Package 'stagedtrees'

March 30, 2020

Type Package

Title Staged Event Trees

Version 1.0.2

Description Creates and fits staged event tree probability models.
Staged event trees are probabilistic graphical models capable of
representing asymmetric conditional independence statements
among categorical variables.
This package contains functions to create, plot and fit
staged event trees from data, moreover different structure learning
algorithms are available.
References:
Collazo R. A., Görgen C. and Smith J. Q.
(2018, ISBN:9781498729604).
Görgen C., Bigatti A., Riccomagno E. and Smith J. Q. (2018)
<arXiv:1705.09457>.
Thwaites P. A., Smith, J. Q. (2017) <arXiv:1510.00186>.
Barclay L. M., Hutton J. L. and Smith J. Q. (2013)
<doi:10.1016/j.ijar.2013.05.006>.
Smith J. Q. and Anderson P. E. (2008)
<doi:10.1016/j.artint.2007.05.004>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

URL https://github.com/gherardovarando/stagedtrees

BugReports https://github.com/gherardovarando/stagedtrees/issues

Imports stats, graphics, methods

Suggests testthat, bnlearn, covr, clue

NeedsCompilation no

Author Gherardo Varando [aut, cre] (<https://orcid.org/0000-0002-6708-1103>),
Federico Carli [aut],
Manuele Leonelli [aut] (<https://orcid.org/0000-0002-2562-5192>),
Eva Riccomagno [aut]

Maintainer Gherardo Varando <gherardo.varando@gmail.com>

Repository CRAN

Date/Publication 2020-03-30 18:20:02 UTC

R topics documented:

`barplot_stages` *Barplot method for staged event trees*

Description

Barplot method for staged event trees

Usage

```
barplot_stages(object, var, legend.text = FALSE, col = NULL, ...)
```

Arguments

`object`	a staged tree object
`var`	name of varaibles in the model
`legend.text`	logical
`col`	color mapping for the stages, see `col` argument in `plot.sevt`
`...`	addittional arguments passed to `barplot`

`bhc.sevt` *Backward hill-climbing*

Description

Backward Hill-climbing search of staged event trees with iterative joining of stages

Usage

```
bhc.sevt(
  object,
  score = function(x) {     return(-BIC(x)) },
  max_iter = Inf,
  trace = 0
)
```

Arguments

object	a staged event tree model
score	the score function to be maximized
max_iter	the maximum number of iterations per variable
trace	if >0 increasingly amount of info is printed (via message)

Details

For each variable the algorithm try to join stages and move to the best model that increase the score. When no increase is possible it moves to the next variable.

Value

The final staged event tree obtained.

Examples

```
DD <- generate_xor_dataset(n = 4, N = 100)
model <- bhc.sevt(full(DD), trace = 2)
summary(model)
```

---

bhcr.sevt *Backward Random Hill-Climbing*

Description

Randomly try to join stages

Usage

```
bhcr.sevt(
  object,
  score = function(x) {     return(-BIC(x)) },
  max_iter = 100,
  trace = 0
)
```

Arguments

object	a staged event tree model
score	the score function to be maximized
max_iter	the maximum number of iteration
trace	if >0 increasingly amount of info is printed (via message)

Details

At each iteration a variable and two of its stages are randomly selected. If joining the stages increase the score, the model is updated. The procedure is repeated until the number of iterations reach `max_iter`.

Value

The final staged event tree object

Examples

```
DD <- generate_xor_dataset(n = 4, N = 100)
model <- bhcr.sevt(full(DD), trace = 2)
summary(model)
```

---

bj.sevt *Backward joining of stages*

Description

Join stages from more complex to simpler models using a distance and a threshold value

Usage

```
bj.sevt(object = NULL, distance = kl, thr = 0.1, trace = 0, ...)
```

Arguments

`object`	the staged event tree from where to start
`distance`	the distance between probabilities to use
`thr`	the threshold for joining stages
`trace`	if >0 increasingly amount of info
`...`	additional parameters to be passed to the distance function is printed (via `message`)

Details

For each variable in the model stages are joined iteratively. At each iteration the two stages with minimum distance are selected and joined if their distance is less than `thr`.

Value

The final staged event tree obtained.

Examples

```
DD <- generate_xor_dataset(n = 5, N = 1000)
model <- bj.sevt(full(DD, lambda = 1), trace = 2)
summary(model)
```

ceg.sevt *Creates the CEG starting from a staged tree*

Description

Creates the CEG starting from a staged tree

Usage

```
ceg.sevt(object)
```

Arguments

object	a staged event tree

Details

A ceg object is a staged event tree object with additional information on the positions.

