Package ‘PhylogeneticEM’

February 15, 2025

Title Automatic Shift Detection using a Phylogenetic EM

Version 1.8.0

Description Implementation of the automatic shift detection method for Brownian Motion (BM) or Ornstein–Uhlenbeck (OU) models of trait evolution on phylogenies. Some tools to handle equivalent shifts configurations are also available. See Bastide et al. (2017) <doi:10.1111/rssb.12206> and Bastide et al. (2018) <doi:10.1093/sysbio/syy005>.

Depends ape (>= 5.3), Matrix (>= 1.2.18), R (>= 3.6.0),

Imports capushe (>= 1.1.1), foreach (>= 1.4.3), gglasso (>= 1.4), glmnet (>= 2.0.5), graphics (>= 3.6.0), grDevices (>= 3.6.0), LINselect (>= 1.1.1), MASS (>= 7.3.45), methods (>= 3.6.0), plyr (>= 1.8.4), Rcpp (>= 1.0.2), robustbase (>= 0.92.6), stats (>= 3.6.0), utils (>= 3.6.0)

Suggests combinat (>= 0.0.8), doParallel (>= 1.0.10), phytools (>= 0.5.38), testthat (>= 1.0.2), TreeSim (>= 2.2), knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 2) | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL `https://github.com/pbastide/PhylogeneticEM`, `https://pbastide.github.io/PhylogeneticEM/`

BugReports `https://github.com/pbastide/PhylogeneticEM/issues`

VignetteBuilder knitr

NeedsCompilation yes

Author Paul Bastide [aut, cre], Mahendra Mariadassou [ctb]

Maintainer Paul Bastide <`paul.bastide@m4x.org`>

Repository CRAN

Date/Publication 2025-02-14 23:10:02 UTC

Contents

`allocate_regimes_from_shifts` *Allocation of regimes to nodes.*

Description

`allocate_regimes_from_shifts` allocate a number (from 0 to the number of shifts) to each node, corresponding to its regime : all nodes below shift i are numbered by i.

Usage

```
allocate_regimes_from_shifts(phylo, shifts_edges)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.
`shifts_edges`	edges were the shifts are.

Value

Vector of size (ntaxa + Nnode) of the regimes of each node and tip.

`allocate_shifts_from_regimes` *Allocation of shifts to edges*

Description

`allocate_shifts_from_regimes` returns the position of the shifts induced by the allocation of the regimes. Only works in an "infinite site" model.

Usage

```
allocate_shifts_from_regimes(phylo, regimes)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.
`regimes`	: vector of size (ntaxa + Nnode) of the regimes of each node and tip.

Value

Vector of edges numbers where the shifts are.

`check_parsimony` *Check Parsimony, assuming no homoplasy*

Description

`check_parsimony` take a vector of shifts edges, and check whether the number of groups of the tips induced by this allocation is exactly the number of shifts plus one. This is equivalent to parsimony when there is no homoplasy (i.e. no convergent regimes).

Usage

```
check_parsimony(tree, edges, ...)
```

Arguments

`tree`	phylogenetic tree
`edges`	a vector of edges of the tree, where the shifts are
`...`	possibly, a list giving the descendant tips of each edge

Details

This function computes explicitly the clustering of the tips, using function `clusters_from_shifts`. By default, this function uses `enumerate_tips_under_edges` to compute the list of tips under each edge, but a list can be provided (to avoid extra computation, if many tests on the same tree are done).

Value

boolean : TRUE if the allocation is parsimonious.

`clusters_from_shifts` *Clustering associated to a shift allocation, assuming no homoplasy.*

Description

`clusters_from_shifts` take a vector of shifts edges, and gives the clustering of the tips induced by them, in a "no homoplasy" model (i.e. no convergence is allowed).

Usage

```
clusters_from_shifts(tree, edges, part.list = enumerate_tips_under_edges(tree))
```

Arguments

`tree`	phylogenetic tree
`edges`	a vector of edges of the tree, where the shifts are
`part.list`	a list giving the descendant tips of each edge

Details

By default, this function uses `enumerate_tips_under_edges` to compute the list of tips under each edge.

Value

list of size n+m-1, entry i is the vector of tips bellow edge i.

`compute_betas_from_shifts` *Computation of the optimal values at nodes and tips.*

Description

`compute_betas_from_shifts` computes the optimal values at the nodes and tips of the tree, given the value at the root and the list of shifts occurring in the tree. It assumes an OU model.

Usage

```
compute_betas_from_shifts(phylo, optimal.value, shifts)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.
`optimal.value`	the optimal value at the root of the tree.
`shifts`	position and values of the shifts .

Details

Note that this is intended to be an internal function, and should not be used. In general, use `node_optimal_values` to get optimal values from a set of parameters.

Value

Vector of size (ntaxa + Nnode) of the optimal values at the tips of the tree.

`compute_dist_phy` *Phylogenetic Distances*

Description

`compute_dist_phy` computes the phylogenetic distances d_ij between all the tips i, j.

Usage

```
compute_dist_phy(phy)
```

Arguments

`phy`	a phylogenetic tree of class `phylo`.

Details

This function relies on ape function `dist.nodes`.

Value

a matrix of phylogenetic distances, ordered as the tips of the tree. The matrix is of type `symmetricMatrix-class`.

`compute_shifts_from_betas`
Computation of shifts from the vector of optimal values

Description

`compute_shifts_from_betas` computes the list of shifts corresponding to the vector of optimal values on nodes.

Usage

```
compute_shifts_from_betas(phylo, betas)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.
`betas`	vector of size (ntaxa + Nnode) of the optimal values at each node and tip.

Details

This function uses function fun on each row of matrix of edges.

Value

vector of shifts.

`compute_stationary_variance` *Compute the stationary variance matrix*

Description

`compute_stationary_variance` computes the stationary variance matrix of an OU process.

Usage

```
compute_stationary_variance(variance, selection.strength)
```

Arguments

`variance`	the variance (rate matrix) of the process.
`selection.strength`	the selection strength (alpha) matrix of the process.

Value

A positive definite Matrix of class `dpoMatrix-class`.

`compute_times_ca` *Common Ancestors Times*

Description

`compute_times_ca` computes the times t_ij between the root and the common ancestor of two tips i, j.

Usage

```
compute_times_ca(phy)
```

Arguments

`phy`	a phylogenetic tree of class `phylo`.

Details

This function relies on ape functions `node.depth.edgelength` and `mrca`.

Value

a matrix of times of shared evolution, ordered as the tips of the tree. The matrix is of type `symmetricMatrix-class`.

`correspondenceEdges` *Correspondence between edges numbers*

Description

`correspondenceEdges` takes edges numbers on an input tree, and gives back their corresponding numbers on the output tree.

Usage

```
correspondenceEdges(edges, from, to)
```

Arguments

`edges`	vector of index of edges in the tree "from"
`from`	initial input tree (format "`phylo`")
`to`	aimed output tree (format "`phylo`")

Value

vector of index of edges in the tree "to"

enlight *Make the result lighter*

Description

`enlight.PhyloEM` takes an object of class `PhyloEM`, and returns the same object, without saving the quantities that can be easily re-computed using function `imputed_traits.PhyloEM`.

Usage

```
enlight(x)

## S3 method for class 'PhyloEM'
enlight(x)
```

Arguments

`x`	an object of class `PhyloEM`.

Details

The resulting object can be much lighter, saving a lot of memory space, but each call to the function `imputed_traits.PhyloEM` will be longer. As function `plot.PhyloEM` relies on this function, this makes the plotting also longer. This has the same effect as setting the option "`light_result=TRUE`" in the call of `PhyloEM`.

Value

Same as entry, lighter.

Methods (by class)

- `enlight(PhyloEM)`: `PhyloEM` object

See Also

`PhyloEM`, `imputed_traits.PhyloEM`, `plot.PhyloEM`

enumerate_parsimony *Enumerate all the possible regime allocations, given a clustering of the tips.*

Description

`enumerate_parsimony` enumerate all the equivalent allocation of the regimes in the tree, a clustering of the tips being given. The number of such equivalent regimes is given by `parsimonyNumber` (which is faster).

Usage

```
enumerate_parsimony(phylo, clusters = rep(1, length(phylo$tip.label)))
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.
`clusters`	a vector representing the group of each tip. (Default to only one group with all the tips.)

Details

Function `extract.enumerate_parsimony` furnishes the result in a human readable form (for any subtree). Function `plot.enumerate_parsimony` plots all the solutions found on the tree.

Value

an S3 object of class "`enumerate_parsimony`", with:

nbrReconstructions an object of class "`parsimonyCost`", result of function `parsimonyCost`.

allocations a list of size Nnode + ntaxa. Each entry i of the list represents the solutions for the subtree starting at node i. It is a list with nclus entries, each entry being a matrix. A line of the kth matrix for the ith node is one possible allocation of the shifts, starting with regime k for node i.

phylo the entry phylogenetic tree

See Also

`extract.enumerate_parsimony`, `plot.enumerate_parsimony`, `parsimonyCost`, `parsimonyNumber`, `partitionsNumber`, `equivalent_shifts`

Examples

```
tree <- read.tree(text="(((A,B),C),D);")
plot(tree)
clusters <- c(0, 1, 2, 2)
sols <- enumerate_parsimony(tree, clusters)
plot(sols)

## Extract the parsimonious solutions from the root
extract(sols) # each line is a solution, with states of each node

## Extract the number of solutions from the root
extract(sols, what = "number")
extract(parsimonyNumber(tree, clusters)) # same result, more efficient

## Extract the cost of the solutions from the root
extract(sols, what = "cost")
extract(parsimonyCost(tree, clusters)) # same result, more efficient:

## Extract for the sub-tree below node 7
extract(sols, 7) # NAs: non-existing nodes in the sub-tree
```

`enumerate_tips_under_edges` *Tips descendants of nodes.*

Description

`enumerate_tips_under_edges` gives, for each edge of the tree, the labels of the tips that have this edge as an ancestor.

Usage

