (predicate names) must correspond to the `x`'s names (of columns if `x` is a matrix).
`tnorm`	A character string representing a triangular norm to be used (either `"goedel"`, `"goguen"`, or `"lukasiewicz"`) or an arbitrary function that takes a vector of truth values and returns a t-norm computed of them.
`onlyAnte`	TRUE if only antecedent-part of a rule should be evaluated. Antecedent-part of a rule are all predicates in rule vector starting from the 2nd position. (First element of a rule is the consequent - see above.) If FALSE, then the whole rule will be evaluated (antecedent part together with consequent).
`parallel`	Whether the processing should be run in parallel or not. Parallelization is implemented using the `foreach` package. The parallel environment must be set properly in advance, e.g. with the `registerDoMC` function.

Details

The aim of this function is to compute the truth value of each rule in a list on given data. Each rule in the `rules` list is represented as a character vector of predicates with the first element being considered as a rule's consequent.

`x` is data either in a form of a numeric vector or numeric matrix. If vector is given then `names(x)` must correspond to the predicate names in `rules`. If `x` is a matrix then each column represents a predicate and thus `colnames(x)` must correspond to the predicate names in `rules`.

Values of either an input vector or matrix are interpreted as truth values. If matrix is given, the resulting truth values are computed row-wisely.

The type of conjunction to be used can be specified with the `tnorm` argument.

Value

If `x` is a vector then the result of this function is a list with a truth value of each rule. If `x` is a matrix, then a list of vectors of truth values is returned with truth values of the rules being computed row-wisely.

Author(s)

Michal Burda

See Also

`aggregateConsequents`, `defuzz` `perceive`, `pbld`, `fcut`, `lcut`, `farules`

Examples

```
# fire whole rules on a vector
x <- 1:10 / 10
names(x) <- letters[1:10]
rules <- list(c('a', 'c', 'e'),
              c('b'),
              c('d', 'a'),
              c('c', 'a', 'b'))
fire(x, rules, tnorm='goguen')

# fire antecedents of the rules on a matrix
x <- matrix(1:20 / 20, nrow=2)
colnames(x) <- letters[1:10]
rules <- list(c('a', 'c', 'e'),
              c('b'),
              c('d', 'a'),
              c('c', 'a', 'b'))
fire(x, rules, tnorm='goedel', onlyAnte=TRUE)

# the former command should be equal to
fire(x, antecedents(rules), tnorm='goedel')
```

---

frbe *Fuzzy Rule-Based Ensemble (FRBE) of time-series forecasts*

Description

This function computes the fuzzy rule-based ensemble of time-series forecasts. Several forecasting methods are used to predict future values of given time-series and a weighted sum is computed from them with weights being determined from a fuzzy rule base.

Usage

```
frbe(d,
     h=10)
```

Arguments

d	A source time-series in the ts time-series format. Note that the frequency of the time-series must to be set properly.
h	A forecasting horizon, i.e. the number of values to forecast.

Details

This function computes the fuzzy rule-based ensemble of time-series forecasts. The evaluation comprises of the following steps:

1. Several features are extracted from the given time-series d:
   - length of the time-series
   - strength of trend
   - strength of seasonality
   - skewness
   - kurtosis
   - variation coefficient
   - stationarity
   - frequency

   These features are used later to infer weights of the forecasting methods.
2. Several forecasting methods are applied on the given time-series d to obtain forecasts. Actually, the following methods are used:
   - ARIMA - by calling auto.arima of the forecast package
   - Exponential Smoothing - by calling ets of the forecast package
   - Random Walk with Drift - by calling rwf of the forecast package
   - Theta - by calling thetaf of the forecast package
3. Computed features are input to the fuzzy rule-based inference mechanism which yields into weights of the forecasting methods. The fuzzy rule base is hardwired in this package and it was obtained by performing data mining with the use of the farules function.
4. A weighted sum of forecasts is computed and returned as a result.

Value

Result is a list of class frbe with the following elements:

- features - a data frame with computed features of the given time-series;
- forecasts - a data frame with forecasts to be ensembled;
- weights - weights of the forecasting methods as inferred from the features and the hard-wired fuzzy rule base;
- mean - the resulting ensembled forecast (computed as a weighted sum of forecasts).

Author(s)

Michal Burda

References

Štěpnička, M., Burda, M., Štěpničková, L. Fuzzy Rule Base Ensemble Generated from Data by Linguistic Associations Mining. FUZZY SET SYST. 2015.

See Also

evalfrbe

Examples

```
# prepare data (from the forecast package)
library(forecast)
horizon <- 10
train <- wineind[-1 * (length(wineind)-horizon+1):length(wineind)]
test <- wineind[(length(wineind)-horizon+1):length(wineind)]

# perform FRBE
f <- frbe(ts(train, frequency=frequency(wineind)), h=horizon)

# evaluate FRBE forecasts
evalfrbe(f, test)

# display forecast results
f$mean
```

---

fsets *A class of a table with several fuzzy sets.*

Description

The aim of the `fsets` S3 class is to store several fuzzy sets in the form of numeric matrix where columns represent fuzzy sets and values are membership degrees. The `fsets` class also stores the information about the origin of the fuzzy sets as well as a relation of specificity among them.

Usage

```
fsets(x, vars, specs)

vars(x)

specs(x)
```

Arguments

x	A matrix of membership degrees. Columns of the matrix represent fuzzy sets, colnames are names of the fuzzy sets (and must not be NULL).
vars	A (typically character) vector that must correspond to the columns of x. It is a vector of names of original variables that the fuzzy sets were created from. In other words, the vars vector should contain the same value for each x's column that corresponds to the same variable. Moreover, the names of the vars vector must be the same as colnames(x). For instance, a function fcut can transform a single numeric vector into several different fuzzy sets. To indicate that all of them in fact describe the same original variable, the same name is stored on appropriate positions of the vars vector.
specs	A square numeric matrix containing values from $\{0, 1\}$. It is a specificity matrix, for which both rows and columns correspond to x's columns and where specs[i][j] = 1 if and only if the i-th fuzzy set (i.e. x[, i]) is more specific (i.e. is a subset of) than the j-th fuzzy set (i.e. x[, j]).

Details

The fsets function is a constructor for an instance of the fsets class. Their vars and specs arguments are stored into attributes of the objects. The functions vars and specs can be used to access that objects.

Value

fsets returns an object of class fsets.

vars returns a vector of original variable names of the fsets object (see the description of the vars argument above).

specs returns the specificity matrix of the fsets object (see the description of the specs argument above).

