Package 'aphylo'

September 7, 2023

Title Statistical Inference and Prediction of Annotations in Phylogenetic Trees

Version 0.3-3

Description Implements a parsimonious evolutionary model to analyze and predict gene-functional annotations in phylogenetic trees as described in Vega Yon et al. (2021) <doi:10.1371/journal.pcbi.1007948>. Focusing on computational efficiency, 'aphylo' makes it possible to estimate pooled phylogenetic models, including thousands (hundreds) of annotations (trees) in the same run. The package also provides the tools for visualization of annotated phylogenies, calculation of posterior probabilities (prediction) and goodness-of-fit assessment featured in Vega Yon et al. (2021).

Depends R (>= 3.5.0), ape (>= 5.0)

LazyData true

Imports Rcpp (>= 0.12.1), Matrix, methods, coda, fmcmc, utils, MASS, xml2

Suggests covr, knitr, tinytest, AUC, rmarkdown,

VignetteBuilder knitr

LinkingTo Rcpp

RoxygenNote 7.2.3

Encoding UTF-8

URL https://github.com/USCbiostats/aphylo

BugReports https://github.com/USCbiostats/aphylo/issues

Classification/MSC 90C35, 90B18, 91D30

License MIT + file LICENSE

NeedsCompilation yes

Author George Vega Yon [aut, cre] (<https://orcid.org/0000-0002-3171-0844>), National Cancer Institute (NCI) [fnd] (Grant Number 5P01CA196569-02), USC Biostatistics [cph]

Maintainer George Vega Yon <g.vegayon@gmail.com>

Repository CRAN

Date/Publication 2023-09-07 18:40:02 UTC

R topics documented:

aphylo-package *Statistical Inference in Annotated Phylogenetic Trees*

Description

Statistical Inference in Annotated Phylogenetic Trees

accuracy_sifter *Accuracy calculation as defined in Engelhardt et al. (2011)*

Description

Uses SIFTER's 2011 definition of accuracy, where a protein is tagged as accurately predicted if the highest ranked prediction matches it.

Usage

```
accuracy_sifter(pred, lab, tol = 1e-10, highlight = "", ...)

## S3 method for class 'aphylo_estimates'
accuracy_sifter(pred, lab, tol = 1e-10, highlight = "", ...)

## Default S3 method:
accuracy_sifter(pred, lab, tol = 1e-10, highlight = "", nine_na = TRUE, ...)
```

Arguments

`pred`	A matrix of predictions, or an aphylo_estimates object.
`lab`	A matrix of labels (0,1,NA, or 9 if `nine_na = TRUE`).
`tol`	Numeric scalar. Predictions within `tol` of the max score will be tagged as the prediction made by the model (see deails).
`highlight`	Pattern passed to sprintf used to highlight predicted functions that match the observed.
`...`	Further arguments passed to the method. In the case of `aphylo_estimates`, the arguments are passed to `predict.aphylo_estimates()`.
`nine_na`	Treat 9 as NA.

Details

The analysis is done at the protein level. For each protein, the function compares the YES annotations of that proteins with the predicted by the model. The algorithm selects the predicted annotations as those that are within `tol` of the maximum score.

This algorithm doesn't take into account NOT annotations (0s), which are excluded from the analysis.

When `highlight = ""`, no highlight is done.

Value

A data frame with `Ntip()` rows and four variables. The variables are:

- Gene: Label of the gene
- Predicted: The assigned gene function.
- Observed: The true set of gene functions.
- Accuracy: The measurement of accuracy according to Engelhardt et al. (2011).

Examples

```
set.seed(81231)
atree <- raphylo(50, psi = c(0,0), P = 3)
ans <- aphylo_mcmc(atree ~ mu_d + mu_s + Pi)

accuracy_sifter(ans)
```

`ape-methods` *Available methods from the APE package*

Description

The generics `ape::Nedge()`, `ape::Nnode()`, and `ape::Ntip()` can be used directly on objects of class aphylo, aphylo_estimates, multiAphylo

Value

Integer with the number of edges, nodes, or tips accordingly.

See Also

Other information: `aphylo-info`

Examples

```
set.seed(12312)
atree <- raphylo(50, P = 2)
Nnode(atree)
Ntip(atree)
Nedge(atree)

multitree <- rmultiAphylo(10, 50, P = 2)
Nnode(multitree)
Ntip(multitree)
Nedge(multitree)
```

aphylo-class *Annotated Phylogenetic Tree*

Description

The aphylo class tree holds both the tree structure represented as a partially ordered phylogenetic tree, and node annotations. While annotations are included for both leafs and inner nodes, the algorithms included in this package only uses the leaf annotations.

Usage

```
new_aphylo(tree, tip.annotation, ...)

## S3 method for class 'phylo'
new_aphylo(
  tree,
  tip.annotation,
  node.annotation = NULL,
  tip.type = NULL,
  node.type = NULL,
  ...
)
```

Arguments

tree	An object of class phylo
tip.annotation, node.annotation	Annotation data. See aphylo.
...	Further argmuents passed to the method.
tip.type, node.type	Integer vectors with values 0,1. 0 denotes duplication node and 1 speciation node. This is used in LogLike.

Value

A list of class aphylo with the following elements:

tree	An object of class phylo.
tip.annotation	An integer matrix. Tip (leaf) nodes annotations.
node.annotation	An integer matrix (optional). Internal nodes annotations.
offspring	A list. List of offspring of each node.
pseq	Integer vector. The pruning sequence (postorder).
reduced_pseq	Integer vector. The reduced version of pseq.

Ntips.annotated	Integer. Number of tips with annotations.
tip.type	Binary of length Ntip(). 0 means duplication and 1 speciation.
tip.type	Binary of length Nnode(). 0 means duplication and 1 speciation.

See Also

Other Data management functions: aphylo_from_data_frame()

Other aphylo methods: aphylo-methods

Examples

```
# A simple example ------------------------------------------------------------

data(fakeexperiment)
data(faketree)
ans <- new_aphylo(fakeexperiment[,2:3], tree = as.phylo(faketree))

# We can visualize it
plot(ans)
```

---

aphylo-index *Indexing aphylo objects*

Description

Indexing aphylo objects

Usage

```
## S3 method for class 'aphylo'
x[i, j, drop = FALSE]

## S3 replacement method for class 'aphylo'
x[i, j] <- value
```

Arguments

x	An object of class aphylo.
i, j	Integer vector. Indices of genes or functions.
drop	Logical scalar. When TRUE, the function returns a matrix of annotations. Otherwise an object of class aphylo.
value	Integer vector. Replacing values, can be either c(0, 1, 9, NA).

Details

The subsetting method allows selecting one or more annotations from the aphylo object. Whenever i is specified, then aphylo returns the corresponding annotations.