```
enumerate_tips_under_edges(tree)
```

Arguments

`tree`	phylogenetic tree, class `phylo`.

Details

This function uses function `prop.part` from package ape.

Value

list of size Nedge, entry i is the vector of tips bellow edge i.

equivalent_shifts *Find all equivalent shifts allocations and values.*

Description

`equivalent_shifts` computes the equivalent shifts positions and their corresponding values, assuming an ultrametric tree.

Usage

```
equivalent_shifts(
  phylo,
  params,
  T_tree = incidence.matrix(phylo),
  part.list = enumerate_tips_under_edges(phylo),
  times_shared = NULL
)
```

Arguments

`phylo`	a phylogenetic tree, of class `phylo`.
`params`	an object of class `params_process`, result inference by function `PhyloEM`, or constructed through function `params_process`
`T_tree`	(optional) matrix of incidence of the tree, result of function `incidence.matrix`
`part.list`	(optional) list of partition of the tree, result of function `enumerate_tips_under_edges`.
`times_shared`	(optional) a matrix, result of function `compute_times_ca`.

Details

This function is only valid for ultrametric trees, and for models: BM, OU with fixed root or stationary root. It assumes that there are no homoplasies.

Value

object of class `equivalent_shifts`, with entries:

eq_shifts_edges matrix of equivalent shifts

shifts_and_betas matrix of corresponding shifts values

phylo the entry phylogenetic tree

p the dimension

See Also

`plot.equivalent_shifts`, `extract.equivalent_shifts`, `params_BM`, `params_OU`, `enumerate_parsimony`

Examples

```
if (requireNamespace("TreeSim", quietly = TRUE)) {
  ## Simualte a tree
  set.seed(17920902)
  ntaxa = 20
  phylo <- TreeSim::sim.bd.taxa.age(n = ntaxa, numbsim = 1, lambda = 0.1,
                                    mu = 0, age = 1, mrca = TRUE)[[1]]

  ## Define parameters (BM, fixed root)
  params <- params_BM(p = 4, edges = c(6, 17, 31),
                      values = cbind(1:4, -(1:4), rep(1, 4)))
  ## Find equivalent solutions and plot them
  eq_shifts <- equivalent_shifts(phylo, params)
  eq_shifts
  plot(eq_shifts)
  ## Extract the values
  # Shifts values for trait 2, for the three shifts (rows), and three solutions (columns)
  extract(eq_shifts, trait = 2, what = "shifts_values")
  # Root values for trait 4, for the tree solutions (columns)
  extract(eq_shifts, trait = 4, what = "root_values")
  ## Define parameters (OU, stationary root
  params <- params_OU(p = 4, edges = c(6, 17, 31),
                      selection.strength = 0.1,
                      values = cbind(1:4, -(1:4), rep(1, 4)),
                      random = TRUE)
  ## Find equivalent solutions and plot them
  eq_shifts <- equivalent_shifts(phylo, params)
  eq_shifts
  plot(eq_shifts)
  ## Extract the values
  # Shifts values for trait 2, for the three shifts (rows), and three solutions (columns)
  extract(eq_shifts, trait = 2, what = "shifts_values")
  # Root values for trait 4, for the three solutions (columns)
  extract(eq_shifts, trait = 4, what = "root_values")
}
```

`estimateEM` *Perform One EM*

Description

`EstimateEM` performs one EM for one given number of shifts. It is called from function `PhyloEM`. Its use is mostly internal, and most user should not need it.

Usage

```
estimateEM(
  phylo,
  Y_data,
  Y_data_imp = Y_data,
  process = c("BM", "OU", "scOU", "rBM"),
  independent = FALSE,
 tol_EM = list(variance = 10^(-2), value.root = 10^(-2), exp.root = 10^(-2), var.root =
   10^(-2), selection.strength = 10^(-2), normalized_half_life = 10^(-2), log_likelihood
    = 10^(-2)),
  Nbr_It_Max = 500,
  method.variance = c("simple", "upward_downward"),
  method.init = c("default", "lasso"),
  method.init.alpha = c("default", "estimation"),
  method.init.alpha.estimation = c("regression", "regression.MM", "median"),
  nbr_of_shifts = 0,
  random.root = TRUE,
  stationary.root = TRUE,
  alpha_known = FALSE,
  eps = 10^(-3),
  known.selection.strength = 1,
  init.selection.strength = 1,
  max_selection.strength = 100,
  use_sigma_for_lasso = TRUE,
  max_triplet_number = 10000,
 min_params = list(variance = 0, value.root = -10^(5), exp.root = -10^(5), var.root = 0,
    selection.strength = 0),
 max_params = list(variance = 10^(5), value.root = 10^(5), exp.root = 10^(5), var.root =
    10^(5), selection.strength = 10^(5)),
  var.init.root = diag(1, nrow(Y_data)),
  variance.init = diag(1, nrow(Y_data), nrow(Y_data)),
  methods.segmentation = c("lasso", "same_shifts", "best_single_move"),
  check.tips.names = FALSE,
  times_shared = NULL,
  distances_phylo = NULL,
  subtree.list = NULL,
  T_tree = NULL,
  U_tree = NULL,
  h_tree = NULL,
  F_moments = NULL,
  tol_half_life = TRUE,
  warning_several_solutions = TRUE,
  convergence_mode = c("relative", "absolute"),
  check_convergence_likelihood = TRUE,
  sBM_variance = FALSE,
  method.OUsun = c("rescale", "raw"),
  K_lag_init = 0,
  allow_negative = FALSE,
  trait_correlation_threshold = 0.9,
  ...
)
```

Arguments

`phylo`	A phylogenetic tree of class `phylo` (from package `ape`).
`Y_data`	Matrix of data at the tips, size p x ntaxa. Each line is a trait, and each column is a tip. The column names are checked against the tip names of the tree.
`Y_data_imp`	(optional) imputed data if previously computed, same format as `Y_data`. Mostly here for internal calls.
`process`	The model used for the fit. One of "BM" (for a full BM model, univariate or multivariate); "OU" (for an OU with independent traits, univariate or multivariate); or "scOU" (for a "scalar OU" model, see details).
`independent`	Are the trait assumed to be independent from one another? Default to FALSE. OU in a multivariate setting only works if TRUE.
`tol_EM`	the tolerance for the convergence of the parameters. A named list, with items: **variance** default to 10^(-2) **value.root** default to 10^(-2) **exp.root** default to 10^(-2) **var.root** default to 10^(-2) **selection.strength** default to 10^(-2) **normalized_half_life** default to 10^(-2) **log_likelihood** default to 10^(-2)
`Nbr_It_Max`	the maximal number of iterations of the EM allowed. Default to 500 iterations.
`method.variance`	Algorithm to be used for the moments computations at the E step. One of "simple" for the naive method; of "upward_downward" for the Upward Downward method (usually faster). Default to "upward_downward".
`method.init`	The initialization method. One of "lasso" for the LASSO base initialization method; or "default" for user-specified initialization values. Default to "lasso".
`method.init.alpha`	For OU model, initialization method for the selection strength alpha. One of "estimation" for a cherry-based initialization, using `nlrob`; or "default" for user-specified initialization values. Default to "estimation".
`method.init.alpha.estimation`	If method.init.alpha="estimation", choice of the estimation(s) methods to be used. Choices among "regression", (method="M" is passed to `nlrob`); "regression.MM" (method="MM" is passed to `nlrob`) or "median" (`nlrob` is not used, a simple median is taken). Default to all of them.
`nbr_of_shifts`	the number of shifts allowed.
`random.root`	whether the root is assumed to be random (TRUE) of fixed (FALSE). Default to TRUE
`stationary.root`	whether the root is assumed to be in the stationary state. Default to TRUE.
`alpha_known`	is the selection strength assumed to be known ? Default to FALSE.
`eps`	tolerance on the selection strength value before switching to a BM. Default to 10^(-3).

`known.selection.strength`	if `alpha_known=TRUE`, the value of the known selection strength.
`init.selection.strength`	(optional) a starting point for the selection strength value.
`max_selection.strength`	the maximal value allowed of the selection strength. Default to 100.
`use_sigma_for_lasso`	whether to use the first estimation of the variance matrix in the lasso regression. Default to TRUE.
`max_triplet_number`	for the initialization of the selection strength value (when estimated), the maximal number of triplets of tips to be considered.
`min_params`	a named list containing the minimum allowed values for the parameters. If the estimation is smaller, then the EM stops, and is considered to be divergent. Default values: **variance** default to 0; **value.root** default to -10^(5); **exp.root** default to -10^(5); **var.root** default to 0; **selection.strength** default to 0
`max_params`	a named list containing the maximum allowed values for the parameters. If the estimation is larger, then the EM stops, and is considered to be divergent. Default values: **variance** default to 10^(5); **value.root** default to 10^(5); **exp.root** default to 10^(5); **var.root** default to 10^(5); **selection.strength** default to 10^(5)
`var.init.root`	optional initialization value for the variance of the root.
`variance.init`	optional initialization value for the variance.
`methods.segmentation`	For OU, method(s) used at the M step to find new candidate shifts positions. Choices among "lasso" for a LASSO-based algorithm; and "best_single_move" for a one-move at a time based heuristic. Default to both of them. Using only "lasso" might speed up the function a lot.
`check.tips.names`	whether to check the tips names of the tree against the column names of the data. Default to TRUE.
`times_shared`	(optional) times of shared ancestry of all nodes and tips, result of function `compute_times_ca`
`distances_phylo`	(optional) phylogenetic distances, result of function `compute_dist_phy`.
`subtree.list`	(optional) tips descendants of all the edges, result of function `enumerate_tips_under_edges`.
`T_tree`	(optional) matrix of incidence of the tree, result of function `incidence.matrix`.

`U_tree`	(optional) full matrix of incidence of the tree, result of function `incidence.matrix.full`.
`h_tree`	(optional) total height of the tree.
`F_moments`	(optional, internal)
`tol_half_life`	should the tolerance criterion be applied to the phylogenetic half life (TRUE, default) or to the raw selection strength ?
`warning_several_solutions`	whether to issue a warning if several equivalent solutions are found (default to TRUE).
`convergence_mode`	one of "relative" (the default) or "absolute". Should the tolerance be applied to the raw parameters, or to the renormalized ones ?
`check_convergence_likelihood`	should the likelihood be taken into consideration for convergence assessment ? (default to TRUE).
`sBM_variance`	Is the root of the BM supposed to be random and "stationary"? Used for BM equivalent computations. Default to FALSE.
`method.OUsun`	Method to be used in univariate OU. One of "rescale" (rescale the tree to fit a BM) or "raw" (directly use an OU, only available for univariate processes).
`K_lag_init`	Number of extra shifts to be considered at the initialization step. Increases the accuracy, but can make computations quite slow of taken too high. Default to 5.
`allow_negative`	whether to allow negative values for alpha (Early Burst). See documentation of `PhyloEM` for more details. Default to FALSE.
`trait_correlation_threshold`	the trait correlation threshold to stop the analysis. Default to 0.9.
`...`	Further arguments to be passed to `estimateEM`, including tolerance parameters for stopping criteria, maximal number of iterations, etc.

Details

See documentation of `PhyloEM` for further details. All the parameters monitoring the EM (like `tol_EM`, `Nbr_It_Max`, etc.) can be called from `PhyloEM`.

Value

An object of class `EstimateEM`.

See Also

`PhyloEM`

extract — *Extraction function*

Description

`extract` the needed quantities out of an S3 object.

Usage