Author(s)

Michal Burda

See Also

fcut, lcut, is.specific

Examples

```
# create a matrix of random membership degrees
m <- matrix(runif(30), ncol=5)
colnames(m) <- c('a1', 'a2', 'a12', 'b1', 'b2')

# create vars - first three (a1, a2, a3) and next two (b1, b2)
# fuzzy sets originate from the same variable
v <- c('a', 'a', 'a', 'b', 'b')
names(v) <- colnames(m)
```

```
# create specificity matrix - a1 and a2 are subsets of a12,
# the rest is incomparable
s <- matrix(c(0, 0, 1, 0, 0,
              0, 0, 1, 0, 0,
              0, 0, 0, 0, 0,
              0, 0, 0, 0, 0,
              0, 0, 0, 0, 0), byrow=TRUE, ncol=5)
colnames(s) <- colnames(m)
rownames(s) <- colnames(m)

# create a valid instance of the fsets class
o <- fsets(m, v, s)
```

`head.farules` *Return the first part of an instance of the* `farules` *class*

Description

Returns the first part of an instance of the `farules` class.

Usage

```
## S3 method for class 'farules'
head(x, n=6L, ...)
```

Arguments

`x`	An instance of `farules` class
`n`	A single integer. If positive, return first n elements of `x`. If negative, return all but the n first number of elements of `x`.
`...`	Unused.

Details

Return a part of `x`.

Value

The instance of the `farules` class.

Author(s)

Michal Burda

See Also

`tail.farules`, `farules`

Examples

```
d <- lcut3(CO2[, 1:2])
print(head(d))
```

`head.fsets` *Return the first part of an instance of the* `fsets` *class*

Description

Returns the first part of an instance of the `fsets` class.

Usage

```
## S3 method for class 'fsets'
head(x, n=6L, ...)
```

Arguments

`x`	An instance of `fsets` class
`n`	A single integer. If positive, return first n rows of `x`. If negative, return all but the n first number of elements of `x`.
`...`	Unused.

Details

Return a part of `x`.

Value

The instance of the `fsets` class.

Author(s)

Michal Burda

See Also

`tail.fsets`, `fsets`, `fcut`, `lcut`

Examples

```
d <- lcut3(CO2[, 1:2])
print(head(d))
```

is.farules *Test whether* x *is a valid object of the* farules *class*

Description

Test whether x has a valid format for the objects of the farules class.

Usage

```
is.farules(x)
```

Arguments

x	An object being tested.

Details

This function tests wheter x inherits from farules and whether it is a list with x$rules being a list of rules and x$stats being a matrix.

Value

TRUE if x is a valid farules object and FALSE otherwise.

Author(s)

Michal Burda

See Also

farules

is.frbe *Test whether* x *is a valid object of the* frbe *class*

Description

Test whether x has a valid format for the objects of the frbe class.

Usage

```
is.frbe(x)
```

Arguments

x	An object being tested.

Details

This function tests wheter x inherits from frbe i.e. whether it is a list with the following elements: forecasts data frame, features data frame, weights vector, and mean vector.

Value

TRUE if x is a valid frbe object and FALSE otherwise.

Author(s)

Michal Burda

References

Štěpnička, M., Burda, M., Štěpničková, L. Fuzzy Rule Base Ensemble Generated from Data by Linguistic Associations Mining. FUZZY SET SYST. 2015.

See Also

frbe

---

is.fsets *Test whether* x *is a valid object of the* fsets *class*

Description

Test whether x has a valid format for the objects of the fsets class.

Usage

```
is.fsets(x)
```

Arguments

x	An object being tested.

Details

This function tests wheter x inherits from fsets and whether it is a numeric matrix with vars and specs attributes of correct size and column/row names.

Value

TRUE if x is a valid fsets object and FALSE otherwise.

Author(s)

Michal Burda

See Also

lcut, fcut

`is.specific` *Determine whether the first set of predicates is more specific (or equal) than the other.*

Description

Take two character vectors of predicates and determine whether `x` is more specific (or equal w.r.t. the specificity) than `y`. The specificity relation is fully determined with the values of `vars` vector and `specs` incidence matrix.

Usage

```
is.specific(x, y, vars, specs)
```

Arguments

`x`	The first character vector of predicates.
`y`	The second character vector of predicates.
`vars`	A named (typically character) vector that determines which predicates originate from the same variable, i.e. which of them semantically deal with the same property. For that purpose, each value of `x` or `y` must be present in `names(vars)`. See also `vars` function of the `fsets` class.
`specs`	A square numeric matrix containing values from $\{0, 1\}$. It is a specificity matrix for which rows and columns represent predicates. `specs[i][j] = 1` if and only if the i-th predicate is more specific (i.e. the corresponding fuzzy set is a subset of) than the j-th predicate (i.e. `x[, j]`). See also `specs` function of the `fsets` class.

Details

Let x_i, y_j represent any predicate of the vectors `x`, `y`. Function assumes that each vector `x` and `y` does not contain two or more predicates with the same value of `vars`.

This function returns TRUE iff all of the following conditions hold:

- for any y_j there exists x_i such that $vars[y_j] = vars[x_i]$;
- for any x_i there either does not exist y_j such that $vars[x_i] = vars[y_j]$ or $x_i = y_j$ or $specs[x_i, y_j] = 1$.

`x`

Value

TRUE or FALSE (see above).

Author(s)

Michal Burda

See Also

perceive, pbld, vars, specs

Examples

```
# create vars (v) and specs (s)
v <- c(rep('a', 3), rep('b', 3), rep('c', 3), rep('d', 3))
names(v) <- paste(rep(c('VeSm', 'Sm', 'Bi'), times=4),
                  rep(c('a', 'b', 'c', 'd'), each=3),
                  sep='.')

s <- matrix(c(0,1,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,

              0,0,0, 0,1,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,

              0,0,0, 0,0,0, 0,1,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,

              0,0,0, 0,0,0, 0,0,0, 0,1,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0),
            byrow=TRUE,
            ncol=12)
colnames(s) = names(v)
rownames(s) = names(v)

# returns TRUE
is.specific(c('VeSm.a', 'Bi.c'),
            c('VeSm.a', 'Bi.c'),
            v, s)

# returns TRUE (x and y swapped return FALSE)
is.specific(c('VeSm.a', 'Bi.c', 'Sm.d'),
            c('Sm.a', 'Bi.c', 'Sm.d'),
            v, s)

# returns TRUE (x and y swapped return FALSE)
is.specific(c('VeSm.a', 'Bi.c', 'Sm.d'),
            c('VeSm.a', 'Bi.c'),
            v, s)

# returns TRUE (x and y swapped return FALSE)
is.specific(c('VeSm.a', 'Bi.c', 'Sm.d'),
```

```
            NULL,
            v, s)

# returns FALSE
is.specific(c('Sm.a'), c('Bi.c'), v, s)

# returns FALSE
is.specific(c('VeSm.a', 'Sm.c'),
            c('Sm.a', 'Bi.c'),
            v, s)
```

---

lcut *Transform data into a set of linguistic fuzzy attributes*

Description

This function creates a set of linguistic fuzzy attributes from crisp data. Numeric vectors, matrix or data frame columns are transformed into a set of fuzzy attributes, i.e. columns with membership degrees. Factors and other data types are transformed to fuzzy attributes by calling the fcut function.

