Package ‘ceg’

October 12, 2022

Title Chain Event Graph

Date 2017-11-26

Version 0.1.0

Description Create and learn Chain Event Graph (CEG) models using a Bayesian framework. It provides us with a Hierarchical Agglomerative algorithm to search the CEG model space.
The package also includes several facilities for visualisations of the objects associated with a CEG. The CEG class can represent a range of relational data types, and supports arbitrary vertex, edge and graph attributes. A Chain Event Graph is a tree-based graphical model that provides a powerful graphical interface through which domain experts can easily translate a process into sequences of observed events using plain language. CEGs have been a useful class of graphical model especially to capture context-specific conditional independences. References: Collazo R, Gorgen C, Smith J. Chain Event Graph. CRC Press, ISBN 9781498729604, 2018 (forthcoming); and Barday LM, Collazo RA, Smith JQ, Thwaites PA, Nicholson AE. The Dynamic Chain Event Graph. Electronic Journal of Statistics, 9 (2) 2130-2169 <doi:10.1214/15-EJS1068>.

Depends R (>= 3.2.2)

Imports graph, grDevices, graphics, methods, stats, utils, Rgraphviz

License GPL-2 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat

URL `https://github.com/ptaranti/ceg`

BugReports `https://github.com/ptaranti/ceg/issues`

Collate 'category.R' 'ceg.R' 'event_tree.R' 'staged_tree.R' 'ceg_model.R' 'cont_table.R' 'data_documentation.R' 'model_search_algorithm.R' 'exhaustive_model_search_algorithm.R' 'dinamic_programming.R'

'distribution_of_probability.R' 'dirchlet_distribution.R'
'dirchlet_mpnl_distribution.R'
'heuristic_model_search_algorithm.R' 'lib_funtions.R'
'multinomial_distribution.R' 'oahc.R'
'posterior_distribution.R' 'prior_distribution.R'
'stratified_event_tree.R' 'stratified_staged_tree.R'
'stratified_ceg_model.R' 'variable.R'

NeedsCompilation no

Author Rodrigo Collazo [aut],
Pier Taranti [aut, cre]

Maintainer Pier Taranti <ptaranti@gmail.com>

Repository CRAN

Date/Publication 2017-11-27 12:44:01 UTC

R topics documented:

`ceg-package` *Chain Event Graph (ceg)*

Description

This package has functionalities that allow us to create and learn Chain Event Graph (CEG) models using a Bayesian framework. It provides us with a Hierarchical Agglomerative algorithm to search the CEG model space.

Details

The package also includes several facilities for visualisations of the objects associated with a CEG. The CEG class can represent a range of relational data types, and supports arbitrary vertex, edge and graph attributes. A Chain Event Graph is a tree-based graphical model that provides a powerful graphical interface through which domain experts can easily translate a process into sequences of observed events using plain language. CEGs have been a useful class of graphical model especially to capture context-specific conditional independences.

Currently, ceg provides implementation to support the stratified family, the user will use the following classes:

- Stratified.ceg.model
- Stratified.staged.tree
- Stratified.event.tree

These classes are implemented as S4 classes and have constructor methods with the same name as the class. A `plot` method is also provided.

Author(s)

Maintainer: Pier Taranti <`ptaranti@gmail.com`>

Authors:

- Rodrigo Collazo <`rodrigocollazo@gmail.com`>

See Also

Useful links:

- `https://github.com/ptaranti/ceg`
- Report bugs at `https://github.com/ptaranti/ceg/issues`

`AlphaEdgeInternal` *AlphaEdgeInternal*

Description

`AlphaEdgeInternal` yields a possible objective prior distribution for each situation associated with a particular variable in the event tree.

Usage

```
AlphaEdgeInternal(level, stratified.event.tree, alpha)
```

Arguments

`level`	numeric - It indicates the level in the event tree.
`stratified.event.tree`	Stratified.event.tree - S4 object that represents an event tree.
`alpha`	numeric - It plays a role of a phantom sample to construct the prior probability distribution of a situation associated with a particular variable in the event tree.

Value

"vector" - Dirichlet hyperparameter vector of a situation associated with a particular variable.

`artificial.chds` *test dataset - artificial.chds.*

Description

A dataset with dummy data, based on Child Health and Development Studies (CHDS).

