Package 'BioGeoBEARS'

July 2, 2014

Type Package

Title BioGeography with Bayesian (and Likelihood) Evolutionary Analysis in R Scripts

Version 0.2.1

Date 2013-07-27

Maintainer Nicholas J. Matzke <matzke@berkeley.edu>

Depends rexpokit, cladoRcpp, ape, phylobase, methods

Imports optimx, FD, parallel, xtable, plotrix, gdata

Description BioGeoBEARS allows probabilistic inference of both historical biogeography (ancestral geographic ranges on a phylogeny) as well as comparison of different models of range evolution. It reproduces the model available in LAGRANGE (Ree and Smith 2008), as well as making available numerous additional models. For example, LAGRANGE as typically run has two free parameters, d (dispersal rate, i.e. the rate of range addition along a phylogenetic branch) and e (extinction rate, really the rate of local range loss along a phylogenetic branch). LAGRANGE also has a fixed cladogenic model which gives equal probability to a number of allowed range inheritance events, e.g.: (1) vicariance, (2) a new species starts in a subset of the ancestral range, (3) the ancestral range is copied to both species; in all cases, at least one species must have a starting range of size 1. LAGRANGE assigns equal probability to each of these events, and zero probability to other events. BioGeoBEARS adds an additional cladogenic event: founder-event speciation (the new species jumps to a range outside of the ancestral range), and also allows the relative weighting of the different sorts of events to be made into free parameters, allowing optimization and standard model choice procedures to pick the best model. The relative probability of different descendent range sizes is also parameterized and thus can also be specified or estimated. The flexibility available in BioGeoBEARS also enables the natural incorporation of (1) imperfect detection of geographic ranges in the tips, and (2) inclusion of fossil geographic range data, when the fossils are tips on the phylogeny. Bayesian analysis has been implemented through use of the ``LaplacesDemon'' package, however this package is now maintained off of CRAN, so its usage is not formally included in BioGeoBEARS at the current time. CITATION INFO: This package is the result of my Ph.D. research, please cite the package if you use it! Type: citation(package=``BioGeoBEARS'') to get the citation information.

URL http://phylo.wikidot.com/biogeobears

License GPL (>= 2)

LazyLoad yes

ByteCompile true

Author Nicholas J. Matzke [aut, cre, cph]

NeedsCompilation no

Repository CRAN

Date/Publication 2014-01-02 15:15:11

R topics documented:

`BioGeoBEARS-package` *BioGeography with Bayesian (and likelihood) Evolutionary Analysis of RangeS*

Description

BioGeoBEARS: BioGeography with Bayesian (and Likelihood) Evolutionary Analysis in R Scripts

Details

Package:	BioGeoBEARS
Type:	Package
Version:	0.2.1
Date:	2012-07-27
License:	GPL (>= 3)
LazyLoad:	yes

Summary: This package performs model-based statistical inference for historical biogeography. This includes inference of model parameters, ancestral states, and model comparison. This package performs ML (maximum-likelihood) based inference, but the same functions can easily be integrated into a Bayesian analysis via use of MCMC sampling functions from other packages.

Details: BioGeoBEARS allows probabilistic inference of both historical biogeography (ancestral geographic ranges on a phylogeny) as well as comparison of different models of range evolution. It reproduces the model available in LAGRANGE (Ree and Smith 2008), as well as making available numerous additional models. For example, LAGRANGE as typically run has two free parameters, d (dispersal rate, i.e. the rate of range addition along a phylogenetic branch) and e (extinction rate, really the rate of local range loss along a phylogenetic branch). LAGRANGE also has a fixed cladogenic model which gives equal probability to a number of allowed range inheritance events, e.g.: (1) vicariance, (2) a new species starts in a subset of the ancestral range, (3) the ancestral range is copied to both species; in all cases, at least one species must have a starting range of size 1. LAGRANGE assigns equal probability to each of these events, and zero probability to other events. BioGeoBEARS adds an additional cladogenic event: founder-event speciation (the new species jumps to a range outside of the ancestral range), and also allows the relative weighting of the different sorts of events to be made into free parameters, allowing optimization and standard model choice procedures to pick the best model. The relative probability of different descendent range sizes is also parameterized and thus can also be specified or estimated. The flexibility available in BioGeoBEARS also enables the natural incorporation of (1) imperfect detection of geographic ranges in the tips, and (2) inclusion of fossil geographic range data, when the fossils are tips on the phylogeny. Bayesian analysis has been implemented through use of the "LaplacesDemon" package, however this package is now maintained off of CRAN, so its usage is not formally included in BioGeoBEARS at the current time.

CITATION INFO: This package is the result of my Ph.D. research, please cite the package if you use it! Type: citation(package="BioGeoBEARS") to get the citation information.

See also the citation information for the sister packages, `citation(package="rexpokit")` and `citation(package="cladoRcpp")`.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

`rexpokit` `cladoRcpp`

Examples

```
test=1

# To get citation information for BioGeoBEARS, type:
```

```
citation(package="BioGeoBEARS")

# Please also cite the accessory packages I created to make BioGeoBEARS work:
citation(package="cladoRcpp")
citation(package="rexpokit") # Roger Sidje is a coauthor of rexpokit
                                # and author of the FORTRAN EXPOKIT
```

---

addslash *Add a slash to a directory name if needed*

Description

This function adds a slash to the end of the string, if one is not present. Handy for standardizing paths.

Usage

```
addslash(tmpstr)
```

Arguments

tmpstr	a path that you want to possibly add a slash to

Value

outstr a string of the fixed path

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

getwd, setwd, gsub

Examples

```
tmpstr = "/Dropbox/_njm/__packages"
tmpstr
outstr = addslash(tmpstr)
outstr

# Annoying, getwd() often doesn't return the ending slash, which
# can make life hard for paste() later on
tmpstr = getwd()
tmpstr
outstr = addslash(tmpstr)
outstr
```

add_corners *Iterate up through a plotted tree, getting the coordinates of the corners*

Description

What it says.

Usage

```
add_corners(startnode, tr, nodecoords, corners_list)
```

Arguments

startnode	The node to start at (this is a recursive function)
tr	A tree object in phylo format.
nodecoords	The accumulating list of node coordinates
corners_list	The accumulating list of corners

Value

corners_list

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo, get_nodenums

Examples

```
blah=1
```

add_to_downpass_labels *Iterate up and down a tree in C++ LAGRANGE downpass order*

Description

This is the utility function for get_lagrange_nodenums, which traces a tree down and up in C++ LAGRANGE's downpass order.

Usage

```
add_to_downpass_labels(tr, downpass_node_matrix,
  currnode)
```

Arguments

tr	A phylo tree object.
downpass_node_matrix	A matrix (tr$Nnode rows, 2 columns). Column 1 has R's native internal numbering scheme, and column 2 has the node numbers in a LAGRANGE downpass.
currnode	The current node being viewed

Details

This returns a matrix containing (column 1) R's native internal numbering scheme, and (column 2) the node numbers in a LAGRANGE downpass. Note that this is different from LAGRANGE's downpass ordering (see get_lagrange_nodenums).

Value

downpass_node_matrix A matrix containing node numbers.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_lagrange_nodenums

Examples

```
test=1
```

---

adf *Convert to data.frame, without factors*

Description

Shortcut for: `as.data.frame(x, row.names=NULL, stringsAsFactors=FALSE)`

Usage

```
adf(x)
```

Arguments

x	matrix or other object transformable to data.frame

Details

This function, and adf2, are useful for dealing with errors due to automatic conversion of some columns to factors. Another solution may be to prepend `options(stringsAsFactors = FALSE)` at the start of one's script, to turn off all default stringsAsFactors silliness.

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

adf2

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
adf(x)
```

adf2 *Convert to data.frame, without factors*

Description

Shortcut for: `tmp_rownames = 1:nrow(x); as.data.frame(x, row.names=tmp_rownames, stringsAsFactors=FAL`

Usage

```
adf2(x)
```

Arguments

x	matrix or other object transformable to data.frame

Details

This function, and adf2, are useful for dealing with errors due to automatic conversion of some columns to factors. Another solution may be to prepend `options(stringsAsFactors = FALSE)` at the start of one's script, to turn off all default stringsAsFactors silliness.

In adf2, rownames are forced to be numbers; this can prevent errors due to e.g. repeated rownames after an `rbind` operation.

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

adf

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
adf2(x)
```

AICstats_2models *Calculate all the AIC and LRT stats between two models*

Description

The Likelihood Ratio Test (LRT) is a standard method for testing whether or not the data likelihood conferred by a more complex is significantly better than the data likelihood conferred by the simpler model. See lrttest and lrttest_on_summary_table for more discussion.

Usage

```
AICstats_2models(LnL_1, LnL_2, numparams1, numparams2)
```

Arguments

LnL_1	Log-likelihood of more complex model.
LnL_2	Log-likelihood of simpler complex model.
numparams1	Number of free parameters of the more complex model.
numparams2	Number of free parameters of the less complex model.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

This function assumes that LnL_1 and numparams1 refer to the more complex model, and that LnL_2 and numparams2 refer to the simpler model nested within the more complex one.

Value

LRT_AIC_results A table of LRT and AIC results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

lrttest, lrttest_on_summary_table

Examples

```
test=1
```

AkaikeWeights_and_Ratios_pairwise_on_summary_table_compared_to_ref
Get the ratio between the pairwise Akaike Weights

Description

Given the relative likelihoods of the models, calculate the Akaike weight of the models. Akaike weights sum to 1.

Usage

```
AkaikeWeights_and_Ratios_pairwise_on_summary_table_compared_to_ref(restable,
  colname_to_use = "AIC", ref_model = "best",
  add_to_table = TRUE)
```

Arguments

restable	A data.frame with at least columns named "LnL" and "nparams".
colname_to_use	The name of the column containing AIC values.
ref_model	What is the row of the reference model? "best", "worst", or a row number.
add_to_table	If TRUE, add to the main table and return the main table. If FALSE, return just the Akaike Weights results.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

restable, the modified table, or AICstats_pairwise, the pairwise Akaike statistics.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_Akaike_weights_from_rel_likes_pairwise, get_Akaike_weights_from_rel_likes, rel_likes_from_deltaAI
getAIC

Examples

```
test=1

tmptable = adf(c(40, 50, 60))
names(tmptable) = "AIC"
AkaikeWeights_and_Ratios_pairwise_on_summary_table_compared_to_ref(
restable=tmptable, colname_to_use="AIC", ref_model="best", add_to_table=TRUE)
```

AkaikeWeights_on_summary_table

Calculate Akaike Weights, and add to table

Description

This calculates Akaike Weights (relative probabilities on models explaining the same data) for the models in a column in a table.

Usage

```
AkaikeWeights_on_summary_table(restable,
  colname_to_use = "AIC", add_to_table = TRUE)
```

Arguments

restable	A data.frame with at least a column named as in add_to_table.
colname_to_use	The name of the column containing AIC values.
add_to_table	If TRUE, add to the main table and return the main table. If FALSE, return just the Akaike Weights results.

Value

restable, the modified table, or wt_vBest, the Akaike Weights results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Matzke_2012_IBS

Burnham_Anderson_2002

See Also

calc_AIC_column, calc_AICc_column

Examples

```
test=1
```

---

areas_list_to_states_list_new

Convert a list of areas to a list of geographic ranges (states); R version

---

Description

R version of areas_list_to_states_list_old, which makes use of cladoRcpp's rcpp_areas_list_to_states_list.

Usage

```
areas_list_to_states_list_new(areas = c("A", "B", "C"),
  maxareas = length(areas), include_null_range = TRUE,
  split_ABC = TRUE)
```

Arguments

areas	a list of areas (character or number; the function converts these to numbers, starting with 0)
maxareas	maximum number of areas in this analyses
include_null_range	TRUE or FALSE, should the NULL range be included in the possible states? (e.g., LAGRANGE default is yes)
split_ABC	TRUE or FALSE If TRUE the output will consist of a list of lists (c("A","B","C"), c("A","B"), c("A","D"), etc.); if FALSE, the list of areas will be collapsed ("ABC", "AB", "AD", etc.).

Details

This is the original R version of the function which converts a list of possible areas to a list of all possible states (geographic ranges). This gets slow for large numbers of areas.

The function is mostly replaced by rcpp_areas_list_to_states_list in optimized code, but is still used in some places for display purposes.

Value

states_list A list of the states.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

numstates_from_numareas, rcpp_areas_list_to_states_list

Examples

```
areas = c("A","B","C")
areas_list_to_states_list_new(areas=areas, maxareas=length(areas),
include_null_range=TRUE, split_ABC=TRUE)
areas_list_to_states_list_new(areas=areas, maxareas=length(areas),
include_null_range=TRUE, split_ABC=FALSE)
areas_list_to_states_list_new(areas=areas, maxareas=length(areas),
include_null_range=FALSE, split_ABC=TRUE)
areas_list_to_states_list_new(areas=areas, maxareas=length(areas),
include_null_range=FALSE, split_ABC=FALSE)
areas_list_to_states_list_new(areas=areas, maxareas=2,
include_null_range=TRUE, split_ABC=TRUE)
areas_list_to_states_list_new(areas=areas, maxareas=2,
include_null_range=TRUE, split_ABC=FALSE)
areas_list_to_states_list_new(areas=areas, maxareas=2,
include_null_range=FALSE, split_ABC=TRUE)
areas_list_to_states_list_new(areas=areas, maxareas=2,
include_null_range=FALSE, split_ABC=FALSE)
areas_list_to_states_list_new(areas=areas, maxareas=1,
include_null_range=TRUE, split_ABC=TRUE)
```

```
areas_list_to_states_list_new(areas=areas, maxareas=1,
include_null_range=TRUE, split_ABC=FALSE)
areas_list_to_states_list_new(areas=areas, maxareas=1,
include_null_range=FALSE, split_ABC=TRUE)
areas_list_to_states_list_new(areas=areas, maxareas=1,
include_null_range=FALSE, split_ABC=FALSE)
```

`average_tr_tips` *Average the heights of (non-fossil) tips to make ultrametric-ish.*

Description

When you have a digitized tree, or other slightly uneven source tree, average the tips to get them all to line up at 0 my before present. This makes an ultrametric tree if and only if there are no fossil tips in the tree.

Usage

```
average_tr_tips(tr, fossils_older_than = 0.6)
```

Arguments

`tr`	An ape phylo object
`fossils_older_than`	Tips that are older than `fossils_older_than` will be excluded from the tips that are going to be averaged. This is not currently set to 0, because Newick files can have slight precision issues etc. that mean not all tips quite come to zero (which is why you need `average_tr_tips` in the first place!). Obviously you should be cautious about the value of , depending on the absolute timescale of your tree. Make sure you do not inappropriately average in fossils!!

Details

If the user includes fossils accidentally, this function can easily lead to pathological results (negative branch lengths etc.), so use with care!!

Value

`edge_times_bp` A 2-column matrix with the age (from the present) of the top and bottom of each edge.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2, extend_tips_to_ultrametricize

Examples

```
test=1
```

`axisPhylo2` *axisPhylo with more flexibility in labeling*

Description

Hacking axisPhylo to make it more flexible

Usage

```
axisPhylo2(side = 1, roundlabels = FALSE, minage = 0,
  ...)
```

Arguments

`side`	The side to plot on (default 1, bottom)
`roundlabels`	Number of digits to round to, if desired
`minage`	Starting age, if desired
`...`	Additional arguments to standard functions

Value

nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

FosterIdiots

Examples

```
testval=1
```

`bears_2param_DIVA_fast` *2-parameter model, fixed cladogenesis model (as in LAGRANGE)*

Description

This function implements a biogeographical model with 2 free parameters (*d*, rate of dispersal/range addition, and *e*, rate of extinction/range contraction), and a fixed cladogenesis model copying the DIVA model (*Ronquist (1997)*. This has: equal probability of vicariance at all range sizes, but NO sympatric-subset speciation, no jump/founder-event speciation, and sympatric-range-copying events are limited to the smaller descendant always having a range size of 1 area(*Ronquist et al. (2011)*).

Usage

```
bears_2param_DIVA_fast(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

`trfn`	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See `read.tree` in APE for reading in phylogenetic trees.
`geogfn`	A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
`max_range_size`	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.
`num_cores_to_use`	If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.

Details

Once the model is set up, it is input into the optimization routine `optimx` (the more common `optim` can also be used by editing the function), and `calc_loglike_sp` is used to calculate the log-likelihood of each set of parameters. Once the parameter values that give the data the maximum likelihood are found, they are reported back to the function and returned to the user.

This duplicates the model used in the standard DIVA implementation (*Ree et al. (2008)*, *Ree (2009)*, *Smith et al. (2010)*, with no constraints on dispersal or range size.

Here, all of the fastest processing options have been used.

Model implementations are provided to show the user how a specific model can be set up and optimized. This is preferable compared to the "black-box" nature of most other inference packages. Users are encouraged to experiment. Useful models can be added to later versions of BioGeoBEARS.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Ronquist_1997_DIVA

Ronquist_Sanmartin_2011

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Matzke_2012_IBS

Ronquist1996_DIVA

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))
```

```
# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

`bears_2param_standard_fast` *2-parameter model, fixed cladogenesis model (as in LAGRANGE)*

Description

This function implements a biogeographical model with 2 free parameters (*d*, rate of dispersal/range addition, and *e*, rate of extinction/range contraction), and a fixed cladogenesis model with equal probability of vicariance, sympatric-subset, and sympatric-range-copying events, and with the smaller descendant always having a range size of 1 area. Once the model is set up, it is input into the optimization routine `optimx` (the more common `optim` can also be used by editing the function), and `calc_loglike_sp` is used to calculate the log-likelihood of each set of parameters. Once the parameter values that give the data the maximum likelihood are found, they are reported back to the function and returned to the user.

Usage

```
bears_2param_standard_fast(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

`trfn`	The filename of the phylogenetic tree, in NEWICK format (`http://evolution.genetics.washington.edu/phylip/newicktree.html`). Tipnames should match the names in geogfn. See `read.tree` in APE for reading in phylogenetic trees.
`geogfn`	A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
`max_range_size`	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.
`num_cores_to_use`	If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.

Details

This duplicates the model used in the standard LAGRANGE implementation (*Ree et al. (2008)*, *Ree (2009)*, *Smith et al. (2010)*, with no constraints on dispersal or range size.

Here, all of the fastest processing options have been used.

Model implementations are provided to show the user how a specific model can be set up and optimized. This is preferable compared to the "black-box" nature of most other inference packages. Users are encouraged to experiment. Useful models can be added to later versions of `BioGeoBEARS`.