```
extract(x, ...)
```

Arguments

`x`	an S3 object.
`...`	further arguments to be passed to the specific method.

Value

An integer giving the number of equivalent parsimonious solutions.

See Also

`extract.parsimonyNumber`, `extract.parsimonyCost`, `extract.enumerate_parsimony`, `extract.partitionsNumber`

extract.enumerate_parsimony — *Extract the result of* `enumerate_parsimony` *at a node.*

Description

`extract.enumerate_parsimony` returns a matrix containing all the possible regime allocations for the nodes of a given subtree.

Usage

```
## S3 method for class 'enumerate_parsimony'
extract(
  x,
  node = attr(x$allocations, "ntaxa") + 1,
  what = c("solutions", "number", "cost"),
  ...
)
```

Arguments

`x`	an object of class "`enumerate_parsimony`", result of function `enumerate_parsimony`.
`node`	the node where to retrieve the parsimony number. Default to the root of the tree.
`what`	the quantity to retrieve. Either "solutions" for the full solutions, "number" for the number of solutions, or "cost" for the minimal cost of a solution. Default to "solutions"
`...`	unused

Value

A matrix with ntaxa + Nnode columns, and as many rows as the number of possible parsimonious reconstructions.

See Also

`enumerate_parsimony`, `plot.enumerate_parsimony`

---

extract.equivalent_shifts

Extract the shifts values for one trait.

Description

`extract.equivalent_shifts` takes an object of class `equivalent_shifts`, result of function `equivalent_shifts`, and returns the shifts of root values for a given trait.

Usage

```
## S3 method for class 'equivalent_shifts'
extract(x, trait = 1, what = c("shifts_values", "root_values"), ...)
```

Arguments

`x`	an object of class `equivalent_shifts`, result of function `equivalent_shifts`
`trait`	the number of the trait to be extracted. Default to 1.
`what`	one of "shifts_values" or "root_values".
`...`	unused.

Value

A matrix with the values of the shifts (`what = "shifts_values"`) or the root (`what = "root_values"`) for the trait for each equivalent configuration. Each column is one configuration.

See Also

`equivalent_shifts`, `plot.equivalent_shifts`, `equivalent_shifts_edges`

extract.parsimonyCost *Extraction of the actual number of solutions.*

Description

`extract.parsimonyCost` takes an object of class "`parsimonyCost`", result of function `parsimonyCost`, and computes the minimum cost at the given node.

Usage

```
## S3 method for class 'parsimonyCost'
extract(x, node = attr(x, "ntaxa") + 1, ...)
```

Arguments

`x`	an object of class "`parsimonyCost`", result of function `parsimonyCost`.
`node`	the root node of the subtree. By default, the root of the tree.
`...`	unused

Value

An integer giving the minimum cost of the subtree.

See Also

`parsimonyCost`

extract.parsimonyNumber *Extraction of the actual number of solutions.*

Description

`extract.parsimonyNumber` takes the two matrices computed by `parsimonyNumber`, and compute the actual number of parsimonious solution for any subtree starting from a given node.

Usage

```
## S3 method for class 'parsimonyNumber'
extract(
  x,
  node = attr(x$nbrReconstructions, "ntaxa") + 1,
  what = c("number", "cost"),
  ...
)
```

Arguments

`x`	an object of class "`parsimonyNumber`", result of function `parsimonyNumber`.
`node`	the root node of the subtree. By default, the root of the tree.
`what`	the quantity to retrieve. Either "number" for the number of solutions, or "cost" for the minimal cost of a solution. Default to "number".
`...`	unused

Details

The parsimonious solutions are the one with the minimum number of shifts (that are given by matrix costReconstructions). This function sums the number of solutions (given in matrix nbrReconstructions) that have the minimum number of shifts.

Value

An integer giving the number of equivalent parsimonious solutions.

See Also

`parsimonyNumber`

`extract.partitionsNumber`

Extract from object `partitionsNumber`

Description

`extract.partitionsNumber` extracts the number of partitions for a given sub-tree, either marked or non-marked.

Usage

```
## S3 method for class 'partitionsNumber'
extract(
  x,
  node = attr(x, "ntaxa") + 1,
  npart = attr(x, "npart"),
  marked = FALSE,
  ...
)
```

Arguments

`x`	an object of class `partitionsNumber`, result of function `partitionsNumber`.
`node`	the root node of the subtree where to get the result. Default to the root of the tree.
`npart`	the number of partitions (colors) allowed at the tips. Default to the value used in the call of function `partitionsNumber` (the maximum).
`marked`	whether to extract the marked (TRUE) or un-marked (FALSE) partitions. The number of models is the number of un-marked partitions. Default to FALSE.
`...`	unused.

Value

the number of partitions with npart colors, on the sub-tree starting at node, marked or not.

See Also

`partitionsNumber`

`extract.simul_process` *Extraction of simulated traits*

Description

`extract.simul_process` takes an object of class "`simul_process`", result of function `simul_process`, and extracts the traits values, expectations or optimal values at the tips or the internal nodes.

Usage

```
## S3 method for class 'simul_process'
extract(
  x,
  where = c("tips", "nodes"),
  what = c("states", "expectations", "optimal.values"),
  ...
)
```

Arguments

`x`	an object of class "`simul_process`", result of function `simul_process`.
`where`	one of "tips" (the default) or "nodes". Where to extract the results.
`what`	one of "states" (the default), "expectation", or "optimal.values".
`...`	unused

Details

##

Value

A matrix giving the selected quantities at the selected nodes or tips. If the tips or nods are labeled, then the colnames of the matrix are set accordingly.

See Also

`simul_process`

`find_grid_alpha` *Find a reasonable grid for alpha*

Description

Grid so that 2*ln(2)*quantile(d_ij)/factor_up_alpha < t_1/2 < factor_down_alpha * ln(2) * h_tree, with t_1/2 the phylogenetic half life: t_1/2 = log(2)/alpha. Ensures that for alpha_min, it is almost a BM, and for alpha_max, almost all the tips are decorrelated.

Usage

```
find_grid_alpha(
  phy,
  alpha = NULL,
  nbr_alpha = 10,
  factor_up_alpha = 2,
  factor_down_alpha = 3,
  quantile_low_distance = 1e-04,
  log_transform = TRUE,
  allow_negative = FALSE,
  ...
)
```

Arguments

`phy`	phylogenetic tree of class "phylo"
`alpha`	fixed vector of alpha values if already known. Default to NULL.
`nbr_alpha`	the number of elements in the grid
`factor_up_alpha`	factor for up scalability
`factor_down_alpha`	factor for down scalability
`quantile_low_distance`	quantile for min distance
`log_transform`	whether to take a log scale for the spacing of alpha values. Default to TRUE.
`allow_negative`	whether to allow negative values for alpha (Early Burst). See documentation of `PhyloEM` for more details. Default to FALSE.
`...`	not used.

Details

If `quantile_low_distance=0`, then `quantile(d_ij)=min(d_ij)`, and, for any two tips i,j, the correlation between i and j is bounded by exp(-factor_up_alpha/2). Those values of alpha will be used for the re-scaling of the tree, which has an exponential term in exp(2*alpha*h). The function makes sure that this number is below the maximal float allowed (equals to `.Machine$double.xmax`).

Value

A grid of alpha values

See Also

`transform_branch_length`, `.Machine`

`find_rotation` *Test for rotation invariant datasets*

Description

`find_rotation` takes two fits from from `PhyloEM`, and test if their datasets are equal up to a rotation.

Usage

```
find_rotation(res1, res2, tol = NULL)
```

Arguments

`res1`	an object of class `PhyloEM`.
`res2`	an object of class `PhyloEM`.
`tol`	relative numerical tolerance. Default to `.Machine$double.eps^(0.5)`.

Value

If appropriate, the rotation matrix rot such that dat1 = rot

`get_criterion` *Get Model Selection Criterion*

Description

This function takes an object of class `PhyloEM`, result of function `PhyloEM`, and return the values of the model selection criterion for each value of K.

Usage

```
get_criterion(res, method.selection = NULL)
```

Arguments

`res`	an object of class `PhyloEM`, result of function `PhyloEM`.
`method.selection`	select the parameters to plot. One of "LINselect", "DDSE", "Djump" or "likelihood" (for un-penalized likelihood). Default to "LINselect". See `params_process.PhyloEM`.

Value

A named vector with the values of the criterion for each number of shift K.

See Also

`params_process.PhyloEM`, `plot.PhyloEM`, `plot_criterion`

`imputed_traits` *Ancestral State Reconstruction*

Description

`imputed_traits.PhyloEM` takes an object of class `PhyloEM`, and returns the imputed traits values, either at the internal nodes (ancestral state reconstruction) or at the tips (data imputation)

Usage

```
imputed_traits(x, ...)