Usage

```
lcut3(x, ...)
## S3 method for class 'matrix'
lcut3(x, ...)
## S3 method for class 'data.frame'
lcut3(x,
     context=NULL,
     name=NULL,
     parallel=FALSE,
     ...)
## S3 method for class 'numeric'
lcut3(x,
     context=NULL,
     defaultCenter=0.5,
     atomic=c("sm", "me", "bi"),
     hedges=c("ex", "si", "ve", "ml", "ro", "qr", "vr"),
     name=NULL,
     parallel=FALSE,
     ...)

lcut5(x, ...)
## S3 method for class 'matrix'
lcut5(x, ...)
## S3 method for class 'data.frame'
lcut5(x,
     context=NULL,
```

```
     name=NULL,
     parallel=FALSE,
     ...)
## S3 method for class 'numeric'
lcut5(x,
     context=NULL,
     defaultCenter=0.5,
     atomic=c('sm', 'lm', 'me', 'um', 'bi'),
     hedges=c("ex", "ve", "ml", "ro", "ty"),
     name=NULL,
     parallel=FALSE,
     ...)
```

Arguments

x	Data to be transformed: if it is a numeric vector, matrix, or data frame, then the creation of linguistic fuzzy attributes takes place. For other data types the fcut function is called.
context	A definition of context of a numeric attribute. Context determines how people understand the notions "small", "medium", or "big" with respect to that attribute. If x is a numeric vector then context should be a vector of 3 numbers: typical small, medium, and big value. If the context is set to NULL, these values are taken directly from x as follows: • small= $min(x)$; • medium= $(max(x) - min(x)) * defaultCenter + min(x)$; • big= $max(x)$. If x is a matrix or data frame then context should be a named list of contexts for each x's column. If some context is omitted, it will be determined directly from data as explained above. Regardless of the value of the atomic argument, all 3 numbers of the context must be provided everytime.
defaultCenter	A value used to determine a typical "medium" value from data (see context above). If context is not specified then typical "medium" is determined as $$(max(x) - min(x)) * defaultCenter + min(x).$$ Default value of defaultCenter is 0.5, however, some literature specifies 0.42 as another sensible value with proper linguistic interpretation.
atomic	A vector of atomic linguistic expressions to be used for creation of fuzzy attributes. The possible values for lcut3 are: • smsmall; • memedium; • bibig. For lcut5, the following values are possible: • smsmall; • lmlower medium;

- memedium;
- umupper medium;
- bibig.

Several values are allowed in this argument.

`hedges` A vector of linguistic hedges to be used for creation of fuzzy attributes.

For `lcut3` variant, the following hedges are allowed:

- exextremely (sm, bi);
- sisignificantly (sm, bi);
- vevery (sm, bi);
- mlmore or less (sm, me, bi);
- roroughly (sm, me, bi);
- qrquite roughly (sm, me, bi);
- vrvery roughly (sm, me, bi).

For `lcut5` variant, the following hedges are allowed:

- exextremely (sm, bi);
- vevery (sm, bi);
- mlmore or less (sm, me, bi);
- roroughly (sm, me, bi);
- tytypically (me).

By default, a fuzzy attribute is created for each atomic expression (i.e. "small", "medium", "big") with empty hedge. Additionally, another fuzzy attributes are created based on the set of hedges selected with this argument. Not all hedges are usable to any atomic expression. In the list above, one can find the allowed atomic expressions in parentheses.

`name` A name to be added as a suffix to the created fuzzy attribute names. This parameter can be used only if `x` is a numeric vector. If `x` is a matrix or data frame, `name` should be NULL because the fuzzy attribute names are taken from column names of parameter `x`.

`parallel` Whether the processing should be run in parallel or not. Parallelization is implemented using the `foreach` package. The parallel environment must be set properly in advance, e.g. with the `registerDoMC` function.

`...` Other parameters to some methods.

Details

The aim of this function is to transform numeric data into a set of fuzzy attributes. The resulting fuzzy attributes have direct linguistic interpretation. This is a unique variant of fuzzification that is suitable for the inference mechanism based on Perception-based Linguistic Description (PbLD) – see `pbld`.

A numeric vector is transformed into a set of fuzzy attributes accordingly to the following scheme:

$< hedge >< atomicexpression >$

where $< atomicexpression >$ is a linguistic expression "small" ("sm"), "lower medium" ("lm"), "medium" ("me"), "upper medium" ("um") or "big" ("bi") – see the `atomic` argument. A $< hedge >$ is a modifier that further concretizes the atomic expression. It can be empty ("") or some value of:

- tytypically;
- exextremely;
- sisignificantly;
- vevery;
- mlmore or less;
- roroughly;
- qrquite roughly;
- vrvery roughly.

Accordingly to the theory developed by Novak (2008), not every hedge is suitable with each atomic expression (see the description of the `hedges` argument). The hedges to be used can be selected with the `hedges` argument. Function takes care of not to use hedge together with an un-applicable atomic expression by itself.

Obviously, distinct data have different meaning of what is "small", "medium", or "big". Therefore, a `context` has to be set that specifies sensible values for these linguistic expressions.

If a matrix (resp. data frame) is provided to this function instead of single vector, all columns are processed the same way.

The function also sets up properly the `vars` and `specs` properties of the result.

Value

An object of class "fsets" is returned, which is a numeric matrix with columns representing the fuzzy attributes. Each source columm of the `x` argument corresponds to multiple columns in the resulting matrix. Columns will have names derived from used $hedges$, atomic expression, and $name$ specified as the optional parameter.