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Matzke_2012_IBS

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))
```

```
# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_2param_standard_fast_fixnode *2-parameter model, fixed cladogenesis model (as in LAGRANGE)*

Description

This function implements a biogeographical model with 2 free parameters (*d*, rate of dispersal/range addition, and *e*, rate of extinction/range contraction), and a fixed cladogenesis model with equal probability of vicariance, sympatric-subset, and sympatric-range-copying events, and with the smaller descendant always having a range size of 1 area. Once the model is set up, it is input into the optimization routine optimx (the more common optim can also be used by editing the function), and calc_loglike_sp is used to calculate the log-likelihood of each set of parameters. Once the parameter values that give the data the maximum likelihood are found, they are reported back to the function and returned to the user.

Usage

```
bears_2param_standard_fast_fixnode(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL, fixnode = NULL,
  fixlikes = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

fixnode	If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number.
fixlikes	The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others.

Details

This duplicates the model used in the standard LAGRANGE implementation (*Ree et al. (2008)*, *Ree (2009)*, *Smith et al. (2010)*, with no constraints on dispersal or range size.

Here, all of the fastest processing options have been used.

Model implementations are provided to show the user how a specific model can be set up and optimized. This is preferable compared to the "black-box" nature of most other inference packages. Users are encouraged to experiment. Useful models can be added to later versions of BioGeoBEARS.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Matzke_2012_IBS

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
```

```
# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

`bears_2param_standard_fast_fortest` *2-parameter model, fixed cladogenesis model (as in LAGRANGE) – older test version*

Description

This is an older, test version of `bears_2param_standard_fast`.

Usage

```
bears_2param_standard_fast_fortest(trfn = "test.newick",
  geogfn = "test.data")
```

Arguments

`trfn`	The filename of the phylogenetic tree, in NEWICK format (`http://evolution.genetics.washington.edu/phylip/newicktree.html`). Tipnames should match the names in geogfn. See `read.tree` in APE for reading in phylogenetic trees.
`geogfn`	A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast_fortest(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

bears_2param_standard_fast_symOnly *2-parameter model, no cladogenesis model (as in BayArea or other purely continuous-time model)*

Description

This implements a 2-parameter model, as in LAGRANGE or bears_2param_standard_fast, but omits the speciation/cladogenesis model. This means that the model is purely continuous-time, as when biogeographic range is treated as a discrete character in software designed for inference on morphological () or molecular data (). This model is that implemented in BayArea, if no distance-dependent effect on dispersal probability is assumed. Such distance-dependence could easily be added with a third parameter, however.

Usage

```
bears_2param_standard_fast_symOnly(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Details

BayArea is a new program by Landis, Matzke, Moore, and Huelsenbeck; see *Landis et al. (2013).* However, BayArea does not currently implement cladogenesis models; it only has continuous-time model for evolutionary change along branches. In effect, this means that the cladogenesis model is sympatric speciation with complete range copying with probability 1.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Landis_Matzke_etal_2013_BayArea

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast_symOnly(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

bears_2param_standard_fast_symOnly_simp
2-parameter model, no cladogenesis model (as in BayArea or other purely continuous-time model)

Description

(Forcing no speciation model.) This implements a 2-parameter model, as in LAGRANGE or bears_2param_standard_fast, but omits the speciation/cladogenesis model. This means that the model is purely continuous-time, as when biogeographic range is treated as a discrete character in software designed for inference on morphological () or molecular data (). This model is that implemented in BayArea, if no distance-dependent effect on dispersal probability is assumed. Such distance-dependence could easily be added with a third parameter, however.

Usage

```
bears_2param_standard_fast_symOnly_simp(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Details

BayArea is a new program by Landis, Matzke, Moore, and Huelsenbeck; see *Landis et al. (2013)*. However, BayArea does not currently implement cladogenesis models; it only has continuous-time model for evolutionary change along branches. In effect, this means that the cladogenesis model is sympatric speciation with complete range copying with probability 1.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Landis_Matzke_etal_2013_BayArea

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_fast_symOnly(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

bears_2param_standard_slowQ_slowSP
2-parameter model, fixed cladogenesis model – slow version

Description

This implements the same 2-parameter model found in LAGRANGE or bears_2param_standard_fast, but using the original slower options for matrix exponentiation and cladogenesis events.

Usage

```
bears_2param_standard_slowQ_slowSP(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_2param_standard_slowQ_slowSP(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_3param_standard_fast

3-parameter model, adding j (founder-event speciation)

Description

This implements a 3-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*) versus vicariance+sympatric speciation (which are mandated in LAGRANGE and bears_2param_standard_fast.

Usage

```
bears_3param_standard_fast(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.

`geogfn` A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).

`max_range_size` The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.

`num_cores_to_use` If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

`numstates_from_numareas`, `getranges_from_LagrangePHYLIP`, `read.tree`, `calc_loglike_sp`

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)
```

```
geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_3param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

`bears_3param_standard_fast_fixnode` *3-parameter model, adding j (founder-event speciation)*

Description

This implements a 3-parameter model, basically LAGRANGE or `bears_2param_standard_fast`, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*) versus vicariance+sympatric speciation (which are mandated in LAGRANGE and `bears_2param_standard_fast`.

Usage

```
bears_3param_standard_fast_fixnode(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL, fixnode = fixnode,
  fixlikes = fixlikes)
```

Arguments

`trfn`	The filename of the phylogenetic tree, in NEWICK format (`http://evolution.genetics.washington.edu/phylip/newicktree.html`). Tipnames should match the names in geogfn. See `read.tree` in APE for reading in phylogenetic trees.
`geogfn`	A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
`max_range_size`	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.
`num_cores_to_use`	If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.
`fixnode`	If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number.
`fixlikes`	The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others.

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_3param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

bears_3param_standard_fast_noJ *3-parameter model, adding v (vicariance proportion), but no j (founder-event speciation)*

Description

This implements a 3-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *v* controlling the relative weight of vicariance versus the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast.

Usage

```
bears_3param_standard_fast_noJ(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_3param_standard_fast_noJ(trfn=trfn, geogfn=geogfn)
bears_output#'

## End(Not run)
```

`bears_4param_standard_fast` *4-parameter model, adding j (founder-event speciation) and v (vicariance proportion)*

Description

This implements a 4-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*) and another parameter *v* controlling the relative weight of vicariance. The remainder of the weight (weights must sum to 1) is taken up by the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast.

Usage

```
bears_4param_standard_fast(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_4param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_5param_standard_fast

5-parameter model, adding j (founder-event speciation), v (vicariance proportion), and maxent_constraint_01 (weighting for size of smaller-ranged descendant lineage)

Description

This implements a 5-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*), and another parameter *v* controlling the relative weight of vicariance. The remainder of the weight (weights must sum to 1) is taken up by the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast. A fifth parameter, *maxent_constraint_01*, controls the relative probability of daughter lineages of different rangesizes. If maxent_constraint_01=0.0001, the smaller-ranged daughter lineage will have size 1 area, with probability 1. If maxent_constraint_01=0.5, all different rangesizes will have equal probability, and if maxent_constraint_01=0.9999, the largest possible range will have probability 1.

Usage

```
bears_5param_standard_fast(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

bears_2param_standard_fast, numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_5param_standard_fast(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_5param_standard_fast_diffstart

5-parameter model, with different starting points for optimization

---

Description

This implements the same model as bears_5param_standard_fast, but uses different starting points and slightly different constraints.

Usage

```
bears_5param_standard_fast_diffstart(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).

`max_range_size` The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.

`num_cores_to_use`
If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.

Details

As the number of parameters increases, the importance of starting ML optimization runs from different places increases. Several starting points should be tried, especially if the likelihood surface seems flat.

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

`bears_2param_standard_fast`, `numstates_from_numareas`, `getranges_from_LagrangePHYLIP`, `read.tree`, `calc_loglike_sp`

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
```

```
# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_5param_standard_fast_diffstart(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_5param_standard_fast_v *5-parameter model, adding j (founder-event speciation), v (vicariance proportion), and maxent_constraint_01v (vicariance daughter sizes)*

Description

This implements a 5-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*), and another parameter *v* controlling the relative weight of vicariance. The remainder of the weight (weights must sum to 1) is taken up by the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast. A fifth parameter, *maxent_constraint_01v*, controls the relative probability of daughter lineages of different rangesizes, but only for the vicariance events, which are rather different from other types of speciation events. If maxent_constraint_01v=0.0001, the smaller-ranged daughter lineage will have size 1 area, with probability 1. If maxent_constraint_01v=0.5, all different rangesizes will have equal probability, and if maxent_constraint_01v=0.9999, the largest possible range will have probability 1 – but note that in a vicariance context, this would mean at maximum rangesize of 50 the areas.

Usage

```
bears_5param_standard_fast_v(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.

`geogfn` A PHYLIP-style file with geographic range data (see `getranges_from_LagrangePHYLIP`) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).

`max_range_size` The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see `numstates_from_numareas`.

`num_cores_to_use` If >1, parallel processing will be attempted. **Note:** parallel processing via `library (parallel)` will work in Mac command-line R, but not in Mac GUI `R.app`.

Details

Non-vicariance events have hard-coded `maxent_constraint_01=0.0001`

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

`bears_2param_standard_fast`, `numstates_from_numareas`, `getranges_from_LagrangePHYLIP`, `read.tree`, `calc_loglike_sp`

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_5param_standard_fast_v(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_6param_standard_fast_ys_v

6-parameter model, adding j (founder-event speciation), v (vicariance proportion), and both maxent_constraint_01 and maxent_constraint_01v

---

Description

This implements a 6-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*), and another parameter *v* controlling the relative weight of vicariance. The remainder of the weight (weights must sum to 1) is taken up by the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast. A fifth parameter, *maxent_constraint_01*, controls the relative probability of daughter lineages of different rangesizes. A sixth parameter, *maxent_constraint_01v*, controls the relative probability of daughter lineages of different rangesizes, but only for the vicariance events, which are rather different from other types of speciation events. If maxent_constraint_01v=0.0001, the smaller-ranged daughter lineage will have size 1 area, with probability 1. If maxent_constraint_01v=0.5, all different rangesizes will have equal probability, and if maxent_constraint_01v=0.9999, the largest possible range will have probability 1 – but note that in a vicariance context, this would mean at maximum rangesize of 50 the areas.

Usage

```
bears_6param_standard_fast_ys_v(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

bears_2param_standard_fast, numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_6param_standard_fast_ys_v(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

---

bears_9param_standard_fast_ys_v_cb

6-parameter model, adding j (founder-event speciation), v (vicariance proportion), and both maxent_constraint_01 and maxent_constraint_01v

---

Description

This implements a 6-parameter model, basically LAGRANGE or bears_2param_standard_fast, but with a parameter *j* controlling the relative weight of "founder-event speciation" (*Matzke (2012)*), and another parameter *v* controlling the relative weight of vicariance. The remainder of the weight (weights must sum to 1) is taken up by the range-copying and range-subset forms of sympatric speciation utilized in LAGRANGE and bears_2param_standard_fast. A fifth parameter, *maxent_constraint_01*, controls the relative probability of daughter lineages of different rangesizes. A sixth parameter, *maxent_constraint_01v*, controls the relative probability of daughter lineages of different rangesizes, but only for the vicariance events, which are rather different from other types of speciation events. If maxent_constraint_01v=0.0001, the smaller-ranged daughter lineage will have size 1 area, with probability 1. If maxent_constraint_01v=0.5, all different rangesizes will have equal probability, and if maxent_constraint_01v=0.9999, the largest possible range will have probability 1 – but note that in a vicariance context, this would mean at maximum rangesize of 50 the areas.

Usage

```
bears_9param_standard_fast_ys_v_cb(trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", max_range_size = NULL,
  num_cores_to_use = NULL)
```

Arguments

trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees.
geogfn	A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*).
max_range_size	The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.
num_cores_to_use	If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

Value

bears_output A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

bears_2param_standard_fast, numstates_from_numareas, getranges_from_LagrangePHYLIP, read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_6param_standard_fast_ys_v(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
# Add c (direct changes), b (branch length exponent), x (distance exponent)
```

`bears_optim_run` *Run ML search from* `BioGeoBEARS_run` *object*

Description

Uses a BioGeoBEARS_run_object to simplify input.

Usage

```
bears_optim_run(BioGeoBEARS_run_object = define_BioGeoBEARS_run())
```

Arguments

`BioGeoBEARS_run_object`
: Contains all inputs

Value

`bears_output` A list of outputs. bears_output$optim_result

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Felsenstein, Joe. The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

Ree2009configurator

SmithRee2010_CPPversion

Landis_Matzke_etal_2013_BayArea

See Also

readfiles_BioGeoBEARS_run, bears_2param_standard_fast, numstates_from_numareas, getranges_from_Lagrange
read.tree, calc_loglike_sp

Examples

```
test=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"

# Set the filenames (Hawaiian Psychotria from Ree & Smith 2008)
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(file=trfn)

geogfn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Look at the tree and ranges, for kicks
getranges_from_LagrangePHYLIP(lgdata_fn=geogfn)
tr

## Not run:
# Run the ML search
bears_output = bears_optim_run(trfn=trfn, geogfn=geogfn)
bears_output

## End(Not run)
```

`binary_ranges_to_letter_codes` *Convert binary presence/absence codes (1/0) to a list of text area names*

Description

Given a row of a `tipranges` object, converts to a list of the corresponding statenames for each row.

Usage

```
binary_ranges_to_letter_codes(tipranges, areanames)
```

Arguments

`tipranges`	a `tipranges` object.
`areanames`	a list of the names of the areas

Value

`letter_code_ranges` A list of the states – there will be as many states as there are rows/tips in `tipranges`. Each state will be a list of area names.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

`binary_range_to_letter_code_list`, `letter_string_to_binary`, `letter_strings_to_tipranges_df`, `tipranges_to_tip_condlikes_of_data_on_each_state`

Examples

```
testval=1
# Define a tipranges object
tipranges_object = define_tipranges_object()
tipranges_object

areanames = getareas_from_tipranges_object(tipranges_object)
areanames

letter_code_ranges = binary_ranges_to_letter_codes(tipranges=tipranges_object,
areanames)
letter_code_ranges
```

---

binary_range_to_letter_code_list

Convert binary presence/absence codes (1/0) to a list of text area names

---

Description

Given a row of a `tipranges` object, converts to a list of the corresponding name(s). E.g., if the areas were `(A,B,C,D)`, and the tipranges row had `(1 0 1 0)`, the output statename would be ("A","C").

Usage

```
binary_range_to_letter_code_list(tipranges_row,
  areanames)
```

Arguments

`tipranges_row`	row of a `tipranges` object.
`areanames`	a list of the names of the areas

Value

`list_of_areas_in_the_state` A list of the name(s) of the areas corresponding to the presence/absence coding in the row

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

binary_ranges_to_letter_codes, letter_string_to_binary, letter_strings_to_tipranges_df, binary_range_to_letter_code_txt, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
testval=1
tipranges_row = c(1, 0, 1, 0)
areanames = c("A", "B", "C", "D")
list_of_areas_in_the_state = binary_range_to_letter_code_list(tipranges_row,
areanames)
list_of_areas_in_the_state
```

---

binary_range_to_letter_code_txt

Convert binary presence/absence codes (1/0) to text area names

---

Description

Given a row of a tipranges object, converts to the corresponding name(s), collapsed into a string. E.g., if the areas were (A,B,C,D), and the tipranges row had (1 0 1 0), the output statename would be "AC".

Usage

```
binary_range_to_letter_code_txt(tipranges_row, areanames)
```

Arguments

tipranges_row	row of a tipranges object.
areanames	a list of the names of the areas

Value

statename The corresponding name(s), collapsed into a string

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

binary_range_to_letter_code_list, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
testval=1
tipranges_row = c(1, 0, 1, 0)
areanames = c("A", "B", "C", "D")
statename = binary_range_to_letter_code_txt(tipranges_row, areanames)
statename
```

BioGeoBEARS_model *An object of class BioGeoBEARS_model holding the model inputs*

Description

An object of class BioGeoBEARS_model holding the model inputs

Slots

df: Data.frame of class "numeric", containing data from df

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_tipranges_object, getareas_from_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
tipranges_object = define_tipranges_object()
tipranges_object
```

`BioGeoBEARS_model_defaults` *Set up a default BioGeoBEARS model object*

Description

What it says.

Usage

```
BioGeoBEARS_model_defaults(minval_anagenesis = 1e-15,
  minval_cladogenesis = 1e-05, maxval = 5)
```

Arguments

`minval_anagenesis`	Minimum value above zero for d, e, a, b parameters.
`minval_cladogenesis`	Minimum value above zero for j, v, etc.
`maxval`	Maximum value for d, e, a

Value

`param_table` Return the parameter table object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`rbind`

Examples

```
test=1
```

`BioGeoBEARS_model_object_to_est_params` *Extract estimated parameters from a BioGeoBEARS model object*

Description

What it says.

Usage

```
BioGeoBEARS_model_object_to_est_params(BioGeoBEARS_model_object)
```

Arguments

`BioGeoBEARS_model_object` The BioGeoBEARS_model object, of class `BioGeoBEARS_model`

Value

`params` parameter vector

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`define_BioGeoBEARS_model_object`

Examples

```
test=1
```

BioGeoBEARS_model_object_to_init_params
Produce initial parameters from a BioGeoBEARS model object

Description

What it says.

Usage

```
BioGeoBEARS_model_object_to_init_params(BioGeoBEARS_model_object)
```

Arguments

BioGeoBEARS_model_object
The BioGeoBEARS_model object, of class BioGeoBEARS_model

Value

params parameter vector

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_model_object

Examples

```
test=1
```

BioGeoBEARS_model_object_to_params_lower *Produce the lower limit on the parameters from a BioGeoBEARS model object*

Description

What it says.

Usage

```
BioGeoBEARS_model_object_to_params_lower(BioGeoBEARS_model_object)
```

Arguments

BioGeoBEARS_model_object
: The BioGeoBEARS_model object, of class BioGeoBEARS_model

Value

params parameter vector

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_model_object

Examples

```
test=1
```

`BioGeoBEARS_model_object_to_params_upper` *Produce the upper limit on the parameters from a BioGeoBEARS model object*

Description

What it says.