Value

- When indexing with `i`: A data frame with the annotations of the selected genes.
- When only indexing with `j` (`drop = FALSE`): An `aphylo` object with the selected sets of annotations.
- When only indexing with `j` (`drop = TRUE`): A data.frame with the selected annotations.
- When indexing on both `i` and `j`: A data.frame with the selected genes and annotations.

Examples

```
set.seed(12312)
atree <- raphylo(50, P = 4)
atree[1:10,]
atree[,2:3]
atree[, 2:3, drop = TRUE]
atree[1:10, 2:3]
```

---

aphylo-info *Information about* `aphylo` *and* `multiAphylo` *objects*

Description

Information about annotations, in particular, number of annotations (`Nann`), number of annotated leaves (`Nannotated`), number of unnanotated leaves (`Nunannotated`), and number of trees (`Ntrees`).

Usage

```
Nann(phy)

Nannotated(phy)

Ntrees(phy)
```

Arguments

`phy`	Either an object of class aphylo, multiAphylo, or aphylo_estimates.

Value

If `phy` is of class `aphylo`, then a single scalar. otherwise, if `phy` is of class `multiAphylo`

See Also

Other information: `ape-methods`

Examples

```
# Generating data for the example
set.seed(223)
dat <- rmultiAphylo(10, n = 5, P = 2)
Nann(dat)
Nannotated(dat)
Ntrees(dat)
```

`aphylo-methods` *Plot and print methods for* `aphylo` *objects*

Description

Plot and print methods for `aphylo` objects

Usage

```
## S3 method for class 'aphylo'
plot(
  x,
  y = NULL,
  prop = 0.15,
  node.type.col = c(dupl = "black", other = "gray"),
  node.type.size = c(dupl = 0, other = 0),
  rect.args = list(),
  as_ci = NULL,
  ...
)
```

Arguments

`x`	An object of class `aphylo`.
`y`	Ignored.
`prop`	Numeric scalar between 0 and 1. Proportion of the device that the annotations use in `plot.aphylo`.
`node.type.col, node.type.size`	Vectors of length 2. In the case of `node.type.col` the color of the duplication and other nodes. `node.type.size` sets the size of circles.
`rect.args`	List of arguments passed to graphics::rect.
`as_ci`	Integer vector. Internal use only.
`...`	Further arguments passed to ape::plot.phylo.

Details

The `plot.aphylo` function is a wrapper of ape::plot.phylo.

Value

In the case of `plot.aphylo`, `NULL`.

See Also

Other aphylo methods: `aphylo-class`

Examples

```
set.seed(7172)
atree <- raphylo(20)
plot(atree)
```

`aphylo-model` *Formulas in* `aphylo`

Description

This function the the workhorse behind the likelihood function. It creates arbitrary models by modifying the call to `LogLike()` function according to what the user specifies as model.

Usage

```
eta(..., env)

psi(..., env)

Pi(..., env)

mu_d(..., env)

mu_s(..., env)

aphylo_formula(fm, params, priors, env = parent.frame())
```

Arguments

`...`	Either 0, 1 or both. Depending on the parameter, the index of the model parameter that will be set as fixed.
`env`	Environment (not to be called by the user).
`fm`	A formula. Model of the type `<aphylo-object> ~ <parameters>` (see examples).
`params`	Numeric vector with model parameters.
`priors`	(optional) A function. Prior for the model.

Value

A list with the following elements:

- `fun` A function. The log-likelihood function.
- `fixed` Logical vector.

Examples

```
set.seed(12)
x <- raphylo(10)

# Baseline model
aphylo_formula(x ~ mu_d)

# Mislabeling probabilities
aphylo_formula(x ~ mu_d + psi)

# Different probabilities for speciation and duplication node
# (only works if you have both types)
aphylo_formula(x ~ mu_d + mu_s + psi)

# Mislabeling probabilities and etas(fixed)
aphylo_formula(x ~ mu_d + psi + eta(0, 1))

# Mislabeling probabilities and Pi
aphylo_formula(x ~ mu_d + psi + Pi)
```

---

aphylo_cv *Leave-one-out Cross Validation*

Description

This implements Leave-one-out cross-validation (LOO-CV) for trees of class aphylo and multiAphylo.

Usage

```
aphylo_cv(...)

## S3 method for class 'formula'
aphylo_cv(model, ...)
```

Arguments

`...`	Further arguments passed to the method.
`model`	As passed to aphylo_mcmc.

Details

For each observation in the dataset (either a single gene if of class aphylo, or an entire tree if of class multiAphylo), we restimate the model removing the observation and use the parameter estimates to make a prediction on it. The prediction is done using the function predict.aphylo_estimates with argument `loo = TRUE`.

Value

An object of class `aphylo_cv` with the following components:

- `pred_out` Out of sample prediction.
- `expected` Expected annotations
- `call` The call
- `ids` Integer vector with the ids of the leafs used in the loo process.

Examples

```
# It takes about two minutes to run this example


  set.seed(123)
  atrees <- rmultiAphylo(10, 10, P = 1)

  cv_multi  <- aphylo_cv(atrees ~ mu_d + mu_s + Pi)
  cv_single <- aphylo_cv(atrees[[1]] ~ mu_d + mu_s + Pi)
```

APHYLO_DEFAULT_MCMC_CONTROL

Model estimation using Markov Chain Monte Carlo

Description

The function is a wrapper of `fmcmc::MCMC()`.

Usage

```
APHYLO_DEFAULT_MCMC_CONTROL

aphylo_mcmc(
  model,
  params,
  priors = uprior(),
  control = list(),
  check_informative = getOption("aphylo_informative", FALSE),
  reduced_pseq = getOption("aphylo_reduce_pseq", TRUE)
```

)

APHYLO_PARAM_DEFAULT

Arguments

model	A model as specified in aphylo-model.
params	A vector of length 7 with initial parameters. In particular psi[1], psi[2], mu[1], mu[2], eta[1], eta[2] and Pi.
priors	A function to be used as prior for the model (see bprior).
control	A list with parameters for the optimization method (see details).
check_informative	Logical scalar. When TRUE the algorithm stops with an error when the annotations are uninformative (either 0s or 1s).
reduced_pseq	Logical. When TRUE it will use a reduced peeling sequence in which it drops unannotated leafs. If the model includes eta this is set to FALSE.

Format

An object of class list of length 6.

An object of class numeric of length 9.

Details

APHYLO_DEFAULT_MCMC_CONTROL lists the default values for the MCMC estimation:

- nsteps: 1e4L
- burnin: 5e3L
- thin : 10L
- nchains : 2L
- multicore : FALSE
- conv_checker : fmcmc::convergence_auto(5e3)

For more information about the MCMC estimation process, see fmcmc::MCMC().

Methods base::print(), base::summary(), stats::coef, stats::window(), stats::vcov(), stats::logLik(), predict(), and the various ways to query features of the trees via Ntip() are available post estimation.