The resulting object would also have set the `vars` and `specs` properties with the former being created from original column names (if `x` is a matrix or data frame) or the `name` argument (if `x` is a numeric vector). The `specs` incidency matrix would be created to reflect the following order of the hedges: "ex" $<$ "si" $<$ "ve" $<$ "" $<$ "ml" $<$ "ro" $<$ "qr" $<$ "vr" and "ty" $<$ "". Fuzzy attributes created from the same source numeric vector (or column) would be ordered that way, with other fuzzy attributes (from the other source) being incomparable.

Author(s)

Michal Burda

References

V. Novak, A comprehensive theory of trichotomous evaluative linguistic expressions, Fuzzy Sets and Systems 159 (22) (2008) 2939–2969.

See Also

`fcut`, `farules`, `pbld vars`, `specs`, `cbind.fsets`

Examples

```
# transform a single vector
x <- runif(10)
lcut3(x, name='age')
lcut5(x, name='age')


# transform single vector with custom context
lcut3(x, context=c(0, 0.2, 0.5), name='age')
lcut5(x, context=c(0, 0.2, 0.5), name='age')


# transform all columns of a data frame
# and do not use any hedges
data <- CO2[, c('conc', 'uptake')]
lcut3(data, hedges=NULL)
lcut5(data, hedges=NULL)


# definition of custom contexts for different columns
# of a data frame while selecting only "ve" and "ro" hedges.
lcut3(data,
     context=list(conc=c(0, 500, 1000),
                  uptake=c(0, 25, 50)),
     hedges=c('ve', 'ro'))


# lcut on non-numeric data is the same as fcut()
ff <- factor(substring("statistics", 1:10, 1:10), levels = letters)
lcut3(ff)
lcut5(ff)
```

`mult` *Callback-based Multiplication of Matrices*

Description

Perform a custom multiplication of the matrices `x` and `y` by using the callback function `f`.

Usage

```
mult(x, y, f, ...)
```

Arguments

`x`	A first matrix. The number of columns must match with the number of rows of the `y` matrix.
`y`	A second matrix. The number of rows must match with the number of columns of the `x` matrix.

f	A function to be applied to the matrices in order to compute the multiplication. It must accept at least two arguments.
...	Additional arguments that are passed to the function f.

Details

For a matrix x of size (u, v) and a matrix y of size (v, w), mult calls the function f uw-times to create a resulting matrix of size (u, w). Each (i, j)-th element of the resulting matrix is obtained from a call of the function f with x's i-th row and y's j-th column passed as its arguments.

Value

A matrix with v rows and w columns, where v is the number of rows of x and w is the number of columns of y.

Author(s)

Michal Burda

See Also

compose

Examples

```
x <- matrix(runif(24, -100, 100), ncol=6)
y <- matrix(runif(18, -100, 100), nrow=6)

mult(x, y, function(xx, yy) sum(xx * yy)) # the same as "x %*% y"
```

---

pbld *Perform a Perception-based Logical Deduction (PbLD) with given rule-base on given dataset*

Description

Take a set of rules (a rule-base) and perform a Perception-based Logical Deduction (PbLD) on each row of a given fsets object.

Usage

```
pbld(x,
     rules,
     partition,
     values,
     type=c('global', 'local'),
     parallel=FALSE)
```

Arguments

x	Input to the inference. It should be an object of class fsets (e.g. created by using the fcut or lcut functions). It is basically a matrix with columns representing fuzzy sets. Each row represents a single case of inference. Columns should be named after predicates in rules' antecedents.
rules	A rule-base (a.k.a. linguistic description) either in the form of the farules object or as a list of character vectors where each element is a fuzzy set name (a predicate) and thus each such vector forms a rule.
partition	A fsets object with columns that are consequents in rules. These membership degrees must correspond to values.
values	Crisp values that correspond to rows of memberhsip degrees in the partition matrix. Function assumes that the values are sorted in the ascending order.
type	The type of inference to use. It can be either "local" or "global" (default).
parallel	Whether the processing should be run in parallel or not. Parallelization is implemented using the foreach package. The parallel environment must be set properly in advance, e.g. with the registerDoMC function.

Details

Perform a Perception-based Logical Deduction (PbLD) with given rule-base rules on each row of input x. Columns of x are truth values of predicates that appear in the antecedent part of rules, partition together with values determine the shape of predicates in consequents: each element in values corresponds to a row of membership degrees in partition.

Value

A vector of inferred defuzzified values. The number of resulting values corresponds to the number of rows of the x argument.

Author(s)

Michal Burda

References

A. Dvořák, M. Štěpnička, On perception-based logical deduction and its variants, in: Proc. 16th World Congress of the International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijon, 2015.

See Also

lcut, searchrules, slices, fire, aggregateConsequents, defuzz

Examples

```
# --- TRAINING PART ---
# split data into training and testing set
testing <- CO2[1:5, ]
training <- CO2[-1 * 1:5, ]

# custom context of the RHS variable
uptakeContext <- c(7, 28.3, 46)

# convert training data into fuzzy sets
d <- lcut3(training, context=list(uptake=uptakeContext))

# search for rules
r <- searchrules(d, lhs=1:38, rhs=39:58)

# --- TESTING PART ---
# convert testing data info fuzzy sets
x <- lcut3(testing, context=list(uptake=uptakeContext))

# prepare values and partition
v <- slices(uptakeContext[1], uptakeContext[3], 1000)
p <- lcut3(v, name='uptake', context=uptakeContext)

# do the inference
pbld(x, r, p, v)
```

`perceive` *From a set of rules, remove each rule for which another rule exists that is more specific.*

Description

Examine rules in a list and remove all of them for whose other more specific rules are present in the list. The specificity is determined by calling the `is.specific` function. This operation is a part of the `pbld` inference mechanism.

Usage

```
perceive(rules,
         vars,
         specs,
         type=c('global', 'local'),
         fired=NULL)
```

Arguments

`rules`	A list of character vectors where each element is a fuzzy set name (a predicate) and thus each such vector forms a rule.

vars	A named (typically character) vector that determines which predicates originate from the same variable, i.e. which of them semantically deal with the same property. For that purpose, each value from any vector stored in the rules list must be present in names(vars). See also vars function of the fsets class.
specs	A square numeric matrix containing values from $\{0, 1\}$. It is a specificity matrix for which each row and column corresponds to an rules'es predicate specs[i][j] = 1 if and only if the i-th predicate is more specific (i.e. the corresponding fuzzy set is a subset of) than the j-th predicate (i.e. x[, j]). See also specs function of the fsets class.
type	The type of perception to use. It can be either "local" or "global" (default).
fired	If type=="global" then this argument can be NULL. If type is "local" then fired must be a numeric vector of values in the interval $[0, 1]$ indicating the truth values of all rules, i.e. the length of the vector must be equal to the number of rules in the rules argument.