Usage

```
BioGeoBEARS_model_object_to_params_upper(BioGeoBEARS_model_object)
```

Arguments

`BioGeoBEARS_model_object` The BioGeoBEARS_model object, of class `BioGeoBEARS_model`

Value

`params` parameter vector

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`define_BioGeoBEARS_model_object`

Examples

```
test=1
```

BioGeoBEARS_run *An object of class BioGeoBEARS_run holding the model inputs*

Description

An object of class BioGeoBEARS_run holding the model inputs

Slots

`list`: List of class `"list"`, containing inputs list from define_BioGeoBEARS_run

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_tipranges_object, getareas_from_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
test=1
```

calcP_n *Calculate Z (part of equation 6.4 of Harte 2011)*

Description

This function is a used by get_probvals, which calculates the Maximum Entropy (*Harte (2011)*) discrete probability distribution of a number of ordered states (e.g., faces of a 6-sided die) given the mean of many rolls. Here, this is merely used so that a single parameter can control the probability distribution of small versus large descendant areas during cladogenesis.

Usage

```
calcP_n(n, lambda1, Z)
```

Arguments

n	Value of the state (e.g., which of a number of faces on a die, or number of different size classes of geographic range).
lambda1	Lambda parameter (*Harte2011*).
Z	numeric values from calcZ_part.

Details

See also: Maximum Entropy probability distribution for discrete variable with given mean (and discrete uniform flat prior) http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Value

Prob_n, numeric value of the probability of state n.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Harte2011

Matzke_2012_IBS

See Also

calcZ_part, maxent, symbolic_to_relprob_matrix_sp

Examples

```
testval=1
n = 6
lambda1 = 0.5
Z = 1
calcP_n(n, lambda1, Z)
```

calcZ_part *Calculate Z (equation 6.3 of Harte 2011)*

Description

This function is a used by calcP_n via apply, all within get_probvals. get_probvals calculates the Maximum Entropy (*Harte (2011)*) discrete probability distribution of a number of ordered states (e.g., faces of a 6-sided die) given the mean of many rolls. Here, this is merely used so that a single parameter can control the probability distribution of small versus large descendant areas during cladogenesis.

Usage

```
calcZ_part(n, lambda1)
```

Arguments

n	Value of the state (e.g., which of a number of faces on a die, or number of different size classes of geographic range)
lambda1	Lambda parameter (*Harte2011*).

Details

See also: Maximum Entropy probability distribution for discrete variable with given mean (and discrete uniform flat prior) http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Value

Z, numeric value

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Harte2011

Matzke_2012_IBS

See Also

calcP_n, maxent, symbolic_to_relprob_matrix_sp

Examples

```
testval=1
n=6
lambda1 = 0.5
calcZ_part(n, lambda1)
```

---

calc_AICc_column *Calculate AICc values for a list of models*

Description

A list of AICc values (second order Akaike Information Criterion) is calculated from two input lists. Lower values of AICc indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters). AICc contains a correction for sample size.

Usage

```
calc_AICc_column(LnL_vals, nparam_vals, samplesize)
```

Arguments

LnL_vals	A vector of log-likelihoods (typically negative, but may not be for continuous data).
nparam_vals	A vector of the number of parameters for each model.
samplesize	A single samplesize, or a vector of the samplesizes each model. However, samplesize should always be the same for all comparisons, since maximum likelihood and AIC/AICc model-selection methods are always comparing different models on the *same* data, not different data on the same mode.

Details

The two input lists are:

1. A list of data likelihoods under a variety of models.
2. A list of the number of free parameters under each model.

samplesize can be a scalar or vector; but see below.

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC, AICc and their uses.

Value

AICc_col A data.frame column of AICc results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AICc_vals, calc_AIC_column

Examples

```
LnL_vals = c(-34.5, -20.9)
nparam_vals = c(2, 3)
calc_AICc_column(LnL_vals, nparam_vals, samplesize=20)

LnL_vals = c(-20.9, -20.9, -20.9, -20.9)
nparam_vals = c(3, 4, 5, 6)
calc_AICc_column(LnL_vals, nparam_vals, samplesize=20)
```

---

calc_AICc_vals *Calculate AICc values for a list of models*

Description

A list of AICc values (second order Akaike Information Criterion) is calculated from two input lists. Lower values of AICc indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters). AICc contains a correction for sample size.

Usage

```
calc_AICc_vals(LnL_vals, nparam_vals, samplesize)
```

Arguments

LnL_vals	A vector of log-likelihoods (typically negative, but may not be for continuous data).
nparam_vals	A vector of the number of parameters for each model.
samplesize	A single samplesize, or a vector of the samplesizes each model. However, samplesize should always be the same for all comparisons, since maximum likelihood and AIC/AICc model-selection methods are always comparing different models on the *same* data, not different data on the same mode.

Details

The two input lists are:

1. A list of data likelihoods under a variety of models.
2. A list of the number of free parameters under each model.

samplesize can be a scalar or vector; but see below.

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC, AICc and their uses.

Value

AICc_vals A vector of AICc results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AIC_vals, calc_AICc_column

Examples

```
LnL_vals = c(-34.5, -20.9)
nparam_vals = c(2, 3)
calc_AICc_vals(LnL_vals, nparam_vals, samplesize=20)

LnL_vals = c(-20.9, -20.9, -20.9, -20.9)
nparam_vals = c(3, 4, 5, 6)
calc_AICc_vals(LnL_vals, nparam_vals, samplesize=20)
```

`calc_AIC_column` *Calculate AICs to make a column in a table*

Description

A list of AICs (Akaike Information Criterion) is calculated from two input lists. Lower values of AIC indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters).

Usage

```
calc_AIC_column(LnL_vals, nparam_vals)
```

Arguments

`LnL_vals`	A vector of log-likelihoods (typically negative, but may not be for continuous data).
`nparam_vals`	A vector of the number of parameters for each model.

Details

The two input lists are:

1. A list of data likelihoods under a variety of models.
2. A list of the number of free parameters under each model.

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`AIC_col` A `data.frame` column of AIC results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AIC_vals, calc_AICc_vals

Examples

```
test=1

LnL_vals = c(-20.9, -20.9, -20.9, -20.9)
nparam_vals = c(3, 4, 5, 6)
calc_AIC_column(LnL_vals, nparam_vals)
```

---

calc_AIC_vals *Calculate AICs for a list of models*

Description

A list of AICs (Akaike Information Criterion) is calculated from two input lists. Lower values of AIC indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters).

Usage

```
calc_AIC_vals(LnL_vals, nparam_vals)
```

Arguments

LnL_vals	A vector of log-likelihoods (typically negative, but may not be for continuous data).
nparam_vals	A vector of the number of parameters for each model.

Details

The two input lists are:

1. A list of data likelihoods under a variety of models.
2. A list of the number of free parameters under each model.

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

AIC_vals A vector of AIC results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AIC_column, calc_AICc_column

Examples

```
LnL_vals = c(-34.5, -20.9)
nparam_vals = c(2, 3)
calc_AIC_vals(LnL_vals, nparam_vals)

LnL_vals = c(-20.9, -20.9, -20.9, -20.9)
nparam_vals = c(3, 4, 5, 6)
calc_AIC_vals(LnL_vals, nparam_vals)
```

---

calc_linked_params_BioGeoBEARS_model_object

Update parameters that are deterministic functions of free parameters

---

Description

This function updates the linked parameters (which are listed as neither "fixed" nor "free" in params_table$type; i.e., they are equations which are calculated from #' the fixed and free parameters, which should have already been set by other functions).

Usage

```
calc_linked_params_BioGeoBEARS_model_object(BioGeoBEARS_model_object,
  update_init = FALSE)
```

Arguments

BioGeoBEARS_model_object	The BioGeoBEARS_model object, of class BioGeoBEARS_model
update_init	If TRUE, put the estimates into the initial values in the params_table. Default: FALSE.

Details

params_table$type is typically stored in: BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table.

Value

BioGeoBEARS_model_object Updated version of the BioGeoBEARS_model object, of class BioGeoBEARS_model.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_model_object define_BioGeoBEARS_run

Examples

```
# Define a BioGeoBEARS run object
BioGeoBEARS_run_object = define_BioGeoBEARS_run()
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table

# Set 'j' to be free, i.e. as in a DEC+J model (adding jump dispersal
# to the LAGRANGE DEC model)
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","type"] = "free"
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","init"] = 0.25
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","est"] = 0.25

# Display result
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table

# Update the other parameters
BioGeoBEARS_run_object$BioGeoBEARS_model_object =
calc_linked_params_BioGeoBEARS_model_object(
BioGeoBEARS_model_object=BioGeoBEARS_run_object$BioGeoBEARS_model_object)
```

```
# Display result
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table
```

`calc_loglike_for_optim` *Take model parameters and the data and calculate the log-likelihood*

Description

This function is an input to optim or optimx, the ML estimation routines.

Usage

```
calc_loglike_for_optim(params, BioGeoBEARS_run_object,
  phy, tip_condlikes_of_data_on_each_state,
  print_optim = TRUE, areas_list = areas_list,
  states_list = states_list, force_sparse = force_sparse,
  cluster_already_open = cluster_already_open,
  return_what = "loglike", calc_ancprobs = FALSE)
```

Arguments

`params`	A vector of parameters for optimization.
`BioGeoBEARS_run_object`	Object containing the run parameters and the model.
`phy`	An ape tree object
`tip_condlikes_of_data_on_each_state`	A numeric matrix with rows representing tips, and columns representing states/geographic ranges. The cells give the likelihood of the observation data under the assumption that the tip has that state; typically this means that the known geographic range gets a '1' and all other states get a 0.
`force_sparse`	Should sparse matrix exponentiation be used?
`print_optim`	If TRUE (default), print the optimization steps as ML estimation progresses.
`areas_list`	A list of the desired area names/abbreviations/letters (?).
`states_list`	A list of the possible states/geographic ranges, in 0-based index form.
`cluster_already_open`	If the user wants to distribute the matrix exponentiation calculations from all the branches across a number of processors/nodes on a cluster, specify the cluster here. E.g. `cluster_already_open = makeCluster(rep("localhost",num_cores_to_use), type "SOCK")`. Note: this will work on most platforms, including Macs running R from command line, but will NOT work on Macs running the R GUI `R.app`, because parallel processing functions like `MakeCluster` from e.g. `library(parallel)` for some reason crash R.app. The program runs a check for R.app and will just run on 1 node if found.

`return_what` What should be returned to the user? Options are "loglike" (the log-likelihood of the data under the tree, model, and model parameters), "nodelikes" (the scaled conditional likelihoods at the nodes), "rootprobs" (the relative probability of the geographic ranges/states at the root), or "all" (all of the above in a list). Typically the user will only want to return "loglike" while doing ML optimization, but then return "all" once the ML parameter values have been found.

`calc_ancprobs` Just use this function once, return the anc probs of states.

Value

`ttl_loglike` The log-likelihood of the data under the input model and parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prune_states_list

Examples

```
test=1
```

---

calc_loglike_for_optim_stratified

Take model parameters and the data and calculate the log-likelihood – stratified version

---

Description

This is the stratified version of calc_loglike_for_optim. This function is an input to optim or optimx, the ML estimation routines.

Usage

```
calc_loglike_for_optim_stratified(params,
  BioGeoBEARS_run_object, phy,
  tip_condlikes_of_data_on_each_state,
  print_optim = TRUE, areas_list, states_list,
  force_sparse = FALSE, cluster_already_open = FALSE)
```

Arguments

params	A vector of parameters for optimization.
BioGeoBEARS_run_object	Object containing the run parameters, and the model.
phy	An ape tree object
tip_condlikes_of_data_on_each_state	Conditional likelihoods at tips. A numeric matrix with rows representing tips, and columns representing states/geographic ranges. The cells give the likelihood of the observation data under the assumption that the tip has that state; typically this means that the known geographic range gets a '1' and all other states get a 0.
print_optim	If TRUE (default), print the optimization steps as ML estimation progresses.
areas_list	A list of the desired area names/abbreviations/letters (?).
states_list	A list of the possible states/geographic ranges, in 0-based index form.
force_sparse	Should sparse matrix exponentiation be used? Default FALSE.
cluster_already_open	The cluster object, if it has already been started.

Value

ttl_loglike The log-likelihood of the data under the input model and parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve chainsaw_result

Examples

```
test=1
```

`calc_loglike_sp` *Calculate log-likelihood with a transition matrix and speciation events – byte-compiled*

Description

This is the workhorse function of BioGeoBEARS. It calculates the likelihood of the tip data (the geographic ranges observed at the tips) given a phylogenetic tree, a Q transition matrix specifying the model of range evolution along branches, and a speciation probability matrix specifying the probability of the various possible ancestor–>(Left descendant, Right descendant) range evolution events at phylogenetic nodes/speciation events.

Usage

```
calc_loglike_sp(tip_condlikes_of_data_on_each_state, phy,
  Qmat, spPmat = NULL, min_branchlength = 1e-21,
  return_what = "loglike",
  probs_of_states_at_root = NULL, rootedge = FALSE,
  sparse = FALSE, printlevel = 1, use_cpp = TRUE,
  input_is_COO = FALSE, spPmat_inputs = NULL,
  cppSpMethod = 3, cluster_already_open = NULL,
  calc_ancprobs = FALSE, null_range_allowed = TRUE,
  fixnode = NULL, fixlikes = NULL, stratified = FALSE,
  states_allowed_TF = NULL)
```

Arguments

`tip_condlikes_of_data_on_each_state`
: A numeric matrix with rows representing tips, and columns representing states/geographic ranges. The cells give the likelihood of the observation data under the assumption that the tip has that state; typically this means that the known geographic range gets a '1' and all other states get a 0.

`phy`
: A phylogeny object. The function converts it to pruningwise order.

`Qmat`
: A Q transition matrix representing the along-branch model for the evolution of geographic range, using parameters *d* (dispersal/range expansion), *e* (extinction/range contraction/local extirpation), and perhaps others (e.g. distance). This matrix can be input in either dense or sparse (COO) format, as specified by `input_is_COO`.

`spPmat`
: Default is `NULL`; users should usually use `spPmat_inputs`. `spPmat` is A numeric matrix representing the probability of each ancestor range–>(Left range, Right range) transition at cladogenesis events. There are different ways to represent this matrix. In the simplest representation, this is just a rectangular matrix with numstates rows (representing the ancestral states) and numstates^2

columns (representing all possible descendant pairs). Use of this type of matrix is specified by cppSpMethod=1. It is calculated from a textual speciation matrix (typically spmat in the code) via symbolic_to_relprob_matrix_sp. However, this matrix gets huge and slow for large numbers of states/ranges. cppSpMethod=2 and cppSpMethod=3 implement successively more efficient and faster representation and processing of this matrix in COO-like formats. See rcpp_calc_anclikes_sp_COOprobs for the cppSpMethod=2 method, and rcpp_calc_anclikes_sp_CO for the cppSpMethod=3 method (the fastest).

min_branchlength

Nodes with branches below this branchlength will not be treated as cladogenesis events; instead, they will be treated as if an OTU had been sampled from an anagenetic lineage, i.e. as if you had a direct ancestor. This is useful for putting fossils into the biogeography analysis, when you have fossil species that range through time. (Note: the proper way to obtain such trees, given that most phylogenetic methods force all OTUs to be tips rather than direct ancestors, is another question subject to active research. However, one method might be to just set a branch-length cutoff, and treat any branches sufficiently small as direct ancestors.)

return_what — What should be returned to the user? Options are "loglike" (the log-likelihood of the data under the tree, model, and model parameters), "nodelikes" (the scaled conditional likelihoods at the nodes), "rootprobs" (the relative probability of the geographic ranges/states at the root), or "all" (all of the above in a list). Typically the user will only want to return "loglike" while doing ML optimization, but then return "all" once the ML parameter values have been found.

probs_of_states_at_root

The prior probability of the states/geographic ranges at the root. The default, NULL, effectively means an equal probability for each state (this is also what LAGRANGE assumes; and running with NULL will reproduce exactly the LAGRANGE parameter inferences and log-likelihood).

rootedge — Should the root edge be included in the calculation (i.e., calculate to the bottom of the root), if a root edge is present? Default FALSE.

sparse — Should sparse matrix exponentiation be performed? This should be faster for very large matrices (> 100-200 states), however, the calculations appear to be less accurate. The function will transform a dense matrix to COO format (see mat2coo) if necessary according to the input_is_COO parameter.

printlevel — If >= 1, various amounts of intermediate output will be printed to screen. Note: Intermediate outputs from C++ and FORTRAN functions have been commented out, to meet CRAN guidelines.

use_cpp — Should the C++ routines from cladoRcpp be used to speed up calculations? Default TRUE.

input_is_COO — Is the input Q matrix a sparse, COO-formatted matrix (TRUE) or a standard dense matrix (FALSE). Default FALSE.

spPmat_inputs — A list of parameters so that spPmat (the speciation transition probability matrix) can be calculated on-the-fly, according to the method in cppSpMethod. See example.

cppSpMethod Three C++ methods from cladoRcpp for calculating and using the cladogenesis probability matrix. 1 is slowest but easiest to understand; 3 is fastest. If spPmat_inputs is given, the program will generate the appropriate spPmat on-the-fly, and the user does not have to input the full spPmat manually.

cluster_already_open
If the user wants to distribute the matrix exponentiation calculations from all the branches across a number of processors/nodes on a cluster, specify the cluster here. E.g. cluster_already_open = makeCluster(rep("localhost",num_cores_to_use), type "SOCK"). Note: this will work on most platforms, including Macs running R from command line, but will NOT work on Macs running the R GUI R.app, because parallel processing functions like MakeCluster from e.g. library(parallel) for some reason crash R.app. The program runs a check for R.app and will just run on 1 node if found.

calc_ancprobs Should ancestral state estimation be performed (adds an uppass at the end).

null_range_allowed
Does the state space include the null range?#'

fixnode If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number.

fixlikes The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others.

stratified Default FALSE. If TRUE, you are running a stratified analysis, in which case uppass probs should be calculated elsewhere.

states_allowed_TF
Default NULL. If user gives a vector of TRUE and FALSE values, these states will be set to 0 likelihood throughout the calculations.

Details

This likelihood calculation will be repeated many hundreds or thousands of times in any ML (maximum likelihood) or Bayesian estimation procedure. Thus, if the calculation of the log-likelihood of the data under one set of parameter values is too slow, inference takes days or becomes impossible. However, by using fast matrix exponentiation (package rexpokit) and fast C++ routines for calculating the probabilities of range inheritance scenarios at cladogenesis (package cladoRcpp), major speed gains can be achieved. Most of the complexity in the input parameters and the code serves these more rapid alternatives.

However, note that due to the explosion of the geographic range state space with more geographic areas (see numstates_from_numareas), any computational method that explicitly calculates the likelihood of all states will eventually become unusable between 8-20 areas, depending on details. An alternative method, which is fast for large numbers of areas, is BayArea, by Landis, Matzke, Moore, and Huelsenbeck; see *Landis et al. (2013)*. However, BayArea does not currently implement cladogenesis models; it only has continuous-time model for evolutionary change along branches. In effect, this means that the cladogenesis model is sympatric speciation with complete range copying with probability 1.

Value

Return whatever is specified by return_what.

Note

Go BEARS!

(COO = Coordinate list format for a matrix, see http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

Author(s)

Nicholas Matzke <matzke@berkeley.edu>

References

Landis_Matzke_etal_2013_BayArea

Matzke_2012_IBS

ReeSmith2008

See Also

calc_loglike_sp, rcpp_calc_anclikes_sp, rcpp_calc_anclikes_sp_COOprobs, rcpp_calc_anclikes_sp_COOweight
mat2coo, rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

---

calc_loglike_sp_prebyte

Calculate log-likelihood with a transition matrix and speciation events – pre-byte-compiled

---

Description

This function is the pre-byte-compiled version of calc_loglike_sp.

Usage