Value

a ceg object

Examples

```
DD <- generate_xor_dataset(3, 100)
model <- bhc.sevt(full(DD))
ceg <- ceg.sevt(model)
ceg$positions
```

ceg2adjmat *Ceg to adjmat of graph*

Description

Ceg to adjmat of graph

Usage

```
ceg2adjmat(x)
```

Arguments

x	the ceg object

Details

it is used to prepare the plot of the ceg. It transforms the ceg object into an adjacent matrix.

Value

the adj matrix

Examples

```
DD <- generate_xor_dataset(3, 100)
model <- bhc.sevt(full(DD))
ceg <- ceg.sevt(model)
ceg2adjmat(ceg)
```

---

compare.sevt *Compare two staged event tree*

Description

Compare two stages event tree, return the differences of the stages structure and plot the difference tree. Three different methods to compute the difference tree are available.

Usage

```
compare.sevt(
  object1,
  object2,
  method = "naive",
  return.tree = FALSE,
  plot = FALSE,
  ...
)

hamming.sevt(object1, object2, return.tree = FALSE)

stagesdiff.sevt(object1, object2)
```

Arguments

object1	a staged event tree
object2	a staged event tree
method	method to compare staged event trees. It can be: "naive", "hamming" or "stages"
return.tree	logical, if TRUE the difference tree is returned
plot	logical
...	additional parameters to be passed to plot.sevt

Details

compare.sevt tests if the stage structure of two sevt objects is the same. Three methods are available:

- naive first applies stndnaming.sevt to both objects and then simply compares the resulting stages lists (stages.sevt(object1) and stages.sevt(object2)).
- hamming uses the hamming.sevt function that try to map stages in the different objects finding the few number of nodes that must be changed to obtain the same structure.
- stages uses the stagesdiff.sevt function that compare stages to check if the same stage structure is present in both models.

Setting return.tree = TRUE will return the stages structure difference obtained with the selected method.

With plot = TRUE the plot of the difference tree is obtained.

If return.tree = FALSE the logical output is the same for the three methods and thus the naive method should be used since it is computationally faster.

To use the hamming method, the package clue must be installed.

Functions hamming.sevt and stagesdiff.sevt can also be used directly.

Value

if return.tree = FALSE, logical: TRUE if the two models are exactly equal, otherwise FALSE. Else If return.tree = TRUE it returns the differences between the two trees, according to the selected method.

Examples

```
data("PhDArticles")
mod1 <- bhc.sevt(full(PhDArticles[, 1:4], lambda = 1))
mod2 <- fbhc.sevt(full(PhDArticles[, 1:4], lambda = 1))
compare.sevt(mod1, mod2)

##########
m0 <- full(PhDArticles[, 1:4], fit = TRUE, lambda = 0)
m1 <- bhc.sevt(m0)
m2 <- bj.sevt(m0, distance = tv, thr = 0.25)
stagesdiff.sevt(m1, m2)
```

---

df.sevt *Number of parameters of a staged event tree*

Description

Return the number of parameters of the model.

Usage

```
df.sevt(x)
```

Arguments

x	a staged event tree object

Value

integer, degrees of freedom of the staged event tree

Examples

```
#########
mod_f <- full(PhDArticles)
df.sevt(mod_f)

mod_i <- indep(PhDArticles)
df.sevt(mod_i)
```

---

fbhc.sevt *Fast backward hill-climbing*

Description

Fast backward hill-climbing search of staged event trees with iterative joining of stages.

Usage

```
fbhc.sevt(
  object = NULL,
  score = function(x) {     return(-BIC(x)) },
  max_iter = Inf,
  trace = 0
)
```

Arguments

object	a staged event tree model
score	the score function to be maximized
max_iter	the maximum number of iteration
trace	if >0 increasingly amount of info is printed (via message)

Details

For each variable the algorithm try to join stages and move to the first model that increase the score. When no increase is possible it moves to the next variable.

Value

The final staged event tree obtained.

Examples

```
DD <- generate_xor_dataset(n = 5, N = 100)
model <- fbhc.sevt(full(DD), trace = 2)
summary(model)
```

`fit.strt_ev_tree` *Fit a stratified event tree*

Description

Fit a stratified event tree

Usage

```
fit.strt_ev_tree(evt, data = NULL, lambda = 0)
```

Arguments

`evt`	The stratified event tree object to be fitted
`data`	the data.frame used to fit the event tree
`lambda`	the laplace smoothing

Details

Distribute the counts along the event tree

Value

A stratified event tree object

generate_linear_dataset

generate a random binary dataset for classification

Description

Randomly generate a simple classification problem

Usage

```
generate_linear_dataset(
  n = 2,
  N = 10000,
  eps = 1.2,
  gamma = runif(1, min = -n, max = n),
  alpha = runif(n, min = -n, max = n)
)
```

Arguments

n	number of variables
N	number of observations
eps	noise
gamma	numeric
alpha	numeric vector of length n

Value

A data.frame with n independent random variables and one class variable C computed as `sign(sum(x * alpha) + runif(1,-eps,eps) + gamma)`