Details

For each rule x in the rules list, it searches for another rule y such that is.specific(y, x) returns TRUE. If yes then x is removed from the list.

Value

A modified list of rules for which no other more specific rule exists. (Each rule is a vector.)

Author(s)

Michal Burda

See Also

is.specific, fsets, fcut, lcut

Examples

```
# prepare vars
v <- c(rep('a', 3),
       rep('b', 3),
       rep('c', 3),
       rep('d', 3))
names(v) <- paste(rep(c('VeSm', 'Sm', 'Bi'), times=4),
                  rep(c('a', 'b', 'c', 'd'), each=3),
                  sep='.')
print(v)

# prepare specs
s <- matrix(c(0,1,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,
              0,0,0, 0,0,0, 0,0,0, 0,0,0,

              0,0,0, 0,1,0, 0,0,0, 0,0,0,
```

```
                0,0,0, 0,0,0, 0,0,0, 0,0,0,
                0,0,0, 0,0,0, 0,0,0, 0,0,0,

                0,0,0, 0,0,0, 0,1,0, 0,0,0,
                0,0,0, 0,0,0, 0,0,0, 0,0,0,
                0,0,0, 0,0,0, 0,0,0, 0,0,0,

                0,0,0, 0,0,0, 0,0,0, 0,1,0,
                0,0,0, 0,0,0, 0,0,0, 0,0,0,
                0,0,0, 0,0,0, 0,0,0, 0,0,0),
            byrow=TRUE,
            ncol=12)
colnames(s) = names(v)
rownames(s) = names(v)
print(s)

# run perceive function: (Sm.a, Bi.c) has
# more specific rule (VeSm.a, Bi.c)
perceive(list(c('Sm.a', 'Bi.c'),
              c('VeSm.a', 'Bi.c'),
              c('Sm.b', 'Sm.d')),
         v, s)
```

`plot.fsets` *Plot a 'fsets' object*

Description

Plot the membership degrees stored in the instance of class `fsets` using the line diagram.

Usage

```
## S3 method for class 'fsets'
plot(x, ...)
```

Arguments

`x`	An instance of class `fsets`
`...`	Other arguments that are passed to the `ts.plot` function.

Details

This function plots the membership degrees stored in the instance of the `fsets` class. Internally, the membership degrees are transformed into a time-series object and viewed in a plot using the `ts.plot` function. This function is useful mainly to see the shape of fuzzy sets on regularly sampled inputs.

Value

Result of the `ts.plot` method.

Author(s)

Michal Burda

See Also

fsets, fcut, lcut, ts.plot

Examples

```
d <- lcut3(slices(0, 1, 1000), name='x')

# plot the resulting fuzzy sets
plot(d)

# Additional arguments are passed to the ts.plot method
# Here thick lines represent atomic linguistic expressions,
# i.e. ``small'', ``medium'', and ``big''.
plot(d,
     ylab='membership degree',
     xlab='values',
     gpars=list(lwd=c(5, rep(1, 7), 5, rep(1, 4), 5, rep(1, 7))))
```

---

print.farules *Print an instance of the farules class*

Description

Format an object of the farules class into human readable form and print it to the output.

Usage

```
## S3 method for class 'farules'
print(x, ...)
```

Arguments

x	An instance of farules class
...	Unused.

Details

Format an object of the farules class into human readable form and print it to the output.

Value

None.

Author(s)

Michal Burda

See Also

farules, searchrules, reduce, sel

---

print.frbe *Print an instance of the frbe class*

Description

Format an object of the frbe class into human readable form and print it to the output.

Usage

```
## S3 method for class 'frbe'
print(x, ...)
```

Arguments

x	An instance of frbe class
...	Unused.

Details

Format an object of the frbe class into human readable form and print it to the output.

Value

None.

Author(s)

Michal Burda

References

Štěpnička, M., Burda, M., Štěpničková, L. Fuzzy Rule Base Ensemble Generated from Data by Linguistic Associations Mining. FUZZY SET SYST. 2015.

See Also

frbe

Examples

```
# prepare data (from the forecast package)
library(forecast)
horizon <- 10
train <- wineind[-1 * (length(wineind)-horizon+1):length(wineind)]
test <- wineind[(length(wineind)-horizon+1):length(wineind)]
f <- frbe(ts(train, frequency=frequency(wineind)), h=horizon)
print(f)
print(test)
```

---

print.fsets *Print an instance of the fsets class*

Description

Format an object of the fsets class into human readable form and print it to the output.

Usage

```
## S3 method for class 'fsets'
print(x, ...)
```

Arguments

x	An instance of fsets class
...	Unused.

Details

Format an object of the fsets class into human readable form and print it to the output.

Value

None.

Author(s)

Michal Burda

See Also

fsets, fcut, lcut

Examples

```
d <- lcut3(CO2[, 1:2])
print(d)
```

rbcoverage *Compute rule base coverage of data*

Description

This function computes rule base coverage, i.e. a an average of maximum membership degree at which each row of data fires the rules in rule base.

Usage

```
rbcoverage(x,
           rules,
           tnorm=c("goedel", "goguen", "lukasiewicz"),
           onlyAnte=TRUE)
```

Arguments

x	Data for the rules to be evaluated on. Could be either a numeric matrix or numeric vector. If matrix is given then the rules are evaluated on rows. Each value of the vector or column of the matrix represents a predicate - it's numeric value represents the truth values (values in the interval [0, 1]).
rules	Either an object of class "farules" or list of character vectors where each vector is a rule with consequent being the first element of the vector. Elements of the vectors (predicate names) must correspond to the x's names (of columns if x is a matrix).
tnorm	A character string representing a triangular norm to be used (either "goedel", "goguen", or "lukasiewicz") or an arbitrary function that takes a vector of truth values and returns a t-norm computed of them.
onlyAnte	TRUE if only antecedent-part of a rule should be evaluated. Antecedent-part of a rule are all predicates in rule vector starting from the 2nd position. (First element of a rule is the consequent - see above.) If FALSE, then the whole rule will be evaluated (antecedent part together with consequent).