## S3 method for class 'PhyloEM'
imputed_traits(
  x,
  trait = 1,
  save_all = FALSE,
  where = c("nodes", "tips"),
  what = c("imputed", "variances", "expectations"),
  params = NULL,
  method.selection = NULL,
  reconstructed_states = NULL,
  ...
)
```

Arguments

`x`	an object of class `PhyloEM`.
`...`	further arguments to be passed on to `params_process.PhyloEM`
`trait`	an integer giving the trait to extract. Default to 1.
`save_all`	if TRUE, arguments `where` and `what` are ignored, and all the moments are kept for further extraction with the same function, specifying the argument `reconstructed_states`. Default to FALSE.
`where`	either "nodes" for ancestral state reconstruction, or "tips" for data imputation.
`what`	the quantity to retrieve. Either the imputed traits (default), their conditional variances, or the simple expectations under the selected process.
`params`	(optional) some user-specified parameters. Must be of class `params_process`. If left blank, they are extracted using the `method.selection` argument (see below).
`method.selection`	(optional) the method selection to be used. One of "LINselect", "DDSE", "Djump". Default to "LINselect".
`reconstructed_states`	if the reconstructed states have already been computed (by a previous call of the function, with `save_all=TRUE`), they can be passed on here (avoids multiple computations of the E step).

Value

A matrix or array with the computed quantities.

Methods (by class)

- `imputed_traits(PhyloEM)`: `PhyloEM` object

See Also

`params_process.PhyloEM`, `PhyloEM`

`incidence.matrix` *Incidence matrix of a tree.*

Description

`incidence.matrix` computes the incidence matrix T of a tree : for a lineage i and a branch b, T[i,b]=1 if b is in the lineage i, and 0 otherwise.

Usage

```
incidence.matrix(phylo)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.

Value

Matrix of incidence, size Nedge x ntaxa.

See Also

`incidence.matrix.full`

`incidence.matrix.full` *Incidence matrix of a tree.*

Description

`incidence.matrix.full` computes the incidence matrix U of a tree : for a node i and a branch b, U[i,b]=1 if b is in the lineage i, and 0 otherwise.

Usage

```
incidence.matrix.full(phylo)
```

Arguments

`phylo`	a phylogenetic tree, class `phylo`.

Value

Matrix of incidence, size ntaxa + Nnode.

See Also

`incidence.matrix`

log_likelihood *Log Likelihood of a fitted object*

Description

log_likelihood computes the log likelihood of some parameters.

Usage

```
log_likelihood(x, ...)

## S3 method for class 'params_process'
log_likelihood(x, Y_data, phylo, ...)

## S3 method for class 'PhyloEM'
log_likelihood(x, ...)
```

Arguments

x	an object of class params_process or PhyloEM.
...	for a PhyloEM object, further arguments to be passed on to params_process.PhyloEM (to choose which parameters to extract from the results, see documentation of this function).
Y_data	matrix of data at the tips, size p x ntaxa. Each line is a trait, and each column is a tip. The column names are checked against the tip names of the tree.
phylo	a phylogenetic tree, class phylo. incidence.matrix.full. Can be specified to avoid extra computations.

Value

The log likelihood of the data with the provided parameters on the tree.

Methods (by class)

- log_likelihood(params_process): params_process object
- log_likelihood(PhyloEM): PhyloEM object

See Also

params_process, PhyloEM

`merge_alpha_grids` *Merge PhyloEM fits on various grids of alpha values*

Description

`merge_alpha_grids` takes several fits from `PhyloEM`, and merge them so as to take into account all alpha values. This can be used to break down computations into smaller chunks to be run independently.

Usage

```
merge_alpha_grids(...)
```

Arguments

`...`	objects of class `PhyloEM` fitted on the same dataset with the same parameters, but different grids of alpha values.

Value

An object of class `PhyloEM`, result of the merge.

Examples

```
## Not run:
## Load Data
data(monkeys)
## First fit with coarse grid
res1 <- PhyloEM(Y_data = monkeys$dat,
                phylo = monkeys$phy,
                process = "scOU",
                random.root = TRUE,
                stationary.root = TRUE,
                K_max = 10,
                alpha = c(0.2, 0.3),
                parallel_alpha = TRUE,
                Ncores = 2)
## Second fit with finer grid
res2 <- PhyloEM(Y_data = monkeys$dat,
                phylo = monkeys$phy,
                process = "scOU",
                random.root = TRUE,
                stationary.root = TRUE,
                K_max = 10,
                alpha = c(0.22, 0.24),
                parallel_alpha = TRUE,
                Ncores = 2)
## Thrid fit with redundancies
res3 <- PhyloEM(Y_data = monkeys$dat,
                phylo = monkeys$phy,
                process = "scOU",
                random.root = TRUE,
                stationary.root = TRUE,
                K_max = 10,
                alpha = c(0.26, 0.3),
                parallel_alpha = TRUE,
                Ncores = 2)

## Merge the three
res_merge <- merge_alpha_grids(res1, res2, res3)
## Plot the selected result
plot(res_merge)
## Plot the model selection criterion
plot_criterion(res_merge)

## End(Not run)
```

merge_rotations *Merge fits from independent runs of PhyloEM.*

Description

merge_rotations takes several fits from PhyloEM, and merge them according to the best score (maximum likelihood or least squares). For each number of shifts, The datasets needs to be equal up to a rotation. This is tested thanks to a QR decomposition, see function find_rotation.

Usage

```
merge_rotations(..., method.selection = NULL, tol = NULL)
```

Arguments

...	objects of class PhyloEM fitted on datasets that are equal up to a rotation.
method.selection	(optional) selection method to be applied to the merged fit. See params_process.PhyloEM.
tol	(optional) relative numerical tolerance. See find_rotation.

Value

An object of class PhyloEM, result of the merge.

Examples

```
## Not run:
## Load Data
data(monkeys)
## Run method
# Note: use more alpha values for better results.
res <- PhyloEM(Y_data = monkeys$dat,        ## data
               phylo = monkeys$phy,         ## phylogeny
               process = "scOU",            ## scalar OU
               random.root = TRUE,          ## root is stationary
               stationary.root = TRUE,
               K_max = 10,                  ## maximal number of shifts
               nbr_alpha = 4,               ## number of alpha values
               parallel_alpha = TRUE,       ## parallelize on alpha values
               Ncores = 2)
## Rotate dataset
rot <- matrix(c(cos(pi/4), -sin(pi/4), sin(pi/4), cos(pi/4)), nrow= 2, ncol = 2)
Yrot <- t(rot) %*% monkeys$dat
rownames(Yrot) <- rownames(monkeys$dat)
## Fit rotated dataset
# Note: use more alpha values for better results.
res_rot <- PhyloEM(Y_data = Yrot,               ## rotated data
                   phylo = monkeys$phy,
                   process = "scOU",
                   random.root = TRUE,
                   stationary.root = TRUE,
                   K_max = 10,
                   nbr_alpha = 4,
                   parallel_alpha = TRUE,
                   Ncores = 2)
## Merge the two
res_merge <- merge_rotations(res, res_rot)
## Plot the selected result
plot(res_merge)
## Plot the model selection criterion
plot_criterion(res_merge)

## End(Not run)
```

`model_selection` *Model Selection of a fitted object*

Description

`model_selection` does the model selection on a fitted `PhyloEM` object, and returns the same fitted object.

Usage

```
model_selection(x, ...)

## S3 method for class 'PhyloEM'
model_selection(
  x,
  method.selection = c("LINselect", "DDSE", "Djump"),
  C.BM1 = 0.1,
  C.BM2 = 2.5,
  C.LINselect = 1.1,
  independent = FALSE,
  ...
)
```

Arguments

`x`	a fitted `PhyloEM` object
`...`	Further arguments to be passed to `estimateEM`, including tolerance parameters for stopping criteria, maximal number of iterations, etc.
`method.selection`	Method selection to be used. Several ones can be used at the same time. One of "LINselect" for the Baraud Giraud Huet LINselect method; "DDSE" for the Slope Heuristic or "Djump" for the Jump Heuristic, last two based the Birgé Massart method.
`C.BM1`	Multiplying constant to be used for the BigeMassart1 method. Need to be positive. Default to 0.1.
`C.BM2`	Multiplying constant to be used for the BigeMassart2 method. Default to 2.5.
`C.LINselect`	Multiplying constant to be used for the LINselect method. Need to be greater than 1. Default to 1.1.
`independent`	Are the trait assumed to be independent from one another? Default to FALSE. OU in a multivariate setting only works if TRUE.

Value

The same object, but with a slot corresponding to the model selection used. See function `params_process.PhyloEM` to retrieve the selected parameters.

Methods (by class)

- `model_selection(PhyloEM)`: `PhyloEM` object

See Also

`PhyloEM`, `params_process.PhyloEM`, `imputed_traits.PhyloEM`

`monkeys` *New World Monkeys dataset*

Description

Morphometric dataset and phylogeny for brain shape variation of 50 species of New World monkeys (platyrrhine).

Usage

```
monkeys
```

Format

A list containing two objects:

phy The Phylogenetic tree for the platyrrhine species, pruned to match the species in the morphometric dataset

dat First two PC scores from a PCA of the species-averaged Procrustes coordinates

References

Aristide, L., dos Reis, S. F., Machado, A. C., Lima, I., Lopes, R. T. & Perez, S. I. (2016). Brain shape convergence in the adaptive radiation of New World monkeys. Proceedings of the National Academy of Sciences, 113(8), 2158–2163. http://doi.org/10.1073/pnas.1514473113

`node_optimal_values` *Computation of the optimal values at nodes and tips.*

Description

`compute_betas_from_shifts` computes the optimal values at the nodes and tips of the tree, given the value at the root and the list of shifts occurring in the tree. It assumes an OU model.