The vector APHYLO_PARAM_DEFAULT lists the starting values for the parameters in the model. The current defaults are:

- psi0: 0.10
- psi1: 0.05
- mu_d0: 0.90
- mu_d1: 0.50
- mu_s0: 0.10
- mu_s1: 0.05

- `eta0`: 1.00
- `eta1`: 1.00
- `Pi`: 0.50

Value

An object of class aphylo_estimates.

See Also

Other parameter estimation: `aphylo_mle()`

Examples

```
# Using the MCMC ------------------------------------------------------------



set.seed(1233)
# Simulating a tree
tree <- sim_tree(200)

# Simulating functions
atree <- raphylo(
  tree = tree,
  psi  = c(.01, .03),
  mu_d = c(.05, .02),
  Pi   = .5
)

# Running the MCMC
set.seed(1231)

ans_mcmc <- aphylo_mcmc(
  atree ~ mu_d + psi + eta + Pi,
  control = list(nsteps = 2e5, burnin=1000, thin=200)
)
```

`aphylo_estimates` *Objects of class* `aphylo_estimates`

Description

The model fitting of annotated phylogenetic trees can be done using either MLE via `aphylo_mle()` or MCMC via `aphylo_mcmc()`. This section describes the object of class `aphylo_estimates` that these functions generate and the post estimation methods/functions that can be used.

Usage

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

## S3 method for class 'aphylo_estimates'
coef(object, ...)

## S3 method for class 'aphylo_estimates'
vcov(object, ...)

## S3 method for class 'aphylo_estimates'
plot(
  x,
  y = NULL,
  which.tree = 1L,
  ids = list(1:Ntip(x)[which.tree]),
  loo = TRUE,
  nsamples = 1L,
  ncores = 1L,
  centiles = c(0.025, 0.5, 0.975),
  cl = NULL,
  ...
)
```

Arguments

`x, object`	Depending of the method, an object of class `aphylo_estimates`.
`...`	Further arguments passed to the corresponding method.
`y`	Ignored.
`which.tree`	Integer scalar. Which tree to plot.
`ids, nsamples, ncores, centiles, cl`	passed to `predict.aphylo_estimates()`
`loo`	Logical scalar. When `loo = TRUE`, predictions are preformed similar to what a leave-one-out cross-validation scheme would be done (see predict.aphylo_estimates).

Details

The plot method for the object of class `aphylo_estimates` plots the original tree with the predicted annotations.

Value

Objects of class `aphylo_estimates` are a list withh the following elements:

`par`	A numeric vector of length 5 with the solution.
`hist`	A numeric matrix of size `counts*5` with the solution path (length 2 if used `optim` as the intermediate steps are not available to the user). In the case of `aphylo_mcmc`, `hist` is an object of class `coda::mcmc.list()`.

ll	A numeric scalar with the value of fun(par, dat). The value of the log likelihood.
counts	Integer scalar number of steps/batch performed.
convergence	Integer scalar. Equal to 0 if optim converged. See optim.
message	Character scalar. See optim.
fun	A function (the objective function).
priors	If specified, the function priors passed to the method.
dat	The data dat provided to the function.
par0	A numeric vector of length 5 with the initial parameters.
method	Character scalar with the name of the method used.
varcovar	A matrix of size 5*5. The estimated covariance matrix.

The plot method for aphylo_estimates returns the selected tree (which.tree) with predicted annotations, also of class aphylo.

Examples

```
set.seed(7881)
atree <- raphylo(40, P = 2)
res   <- aphylo_mcmc(atree ~ mu_d + mu_s + Pi)

print(res)
coef(res)
vcov(res)
plot(res)
```

---

aphylo_from_data_frame

Create an aphylo *object with partial annotations*

Description

Create an aphylo object with partial annotations

Usage

```
aphylo_from_data_frame(tree, annotations, types = NULL)
```

Arguments

tree	An object of class phylo.
annotations	A data.frame with annotations. The first column should be the gene id (see details).
types	A data.frame with types. Just like the annotations, the first column should be the gene id.

Details

Each row in the the `annotations` data frame passed to this function must have a unique row per gene, and one column per function (GO term). The id of each gene must match the labels in the `tree` object. Missing genes are annotated with `NA` (9).

In the case of `types`, while tips can also be annotated with a type, which should be either 0, duplication, or 1, speciation, only internal nodes are required. Tip types are ignored.

Value

An object of class aphylo.

See Also

Other Data management functions: `aphylo-class`

Examples

```
# Generating a test dataset
set.seed(1371)
x <- raphylo(20)

# Extracting the tree and annotations
tree <- x$tree

anno <- with(x, rbind(tip.annotation, node.annotation))
anno <- data.frame(id = with(tree, c(tip.label, node.label)), anno)

types <- data.frame(id = tree$node.label, x$node.type)

# Creating a aphylo tree without node types
aphylo_from_data_frame(tree, anno)

# Now including types
aphylo_from_data_frame(tree, anno, types)

# Dropping some data
aphylo_from_data_frame(tree, anno[sample.int(nrow(anno), 10),])
```

---

aphylo_mle *Model estimation using Maximum Likelihood Estimation*

Description

The function is a wrapper of `stats::optim()`.

Usage

```
aphylo_mle(
  model,
  params,
  method = "L-BFGS-B",
  priors = function(p) 1,
  control = list(),
  lower = 1e-05,
  upper = 1 - 1e-05,
  check_informative = getOption("aphylo_informative", FALSE),
  reduced_pseq = getOption("aphylo_reduce_pseq", TRUE)
)
```

Arguments

`model`	A model as specified in aphylo-model.
`params`	A vector of length 7 with initial parameters. In particular `psi[1]`, `psi[2]`, `mu[1]`, `mu[2]`, `eta[1]`, `eta[2]` and `Pi`.
`method, control, lower, upper`	Arguments passed to `stats::optim()`.
`priors`	A function to be used as prior for the model (see bprior).
`check_informative`	Logical scalar. When `TRUE` the algorithm stops with an error when the annotations are uninformative (either 0s or 1s).
`reduced_pseq`	Logical. When `TRUE` it will use a reduced peeling sequence in which it drops unannotated leafs. If the model includes `eta` this is set to `FALSE`.

Details

The default starting parameters are described in APHYLO_PARAM_DEFAULT.

Value

An object of class aphylo_estimates.

See Also

Other parameter estimation: `APHYLO_DEFAULT_MCMC_CONTROL`

Examples

```
# Using simulated data ------------------------------------------------------
set.seed(19)
dat <- raphylo(100)
dat <- rdrop_annotations(dat, .4)

# Computing Estimating the parameters
```

```
ans  <- aphylo_mle(dat ~ psi + mu_d + eta + Pi)
ans

# Plotting the path
plot(ans)

# Computing Estimating the parameters Using Priors for all the parameters
mypriors <- function(params) {
    dbeta(params, c(2, 2, 2, 2, 1, 10, 2), rep(10, 7))
}

ans_dbeta <- aphylo_mle(dat ~ psi + mu_d + eta + Pi, priors = mypriors)
ans_dbeta
```

`as.phylo` *Extensions to the* `as.phylo` *function*

Description

This function takes an edgelist and recodes (relabels) the nodes following **ape**'s coding convention.

Usage

```
## S3 method for class 'matrix'
as.phylo(x, edge.length = NULL, root.edge = NULL, ...)