```
calc_loglike_sp_prebyte(tip_condlikes_of_data_on_each_state,
  phy, Qmat, spPmat = NULL, min_branchlength = 1e-21,
  return_what = "loglike",
  probs_of_states_at_root = NULL, rootedge = FALSE,
  sparse = FALSE, printlevel = 1, use_cpp = TRUE,
  input_is_COO = FALSE, spPmat_inputs = NULL,
  cppSpMethod = 3, cluster_already_open = NULL,
  calc_ancprobs = FALSE, null_range_allowed = TRUE,
  fixnode = NULL, fixlikes = NULL, stratified = FALSE,
  states_allowed_TF = NULL)
```

Arguments

`tip_condlikes_of_data_on_each_state`
: A numeric matrix with rows representing tips, and columns representing states/geographic ranges. The cells give the likelihood of the observation data under the assumption that the tip has that state; typically this means that the known geographic range gets a '1' and all other states get a 0.

`phy`
: A phylogeny object. The function converts it to pruningwise order.

`Qmat`
: A Q transition matrix representing the along-branch model for the evolution of geographic range, using parameters *d* (dispersal/range expansion), *e* (extinction/range contraction/local extirpation), and perhaps others (e.g. distance). This matrix can be input in either dense or sparse (COO) format, as specified by `input_is_COO`.

`spPmat`
: Default is `NULL`; users should usually use `spPmat_inputs`. `spPmat` is A numeric matrix representing the probability of each ancestor range–>(Left range, Right range) transition at cladogenesis events. There are different ways to represent this matrix. In the simplest representation, this is just a rectangular matrix with numstates rows (representing the ancestral states) and numstates^2 columns (representing all possible descendant pairs). Use of this type of matrix is specified by `cppSpMethod=1`. It is calculated from a textual speciation matrix (typically `spmat` in the code) via `symbolic_to_relprob_matrix_sp`. However, this matrix gets huge and slow for large numbers of states/ranges. `cppSpMethod=2` and `cppSpMethod=3` implement successively more efficient and faster representation and processing of this matrix in COO-like formats. See `rcpp_calc_anclikes_sp_COOprobs` for the `cppSpMethod=2` method, and `rcpp_calc_anclikes_sp_CO` for the `cppSpMethod=3` method (the fastest).

`min_branchlength`
: Nodes with branches below this branchlength will not be treated as cladogenesis events; instead, they will be treated as if an OTU had been sampled from an anagenetic lineage, i.e. as if you had a direct ancestor. This is useful for putting fossils into the biogeography analysis, when you have fossil species that range through time. (Note: the proper way to obtain such trees, given that most phylogenetic methods force all OTUs to be tips rather than direct ancestors, is another question subject to active research. However, one method might be to just set a branch-length cutoff, and treat any branches sufficiently small as direct ancestors.)

`return_what`
: What should be returned to the user? Options are "loglike" (the log-likelihood of the data under the tree, model, and model parameters), "nodelikes" (the scaled conditional likelihoods at the nodes), "rootprobs" (the relative probability of the geographic ranges/states at the root), or "all" (all of the above in a list). Typically the user will only want to return "loglike" while doing ML optimization, but then return "all" once the ML parameter values have been found.

`probs_of_states_at_root`
: The prior probability of the states/geographic ranges at the root. The default, `NULL`, effectively means an equal probability for each state (this is also what `LAGRANGE` assumes; and running with NULL will reproduce exactly the `LAGRANGE` parameter inferences and log-likelihood).

`rootedge` Should the root edge be included in the calculation (i.e., calculate to the bottom of the root), if a root edge is present? Default `FALSE`.

`sparse` Should sparse matrix exponentiation be performed? This should be faster for very large matrices (> 100-200 states), however, the calculations appear to be less accurate. The function will transform a dense matrix to COO format (see `mat2coo`) if necessary according to the `input_is_COO` parameter.

`printlevel` If >= 1, various amounts of intermediate output will be printed to screen. Note: Intermediate outputs from C++ and FORTRAN functions have been commented out, to meet CRAN guidelines.

`use_cpp` Should the C++ routines from `cladoRcpp` be used to speed up calculations? Default `TRUE`.

`input_is_COO` Is the input Q matrix a sparse, COO-formatted matrix (`TRUE`) or a standard dense matrix (`FALSE`). Default `FALSE`.

`spPmat_inputs` A list of parameters so that `spPmat` (the speciation transition probability matrix) can be calculated on-the-fly, according to the method in `cppSpMethod`. See example.

`cppSpMethod` Three C++ methods from cladoRcpp for calculating and using the cladogenesis probability matrix. 1 is slowest but easiest to understand; 3 is fastest. If `spPmat_inputs` is given, the program will generate the appropriate spPmat on-the-fly, and the user does not have to input the full `spPmat` manually.

`cluster_already_open`
If the user wants to distribute the matrix exponentiation calculations from all the branches across a number of processors/nodes on a cluster, specify the cluster here. E.g. `cluster_already_open = makeCluster(rep("localhost",num_cores_to_use), type "SOCK")`. Note: this will work on most platforms, including Macs running R from command line, but will NOT work on Macs running the R GUI `R.app`, because parallel processing functions like `MakeCluster` from e.g. `library(parallel)` for some reason crash R.app. The program runs a check for R.app and will just run on 1 node if found.

`calc_ancprobs` Should ancestral state estimation be performed (adds an uppass at the end).

`null_range_allowed`
Does the state space include the null range? Default is `NULL` which means running on a single processor.

`fixnode` If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number.

`fixlikes` The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others.

`stratified` Default FALSE. If TRUE, you are running a stratified analysis, in which case uppass probs should be calculated elsewhere.

`states_allowed_TF`
Default NULL. If user gives a vector of TRUE and FALSE values, these states will be set to 0 likelihood throughout the calculations.

Details

Byte-compiling is supposed to speed up functions; this is an attempt to do this on the rexpokit function expokit_dgpadm_Qmat. It is also possible to byte-compile everything during package installation (via ByteCompile: true in the DESCRIPTION file), which is implemented in BioGeoBEARS, so this may be redundant.

calc_loglike_sp_prebyte gets byte-compiled into calc_loglike_sp.

See http://dirk.eddelbuettel.com/blog/2011/04/12/ for discussion of the compile package.

Value

Return whatever is specified by return_what.

Note

Go BEARS!

(COO = Coordinate list format for a matrix, see http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

Author(s)

Nicholas Matzke <matzke@berkeley.edu>

References

Matzke_2012_IBS

ReeSmith2008

Landis_Matzke_etal_2013_BayArea

See Also

calc_loglike_sp, rcpp_calc_anclikes_sp, rcpp_calc_anclikes_sp_COOprobs, rcpp_calc_anclikes_sp_COOweight
mat2coo, rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

`calc_loglike_sp_stratified` *Calculate log-likelihood with a transition matrix and speciation events, and with stratification*

Description

This function is the stratified version of `calc_loglike_sp`.

Usage

```
calc_loglike_sp_stratified(tip_condlikes_of_data_on_each_state,
  phy, Qmat = NULL, spPmat = NULL,
  min_branchlength = 1e-21, return_what = "loglike",
  probs_of_states_at_root = NULL, rootedge = TRUE,
  sparse = FALSE, printlevel = 0, use_cpp = TRUE,
  input_is_COO = FALSE, spPmat_inputs = NULL,
  cppSpMethod = 3, cluster_already_open = NULL,
  calc_ancprobs = FALSE, null_range_allowed = TRUE,
  fixnode = NULL, fixlikes = NULL, inputs = inputs,
  allareas = allareas, all_states_list = all_states_list,
  return_condlikes_table = FALSE,
  calc_TTL_loglike_from_condlikes_table = TRUE)
```

Arguments

`tip_condlikes_of_data_on_each_state`
: A numeric matrix with rows representing tips, and columns representing states/geographic ranges. The cells give the likelihood of the observation data under the assumption that the tip has that state; typically this means that the known geographic range gets a '1' and all other states get a 0.

`phy`
: A phylogeny object. The function converts it to pruningwise order.

`Qmat`
: A Q transition matrix representing the along-branch model for the evolution of geographic range, using parameters *d* (dispersal/range expansion), *e* (extinction/range contraction/local extirpation), and perhaps others (e.g. distance). This matrix can be input in either dense or sparse (COO) format, as specified by `input_is_COO`.

`spPmat`
: Default is `NULL`; users should usually use `spPmat_inputs`. `spPmat` is A numeric matrix representing the probability of each ancestor range–>(Left range, Right range) transition at cladogenesis events. There are different ways to represent this matrix. In the simplest representation, this is just a rectangular matrix with numstates rows (representing the ancestral states) and numstates^2 columns (representing all possible descendant pairs). Use of this type of matrix is specified by `cppSpMethod=1`. It is calculated from a textual speciation matrix (typically `spmat` in the code) via `symbolic_to_relprob_matrix_sp`. However, this matrix gets huge and slow for large numbers of states/ranges.

cppSpMethod=2 and cppSpMethod=3 implement successively more efficient and faster representation and processing of this matrix in COO-like formats. See rcpp_calc_anclikes_sp_COOprobs for the cppSpMethod=2 method, and rcpp_calc_anclikes_sp_CO for the cppSpMethod=3 method (the fastest).

min_branchlength
: Nodes with branches below this branchlength will not be treated as cladogenesis events; instead, they will be treated as if an OTU had been sampled from an anagenetic lineage, i.e. as if you had a direct ancestor. This is useful for putting fossils into the biogeography analysis, when you have fossil species that range through time. (Note: the proper way to obtain such trees, given that most phylogenetic methods force all OTUs to be tips rather than direct ancestors, is another question subject to active research. However, one method might be to just set a branch-length cutoff, and treat any branches sufficiently small as direct ancestors.)

return_what
: What should be returned to the user? Options are "loglike" (the log-likelihood of the data under the tree, model, and model parameters), "nodelikes" (the scaled conditional likelihoods at the nodes), "rootprobs" (the relative probability of the geographic ranges/states at the root), or "all" (all of the above in a list). Typically the user will only want to return "loglike" while doing ML optimization, but then return "all" once the ML parameter values have been found.

probs_of_states_at_root
: The prior probability of the states/geographic ranges at the root. The default, NULL, effectively means an equal probability for each state (this is also what LAGRANGE assumes; and running with NULL will reproduce exactly the LAGRANGE parameter inferences and log-likelihood).

rootedge
: Should the root edge be included in the calculation (i.e., calculate to the bottom of the root), if a root edge is present? Default FALSE.

sparse
: Should sparse matrix exponentiation be performed? This should be faster for very large matrices (> 100-200 states), however, the calculations appear to be less accurate. The function will transform a dense matrix to COO format (see mat2coo) if necessary according to the input_is_COO parameter.

printlevel
: If >= 1, various amounts of intermediate output will be printed to screen. Note: Intermediate outputs from C++ and FORTRAN functions have been commented out, to meet CRAN guidelines.

use_cpp
: Should the C++ routines from cladoRcpp be used to speed up calculations? Default TRUE.

input_is_COO
: Is the input Q matrix a sparse, COO-formatted matrix (TRUE) or a standard dense matrix (FALSE). Default FALSE.

spPmat_inputs
: A list of parameters so that spPmat (the speciation transition probability matrix) can be calculated on-the-fly, according to the method in cppSpMethod. See example.

cppSpMethod
: Three C++ methods from cladoRcpp for calculating and using the cladogenesis probability matrix. 1 is slowest but easiest to understand; 3 is fastest. If spPmat_inputs is given, the program will generate the appropriate spPmat on-the-fly, and the user does not have to input the full spPmat manually.

cluster_already_open
: If the user wants to distribute the matrix exponentiation calculations from all the branches across a number of processors/nodes on a cluster, specify the cluster here. E.g. `cluster_already_open = makeCluster(rep("localhost",num_cores_to_use), type "SOCK")`. Note: this will work on most platforms, including Macs running R from command line, but will NOT work on Macs running the R GUI `R.app`, because parallel processing functions like `MakeCluster` from e.g. `library(parallel)` for some reason crash R.app. The program runs a check for R.app and will just run on 1 node if found.

calc_ancprobs
: Should ancestral state estimation be performed (adds an uppass at the end).

null_range_allowed
: Does the state space include the null range? Default is `NULL` which means running on a single processor.

fixnode
: If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number. (Trial implementation for stratified analysis.)

fixlikes
: The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others. (Trial implementation for stratified analysis.)

inputs
: A list of inputs containing the dispersal matrix for each time period, etc.

allareas
: A list of all the areas in the total analysis

all_states_list
: A list of all the stats in the total analysis (0-based coding - ?)

return_condlikes_table
: If `TRUE`, return the table of ALL conditional likelihood results, including at branch subsections (only some should be used in calculating the final log-likelihood of the geography range data on the tree!)

calc_TTL_loglike_from_condlikes_table
: If TRUE, force making of the condlikes table, and use it to calculate the log-likelihood (default=TRUE; matches LAGRANGE).

Value

grand_total_likelihood The total log-likelihood of the data on the tree (default). Or, if `return_condlikes_table==TRUE`, the function returns `calc_loglike_sp_stratified_results`, with `calc_loglike_sp_stratified_results$condlikes_` and `calc_loglike_sp_stratified_results$grand_total_likelihood` as list items. This can be useful for debugging stratified analyses, which have a lot of extra book-keeping that is easy to mess up.

Note

Go BEARS!

(COO = Coordinate list format for a matrix, see http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

Author(s)

Nicholas Matzke <matzke@berkeley.edu>

References

Matzke_2012_IBS

ReeSmith2008

Landis_Matzke_etal_2013_BayArea

See Also

calc_loglike_sp, rcpp_calc_anclikes_sp, rcpp_calc_anclikes_sp_COOprobs, rcpp_calc_anclikes_sp_COOweight
mat2coo, rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

`calc_obs_like` *Calculate likelihood of count data given true presence/absence and parameters*

Description

This function calculates P(data|presence,parameters), i.e. the probability of some detection and taphonomic control counts, given the true geographic range/state, and parameters such as `dp`, a detection probability (and, optionally, a false detection probability, `fdp`).

Usage

```
calc_obs_like(truly_present = TRUE, obs_target_species,
  obs_all_species, mean_frequency = 0.1, dp = 1, fdp = 0)
```

Arguments

`truly_present`	Is the OTU of interest known/conditionally assumed to be truly present (`TRUE`) or truly absent (`FALSE`)?
`obs_target_species`	A count of detections of your OTU of interest, e.g. as produced from a cell of the matrix output from `read_detections`.
`obs_all_species`	A count of detections of your taphonomic controls, e.g. as produced from a cell of the output from `read_controls`.
`mean_frequency`	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of

detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.

`dp` The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.

`fdp` The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer `mean_frequency`, `dp` and `fdp` all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.

Details

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

`lnlike_allobs_given_absence` The natural log-likelihood of the data, given the model & assumption of true presence or absence.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

mapply_calc_post_prob_presence, calc_post_prob_presence, mapply_calc_obs_like

Examples

```
# Example: 10 observations of the species mean dramatically higher likelihood of the
# data on the hypothesis that it is truly present.

