Rtreemix

October 25, 2011

`distances` *Different distances between two given vectors*

Description

These functions are used for calculating different distances between two given vectors. Thus, `L1.dist` calculates the L1 distance, `cosin.dist` calculates the cosine distance, `euclidian.dist` computes the Euclidian distance, and `rank.cor.dist` computes the rank correlation distance. The vectors have to have same length. When using `rank.cor.dist` the vectors have to have length larger than 4.

Usage

```
L1.dist(p, q)
cosin.dist(p, q)
euclidian.dist(x, y)
rank.cor.dist(x, y)
```

Arguments

`p`	A `numeric` vector specifying the first component for the distance calculation. It has to have the same length as `q`.
`q`	A `numeric` vector specifying the second component for the distance calculation.
`x`	Same as `p`.
`y`	Same as `q`.

Value

The functions return the distance between the two given vectors.

Author(s)

Jasmina Bogojeska

See Also

`kullback.leibler`, `L2.norm`, `stability.sim`

Examples

```
## Define two numeric vectors with equal lengths (> 4).
x <- c(1, 2, 3, 4, 5)
y <- c(5, 6, 7, 8, 9)

## Calculate the L1 distance between the vectors x and y
L1.dist(x, y)

## Calculate the Euclidian distance between the vectors x and y
euclidian.dist(x, y)

## Calculate the cosine distance between the vectors x and y
cosin.dist(x, y)

## Calculate the rank-correlation distance between the vectors x and y
rank.cor.dist(x, y)
```

L2.norm *L2 norm of a given vector*

Description

A function for calculating the L2 norm of a given `numeric vector`.

Usage

```
L2.norm(x)
```

Arguments

`x`	A `numeric` vector.

Value

The function returns the L2 norm of the given vector `x`.

Author(s)

Jasmina Bogojeska

See Also

`L1.dist`

Examples

```
## Define a numeric vector
y <- c(1, 2, 3, 4)

## Calculate the L2 norm of the vector y
L2.norm(y)
```

`Pval.dist` *p-value of a given similarity value*

Description

This function calculates the p-value of a given similarity value, i.e. the probability for obtaining the same or a smaller value than the given one in a vector of random similarity values. The p-value is used to determine whether the given similarity value is significant.

Usage

```
Pval.dist(dist.val, random.vals)
```

Arguments

`dist.val`	A `numeric` value quanifying the similarity for which a p-value should be calculated.
`random.vals`	A `numeric` vector of random similarities used for calculating the p-value.

Value

It returns a `numeric` value between 0 and 1 that specifies the p-value of the given `dist.val`.

Author(s)

Jasmina Bogojeska

See Also

`L1.dist`, `kullback.leibler`, `comp.models`, `stability.sim`

Examples

```
## The function is currently defined as
function(dist.val, random.vals) {
  return((sum(random.vals <= dist.val) + 1) /(length(random.vals) + 1))
  }

## Define the similarity value and a vector of random similarities
sim.val <- 0.2
rand.vals <- c(0.1, 0.24, 0.28, 0.35, 0.15, 0.5, 0.14, 0.6, 0.8, 0.3)

## Calculate the p-value of sim.val using the vector of random
## similarities
Pval.dist(dist.val = sim.val, random.vals = rand.vals)
```

RtreemixData-class *Class "RtreemixData"*

Description

This class is used to represent the results of genetic measurements of the occurence of subsets of a given set of genetic events in a group of patients. Each observation is a binary vector that indicates which events occured in a specific patient. The length of the vector equals the size of the set of genetic events that is taken into consideration.

Objects from the Class

Objects can be created by calls of the form `new("RtreemixData", Sample, Patients, Events, Description, File)`. The `RtreemixData` class represents patterns of occurences of subsets of a given set of genetic events in a specific group of patients. The patterns are given as binary vectors with length equal to the size of the set of genetic events. In other words, it provides a representation of the dataset used for learning an mutagenetic trees mixture model.

The `Sample` is a binary `matrix` where each row corresponds to the pattern of genetic events observed in one of the given patients. Hence, the number of rows gives the number of patients, i.e. the size of the dataset. Each column corresponds to one of the genetic events. Missing measurement for the presence or absence of a certain genetic event in a given pattern is marked with -1. The initial null event (that initially occurs in all patients) is not present in the sample, i.e. the first component in each observation (which is always equal to 1) is left out. This is done for saving space and avoiding the process of checking for correctly specified samples.

The `Patients` is a character `vector` that contains the IDs of the patients. The length of this vector must be equal to the number of rows in the `Sample`.

The `Events` is a character `vector` that contains the labels of the genetic events taken into consideration. Its length equals one plus the number of columns in the `Sample`. This is because of the label of the null event. When the object of class `RtreemixData` is a parent of a randomly generated `RtreemixModel` object, the events specify the labels of the genetic events present in the random model, although the `Sample` slot is an empty `matrix`. This is because the random mixture models are not estimated from a given dataset, but generated randomly for some set of genetic events.

The `Description` is a character giving a short description for the created object.

The `File` specifies the path to a text file with a specific format which contains the infromation needed to create an `RtreemixData` object (the patient IDs, the names of the events, the matrix with the observations).

Slots

`Sample`**:** Object of class `"matrix"`.

`Patients`**:** Object of class `"character"`. The `Patients` must be of same length as the number of rows in `Sample`.

`Events`**:** Object of class `"character"`. The length of `Events` must be identical to the number of columns in `Sample` plus one (for specifying the label of the null event).

`Description`**:** Object of class `"character"`.

Methods

Description `signature(object = "RtreemixData")`: A method for obtaining the description of the `"RtreemixData"` object.

Description<- `signature(object = "RtreemixData")`: A method for specifying the `Description` of the data object.

Events `signature(object = "RtreemixData")`: A method for obtaining the labels of the genetic events.

Events<- `signature(object = "RtreemixData")`: A method for replacing the names of the genetic events in the data object. It checks to be sure the values have the right length. As a parent data of a random `RtreemixModel` object the suitable labels of events present in the model components can be specified although the `Sample` slot is an empty `matrix`.

Patients `signature(object = "RtreemixData")`: A method for obtaining the IDs of the patients.

Patients<- `signature(object = "RtreemixData")`: A method for replacing the IDs of the patients in the data object. It checks to be sure the values have the right length.

Sample `signature(object = "RtreemixData")`: A method for obtaining the matrix of observations.

eventsNum `signature(object = "RtreemixData")`: A method for obtaining the number of genetic events.

sampleSize `signature(object = "RtreemixData")`: A method for obtaining the size of the sample (the number of patients).