Usage

```
data(artificial.chds)
```

Format

a data.frame with 1500 rows and 4 categorical variables. The variables names and values are compliant with CHDS, but the values are randomly filled.

Social High, Low

Economic High, Low

Event High, Average, Low

Admission No, Yes

Examples

```
data(artificial.chds)
```

`Category` *Category(label)*

Description

Category(label) is a function that act as constructor to Category S4 object. Category S4 class contains a single slot with the category labels. It is used to construct S4 Variable objects, which, in turn, aim at being parameters in Stratified.event.tree objects manual constructions.

Usage

```
Category(label)
```

Arguments

`label`	caracter, the category name

Value

a `Category` S4 object

Examples

```
cat <- Category("category.name")

Category("category.name2")
```

`Category-class` *Category S4 Class*

Description

Category S4 class contains a single slot with the category label. It is used to construct Stratified.event.tree objects.

Slots

`label` character

`Ceg.model-class` *Ceg.model S4 class*

Description

`Ceg.model` is a S4 class whose objects represent a Chain-Event Graph (CEG) model, which is composed by a Staged Tree object and its corresponding staged structure.

Slots

`staged.tree` Staged.tree S4 object

`position` list

`CegGraphSimple` *CegGraphSimple*

Description

Simple ceg structure to be ploted in RGraphviz. This function yields a data structure corresponding a simplified CEG to be plotted using the package Rgraphviz.

Usage

```
CegGraphSimple(stratified.event.tree, position, range.color = 1)
```

Arguments

`stratified.event.tree`	stratified.event.tree S4 object
`position`	list an object ceg.position.
`range.color`	numeric it chooses the palette. If 1, it is used a 8-color palette. If 2, it is used a 501-color palette.

Value

list

- $node - node attributes
- $node$nodes (vector) - set of positions.
- node$variable (vector) - it identifies the variable asscoiated with each position.
- node$color (vector) - color of each position. All positions coincident with a stage are depicted in white.
- $edge - edge attributes
- $edge$edges (list) - set of list that emanates from each position.
- edge$label (vector) - position labels.
- weight (vector) - edge weight.

Note

This function mitigates a limitation from Rgraphviz, which does not support plotting multiple edges between two nodes presenting the correct edge label for each one. The decision was to merge all edges in one, and presenting all labels in this resulting edge.#' This approach is temporary and not ideal, since the ceg is no more a multi-graph. However, the authors did not find a graphical package which provides the needed plotting features. Contributions are wellcomed.

`CheckAndCleanData` *CheckAndCleanData*

Description

RemoveRowsWithNAandVoid remove all rows with NA and void ("") values data from a data.frame

Usage

```
CheckAndCleanData(data.frame)
```

Arguments

`data.frame`	a data frame to be used to create stratified event/staged trees

Value

data.frame with no void or NA values.

`ContingencyTable` *ContingencyTable*

Description

This function creates the contigency tables associated with each variable in the event tree.

Usage

```
ContingencyTable(data, stratified.event.tree)
```

Arguments

`data`	data.frame whose columns depict variables and rows correspond to units that are observed in the system
`stratified.event.tree`	Stratified.event.tree S4 object

Value

a list of matrices that represent the contigency tables associated with each variable in the event tree. The matrix corresponding to a particular variable presents the counts of each combination of the categories of the variables that precede it in the event tree according to its categories. The combinations of the categories of the upstream variables are displayed on the rows and represent the situations associated with the target variable. The categories of the target variable are represented on the columns and corresponds to each event that can unfold from a situation associated with the target variable.

`ContingencyTableVariable` *ContingencyTableVariable*

Description

This function calculates the contigency table associated with a specific variable.

Usage

```
ContingencyTableVariable(variable, data, stratified.event.tree)
```

Arguments

`variable`	numeric
`data`	data.frame whose columns depict variables and rows correspond to units that are observed in the system
`stratified.event.tree`	Stratified.event.tree S4 object

Value

a matrix that presents the counts of each combination of the categories of the variables that precede the target variable in the event tree according to the categories of the target variable. The combinations of the categories of the upstream variables are displayed on the rows and represents a situation associated with the target variable. The categories of the target variable are represented on the columns and corresponds to each event that can unfold from a situation associated with the target variable.

`Dinamic.programming-class` *Dinamic.programming S4 Class*

Description

Dinamic.programming S4 Class

Note

Inserted fot future use

`Dirchlet.distribution-class` *Dirchlet.distribution*

Description

Dirchlet.distribution

Slots

`score` numeric.