Details

Let f_{ij} be a truth value of i-th rule on j-th row of data x. Then $m_j = max(f_{.j})$ is a maximum truth value that is reached for the j-th data row with the rule base. Then the rule base coverage is a mean of that truth values, i.e. $rbcoverage = mean(m_.)$.

Value

A numeric value of the rule base coverage of given data.

Author(s)

Michal Burda

References

M. Burda, M. Štěpnička, Reduction of Fuzzy Rule Bases Driven by the Coverage of Training Data, in: Proc. 16th World Congress of the International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijon, 2015.

See Also

fire, reduce

Examples

```
x <- matrix(1:20 / 20, nrow=2)
colnames(x) <- letters[1:10]

rules <- list(c('a', 'c', 'e'),
              c('b'),
              c('d', 'a'),
              c('c', 'a', 'b'))
rbcoverage(x, rules, "goguen", TRUE)  # returns 1


rules <- list(c('d', 'a'),
              c('c', 'a', 'b'))
rbcoverage(x, rules, "goguen", TRUE)  # returns 0.075)
```

---

reduce *Reduce the size of rule base*

Description

From given rule base, select such set of rules that influence mostly the rule base coverage of the input data.

Usage

```
reduce(x, rules, ratio,
       tnorm = c("goedel", "goguen", "lukasiewicz"),
       tconorm = c("goedel", "goguen", "lukasiewicz"),
       numThreads = 1)
```

Arguments

x	Data for the rules to be evaluated on. Could be either a numeric matrix or numeric vector. If matrix is given then the rules are evaluated on rows. Each value of the vector or column of the matrix represents a predicate - it's numeric value represents the truth values (values in the interval [0, 1]).

`rules`	Either an object of class "farules" or list of character vectors where each vector is a rule with consequent being the first element of the vector. Elements of the vectors (predicate names) must correspond to the `x`'s names (of columns if `x` is a matrix).
`ratio`	A percentage of rule base coverage that must be preserved. It must be a value within the $[0, 1]$ interval. Value of 1 means that the rule base coverage of the result must be the same as coverage of input `rules`. A sensible value is e.g. 0.9.
`tnorm`	Which t-norm to use as a conjunction of antecedents. The default is `"goedel"`.
`tconorm`	Which t-norm to use as a disjunction, i.e. to combine multiple antecedents to get coverage of the rule base. The default is `"goedel"`.
`numThreads`	How many threads to use for computation. Value higher than 1 causes that the algorithm runs in several parallel threads (using the OpenMP library).

Details

From a given rulebase, a rule with greatest coverage is selected. After that, additional rules are selected that increase the rule base coverage the most. Addition stops after the coverage exceeds $originalcoverage * ratio$.

Note that the size of the resulting rule base is not necessarily minimal because the algorithm does not search all possible combination of rules. It only finds a local minimum of rule base size.

Value

Function returns an instance of class `farules` or a list depending on the type of the `rules` argument.

Author(s)

Michal Burda

References

M. Burda, M. Štěpnička, Reduction of Fuzzy Rule Bases Driven by the Coverage of Training Data, in: Proc. 16th World Congress of the International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijon, 2015.

See Also

`rbcoverage`, `farules`

`searchrules` *Searching for fuzzy association rules*

Description

This function searches the given `fsets` object `d` for all fuzzy association rules that satisfy defined constraints. It returns a list of fuzzy association rules together with some statistics characterizing them (such as support, confidence etc.).

Usage

```
searchrules(d,
            lhs=2:ncol(d),
            rhs=1,
            tnorm=c("goedel", "goguen", "lukasiewicz"),
            n=100,
            best=c("confidence"),
            minSupport=0.02,
            minConfidence=0.75,
            maxConfidence=1,
            maxLength=4,
            numThreads=1,
            trie=(maxConfidence < 1))
```

Arguments

`d`	An object of class `fsets` - it is basically a matrix where columns represent the fuzzy sets and values are the membership degrees. For creation of such object, use `fcut` or `lcut` function.
`lhs`	Indices of fuzzy attributes that may appear on the left-hand-side (LHS) of association rules, i.e. in the antecedent.
`rhs`	Indices of fuzzy attributes that may appear on the right-hand-side (RHS) of association rules, i.e. in the consequent.
`tnorm`	A t-norm to be used for computation of conjunction of fuzzy attributes. (Allowed are even only starting letters of "lukasiewicz", "goedel" and "goguen").
`n`	The non-negative number of rules to be found. If zero, the function returns all rules satisfying the given conditions. If positive, only `n` best rules are returned. The criterium of what is "best" is specified with the `best` argument.
`best`	Specifies measure accordingly to which the rules are ordered from best to worst. This argument is used mainly in combination with the `n` argument. Currently, only single value ("confidence") can be used.
`minSupport`	The minimum support degree of a rule. Rules with support below that number are filtered out. It must be a numeric value from interval [0, 1]. See below for details on how the support degree is computed.

`minConfidence` The minimum confidence degree of a rule. Rules with confidence below that number are filtered out. It must be a numeric value from interval [0, 1]. See below for details on how the confidence degree is computed.

`maxConfidence` Maximum confidence threshold. After finding a rule that has confidence degree above the `maxConfidence` threshold, no other rule is resulted based on adding some additional attribute to its antecedent part. I.e. if "Sm.age & Me.age => Sm.height" has confidence above `maxConfidence` threshold, no another rule containing "Sm.age & Me.age" will be produced regardless of its interest measures.

If you want to disable this feature, set `maxConfidence` to 1.

`maxLength` Maximum allowed length of the antecedent, i.e. maximum number of predicates that are allowed on the left-hand side of the rule. If negative, the maximum length of rules is unlimited.

`numThreads` Number of threads used to perform the algorithm in parallel. If greater than 1, the OpenMP library (not to be confused with Open MPI) is used for parallelization. Please note that there are known problems of using OpenMP together with another means of parallelization that may be used within R. Therefore, if you plan to use the `searchrules` function with some of the external parallelization mechanisms such as library `doMC`, make sure that `numThreads` equals 1. This feature is available only on systems that have installed the OpenMP library.

`trie` Whether or not to use internal mechanism of Tries. If FALSE, then in the output may appear such rule that is a descendant of a rule that has confidence above `maxConfidence` threshold.

Tries consume very much memory, so if you encounter problems with insufficient memory, set this argument to FALSE. On the other hand, the size of result (if `n` is set to 0) can be very high if trie is set to FALSE.