Examples

```
DD <- generate_linear_dataset(n = 5, 1000)
```

generate_random_dataset

generate a random binary dataset

Description

Randomly generate a data.frame of independent binary variables

Usage

```
generate_random_dataset(n = 2, N = 10000)
```

Arguments

n	number of variables
N	number of observations

Value

A data.frame with n independent random variables

Examples

```
DD <- generate_random_dataset(n = 5, 1000)
```

---

generate_xor_dataset *generate a xor dataset*

Description

generate a xor dataset

Usage

```
generate_xor_dataset(n = 2, N = 100, eps = 1.2)
```

Arguments

n	number of variables
N	number of observations
eps	error

Value

The xor dataset with n + 1 variables, where the last one is the class variable C computed as xor logical operator.

Examples

```
DD <- generate_xor_dataset(n = 5, N = 1000, eps = 1.2)
```

getstagepath *Retrieve stage or path*

Description

Utility functions to obtain stages from paths and paths from stages.

Usage

```
get_stage(object, path)

get_path(object, var, stage)
```

Arguments

object	a staged event tree object
path	a vector contating the path from root or a two dimensional array where each row is a path from root
var	string, one of the variable in the staged tree
stage	string or vector, the name of the stages for which the paths should be returned

Value

`get_stage` returns the name of the stage for a given path (or paths).

`get_paths` return a data.frame containing the paths corresponding to the given stage (or stages).

Examples

```
model <- fbhc.sevt(full(PhDArticles))
get_stage(model, c("0", "male"))
paths <- expand.grid(model$tree[2:1])[, 2:1]
get_stage(model, paths)
get_path(model, "Kids", "11")
get_path(model, "Gender", "2")
get_path(model, "Kids", c("5", "6"))
```

hc.sevt *Hill-Climb Score optimization*

Description

Hill-climbing search of staged event trees with iterative moving of nodes between stages.

Usage

```
hc.sevt(
  object,
  score = function(x) {     return(-BIC(x)) },
  max_iter = Inf,
  trace = 0
)
```

Arguments

object	a staged event tree object
score	a function that score staged event tree objects
max_iter	the maximum number of iterations per variable
trace	integer, if positive information on the progress is printed to console

Details

For each variable the node-move that best increase the score is performed. A node-move is either changing the stage associate to a node or move the node to a new stage.

Value

The final staged event tree obtained.

Examples

```
model <- hc.sevt(full(PhDArticles[, 1:3], lambda = 1))
summary(model)
```

---

inclusion.stages *Inclusion relations between stage structures of two models estimated on the same dataset*

Description

Inclusion relations between stage structures of two models estimated on the same dataset

Usage

```
inclusion.stages(object1, object2)
```

Arguments

object1	first staged event tree to compare
object2	second staged event tree to compare

Details

it computes the inclusion/exclusion/equality/diversity between the estimated stages between the two given models, in `object1` and `object2`.

Value

a list with inclusion relations between stage structures of each variable in the model

Examples

```
mod1 <- bhc.sevt(full(PhDArticles[, 1:5], lambda = 1))
mod2 <- fbhc.sevt(full(PhDArticles[, 1:5], lambda = 1))
inclusion.stages(mod1, mod2)
```

`is_fitted.sevt` *Check if a staged event tree is fitted*

Description

Check if a staged event tree is fitted

Usage

```
is_fitted.sevt(x)
```

Arguments

`x`	a staged event tree object

Value

logical

`join_stages` *Join stages*

Description

Join two stages in a staged event tree object, updating probabilities and log-likelihood accordingly.

Usage

```
join_stages(object, v, s1, s2)
```

Arguments

object	staged event tree
v	variable
s1	first stage
s2	second stage

Details

this function joins together two stages associated to the same variable.

Value

the staged event tree where s1 and s2 are joined

Examples

```
model <- full(PhDArticles, fit = TRUE, lambda = 0)
model <- fbhc.sevt(model)
model$stages$Kids
model <- join_stages(model, "Kids", "5", "6")
model$stages$Kids
```

---

join_zero_counts *Join situations with no observations*

Description

Join situations with no observations

Usage

```
join_zero_counts(object, fit = TRUE, trace = 0, name = NULL)

join_zero(object, fit = TRUE, trace = 0, name = NULL)
```

Arguments

object	a fitted staged event tree
fit	if the probability should be re-computed
trace	if > 0 print information to console
name	string with a name for the new stage

Details

It takes as input a (fitted) staged event tree object and looking at the `ctables` it joins, in the same stage, all the situations with zero recorded observations. Since such joining does not change the log-likelihood of the model, it is a useful (time-wise) pre-processing before others model selection algorithms. If `fit=TRUE` the model will be then re-fitted, if user sets `fit=FALSE` the returned model will have no probabilities.