Author(s)

Jasmina Bogojeska

See Also

`RtreemixGPS-class`, `RtreemixStats-class`, `RtreemixModel-class`, `fit-methods`, `bootstrap-methods`

Examples

```
## Create an RtreemixData object from a file given in the examples directory of the packa
data1 <- new("RtreemixData", File = paste(system.file(package = "Rtreemix"), "/examples/t
show(data1) ## show the RtreemixData object

## Create an RtreemixData object from a randomly generated RtreemixModel object.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
data2 <- sim(model = rand.mod, no.draws = 300)
show(data2)

## Create an RtreemixData object from a given binary matrix.
bin.mat <- cbind(c(1, 0, 0, 1, 1), c(0, 1, 0, 0, 1), c(1, 1, 0, 1, 0))
data3 <- new("RtreemixData", Sample = bin.mat, Events = c("0", "1", "2",
"3"))
show(data3)
```

RtreemixGPS-class *Class "RtreemixGPS"*

Description

A class for describing the genetic progression scores (GPS) of a given set of patterns resulting from a waiting time simulation along the edges of the tree components of a given mutagenetic trees mixture model. It also contains GPS confidence intervals derived with the bootstrap method.

Objects from the Class

Objects can be created by calls of the form `new("RtreemixGPS", Data, Model, SamplingMode, SamplingParam, GPS, gpsCI)`. The `RtreemixGPS` class contains the GPS values each assigned to the corresponding pattern from the dataset given by `Data` (the parent class). The GPS values are derived in a waiting time simulation for a specified sampling mode and its corresponding sampling parameter. Moreover, this class specifies the confidence intervals for the GPS values derived with the bootstrap method.

The `Data` is an `RtreemixData` object that specifies the patterns for which the GPS values are calculated.

The `Model` is an `RtreemixModel` object that specifies the mutagenetic trees mixture model used for deriving the GPS values.

The `SamplingMode` is a `character` that specifies the sampling mode ("constant" or "exponential") used in the waiting time simulations.

The `SamplingParam` is a `numeric` that specifies the sampling parameter corresponding to the sampling mode given by `SamplingMode`.

The `GPS` is a numeric `vector` that specifies the GPS value of each pattern in the given dataset `Data`. Its length equals the number of patterns in `Data`.

The `gpsCI` is a numeric `matrix` that specifies the confidence intervals for each GPS value in the vector `GPS`. The number of rows equals the number of patients in `Data` and the number of columns equals 2. The first column gives the lower bound and the second column gives the upper bound of each confidence interval.

Slots

`Model`**:** Object of class `"RtreemixModel"`.

`SamplingMode`**:** Object of class `"character"`. It can have one of the two possible values: "constant" or "exponential".

`SamplingParam`**:** Object of class `"numeric"`.

`GPS`**:** Object of class `"numeric"`. The length of `GPS` must be equal to the number of patterns in the parent `RtreemixData` object.

`gpsCI`**:** Object of class `"matrix"`. It number of columns has to be 2 and the number of rows has to be equal to the length of `GPS`.

Extends

Class `"RtreemixData"`, directly.

Methods

GPS `signature(object = "RtreemixGPS")`: A method for obtaining the GPS values corresponding to the patterns in the parent `RtreemixData` object.

Model `signature(object = "RtreemixGPS")`: A method for obtaining the model used for deriving the GPS values.

SamplingMode `signature(object = "RtreemixGPS")`: A method for obtaining the sampling mode ("constant" or "exponential") used for the waiting time simulations.

SamplingParam `signature(object = "RtreemixGPS")`: A method for obtaining the sampling parameter corresponding to the specified `SamplingMode`.

getData `signature(object = "RtreemixGPS")`: A method for obtaining the set of patterns for which the GPS values are calculated.

gpsCI `signature(object = "RtreemixGPS")`: A method for obtaining the GPS confidence intervals.

Note

The GPS examples are time consuming. They are commented out because of the time restrictions of the check of the package. For trying out the code please copy it and uncomment it.

Author(s)

Jasmina Bogojeska

References

Estimating cancer survival and clinical outcome based on genetic tumor progression scores, J. Rahnenf\"urer et al.

See Also

`RtreemixData-class`, `RtreemixModel-class`, `gps-methods`, `fit-methods`, `confIntGPS-methods`

Examples

```
## Generate a random RtreemixModel object with 3 components and 9 genetic events.
#mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
#show(mod)
## Generate an artificial dataset from the model mod.
#data <- sim(model = mod, no.draws = 300)
#show(data)

## Create an RtreemixGPS object by calculating the GPS for all possible patterns.
#modGPS.all <- gps(model = mod, no.sim = 1000)
#show(modGPS.all)
## Create an RtreemixGPS object by calculating the GPS for the data based on the model mo
#modGPS <- gps(model = mod, data = data, no.sim = 1000)
#show(modGPS)

## See the slots from the RtreemixGPS object.
#Model(modGPS)
#SamplingMode(modGPS)
#SamplingParam(modGPS)
```

```
#GPS(modGPS)
## See data.
#getData(modGPS)

## Create an RtreemixGPS object by calculating GPS values for a given dataset
## and their 95% confidence intervals using the bootstrap method.
#modGPS2 <- confIntGPS(data = data, K = 2, B = 10)
#show(modGPS2)

## See the GPS values for the object modGPS2 and their confidence intervals.
#GPS(modGPS2)
#gpsCI(modGPS2)
```

`RtreemixModel-class`

Class "RtreemixModel"

Description

This class contains all the data needed for characterizing the mutagenetic trees mixture model (mixture parameters, mixture components, ...). The tree components of the model are given as a list of directed `graphNEL` objects.

Objects from the Class

Objects can be created by calls of the form `new("RtreemixModel", ParentData, Weights, WeightsCI, Resp, CompleteMat, Star, Trees)`. The `RtreemixModel` class extends the `RtreemixData` class and specifies the mutagenetic trees mixture model. If the model is not randomly generated the parent class gives the `RtreemixData` used for learning the mixture model. The directed trees that build up the model are represented as a list of directed `graphNEL` objects, and their weights (the mixture parameters) are given as a numeric vector. This class can also contain other useful information connected with the mixture model like confidence intervals for the mixture parameters and the edge weights (resulting from a bootstrap analysis), an indicator for the presence of the star component, etc. They are all listed in the text below with brief descriptions.

The `ParentData` is an `RtreemixData` object that specifies the data used for estimating the mutagenetic trees mixture model. It is not specified for random mixture models, since they are not estimated from a given dataset but generated randomly.

The `Weights` is a numeric `vector` that contains the mixture parameters of the model. Its length equals the length of the `list` of tree components `Trees`.

The `WeightsCI` is a named `list` with length equal to the length of the `Weights`. Each list element is a numeric `vector` of length two specifying the lower and upper bound of the confidence interval for the corresponding mixture parametar. The confidence intervals are derived using the bootstrap method.

The `Resp` is a numeric `matrix` that specifies the responsibility of each tree component to generate each of the patterns in the `ParentData`. The number of rows in `Resp` is equal to the number of trees in the mixture (the length of the list `Trees`) and the number of columns equals the number of patients in `ParentData`. For random mixture models it is an empty matrix, since they are not estimated from a given dataset.