`cluster` list.

`Dirchlet.MPNL.distribution-class` *Dirchlet.MPNL.distribution*

Description

Dirchlet.MPNL.distribution

Slots

`score` numeric.

`cluster` list.

`Distribution.of.probability-class` *Distribution.of.probability S4 Class*

Description

Distribution.of.probability S4 Class

Slots

`score` numeric.

`cluster` list.

`EdgeLabel` *EdgeLabel*

Description

This function yields the edge labels. The edges are labeled accordingly the original data provided.

Usage

```
EdgeLabel(num.variable, num.situation, label)
```

Arguments

`num.variable`	numeric - number of variables.
`num.situation`	vector - number of stages associated with each variable.
`label`	list of vectors - each component is a vector that contains the event names associated with each variable.

Value

vector - edge labels

`EdgeList` *EdgeList*

Description

Function EdgeList genereates the list of edges of an event tree.

Usage

```
EdgeList(stratified.event.tree, node)
```

Arguments

`stratified.event.tree`	Stratified.event.tree S4 object
`node`	(vector) - an object generated by the function node.list

Value

list of lists - each list component is a vector that represents the edges that emanate from a vertice.

`EdgeSituation` *EdgeSituation*

Description

EdgeSituation identifies the edges from a situation (node). This function identifies the edges that emanate from a particular situation in an EventTree.

Usage

```
EdgeSituation(situation, start.situation, num.category)
```

Arguments

`situation`	numeric - it identifies the target situation whose emanating edges are our interesting.
`start.situation`	vector - it identifies the situation that begins a new level.
`num.category`	vector - it identifies the number of edges that emanate from situations in each level.

Value

list of lists - each list component is a vector that represents the edges that emanate from a vertice.

`Event.tree-class` *Event.tree S4 object*

Description

Event.tree S4 object

`Exhaustive.model.search.algorithm-class` *Exhaustive.model.search.algorithm S4 Class*

Description

Exhaustive.model.search.algorithm S4 Class

`Heuristic.model.search.algorithm-class` *Heuristic.model.search.algorithm*

Description

Heuristic.model.search.algorithm

`KeepLexOrder` *KeepLexOrder*

Description

This function keep a lexicographical order of a vector

Usage

```
KeepLexOrder(ref, order.vector, score.vector)
```

Arguments

`ref`	numeric
`order.vector`	vector
`score.vector`	vector

`LabelStage` *LabelStage*

Description

This function identifies the edges arriving at the target level for paths that exist from the root node to each situation in the event tree that are in levels greater than the target level.

Usage

```
LabelStage(k, num.variable, num.situation, label.category, num.category)
```

Arguments

`k`	numeric
`num.variable`	numeric
`num.situation`	numeric
`label.category`	list
`num.category`	list @return label a vector

See Also

`TruncatedPath`

`ListToVector` *ListToVector*

Description

This function change a list of vectors in a vector.

Usage

```
ListToVector(x, n)
```

Arguments

`x`	list of vectors
`n`	numeric

Value

vector

`MergeLabels` *MergeLabels*

Description

Merge labels of multiple edges in order to plot them all.

Usage

```
MergeLabels(edge.list, edge, level)
```

Arguments

`edge.list`	vector list of positions that a children of a specific position v1.
`edge`	numeric a particular children "edge" of a specific position v1
`level`	vector labels of classes corrresponding to the variable associated with a position.

Value

list merged labels associated with a specific position v1.

Note

This function mitigates a limitation from Rgraphviz, since it is not possible to plot multiple edges between two nodes presenting the correct edge label for each one. The authors did not find a graphical package providing this capability. Contributions are wellcomed.

`Model.search.algorithm-class` *Model.search.algorithm*

Description

Model.search.algorithm

`Multinomial.distribution-class` *Multinomial.distribution*

Description

Multinomial.distribution

Slots

`score` numeric.

`cluster` list.

`NodeColor` *NodeColor*

Description

This function yields the node colors.