## S3 method for class 'aphylo'
as.phylo(x, ...)
```

Arguments

`x`	Either an edgelist or an object of class aphylo.
`edge.length`	A vector with branch lengths (optional).
`root.edge`	A numeric scalar with the length for the root node (optional).
`...`	Further arguments passed to the method.

Value

An integer matrix of the same dimmension as `edges` with the following aditional attribute:

`labels`	Named integer vector of size `n`. Original labels of the edgelist where the first `n` are leaf nodes, `n+1` is the root node, and the reminder are the internal nodes.

Examples

```
# A simple example ----------------------------------------------------------
# This tree has a coding different from ape's

mytree <- matrix(c(1, 2, 1, 3, 2, 4, 2, 5), byrow = TRUE, ncol=2)
mytree

ans <- as.phylo(mytree)
ans
plot(ans)
```

---

auc *Area Under the Curve and Receiving Operating Curve*

Description

The AUC values are computed by approximation using the area of the polygons formed under the ROC curve.

Usage

```
auc(pred, labels, nc = 200L, nine_na = TRUE)

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

## S3 method for class 'aphylo_auc'
plot(x, y = NULL, ...)
```

Arguments

pred	A numeric vector with the predictions of the model. Values must range between 0 and 1.
labels	An integer vector with the labels (truth). Values should be either 0 or 1.
nc	Integer. Number of cutoffs to use for computing the rates and AUC.
nine_na	Logical. When TRUE, 9 is treated as NA.
x	An object of class aphylo_auc.
...	Further arguments passed to the method.
y	Ignored.

Value

A list:

- tpr A vector of length nc with the True Positive Rates.
- tnr A vector of length nc with the True Negative Rates.
- fpr A vector of length nc with the False Positive Rates.
- fnr A vector of length nc with the False Negative Rates.
- auc A numeric value. Area Under the Curve.
- cutoffs A vector of length nc with the cutoffs used.

Examples

```
set.seed(8381)
x   <- rdrop_annotations(raphylo(50), .3)
ans <- aphylo_mcmc(x ~ mu_d + mu_s + Pi)
ans_auc <- auc(predict(ans, loo = TRUE), x[,1,drop=TRUE])
print(ans_auc)
plot(ans_auc)
```

---

balance_ann *Functional balance of a tree*

Description

This function computes the distance between .5 and the observed proportion of ones for each function in a tree.

Usage

```
balance_ann(phy)
```

Arguments

phy	An object of class aphylo or multiAphylo

Details

Functional balance is defined as follows

$$P^{-1}\sum_p\left(1-\left|0.5-N^{-1}\sum_n a_{np}\right|\right)$$

Where A is the matrix of annotations.

With values ranging between 0 and 1, one been perfect balance, this is, equal number of zeros and ones in the annotations. In the case of multiple functions, as noted in the formula, the balance is the average across functions.

Value

If phy is an object of class phylo, a single scalar, otherwise, it returns a vector of length Ntrees(phy).

Examples

```
x <- raphylo(20, P = 2)
balance_ann(x)

balance_ann(c(x, x))
```

---

bprior *Default priors for aphylo_mcmc*

Description

Convenient wrappers to be used with the aphylo estimation methods.

Usage

```
bprior(shape1 = 1, shape2 = 9, ...)

uprior()
```

Arguments

shape1, shape2, ...
: Arguments passed to stats::dbeta

Value

In the case of bprior, a wrapper of the function stats::dbeta. uprior returns a function function(p) 1 (the uniform prior)

Examples

```
bprior(1, 9)
uprior()
```

dist2root *Pointer to* pruner

Description

Creates an external pointer to an object of class aphylo_pruner. This is mostly used to compute the model's likelihood function faster by reusing underlying C++ class objects to store probability matrices and data. This is intended for internal use only.

Usage

```
dist2root(ptr)

get_postorder(ptr)

new_aphylo_pruner(x, ...)
```

Arguments

ptr	An object of class aphylo_pruner.
x	An object of class aphylo or multiAphylo.
...	Further arguments passed to the method

Details

The underlying implementation of the pruning function is based on the pruner C++ library that implements Felsenstein's tree pruning algorithm. See https://github.com/USCbiostats/pruner.

Value

dist2root: An integer vector with the number of steps from each node (internal or not) to the root node.

get_postorder: An integer vector with the postorder sequence for pruning the tree (indexed from 0).

The function new_aphylo_pruner returns an object of class aphylo_pruner or multiAphylo_pruner, depending on the class of x.

Examples

```
set.seed(1)
x  <- raphylo(20)
pruner <- new_aphylo_pruner(x)

# Computing loglike
LogLike(
  pruner,
  psi  = c(.10, .20),
```

```
  mu_d = c(.90, .80),
  mu_s = c(.10, .05),
  Pi   = .05,
  eta  = c(.90, .80)
  )

dist2root(pruner)
get_postorder(pruner)
```

`fakeexperiment` *Fake Experimental Data*

Description

A fake dataset containing 2 functional state of the leaf nodes. Each function can have either 0 (unactive), 1 (active) or 9 (n/a). This dataset is inteded for testing only.

Format

A data frame with 4 rows and 3 variables:

f1 State of function 1.

f2 State of function 1.

LeafId Integer, ID of the leaf.

Source

BiostatsUSC

`faketree` *Fake Phylogenetic Tree*

Description

A fake dataset containing the parent-offspring relations between genes. This dataset is inteded for testing only.

Format

A data frame with 6 rows and 2 variables:

NodeId Integer, ID of the offspring.

ParentId Integer, ID of the parent.

Source

BiostatsUSC

`imputate_duplications` *Impute duplication events based on a vector of species*

Description

Uses a simple algorithm to impute duplication events based on the terminal genes of the tree. An interior node is a duplication event if a specie has two or more leafs within its clade.

Usage

```
imputate_duplications(tree, species)
```

Arguments

`tree`	An object of class ape::phylo.
`species`	A character vector of length `ape::Ntip(tree)` (see details).

Details

This function will take a vector of species and, based on that, assign duplication events throughout the interior nodes. An interior node is labeled as a duplication event if two or more of the leaves within it are from the same species.

Value

A logical vector of length `ape::Nnode(tree, internal.only = FALSE)` with `TRUE` to indicate that the corresponding node is a duplication event. The order matches that in the input tree.

Examples

```
# Data from PANTHER
path <- system.file("tree.tree", package="aphylo")
ptree <- read_panther(path)

# Extracting the species
sp <- gsub(".+[:]|[|].+", "" , ptree$tree$tip.label)

# Imputing duplications
imputate_duplications(ptree$tree, species = sp)
```

list_offspring *List each nodes' offspring or parent*

Description

For each node in a tree, the functions `list_offspring` and `list_parents` lists all its offspring and parents, respectively.

Usage