Usage

```
node_optimal_values(param, phylo)
```

Arguments

`param`	an object of class `params_process`.
`phylo`	a phylogenetic tree, class `phylo`.

Value

Matrix of size ntraits x (ntaxa + Nnode) of the optimal values at the node and tips of the tree. Column names correspond to the number of the node in the phylo object.

Examples

```
set.seed(1792)
ntaxa = 10
tree <- rphylo(ntaxa, 1, 0.1)
# parameters of the process
par <- params_process("BM",                          ## Process
                      p = 2,                         ## Dimension
                      variance = diag(0.5, 2, 2) + 0.5, ## Rate matrix
                      edges = c(4, 10, 15),          ## Positions of the shifts
                      values = cbind(c(5, 4),        ## Values of the shifts
                                     c(-4, -5),
                                     c(5, -3)))
plot(par, phylo = tree, traits = 1, value_in_box = TRUE,
     shifts_bg = "white", root_bg = "white", ancestral_as_shift = TRUE, root_adj = 5)
nodelabels()
node_optimal_values(par, tree)
```

`params_BM` *Create an object* `params_process` *for a BM*

Description

`params_BM` creates a coherent object params_process from user provided values of the parameters. Non specified parameters are set to default values.

Usage

```
params_BM(
  p = 1,
  variance = diag(1, p, p),
  random = FALSE,
  value.root = rep(0, p),
  exp.root = rep(0, p),
  var.root = diag(1, p, p),
  edges = NULL,
  values = matrix(0, p, length(edges)),
  relativeTimes = NULL,
  nbr_of_shifts = length(edges),
  phylo = NULL,
  sBM_variance = FALSE,
  trait_names = NULL,
  ...
)
```

Arguments

`p`	the dimension (number of traits) of the parameters. Default to 1.
`variance`	the variance (rate matrix) of the BM. Default to `diag(1, p, p)`.
`random`	whether the root of the BM is random (TRUE) or fixed (FALSE). Default to FALSE.
`value.root`	if random=FALSE, the root value. Default to 0.
`exp.root`	if random=TRUE, the root expectation. Default to 0.
`var.root`	if random=TRUE, the root variance. Default to `diag(1, p, p)`.
`edges`	a vector of edges where the shifts occur. Default to NULL (no shift).
`values`	a matrix of shift values, with p lines and as many columns as the number of shifts. Each column is the p values for one shift. Default to `matrix(0, p, length(edges))`.
`relativeTimes`	(unused) the relative position of the shift on the branch, between 0 (beginning of the branch) and 1 (end of the branch). Default to 0.
`nbr_of_shifts`	the number of shifts to use (randomly drawn). Use only if `edges` is not specified. In that case, a phylogenetic tree must be provided (to allow a random sampling of its edges).
`phylo`	a phylogenetic tree of class `phylo`. Needed only if the shifts edges are not specified, or if sBM_variance=TRUE. Default to NULL. If sBM_variance=TRUE, it must have a specified value for the root branch length (slot root.edge).
`sBM_variance`	if the root is random, does it depend on the length of the root edge ? (For equivalent purposes with a rescaled OU). Default to FALSE. If TRUE, a phylogenetic tree with root edge length must be provided.
`trait_names`	vector of trait names values. Must be of length p.
`...`	unused.

Value

an object of class `params_process`.

See Also

`params_process`, `params_OU`

`params_OU` *Create an object* `params_process` *for an OU*

Description

`params_OU` creates a coherent object params_process from user provided values of the parameters. Non specified parameters are set to default values.

Usage

```
params_OU(
  p = 1,
  variance = diag(1, p, p),
  selection.strength = diag(1, p, p),
  optimal.value = rep(0, p),
  random = TRUE,
  stationary.root = TRUE,
  value.root = rep(0, p),
  exp.root = rep(0, p),
  var.root = diag(1, p, p),
  edges = NULL,
  values = matrix(0, p, length(edges)),
  relativeTimes = NULL,
  nbr_of_shifts = length(edges),
  phylo = NULL,
  trait_names = NULL,
  ...
)
```

Arguments

`p`	the dimension (number of traits) of the parameters. Default to 1.
`variance`	the variance (rate matrix) of the BM. Default to `diag(1, p, p)`.
`selection.strength`	the selection strength matrix. Default to `diag(1, p, p)`.
`optimal.value`	the vector of the optimal values at the root. Default to `rep(0, p)`.
`random`	whether the root of the OU is random (TRUE) or fixed (FALSE). Default to TRUE.
`stationary.root`	whether the root of the OU is stationary (TRUE) or not. Default to TRUE.
`value.root`	if random=FALSE, the root value. Default to 0.
`exp.root`	if random=TRUE, the root expectation. Default to 0. If stationary.root=TRUE, default to `optimal.value`.
`var.root`	if random=TRUE, the root variance. Default to `diag(1, p, p)`. If stationary.root=TRUE, default to the stationary variance computed from `variance` and `selection.strength`, see function `compute_stationary_variance`.
`edges`	a vector of edges where the shifts occur. Default to NULL (no shift).
`values`	a matrix of shift values, with `p` lines and as many columns as the number of shifts. Each column is the `p` values for one shift. Default to `matrix(0, p, length(edges))`.
`relativeTimes`	(unused) the relative position of the shift on the branch, between 0 (beginning of the branch) and 1 (end of the branch). Default to 0.
`nbr_of_shifts`	the number of shifts to use (randomly drawn). Use only if `edges` is not specified. In that case, a phylogenetic tree must be provided (to allow a random sampling of its edges).

`phylo`	a phylogenetic tree of class `phylo`. Needed only if the shifts edges are not specified.
`trait_names`	vector of trait names values. Must be of length p.
`...`	unused.

Value

an object of class `params_process`.

See Also

`params_process`, `params_BM`

`params_process` *Create an object params_process*

Description

`params_process` creates or extracts a set of parameters of class `params_process`.

Usage

```
params_process(x, ...)
```

Arguments

`x`	an S3 object.
`...`	further arguments to be passed to the specific method.

Value

An S3 object of class `params_process`. This is essentially a list containing the following entries:

process The model used. One of "BM" (for a full BM model, univariate or multivariate); "OU" (for a full OU model, univariate or multivariate); or "scOU" (for a "scalar OU" model).

p Dimension of the trait.

root.state List describing the state of the root, with:

- **random** random state (TRUE) or deterministic state (FALSE)
- **value.root** if deterministic, value of the character at the root
- **exp.root** if random, expectation of the character at the root
- **var.root** if random, variance of the character at the root (pxp matrix)

shifts List with position and values of the shifts:

- **edges** vector of the K id of edges where the shifts are
- **values** matrix p x K of values of the shifts on the edges (one column = one shift)

relativeTimes vector of dimension K of relative time of the shift from the parent node of edges

variance Variance-covariance matrix size p x p.

selection.strength Matrix of selection strength size p x p (OU).

optimal.value Vector of p optimal values at the root (OU).

See Also

`params_process.character`, `params_process.PhyloEM`, `params_BM`, `params_OU` `simul_process.params_process`

`params_process.character` *Create an object* `params_process`

Description

`params_process` creates a coherent object params_process from user provided values of the parameters.

Usage

```
## S3 method for class 'character'
params_process(x, ...)
```

Arguments

`x`	one of "BM" or "OU"
`...`	specified parameters, see functions `params_BM` and `params_OU` for details.

Value

an object of class `params_process`.

See Also

`params_BM`, `params_OU`

params_process.PhyloEM

Parameter estimates

Description

params takes an object of class PhyloEM, and returns the inferred parameters of the process.

Usage

```
## S3 method for class 'PhyloEM'
params_process(
  x,
  method.selection = NULL,
  K = NULL,
  alpha = NULL,
  rBM = FALSE,
  init = FALSE,
  ...
)
```

Arguments

x	an object of class PhyloEM
method.selection	(optional) the method selection to be used. One of "LINselect", "DDSE", "Djump". Default to "LINselect".
K	(optional) an integer giving the number of shifts for which to retrieve the parameters. Default to NULL (automatically selected number of shifts, see method.selection argument).
alpha	(optional) a value of alpha for which to retrieve the parameters. Can be an (unambiguous) estimation of the true value. If specified, then K must be precised too. Default to NULL (automatically selected value, see method.selection argument).
rBM	(optional) if TRUE, and if the process is "scOU", returns the raw parameters of the BM on the re-scaled tree. Default to FALSE, except if the selection strength is negative (see doc of PhyloEM for an explanation of this particular case).
init	(optional) if TRUE, gives the parameters from the initialization of the EM. Default to FALSE. This has no effect if K is not specified.
...	unused.

Value

An object of class params_process.

See Also

PhyloEM, imputed_traits.PhyloEM

---

parsimonyCost *Minimal number of shifts needed to get a clustering.*

Description

parsimonyCost is an implementation of the Sankoff algorithm, when the cost of transition between two state is always one. It is used in functions parsimonyNumber and enumerate_parsimony to count or enumerate all the parsimonious solutions given one clustering of the tips.

Usage

```
parsimonyCost(phylo, clusters = rep(1, length(phylo$tip.label)))
```

Arguments

phylo	a phylogenetic tree, class phylo.
clusters	the vector of the clusters of the tips. (Default to all the tips in a single group).

Value

An S3 class "parsimonyCost" containing a (ntaxa + Nnode) x (nclus) matrix of the total number of shifts needed to get the clustering, if starting from a node in state k. The cost can be extract from any subtree with function extract.parsimonyCost.

See Also

extract.parsimonyCost, parsimonyNumber, enumerate_parsimony, partitionsNumber, equivalent_shifts

Examples

```
tree <- read.tree(text="(((1,1),2),2);")
plot(tree); nodelabels()
clusters <- c(1, 1, 2, 2)
costs <- parsimonyCost(tree, clusters)
costs

## Extract the parsimony cost at the root
extract(costs)

## Extract the cost for the sub-tree below node 7
extract(costs, 7)
```

`parsimonyNumber` *Number of equivalent parsimonious allocations.*

Description

`parsimonyNumber` aims at finding the number of equivalent allocations of the shifts on the tree, i.e allocations that are parsimonious and compatible with a given clustering of the tips.