Usage

```
NodeColor(num.variable, num.situation, num.category, stage.structure,
  range.color)
```

Arguments

`num.variable`	(numeric) - number of variables.
`num.situation`	(vector) - number of stages associated with each variable.
`num.category`	(vector) - it identifies the number of edges that emanate from situations in each level.
`stage.structure`	list with two components: • numeric - score associated with a level • list of vectors - stage structure
`range.color`	(numeric) - it chooses the palette. If 1, it is used a 8-color palette. If 2, it is used a 501-color palette.

Value

vector - node colors

NodeLabel *NodeLabel*

Description

This function yields the node labels. The nodes are labeled accordingly, to indicate diferente positions.

Usage

```
NodeLabel(num.variable, num.situation, num.category, label)
```

Arguments

`num.variable`	numeric - number of variables.
`num.situation`	vector - number of stages associated with each variable.
`num.category`	vector - it identifies the number of edges that emanate from situations in each level.
`label`	list of vectors - each component is a vector that contains the event names associated with each variable.

Value

vector - node labels

`NodeSet` *NodeSet*

Description

This function genereates the nodes of an event tree.

Usage

```
NodeSet(tree)
```

Arguments

`tree`	Event.tree S4 object

Value

vector

`OAHC` *OAHC Constructor*

Description

This function calculates the best stage configuration of a hyperstage associated with a specific variable of time-slice t_0 or t_k, k>=1, using the oahc algorithm (oahc - Optimised Agglomerative Hierarchical Clustering)

Usage

```
OAHC(level, prior.distribution, contingency.table, tree)
```

Arguments

`level`	numeric - level under optimisation
`prior.distribution`	(list of matrices) - see function `prior.distribution`
`contingency.table`	(list of matrices) - see function ContingencyTable
`tree`	an object 'Event.tree'

Value

a OAHC S4 object

See Also

SingleScore, PairwiseScore, SingleReorder, NaReorder, KeepLexOrder

`OAHC-class` *OAHC S4 Class*

Description

@include heuristic_model_search_algorithm.R

Slots

`score` numeric

`cluster` list

`PairwisePosition` *PairwisePosition*

Description

The `PairwisePosition` function identifies if two situations are in the same position given that they are in the same stage.

Usage

```
PairwisePosition(pair.situation, num.category, pos.next.level)
```

Arguments

`pair.situation`	(vector) - situations to be analysed
`num.category`	(numeric) - number of edges that unfolds from the situations
`pos.next.level`	(vector) - It identifies the positions corresponding to all situations that are children of situations associated with the variable spanning our target stage.

Value

boolean

plot,Stratified.ceg.model,ANY-method

Stratified.ceg.model Plotting

Description

This Method is used to plot a chain event graph from a Stratified.ceg.model S4 object. The current ceg package implementation depends on Rgraphviz package from Bioconductor to draw the CEG graph.

Usage

```
## S4 method for signature 'Stratified.ceg.model,ANY'
plot(x)
```

Arguments

x	Stratified.ceg.model S4 object.

Value

the plot and also a pdf version is saved in the working directory.

Examples

```
plot(scm)
```

plot,Stratified.event.tree,ANY-method

Stratified.event.tree Plotting

Description

Method to plot a Stratified.event.tree S4 object. The current ceg package implementation depends on Rgraphviz package from Bioconductor for plotting.

Usage

```
## S4 method for signature 'Stratified.event.tree,ANY'
plot(x)
```

Arguments

x	Stratified.event.tree S4 object

Value

the plot and also a pdf version is saved in the working directory.

Examples

```
plot(set)
```

`plot,Stratified.staged.tree,ANY-method`
Stratified.staged.tree Plotting

Description

Method to plot a Staged.tree S4 object. The current package `ceg` depends on `Rgraphviz` package from Bioconductor to draw graphs.

Usage

```
## S4 method for signature 'Stratified.staged.tree,ANY'
plot(x)
```

Arguments

`x`	Stratified.staged.tree S4 object

Value

the plot. A pdf version is also saved in the working directory.

Examples

```
plot(sst)
```

PositionLevel *PositionLevel*

Description

This function obtains the position structure associated with a particular variable of a CEG.

Usage

```
PositionLevel(stage.list, num.category, num.situation.next,
  pos.next.level = list())
```

Arguments

`stage.list`	(list) - stage structure associated with a particular variable.
`num.category`	(vector) - number of edges that unfolds from each position asscoiated with our target variable
`num.situation.next`	(numeric) - number of situations associated with the variable that follows our target variable in the event tree.
`pos.next.level`	(list) - position structure associated with the variable that follows our target variable in the event tree (see function PositionLevel)

Value

list of lists - The first list level identifies a stage 'i' and the second list level identifies the positions associated with this stage 'i'.