# With zero error rate
obs_target_species = 10
obs_all_species = 100
mean_frequency=0.1
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence
# Note that the probability of getting detections, under the hypothesis of
# true absence, is -Inf


# With a small error rate, there is some small but positive probability of
# falsely getting 10 detections
obs_target_species = 10
obs_all_species = 100
mean_frequency=0.1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence
# i.e. the prob. of the data is 1 under the hypothesis of presence, and 0
# under the hypothesis of absence (ln(prob) = 0 & -Inf, respectively)


# Note that with very high error rates, your conclusion could reverse
obs_target_species = 10
obs_all_species = 100
mean_frequency=0.1
dp=0.5
fdp=0.3
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
```

```
LnL_under_absence


# Example #2 -- what if you have ZERO detections, but lots of detections
# of your taphonomic control?
obs_target_species = 0
obs_all_species = 1
mean_frequency=0.1
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence

# With a slight error rate
obs_target_species = 0
obs_all_species = 1
mean_frequency=0.1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence


obs_target_species = 0
obs_all_species = 2
mean_frequency=0.1
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence

# With a slight error rate
obs_target_species = 0
obs_all_species = 2
mean_frequency=0.1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
```

```
LnL_under_presence
LnL_under_absence




# Example #3 -- what if you have ZERO detections, but only a few
# detections of your taphonomic control?
obs_target_species = 0
obs_all_species = 100
mean_frequency=0.1
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence

# With a slight error rate
obs_target_species = 0
obs_all_species = 100
mean_frequency=0.1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence



# Special cases -- e.g., no data
# Prob(data)=1, ln(prob)=0
obs_target_species = 0
obs_all_species = 0
mean_frequency=0.1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence

obs_target_species = 0
obs_all_species = 0
mean_frequency=0.1
```

```
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence


# What if, for some reason, you put in identical detections and taphonomic control
# counts? (e.g., you load in a standard tipranges file)
obs_target_species = 1
obs_all_species = 1
mean_frequency=1
dp=1
fdp=0
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence

# What if, for some reason, you put in identical detections and taphonomic control
# counts? (e.g., you load in a standard tipranges file)
obs_target_species = 1
obs_all_species = 1
mean_frequency=1
dp=0.99
fdp=0.001
LnL_under_presence = calc_obs_like(truly_present=TRUE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_absence = calc_obs_like(truly_present=FALSE, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
LnL_under_presence
LnL_under_absence
```

`calc_post_prob_presence` *Calculate posterior probability of presence, given count data and parameters*

Description

This function calculates P(presence|count data,parameters), i.e. the posterior probability of presence in an area, given data on detection counts and taphonomic control counts, and a detection model with the parameters mean_frequency, dp, a detection probability (and, optionally, a false detection probability, `fdp`).

Usage

```
calc_post_prob_presence(prior_prob_presence = 0.01,
  obs_target_species, obs_all_species,
  mean_frequency = 0.1, dp = 1, fdp = 0,
  print_progress = "")
```

Arguments

prior_prob_presence	The prior probability of presence, i.e. when no detection or taphonomic control data whatsoever is available. Default is set to 0.01 which expresses my totally uninformed bias that in whatever your data is, your species of interest probably doesn't live in the typical area you are looking at.
obs_target_species	A count of detections of your OTU of interest, e.g. as produced from a cell of the matrix output from read_detections.
obs_all_species	A count of detections of your taphonomic controls, e.g. as produced from a cell of the output from read_controls.
mean_frequency	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.
dp	The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.
fdp	The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer mean_frequency, dp and fdp all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.
print_progress	If not the default (""), print whatever is in print_progress, followed by a space (for error checking/surveying results).

Details

Essentially, this function combines a prior probability, with the likelihood function (coded in calc_obs_like) to produce a posterior probability of presence given Bayes' Theorem (Bayes & Price, 1763).

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

post_prob The posterior probability of presence, given the prior probability, the model parameters, and the data.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Bayes'_theorem

Matzke_2012_IBS

Bottjer_Jablonski_1988

Bayes_1763

See Also

calc_obs_like, mapply_calc_post_prob_presence, mapply_calc_obs_like

Examples

```
# Calculate posterior probability of presence in an area,
# given a dp (detection probability) and detection model.

# With zero error rate
obs_target_species = 10
obs_all_species = 100
```

```
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob
# i.e., with perfect detection, the prob. of presence is 1 under the
# hypothesis of presence, and 0 under the hypothesis of
# (This is because the likelihood of the data under
# presence and absence are ln(prob) = 0 & -Inf, respectively.)


# Note that with very high error rates, your conclusion could reverse
obs_target_species = 10
obs_all_species = 100
mean_frequency=0.1
dp=0.5
fdp=0.3
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob


# With 0 error rate, even 1 observation makes P(presence) = 1
obs_target_species = 1
obs_all_species = 100
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob


# With a small error rate, there is some small but positive probability of
# falsely getting 10 detections; but it may be effectively 0
obs_target_species = 10
obs_all_species = 100
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob


# If you have only 1 detection, and you have 100 taphonomic controls and
# a mean_frequency of sampling the OTU of interest of 0.1, then there is
# still a very low probability of presence (since, under your model,
```

```
# you should expect to see about 10 detections, not 1)
obs_all_species = 100
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01

obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)



# Note how quickly this chances if you drop the mean_frequency from 0.1
# to 0.01. This means that if you want single detections to count for
# a lot, you need either a low mean_frequency which matches the observed
# frequency, or an extremely high/perfect detection probability (dp).
obs_all_species = 100
mean_frequency=0.01
dp=0.99
fdp=0.001
prior_prob_presence = 0.01

obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)


# Changing mean_frequency from 0.01 to 0.001 actually LOWERS the posterior
```

```
# probability of presence based on 1 detection, as we have a somewhat
# significant false detection rate:
obs_target_species = 1
obs_all_species = 100
mean_frequency=0.001
dp=0.99
fdp=0.001
prior_prob_presence = 0.01

obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)


# Change false detection probability to a much lower value
obs_all_species = 100
mean_frequency=0.001
dp=0.99
fdp=0.00001
prior_prob_presence = 0.01

obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)




# Change false detection probability to 0
```

```
obs_all_species = 100
mean_frequency=0.001
dp=0.99
fdp=0.0
prior_prob_presence = 0.01


obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)




# Change mean_frequency to 0.001
obs_all_species = 100
mean_frequency=0.001
dp=0.99
fdp=0.0
prior_prob_presence = 0.01

obs_target_species = 0
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 1
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 2
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)

obs_target_species = 3
calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
```

```
# Example #2 -- what if you have ZERO detections, but lots of detections
# of your taphonomic control?
obs_target_species = 0
obs_all_species = 100
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob

# With a slight error rate
obs_target_species = 0
obs_all_species = 100
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob


obs_target_species = 0
obs_all_species = 2
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob

# With a slight error rate
obs_target_species = 0
obs_all_species = 2
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob
```

```
# Example #3 -- what if you have ZERO detections, but only a few
# detections of your taphonomic control?
obs_target_species = 0
obs_all_species = 1
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob

# With a slight error rate
obs_target_species = 0
obs_all_species = 1
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob



# Special cases -- e.g., no data
# Prob(data)=1, ln(prob)=0
obs_target_species = 0
obs_all_species = 0
mean_frequency=0.1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob

obs_target_species = 0
obs_all_species = 0
mean_frequency=0.1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob


# What if, for some reason, you put in identical detections and taphonomic control
# counts? (e.g., you load in a standard tipranges file)
```

```
obs_target_species = 1
obs_all_species = 1
mean_frequency=1
dp=1
fdp=0
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob

# What if, for some reason, you put in identical detections and taphonomic control
# counts? (e.g., you load in a standard tipranges file)
obs_target_species = 1
obs_all_species = 1
mean_frequency=1
dp=0.99
fdp=0.001
prior_prob_presence = 0.01
post_prob = calc_post_prob_presence(prior_prob_presence, obs_target_species,
obs_all_species, mean_frequency, dp, fdp)
post_prob
```

`calc_prob_forward_onebranch_dense` *Dense matrix exponentiation forward on a branch, with rexpokit*

Description

Take input probabilities, and get the probabilities at the end of a branch using matrix exponentiation.

Usage

```
calc_prob_forward_onebranch_dense(relprobs_branch_bottom,
  branch_length, Qmat)
```

Arguments

`relprobs_branch_bottom`	The relative probability of each state at the base of the branch (should sum to 1).
`branch_length`	The length of the branch.
`Qmat`	A Q transition matrix in square (dense) format

Details

The `calc_loglike_sp` function calculates most transition probabilities internally via `rexpokit`. These are then stored and can be used again when an uppass is being done for ancestral state estimates. However, if there is a root branch below the lowest fork, the uppass needs to calculate the forward probabilities.

Value

actual_probs_after_forward_exponentiation The probabilities of each state at the top of the branch.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.bioinf.org/molsys/data/idiots.pdf

Matzke_2012_IBS

FosterIdiots

See Also

expokit_dgpadm_Qmat2, expokit_dgpadm_Qmat, rexpokit

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

relprobs_branch_bottom = c(0.25, 0.25, 0.25, 0.25)

# Make a series of t values
branch_length = 0.1

calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length, Qmat)
calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length=0.5, Qmat)
calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length=1, Qmat)
calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length=2, Qmat)
calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length=10, Qmat)
calc_prob_forward_onebranch_dense(relprobs_branch_bottom, branch_length=20, Qmat)
```

calc_prob_forward_onebranch_sparse *Sparse matrix exponentiation forward on a branch, with rexpokit*

Description

Take input probabilities, and get the probabilities at the end of a branch using matrix exponentiation.

Usage

```
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom,
  branch_length, tmpQmat_in_REXPOKIT_coo_fmt,
  coo_n = coo_n, anorm = anorm, check_for_0_rows = TRUE,
  TRANSPOSE_because_forward = TRUE)
```

Arguments

relprobs_branch_bottom	The relative probability of each state at the base of the branch (should sum to 1).
branch_length	The length of the branch.
tmpQmat_in_REXPOKIT_coo_fmt	A Q transition matrix in sparse (COO) format. See mat2coo.
coo_n	If a COO matrix is input, coo_n specified the order (# rows, equals # columns) of the matrix.
anorm	dgexpv requires an initial guess at the norm of the matrix. Using the R function norm might get slow with large matrices. If so, the user can input a guess manually (Lagrange seems to just use 1 or 0, if I recall correctly).
check_for_0_rows	If TRUE or a numeric value, the input Qmat is checked for all-zero rows, since these will crash the FORTRAN wrapalldmexpv function. A small nonzero value set to check_for_0_rows or the default (0.0000000000001) is input to off-diagonal cells in the row (and the diagonal value is normalized), which should fix the problem.
TRANSPOSE_because_forward	For non-time-reversible models, the forward calculation is different than the backward one. Fortunately this just means switching the rows and columns of a transition matrix.

Details

The calc_loglike_sp function calculates most transition probabilities internally via rexpokit. These are then stored and can be used again when an uppass is being done for ancestral state estimates. However, if there is a root branch below the lowest fork, the uppass needs to calculate the forward probabilities.

Value

actual_probs_after_forward_exponentiation The probabilities of each state at the top of the branch.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.bioinf.org/molsys/data/idiots.pdf

Matzke_2012_IBS

FosterIdiots

See Also

expokit_dgpadm_Qmat2, expokit_dgpadm_Qmat, rexpokit

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
require("rexpokit")

Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504,
0.168, 0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)
tmpQmat_in_REXPOKIT_coo_fmt = mat2coo(Qmat)

relprobs_branch_bottom = c(0.25, 0.25, 0.25, 0.25)

# Make a series of t values
branch_length = 0.1

calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length,
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length=0.5,
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length=1,
```

```
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length=2,
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length=10,
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
calc_prob_forward_onebranch_sparse(relprobs_branch_bottom, branch_length=20,
tmpQmat_in_REXPOKIT_coo_fmt, coo_n=4, anorm=1, check_for_0_rows=TRUE,
TRANSPOSE_because_forward=TRUE)
```

`chainsaw2` *Saw a tree off at a particular time before present*

Description

This function chops a tree like a hedge-trimmer, cutting straight across at a particular timepoint. The pieces are returned, as is the leftover tree, with branches shortened appropriately. Pieces that are mini-trees are returned as ape objects, whereas single branches are just lengths.

Usage

```
chainsaw2(tr, timepoint = 10, return_pieces = TRUE)
```

Arguments

`tr`	An ape phylo object.
`timepoint`	The time at which the tree should be "chopped".
`return_pieces`	Default TRUE, which means pieces should be returned

Details

This function is used during stratification, but could have other uses as well.

Value

`chainsaw_result` (a list object with the pieces) or `tree_to_chainsaw`, just the leftover tree

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

section_the_tree

Examples

```
test=1
```

`check_BioGeoBEARS_run` *Check the inputs for various problems*

Description

Numerous subtle mistakes in the input files for a BioGeoBEARS run can cause the run to crash. As I come across these, I am putting in error checks for them.

Usage

```
check_BioGeoBEARS_run(inputs, allow_huge_ranges = FALSE)
```

Arguments

`inputs`	The inputs list
`allow_huge_ranges`	Default FALSE, which will stop the run if there are more than 500 states. If TRUE, this will just print a warning, and continue, at which point you will wait for weeks or forever for the analysis to finish. See cladoRcpp's numstates_from_numareas function to calculate the size of the state space ahead of time, and links therein to see how the number of states scales with areas (2^number of areas, in an unconstrained analysis), how the size of the transition matrix you will be exponentiating scales (size = numstates * numstates), and the size of the ancestor/left-descendant/right-descendant cladogenesis matrix scales (numstates * numstates * numstates). At 500 states, this is 500^3 = 125,000,000 combinations of ancestor/left/right to check at every cladogenesis event, although cladoRcpp's tricks speed this up substantially.

Details

Some include:

- Trees with negative branchlengths (as produced sometimes by e.g. BEAST MCC consensus trees (MCC = majority clade consensus). These trees are always fully resolved, but the median node heights can sometimes be behind the node position in the tree. Users should fix this manually, pathological results or crashes will result otherwise.

- Trees with polytomies. BioGeoBEARS (and LAGRANGE, and DIVA) assume a model where lineages bifurcate, and never multifurcate. Users can convert multifurcating trees to bifurcating trees with APE's multi2di (they will have to decide what branchlength to use for the new branches; it should be small, but bigger than the minimum branchlength used to identify fossils hooks (as hooks are considered to be anagenetic members of a lineage, and thus are connected to the tree without a cladogenesis event invoked). Users can then run their analysis several times on differently-resolved trees.

NOTE: After the above correction, users may wish to correct the tip branchlengths (or make some other adjustment) so that all the tips are at age 0 my before present, as in an ultrametric tree. (However, note that trees with fossil tips are not ultrametric according to APE's is.ultrametric, even though they are time-scaled. To make living (nonfossil) tips line up to zero, see average_tr_tips or the (different!) . They should be used with care. Alternately, a small amount of error in tip heights will make very little difference in the likelihood calculations (e.g. if some tips are 0.1 my too high, but the tree spans 200 my), which would be an argument for not requiring perfection after the (crucial) corrections of negative branchlengths, zero-branchlengths, and polytomies have been made.

- Check for an absurdly large number of states. I've set the limit at 500 (it starts getting slow around 200), users can override with allow_huge_ranges=TRUE.

- Geography tipranges files should have same number of area labels as columns.

- Geography tipranges files should have same number of taxa as the tree, and with the (exact!!) same names. This can be the source of many headaches, as different programs (Mesquite, etc.) treat spaces, periods, etc. in different ways, and re-write tipnames with/without quotes, underscores, etc.; and in my experience, my biologist colleagues find it very difficult to guarantee that the tipnames in their tree and their data tables will match exactly. The SAFEST approach is to NEVER use these characters in tipnames or table names: space, comma, semicolon, dashes, parentheses, brackets, apostrophes or quote marks, or periods. Use ONLY letters, numbers, and underscores (_). When plotting trees, APE automatically reads underscores as spaces, which is nice for display.

- There must be the same or more timeperiods than the other stratified items (distances matrices, etc.)

Value

TRUE if no errors found; otherwise a stop() is called.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

average_tr_tips,

Examples

```
test=1
```

---

check_if_state_is_allowed *Check if a geographic range/state is allowed, given an areas-allowed matrix.*

Description

If the user has specified a matrix stating which areas are allowed to be connected (and thus have a species with a range in both areas), this function checks if the input list of areas (as a 0-based vector of areas) in a single state/geographic range is consistent with the areas-allowed matrix.

Usage

```
check_if_state_is_allowed(state_0based_indexes,
  areas_allowed_mat)
```

Arguments

state_0based_indexes	The input state is a 0-based vector of area indices.
areas_allowed_mat	A matrix (number of areas x number of areas) with 1s indicating allowed connections between areas, and 0s indicating disallowed connections.

Details

This function may be used by e.g. apply.

Value

TRUE or FALSE

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

apply

Examples

```
test=1
```

---

cls.df *Get the class for each column in a list*

Description

This function returns the class of each column in a data.frame.

Usage

```
cls.df(dtf, printout = FALSE)
```

Arguments

dtf	Input data.frame.
printout	Print the results to screen, if desired.

Details

R does lots of weird and unpredictable things when you build up tables/matrices/data.frames by e.g. cbind and rbind on vectors of results. The major problems are (1) columns get made into class list; (2) numeric columns are converted to class factor; (3) numeric columns are converted to class character; (4) you have a matrix when you think you have a data.frame.

All of this could be taken care of by detailed understanding and tracking of when R recasts values in vectors, matrices, and data frames...but this is a huge pain, it is easier to just have a function that jams everything back to a data.frame with no lists, no factors, and with columns being numeric where possible. See dfnums_to_numeric and unlist_df4 for these options.

Value

dtf_classes A data.frame showing the column, column name, and column class.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

dfnums_to_numeric, unlist_df4

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)

x = matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2)
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)

x = adf(matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2))
names(x) = c("A","B")
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)
```

---

colors_legend *Plot a colors legend for geographic ranges*

Description

Like it says.

Usage

```
colors_legend(possible_ranges_list_txt,
  colors_list_for_states, legend_ncol = NULL,
  legend_cex = 1)
```

Arguments

possible_ranges_list_txt	A list of the allowed ranges/states
colors_list_for_states	The corresponding colors
legend_ncol	The number of columns in the legend. If NULL (default), the function calculates floor(sqrt(length(possible_ranges_list_txt) / 2)). Note that when you have hundreds of states, there is probably no good way to have a coherent legend, and it is easier to just rely upon printing the character codes for the ML states in the plots, with the colors, and users can then see and trace the common colors/states by eye.
legend_cex	The cex (character expansion size) for the legend. Defaults to 1, which means the legend function determines the size. The value 2.5 works well for 15 or 16 states/ranges.

Value

Nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

legend, floor, ceiling, floor

Examples

```
testval=1
```

conditional_format_cell *Conditionally format a number (mostly)*

Description

When a table has numbers that range over many orders of magnitude, it can be very distracting if the display program forces each column to the same format. This function formats a cell much like Excel would.

Usage

```
conditional_format_cell(cellval,
  numbers_below_this_get_scientific = 1e-04,
  numdigits_for_superlow_scientific = 1,
  numbers_above_this_get_scientific = 1e+07,
  numdigits_for_superhigh_scientific = 2,
  numdigits_inbetween_have_fixed_digits = 4)
```

Arguments

cellval	The cell value to format.
numbers_below_this_get_scientific	When the absolute value of a number is below this value, scientific notation is used.
numdigits_for_superlow_scientific	Number of digits after the '.' for scientific notation of small numbers.
numbers_above_this_get_scientific	When the absolute value of a number is above this value, scientific notation is used.
numdigits_for_superhigh_scientific	Number of digits after the '.' for scientific notation of large numbers.
numdigits_inbetween_have_fixed_digits	Numbers of medium size have this many fixed digits. Note that other cutoffs are specified in the code, and signif is used to make e.g. integers appear as 0, 1, 2..

Details

The defaults seem to work well, but could be modified. The current function also extracts just the filename, if a full path is given.