Usage

```
parsimonyNumber(phylo, clusters = rep(1, length(phylo$tip.label)))
```

Arguments

`phylo`	phylogenetic tree, class `phylo`.
`clusters`	the vector of the clusters of the tips. Default to all the tips in one single cluster.

Details

This function does a recursion up the tree. The function `extract.parsimonyNumber` gives the result sought for any subtree. The matrix of costs of the states (number of shifts) is also required, it is computed by function `parsimonyCost`.

Value

an object of S3 class "`parsimonyNumber`" with:

nbrReconstructions a (ntaxa + Nnode) x (nclus) matrix of locally parsimonious solutions starting from a cluster k at a given node

costReconstructions an object of class "`parsimonyCost`", result of function `parsimonyCost`.

See Also

`extract.parsimonyNumber`, `parsimonyCost`, `enumerate_parsimony`, `partitionsNumber`, `equivalent_shifts`

Examples

```
tree <- read.tree(text="(((0,1),2),2);")
plot(tree); nodelabels()
clusters <- c(0, 1, 2, 2)
n_sols <- parsimonyNumber(tree, clusters)
n_sols

## Extract the number of parsimonious solutions at the root
extract(n_sols)

## Extract the cost of the solutions from the root
extract(n_sols, what = "cost")
extract(parsimonyCost(tree, clusters)) # same, more efficient

## Extract for the sub-tree below node 7
extract(n_sols, 7) # Result: 2 (the ancestral state is either "0" or "1").
```

---

partitionsNumber *Number of different models*

Description

partitionsNumber computes the number of different models with a given number of shifts K. It is also the number of colorings of the tips to the tree in npart = K + 1 colors.

Usage

```
partitionsNumber(phylo, npart)
```

Arguments

phylo	a phylogenetic tree of class phylo.
npart	the numbers of partitions (colors) allowed at the tips. This is the number of shifts plus one (npart = K + 1).

Value

an object of class partitionsNumber. This is made of a matrix with (Nnode + ntaxa) rows and (2*npart) columns. Each column contains two vectors: for k=1:npart it contains the number of partitions with k groups compatible with the tree and the shift process; and for k=(npart+1):2*npart, it contains the number of "marked" partitions with (k-npart) groups compatible with the tree and the shift process. The actual number can be extracted with function extract.partitionsNumber (see examples below).

See Also

extract.partitionsNumber, parsimonyNumber, equivalent_shifts

Examples

```
if (requireNamespace("combinat", quietly = TRUE)) {
  npart <- 8 # number of colors at the tips allowed
  tree <- read.tree(text="(A,(A,(A,A,A),A,A));") # a tree with polytomies
  plot(tree)
  parts_num <- partitionsNumber(tree, npart)
  parts_num

  ## Number of possible colorings of the tips in npart colors
  extract(parts_num)
  ## Get all the solutions for colorings with 1 to nparts colors
  extract(parts_num, npart = 1:npart)

  ## Number of possible colorings of the tips in npart colors
  ## For the sub-tree starting at node 17
  extract(parts_num, node = 10)

  ## Number of possible colorings of the tips in npart colors
  ## with one marked color
  extract(parts_num, marked = TRUE)
}
```

PhyloEM — *Model Estimation with Detection of Shifts*

Description

PhyloEM is the main function of the package. It uses maximum likelihood methods to fit a BM or an OU process for several traits evolving along a phylogenetic tree, with automatic shift detection on the branches of the tree. This function can handle missing data.

Usage

```
PhyloEM(
  phylo,
  Y_data,
  process = c("BM", "OU", "scOU", "rBM"),
  check_postorder = TRUE,
  independent = FALSE,
  K_max = max(floor(sqrt(length(phylo$tip.label))), 10),
  use_previous = FALSE,
  order = TRUE,
  method.selection = c("LINselect", "DDSE", "Djump"),
  C.BM1 = 0.1,
  C.BM2 = 2.5,
  C.LINselect = 1.1,
  method.variance = c("upward_downward", "simple"),
  method.init = "lasso",
  method.init.alpha = "estimation",
  method.init.alpha.estimation = c("regression", "regression.MM", "median"),
  methods.segmentation = c("lasso", "best_single_move"),
  alpha_grid = TRUE,
  nbr_alpha = 10,
  random.root = TRUE,
  stationary.root = random.root,
  alpha = NULL,
  check.tips.names = TRUE,
  progress.bar = TRUE,
  estimates = NULL,
  save_step = FALSE,
  rescale_OU = TRUE,
  parallel_alpha = FALSE,
  Ncores = 3,
  K_lag_init = 5,
  light_result = TRUE,
  tol_tree = .Machine$double.eps^0.5,
  allow_negative = FALSE,
  option_is.ultrametric = 1,
  trait_correlation_threshold = 0.9,
  ...
)
```

Arguments

`phylo`	A phylogenetic tree of class `phylo` (from package ape).
`Y_data`	Matrix of data at the tips, size p x ntaxa. Each line is a trait, and each column is a tip. The column names are checked against the tip names of the tree.
`process`	The model used for the fit. One of "BM" (for a full BM model, univariate or multivariate); "OU" (for an OU with independent traits, univariate or multivariate); or "scOU" (for a "scalar OU" model, see details).
`check_postorder`	Re-order the tree in post-order. If the Upward-Downward algorithm is used, the tree need to be in post-order. Default to TRUE if the upward-downward is used, otherwise automatically set to FALSE.
`independent`	Are the trait assumed to be independent from one another? Default to FALSE. OU in a multivariate setting only works if TRUE.
`K_max`	The maximum number of shifts to be considered. Default to $max(\lfloor\sqrt{n}taxa\rfloor, 10)$.
`use_previous`	Should the initialization for K+1 shifts use the estimation for $K$ shifts already obtained? Default to FALSE.
`order`	Should the estimations be done for K increasing (TRUE) or K decreasing (FALSE)? If use_previous=FALSE, this has no influence, except if one initialization fails. Default to TRUE.
`method.selection`	Method selection to be used. Several ones can be used at the same time. One of "LINselect" for the Baraud Giraud Huet LINselect method; "DDSE" for the Slope Heuristic or "Djump" for the Jump Heuristic, last two based the Birgé Massart method.
`C.BM1`	Multiplying constant to be used for the BigeMassart1 method. Need to be positive. Default to 0.1.
`C.BM2`	Multiplying constant to be used for the BigeMassart2 method. Default to 2.5.
`C.LINselect`	Multiplying constant to be used for the LINselect method. Need to be greater than 1. Default to 1.1.

`method.variance`	Algorithm to be used for the moments computations at the E step. One of "simple" for the naive method; of "upward_downward" for the Upward Downward method (usually faster). Default to "upward_downward".
`method.init`	The initialization method. One of "lasso" for the LASSO base initialization method; or "default" for user-specified initialization values. Default to "lasso".
`method.init.alpha`	For OU model, initialization method for the selection strength alpha. One of "estimation" for a cherry-based initialization, using `nlrob`; or "default" for user-specified initialization values. Default to "estimation".
`method.init.alpha.estimation`	If method.init.alpha="estimation", choice of the estimation(s) methods to be used. Choices among "regression", (method="M" is passed to `nlrob`); "regression.MM" (method="MM" is passed to `nlrob`) or "median" (`nlrob` is not used, a simple median is taken). Default to all of them.
`methods.segmentation`	For OU, method(s) used at the M step to find new candidate shifts positions. Choices among "lasso" for a LASSO-based algorithm; and "best_single_move" for a one-move at a time based heuristic. Default to both of them. Using only "lasso" might speed up the function a lot.
`alpha_grid`	whether to use a grid for alpha values. Default to TRUE. This is the only available method for scOU. This method is not available for OU with multivariate traits. OU with univariate traits can take both TRUE or FALSE. If TRUE, a grid based on the branch length of the tree is automatically computed, using function `find_grid_alpha`.
`nbr_alpha`	If `alpha_grid=TRUE`, the number of alpha values on the grid. Default to 10.
`random.root`	whether the root is assumed to be random (TRUE) of fixed (FALSE). Default to TRUE
`stationary.root`	whether the root is assumed to be in the stationary state. Default to TRUE.
`alpha`	If the estimation is done with a fixed alpha (either known, or on a grid), the possible value for alpha. Default to NULL.
`check.tips.names`	whether to check the tips names of the tree against the column names of the data. Default to TRUE.
`progress.bar`	whether to display a progress bar of the computations. Default to TRUE.
`estimates`	The result of a previous run of this same function. This function can be re-run for other model election method. Default to NULL.
`save_step`	If alpha_grid=TRUE, whether to save the intermediate results for each value of alpha (in a temporary file). Useful for long computations. Default to FALSE.
`rescale_OU`	For the Univariate OU, should the tree be re-scaled to use a BM ? This can speed up the computations a lot. However, it can make it harder for the EM to explore the space of parameters, and hence lead to a sub-optimal solution. Default to TRUE.

`parallel_alpha`	If alpha_grid=TRUE, whether to run the estimations with different values of alpha on separate cores. Default to FALSE. If TRUE, the log is written as a temporary file.
`Ncores`	If parallel_alpha=TRUE, number of cores to be used.
`K_lag_init`	Number of extra shifts to be considered at the initialization step. Increases the accuracy, but can make computations quite slow of taken too high. Default to 5.
`light_result`	if TRUE (the default), the object returned is made light, without easily computable quantities. If FALSE, the object can be very heavy, but its subsequent manipulations can be faster (especially for plotting).
`tol_tree`	tolerance to consider a branch length significantly greater than zero, or two lineages lengths to be different, when checking for ultrametry. (Default to .Machine$double.eps^0.5). See `is.ultrametric` and `di2multi`.
`allow_negative`	whether to allow negative values for alpha (Early Burst). See details. Default to FALSE.
`option_is.ultrametric`	option for `is.ultrametric` check. Default to 1.
`trait_correlation_threshold`	the trait correlation threshold to stop the analysis. Default to 0.9.
`...`	Further arguments to be passed to `estimateEM`, including tolerance parameters for stopping criteria, maximal number of iterations, etc.