Details

The function searches data frame `d` for fuzzy association rules that satisfy conditions specified by the parameters.

Value

A list of the following elements: `rules` and `statistics`.

`rules` is a list of mined fuzzy association rules. Each element of that list is a character vector with consequent attribute being on the first position.

`statistics` is a data frame of statistical characteristics about mined rules. Each row corresponds to a rule in the `rules` list. Let us consider a rule "a & b => c", let $\otimes$ be a t-norm specified with the `tnorm` parameter and i goes over all rows of a data table `d`. Then columns of the `statistics` data frame are as follows:

- supporta rule's support degree: $1/nrow(d) * \sum_{\forall i} a(i) \otimes b(i) \otimes c(i)$
- lhsSupporta support of rule's antecedent (LHS): $1/nrow(d) * \sum_{\forall i} a(i) \otimes b(i)$
- rhsSupporta support of rule's consequent (RHS): $1/nrow(d) * \sum_{\forall i} c(i)$
- confidencea rule's confidence degree: $support/lhsSupport$

Author(s)

Michal Burda

See Also

fcut, lcut, farules, fsets, pbld

Examples

```
d <- lcut3(CO2)
searchrules(d, lhs=1:ncol(d), rhs=1:ncol(d))
```

---

sel *Select several rows and columns from a data object*

Description

This function acts similarly as the [] operator, i.e. selects rows and columns of a data object.

Usage

```
sel(x, ...)
## S3 method for class 'farules'
sel(x,
                      i=rep(TRUE,
                      nrow(x)),
                      ...)
## S3 method for class 'fsets'
sel(x,
                    i=rep(TRUE, nrow(x)),
                    j=rep(TRUE, ncol(x)),
                    ...)
```

Arguments

x	Original data: originally an instance of class fsets or farules.
i	The specification of rows to be selected. It can be a vector of integer indices which have to be selected, or vector of negative integers specifying which rows to discard, or vector of logical values etc. See [for details.
j	The specification of columns to be selected (only applicable for instances of fsets). It can be a vector of integer indices which have to be selected, or vector of negative integers specifying which columns to discard, or vector of logical values etc. See [for details.
...	Other parameters to some methods.

Details

Originally, it is an S3 method defined for `fsets` and `farules` classes.

For instances of the `fsets` class, `sel` behaves similarly as the `[]` operator except that it also takes care of the `vars` and `specs` attributes. Therefore, if you want to select some rows or columns from the `fsets` object and obtain again a valid `fsets` object, use `sel`. If you want only a matrix or vector of membership degrees in the `fsets` object without any other attributes, use the `[]` operator.

For instances of the `farules` class, `sel` returns a valid `farules` object with a subset of rules specified with the `i` argument. Also the matrix of statistics is handled appropriately.

Value

A subset of data (originally `fsets` or `farules`) of the same type as original data, with attributes and additional meta data handled correctly.

Author(s)

Michal Burda

See Also

`fsets`, `fcut`, `lcut`, `cbind.fsets`, `farules`, `searchrules`

Examples

```
x <- 0:100
d <- fcut(x,
          breaks=c(0, 25, 50, 75, 100),
          type='triangle')

# select only first two columns
res <- sel(d, , 1:2)
print(res)

# select only the 3rd row
res <- sel(d, 3)
print(res)
```

---

slices *Return vector of values from given interval*

Description

Returns an ordered vector of values from given interval, of given size, generated by equal steps.

Usage

```
slices(from, to, n)
```

Arguments

from	The lower bound of the interval.
to	The upper bound of the interval.
n	The length of the vector to be produced.

Details

Returns a vector of values from from to to (inclusive), with equal difference between two consecutive values, with total length n. Function is useful e.g. together with the pbld or defuzz functions (for the values argument; see also lcut or fcut) or defuzz).

Value

A vector of numbers in the given interval and size.

Author(s)

Michal Burda

See Also

pbld, defuzz, fcut, lcut

Examples

```
slices(1, 5, 10) # 1, 1.5, 2, 2.5, 3, 3.5 4, 4.5, 5
```

---

tail.farules *Return the last part of an instance of the farules class*

Description

Returns the last part of an instance of the farules class.

Usage

```
## S3 method for class 'farules'
tail(x, n=6L, ...)
```

Arguments

x	An instance of farules class
n	A single integer. If positive, return last n elemens of x. If negative, return all but the n last number of elements of x.
...	Unused.

Details

Return a part of x.

Value

The instance of the farules class.

Author(s)

Michal Burda

See Also

head.farules, farules

---

tail.fsets *Return the last part of an instance of the fsets class*

Description

Returns the last part of an instance of the fsets class.

Usage

```
## S3 method for class 'fsets'
tail(x, n=6L, ...)
```

Arguments

x	An instance of fsets class
n	A single integer. If positive, return last n rows of x. If negative, return all but the n last number of elements of x.
...	Unused.

Details

Return a part of x.

Value

The instance of the fsets class.

Author(s)

Michal Burda

See Also

head.fsets, fsets, fcut, lcut

Examples

```
d <- lcut3(CO2[, 1:2])
print(tail(d))
```

---

triangle *Compute membership degrees of values to the fuzzy set*

Description

This function computes membership degrees of values to a fuzzy set that is defined on three numbers lo, center, and hi that form a triangle/raisedcos.

Usage

```
triangle(x, lo, center, hi)
raisedcos(x, lo, center, hi)
```

Arguments

x	Numeric vector of data to be converted.
lo	A lower bound (can be -Inf).
center	A peak value.
hi	An upper bound (can be Inf).

Details

This function computes membership degrees of values to a fuzzy set that is defined on three numbers lo, center, and hi that form a triangle/raisedcos, i.e. value equal to center has a membership degree equal to 1, values lower than lo or greater than hi have memberhsip degree equal to 0. Between lo and hi, there is a transition of the triangular (resp. raised cosinal) shape (with peak at center).

If lo == -Inf then any value that is lower or equal to center gets memberhsip degree 1. Similarly, if hi == Inf then any value that is greater or equal to center gets memberhsip degree 1.

triangle produces fuzzy sets of a triangular shape (with peak at center), raisedcos produces fuzzy sets defined as a raised cosine hill.

Value

A numeric vector of membership degrees.