Value

a staged event tree with situations with 0 observations merged in a single stage

Examples

```
DD <- generate_xor_dataset(n = 5, N = 10)
model_full <- full(DD, lambda = 1)
model <- join_zero_counts(model_full, fit = TRUE)
logLik(model_full)
logLik(model)
BIC(model_full, model)
```

---

logLik.sevt *Log-Likelihood of a staged event tree*

Description

Compute, or extract the log-likelihood of a staged event tree.

Usage

```
## S3 method for class 'sevt'
logLik(object, ...)
```

Arguments

`object`	the staged event tree object
`...`	additional parameters

Value

An object of class `logLik`.

Examples

```
data("PhDArticles")
mod <- staged_ev_tree(PhDArticles)
logLik(mod)
```

logLik.strt_ev_tree *Log-Likelihood of a stratified event tree*

Description

Compute the log-likelihood of a stratified event tree.

Usage

```
## S3 method for class 'strt_ev_tree'
logLik(object, ...)
```

Arguments

object	the stratified event tree object
...	additional parameters

Value

An object of class logLik.

Examples

```
DD <- as.data.frame(sapply(1:5, function(i) {
  return(as.factor(sample(c(1:3),
    size = 100, replace = TRUE
  )))
}))
sevt <- staged_ev_tree(DD, fit = TRUE)
evt <- strt_ev_tree(sevt)
logLik(evt)
```

naive.sevt *Naive staged event tree*

Description

Build a stage event tree with two stages for each variable

Usage

```
naive.sevt(object, distance = kl, k = length(object$tree))
```

Arguments

object	a full staged event tree
distance	a distance between probabilities
k	the maximum number of variable to consider

Value

A staged event tree with two stages per variable

Examples

```
DD <- generate_xor_dataset(n = 4, N = 1000)[, 5:1]
naive_model <- naive.sevt(full(DD, lambda = 1))
pr <- predict(naive_model, newdata = DD[501:1000, ])
table(pr, DD$C[501:1000])
```

---

nvar.sevt *Get the number of variables*

Description

Get the number of variables

Usage

```
nvar.sevt(x)
```

Arguments

x	a staged event tree object

Value

integer, the number of variables

Examples

```
mod <- indep(PhDArticles)
nvar.sevt(mod)
```

path_probability.sevt *Compute probability of a path from root*

Description

Compute probability of a path from root

Usage

```
path_probability.sevt(object, x, log = FALSE)
```

Arguments

object	a staged event tree object
x	the path, expressed as a character vector containing the sequence of the assumed levels
log	logical, if TRUE log-probability is returned

Details

it computes the probability of following a given path (x) starting from the root. Can be a full path from the root to a leaf or a shorter path.

Value

The probability of the given path or its logarithm if log=TRUE

Examples

```
DD <- generate_random_dataset(5, 100)
model <- staged_ev_tree(DD, fit = TRUE, lambda = 1)
path_probability.sevt(model, c("1", "-1", "1", "-1", "1"), log = TRUE) # root to leaf path
path_probability.sevt(model, c("1", "-1")) # short path

grid <- expand.grid(model$tree) # all paths from root to leaves

# joint distribution. it sums up to 1.
grid.prob <- apply(t(apply(grid, 1, as.character)), 1, path_probability.sevt, object = model)
cbind(grid, grid.prob)
```

PhDArticles *PhD Students Publications*

Description

Number of publications of 915 PhD biochemistry students during the 1950's and 1960's

Usage

```
PhDArticles
```

Format

A data frame with 915 rows and 6 variables:

Articles Number of articles during the last 3 years of PhD: either `0`, `1-2` or `>2`.

Gender `male` or `female`.

Kids `yes` if the student has at least one kid 5 or younger, `no` otherwise.

Married `yes` or `no`.

Mentor Number of publications of the student's mentor: `low` between 0 and 3, `medium` between 4 and 10, `high` otherwise.

Prestige `low` if the student is at a low-prestige university, `high` otherwise.

Source

The data has been modified from the `Rchoice` package.

References

Long, J. S. (1990). The origins of sex differences in science. *Social Forces*, 68(4), 1297-1316.

plot.sevt *Plot method for staged event trees*

Description

`plot.sevt` is the plot method for staged event tree objects. It allows easy plotting for staged event tree with some options, mainly different ways to specify colors for the stages (see examples).

Usage

```
## S3 method for class 'sevt'
plot(
  x,
  limit = 10,
  xlim = c(0, 1),
  ylim = c(0, 1),
  asp = 1,
  cex.label.nodes = 1,
  cex.label.edges = 1,
  cex.nodes = 2,
  col = NULL,
  ...
)
```

Arguments

x	staged event tree object
limit	maximum number of variables plotted
xlim	graphical parameter
ylim	graphical parameter
asp	graphical parameter
cex.label.nodes	graphical parameter
cex.label.edges	graphical parameter
cex.nodes	graphical parameter
col	color mapping for the stages, a named list with names equal to the variables names in the model and vectors named with stages names as components; otherwise if col == "stages" the stage names will be used as colors; otherwise if col is a function it will take as input a vector of stages and output the corresponding colors.
...	additional graphical parameters to be passed to points, lines, title, text and plot.window.