Details

Several models can be used:

- BM with fixed root, univariate or multivariate.
- OU with fixed or stationary root, univariate or multivariate.

For the OU in the multivariate setting, two assumptions can be made:

- Independent traits. This amounts to diagonal rate and selection matrices.
- "Scalar OU" (scOU): the rate matrix can be full, but the selection strength matrix is assumed to be scalar, i.e. all the traits are supposed to go to their optimum values with the same speed.

Note that the "scalar OU" model can also be seen as a re-scaling of the tree. The selection strength parameter alpha can then be interpreted as a measure of the "phylogenetic signal":

- If alpha is close to 0, then the process is similar to a BM on the original tree, and the signal is strong.
- If alpha is large, then the re-scaled tree is similar to a star-tree, and the signal is weak.

When there are no shifts, and the root is taken to be constant, this model is actually equivalent to an AC model (Uyeda et al. 2015). With this interpretation in mind, one might want to explore negative values of alpha, in order to fit a DC (or Early Burst) model. With no shift and a fixed root, the same proof shows that the scOU with alpha negative is equivalent to the DC model. There are two strong caveats in doing that.

- The interpretation of the OU as modeling the dynamic of a trait undergoing stabilizing selection is lost. In this case, the scOU can only be seen as a re-scaling of the tree, similar to Pagel's delta.
- The values of the "optimal values", and of the shifts on them, cannot be interpreted as such (the process is actually going away from this values, instead of being attracted). When looking at these values, one should only use the un-normalized values happening of the underlying BM. You can extract those using the `params_process` function with `rBM` = `TRUE`.

Value

An object of class `PhyloEM`. Relevant quantities can be extracted from it using helper functions `params_process.PhyloEM`, `imputed_traits.PhyloEM`

See Also

`plot.PhyloEM`, `params_process.PhyloEM`, `imputed_traits.PhyloEM`

Examples

```
## Not run:
## Load Data
data(monkeys)
## Run method
# Note: use more alpha values for better results.
res <- PhyloEM(Y_data = monkeys$dat,        ## data
               phylo = monkeys$phy,         ## phylogeny
               process = "scOU",            ## scalar OU
               random.root = TRUE,          ## root is stationary
               stationary.root = TRUE,
               K_max = 10,                  ## maximal number of shifts
               nbr_alpha = 4,               ## number of alpha values
               parallel_alpha = TRUE,       ## parallelize on alpha values
               Ncores = 2)
## Plot selected solution (LINselect)
plot(res) # three shifts
## Plot selected solution (DDSE)
plot(res, method.selection = "DDSE") # no shift
## Extract and solution with 5 shifts
params_5 <- params_process(res, K = 5)
plot(res, params = params_5)
## Show all equivalent solutions
eq_sol <- equivalent_shifts(monkeys$phy, params_5)
plot(eq_sol)

## End(Not run)
```

plot.enumerate_parsimony

Plot all the equivalent solutions.

Description

`plot.enumerate_parsimony` plots a representation of all the equivalent solutions.

Usage

```
## S3 method for class 'enumerate_parsimony'
plot(x, numbering = FALSE, nbr_col = 3, ...)
```

Arguments

`x`	an object of class `enumerate_parsimony`, result of function `enumerate_parsimony`
`numbering`	whether to number the solutions. Default to FALSE.
`nbr_col`	the number of columns on which to display the plot. Default to 3.
`...`	further arguments to be passed to `plot.phylo` or `nodelabels`.

Details

This function uses functions `plot.phylo` and `nodelabels` for the actual plotting of the trees.

Value

A plot of the equivalent shifts allocations.

See Also

`plot.phylo`, `enumerate_parsimony`, `plot.equivalent_shifts`, `nodelabels`

plot.equivalent_shifts

Plot all the equivalent solutions.

Description

`plot.equivalent_shifts` plots a representation of all the equivalent shifts allocations, with a representation of the shifts and their values, and a coloration of the branches in term of regimes.

Usage

```
## S3 method for class 'equivalent_shifts'
plot(
  x,
  trait = 1,
  show_shifts_values = TRUE,
  numbering = FALSE,
  colors_tips = NULL,
  nbr_col = 3,
  gray_scale = FALSE,
  edge.width = 2,
  shifts_cex = 1.2,
  ...
)
```

Arguments

`x`	an object of class `equivalent_shifts`, result of function `equivalent_shifts`
`trait`	(integer) the trait to be plotted, if multivariate. Default to 1.
`show_shifts_values`	whether to show the equivalent shifts values or not. Default to FALSE.
`numbering`	whether to number the solutions. Default to FALSE.
`colors_tips`	user-provided colors for the tips of the tree. A vector vector with as many colors as there are tips. Will be automatically computed if not provided.
`nbr_col`	the number of columns on which to display the plot. Default to 3.
`gray_scale`	if TRUE, the colors are replaced by a gray scale. Default to FALSE.
`edge.width`	width of the edge. Default to 1.
`shifts_cex`	if `value_in_box=TRUE`, the size of the text in the boxes. Default to 0.8.
`...`	further arguments to be passed to `plot.phylo`.

Details

This function uses function `plot.phylo` for the actual plotting of the trees.

Value

A plot of the equivalent shifts allocations.

See Also

`equivalent_shifts`, `plot.phylo`

plot.params_process *Plot for class* simul_process

Description

This function takes an object of class params_process, and plots them along with some data at the tips of the tree.

Usage

```
## S3 method for class 'params_process'
plot(
  x,
  phylo,
  data = NULL,
  traits,
  automatic_colors = TRUE,
  color_characters = "black",
  color_edges = "black",
  plot_ancestral_states = FALSE,
  ancestral_states = NULL,
  imposed_scale,
  ancestral_cex = 2,
  ancestral_pch = 19,
  value_in_box = FALSE,
  ancestral_as_shift = FALSE,
  shifts_cex = 0.6,
  shifts_bg = "chocolate4",
  root_bg = "chocolate4",
  shifts_adj = 0,
  root_adj = 1,
  color_shifts_regimes = FALSE,
  regime_boxes = FALSE,
  alpha_border = 70,
  show.tip.label = FALSE,
  label_cex = 0.5,
  label_offset = 0,
  axis_cex = 0.7,
  edge.width = 1,
  margin_plot = NULL,
  gray_scale = FALSE,
  ...
)
```

Arguments

x	an object of class params_process.

`phylo`	a phylogenetic tree.
`data`	a matrix of data at the tips of the tree. Must have p rows and ntaxa columns. If these are simulated, use the `extract.simul_process` function.
`traits`	a vector of integers giving the numbers of the trait to be plotted. Default to 1:p (all the traits).
`automatic_colors`	whether to color the edges automatically according to their regimes. Default to TRUE. If FALSE, colors can be manually specified through arguments `color_characters` and `colro_edges` (see below).
`color_characters`	if `automatic_colors=FALSE`, a vector of colors for the tips of the tree.
`color_edges`	if `automatic_colors=FALSE`, a vector of colors for the edges of the tree.
`plot_ancestral_states`	whether to plot the ancestral traits inferred at the internal nodes of the tree. Only available if only one trait is plotted. Default to FALSE.
`ancestral_states`	if `plot_ancestral_states=TRUE`, the ancestral states must be specified. If these are simulated, use the `extract.simul_process` function.
`imposed_scale`	if `plot_ancestral_states=TRUE`, a vector specifying the imposed scale for the ancestral states plotting. Useful to make comparisons. Default to the plotted trait.
`ancestral_cex`	if `plot_ancestral_states=TRUE`, the size of the ancestral states on the tree. Default to 2.
`ancestral_pch`	if `plot_ancestral_states=TRUE`, the symbol used of the ancestral states. Default to circles (`pch=19`).
`value_in_box`	whether to plot the value of the shift in a box on the edges. Only available when only one trait is plotted. Can be difficult to read on big trees. The size of the text in the boxes is controlled by parameter. Default to FALSE.
`ancestral_as_shift`	whether to represent the ancestral value at the root as an ancestral shift on the root edge. Default to FALSE. `shifts_cex` (see below).
`shifts_cex`	if `value_in_box=TRUE`, the size of the text in the boxes. Default to 0.8.
`shifts_bg`	if `value_in_box=TRUE`, the background color of the boxes.
`root_bg`	if `value_in_box=TRUE` and `ancestral_as_shift=TRUE`, the background color of the ancestral box.
`shifts_adj`	the adj parameter for the shifts position on the edges. Default to 0 (beginning of the edge).
`root_adj`	if `ancestral_as_shift=TRUE`, the adj parameter for the ancestral value position on the root edge. Default to 1.
`color_shifts_regimes`	whether to color each shift according to its regime (default to the same color of the edge it's on). Default to FALSE.
`regime_boxes`	whether to draw a box showing all the tips below a given. The transparency of the border of the box is controlled by parameter `alpha_border` (see below).

alpha_border	if regime_boxes=TRUE, the alpha parameter of the border of the box. Default to 70.
show.tip.label	whether to show the tip labels. Default to FALSE.
label_cex	if show.tip.label=TRUE, the size of the labels. Default to 0.5.
label_offset	if show.tip.label=TRUE, the size of the offset between the tree and the labels. Default to 0.
axis_cex	cex for the label values of the plot. Default to 0.7.
edge.width	width of the edge. Default to 1.
margin_plot	vector giving the margin to around the plot. Default to c(0, 0, 0, 0).
gray_scale	if TRUE, the colors are replaced by a gray scale. Default to FALSE.
...	further arguments to be passed to plot.phylo.

See Also

simul_process, plot.PhyloEM, params_BM, params_OU

---

plot.PhyloEM *Plot for class* PhyloEM

Description

This function takes an object of class PhyloEM, result of function PhyloEM, and plots the result of the inference.