```
list_offspring(x)

list_parents(x)
```

Arguments

`x`	An object of class `phylo` or `aphylo`.

Value

List of length `n` (total number of nodes).

Examples

```
# A simple example with phylo tree ----------------------------------------

set.seed(4)
x <- ape::rtree(10)
list_offspring(x)
```

LogLike *Likelihood of an observed annotated phylogenetic tree*

Description

This function computes the log-likelihood of the chosen parameters given a particular dataset. The arguments `annotations`, and `offspring` should be as those returned by `new_aphylo()`. For complete parameter estimation see aphylo_estimates.

Usage

```
LogLike(tree, psi, mu_d, mu_s, eta, Pi, verb_ans = TRUE, check_dims = TRUE)
```

Arguments

`tree`	A phylogenetic tree of class aphylo.
`psi`	Numeric vector of length 2. Misclasification probabilities. (see `LogLike`).
`mu_d, mu_s`	Numeric vector of length 2. Gain/loss probabilities (see `LogLike`).
`eta`	Numeric vector of length 2. Annotation bias probabilities (see `LogLike`).
`Pi`	Numeric scalar. Root node probability of having the function (see `LogLike`).
`verb_ans`	Logical scalar. When `FALSE` (default) the function returns a list with a single scalar (the log-likelihood).
`check_dims`	Logical scalar. When `TRUE` (default) the function checks the dimmension of the passed parameters.

Details

The parameters to estimate are described as follows:

1. `psi`: A vector of length 2 with ψ_0 and ψ_1, which are the misclassification probabilities fo $s_p = 0$ and $s_p = 1$ respectively.
2. `mu_d`, `mu_s`: A vector of length 2 with μ_0 and μ_1 which are the gain and loss probabilities respectively. The subscript d denotes duplication nodes and s speciation node.
3. `eta`: A vector of length 2 with η_0 and η_1 which are the annotation bias probabilities.
4. `Pi`: A numeric scalar which for which equals the probability of the root node having the function.

Value

A list of class `phylo_LogLik` with the following elements:

`S`	An integer matrix of size $2^p \times p$ as returned by `states`.
`Pr`	A numeric matrix of size $G \times 2^p$ with node/state probabilities.
`ll`	A numeric scalar with the log-likelihood value given the chosen parameters.

`mislabel` *Switch labels acoording to mislabeling probabilities*

Description

Switch labels acoording to mislabeling probabilities

Usage

```
mislabel(atree, psi)
```

Arguments

atree	An object of class aphylo.
psi	Numeric vector of length 2. Misclasification probabilities. (see LogLike).

Value

An object of class aphylo with modified labels.

Examples

```
set.seed(131)
x <- raphylo(5, P=2, psi=c(0,0))
x$tip.annotation

# Flipping 0s to 1s and vice versa
mislabel(x, psi = c(1,1))$tip.annotation
```

---

multiAphylo *Building Lists of Annotated Trees*

Description

This is equivalent to what ape::c.phylo does.

Usage

```
## S3 method for class 'aphylo'
c(...)

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

Arguments

...	One or several object of class aphylo or multiAPhylo. Ignored in the case of print.multiAphylo.
x	An object of class multiAphylo

Value

A list of class multiAphylo. Each element corresponds to a single aphylo object.

Examples

```
data(fakeexperiment)
data(faketree)
ans <- new_aphylo(fakeexperiment[,2:3], tree = as.phylo(faketree))
c(ans, ans)
```

`panther-tree` *Reads PANTHER db trees*

Description

The PANTHER Project handles a modified version of newick tree files which, besides of the tree structure, includes the type of node and ancestor labels. This function is a wrapper of `ape::read.tree()`.

Usage

```
read_panther(x, tree.reader = ape::read.tree, ...)

read.panther(x, tree.reader = ape::read.tree, ...)
```

Arguments

`x`	Character scalar. Full path to the panther file.
`tree.reader`	Function that will be used to read the tree file. It can be either `ape::read.tree` or `rncl::read_newick_phylo`.
`...`	Further arguments passed to `ape::read.tree()`.

Value

A list consisting of a data.frame and a `phylo` object. The data.frame has the following columns:

`branch_length`	Numeric vector. Length of the branch to its parent node.
`type`	Character vector. Can be either `"S"` (speciation), `"D"` (duplication), or `"T"` (horizontal transfer).
`ancestor`	Character vector. Name of the ancestor.

The nodeids can be identified using the rownames.

See Also

Other reading: `read_nhx()`, `read_pli()`

Examples

```
path <- system.file("tree.tree", package="aphylo")
read_panther(path)
```

plot.aphylo_prediction_score *Visualize predictions*

Description

Visualize predictions

Usage

```
## S3 method for class 'aphylo_prediction_score'
plot(
  x,
  y = NULL,
  main = "Prediction Accuracy: Observed versus predicted values",
  main.colorkey = "Probability of Functional Annotation",
  which.fun = seq_len(ncol(x$expected)),
  include.labels = NULL,
  labels.col = "black",
  leafs_only = TRUE,
  ...
)
```

Arguments

x	An object of class aphylo_prediction_score.
y	Ignored.
main	Passed to title.
main.colorkey	Character scalar. Title of the colorkey (optional).
which.fun	Integer vector. Which function to plot.
include.labels	Logical scalar. When TRUE, draws nice labels at each slice which by default are specified as the rownames of x$expected. This is mostly useful when the number of predictions is small.
labels.col	Character scalar. Color of the labels.
leafs_only	Logical. When TRUE (default) only plots the leaf nodes.
...	Ignored

Details

If include.labels = NULL and ncol(x$expected) > 40, then include.labels=FALSE by default.

Value

NULL (invisible) Generates a plot of the predictions.

Examples

```
set.seed(8783)
atree  <- raphylo(29)
ans    <- aphylo_mle(atree ~ mu_d + mu_s + Pi)
pred_s <- prediction_score(ans)

pred_s
plot(pred_s)
```

`plot_logLik` *Plot Log-Likelihood function of the model*

Description

Plot Log-Likelihood function of the model

Usage

```
plot_logLik(x, sets, ...)

## S3 method for class 'aphylo'
plot_logLik(x, sets, ...)

## S3 method for class 'formula'
plot_logLik(x, sets, ...)

## S3 method for class 'aphylo_estimates'
plot_logLik(x, sets, ...)
```

Arguments

`x`	An object of class `aphylo()`
`sets`	(optional) Character matrix of size `2 x # of combinations`. contains the names of the pairs to plot. If nothing passed, the function will generate all possible combinations as `combn(names(params), 2)`.
`...`	Aditional parameters to be passed to `plotfun`.

Value

NULL (invisible). Generates a plot of the loglikelihood of the model.