Examples

```
data("PhDArticles")
mod <- bj.sevt(full(PhDArticles))

### simple plotting
plot(mod)

### removing lables from nodes and edges and fill nodes
plot(mod, cex.label.nodes = 0, cex.label.edges = 0, pch = 16)
```

```
### reduce nodes size
plot(mod, cex.nodes = 1)

### change line width and nodes style
plot(mod, lwd = 3, pch = 5)

### changing palette
plot(mod, col = function(s) heat.colors(length(s)))

### or changing global palette
palette(hcl.colors(10, "Harmonic"))
plot(mod)

### manually give stages colors
simple <- naive.sevt(full(PhDArticles, lambda = 1))
#### simple has 2 stages per variable "1" and "2"
col <- lapply(simple$stages, function(s) {
  c("1" = "purple", "2" = "cyan")
})
plot(simple, col = col)
```

Pokemon *Pokemon Go Users*

Description

Demographic information of a population of possible Pokemon Go users.

Usage

```
Pokemon
```

Format

A data frame with 999 rows and 5 variables:

Use `Y` if the individual used the app, `N` otherwise

Age `>30` if the individual is older than 30, `<=30` otherwise

Degree `Yes` if the individual completed a Higher Education degree, `No` otherwise

Gender `Male` or `Female`

Activity `Yes` if the individual was physically active (i.e. had a walk longer than 30 mins, went for a run or had a bike ride to get some exercise) in the past week before the experiment, `No` otherwise

Source

https://osf.io/xy5g6/

References

Gabbiadini, Alessandro, Christina Sagioglou, and Tobias Greitemeyer. "Does Pokémon Go lead to a more physically active life style?." *Computers in Human Behavior* 84 (2018): 258-263.

`predict.sevt` *Predict method for staged event tree*

Description

Predict class values from a staged event tree model.

Usage

```
## S3 method for class 'sevt'
predict(object, newdata = NULL, class = NULL, prob = FALSE, log = FALSE, ...)
```

Arguments

`object`	A staged event tree
`newdata`	The newdata to perform predictions
`class`	A string indicating the name of the variable to use as the class variable, otherwise the first name in `names(object$tree)` will be used.
`prob`	logical, if `TRUE` the probabilities of class are returned
`log`	logical, if `TRUE` log-probabilities are returned
`...`	additional parameters, see details

Details

Predict the most probable a posterior value for the class variable given all the other variables in the model. Ties are broken at random and if, for a given vector of predictor variables, all conditional probabilities are 0, NA is returned.

if `prob = TRUE`, a matrix with number of rows equals to the number of rows in the `newdata` and number of columns as the number of levels of the `class` variable is returned. if `log = TRUE`, log-probabilities are returned.

if `prob = FALSE`, a vector of length as the number of rows in the `newdata` with the level with higher estimated probability for each new observations is returned.

Value

A vector of predictions or the corresponding probabilities

Examples

```
DD <- generate_xor_dataset(n = 4, 600)
order <- c("C", "X1", "X2", "X3", "X4")
train <- DD[1:500, order]
test <- DD[501:600, order]
model <- full(train)
model <- bhc.sevt(model)
pr <- predict(model, newdata = test, class = "C")
table(pr, test$C)
# class values:
predict(model, newdata = test, class = "C")
# probabilities:
predict(model, newdata = test, class = "C", prob = TRUE)
# log-probabilities:
predict(model, newdata = test, class = "C", prob = TRUE, log = TRUE)
```

`print.sevt` *Print a staged event tree*

Description

Print a staged event tree

Usage

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

Arguments

`x`	the staged event tree object
`...`	additional parameters (compatibility)

Details

The order of the variables in the stratified tree is printed (from root). In addition the number of levels of each variable is shown in square brackets. If available the log-likelihood of the model is printed.

Value

An invisible copy of `x`

Examples

```
DD <- generate_xor_dataset(5, 100)
model <- staged_ev_tree(DD, fit = TRUE, lambda = 1)
print(model)
```

prob.sevt *Compute probabilities for a staged event tree*

Description

Compute probabilities for a staged event tree

Usage

```
prob.sevt(object, x, log = FALSE, nan0 = TRUE)
```

Arguments

object	a (fitted) staged event tree object
x	the vector or data.frame of observations
log	logical, if TRUE log-probabilities are returned
nan0	logical, if NaN should be converted to 0

Details

it computes probabilities related to a vector or a data.frame of observations. They can be as an expand.grid data.frame or a simple subset of the dataset on which the model is estimated.