Value

cellval The value, reformatted and of class character.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster
Matzke_2012_IBS

See Also

signif, sprintf

Examples

```
test=1

cellval = 143514514514532
conditional_format_cell(cellval)

cellval = -42.235235
conditional_format_cell(cellval)

cellval = -42.0000000
conditional_format_cell(cellval)

cellval = 0.0000
conditional_format_cell(cellval)

cellval = 0.0001
conditional_format_cell(cellval)

cellval = 0.00001
conditional_format_cell(cellval)

cellval = 0.0000111
conditional_format_cell(cellval)
```

---

conditional_format_table

Conditionally format the numbers (mostly) in a table

Description

When a table has numbers that range over many orders of magnitude, it can be very distracting if the display program forces each column to the same format. This function uses conditional_format_cell via sapply to format a cell much like Excel would.

Usage

```
conditional_format_table(input_table,
  numbers_below_this_get_scientific = 1e-04,
  numdigits_for_superlow_scientific = 1,
  numbers_above_this_get_scientific = 1e+07,
  numdigits_for_superhigh_scientific = 2,
  numdigits_inbetween_have_fixed_digits = 4)
```

Arguments

input_table	The table to format.
numbers_below_this_get_scientific	When the absolute value of a number is below this value, scientific notation is used.
numdigits_for_superlow_scientific	Number of digits after the '.' for scientific notation of small numbers.
numbers_above_this_get_scientific	When the absolute value of a number is above this value, scientific notation is used.
numdigits_for_superhigh_scientific	Number of digits after the '.' for scientific notation of large numbers.
numdigits_inbetween_have_fixed_digits	Numbers of medium size have this many fixed digits. Note that other cutoffs are specified in the code, and signif is used to make e.g. integers appear as 0, 1, 2..

Details

The defaults seem to work well, but could be modified. The current function also extracts just the filename, if a full path is given.

Value

output_table The table, reformatted with cells of class character.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

signif, sprintf

Examples

```
test=1

input_table = adf(c(143514514514532, -42.235235, -42.0000000,
0.0000, 0.0001, 0.00001, 0.0000111))
conditional_format_table(input_table=input_table)
```

`cornerlabels` *Make labels for plotting ranges on corners*

Description

This function makes labels for plotting ranges on corners.

Usage

```
cornerlabels(text, coords, bg = "green3", col = "black",
  adj = c(0.5, 0.5), ...)
```

Arguments

`text`	The text to put at the corners.
`coords`	The coordinates at which to plot the labels
`bg`	The background color
`col`	The text color
`adj`	Position adjustment; default `adj=c(0.5,0.5)`
`...`	Additional arguments to standard functions

Value

nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

cornerpies, corner_coords, get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

`cornerpies` *Make pie charts for plotting ranges on corners*

Description

This function makes pie charts for plotting ranges on corners. It makes use of `ape:::floating.pie.asp` to plot the pie charts on the corners.

Usage

```
cornerpies(pievals, coords, piecol, adj = c(0.5, 0.5),
  ...)
```

Arguments

`pievals`	The matrix (numnodes x numstates) of probabilities to plot.
`coords`	The coordinates at which to plot the labels.
`piecol`	The color for each possible state.
`adj`	Position adjustment; default `adj=c(0.5,0.5)`
`...`	Additional arguments to standard functions

Details

To get the corner coordinates, use corner_coords. Please note the special input required in that function to get it to access a corner-coordinates function in the extensions data (`extdata`) directory.

Value

nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

cornerlabels, corner_coords, get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

---

corner_coords *Get the corner coordinates*

Description

Gets the coordinates of the corners when the tree is plotted.

Usage

```
corner_coords(tr, coords_fun = "plot_phylo3_nodecoords",
  tmplocation = "manual")
```

Arguments

tr	A tree object in phylo format.
coords_fun	The name of the function to use to get node coordinates. Default: "plot_phylo3_nodecoords".
tmplocation	Default is "manual", which throws an error check unless your path structure matches the developer's. Most users should probably use the system.file command in the examples, below. The directory location containing the R script plot_phylo3_nodecoords.R. This function, modified from the ape function plot.phylo, cannot be included directly in the R package as it contains C code that does not pass CRAN's R CMD check. The default, cornercoords_loc="manual", will not allow split states to be plot. The R script plot_phylo3_nodecoords.R is located in the BioGeoBEARS extension data directory, extdata/a_scripts. You should be able to get the full path with list.files(system.file("extdata/a_scripts", packa

Details

Because this function needs to use a modified version of the APE plot.phylo function, and for complex reasons APE's .C functions cannot be used elsewhere without causing problems with R CMD check, this function is left up to user specification. Basically, the user puts in the name of the function, which is available in the extension data (`extdata/a_scripts`) directory of the package. The defaults work on the developer's machine, other users may have to e.g. change "manual" to `tmplocation`, where `tmplocation` is specified as in the example.

Value

`corners_list`

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo, get_nodenums

Examples

```
# Set location like this if you don't have plot_phylo3_nodecoords
# hardcoded/sourced elsehwhere
# tmplocation = np(system.file("extdata/a_scripts", package="BioGeoBEARS"))
#
## Not run:
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(trfn)
tmplocation = np(system.file("extdata/a_scripts", package="BioGeoBEARS"))
corner_coords(tr, coords_fun="plot_phylo3_nodecoords", tmplocation=tmplocation)

## End(Not run)
```

default_states_list *Default input for a states_list*

Description

R CMD check limits the length of inputs to variables for functions; this is a workaround.

Usage

```
default_states_list()
```

Value

`states_list` The list of states

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

make_dispersal_multiplier_matrix

Examples

```
states_list = default_states_list()
```

define_BioGeoBEARS_model_object *Define a BioGeoBEARS_model class and object*

Description

Class BioGeoBEARS_model is an extension of the data.frame class. It is used for holding discrete geographic range data for the tips on a phylogeny. Geographic ranges are represented with bit encoding (0/1) indicating absence or presence in each possible area.

Usage

```
define_BioGeoBEARS_model_object(minval_anagenesis = 1e-15,
  minval_cladogenesis = 1e-05, maxval = 5)
```

Arguments

minval_anagenesis	Minimum value above zero for d, e, a, b parameters.
minval_cladogenesis	Minimum value above zero for j, v, etc.
maxval	Maximum value for d, e, a

Details

This is just a data.frame with: rows = taxanames
columns = area names
cells = 0/1 representing empty/occupied

Value

BioGeoBEARS_model_object The BioGeoBEARS_model object, of class BioGeoBEARS_model

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

areas_list_to_states_list_old, areas_list_to_states_list_new

Examples

```
testval=1
BioGeoBEARS_model_object = define_BioGeoBEARS_model_object()
BioGeoBEARS_model_object
define_BioGeoBEARS_model_object()
```

---

define_BioGeoBEARS_run

Define a maximum likelihood search, perhaps stratified

Description

Set up the inputs object for an ML search. See parameter descriptions for defaults.

Usage

```
define_BioGeoBEARS_run(abbr = "default",
  description = "defaults",
  BioGeoBEARS_model_object = define_BioGeoBEARS_model_object(),
  trfn = "Psychotria_5.2.newick",
  geogfn = "Psychotria_geog.data", timesfn = NA,
  distsfn = NA, dispersal_multipliers_fn = NA,
  area_of_areas_fn = NA, areas_allowed_fn = NA,
  detects_fn = NA, controls_fn = NA, max_range_size = NA,
  states_list = NULL, force_sparse = FALSE,
  use_detection_model = FALSE, print_optim = TRUE,
  num_cores_to_use = NA, cluster_already_open = FALSE,
  use_optimx = TRUE, return_condlikes_table = FALSE,
  calc_TTL_loglike_from_condlikes_table = TRUE,
  calc_ancprobs = TRUE, fixnode = NULL, fixlikes = NULL,
  speedup = TRUE, tmpwd = getwd())
```

Arguments

abbr	Text abbreviation of run, e.g. "default"
description	Text description of run, e.g. "defaults"
BioGeoBEARS_model_object	Default is define_BioGeoBEARS_model_object()
trfn	The filename of the phylogenetic tree, in NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). Tipnames should match the names in geogfn. See read.tree in APE for reading in phylogenetic trees. Default "Psychotria_5.2.newick"

geogfn A PHYLIP-style file with geographic range data (see getranges_from_LagrangePHYLIP) for each tipname. This is the same format used by C++ LAGRANGE (*SmithRee2010_CPPversion*). Default "Psychotria_geog.data"

timesfn Filename for the stratified times.

distsfn Filename for the changing distances.

dispersal_multipliers_fn
Filename for the changing hard-coded dispersal multipliers

area_of_areas_fn
Filename for the area of each area

areas_allowed_fn
Filename for the allowed connections between areas for single-species ranges.

detects_fn Filename for the counts of detections of OTUs of interest. See calc_obs_like.

controls_fn Filename for the counts of taphonomic controls (which INCLUDE the OTUs of interest). See calc_obs_like.

max_range_size The maximum rangesize, in number of areas. Having a smaller maximum range size means that you can have more areas (the size of the state space is greatly reduced; see numstates_from_numareas.

states_list A list of the possible states/geographic ranges, in 0-based index form.

force_sparse Should sparse matrix exponentiation be used? Default FALSE, which means dense matrix exponentiation is always used. If NA, the program will use sparse matrix exponentiation for transition matrices above rank 128 (size 128x128). NOTE: Sparse matrix exponentiation seems to give correlated, but not exact, results, and these errors may accumulate. Presumably the problems become less with larger matrices, but I have not explored this in detail.

use_detection_model
If TRUE, use the detection model (with parameters mf, dp, and fdp) and counts of detections and counts of taphonomic controls to calculate the tip_condlikes_of_data_on_each_sta

print_optim If TRUE (default), print the optimization steps as ML estimation progresses.

tmpwd The working directory in which the input and output files will be placed. Default is getwd. This is stored mostly for future reference; users are responsible for manually navigating to the appropriate directory ahead of time, using setwd.

num_cores_to_use
If >1, parallel processing will be attempted. **Note:** parallel processing via library (parallel) will work in Mac command-line R, but not in Mac GUI R.app.

cluster_already_open
If the user wants to distribute the matrix exponentiation calculations from all the branches across a number of processors/nodes on a cluster, specify the cluster here. E.g. cluster_already_open = makeCluster(rep("localhost",num_cores_to_use), type "SOCK"). Note: this will work on most platforms, including Macs running R from command line, but will NOT work on Macs running the R GUI R.app, because parallel processing functions like MakeCluster from e.g. library(parallel) for some reason crash R.app. The program runs a check for R.app and will just run on 1 node if found.

use_optimx If TRUE, use optimx rather that optim.

return_condlikes_table
: If TRUE, return the table of ALL conditional likelihood results, including at branch subsections (only some should be used in calculating the final log-likelihood of the geography range data on the tree!)

calc_TTL_loglike_from_condlikes_table
: If TRUE, force making of the condlikes table, and use it to calculate the log-likelihood (default=TRUE; matches LAGRANGE).

calc_ancprobs
: If TRUE (default), calculate and return the necessary pieces (uppass and downpass probs) for ancestral states.

fixnode
: If the state at a particular node is going to be fixed (e.g. for ML marginal ancestral states), give the node number.

fixlikes
: The state likelihoods to be used at the fixed node. I.e. 1 for the fixed state, and 0 for the others.

speedup
: If TRUE (default), set the maximum number of iterations to `itnmax=50*(number of free parameters)`, instead of the optimx default, 250. Also set optimx reltol parameter to 0.001 (instead of the default, ~1e-8).

Value

inputs Inputs for ML search.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

readfiles_BioGeoBEARS_run, define_BioGeoBEARS_model_object, setwd, getwd

Examples

```
test=1
```

define_tipranges_object *Define a tipranges class and object*

Description

Class tipranges is an extension of the data.frame class. It is used for holding discrete geographic range data for the tips on a phylogeny. Geographic ranges are represented with bit encoding (0/1) indicating absence or presence in each possible area.

Usage

```
define_tipranges_object(tmpdf = NULL)
```

Arguments

tmpdf	The user may input a data.frame holding the range data, if they like. Default is NULL, which means the function will produce a temporary data.frame as an example.

Details

This is just a data.frame with: rows = taxanames
columns = area names
cells = 0/1 representing empty/occupied

Value

tipranges_object The tipranges object, of class tipranges

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

getareas_from_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
testval=1
tipranges_object = define_tipranges_object()
tipranges_object
```

`dfnums_to_numeric` *Get the class for each column in a list*

Description

This function converts each column to class `numeric` where possible, and class `character` otherwise.

Usage

```
dfnums_to_numeric(dtf, max_NAs = 0.5, printout = FALSE,
  roundval = NULL)
```

Arguments

`dtf`	Input `data.frame`.
`max_NAs`	Non-numeric cells will get converted to NA, up to the fraction of cells specified by `max_NAs`. Above this fraction, the column is converted to class `character`.
`printout`	Print the results to screen, if desired.
`roundval`	If not NULL, `round` will be run using this for the number of digits.

Details

R does lots of weird and unpredictable things when you build up tables/matrices/data.frames by e.g. `cbind` and `rbind` on vectors of results. The major problems are (1) columns get made into class `list`; (2) `numeric` columns are converted to class `factor`; (3) `numeric` columns are converted to class `character`; (4) you have a `matrix` when you think you have a `data.frame`.

All of this could be taken care of by detailed understanding and tracking of when R recasts values in vectors, matrices, and data frames...but this is a huge pain, it is easier to just have a function that jams everything back to a `data.frame` with no lists, no factors, and with columns being numeric where possible. See `unlist_df4` for more, and `cls.df` to see the class of each column.

WARNING: IF A COLUMN IS A MIX OF NUMBERS AND NON-NUMBERS, THE NON-NUMBERS WILL BE CONVERTED TO NA IF THE COLUMN IS MAJORITY NUMBERS (on default; see `max_NAs`).

Value

`dtf` The output `data.frame`.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

cls.df, unlist_df4

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)

x = matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2)
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)

x = adf(matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2))
names(x) = c("A","B")
cls.df(x)
dfnums_to_numeric(adf(x))
unlist_df4(x)
```

---

divide_probs_by_number_of_options_nums

Divide each type of event by its frequency, return calculated probabilities

---

Description

In a speciation/cladogenesis matrix, the conditional probabilities of each row must sum to 1. This function sums the number of events of each category and scales them accordingly.

Usage

```
divide_probs_by_number_of_options_nums(spPmat, probmat)
```

Arguments

spPmat	A matrix of numbers, where each cell contains the conditional probability of that ancestor–>(Left descendant,Right descendant) range inheritance scenario.
probmat	A matrix of text, describing each of the allowed range-inheritance events.

Details

This function returns the calculated conditional probabilities.

Value

spPmat A matrix of numbers, where each cell contains the conditional probability of that ancestor–>(Left descendant,Right descendant) range inheritance scenario.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

make_relprob_matrix_bi, divide_probs_by_number_of_options_txt

Examples

```
testval=1
spmat = make_relprob_matrix_bi()
spmat

spmat1 = divide_probs_by_number_of_options_txt(spmat)
spmat1


probmat = spmat
spPmat = symbolic_to_relprob_matrix_sp(spmat, cellsplit="\\+",
mergesym="*", ys=1, j=0, v=1, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat
probmat2 = divide_probs_by_number_of_options_nums(spPmat, probmat)
probmat2

probmat = spmat1
spPmat = symbolic_to_relprob_matrix_sp(spmat, cellsplit="\\+",
mergesym="*", ys=1, j=0, v=1, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat
probmat3 = divide_probs_by_number_of_options_nums(spPmat, probmat)
probmat3
```

`divide_probs_by_number_of_options_txt` *Divide each type of event by its frequency*

Description

In a speciation/cladogenesis matrix, the conditional probabilities of each row must sum to 1. This function sums the number of events of each category and scales them accordingly.

Usage

```
divide_probs_by_number_of_options_txt(probmat)
```

Arguments

`probmat`	A character matrix of probabilities in the form of formulas, not normalized by the sum of each row.

Details

This function returns the strings, which can then be processed in other functions by e.g. find/replace or `eval`.

Value

`probmat` A matrix of strings, where each cell contains the parameters describing the conditional probability of that ancestor–>(Left descendant,Right descendant) range inheritance scenario.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`make_relprob_matrix_bi`, `divide_probs_by_number_of_options_nums`

Examples

```
testval=1
probmat = make_relprob_matrix_bi()
probmat

probmat2 = divide_probs_by_number_of_options_txt(probmat)
probmat2
```

`expand.grid.alt` *A faster version of expand.grid*

Description

This should be faster than `expand.grid`, which "[c]reate[s] a data frame from all combinations of the supplied vectors or factors" (R documentation).

Usage

```
expand.grid.alt(seq1, seq2)
```

Arguments

`seq1`	A sequence of elements
`seq2`	A sequence of elements

Details

The source of this function was this discussion thread: http://stackoverflow.com/questions/10405637/use-outer-instead-of-expand-grid

Value

`matrix_of_combinations` A matrix of all the possible combinations.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve, expand.grid, expand.grid.jc

Examples

```
testval=1
seq1 = c("A","B","C")
seq2 = seq1
expand.grid(seq1,seq2)
expand.grid.alt(seq1,seq2)
expand.grid.jc(seq1,seq2)
```

---

expand.grid.jc *An even faster version of expand.grid*

Description

This should be faster than expand.grid, which "[c]reate[s] a data frame from all combinations of the supplied vectors or factors" (R documentation).

Usage

```
expand.grid.jc(seq1, seq2)
```

Arguments

seq1	A sequence of elements
seq2	A sequence of elements

Details

The source of this function was this discussion thread: http://stackoverflow.com/questions/10405637/use-outer-instead-of-expand-grid

Value

matrix_of_combinations A matrix of all the possible combinations.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve, expand.grid, expand.grid.jc

Examples

```
testval=1
seq1 = c("A","B","C")
seq2 = seq1
expand.grid(seq1,seq2)
expand.grid.alt(seq1,seq2)
expand.grid.jc(seq1,seq2)
```

---

expokit_dgpadm_Qmat2 *A byte-compiled version of expokit_dgpadm_Qmat2_prebyte*

Description

Byte-compiling is supposed to speed up functions; this is an attempt to do this on the rexpokit function expokit_dgpadm_Qmat. It is also possible to byte-compile everything during package installation (via ByteCompile: true in the DESCRIPTION file), which is implemented in BioGeoBEARS, so this may be redundant.

Usage

```
expokit_dgpadm_Qmat2(times, Qmat,
  transpose_needed = TRUE)
```

Arguments

times	one or more time values to exponentiate by
Qmat	an input Q transition matrix
transpose_needed	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

Details

expokit_dgpadm_Qmat2_prebyte gets byte-compiled into expokit_dgpadm_Qmat2.