The `CompleteMat` is a binary `matrix` that specifies the complete data in case some measurements for some patients are missing in the data used for learning the model (the `ParentData`).

It has the same size as the matrix specifying the data in ParentData. The missing data are estimated in the EM-algorithm used for fitting the mixture model. When there are no missing data in ParentData, or the model is randomly generated the CompleteMat is an empty matrix.

The Star is an indicator of the presence of a noise (star) component and is mostly relevant for models with a single tree component, since it is assumed that mixture models with at least two components always have the noise as a first component. It is of type logical.

The Trees is a list of directed graphNEL objects, each for every tree component in the mixture model. The genetic events are represented as nodes in the graphs. The edgeData of each tree can have two attributes: "weight" and "ci". Please see the help page on graph-class and graphNEL-class in the package graph. The "weight" attribute is for edge weight, i.e. the conditional probability that the child event of the edge occured given that the parent event already occured. The "ci" attribute is for the bootstrap confidence intervals for the edge weight (a numeric vector with length two).

Slots

Weights: Object of class "numeric". The length of the Weights must be equal to the length of Trees.

WeightsCI: Object of class "list". The length of the WeightsCI must be equal to the length of Weights.

Resp: Object of class "matrix". The number of rows of Resp must be identical to the length of Trees, and its number of columns to the number of patients in the dataset used for estimating the mixture model (ParentData).

CompleteMat: Object of class "matrix". When specified (when there are missing data) the size of the CompleteMat must be equal to the size of the matrix used to estimate the model.

Star: Object of class "logical".

Trees: Object of class "list". The length of Trees equals the length of Weights.

Extends

Class "RtreemixData", directly.

Methods

CompleteMat signature(object = "RtreemixModel"): A method used for obtaining the complete dataset, in case there were any missing measurements for some patients in the dataset used to learn the mixture model.

Resp signature(object = "RtreemixModel"): A method for obtaining the matrix of responsibilities for the trees to generate each of the samples in the dataset used for learning the model (ParentData).

Star signature(object = "RtreemixModel"): A method for checking the presence of a noise component in the mixture model (informative only for models with one tree component).

Trees signature(object = "RtreemixModel"): A method for obtaining the tree components of the mixture model as a list of directed graphNEL objects.

Weights signature(object = "RtreemixModel"): A method for obtaining the mixture parameters (the weights of the trees in the model).

Weights<- signature(object = "RtreemixModel"): A method for replacing the value of the slot Weights with a specified numeric vector. The components of this vector have to sum up to one.

WeightsCI `signature(object = "RtreemixModel")`: A method for obtaining the weights of the mixture parameters.

getData `signature(object = "RtreemixModel")`: A method for obtaining the `ParentData` of the mixture model, i.e. the data used for learning the model.

getTree `signature(object = "RtreemixModel", k = "numeric")`: A method for obtaining the k-th tree component of the mixture model as a directed `graphNEL` object.

numTrees `signature(object = "RtreemixModel")`: A method for obtaining the number of tree components building up the mixture model.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

RtreemixGPS-class, RtreemixStats-class, RtreemixData-class, RtreemixSim-class, fit-methods, bootstrap-methods, generate-methods, comp.models, comp.trees

Examples

```
## Generate a random RtreemixModel object with 2 components.
rand.mod <- generate(K = 2, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
show(rand.mod)
plot(rand.mod) ## plot the tree components of the model
plot(rand.mod, k = 2) ## plot the second component of the model

## Draw data from a specified mixture model.
draws <- sim(model = rand.mod, no.draws = 200)
show(draws)

## Create an RtreemixModel object by fitting model to the drawn data.
mod <- fit(data = draws, K = 2, equal.edgeweights = TRUE, noise = TRUE)
show(mod)

## See the values of the slots of the RtreemixModel object.
Weights(mod)
Resp(mod)
CompleteMat(mod)
Star(mod)
Trees(mod)
## See data used for learning the model.
getData(mod)
## See the number of tree components in the mixture model.
numTrees(mod)
## See a specific tree component k.
getTree(object = mod, k = 2)
## See the conditional probabilities assigned to edges of the second tree component.
edgeData(getTree(object = mod, k = 2), attr = "weight")
## See the probability distribution encoded by the model on the set of all possible patte
distr <- distribution(model = mod)
distr
```

```
## Get the probabilities.
distr$probability
## See the probability distribution encoded by the model on the set of all possible patte
## calculated for given sampling mode, and input and output parameters.
distr1 <- distribution(model = mod, sampling.mode = "exponential", sampling.param = 1, ou
distr1

## Create a RtreemixModel and analyze its variance with the bootstrap method.
mod.boot <- bootstrap(data = draws, K = 2, equal.edgeweights = TRUE, B = 100)

## See the confidence intervals for the mixture parameters (the weights).
WeightsCI(mod.boot)
## See the confidence intervals of the conditional probabilities assigned to the edges.
edgeData(getTree(mod.boot, 2), attr = "ci")
```

`RtreemixSim-class` *Class "RtreemixSim"*

Description

This class contains data simulated from the `RtreemixModel` it extends together with their sampling and waiting times. It also includes the sampling mode and the sampling parameter used for the time simulation.

Objects from the Class

Objects can be created by calls of the form `new("RtreemixSim", Model, SimPatterns, SamplingMode, SamplingParam, WaitingTimes, SamplingTimes)`. The `RtreemixSim` class specifies patterns (`RtreemixData`) simulated from the parent `RtreemixModel` together with their waiting and sampling times resulting from the waiting time simulation along the branchings in the parent model.

The `Model` is an `RtreemixModel` object used in the data and time simulation process. In other words, this model is used for simulating patterns with their sampling and waiting times.

The `SimPatterns` is an `RtreemixData` object that contains the patterns simulated from the given `Model`.

The `SamplingMode` is a `character` that specifies the sampling mode ("constant" or "exponential") used in the time simulations.

The `SamplingParam` is a `numeric` that specifies the sampling parameter corresponding to the sampling mode given by `SamplingMode`.

The `WaitingTimes` is a numeric `vector` that specifies the waiting times for the simulated patterns. Its length equals the number of patterns in `SimPatterns`.

The `SamplingTimes` is a numeric `vector` that specifies the sampling times for the simulated patterns. Its length equals the number of patterns in `SimPatterns`.

Slots

`SimPatterns`**:** Object of class `"RtreemixData"`.

`SamplingMode`**:** Object of class `"character"`. It can have one of the two possible values: "constant" or "exponential".

`SamplingParam`**:** Object of class `"numeric"`.

`WaitingTimes`: Object of class `"numeric"`. The length of `WaitingTimes` must be equal to the number of patterns in `SimPatterns`.