See Also

`PositionVector`, `PositionStage` and `PairwisePosition`

PositionStage *PositionStage*

Description

`PositionStage` function yields the position structure associated with a particular stage of a CEG.

Usage

```
PositionStage(stage.vector, num.category, pos.next.level)
```

Arguments

`stage.vector`	(vector) - a set of situations that constitute a particular stage
`num.category`	(numeric) - number of edges that unfolds from the situations
`pos.next.level`	(vector) - It identifies the positions corresponding to all situations that are children of situations associated with the variable spanning our target stage.

Value

list of vector - Each vector identifies a position.

See Also

`PairwisePosition`

`PositionVector` `PositionVector` *function rewrites a position structure associated with a particular variable: from a list to a vector.*

Description

`PositionVector` function rewrites a position structure associated with a particular variable: from a list to a vector.

Usage

```
PositionVector(num.situation, pos.list)
```

Arguments

`num.situation`	(numeric) - number of situation associated with a particular variable.
`pos.list`	(list) - stage structure associated with a particular variable that follows the variable associated with our target position.

Value

vector

`Posterior.distribution-class` *Posterior.distribution*

Description

Posterior.distribution

`Prior.distribution-class` *Prior.distribution*

Description

Prior.distribution

`PriorDistribution` *PriorDistribution*

Description

`PriorDistribution` initialises the prior distributions under the conservative and uniform assumptions for the hyperparameter 'alpha' over the event tree.

Usage

```
PriorDistribution(stratified.event.tree, alpha)
```

Arguments

`stratified.event.tree`	"Stratified.event.tree" a S4 object that represents an event tree.
`alpha`	numeric It plays a role of a phantom sample to construct the prior probability distribution and reprssents the prior knowledge about the process.

Value

prior is a list of matrices. Each matrix is a collection of vectors that correspond to a prior for each situation associated with a particular variable.

See Also

`PriorVariable`

`PriorVariable` *PriorVariable*

Description

The function `PriorVariable` yields the prior distributions for all situations associated with a particular variable in the event tree.

Usage

```
PriorVariable(ref, alpha.edge)
```

Arguments

`ref`	numeric - It indicates the variable.
`alpha.edge`	vector - Dirichlet hyperparameter vector of a situation associated with a particular variable.

Value

a matrix. Each row represents the Dirichlet hyperparameter vector of a situation associated with a particular variable in the event tree.

See Also

`Prior.distribution` and `AlphaEdgeInternal`

`scm` *test stratified ceg model*

Description

A Stratified.ceg.model S4 object, generated using the command `scm <- Stratified.ceg.model(sst)`

Usage

```
data(scm)
```

Format

a Stratified.ceg.model S4 object

Examples

```
data(scm)
```

`set` *test stratified event tree*

Description

A Stratified.event.tree S4 object, generated using the command `set <- Stratified.event.tree(artificial.chds)`

Usage

```
data(set)
```

Format

a Stratified.event.tree S4 object

Examples

```
data(set)
```

`set.manual` *test stratified event tree (manualy constructed)*

Description

A Stratified.event.tree S4 object, generated using manual input.
See Stratified.event.tree documentation examples.

Usage

```
data(set)
```

Format

a Stratified.event.tree S4 object

Examples

```
data(set.manual)
```

`sst` *test stratified staged tree*

Description

A Stratified.staged.tree S4 object, generated using the command `sst <- Stratified.staged.tree(artificial.chds)`

Usage

```
data(sst)
```

Format

a Stratified.staged.tree S4 object

Examples

```
data(sst)
```

`Staged.tree-class` *Staged.tree*

Description

A staged tree is an event tree embellished with colours using a probabilistic measure. Two situations are said to be in the same stage if they have equivalent probabilistic space and identical conditional probabilities. Each stage is associated with a different colour.

Slots

`event.tree` Event.tree.

`Stratified.ceg.model` *Stratified.ceg.model constructor.*

Description

S3 function to friendly construct S4 Stratified.ceg.model.

Usage

```
Stratified.ceg.model(stratified.staged.tree)
```

Arguments

`stratified.staged.tree`
: Stratified.staged.tree S4 object A staged tree is called stratified if its supporting event tree is stratified and all vertices which are in the same stage are also at the same distance of edges from the root.