Value

the probabilities to observe each observation given in x

Examples

```
DD <- generate_random_dataset(5, 100)
model <- staged_ev_tree(DD, fit = TRUE, lambda = 1)
pr <- prob.sevt(model, expand.grid(model$tree[c(2, 3, 4)]))
sum(pr)
prob.sevt(model, DD[1:10, ])
```

probdist *Distances between probabilities*

Description

Distances between probabilities

Usage

```
lp(x, y, p = 2, ...)

ry(x, y, alpha = 2, ...)

kl(x, y, ...)

tv(x, y, ...)

hl(x, y, ...)

bh(x, y, ...)

cd(x, y, ...)
```

Arguments

x	vector of probabilities
y	vector of probabilities
p	exponent in the L^p norm
...	additional parameters for compatibility
alpha	the order of the Renyi divergence

Details

Functions to compute distances between probabilities:

- lp: the L^p distance, $||x - y||_p^p$
- ry: the symmetric Renyi divergence of order α
- kl: the symmetrized Kullback-Leibler divergence
- tv: the total variation or L^1 norm
- hl: the (squared) Hellinger distance
- bh: the Bhattacharyya distance
- cd: the Chan-Darwiche distance

Value

The distance between p and q

rename_stage *rename stage(s) in staged even tree*

Description

rename stage(s) in staged even tree

Usage

```
rename_stage(object, var, stage, new.label)
```

Arguments

object	staged event tree
var	string, one of the variable in the staged tree
stage	string or vector, the name of the stage(s) to be renamed
new.label	new name for the stage(s)

Value

a staged event tree object where stages stage have been renamed to new.label

sample.sevt *Sample from a staged event tree*

Description

Generate a random sample from the distribution encoded in a staged event tree object.

Usage

```
sample.sevt(object, n = 1)
```

Arguments

object	the staged event tree object
n	number of observations to sample

Details

It samples n observations according to the transition probabilities (object$prob) in the model.

Value

A data frame containing a sample of size n

Examples

```
model <- naive.sevt(full(PhDArticles, lambda = 1))
sample.sevt(model, 10)
```

`sevt.fit` *Fit a staged event tree*

Description

Estimate transition probabilities in a staged event tree from data. Probabilities are estimated with the relative frequencies plus, eventually, an additive (Laplace) smoothing.

Usage

```
sevt.fit(sevt, data = NULL, lambda = 0, ...)

fit.sevt(sevt, data = NULL, lambda = 0, ...)
```

Arguments

`sevt`	a staged event tree
`data`	data.frame, contingency table, or fitted stratified event tree
`lambda`	smoothing parameter
`...`	additional parameters

Details

The log-likelihood of the model will be recomputed and stored in the returned object.

`fit.sevt` is the same as `sevt.fit` and it will be probably removed in the future.

Value

a fitted staged event tree, that is an object of class `sevt` with `ctables` and `prob` arguments.

Examples

```
#########
model <- staged_ev_tree(list(
  X = c("good", "bad"),
  Y = c("high", "low")
))
D <- data.frame(
  X = c("good", "good", "bad"),
  Y = c("high", "low", "low")
)
model.fit <- sevt.fit(model, data = D, lambda = 1)
```

split_stage_random *split randomly a stage*

Description

Randomly assign some of the path to a new stage

Usage

```
split_stage_random(object, var, stage, p = 0.5)
```

Arguments

object	a staged event tree object
var	the variable where to split the stage
stage	the name of the stage to split
p	probability to move a situation from the original stage into the new stage

Details

It splits randomly a given stage into two stages. More precisely, it assigns each situation within the given stage into a new stage with probability p.

Value

a staged event tree object

Examples

```
DD <- generate_random_dataset(5, 100)
model <- staged_ev_tree(DD, fit = TRUE, full = TRUE, lambda = 1)
model <- bhc.sevt(model)
model <- stndnaming.sevt(model)
model$stages$X5

# no split
model1 <- split_stage_random(model, "X5", "1", p = 0)
model1$stages$X5

# all situations in the new stage
model2 <- split_stage_random(model, "X5", "1", p = 1)
model2$stages$X5

# randomly split with probability 0.5
model3 <- split_stage_random(model, "X5", "1", p = 0.5)
model3$stages$X5
```

`stagedtrees` *Staged event trees.*

Description

Algorithms to create, learn, fit and explore staged event tree models. Functions to compute probabilities and make predictions from the fitted models and to plot, analyze and manipulate staged event trees.