`SamplingTimes`: Object of class `"numeric"`. The length of `SamplingTimes` must be equal to the number of patterns in `SimPatterns`.

Extends

Class `"RtreemixModel"`, directly. Class `"RtreemixData"`, by class "RtreemixModel", distance 2.

Methods

SamplingMode `signature(object = "RtreemixSim")`: A method for obtaining the sampling mode ("constant" or "exponential") used for the time simulations.

SamplingParam `signature(object = "RtreemixSim")`: A method for obtaining the sampling parameter corresponding to the specified `SamplingMode`.

SamplingTimes `signature(object = "RtreemixSim")`: A method used for obtaining the sampling times of the patterns in `SimPatterns`.

SimPatterns `signature(object = "RtreemixSim")`: A method used for obtaining the patterns simulated from the parent model.

WaitingTimes `signature(object = "RtreemixSim")`: A method used for obtaining the waiting times of the patterns in `SimPatterns`.

getModel `signature(object = "RtreemixSim")`: A method for obtaining the mixture model used in the simulations.

noDraws `signature(object = "RtreemixSim")`: A method for obtaining the number of simulated patterns, i.e. the size of `SimPatterns`.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

`RtreemixGPS-class`, `RtreemixData-class`, `RtreemixModel-class`, `fit-methods`, `sim-methods`

Examples

```
## Generate a random RtreemixModel object with 3 components and 9 genetic events.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
show(rand.mod)

## Create an RtreemixSim object by simulating patterns with their sampling and waiting ti
sim.data <- sim(model = rand.mod, sampling.mode = "exponential", sampling.param = 1, no.s
show(sim.data)

## See the slots from the RtreemixSim object.
SimPatterns(sim.data)
SamplingMode(sim.data)
```

```
SamplingParam(sim.data)
WaitingTimes(sim.data)
SamplingTimes(sim.data)
## See model.
getModel(sim.data)
## See number of simulated patterns.
noDraws(sim.data)
```

RtreemixStats-class

Class "RtreemixStats"

Description

The RtreemixStats class contains the (weighted, log) likelihoods for a given dataset (specified by the parent class) derived from the probability distribution induced by an underlying mutagenetic trees mixture model.

Objects from the Class

Objects can be created by calls of the form new("RtreemixStats", Data, Model, LogLikelihoods, WLikelihoods). The class RtreemixStats extends the RtreemixData class and specifies (log, weighted) likelihoods for these data derived from a given RtreemixModel. The number of the genetic events in the patterns from the given dataset (Data) has to be equal to the number of genetic events in the branchings from the mixture model given by the slot Model. When having the weighted likelihoods, one can easily derive the responsibilities of the model components in Model for generating the patterns in the specified dataset (Data).

The Data is an RtreemixData object that specifies the patterns for which the likelihoods are calculated.

The Model is an RtreemixModel object that specifies the mutagenetic trees mixture model used for deriving the likelihoods of the given data.

The LogLikelihoods is a numeric vector that contains the log-likelihoods of the patterns in Data. Its length equals the sample size, i.e. the number of patients in Data.

The WLikelihoods is a numeric matrix that specifies the weighted likelihoods of each pattern in the given dataset Data. The number of columns in WLikelihoods equals the number of tree components in Model and the number of rows equals the number of patients in Data.

Slots

Model: Object of class "RtreemixModel".

LogLikelihoods: Object of class "numeric". The length of LogLikelihoods must be equal to the number of patients of the dataset specified with the parent "RtreemixData" class.

WLikelihoods: Object of class "matrix". The number of rows must be equal to the sample size of the dataset specified with the parent "RtreemixData" class. The number of columns must be identical with the number of tree components in the mixture model Model.

Extends

Class "RtreemixData", directly.

Methods

LogLikelihoods `signature(object = "RtreemixStats")`: A method for obtaining the log-likelihoods of the patterns in the dataset specified with the parent `"RtreemixData"` class.

Model `signature(object = "RtreemixStats")`: A method for obtaining the mutagenetic trees mixture model used for deriving the likelihoods.

WLikelihoods `signature(object = "RtreemixStats")`: A method for obtaining the weighted likelihoods of the patterns in the dataset specified with the parent `"RtreemixData"` class.

getData `signature(object = "RtreemixStats")`: A method for obtaining the dataset specified with the parent `"RtreemixData"` class.

getResp `signature(object = "RtreemixStats")`: A method for computing the matrix of responsibilities for the trees to generate each of the samples in the parent dataset from their weighted likelihoods `WLikelihoods`.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

`RtreemixData-class`, `RtreemixModel-class`, `fit-methods`, `likelihoods-methods`

Examples

```
## Generate a random RtreemixModel object with 3 components and 9 genetic events.
mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
show(mod)

## Draw a data sample from the model mod.
data <- sim(model = mod, no.draws = 400)

## Create an RtreemixStats object.
mod.stat <- likelihoods(model = mod, data = data)
show(mod.stat)

## See the slots from the RtreemixStats object.
Model(mod.stat)
LogLikelihoods(mod.stat)
WLikelihoods(mod.stat)
## See data.
getData(mod.stat)
## Calculate the responsibilities from the weighted likelihoods.
getResp(mod.stat)
```

bootstrap-methods *Method for fitting a mutagenetic trees mixture model and analyzing its*

Description

This method fits an RtreemixModel to a given dataset and then analyzes its variance with the bootstrap method. The data and the number of trees K have to be specified.

Usage

```
bootstrap(data, K, ...)
```

Arguments

data	An RtreemixData object giving the dataset used for learning the trees mixture model.
K	An integer larger than 0 specifying the number of branchings in the mixture model.
...	no.start.sol is an integer larger than 0 specifying the number of starting solutions for the k-means algorithm. The default value is 100. eps is a numeric giving the minimum conditional probability to include edge. The default value is 0. weighing is a logical specifying whether to use special weights log(Pr(v)) for the edges (root, v). The default value is FALSE. equal.edgeweights is a logical specifying whether to use equal edge weights in the noise component. The default value is TRUE. When you have few data samples always use its default value (TRUE) to ensure nonzero probabilities for all possible patterns (sets of events). seed is a positive integer specifying the random generator seed. The default value is (-1) and then the time is used as a random generator. B is an integer larger than 0 specifying the number of bootstrap samples. Its default value is 1000. conf.interval is a numeric specifying the Confidence level for the intervals. Its default value is 0.05.

Value

The function returns an object from the class RtreemixModel. This is the mixture model learned on the given data. Besides the edge weights it also contains their confidence intervals resulting from the bootstrap analysis. Confidence intervals for the mixture parameters are also comupted and available.