Examples

```
# Loading data
data(fakeexperiment)
data(faketree)
O <- new_aphylo(fakeexperiment[,2:3], tree = as.phylo(faketree))
```

```
# Baseline plot (all parameters but Pi)
plot_logLik(O)

# No psi parameter
plot_logLik(O ~ mu_d + Pi + eta)
```

`plot_multivariate` *Multiavariate plot (surface)*

Description

Multiavariate plot (surface)

Usage

```
plot_multivariate(
  fun,
  params,
  domain,
  sets,
  nlevels = 20,
  args = list(),
  plotfun = graphics::image,
  plot = TRUE,
  postplot = function(params, res) {
     points(params, cex = 2, pch = 3, col = "red")

    },
  mfrow = NULL,
  ...
)
```

Arguments

`fun`	A function that receives 2 or more parameters and returns a single number.
`params`	Numeric vector with the default parameters.
`domain`	(optional) Named list with as many elements as parameters. Specifies the domain of the function.
`sets`	(optional) Character matrix of size `2 x # of combinations`. contains the names of the pairs to plot. If nothing passed, the function will generate all possible combinations as `combn(names(params), 2)`.
`nlevels`	Integer. Number of levels.
`args`	List of named arguments to be passed to `fun`.
`plotfun`	Function that will be used to plot `x,y,z`.

plot	Logical. When FALSE skips plotting.
postplot	Function to be called after plotfun. Should recieve a vector with the current parameters.
mfrow	Passed to graphics::par.
...	Further arguments passed to plotfun.

Value

A list of length length(sets), each with the following:

- x,y,z vectors of coordinates.
- xlab,ylab vectors with the corresponding labels.

Examples

```
# Example: A model with less parameters
set.seed(1231)
x <- raphylo(20)
ans <- aphylo_mcmc(
  x ~ psi + mu_d + mu_s,
  control = list(nsteps = 1e3, burnin = 0)
  )

# Creating the multivariate plot (using by default image)
plot_multivariate(
  function(...) {
    ans$fun(unlist(list(...)), priors = ans$priors, dat = ans$dat, verb_ans = FALSE)
  },
  sets = matrix(c("mu_d0", "mu_d1", "psi0", "psi1"), ncol=2),
  params = ans$par
)
```

---

posterior-probabilities

Posterior probabilities based on parameter estimates

Description

The function predict_pre_order uses a pre-order algorithm to compute the posterior probabilities, whereas the predict_brute_force computes posterior probabilities generating all possible cases.

Usage

```
## S3 method for class 'aphylo_estimates'
predict(
  object,
  which.tree = NULL,
  ids = NULL,
```

```
  newdata = NULL,
  params = stats::coef(object),
  loo = TRUE,
  nsamples = 1L,
  centiles = c(0.025, 0.5, 0.975),
  cl = NULL,
  ...
)

predict_pre_order(x, ...)

## S3 method for class 'aphylo_estimates'
predict_pre_order(
  x,
  params = stats::coef(x),
  which.tree = 1:Ntrees(x),
  ids = lapply(Ntip(x)[which.tree], seq_len),
  loo = TRUE,
  nsamples = 1L,
  centiles = c(0.025, 0.5, 0.975),
  ncores = 1L,
  cl = NULL,
  ...
)

## S3 method for class 'aphylo'
predict_pre_order(x, psi, mu_d, mu_s, eta, Pi, ...)

predict_brute_force(atree, psi, mu_d, mu_s, Pi, force = FALSE)
```

Arguments

`which.tree`	Integer scalar. Which tree to include in the prediction.
`ids`	Integer vector. Ids (positions) of the nodes that need to be predicted (see details.)
`newdata`	(optional) An aphylo object.
`params`	A numeric vector with the corresponding parameters.
`loo`	Logical scalar. When `loo = TRUE`, predictions are preformed similar to what a leave-one-out cross-validation scheme would be done (see predict.aphylo_estimates).
`nsamples`	Integer scalar. When greater than one, the prediction is done using a random sample from the MCMC chain. This only works if the model was fitted using MCMC, of course.
`centiles`	Used together with `nsamples`, this indicates the centiles to be computed from the distribution of outcomes.
`...`	Ignored.
`ncores, cl`	Passed to `parallel::makeCluster()`.
`psi`	Numeric vector of length 2. Misclasification probabilities. (see `LogLike`).

`mu_d, mu_s`	Numeric vector of length 2. Gain/loss probabilities (see `LogLike`).
`eta`	Numeric vector of length 2. Annotation bias probabilities (see `LogLike`).
`Pi`	Numeric scalar. Root node probability of having the function (see `LogLike`).
`atree, x, object`	Either a tree of class aphylo or an object of class aphylo_estimates
`force`	Logical scalar. When `TRUE` it will try to compute the brute-force probabilities for trees with more than 7 nodes.

Details

The function `predict_brute_force` is only intended for testing. For predictions after estimating the model, see predict.aphylo_estimates.

In the case of the parameter `loo` (leave-one-out), while making tip-level predictions, at each leaf the algorithm will drop annotations regarding that leaf, making its prediction using all the available information except the one include in such leaf.

The `predict_brute_force` function makes the (obviously) brute force calculation of the probabilities. It will perform It returns a list with the following:

- `Pr` The conditional probabilities of observing a tree given a particular state of the leave nodes. The size is given by (2^nnodes x 2^nleaves), each entry is read as "The probability of observing scenario i (row) given that the leaves have state j (colum)." The scenarios are specified in the `row` matrix returned by the function.
- `row` Indicates the state of each node (columns) per scenario (row).
- `col` Indicates the state of each leaf node (columns) per potential leaf scenario.

Value

In the case of the `predict` method, a `P` column numeric matrix with values between $[0, 1]$ (probabilities).

Prediction on specific nodes

The `ids` parameter indicates for which nodes, both internal and tips, the predictions should be made. By default, the function will only make predictions on the leaf nodes.

The ids follow ape's convention, this is, `1:Ntips(x)` are the leaf nodes, `Ntips(x) + 1L` is the root node, and everything else are the interior nodes.

Although the prediction algorithm is fast, indicating only a subset of the nodes could make a difference when `loo = TRUE` and/or `nsamples > 1` (calculating a Credible/Confidence Interval.)

In the case of `multiAphylo`, `ids` should be passed as a list of length `Ntrees(x)`, with each element indicating the nodes. Otherwise, ids are passed as an integer vector.