Value

a Stratified.ceg.model S4 object.

Examples

```
scm <- Stratified.ceg.model(sst)
```

`Stratified.ceg.model-class`
Stratified.ceg.model

Description

The `Stratified.ceg.model` is a S4 class that extends `Ceg.model`. The object represents a CEG model derived from its supporting Stratified.staged.tree using some graphical transformation rules.

`Stratified.event.tree` *Stratified.event.tree*

Description

Constructor method to Stratified.event.tree S4 objects. It accepts different sets for parameters types.

Usage

```
Stratified.event.tree(x, ...)

## S4 method for signature 'missing'
Stratified.event.tree(x)

## S4 method for signature 'ANY'
Stratified.event.tree(x, ...)

## S4 method for signature 'data.frame'
Stratified.event.tree(x = "data.frame")

## S4 method for signature 'list'
Stratified.event.tree(x = "list")
```

Arguments

`x`	(data.frame) , where data.frame is a well behavioured data set; or (list) , list of Variable S4 objects, in the expected order of plotting.
`...`	(not used)

Value

a Stratified.event.tree S4 object

Note

A Stratified.event.tree may be manualy created (see examles)
A call to `Stratified.event.tree( )` with no parameters will return an error message for missing argument.
A call to `Stratified.event.tree(x, ...)`, x not being a data.frame or a list, will return an error message.

Examples

```
set <- Stratified.event.tree(artificial.chds)

set.manual <- Stratified.event.tree(list(Variable("age",list(Category("old"),
Category("medium"), Category("new"))),Variable("state", list(Category("solid"),
Category("liquid"), Category("steam"))), Variable("costumer",
list(Category("good"), Category("average"), Category("very bad"),
Category("bad")))))
```

`Stratified.event.tree-class` *Stratified.event.tree S4 Class*

Description

An event tree is called stratified if the set of events that unfold from all situations, which are at the same distance of edges from the initial situation, are identical.

`Stratified.staged.tree`
Stratified.staged.tree

Description

Constructor method to Stratified.staged.tree S4 objects. It accepts different sets for parameters types.

Usage

```
Stratified.staged.tree(x, y, z, ...)

## S4 method for signature 'missing,ANY,ANY'
Stratified.staged.tree(x, y, z, ...)

## S4 method for signature 'ANY,ANY,ANY'
Stratified.staged.tree(x, y, z, ...)


  ## S4 method for signature 'data.frame,numeric,numeric'
Stratified.staged.tree(x = "data.frame",
  y = 1L, z = 0L)


  ## S4 method for signature 'data.frame,numeric,missing'
Stratified.staged.tree(x = "data.frame",
  y = 1L)


  ## S4 method for signature 'data.frame,missing,missing'
Stratified.staged.tree(x = "data.frame")


  ## S4 method for signature 'Stratified.event.tree,list,ANY'
Stratified.staged.tree(x = "Stratified.event.tree",
  y = "list")
```

Arguments

`x`	(data.frame) is a well behavioured data set or (Stratified.event.tree)
`y`	(numeric) alpha or (list) that represents the stage.structure. To construct it, the user must plot the Stratified.event.tree graph and use the labelled number of each node.
`z`	(numeric) variable.order
`...`	(not used)

Value

a Stratified.staged.tree S4 object

Note

The implementation admits providing the three arguments, or the first two, or even only the data.frame.
The default variable order is as in the data.frame and the default alpha is 1L.
To manualy create a stratified.event.tree from a stratified.event.tree:

1st plot the stratified.event.tree - `plot(set)`

2nd Looking the graph, you can create the stage structure, such as: `stage.structure <- list(list(c(2,3)), list(c(4,7,12),c(5,8,9)))`

3rd Finally you can create your Stratified.event.tree: `st.manual<- Stratified.staged.tree(set, stage.structure)`

A call to `Stratified.staged.tree( )` with no parameters will return an error message for missing argument.
A call to `Stratified.staged.tree(x, ...)`, x not being a data.frame or a Event.tree, will return an error message.