See http://dirk.eddelbuettel.com/blog/2011/04/12/ for discussion of the compile package.

Value

`tmpoutmat` the output matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

expokit_dgpadm_Qmat, expokit_dgpadm_Qmat2, compile, cmpfun

Examples

```
testval=1
```

---

expokit_dgpadm_Qmat2_prebyte

A version of expokit_dgpadm_Qmat to byte-compile

Description

Byte-compiling is supposed to speed up functions; this is an attempt to do this on the rexpokit function expokit_dgpadm_Qmat. It is also possible to byte-compile everything during package installation (via `ByteCompile: true` in the DESCRIPTION file), which is implemented in BioGeoBEARS, so this may be redundant.

Usage

```
expokit_dgpadm_Qmat2_prebyte(times, Qmat,
  transpose_needed = TRUE)
```

Arguments

`times`	one or more time values to exponentiate by
`Qmat`	an input Q transition matrix
`transpose_needed`	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

Details

expokit_dgpadm_Qmat2_prebyte gets byte-compiled into expokit_dgpadm_Qmat2.

See http://dirk.eddelbuettel.com/blog/2011/04/12/ for discussion of the compiler (cmpfun) package.

Value

tmpoutmat the output matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

expokit_dgpadm_Qmat, expokit_dgpadm_Qmat2, compile, cmpfun

Examples

```
testval=1
```

---

extend_tips_to_ultrametricize

Take a tree, extend all tips (including fossils) up to 0.0 my before present

---

Description

Makes tree precisely ultrametric by extending the terminal branches up to the highest tip (which is treated as 0 my before present).

Usage

```
extend_tips_to_ultrametricize(obj, age_of_root = 0,
  tips_end_at_this_date = NA)
```

Arguments

obj	An ape phylo object.
age_of_root	The length of the branch below the root. Default 0.
tips_end_at_this_date	The tips can be set to something other than 0, if desired. (This could produce negative branclengths, however.)

Details

This function ADDS the time_before_present to everything, including fossils. You have been warned.

Value

obj The corrected phylogeny

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

read.tree, prt, average_tr_tips

Examples

```
test=1
```

---

extract_numbers *Extract just the numbers from a string, including decimal points*

Description

This function extracts numbers from a string. Contiguous digits, including decimal points, are made into a single number. A list of numbers is returned.

Usage

```
extract_numbers(tmpstr)
```

Arguments

tmpstr	An input string.

Details

This saves you having to remember the regexp/gregexpr code for this sort of thing, and makes it much easier to parse numbers out of the text output of various programs.

Value

x2 The list of numbers

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

gregexpr

Examples

```
tmpstr = "190Ma - 65Ma"
extract_numbers(tmpstr)

tmpstr = "190.1Ma - 65.5Ma"
extract_numbers(tmpstr)
```

---

findall *Get indices of all matches to a list*

Description

Just a handy shortcut function

Usage

```
findall(what, inlist)
```

Arguments

what	The item to find
inlist	The list to search in

Value

matching_indices List of the matching indices

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_daughters, chainsaw2

Examples

```
test=1
```

---

getAIC *Calculate AIC*

Description

Calculate AIC (Akaike Information Criterion). Lower values of AIC indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters).

Usage

```
getAIC(LnL, numparams)
```

Arguments

LnL	The log-likelihood (typically negative, but may not be for continuous data).
numparams	The number of parameters for each model.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

AICval A vector of AIC results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AIC_column, calc_AIC_column

Examples

```
LnL = -34.5
numparams = 2
getAIC(LnL, numparams)

LnL = -20.9
numparams = 3
getAIC(LnL, numparams)

# It turns out to work on lists, also
LnL = c(-34.5, -20.9)
numparams = c(2, 3)
getAIC(LnL, numparams)
```

getAICc *Calculate AICc*

Description

Calculate AICc (Akaike Information Criterion). Lower values of AICc indicate some combination of better fit to the data and more parsimony in the model (fewer free parameters).

Usage

```
getAICc(LnL, numparams, samplesize)
```

Arguments

LnL	The log-likelihood (typically negative, but may not be for continuous data).
numparams	The number of parameters for each model.
samplesize	The number of data on which the model conferred likelihood.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/AICc for discussion of AICc and its uses.

Value

AICcval A vector of AICc results.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/AICc

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

calc_AICc_column, calc_AICc_column

Examples

```
LnL = -34.5
numparams = 2
samplesize = 20
getAICc(LnL, numparams, samplesize)

LnL = -20.9
numparams = 3
samplesize = 20
getAICc(LnL, numparams, samplesize)

LnL = -34.5
numparams = 2
samplesize = 5
getAICc(LnL, numparams, samplesize)

LnL = -20.9
numparams = 3
samplesize = 5
getAICc(LnL, numparams, samplesize)
```

---

getAIC_weight_for_model1

Calculate Akaike Weight

Description

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Usage

```
getAIC_weight_for_model1(AICval_1, AICvals)
```

Arguments

AICval_1	The AIC of the model of interest.
AICvals	The AICs of all the models being compared.

Value

AICweight AICweight for the models.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

AkaikeWeights_on_summary_table

Examples

```
test=1

AICval_1 = 20
AICvals = c(20,30,40)
getAIC_weight_for_model1(AICval_1, AICvals)
```

---

getareas_from_tipranges_object

Get the names of the areas in a tipranges object

Description

This function extracts the names of the areas in a tipranges object. Just a shortcut for names(tipranges@df).

Usage

```
getareas_from_tipranges_object(tipranges)
```

Arguments

tipranges	An object of class tipranges.

Value

areanames, a list of the names of the areas

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

define_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
testval=1
tipranges_object = define_tipranges_object()
tipranges_object

areanames = getareas_from_tipranges_object(tipranges_object)
areanames
```

---

getname *Collapse range abbreviations to strings*

Description

This is a utility function used by apply in tipranges_to_area_strings. It extracts the present areas and concatenates the abbreviations for one row.

Usage

```
getname(TFrow, tiparea_names, concat = TRUE, sep = "")
```

Arguments

TFrow	A list of TRUE and FALSE
tiparea_names	The names of each area
concat	If TRUE (default), merge the areas in a state into a single string.
sep	The sep argument for paste.

Value

tiparea A string.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

states_list_indexes_to_areastxt, order_tipranges_by_tree_tips, define_tipranges_object, save_tipranges_to_LagrangePHYLIP

Examples

```
getname(TFrow=c(FALSE, TRUE, TRUE, FALSE),
tiparea_names=c("K", "O", "M", "H"), sep="")
getname(TFrow=c(FALSE, TRUE, TRUE, FALSE),
tiparea_names=c("K", "O", "M", "H"), sep="_")
```

---

getranges_from_LagrangePHYLIP

Read a LAGRANGE PHYLIP-style file containing geographic ranges into a tipranges *object*

Description

Given some geographic range data for tips in the Lagrange C++/PHYLIP format (*Smith et al. (2010)*), this function imports the range data into a tipranges-class data.frame structure.

Usage

```
getranges_from_LagrangePHYLIP(lgdata_fn = "lagrange_area_data_file.data")
```

Arguments

lgdata_fn	The LAGRANGE geographic data file to be read.

Details

LAGRANGE C++ geographic range files are ASCII text files with the format:

```
19 4 (A B C D)
P_mariniana_Kokee2   1000
P_mariniana_Oahu 0100
P_mariniana_MauiNui 0010
P_hawaiiensis_Makaopuhi 0001
P_wawraeDL7428 1000
[...]
```

The first row specifies the number of taxa (here, 19), the number of areas (here, 4), and finally, the names/abbreviations of the areas. The rest of the rows give the taxon names, followed by a tab and then the presence/absence in each range with 1s/0s.

The file above is part of the geographic range data for the Hawaiian *Psychotria* dataset used by *Ree et al. (2008)*.

Value

`tipranges_object` An object of class `tipranges`

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

SmithRee2010_CPPversion

ReeSmith2008

Matzke_2012_IBS

See Also

define_tipranges_object, save_tipranges_to_LagrangePHYLIP

Examples

```
testval=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
# Set the filename (Hawaiian Psychotria from Ree & Smith 2008)
fn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))
getranges_from_LagrangePHYLIP(lgdata_fn=fn)
```

get_AICweight_ratio_model1_over_model2 *Calculate ratio of Akaike Weights*

Description

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Usage

```
get_AICweight_ratio_model1_over_model2(AICval_1,
  AICval_2)
```

Arguments

AICval_1	The AIC of the model of interest.
AICval_2	The AIC of another model of interest, for a pairwise comparison.

Value

AICweight_ratio_model1 Ratio of Akaike Weights.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

AkaikeWeights_on_summary_table

Examples

```
test=1

AICval_1 = 20
AICval_2 = 30
get_AICweight_ratio_model1_over_model2(AICval_1, AICval_2)
```

`get_Akaike_weights_from_rel_likes` *Calculate the Akaike Weights, from the relative likelihoods of the models*

Description

Given the relative likelihoods of the models, calculate the Akaike weight of the models. Akaike weights sum to 1.

Usage

```
get_Akaike_weights_from_rel_likes(rel_likes_AIC)
```

Arguments

`rel_likes_AIC`	A vector of relative likelihoods.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`Akaike_weights` A vector of Akaike Weights.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_Akaike_weights_from_rel_likes, rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
deltaAICs = get_deltaAIC(AICvals)
deltaAICs

Akaike_weights = rel_likes_from_deltaAICs(deltaAICs)
Akaike_weights
```

---

get_Akaike_weights_from_rel_likes_pairwise

Calculate the Akaike Weights, from the relative likelihoods of the models

---

Description

Given the relative likelihoods of the models, calculate the Akaike weight of the models. Akaike weights sum to 1.

Usage

```
get_Akaike_weights_from_rel_likes_pairwise(rel_likes_AIC_pairwise)
```

Arguments

`rel_likes_AIC_pairwise`
: A 2-column `data.frame` of relative likelihoods of each pair of models.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`Akaike_weights_pairwise` A `data.frame` of Akaike Weights for each row (column 1) and the reference model (column 2). Note that only 2 models are being compared in each row, not all of them, as in get_Akaike_weights_from_rel_likes.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_Akaike_weights_from_rel_likes, rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
deltaAICs = get_deltaAIC_pairwise_w_ref_model(AICvals, ref_model="best")
deltaAICs

rel_likes_AIC_pairwise = rel_likes_from_deltaAICs_pairwise(deltaAICs)
rel_likes_AIC_pairwise

Akaike_weights_pairwise = get_Akaike_weights_from_rel_likes_pairwise(rel_likes_AIC_pairwise)
Akaike_weights_pairwise
```

---

get_Akaike_weight_ratio_from_Akaike_pairwise_weights

Get the ratio between the pairwise Akaike Weights

Description

Given the relative likelihoods of the models, calculate the Akaike weight of the models. Akaike weights sum to 1.

Usage

```
get_Akaike_weight_ratio_from_Akaike_pairwise_weights(Akaike_weights_pairwise)
```

Arguments

Akaike_weights_pairwise	A 2-column data.frame of Akaike Weights for each pair of models.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`Akaike_weight_ratios_pairwise` A `data.frame` of Akaike Weight Ratios for each row (column 1) and the reference model (column 2). Note that only 2 models are being compared in each row, not all of them.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

`get_Akaike_weights_from_rel_likes_pairwise`, `get_Akaike_weights_from_rel_likes`, `rel_likes_from_deltaAI`
`getAIC`

Examples

```
test=1

AICvals = c(40, 50, 60)
deltaAICs = get_deltaAIC_pairwise_w_ref_model(AICvals, ref_model="best")
deltaAICs

rel_likes_AIC_pairwise = rel_likes_from_deltaAICs_pairwise(deltaAICs)
rel_likes_AIC_pairwise

Akaike_weights_pairwise = get_Akaike_weights_from_rel_likes_pairwise(
rel_likes_AIC_pairwise)
Akaike_weights_pairwise

Akaike_weight_ratios_pairwise = get_Akaike_weight_ratio_from_Akaike_pairwise_weights(
Akaike_weights_pairwise)
Akaike_weight_ratios_pairwise
```

get_all_daughter_tips_of_a_node *Get all the daughter tips of a node*

Description

Like it says. Utility function.

Usage

```
get_all_daughter_tips_of_a_node(nodenum, t)
```

Arguments

nodenum	The node to find
t	A phylo tree object.

Value

temp_tips The list of daughter tipnodes

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

add_to_downpass_labels, extract.clade

Examples

```
test=1
```

`get_all_node_ages` *Get the ages of all the nodes in the tree (above the root)*

Description

A utility function. Use of `dist.nodes` may be slow.

Usage

```
get_all_node_ages(obj)
```

Arguments

`obj`	An ape phylo object

Value

`TF_tips` The age (from the root) of each node.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`prt`, `chainsaw2`

Examples

```
test=1
```

get_APE_nodenums *Get R internal node numbers*

Description

Utility function

Usage

```
get_APE_nodenums(tr)
```

Arguments

tr	A phylo tree object

Value

nodenums A list of node numbers

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_lagrange_nodenums, prt

Examples

```
test=1
```

get_colors_for_numareas *Get colors for a certain number of single areas*

Description

Like it says.

Usage

```
get_colors_for_numareas(numareas, use_rainbow = FALSE)
```

Arguments

numareas	The number of areas
use_rainbow	If TRUE, force use of rainbow()

Value

colors_matrix The colors for the single areas, 1 column per area

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

optim

Examples

```
testval=1
```

get_daughters *Get all the direct daughters nodes of a node*

Description

Get all the direct daughters nodes of a node

Usage

```
get_daughters(nodenum, t)
```

Arguments

nodenum	The node number to get the daughters of
t	An ape phylo object

Value

daughter_nodenums List of the daughter node numbers

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

findall, chainsaw2

Examples

```
test=1
```

get_deltaAIC *Calculate deltaAIC*

Description

Calculate deltaAIC (Akaike Information Criterion), the absolute difference between the best model (lowest AIC) and other models.

Usage

```
get_deltaAIC(AICvals)
```

Arguments

AICvals	A vector of AIC values.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

deltaAICs A vector of deltaAICs.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
get_deltaAIC(AICvals)
```

`get_deltaAIC_pairwise_w_ref_model` *Calculate deltaAIC*

Description

Calculate deltaAIC (Akaike Information Criterion), the absolute difference between the best model (lowest AIC) and other models. This function does it pairwise only, with a reference model.

Usage

```
get_deltaAIC_pairwise_w_ref_model(AICvals,
  ref_model = "best")
```

Arguments

`AICvals`	A vector of AIC values.
`ref_model`	What is the row of the reference model? "best", "worst", or a row number.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`deltaAICs_pairwise` A 2-column `data.frame` of pairwise deltaAICs for each row (column 1) and the reference model (column 2).

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_deltaAIC, rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
get_deltaAIC(AICvals)
get_deltaAIC_pairwise_w_ref_model(AICvals, ref_model="best")
```

---

get_edge_times_before_present

Get the times of the top and bottom of each edge

Description

A utility function.

Usage

```
get_edge_times_before_present(t)
```

Arguments

t	An ape phylo object

Value

edge_times_bp A 2-column matrix with the age (from the present) of the top and bottom of each edge.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2

Examples

```
test=1
```

`get_fn_prefix` *Get everything BEFORE the last suffix (.nex or whatever)*

Description

Extracts the string from before the last suffix. I.e., "filename.nex" becomes "filename".

Usage

```
get_fn_prefix(fn)
```

Arguments

`fn`	The input filename.

Value

`prefix` The output string.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`get_path_last`, `get_path_first`

Examples

```
get_fn_prefix("/Users/nickm/Library/R/Psychotria_geog.data")
get_fn_prefix("Psychotria_geog.data")
```

`get_indices_of_branches_under_tips` *Get the indices of the branches (row number in edge matrix) below each tip*

Description

A utility function. Gets the indices of the branches (row number in edge matrix) below each tip.

Usage

```
get_indices_of_branches_under_tips(obj)
```

Arguments

`obj`	An ape phylo object

Value

`branchnums_under_tips` The indices of the branches (row number in edge matrix) below each tip.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2, get_indices_of_tip_nodes, get_indices_where_list1_occurs_in_list2_noNA

Examples

```
test=1
```

get_indices_of_tip_nodes *Get TRUE/FALSE for nodes being tips*

Description

A utility function that returns indices (node numbers) of the tips. This mostly saves typing.

Usage

```
get_indices_of_tip_nodes(obj)
```

Arguments

obj	An ape phylo object

Value

tip_indices The node numbers of the tips.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2, phylo, get_indices_of_branches_under_tips

Examples

```
test=1
```

`get_indices_where_list1_occurs_in_list2` *Return (first!) indices in second list matching the first list*

Description

This function will return one match (the first) for each item in the list; i.e. the second-list index for each item in the first list. Only the first hit in the second list is returned.

Usage

```
get_indices_where_list1_occurs_in_list2(list1, list2)
```

Arguments

`list1`	The first list.
`list2`	The second list list.

Details

This is used by `prt`.

Value

`match_indices` The match indices.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`prt`, `LETTERS`, `get_indices_where_list1_occurs_in_list2_noNA`

Examples

```
list1 = c("N", "I", "C", "K")
list2 = LETTERS
get_indices_where_list1_occurs_in_list2(list1, list2)
```

get_indices_where_list1_occurs_in_list2_noNA *Return (first!) indices in second list matching the first list, excluding NAs*

Description

This function will return one match (the first) for each item in the list; i.e. the second-list index for each item in the first list. Only the first hit in the second list is returned. Unlike get_indices_where_list1_occurs_in_list non-hits (NAs) are excluded.

Usage

```
get_indices_where_list1_occurs_in_list2_noNA(list1,
  list2)
```

Arguments

list1	The first list.
list2	The second list list.

Details

This is used by get_indices_of_branches_under_tips, which is used by extend_tips_to_ultrametricize, which can be used by section_the_tree.

Value

match_indices The match indices.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, LETTERS, get_indices_where_list1_occurs_in_list2, extend_tips_to_ultrametricize, section_the_tree, return_items_not_NA

Examples

```
list1 = c("N", "I", "C", "K")
list2 = LETTERS
get_indices_where_list1_occurs_in_list2_noNA(list1, list2)
```

`get_infparams_optimx` *Get the inferred parameters from an ML optimization*

Description

This function extracts the ML parameter values, and associated statistics and codes, from the `relprobs_matrix` returned by `bears_2param_standard_fast` and similar functions.

Usage