Examples

```
# Single tree ----------------------------------------------------------------
set.seed(123)
atree <- raphylo(10)
```

```
# Fitting the model with MLE
ans <- aphylo_mle(atree ~ psi + mu_d + mu_s + Pi)

# Prediction on leaves
predict(ans)

# Prediction on all nodes (including root and interior)
predict(ans, ids = 1:Nnode(ans, internal.only = FALSE))

# Multiple trees (multiAphylo) ------------------------------------------------
atree <- c(raphylo(10), raphylo(5))

# Fitting the model with MLE
ans <- aphylo_mle(atree ~ psi + mu_d + mu_s + Pi)

# Prediction on leaves
predict(ans)

# Predicting only interior nodes
predict(ans, ids = list(11:19, 6:9))
```

`prediction_score` *Calculate prediction score (quality of prediction)*

Description

Calculate prediction score (quality of prediction)

Usage

```
prediction_score(x, expected, alpha0 = NULL, alpha1 = NULL, W = NULL, ...)

## Default S3 method:
prediction_score(x, expected, alpha0 = NULL, alpha1 = NULL, W = NULL, ...)

## S3 method for class 'aphylo_estimates'
prediction_score(
  x,
  expected = NULL,
  alpha0 = NULL,
  alpha1 = NULL,
  W = NULL,
  loo = TRUE,
  ...
)

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

Arguments

x	An object of class aphylo_estimates or a numeric matrix.
expected	Integer vector of length n. Expected values (either 0 or 1).
alpha0, alpha1	Probability of observing a zero an a one, respectively.
W	A square matrix. Must have as many rows as genes in expected.
...	Further arguments passed to predict.aphylo_estimates
loo	Logical scalar. When loo = TRUE, predictions are preformed similar to what a leave-one-out cross-validation scheme would be done (see predict.aphylo_estimates).

Details

In the case of prediction_score, ... are passed to predict.aphylo_estimates.

In the case of the method for aphylo estimates, the function takes as a reference using alpha equal to the proportion of observed tip annotations that are equal to 1, this is:

```
mean(x$dat$tip.annotation[x$dat$tip.annotation != 9L], na.rm = TRUE)
```

Value

A list of class aphylo_prediction_score:

- obs : Observed 1 - MAE.
- obs_raw : Unnormalized (raw) scores.
- random_raw: Unnormalized (raw) scores.
- worse_raw : Unnormalized (raw) scores.
- pval : Computed p-value.
- worse : Reference of worse case.
- predicted : Numeric matrix with observed predictions.
- expected : Integer matrix with expected annotations.
- random : Random score (null).
- alpha0 : The passed alpha parameters.
- alpha1 : The passed alpha parameters.
- auc : An object of class aphylo_auc.
- obs.ids : Indices of the ids.
- leaf.ids : IDs of the leafs (if present).
- tree : Of class phylo.

Examples

```
# Example with prediction_score -------------------------------------------
set.seed(11552)
ap  <- raphylo(
  50, P = 1,
  Pi   = 0,
  mu_d = c(.8,.2),
  mu_s = c(0.1,0.1),
  psi  = c(0,0)
  )
ans <- aphylo_mcmc(
  ap ~ mu_d + mu_s + Pi,
  control = list(nsteps=2e3, thin=20, burnin = 500),
  priors = bprior(c(9, 1, 1, 1, 5), c(1, 9, 9, 9, 5))
  )

(pr <- prediction_score(ans, loo = TRUE))
plot(pr)
```

---

raphylo *Simulation of Annotated Phylogenetic Trees*

Description

Simulation of Annotated Phylogenetic Trees

Usage

```
raphylo(
  n = NULL,
  tree = NULL,
  edge.length = NULL,
  tip.type = NULL,
 node.type = function(n) sample.int(2, size = n, replace = TRUE, prob = c(0.2, 0.8)) - 1,
  P = 1L,
  psi = c(0.05, 0.05),
  mu_d = c(0.9, 0.5),
  mu_s = c(0.05, 0.02),
  eta = c(1, 1),
  Pi = 0.2,
  informative = getOption("aphylo_informative", FALSE),
  maxtries = 20L
)

rmultiAphylo(R, ...)
```

Arguments

n	Integer scalar. Number of leafs. If not specified, then
tree	An object of class phylo.
edge.length	Passed to sim_tree.
tip.type, node.type	Integer vectors with values 0,1. 0 denotes duplication node and 1 speciation node. This is used in LogLike.
P	Integer scalar. Number of functions to generate.
psi	Numeric vector of length 2. Misclasification probabilities. (see LogLike).
mu_d, mu_s	Numeric vector of length 2. Gain/loss probabilities (see LogLike).
eta	Numeric vector of length 2. Annotation bias probabilities (see LogLike).
Pi	Numeric scalar. Root node probability of having the function (see LogLike).
informative, maxtries	Passed to sim_fun_on_tree.
R	Integer, number of replicates
...	Further arguments passed to raphylo

Details

The rmultiAphylo function is a wrapper around raphylo.

Value

An object of class aphylo

Examples

```
# A simple example ------------------------------------------------------

set.seed(1231)
ans <- raphylo(n=500)
```

---

rdrop_annotations *Randomly drop leaf annotations*

Description

The function takes an annotated tree and randomly selects leaf nodes to set annotations as 9 (missing). The function allows specifying a proportion of annotations to drop, and also the relative probability that has dropping a 0 with respecto to a 1.

Usage

```
rdrop_annotations(
  x,
  pcent,
  prob.drop.0 = 0.5,
  informative = getOption("aphylo_informative", FALSE)
)
```

Arguments

x	An object of class aphylo.
pcent	Numeric scalar. Proportion of the annotations to remove.
prob.drop.0	Numeric scalar. Probability of removing a 0, conversely, 1 - prob.drop.0 is the probability of removing a 1.
informative	Logical scalar. If TRUE (the default) the algorithm drops annotations only if the number of annotations to drop of either 0s or 1s are less than the currently available in the data.

Value

x with fewer annotations (more 9s).

Examples

```
# The following tree has roughtly the same proportion of 0s and 1s
# and 0 mislabeling.
set.seed(1)
x <- raphylo(200, Pi=.5, mu_d=c(.5,.5), psi=c(0,0))
summary(x)

# Dropping half of the annotations
summary(rdrop_annotations(x, .5))

# Dropping half of the annotations, but 0 are more likely to drop
summary(rdrop_annotations(x, .5, prob.drop.0 = 2/3))
```

---

read_nhx *Read New Hampshire eXtended format for trees*

Description

Read New Hampshire eXtended format for trees

Usage

```
read_nhx(fn, txt)
```

Arguments

fn	Full path to the tree file.
txt	If no file is specified, trees can also be passed as a character scalar (see examples).

Value

A list with the following elements:

- tree An object of class ape
- edge Edge annotations (length and other annotations)
- nhx A list of annotations NHX

References

"NHX - New Hampshire eXtended [version 2.0]", https://en.wikipedia.org/wiki/Newick_format#New_Hampshire_X_format

See Also

Other reading: panther-tree, read_pli()

Examples