Note

The bootstrap examples are time consuming. They are commented out because of the time restrictions of the check of the package. For trying out the code please copy it and uncomment it.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

RtreemixData-class, RtreemixModel-class, fit-methods

Examples

```
## Create an RtreemixData object from a randomly generated RtreemixModel object.
#rand.mod <- generate(K = 2, no.events = 7, noise.tree = TRUE, prob = c(0.2, 0.8))
#data <- sim(model = rand.mod, no.draws = 300)

## Create a RtreemixModel and analyze its variance with the bootstrap method.
#mod.boot <- bootstrap(data = data, K = 2, equal.edgeweights = TRUE, B = 10) ## time cons

## See the confidence intervals for the mixture parameters (the weights).
#WeightsCI(mod.boot)
## See the confidence intervals of the conditional probabilities assigned to the edges.
#edgeData(getTree(mod.boot, 2), attr = "ci")
```

Models *Functions for comparing the tree topologies of two mutagenetic trees*

Description

These functions implement a similarity measure for comparing the topologies of the trees of two mixture models `mixture1` and `mixture2`. `comp.models` chaaracterizes the similarity of the models based on sum of the number of different edges of matched tree components (similarity pairs). `comp.models.levels` quantifies the similarity of two mixture models by adding to the edge ddifference of each similarity pair in the previously described sum the L1 distance of the level vectors of the trees comprising the pair. A level vector can be associated to each tree component and denotes the depth of each of the genetic events in the tree. It is necessary that the two models have the same number of tree components build on the same number of genetic events. It is assumed that the mixtures have at least two tree components.

Usage

```
comp.models(mixture1, mixture2)
comp.models.levels(mixture1, mixture2)
```

Arguments

`mixture1`	An `RtreemixModel` object specifying the first component for the similarity calculation.
`mixture2`	An `RtreemixModel` object specifying the second component for the similarity calculation. The number of tree components equals the one of `mixture1`.

Details

The value returned by the function `comp.models` is between 0 (no similarity) and 1 (identical models).

Value

The functions return a numeric value that quantifies the similarity of the tree topologies of two mixture models.

Author(s)

Jasmina Bogojeska

See Also

RtreemixModel-class, comp.trees, fit-methods, stability.sim

Examples

```
## Generate two random RtreemixModel objects each with 3 components.
rand.mod1 <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob =
c(0.2, 0.8))
rand.mod2 <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob =
c(0.2, 0.8))

## Compare the topologies of the tree components of the two randomly
## generated models
comp.models(rand.mod1, rand.mod2)
comp.models.levels(rand.mod1, rand.mod2)
```

---

comp.trees *Functions for quantifying the diversity of the nontrivial trees in a*

Description

These functions implement a similarity measure for comparing the topologies of the nontrivial tree components of a specified mixture model, and thereby quantifying their diversity. All possible pairs of nontrivial components are considered when computing the similarity. comp.trees uses the sum of the number of different edges of all pairs for caracterizing the difference of the trees in the model. comp.trees.levels uses the sum of the number of different edges of all pairs and the corresponding L1 distances of their level vectors. The model must have at least two nontrivial components.

Usage

```
comp.trees(model)
comp.trees.levels(model)
```

Arguments

model	An RtreemixModel object.

Value

The functions return a numeric value that quantifies the similarity (or diversity) of the nontrivial tree topologies of a given mixture models.

Author(s)

Jasmina Bogojeska

See Also

RtreemixModel-class, comp.models, fit-methods, stability.sim

Examples

```
## Generate two random RtreemixModel objects each with 3 components.
mix1 <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
mix2 <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
## Inspect the diversity of the nontrivial tree components in a given model
## using the number of distinct edges and the levels of the events in
## the treesas dissimilarity measure.
comp.trees.levels(model = mix1)
comp.trees.levels(model = mix2)
```

---

confIntGPS-methods *Method for calculating GPS values and their 95% bootstrap confidence*

Description

The method first calculates the genetic progression score (GPS) for the patterns in a given dataset `data` based on a fitted mutagenetic trees mixture model with `K` components. The `data` and `K` have to be specified. Then, it derives a 95% confidence intervals for the GPS values with bootstrap analysis.

Usage

```
confIntGPS(data, K, ...)
```

Arguments

`data`	An `RtreemixData` object containing the samples (patterns of genetic events) for which the GPS values and their bootstrap confidence intervals are to be calculated. The number of genetic events should NOT be greater than 20.
`K`	An `integer` larger than 0 specifying the number of branchings in the mixture model.
`...`	`sampling.mode` is a `character` that specifies the sampling mode ("constant" or "exponential") used in the waiting time simulations. Its default value is "exponential". `sampling.param` is a `numeric` that specifies the sampling parameter corresponding to the sampling mode given by `sampling.mode`. Its default value is 1. `no.sim` is an `integer` larger than 0 giving the number of iterations for the waiting time simulation. Its default values is 10000. `B` is an `integer` larger than 0 specifying the number of bootstrap samples used in the bootstrap analysis. Its default value is 1000. `equal.star` is a `logical` specifying whether to use equal edge weights in the noise component. The default value is `TRUE`. When you have few data samples always use its default value (`TRUE`) to ensure nonzero probabilities for all possible patterns (sets of events).

Value

The function returns an object from the `RtreemixGPS` class that containes the calculated GPS values, their 95% confidence intervals, the model used for the computation, the data, and so on (see `RtreemixGPS-class`). The GPS values are represented as a `numeric` vector with length equal to the number of samples in `data`. Their corresponding confidence intervals are given in a matrix with two columns.

Note

The data for which the GPS values and their corresponding confidence intervals are to be calculated should not have more than 20 genetic events. The reason for this is that the number of all possible patterns for which the GPS values are calculated during a computationally intensive simulations is in this case 2^20. This demands too much memory. The GPS examples are time consuming. They are commented out because of the time restrictions of the check of the package. For trying out the code please copy it and uncomment it.

Author(s)

Jasmina Bogojeska

See Also

`RtreemixGPS-class`, `gps-methods`, `RtreemixData-class`, `RtreemixModel-class`, `fit-methods`

Examples

```
## Create an RtreemixData object from a randomly generated RtreemixModel object.
#rand.mod <- generate(K = 2, no.events = 7, noise.tree = TRUE, prob = c(0.2, 0.8))
#data <- sim(model = rand.mod, no.draws = 400)

## Create an RtreemixGPS object by calculating GPS values for a given dataset
## and their 95% confidence intervals using the bootstrap method.
#modGPS2 <- confIntGPS(data = data, K = 2, B = 100) ## time consuming computation
#show(modGPS2)

## See the GPS values for the object modGPS2 and their confidence intervals.
#GPS(modGPS2)
#gpsCI(modGPS2)

## See data.
#getData(modGPS2)
```

`distribution-methods`

Method for generating the (scaled) probablility distribution induced

---

Description

These functions generate the probability distribution induced with a given mutagenetic trees mixture model `model` on the space of all possible patterns of genetic events. The `model` has to be specified. The sampling mode and the parameters for the sampling times of the observed input and output probabilities are optional. The number of genetic events in the `model` cannot exceed 30.