Examples

```
sst <- Stratified.staged.tree(artificial.chds)

stt.manual <- Stratified.staged.tree(set.manual,
list(list(c(2,3)), list(c(4,7,12),c(5,8,9))))
```

`Stratified.staged.tree-class` *Stratified.staged.tree*

Description

A stratified staged tree is a staged tree whose supporting event tree is stratified and all vertices which are in the same stage are also at the same distance of edges from the root.

Slots

`event.tree` Stratified.event.tree. An stratified event tree is an event tree whose set of events that unfold from all situations, which are at the same distance of edges from the initial situation, are identical.

`situation` list.

`contingency.table` list of matrices that represent the contigency tables associated with each variable in the event tree.

`stage.structure` list in which each component is a list associated with a variable in the staged tree that has the following data structure:

- $score - numeric. This is the logarithmic form of the marginal likelihood associated with a particular variable.
- $cluster - list whose components are vectors. Each vector represents a stage associated with a particular variable.

`stage.probability` list in which each component is a list associated with a variable in the staged tree. Each component of this sublist is a vector that represents the probability distribution associated with a particular stage of the target variable.

`prior.distribution` list of matrices. Each matrix is a collection of vectors that correspond t a prior distribution for each situation associated with a particular variable.

`posterior.distribution` list of matrices. Each matrix is a collection of vectors that correspond t a prior distribution for each situation associated with a particular variable.

`model.score` numeric. This is the logarithmic form of the marginal likelihood.

`StratifiedCegPosition` *StratifiedCegPosition*

Description

This function obtains the position structure associated with a stratified CEG.

Usage

```
StratifiedCegPosition(stage, num.category, num.situation)
```

Arguments

`stage`	(list) - stage structure associated with a particular variable.
`num.category`	(vector) - number of edges that unfold from stages associated with a particular variable.
`num.situation`	(vector) - number of situations associated with each variable.

Value

list of lists

- First list level identifies a variable ’v’.
- Second list level identifies a stage ’a’ associated with a variable ’v’.
- The third list level identifies the positions associated with a stage ’a’ .

@seealso `PositionLevel`, `PositionVector`, `PositionStage`, `PairwisePosition`

`StratifiedEventTreeGraph` *StratifiedEventTreeGraph*

Description

StratifiedEventTreeGraph

Usage

```
StratifiedEventTreeGraph(event.tree)
```

Arguments

`event.tree`	"Event.tree" S4 object @return list with a data structure that is adequate to plot an event tree

`TreeGraph` *TreeGraph*

Description

A function to produce the data structure needed to plot Event and Staged trees using **RGraphviz**.

Usage

```
TreeGraph(tree, solution = list(), name = c(), range.color = 1)
```

Arguments

`tree`	Event.tree S4 object
`solution`	list with two components: • numeric - score associated with a level • list of vectors - stage structure
`name`	vector of strings - variable names
`range.color`	(numeric) - it chooses the palette. If 1, it is used a 8-color palette. If 2, it is used a 501-color palette.

Value

list:

- $node - node attributes
  - $node$nodes (vector) - set of situations.
  - node$label (vector) - it identifies the variable asscoiated with each position.
  - node$color (vector) - color of each situation. All situations coincident with a stage are depicted in black.
- $edge - edge attributes
  - $edge$edges (list) - set of list that emanates from each situation.
  - edge$label (vector) - edge labels.

`TruncatedPath` *TruncatedPath*

Description

This internal function yields a vector that contains the edges arriving at situations associated with a particular variable for all paths that emanate from the root node and pass through these situations in the event tree.

Usage

```
TruncatedPath(ref, k, var, num.category, num.situation, label.category)
```

Arguments

`ref`	numeric
`k`	numeric
`var`	numeric
`num.category`	list
`num.situation`	list
`label.category`	list

Variable *Variable(name,categories)*

Description

Variable(name,categories) is a function that act as constructor to Variable S4 object. Variable S4 class contains two slots with the Variable name and a list of Categories. It is used to construct Stratified.vent.tree objects.

Usage

```
Variable(name, categories)
```

Arguments

`name`	character, the Variable name
`categories`	a list of S4 Category objects.

Value

a `Variable` S4 object

Examples

```
var <- Variable("variable.name", list(Category("cat1"), Category("cat2"),
Category("cat3")))
```

Variable-class *Variable S4 Class*

Description

Variable S4 class contains two slots with the Variable name and a list of Categories. It is used to construct Stratified.vent.tree objects.

Slots

`name` character.

`categories` list of Category S4 objects.

Index