```
# Example directly extracted from
# https://sites.google.com/site/cmzmasek/home/software/forester/nhx
read_nhx(
 txt = "(((ADH2:0.1[&&NHX:S=human], ADH1:0.11[&&NHX:S=human]):0.05[&&NHX:S=primates:D=Y:B=100],
    ADHY:0.1[&&NHX:S=nematode],ADHX:0.12[&&NHX:S=insect]):0.1[&&NHX:S=metazoa:D=N],
    (ADH4:0.09[&&NHX:S=yeast],ADH3:0.13[&&NHX:S=yeast], ADH2:0.12[&&NHX:S=yeast],
    ADH1:0.11[&&NHX:S=yeast]):0.1 [&&NHX:S=Fungi])[&&NHX:D=N];"
    )
```

---

read_pli *Read PLI files from SIFTER*

Description

Read PLI files from SIFTER

Usage

```
read_pli(fn, dropNAs = TRUE)
```

Arguments

fn	Full path to the file
dropNAs	Logical scalar. When TRUE, the function will discard any protein that has no annotations.

Value

A data table object including the following columns:

- name: Used to match UniProtKB data and GOA,
- number,
- go: A list of the GO annotations
- moc: Evidence code
- fam: Name of the family

See Also

Other reading: panther-tree, read_nhx()

`sim_fun_on_tree` *Simulate functions on a ginven tree*

Description

Simulate functions on a ginven tree

Usage

```
sim_fun_on_tree(
  tree,
  tip.type,
  node.type,
  psi,
  mu_d,
  mu_s,
  eta,
  Pi,
  P = 1L,
  informative = getOption("aphylo_informative", FALSE),
  maxtries = 20L
)
```

Arguments

`tree`	An object of class phylo
`tip.type, node.type`	Integer vectors with values 0,1. 0 denotes duplication node and 1 speciation node. This is used in LogLike.
`psi`	Numeric vector of length 2. Misclasification probabilities. (see `LogLike`).
`mu_d, mu_s`	Numeric vector of length 2. Gain/loss probabilities (see `LogLike`).

`eta`	Numeric vector of length 2. Annotation bias probabilities (see `LogLike`).
`Pi`	Numeric scalar. Root node probability of having the function (see `LogLike`).
`P`	Integer scalar. Number of functions to simulate.
`informative`	Logical scalar. When `TRUE` (default) the function re-runs the simulation algorithm until both 0s and 1s show in the leaf nodes of the tree.
`maxtries`	Integer scalar. If `informative = TRUE`, then the function will try at most `maxtries` times.

Details

Using the model described in the vignette peeling_phylo.html

The optiona `informative` was created such that when needed the function can be forced to simualte annotations while making sure (or at least trying `maxtries` times) that the leafs have both 0s and 9s. From what we've learned while conducting simulation studies, using this option may indirectly bias the data generating process.

Value

An matrix of size `length(offspring)*P` with values 9, 0 and 1 indicating `"no information"`, `"no function"` and `"function"`.

Examples

```
# Example 1 -------------------------------------------------------------------
# We need to simulate a tree
set.seed(1231)
newtree <- sim_tree(1e3)

# Preprocessing the data

# Simulating
ans <- sim_fun_on_tree(
  newtree,
  psi  = c(.01, .05),
  mu_d = c(.90, .80),
  mu_s = c(.1, .05),
  Pi   = .5,
  eta  = c(1, 1)
)

# Tabulating results
table(ans)
```

`sim_tree` *Random tree generation*

Description

An alternative to ape::rtree. This function was written in C++ and is significantly faster than `rtree`.

Usage

```
sim_tree(n, edge.length = stats::runif)
```

Arguments

`n`	Integer scalar. Number of leaf nodes.
`edge.length`	A Function. Used to set the length of the edges.

Details

The algorithm was implemented as follows

1. Initialize `N = {1, ..., n}`, `E` to be empty, `k = 2*n - 1`
2. While `length(N) != 1` do:
   (a) Randomly choose a pair `(i, j)` from `N`
   (b) Add the edges `E = E U {(k, i), (k, j)}`,
   (c) Redefine `N = (N \ {i, j}) U {k}`
   (d) Set `k = k - 1`
   (e) next
3. Use `edge.length(2*n - 1)` (simulating branch lengths).

Value

An object of class ape::phylo with the edgelist as a postorderd, `node.label` and `edge.length`.

Examples

```
# A very simple example ------------------------------------------------
set.seed(1223)
newtree <- sim_tree(50)

plot(newtree)


# A performance benchmark with ape::rtree ----------------------------------
## Not run:
library(ape)
microbenchmark::microbenchmark(
  ape = rtree(1e3),
```

```
  phy = sim_tree(1e3),
 unit = "relative"
)
# This is what you would get.
# Unit: relative
#   expr     min       lq     mean  median       uq      max neval
#    ape 14.7598 14.30809 14.30013 16.7217 14.32843 4.754106   100
#    phy  1.0000  1.00000  1.00000  1.0000  1.00000 1.000000   100

## End(Not run)
```

`states` *Matrix of states*

Description

Matrix of states

Usage

```
states(P)
```

Arguments

`P`	Integer scalar. Number of functions.

Value

A matrix of size 2^P by P with all the possible (0,1) combinations of functions.

Examples

```
states(3)
```

`write_pli` *Write pli files used by SIFTER*

Description

Write pli files used by SIFTER

Usage

```
write_pli(
  family_id,
  protein_name,
  protein_number,
  go_number,
  moc = "EXP",
  file = ""
)
```

Arguments

`family_id`	Character scalar. Name of the family
`protein_name, protein_number, go_number, moc`	Vectors of the same length
`file`	Character scalar passed to cat.

Value

A string with the XML file.

Examples

```
set.seed(882)
atree <- raphylo(5)
write_pli(
  family_id      = "a family",
  protein_name   = atree$tree$tip.label,
  protein_number = 1:Ntip(atree),
  go_number      = "GO:123123123123"
)
# Possible outcome:
#<?xml version="1.0"?>
#<Family>
#  <FamilyID>a family</FamilyID>
#  <Protein>
#    <ProteinName>1</ProteinName>
#    <ProteinNumber>1</ProteinNumber>
#    <GONumber>[GO:123123123123]</GONumber>
#    <MOC>[EXP]</MOC>
#  </Protein>
#  <Protein>
#    <ProteinName>2</ProteinName>
#    <ProteinNumber>2</ProteinNumber>
#    <GONumber>[GO:123123123123]</GONumber>
#    <MOC>[EXP]</MOC>
#  </Protein>
#  <Protein>
#    <ProteinName>3</ProteinName>
#    <ProteinNumber>3</ProteinNumber>
```

```
#    <GONumber>[GO:123123123123]</GONumber>
#    <MOC>[EXP]</MOC>
#  </Protein>
#  <Protein>
#    <ProteinName>4</ProteinName>
#    <ProteinNumber>4</ProteinNumber>
#    <GONumber>[GO:123123123123]</GONumber>
#    <MOC>[EXP]</MOC>
#  </Protein>
#  <Protein>
#    <ProteinName>5</ProteinName>
#    <ProteinNumber>5</ProteinNumber>
#    <GONumber>[GO:123123123123]</GONumber>
#    <MOC>[EXP]</MOC>
#  </Protein>
#</Family>
```

Index