Usage

```
distribution(model, sampling.mode, sampling.param, output.param)
```

Arguments

`model`	An `RtreemixModel` object that encodes a probability distribution on the set of all possible patterns.
`sampling.mode`	A `character` that specifies the sampling mode ("constant" or "exponential") for the observed input and output probabilities.
`sampling.param`	A `numeric` that specifies the sampling parameter for the observed input probabilities corresponding to the sampling mode given by `sampling.mode`.
`output.param`	A `numeric` that specifies the sampling parameter for the observed output probabilities corresponding to the sampling mode given by `sampling.mode`.

Value

The function returns a `dataframe` of all possible patterns with their corresponding probabilities derived from the specified trees mixture model. When the sampling mode and the sampling parameters (input and output) are specified their values are printed out.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

`RtreemixModel-class`, `fit-methods`

Examples

```
## Generate a random RtreemixModel object with 3 components.
mod <- generate(K = 3, no.events = 8, noise.tree = TRUE, prob = c(0.2, 0.8))
show(mod)

## See the probability distribution encoded by the model on the set of all possible patte
distr <- distribution(model = mod)
distr

## Get the probabilities.
distr$probability

## See the probability distribution encoded by the model on the set of all possible patte
## calculated for given sampling mode, and corresponding input and output parameters.
distr1 <- distribution(model = mod, sampling.mode = "exponential", sampling.param = 1, ou
distr1
```

`fit-methods` *Method for fitting mutagenetic trees mixture model to a given dataset*

Description

Function for fitting a mutagenetic trees mixture model to a given dataset `data`. The dataset and the number of trees `K` have to be specified. The function estimates K-oncogenetic trees mixture model from the specified data by using an EM-like learning algorithm. The first tree component of the model has a star topology and is referred to as the noise component.

Usage

```
fit(data, K, ...)
```

Arguments

`data`	An `RtreemixData` object giving the dataset used for learning the trees mixture model.
`K`	An `integer` larger than 0 specifying the number of branchings in the mixture model.
`...`	`no.start.sol` is an `integer` larger than 0 specifying the number of starting solutions for the k-means algorithm. The default value is 100. `eps` is a `numeric` giving the minimum conditional probability to include edge. The default value is 0.01. `weighing` is a `logical` specifying whether to use special weights log(Pr(v)) for the edges (root, v). The default value is `FALSE`. `equal.edgeweights` is a `logical` specifying whether to use equal edge weights in the noise component. The default value is `TRUE`. When you have few data samples always use its default value (`TRUE`) to ensure nonzero probabilities for all possible patterns (sets of events). `seed` is a positive `integer` specifying the random generator seed. The default value is (-1) and then the time is used as a random generator. `noise` is a `logical` indicating the presence of a noise (star) component in the fitted mixture model. It is mostly relevant for models with a single tree component, since it is assumed that mixture models with at least two components always have the noise as a first component.

Details

When K = 1 and noise = FALSE a single mutagenetic tree is fit to the data. When K = 1 and noise = TRUE a star mutagenetic tree is fit to the data. If K > 1 the first mutagenetic tree is always the star, i.e. the case K > 1 and noise = FALSE is not possible.

Value

The method returns an `RtreemixModel` object that represents the K-trees mixture model learned from the given dataset.

Note

When you have too few data samples always use the default value `TRUE` for the `equal.edgeweights`. Like this you make sure that all possible patterns (sets of events) have non-zero probabilities. If they don't the fitting procedure will not be completed and you will get an error!

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

RtreemixData-class, RtreemixModel-class, generate-methods, bootstrap-methods, confIntGPS-methods

Examples

```
## Create an RtreemixData object from a randomly generated RtreemixModel object.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
data <- sim(model = rand.mod, no.draws = 300)
show(data)

## Create an RtreemixModel object by fitting model to the given data.
mod <- fit(data = data, K = 3, equal.edgeweights = TRUE, noise = TRUE)
show(mod)
## See the number of tree components in the mixture model.
numTrees(mod)
## See the weights of the branchings from the fitted mixture model.
Weights(mod)
## See a specific tree component k.
getTree(object = mod, k = 2)
```

---

generate-methods *Method for generating a random mutagenetic trees mixture model*

Description

Function for generating a random mutagenetic mixture model. Each tree component from the model is drawn uniformly at random from the tree topology space by using the Pr\"ufer encoding of trees. The number of tree components and the number of genetic events have to be specified.

Usage

```
generate(K, no.events, ...)
```

Arguments

K	An integer larger than 0 specifying the number of branchings in the mixture model.
no.events	An integer larger than 0 specifying the number of genetic events in the mixture model.

... noise.tree is a logical indicating the presence of a noise (star) component in the random mixture model. The default value is TRUE. equal.edgeweights is a logical specifying whether to use equal edge weights in the noise component. The default value is TRUE. prob is a numeric vector of length 2 specifying the boundaries for the edge weights of the randomly generated trees. The first component of the vector (the lower boundary) must be smaller than the second component (the upper boundary). The default value is (0.0, 1.0). seed is a positive integer specifying the random generator seed. The default value is (-1) and then the time is used as a random generator.

Value

The method returns an RtreemixModel object that represents the randomly generated K-trees mixture model.

Author(s)

Jasmina Bogojeska

References

Beweis eines Satzes \"uber Permutationen, H. Pr\"ufer; Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.; Model Selection for Mixtures of Mutagenetic Trees, Yin et al.

See Also

RtreemixModel-class

Examples

```
## Generate a random RtreemixModel object with 3 components and 9 genetic events.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
show(rand.mod)
```

---

get.tree.levels *Function for constructing level vectors*

Description

Function that assignes to each node the level at which that node is in a specific tree (tree.num) of the mutagenetic trees mixture model mixture. The start.val is the number assigned to the events pruned from the tree. This usually is the maximum depth of the tree with which the tree specified with tree.num will be compared.

Usage

```
get.tree.levels(mixture, tree.num, start.val)
```

Arguments

mixture	An object of the class RtreemixModel.
tree.num	A numeric specifying the tree component from mixture used for creating the level vector.
start.val	A numeric specifying the number assigned to the pruned events.

Value

The function returns a named numeric vector. Its length equals the number of genetic events in mixture minus one (for the initial null event which is always on level 0). The vector names correspond to the names of the genetic events and each vector component gives the level at which the respective event is in the num.tree tree of mixture.

Author(s)

Jasmina Bogojeska

See Also

comp.models, comp.trees, stability.sim, RtreemixModel-class, fit-methods

Examples

```
## Generate two random RtreemixModel objects each with 3 components.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob =
c(0.2, 0.8))

## Get the tree levels of the 2nd component of the model rand.mod.
get.tree.levels(mixture = rand.mod, tree.num = 2, start.val = 10)
```

---

gps-methods *Methods for predicting the GPS of given dataset by using a given*

Description

These functions compute the genetic progression score (GPS) of each sample in the given data by performing a waiting time simulation along the branchings of the mixture model model. The model has to be specified. If a dataset is missing a GPS for all possible patterns is calculated. The number of events of the samples in data equals the number of genetic events in the model.