Details

Model selection algorithms:

- full model `full`
- independence model `indep`
- Hill-Climbing `hc.sevt`
- Backward Hill-Climbing `bhc.sevt`
- Fast Backward Hill-Climbing `fbhc.sevt`
- Backward Hill-Climbing Random `bhcr.sevt`
- Backward joining `bj.sevt`
- Naive Staged Event Tree `naive.sevt`

Probabilities, log-likelihood and predictions:

- Marginal probabilities `prob.sevt`
- Log-Likelihood `logLik.sevt`
- Predict method `predict.sevt`

Plot, explore and compare:

- Plot `plot.sevt`
- Compare `compare.sevt`
- Stages info `stages.sevt`, `summary.sevt`

References

Collazo R. A., Görgen C. and Smith J. Q. Chain event graphs. CRC Press, 2018.

Görgen C., Bigatti A., Riccomagno E. and Smith J. Q. Discovery of statistical equivalence classes using computer algebra. *International Journal of Approximate Reasoning*, vol. 95, pp. 167-184, 2018.

Barclay L. M., Hutton J. L. and Smith J. Q. Refining a Bayesian network using a chain event graph. *International Journal of Approximate Reasoning*, vol. 54, pp. 1300-1309, 2013.

Smith J. Q. and Anderson P. E. Conditional independence and chain event graphs. *Artificial Intelligence*, vol. 172, pp. 42-68, 2008.

Thwaites P. A., Smith, J. Q. A new method for tackling asymmetric decision problems. *International Journal of Approximate Reasoning*, vol. 88, pp. 624–639, 2017.

Examples

```
data("PhDArticles")
mf <- full(PhDArticles)
mod <- fbhc.sevt(mf)
plot(mod)
```

---

staged_ev_tree *Staged (stratified) event tree*

Description

Builds the staged event tree for a set of categorical variables, the order can be specified.

Usage

```
staged_ev_tree(x, ...)

## Default S3 method:
staged_ev_tree(x, ...)

## S3 method for class 'table'
staged_ev_tree(x, order = names(dimnames(x)), full = FALSE, fit = TRUE, ...)

## S3 method for class 'data.frame'
staged_ev_tree(x, order = colnames(x), full = FALSE, fit = TRUE, ...)

## S3 method for class 'list'
staged_ev_tree(x, full = FALSE, ...)

## S3 method for class 'bn.fit'
staged_ev_tree(x, order = NULL, ...)

## S3 method for class 'strt_ev_tree'
staged_ev_tree(x, lambda = 0, ...)

full(x, ...)

indep(x, ...)

## Default S3 method:
indep(x, ...)

## S3 method for class 'data.frame'
indep(x, fit = TRUE, lambda = 0, ...)
```

Arguments

x	data.frame, list, table, bn.fit object
...	additional parameters to be passed to the appropriate method, see sevt.fit
order	order of the variables
full	logical, if TRUE the full model is created
fit	logical, if TRUE the model is fitted
lambda	smoothing parameter

Details

A staged event tree object is a list with components:

- tree: A named list where for each variable, the levels of such variable are listed. The order of the variable is the order of the event tree.
- stages: A named list where each component stores the stages for the given variable.
- ctables: The contingency tables of the data.
- lambda: The smoothing parameter used to estimate the model.
- prob: The conditional probability tables for every variable and every stage.
- ll: The log-likelihood of the estimated model.

Value

A staged event tree object

Examples

```
######### from dataset
DD <- generate_random_dataset(n = 4, 1000)
indep <- staged_ev_tree(DD, fit = TRUE)
full <- staged_ev_tree(DD, full = TRUE, fit = TRUE, lambda = 1)

######### from table
modT <- staged_ev_tree(table(PhDArticles), fit = TRUE, full = TRUE, lambda = 1)
plot(modT)

######### from list
model <- staged_ev_tree(list(
  X = c("good", "bad"),
  Y = c("high", "low")
))

######### full model

DD <- generate_xor_dataset(4, 100)
modfull <- full(DD, lambda = 1)

######### independence model (data.frame)
```

```
DD <- generate_xor_dataset(4, 100)
system.time(model1 <- staged_ev_tree(DD, fit = TRUE, lambda = 1))
system.time(model2 <- indep(DD, lambda = 1))
model1
model2
```

`stages.sevt` *Stages of a variable*

Description

Obtain the stages of a given variable in a staged event tree object.

Usage

```
stages.sevt(object, var = NULL)
```

Arguments

`object`	a staged event tree object
`var`	name of one variable

Value

If `var` is specified, it returns a character vector with stage names for the given variable (that is `object$stages[[var]]`. Otherwise, If `var` is not specified, `stages.sevt` returns a list of character vectors containing the stages associated to each variable in the model (that is `object$stages`).

Examples

```
DD <- generate_xor_dataset(5, 100)
model <- staged_ev_tree(DD, fit = TRUE, lambda = 0)
stages.sevt(model, var = "X5")
```

`stndnaming.sevt` *Standard renaming of stages*

Description

Standard renaming of stages

Usage

```
stndnaming.sevt(object, rep = FALSE)
```

Arguments

object	a staged event tree object
rep	logical, if stages name can be repeated for different variables

Value

a staged event tree object with stages named with consecutive integers.