Author(s)

Michal Burda

See Also

fcut

Examples

```
plot(function(x) triangle(x, -1, 0, 1), from=-2, to=3)
plot(function(x) triangle(x, -1, 0, 2), from=-2, to=3)
plot(function(x) triangle(x, -Inf, 0, 1), from=-2, to=3)
plot(function(x) triangle(x, -1, 0, Inf), from=-2, to=3)

plot(function(x) raisedcos(x, -1, 0, 1), from=-2, to=3)
plot(function(x) raisedcos(x, -1, 0, 2), from=-2, to=3)
plot(function(x) raisedcos(x, -Inf, 0, 1), from=-2, to=3)
plot(function(x) raisedcos(x, -1, 0, Inf), from=-2, to=3)
```

Index

∗Topic **models**
aggregateConsequents, 4
algebra, 6
antecedents, 9
as.data.frame.farules, 10
as.matrix.fsets, 11
cbind.fsets, 12
compose, 13
consequents, 15
defuzz, 15
errors, 17
evalfrbe, 18
farules, 19
fcut, 20
fire, 23
frbe, 24
fsets, 26
head.farules, 28
head.fsets, 29
is.farules, 30
is.frbe, 30
is.fsets, 31
is.specific, 32
lcut, 34
mult, 38
pbld, 39
perceive, 41
plot.fsets, 43
print.farules, 44
print.frbe, 45
print.fsets, 46
rbcoverage, 47
reduce, 48
searchrules, 50
sel, 52
slices, 53
tail.farules, 54
tail.fsets, 55
triangle, 56
∗Topic **multivariate**
as.data.frame.farules, 10
as.matrix.fsets, 11
compose, 13
errors, 17
fcut, 20
fire, 23
lcut, 34
mult, 38
plot.fsets, 43
rbcoverage, 47
reduce, 48
searchrules, 50
sel, 52
triangle, 56
∗Topic **package**
lfl-package, 2
∗Topic **robust**
aggregateConsequents, 4
algebra, 6
antecedents, 9
as.data.frame.farules, 10
as.matrix.fsets, 11
cbind.fsets, 12
compose, 13
consequents, 15
defuzz, 15
errors, 17
evalfrbe, 18
farules, 19
fcut, 20
fire, 23
frbe, 24
fsets, 26
head.farules, 28
head.fsets, 29
is.farules, 30
is.frbe, 30
is.fsets, 31

is.specific, 32
lcut, 34
mult, 38
pbld, 39
perceive, 41
plot.fsets, 43
print.farules, 44
print.frbe, 45
print.fsets, 46
rbcoverage, 47
reduce, 48
searchrules, 50
sel, 52
slices, 53
tail.farules, 54
tail.fsets, 55
triangle, 56
[, *52*

aggregateConsequents, *3*, 4, *16*, *24*, *40*
algebra, 6, *14*
antecedents, 9
as.data.frame.farules, 10
as.matrix.fsets, 11

cbind, *12*
cbind.fsets, 12, *22*, *37*, *53*
compose, 13, *39*
consequents, 15
cut, *21*

data.frame, *10*
defuzz, *3*, *5*, 15, *24*, *40*, *54*

errors, 17
evalfrbe, *3*, *17*, 18, *26*

farules, *10–12*, *15*, 19, *22*, *24*, *25*, *28*, *30*, *37*, *40*, *44*, *45*, *49*, *52–55*
fcut, *3*, *5*, *11*, *12*, *16*, 20, *24*, *27*, *29*, *32*, *34*, *35*, *37*, *40*, *42*, *44*, *46*, *50*, *52–54*, *56*, *57*
fire, *3*, *5*, *16*, 23, *40*, *48*
foreach, *21*, *23*, *36*, *40*
frbe, *3*, *17*, *18*, 24, *31*, *45*
fsets, *11*, 26, *29*, *32*, *39*, *40*, *42–44*, *46*, *50*, *52*, *53*, *55*, *56*

goedel.biresiduum *(*algebra*)*, 6
goedel.residuum *(*algebra*)*, 6
goedel.tconorm *(*algebra*)*, 6
goedel.tnorm *(*algebra*)*, 6
goguen.biresiduum *(*algebra*)*, 6
goguen.residuum *(*algebra*)*, 6
goguen.tconorm *(*algebra*)*, 6
goguen.tnorm *(*algebra*)*, 6

head.farules, 28, *55*
head.fsets, 29, *56*

invol.neg *(*algebra*)*, 6
is.algebra *(*algebra*)*, 6
is.farules, 30
is.frbe, 30
is.fsets, 31
is.specific, *27*, 32, *41*, *42*

lcut, *3*, *5*, *11*, *12*, *16*, *20*, *22*, *24*, *27*, *29*, *32*, 34, *40*, *42*, *44*, *46*, *50*, *52–54*, *56*
lcut3 *(*lcut*)*, 34
lcut5 *(*lcut*)*, 34
lfl *(*lfl-package*)*, 2
lfl-package, 2
lukas.biresiduum *(*algebra*)*, 6
lukas.residuum *(*algebra*)*, 6
lukas.tconorm *(*algebra*)*, 6
lukas.tnorm *(*algebra*)*, 6

mase, *18*
mase *(*errors*)*, 17
mult, *14*, 38

pbld, *3*, *16*, *22*, *24*, *33*, *36*, *37*, 39, *41*, *52*, *54*
perceive, *3*, *5*, *16*, *24*, *33*, 41
pgoedel.tconorm *(*algebra*)*, 6
pgoedel.tnorm *(*algebra*)*, 6
pgoguen.tconorm *(*algebra*)*, 6
pgoguen.tnorm *(*algebra*)*, 6
plot.fsets, 43
plukas.tconorm *(*algebra*)*, 6
plukas.tnorm *(*algebra*)*, 6
print.farules, 44
print.frbe, 45
print.fsets, 46

raisedcos, *21*
raisedcos *(*triangle*)*, 56
rbcoverage, 47, *49*
reduce, *3*, *19*, *45*, *48*, 48
registerDoMC, *21*, *23*, *36*, *40*

rmse, *18*
rmse *(*errors*)*, 17

searchrules, *3*, *19*, *40*, *45*, 50, *53*
sel, *45*, 52
slices, *40*, 53
smape, *18*
smape *(*errors*)*, 17
specs, *11, 12*, *22*, *31–33*, *37*, *42*, *53*
specs *(*fsets*)*, 26
strict.neg *(*algebra*)*, 6

tail.farules, *28*, 54
tail.fsets, *29*, 55
triangle, *21*, 56
ts.plot, *43, 44*

vars, *11, 12*, *22*, *31–33*, *37*, *42*, *53*
vars *(*fsets*)*, 26