Usage

```
gps(model, data, ...)
```

Arguments

model	An object of the class RtreemixModel specifying the mutagenetic trees mixture model used for deriving the GPS values. The model should NOT have more than 20 genetic events.
data	An RtreemixData object or a 0-1 matrix containing the samples (patterns of genetic events) for which the GPS values are to be calculated. The length of each of them has to be equal to the number of genetic events in the model.

`...` `sampling.mode` is a `character` that specifies the sampling mode ("constant" or "exponential") used in the waiting time simulations. Its default value is "exponential". `sampling.param` is a `numeric` that specifies the sampling parameter corresponding to the sampling mode given by `sampling.mode`. Its default value is 1. `no.sim` is an `integer` larger than 0 giving the number of iterations for the waiting time simulations. Its default value is 10. `seed` is a positive `integer` specifying the random generator seed. Its default value is (-1) and then the time is used as a random generator.

Value

The function returns an object from the `RtreemixGPS` class that containes the calculated GPS values, the model used for the computation, the data, and so on (see `RtreemixGPS-class`). The GPS values are represented as a `numeric` vector with length equal to the number of samples in `data`.

Methods

model = "RtreemixModel", data = "RtreemixData", ... A method for calculating the GPS values of the data given as `RtreemixData` object.

model = "RtreemixModel", data = "matrix", ... A method for calculating the GPS values of the data given as 0-1 `matrix`.

model = "RtreemixModel", data = "missing", ... A method for calculating the GPS values of the set of all possible patterns.

Note

The mixture model used for deriving the GPS values should not have more than 20 genetic events. The reason for this is that the number of all possible patterns for which the GPS values are calculated during a computationally intensive simulations is in this case 2^20. This demands too much memory. The GPS examples are time consuming. They are commented out because of the time restrictions of the check of the package. For trying out the code please copy it and uncomment it.

Author(s)

Jasmina Bogojeska

References

Estimating cancer survival and clinical outcome based on genetic tumor progression scores, J. Rahnenf\"urer et al.

See Also

`RtreemixGPS-class`, `RtreemixData-class`, `RtreemixModel-class`, `fit-methods`, `confIntGPS-methods`

Examples

```
## Create an RtreemixData object from a randomly generated RtreemixModel object.
#rand.mod <- generate(K = 2, no.events = 7, noise.tree = TRUE, prob = c(0.2, 0.8))
#data <- sim(model = rand.mod, no.draws = 400)

## Create an RtreemixModel object by fitting model to the given data.
```

```
#mod <- fit(data = data, K = 2, equal.edgeweights = TRUE, noise = TRUE)
#show(mod)

## Create an RtreemixGPS object by calculating the GPS for all possible patterns.
#modGPS.all <- gps(model = mod, no.sim = 1000) ## time consuming copmutations
#show(modGPS.all)

## See the GPS values for all possible data.
#GPS(modGPS.all) ## time consuming copmutations

## Create an RtreemixGPS object by calculating the GPS for the data based on the model mo
#modGPS <- gps(model = mod, data = data, no.sim = 1000)
#show(modGPS) ## time consuming copmutations

## See the GPS values for data.
#GPS(modGPS) ## time consuming copmutations
```

`hiv.data` *Example of an RtreemixData object*

Description

This data object was created by using the Stanford HIV Drug Resistance Database that comprises genetic measurements of 364 HIV patients treated only with the drug zidovudine. The data contains the six classical major zidovudine resistance mutations: M41L, D67N, K70R, L210W, T215F/Y, and K219E/Q.

Usage

```
data(hiv.data)
```

References

Human immunodeficiency virus reverse transcriptase and protease sequence database, S. Rhee et al.

See Also

`RtreemixData-class`

Examples

```
data(hiv.data)

## print the object
hiv.data
```

`kullback.leibler` *Kullback-Leibler divergence*

Description

A function for calculating the Kullback-Leibler divergence between two discrete probability distributions. The vectors specifying the probability distributions must have the same length.

Usage

```
kullback.leibler(p, q)
```

Arguments

`p`	A `numeric` vector specifying the the first probability distribution. It has to have the same length as `q`.
`q`	A `numeric` vector specifying the the second probability distribution.

Value

The function returns the Kullback-Leibler divergence between the two specified descrete probability distributions.

Warning

The function does not check whether the values in the vectors specifying the discrete probability distributions sum up to one.

Author(s)

Jasmina Bogojeska

See Also

`L1.dist`, `L2.norm`, `stability.sim`

Examples

```
## Define two discrete probability distributions with equal lengths.
p <- c(0.1, 0.2, 0.3, 0.4)
q <- c(0.2, 0.5, 0.1, 0.2)

## Calculate the Kullback-Leibler divergence
## between the probability distributions p and q
kullback.leibler(p, q)
```

`likelihoods-methods` *Method for predicting the likelihoods of a set of samples with respect*

Description

This function predicts the (log, weighted) likelihoods of the samples in a given dataset according to a given mutagenetic trees mixture model. The dataset and the model have to be specified.

Usage

```
## S4 method for signature 'RtreemixModel,RtreemixData'
likelihoods(model, data)
```

Arguments

`model`	An `RtreemixModel` object specifying the probabilistic framework in which the likelihoods of the genetic patterns are computed.
`data`	An `RtreemixData` object giving the samples for which the likelihoods are to be calculated.

Value

This method returns an `RtreemixStats` object that containes the weghted- and log-likelihoods of the samples in the given dataset with respect to the given mutagenetic trees mixture model.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

`RtreemixData-class`, `RtreemixModel-class`, `fit-methods`, `distribution-methods`

Examples

```
## Create an RtreemixData object from a randomly generated RtreemixModel object.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))
data <- sim(model = rand.mod, no.draws = 300)
show(data)

## Compute the likelihoods of the samples in data with respect to the model rand.mod
mod.stat <- likelihoods(model = rand.mod, data = data)
show(mod.stat)
```

plot-methods *Method for visualizing mutagenetic trees mixture models*

Description

Function for visualizing the tree components comprising a mutagenetic trees mixture model. The user can also specify the `fontSize` used for the text labels of the nodes and the edges of the plotted trees. Additionally, one can use the parameter `k` to plot a certain tree component from the mixture model.

Usage

```
plot(x, y, ...)
```

Arguments

`x`	An `RtreemixModel` object giving the mixture model that should be visualized.
`y`	Not specified.
`...`	`fontSize` is the size of the text labels of the nodes and the edges of the tree components. The default value is 8. `k` is a `numeric` giving the specific tree component from the given mixture model that should be plotted. Its value can be from one to the number of tree components in the given model.