Usage

```
## S3 method for class 'PhyloEM'
plot(
  x,
  traits = 1:(x$p),
  params = NULL,
  method.selection = NULL,
  automatic_colors = TRUE,
  color_characters = "black",
  color_edges = "black",
  plot_ancestral_states = FALSE,
  name_trait = "Trait Value",
  imposed_scale,
  ancestral_cex = 2,
  ancestral_pch = 19,
  value_in_box = FALSE,
  ancestral_as_shift = FALSE,
  shifts_cex = 0.6,
  shifts_bg = "chocolate4",
  root_bg = "chocolate4",
  shifts_adj = 0,
  root_adj = 1,
  color_shifts_regimes = FALSE,
  regime_boxes = FALSE,
  alpha_border = 70,
  show.tip.label = FALSE,
  label_cex = 0.5,
  label_font = 1,
  label_offset = 0,
  axis_cex = 0.7,
  axis_las = 0,
  show_axis_traits = TRUE,
  edge.width = 1,
  margin_plot = NULL,
  gray_scale = FALSE,
  root.edge = TRUE,
  ...
)
```

Arguments

`x`	an object of class `PhyloEM`, result of function `PhyloEM`.
`traits`	a vector of integers giving the numbers of the trait to be plotted. Default to 1:p (all the traits).
`params`	(optional) some user-specified parameters. Must be of class `params_process`. If left blank, they are extracted using the `method.selection` argument (see below).
`method.selection`	select the parameters to plot. One of "LINselect", "DDSE", "Djump". Default to "LINselect". See `params_process.PhyloEM`.
`automatic_colors`	whether to color the edges automatically according to their regimes. Default to TRUE. If FALSE, colors can be manually specified through arguments `color_characters` and `colro_edges` (see below).
`color_characters`	if `automatic_colors=FALSE`, a vector of colors for the tips of the tree.
`color_edges`	if `automatic_colors=FALSE`, a vector of colors for the edges of the tree.
`plot_ancestral_states`	whether to plot the ancestral traits inferred at the internal nodes of the tree. Only available if only one trait is plotted. Default to FALSE.
`name_trait`	name of the trait scale bar for the ancestral states plotting. Default to "Trait Value".
`imposed_scale`	if `plot_ancestral_states=TRUE`, a vector specifying the imposed scale for the ancestral states plotting. Useful to make comparisons. Default to the plotted trait.
`ancestral_cex`	if `plot_ancestral_states=TRUE`, the size of the ancestral states on the tree. Default to 2.

`ancestral_pch`	if `plot_ancestral_states=TRUE`, the symbol used of the ancestral states. Default to circles (`pch=19`).
`value_in_box`	whether to plot the value of the shift in a box on the edges. Only available when only one trait is plotted. Can be difficult to read on big trees. The size of the text in the boxes is controlled by parameter. Default to FALSE.
`ancestral_as_shift`	whether to represent the ancestral value at the root as an ancestral shift on the root edge. Default to FALSE. `shifts_cex` (see below).
`shifts_cex`	if `value_in_box=TRUE`, the size of the text in the boxes. Default to 0.8.
`shifts_bg`	if `value_in_box=TRUE`, the background color of the boxes.
`root_bg`	if `value_in_box=TRUE` and `ancestral_as_shift=TRUE`, the background color of the ancestral box.
`shifts_adj`	the adj parameter for the shifts position on the edges. Default to 0 (beginning of the edge).
`root_adj`	if `ancestral_as_shift=TRUE`, the adj parameter for the ancestral value position on the root edge. Default to 1.
`color_shifts_regimes`	whether to color each shift according to its regime (default to the same color of the edge it's on). Default to FALSE.
`regime_boxes`	whether to draw a box showing all the tips below a given. The transparency of the border of the box is controlled by parameter `alpha_border` (see below).
`alpha_border`	if `regime_boxes=TRUE`, the alpha parameter of the border of the box. Default to 70.
`show.tip.label`	whether to show the tip labels. Default to FALSE.
`label_cex`	if `show.tip.label=TRUE`, the size of the labels. Default to 0.5.
`label_font`	if `show.tip.label=TRUE`, the font of the labels (see par).
`label_offset`	if `show.tip.label=TRUE`, the size of the offset between the tree and the labels. Default to 0.
`axis_cex`	cex for the label values of the plot. Default to 0.7.
`axis_las`	las for the label values of the plot. Default to 0 (see par).
`show_axis_traits`	control whether the trait values axis is plotted (default to TRUE).
`edge.width`	width of the edge. Default to 1.
`margin_plot`	vector giving the margin to around the plot. Default to `c(0, 0, 0, 0)`.
`gray_scale`	if TRUE, the colors are replaced by a gray scale. Default to FALSE.
`root.edge`	a logical indicating whether to draw the root edge (defaults to TRUE)
`...`	further arguments to be passed to `plot.phylo`.

See Also

`params_process.PhyloEM`, `imputed_traits.PhyloEM`

plot_criterion *Plot Model Selection Criterion*

Description

This function takes an object of class PhyloEM, result of function PhyloEM, and plots a model selection criterion.

Usage

```
plot_criterion(
  res,
  method.selection = NULL,
  add = FALSE,
  select.col = "red",
  ...
)
```

Arguments

res	an object of class PhyloEM, result of function PhyloEM.
method.selection	select the parameters to plot. One of "LINselect", "DDSE", "Djump" or "likelihood" (for un-penalized likelihood). Default to "LINselect". See params_process.PhyloEM.
add	boolean: should the points be added to a current plot (default to FALSE).
select.col	the color of the point selected by the criterion. Default to "red".
...	further argument to be passed to base plot.

See Also

params_process.PhyloEM, plot.PhyloEM, get_criterion

residuals.PhyloEM *Residuals of a fitted object*

Description

residuals computes the residuals of some parameters.

Usage

```
## S3 method for class 'PhyloEM'
residuals(object, ...)
```

Arguments

`object`	an object of class `params_process` or `PhyloEM`. trait, and each column is a tip. The column names are checked against the tip names of the tree. `incidence.matrix.full`. Can be specified to avoid extra computations.
`...`	for a `PhyloEM` object, further arguments to be passed on to `params_process.PhyloEM` (to choose which parameters to extract from the results, see documentation of this function).

Value

The log likelihood of the data with the provided parameters on the tree.

See Also

`params_process`, `PhyloEM`

`shifts.list_to_matrix` *Compute the matrix of shifts.*

Description

`shifts.list_to_matrix` takes the list description of the shifts to give the matrix representation of the shifts : the b th element of the lth line has the value of the shift on character l occurring on that branch b

Usage

```
shifts.list_to_matrix(phy, shifts, p = nrow(shifts$values))
```

Arguments

`phy`	Input tree.
`shifts`	list description of the shifts : shifts$edges, shifts$values.
`p`	number of traits (optional, needed when shifts = NULL).

Value

Matrix p x Nedge of length nbranch.

See Also

`shifts.matrix_to_list`

shifts.matrix_to_list *Compute the list of shifts.*

Description

shifts.matrix_to_list takes the vectorial description of the shifts to create the list description of the shifts.

Usage

```
shifts.matrix_to_list(delta)
```

Arguments

delta	matrix description of the shift.

Value

List describing shifts.

See Also

shifts.list_to_matrix

shifts_to_simmap *Simmap format mapping from list of edges*

Description

shifts_to_simmap takes a vector of edges where the shifts occur, and return a simmap formatted tree, mapped with corresponding regimes.

Usage

```
shifts_to_simmap(tree, shifts_edges)
```

Arguments

tree	input tree in phylo format
shifts_edges	shifts positions on the edges

Details

Ancestral state is always 0, and other states are consecutive integers.

Value

tree a simmap object

simul_process *Simulate a Stochastic Process on a tree*

Description

simulate simulate a stochastic process on a tree.

Usage

```
simul_process(x, ...)

## S3 method for class 'params_process'
simul_process(
  x,
  phylo,
  simulate_random = TRUE,
  checks = TRUE,
  U_tree = NULL,
  times_shared = NULL,
  ...
)

## S3 method for class 'PhyloEM'
simul_process(
  x,
  simulate_random = TRUE,
  checks = TRUE,
  U_tree = NULL,
  times_shared = NULL,
  ...
)
```

Arguments

x	an object of class params_process or PhyloEM.
...	for a PhyloEM object, further arguments to be passed on to params_process.PhyloEM (to choose which parameters to extract from the results, see documentation of this function).
phylo	a phylogenetic tree, class phylo.
simulate_random	set to FALSE if only the expected values are needed (and not the random sample). Default to TRUE.
checks	whether to check the entry parameters for consistency. Default to TRUE.
U_tree	optional, full incidence matrix of the tree, result of function incidence.matrix.full. Can be specified to avoid extra computations.
times_shared	optional, times of shared ancestry of all nodes and tips, result of function compute_times_ca. Can be specified to avoid extra computations.

Value

An S3 object of class `simul_process`. This contains:

sim_traits an array with dimensions p x Nnode x 2 (BM) or p x Nnode x 3 (OU). For each trait t, 1 <= t <= p, sim_traits[t, ,] has tree columns, containing respectively the simulated state, expected value and optimal value for all the nodes.

phylo the phylogenetic tree used for the simulations (class `phylo`).

params the parameters used for the simulations (class `params_proces`).

Methods (by class)

- `simul_process(params_process)`: `params_process` object
- `simul_process(PhyloEM)`: `PhyloEM` object

See Also

`params_process`, `PhyloEM`, `extract.simul_process`

`transform_branch_length` *Transform branch length for a re-scaled BM*

Description

Re-scale the branch length of the tree so that a BM running on the new tree produces the same observations at the tips than an OU with parameter alpha.

Usage

```
transform_branch_length(phylo, alp)
```

Arguments

`phylo`	A phylogenetic tree of class `phylo`, with branch lengths.
`alp`	Value of the selection strength.

Value

phylo The same phylogenetic tree, with transformed branch lengths.

Index