Examples

```
DD <- generate_xor_dataset(4, 100)
model <- staged_ev_tree(DD, full = TRUE)
model <- fbhc.sevt(model)
model$stages
model1 <- stndnaming.sevt(model)
model1$stages
```

---

strt_ev_tree *Stratified event tree*

Description

Builds the complete stratified event tree for a set of categorical variables, the order can be specified.

Usage

```
strt_ev_tree(x, order = NULL, fit = FALSE, ...)
```

Arguments

x	data.frame or list
order	Vector of variables names, the order to build the event tree
fit	If TRUE the contingency tables will be computed from data
...	additional parameters

Value

A stratified event tree object

Examples

```
mod <- strt_ev_tree(PhDArticles)
str(mod)
mod1 <- strt_ev_tree(PhDArticles, fit = TRUE)
str(mod1)
```

strt_ev_tree.data.frame *Stratified event tree from data.frame*

Description

Stratified event tree from data.frame

Usage

```
## S3 method for class 'data.frame'
strt_ev_tree(x, order = colnames(x), fit = FALSE, lambda = 0, ...)
```

Arguments

x	data.frame
order	vector of order to build the tree
fit	logical
lambda	laplace smoothing parameter
...	additional parameters

Value

A stratified event tree object

strt_ev_tree.default *Stratified event tree*

Description

Stratified event tree

Usage

```
## Default S3 method:
strt_ev_tree(x, ...)
```

Arguments

x	object to be coerced to data.frame
...	additional parameters

Value

A stratified event tree object

strt_ev_tree.list *Stratified event tree from list*

Description

Stratified event tree from list

Usage

```
## S3 method for class 'list'
strt_ev_tree(x, ...)
```

Arguments

x	a list with component named as the variables and containing the vector of factor
...	additional parameters

Details

Build the stratified event tree object from a named list containing the levels of the variables. The output is an object with the `tree` component.

Value

A stratified event tree object

strt_ev_tree.sevt *Stratified event tree from a staged event tree*

Description

Stratified event tree from a staged event tree

Usage

```
## S3 method for class 'sevt'
strt_ev_tree(x, ...)
```

Arguments

x	a staged event tree object
...	additional parameters

Details

This function build a stratified event tree object from a staged event tree.

Value

A stratified event tree object

`strt_ev_tree.table` *Stratified event tree from a table*

Description

Stratified event tree from a table

Usage

```
## S3 method for class 'table'
strt_ev_tree(x, order = names(dimnames(x)), fit = FALSE, lambda = 0, ...)
```

Arguments

`x`	table
`order`	vector of order to build the tree
`fit`	logical
`lambda`	laplace smoothing parameter
`...`	additional parameters

Value

A stratified event tree object

`subtree.sevt` *Extract subtree*

Description

Extract subtree

Usage

```
subtree.sevt(object, path)
```

Arguments

`object`	a staged event tree object
`path,`	the path after which extract the subtree

Details

it returns the subtree of the staged event tree given in input as `object`. The new root of the subtree is the given `path`.

Value

the staged event tree object corresponding to the subtree

Examples

```
DD <- generate_random_dataset(4, 100)
model <- staged_ev_tree(DD, full = TRUE)
plot(model)
model1 <- subtree.sevt(model, path = c("-1", "1"))
plot(model1)
```

`summary.sevt` *Summarizing staged event trees*

Description

Summary method for class `sevt`.

Usage

```
## S3 method for class 'sevt'
summary(object, ...)

## S3 method for class 'summary.sevt'
print(x, max = 10, ...)
```

Arguments

`object`	a staged event tree object
`...`	arguments for compatibility
`x`	an object of class `summary.sevt`
`max`	the maximum number of variables for which information is printed

Details

Print model information and summary of stages.

Value

An object of class `summary.sevt` for which a `print` method exist.

Examples

```
model <- naive.sevt(full(PhDArticles, fit = TRUE, lambda = 1))
summary(model)
```

`text.sevt` *Add text to a staged even tree plot*

Description

Add text to a staged even tree plot

Usage

```
## S3 method for class 'sevt'
text(x, y = ylim[1], limit = 10, xlim = c(0, 1), ylim = c(0, 1), ...)
```

Arguments

`x`	staged event tree object
`y`	the position of the labels
`limit`	maximum number of variables plotted
`xlim`	graphical parameter
`ylim`	graphical parameter
`...`	additional parameters

`varnames.sevt` *Get variable names*

Description

Get variable names

Usage

```
varnames.sevt(x)
```

Arguments

`x`	a staged event tree object

Value

vector with variable names

Examples

```
mod <- full(PhDArticles)
varnames.sevt(mod)
```

Index