Details

The value of `k` (that gives the tree component to be plotted) can take `"integer"` values from one to the number of tree components.

Value

The method returns a plot of the mixture model model.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

`RtreemixData-class`, `RtreemixModel-class`, `generate-methods`

Examples

```
## Generate a random RtreemixModel object.
rand.mod <- generate(K = 2, no.events = 7, noise.tree = TRUE, prob = c(0.2, 0.8))

## Visualize it.
plot(rand.mod)

## Increase the font size of the text labels in the plot.
plot(rand.mod, fontSize = 10)

## Plot the second component of the mixture model rand.mod
plot(rand.mod, k = 2)
```

---

sim-methods *Method for simulating data from a mutagenetic trees mixture model*

Description

This function draws a certain number of patterns from a specified mutagenetic trees mixture model. Thus, the mixture model has to be specified. When besides the mixture model also the sampling mode and its respective sampling parameter are specified, this function simulates patterns together with their waiting and sampling times from the respective model.

Usage

```
sim(model, sampling.mode, sampling.param, ...)
```

Arguments

`model`	An object of the class `RtreemixModel` specifying the mutagenetic trees mixture model used for drawing the patterns, or for simulating patterns with their sampling and waiting times.
`sampling.mode`	A `character` that specifies the sampling mode ("constant" or "exponential") used in the time simulations.
`sampling.param`	A `numeric` that specifies the sampling parameter corresponding to the sampling mode given by `sampling.mode`.
`...`	`no.draws` is an `integer` larger than zero specifying the number of patterns that should be drawn from the given mixture model. `no.sim` is an `integer` larger than 0 giving the number of iterations for the waiting time simulations. Its default value is 10. `seed` is a positive `integer` specifying the random generator seed. Its default value is (-1) and then the time is used as a random generator.

Value

The function returns an `RtreemixData` object in the case when one wants to draw a certain number of patterns from a given mixture model, i.e. when only the mutagenetic trees mixture model and the number of patterns to be drawn are specified. When besides the model also the sampling mode and the sampling parameter are given, the function returns an object from the `RtreemixSim` class where the simulated patterns together with their sampling and waiting times are stored.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.

See Also

RtreemixSim-class, RtreemixModel-class, RtreemixData-class

Examples

```
## Create a random RtreemixModel object with 3 branchings and 9 genetic events.
rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE, prob = c(0.2, 0.8))

## Draw 300 samples from the randomly generated model rand.mod
data <- sim(model = rand.mod, no.draws = 300)
show(data)

## Create an RtreemixSim object by simulating patterns with their sampling and waiting ti
sim.data <- sim(model = rand.mod, sampling.mode = "exponential", sampling.param = 1, no.s
show(sim.data)
WaitingTimes(sim.data)
SamplingTimes(sim.data)
```

---

stability.sim *Stability analysis of the mutagenetic trees mixture model*

Description

The function includes stability analysis on different levels of the mutagenetic trees mixture model: GPS values, encoded probability distribution, tree topologies. Each analysis contains the values of different similarity measures with their corresponding p-values.

Usage

```
stability.sim(no.trees = 3, no.events = 9, prob = c(0.2, 0.8),
              no.draws = 300, no.rands = 100, no.sim = 1)
```

Arguments

`no.trees`	An `integer` larger than 2 giving the number of tree components of the mixture models considered in the stability analysis. The default value is 3.
`no.events`	An `integer` larger than 0 giving the number of genetic events of the mixture models considered in the stability analysis.
`prob`	A `numeric` vector of length 2 specifying the boundaries for the edge weights of the randomly generated trees. The first component of the vector (the lower boundary) must be smaller than the second component (the upper boundary). The default value is (0.2, 0.8).
`no.draws`	An `integer` larger than 0 giving the size of the data sample drawn from the random models used for learning the mixture models. The default value is 300.

no.rands — An integer larger than 0 specifying the number of random models used for calculating the p-values. The default value is 100.

no.sim — An integer larger than 0 specifying the number of iterations used for the waiting time simulations (a part of the GPS calculation). The default value is 1.

Details

The stability analysis is performed by first drawing a true mixture model uniformly at random from the model space, and drawing a data sample from it. Afterwards, a mutagenetic trees model is fitted to the drawn sample. The quality of the features derived from the model is then assessed by comparing its quality with the quality of the corresponding features of a sufficient number of random mixture models sampled uniformly from the model space. A p-value is obtained as a percentage of cases in which the true model is closer to a random model tnah to the fitted model.

Value

comp1 — Results from the stability analysis of the GPS values derived from a fitted mixture model. A matrix with 4 columns and no.sim rows. The first two columns give the similarity values and their corresponding p-values when the Euclidian distance is used as a similarity measure for comparing the respective GPS vectors. The last two columns depict the same results, but with the rank correlation distance used as a similarity measure.

comp2 — Results from the stability analysis of the probability distributions induced by a fitted mixture model. A matrix with 6 columns and no.sim rows. Each two columns give the values of the comparissons between the true and the fitted probability distributions and their corresponding p-values, when using the cosine distance, the L1 distance, and the Kullback-Leibler divergence as similarity measures.

comp3 — Results from the stability analysis of the topologies of the tree components of a fitted mixture model. A matrix with 2 columns and no.sim rows that give the value of the comparisson of the topologies between the true and the corresponding fitted model and their p-values. The similarity measure underlying the number of different edges was used.

comp4 — Similar to comp3. However, the similarity measure for comparing the tree topologies besides the number of distinct edges includes the L1 distances of the level vectors of events. See get.tree.levels.

comp5 — A matrix where the columns correspond to the true GPS vector from each simulation iteration. The matrix has no.sim columns and no.draws rows.

comp6 — Same as comp5, but the matrix contains the fitted GPS values from each simulation iteration.

comp7 — A list where each component corresponds to the true models generated in each simulation iteration. the length of the list is no.sim.

comp8 — Same as comp7, but the list contains the fitted models.

Note

The stability simulation examples are time consuming. They are commented out because of the time restrictions of the check of the package. For trying out the code please copy it and uncomment it.

Author(s)

Jasmina Bogojeska

References

Learning multiple evolutionary pathways from cross-sectional data, N. Beerenwinkel et al.; Estimating cancer survival and clinical outcome based on genetic tumor progression scores, J. Rahnenf\"urer et al.

See Also

RtreemixData-class, RtreemixModel-class, RtreemixGPS-class, RtreemixStats-class, fit-methods, gps-methods, distribution-methods, generate-methods, sim-methods, L1.dist, Pval.dist, comp.models, comp.trees, get.tree.levels, kullback.leibler

Examples

```
## Stability analysis - a toy example
#stability.sim(no.trees = 3, no.rands = 5, no.sim = 4, no.draws = 300)
```

Index