```
get_infparams_optimx(results_object, inffn)
```

Arguments

`results_object`	The results returned by `bears_2param_standard_fast` or a similar function.
`inffn`	The filename holding the results_object, which specifies which model was run.

Details

The function has subroutines for recognizing a variety of currently-implemented models, assuming they used `optimx` internally to do the ML search. New models would require addition of new subroutines.

`get_infparams_optimx` and `get_infparams_optimx_nosim` differ only in the format of the filenames.

Value

`infparams` The vector of inferred parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_infparams_optimx_nosim, bears_2param_standard_fast, get_inf_LgL_etc_optimx

Examples

```
testval=1
```

---

get_infparams_optimx_nosim *Get the inferred parameters from an ML optimization (different filenames)*

Description

Like get_infparams_optimx, this function extracts the ML parameter values, and associated statistics and codes, from the results_object returned by bears_2param_standard_fast and similar functions.

Usage

```
get_infparams_optimx_nosim(results_object, inffn)
```

Arguments

results_object	The results returned by bears_2param_standard_fast or a similar function.
inffn	The filename holding the results_object, which specifies which model was run.

Details

The function has subroutines for recognizing a variety of currently-implemented models, assuming they used optimx internally to do the ML search. New models would require addition of new subroutines.

get_infparams_optimx and get_infparams_optimx_nosim differ only in the format of the filenames.

Value

infparams The vector of inferred parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_infparams_optimx, bears_2param_standard_fast, get_inf_LgL_etc_optimx

Examples

```
testval=1
```

---

get_infprobs_of_simstates

Get the probabilities of the true (simulated) states

Description

Basically this function assigns probability 1 to the simulated state/geographic range, and probability 0 for the other states/geographic ranges. These data – the simulated truth – can then be compared to the inferred probabilities for the states, from e.g. get_ML_probs.

Usage

```
get_infprobs_of_simstates(relprobs_matrix, simhist_row)
```

Arguments

relprobs_matrix	A relative probabilities matrix returned by bears_2param_standard_fast or a similar function. The user should specify WHICH matrix in the results_object – i.e., scaled conditional likelihoods on downpass or uppass, or actual marginal probabilities of ancestral states. (The latter is the main thing of interest.) This specification is done via e.g. relprobs_matrix = results_object$relative_probs_of_each_stat
simhist_row	A row from a table, which must have a column named simulated_states_by_node_txt.

Value

infprobs_of_simstates The probability of each state at each node (all 1s and 0s).

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

simulate_biogeog_history, infprobs_to_probs_of_each_area

Examples

```
testval=1
```

---

get_inf_LgL_etc_optimx *Get the inferred parameters from a results object (utility function)*

Description

This function extracts the ML parameter values from the `results_object` returned by `bears_2param_standard_fast` and similar functions.

Usage

```
get_inf_LgL_etc_optimx(results_object)
```

Arguments

`results_object`	The results returned by `bears_2param_standard_fast` or a similar function.

Details

This is primarily a utility function for `get_infparams_optimx`.

Value

`infparams` The vector of inferred parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

bears_2param_standard_fast, get_infparams_optimx

Examples

```
testval=1
```

---

get_lagrange_nodenums *Get internal node numbers in LAGRANGE's downpass order*

Description

There are many ways of numbering nodes in a tree. This returns a matrix containing (column 1) R's native internal numbering scheme, and (column 2) the node numbers in the downpass numbering used by C++ LAGRANGE, in particular in their .bgkey output file. Note that this is different from ape's pruningwise downpass ordering (see get_pruningwise_nodenums).

Usage

```
get_lagrange_nodenums(tr)
```

Arguments

tr	A phylo tree object

Details

The python version of LAGRANGE labels internal nodes differently (sigh), but they are in the same order at least, so can just be renumbered from 1 to tr$Nnode to get them to match the C++ LAGRANGE node numbering.

DIVA has yet a different node numbering scheme; see postorder_nodes_phylo4_return_table

Value

downpass_node_matrix A matrix of node numbers

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_pruningwise_nodenums, prt, postorder_nodes_phylo4_return_table

Examples

```
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
tmppath = paste(extdata_dir,
"/examples/Psychotria_M0/LGcpp/Psychotria_5.2.newick", sep="")
trfn = np(slashslash(tmppath))
tr = read.tree(trfn)
downpass_node_matrix = get_lagrange_nodenums(tr)
downpass_node_matrix


downpass_node_matrix = get_lagrange_nodenums(tr)
downpass_node_matrix = downpass_node_matrix[order(downpass_node_matrix[,2]), ]
plot(tr)
nodelabels(node=20:37, downpass_node_matrix[,1])
tiplabels(1:19)

plot(tr)
nodelabels(node=20:37, downpass_node_matrix[,2])
tiplabels(1:19)

downpass_node_matrix = get_lagrange_nodenums(tr)
downpass_node_matrix = downpass_node_matrix[order(downpass_node_matrix[,1]), ]
plot(tr)
nodelabels(node=20:37, downpass_node_matrix[,1])
tiplabels(1:19)

# THIS WORKS
plot(tr)
nodelabels(node=20:37, downpass_node_matrix[,2])
tiplabels(1:19)
```

---

get_leftright_nodes_matrix_from_results

Make a table of the Right and Left nodes descending from each node

---

Description

This table shows the Right, then Left, descendant nodenums for each node. This gets used later to plot splits at corners.

Usage

```
get_leftright_nodes_matrix_from_results(tr,
  results_object, nodes)
```

Arguments

tr	An ape phylo object
results_object	The results from a BioGeoBEARS ML search.
nodes	A list of internal node numbers for tree tr.

Value

leftright_nodes_matrix A table with the Right, the Left, nodes

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

get_level *Get a node's level in the tree*

Description

Finds how many nodes deep a node is.

Usage

```
get_level(nodenum, t, tmplevel = 0)
```

Arguments

nodenum	The node number to get the parent of
t	An ape phylo object
tmplevel	A starting level (the function is recursive)

Value

tmplevel The level of the node.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2

Examples

```
test=1
```

`get_max_height_tree` *Get the maximum age of all the nodes (above the root)*

Description

I.e., the distance of the highest node above the root. A utility function. Use of `dist.nodes` may be slow.

Usage

```
get_max_height_tree(obj)
```

Arguments

`obj`	An ape phylo object

Value

`max_height` The age (from the root) of the highest node.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`prt`, `chainsaw2`

Examples

```
test=1
```

get_MLsplitprobs_from_results *Extract the ML probs for the base of each branch above a split*

Description

This function takes a BioGeoBEARS results_object from a ML search, extracts the downpass and uppass likelihoods of the data for each possible state at the base of each left and right branch, and produces the ML ancestral split estimates for the bottom of each branch.

Usage

```
get_MLsplitprobs_from_results(results_object)
```

Arguments

results_object	The results from a BioGeoBEARS ML search.

Value

results_object with results_object$ML_marginal_prob_each_split_at_branch_bottom_BELOW_node added

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

get_ML_probs *Get the probability of the ML state for each node, from a BioGeoBEARS model results list*

Description

This function extracts the probability of the ML states from the results list produced by bears_2param_standard_fast or a similar ML search function.

Usage

```
get_ML_probs(relprobs_matrix, unlist_TF = TRUE)
```

Arguments

relprobs_matrix	A relative probabilities matrix returned by bears_2param_standard_fast or a similar function. The user should specify WHICH matrix in the results_object – i.e., scaled conditional likelihoods on downpass or uppass, or actual marginal probabilities of ancestral states. (The latter is the main thing of interest.) This specification is done via e.g. relprobs_matrix = results_object$relative_probs_of_each_stat
unlist_TF	Unlist the output? Default TRUE.

Details

This is useful for displaying e.g. pie charts of the probability of the ML ancestral state at each node.

Note, though, that it is somewhat peculiar and arbitrary to focus on the ancestral states just at nodes, particularly in the context of fossils with time ranges and geographic ranges.

Value

inf_probsvec The inferred vector of probabilities of ML states.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://blog.phytools.org/2013/03/marginal-ancestral-state-reconstruction.html https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

get_ML_probs, bears_2param_standard_fast, get_ML_state_indices

Examples

```
testval=1
```

---

get_ML_states *Get ML states from a BioGeoBEARS model results list*

Description

This function extracts the ML states from the results list produced by bears_2param_standard_fast or a similar ML search function.

Usage

```
get_ML_states(relprobs_matrix, unlist_TF = TRUE)
```

Arguments

relprobs_matrix	A relative probabilities matrix returned by bears_2param_standard_fast or a similar function. The user should specify WHICH matrix in the results_object – i.e., scaled conditional likelihoods on downpass or uppass, or actual marginal probabilities of ancestral states. (The latter is the main thing of interest.) This specification is done via e.g. relprobs_matrix = results_object$relative_probs_of_each_stat
unlist_TF	Unlist the output? Default TRUE.

Details

Currently, the scaled conditional probabilities are used to determine the optimum states. However, this is not strictly correct, as these use only tips-down information (*Felsenstein (2004)*; see also this post by Revell: http://blog.phytools.org/2013/03/marginal-ancestral-state-reconstruction.html). This is what LAGRANGE seems to do when reporting ancestral states, also (personal observation, perhaps imperfect, especially if the scaled conditional likelihoods and the marginal ancestral state probabilities turn out to be very close). What is desired is the marginal ancestral state reconstructions. Most authors discuss ML ancestral state reconstruction as being a matter of re-rooting the tree at each node, yielding the marginal estimate for that node, conditional on the rest of the tree. However, this procedure assumes a time-reversible model on both branches and cladogenesis events, and we have neither in biogeography. Probably, the solution is just an up-pass from the root, calculating the probabilities on the forward model and multiplying by likelihoods from the downpass. However, this has not yet been implemented.

Value

inf_statesvec The inferred vector of states.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://blog.phytools.org/2013/03/marginal-ancestral-state-reconstruction.html https://code.google.com/p/lagrange/

Felsenstein2004

Matzke_2012_IBS

See Also

get_ML_probs, bears_2param_standard_fast, get_ML_state_indices

Examples

```
testval=1
```

---

get_ML_states_from_relprobs

Extract the ML states at each node, from a table of relative probabilities – old version

---

Description

Given a table with the rows representing nodes, and the columns representing the relative probabilities of each state, this function finds the ML (maximum likelihood) state(s) for each node.

Usage

```
get_ML_states_from_relprobs(relprobs, statenames,
  returnwhat = "states", if_ties = "takefirst")
```

Arguments

relprobs	A numeric matrix of relative probabilities
statenames	The names of the states/geographic ranges (e.g., A, AB, CDE, ABD, etc...)
returnwhat	If "indices", return the 0-based indices of the states. If "states", return the name of the state, based on statenames.
if_ties	What to do with ties. Currently, the only option is to take the first (this will be shown in e.g. a pie chart, of course).

Details

If possible, the input matrix should be the actual ML estimate of the state probabilities at each node, rather than just the scaled conditional likelihoods at each node. The latter reflect only the tips-down information, whereas the former (the marginal ancestral state reconstruction) uses all of the information, and the probabilities of the states at the root and in the outgroup(s) can influence the estimates in the ingroups. This would not likely be particularly important in a pure continuous-time model, but in a model with cladogenesis it could matter quite a bit.

See http://blog.phytools.org/2013/03/marginal-ancestral-state-reconstruction.html for more discussion of marginal ancestral state reconstructions, versus mere scaled conditional likelihoods.

Revell and other sources (*Felsenstein (2004)*) advocate the "re-rooting" method for obtaining the marginal ancestral state reconstructions; however, re-rooting requires a time-reversible model and a tree with no root. In biogeography we have a *non*-reversible model, and typically a time-scaled chronogram. However, the same result can be obtained by modifying the scaled conditional likelihoods obtained from a downpass from the tips, via an doing an up-pass from the root scaled conditional likelihoods, being careful to transfer probabilities via the time-forward version of the Q-matrix and cladogenesis/speciation matrix.

Note: further notes as this is implemented (required!)

Value

`ML_states` or `ML_states_indices`, depending on `returnwhat`.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://blog.phytools.org/2013/03/marginal-ancestral-state-reconstruction.html https://code.google.com/p/lagrange/

Felsenstein2004

Matzke_2012_IBS

See Also

get_ML_state_indices

Examples

```
testval=1
```

get_ML_state_indices *Extract the indices for the ML states at each node, given a row of relative probabilities*

Description

Given a table with the rows representing nodes, and the columns representing the relative probabilities of each state, this function finds the ML (maximum likelihood) state(s) for each node; get_ML_state_indices does this for a row, get_ML_states iterates over all the rows.

Usage

```
get_ML_state_indices(relprobs_row, nums, maxprob,
  if_ties = "takefirst")
```

Arguments

relprobs_row	A row from a relprobs, a numeric matrix of relative probabilities
nums	Numbers indexing the states from 1 to numstates
maxprob	The value of the maximum probability for the row.
if_ties	What to do with ties. Currently, the only option is to take the first (this will be shown in e.g. a pie chart, of course).

Value

index_of_ML_state_s

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

get_ML_states

Examples

testval=1

get_nodenums *Get the unique node numbers in a tree*

Description

This is a utility function for get_nodenum_structural_root.

Usage

```
get_nodenums(t)
```

Arguments

t	A tree object in phylo format.

Value

ordered_nodenames The node numbers, in order.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo, get_nodenum_structural_root

Examples

```
blah = 1
```

get_nodenum_structural_root *Gets the root node*

Description

This function gets the root node by finding the node not in the descendants list (edge[,2]). This may be more reliable than e.g. assuming length(tr$tip.label)+1.

Usage

```
get_nodenum_structural_root(t, print_nodenum = FALSE)
```

Arguments

t	A tree object in phylo format.
print_nodenum	Print the node numbers as you go through the list? Default FALSE.

Value

root_nodenums_list

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo, get_nodenums

Examples

```
blah=1
```

get_node_ages_of_tips *Get the ages of each tip above the root*

Description

A utility function.

Usage

```
get_node_ages_of_tips(obj)
```

Arguments

obj	An ape phylo object

Value

TF_tips The age (from the root) of each tip.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2

Examples

```
test=1
```

get_parent *Get the direct parent node of a node*

Description

Get the direct parent node of a node

Usage

```
get_parent(nodenum, t)
```

Arguments

nodenum	The node number to get the parent of
t	An ape phylo object

Value

parent_nodenumThe parent node number

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

findall, chainsaw2

Examples

```
test=1
```

get_path_first *Get the text that comes before the last slash*

Description

Extracts the path from a full path, removing the filename.

Usage

```
get_path_first(inpath, addslash = "FALSE")
```

Arguments

inpath	A string of class character.
addslash	If TRUE, add a slash at the end of the path.

Value

outpath A string with the full path, without the file.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_path_last

Examples

```
get_path_first("/Users/nickm/Library/Psychotria_geog.data")
```

get_path_last *Get the text that comes after the last slash*

Description

Extracts the filename from a full path.

Usage

```
get_path_last(path)
```

Arguments

path	A string of class character.

Value

lastword A string with the filename, without the path.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

FosterIdiots

See Also

get_path_first

Examples

```
get_path_last("/Users/nickm/Psychotria_geog.data")
```

`get_perEvent_probs` *Get the per-event probabilities at cladogenesis*

Description

At a cladogenesis event, a large number of events are possible. The simplest way to compute these is just to assign some weight to each event, then sum all the events and divide by the sum to get the probabilities. More complex schemes can be imagined, but these are fairly pointless as they would all break down once e.g. distance-dependence, user-specified connectivities, etc., are imposed.

Usage

```
get_perEvent_probs(params_table, sumval = 1,
  plotwhat = "est")
```

Arguments

`params_table`	The `params_table` from a `BioGeoBEARS_model_object`.
`sumval`	Default=1.
`plotwhat`	Default "est", use "init" to get the initial starting values instead.

Details

In addition, one could imagine trying to assign total probabilities to each category of event, but each row of the cladogenesis matrix may have a different count of the different types of events (one row may have 1 y event and 2 j events; another row may have 4 j, 2 v, and 2 s, and 0 y events; etc.).

One thing that IS meaningful is the per-event weight, i.e. the values that the program is using for j, v, y, and s. These ARE meaningful, as long as they are forced to sum to some value (default 4). This ensures that they are identifiable (otherwise, j,v,y,s=1 and j,v,y,s=2 would be the same model).

This function calculates the per-event weight as a proportion of some total weight, e.g. default 1. If the optim result was j=0, s=1, y=1, v=1, the `get_perEventprobs()` result would be 0, 0.333, 0.333, 0.333.

Value

`wts` Return the per-event weights

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

rbind

Examples

```
# default DEC+J model
BioGeoBEARS_run_object = define_BioGeoBEARS_run()
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table
params_table = BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table
params_table

get_perEvent_probs(params_table)


# DEC+J model
BioGeoBEARS_run_object = define_BioGeoBEARS_run()
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","type"] = "free"
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","init"] = 1
BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table["j","est"] = 1

BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table

BioGeoBEARS_run_object$BioGeoBEARS_model_object =
calc_linked_params_BioGeoBEARS_model_object(
BioGeoBEARS_model_object=BioGeoBEARS_run_object$BioGeoBEARS_model_object,
update_init=TRUE)


BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table
params_table = BioGeoBEARS_run_object$BioGeoBEARS_model_object@params_table

get_perEvent_probs(params_table)
```

---

get_probvals *Calculate probability of ordered discrete states using a maxent distribution (equations 6.3-6.4 of Harte 2011)*

Description

This function is calculates the Maximum Entropy (*Harte (2011)*) discrete probability distribution of a number of ordered states (e.g., faces of a 6-sided die) given the mean of many rolls. Here, this is merely used so that a single parameter can control the probability distribution of small versus large descendant areas during cladogenesis. This function could then used by relative_probabilities_of_subsets in BioGeoBEARS to weight different descendant range sizes (although, currently, the function maxent from the FD package is used).

Usage

```
get_probvals(die_vals, meanval)
```

Arguments

die_vals	Values of the ordered discrete variable state (e.g., seq(1,6) for a six-sided die)
meanval	Mean value (the knowledge supplied to the MaxEnt function).

Details

This calculation is based on Equations 6.3-6.4 of *Harte (2011)*.

See also: Maximum Entropy probability distribution for discrete variable with given mean (and discrete uniform flat prior) http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Value

Prob_nvals, numeric values of the probability of each state from die_vals.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Harte2011

Matzke_2012_IBS

See Also

calcZ_part, calcP_n, maxent, symbolic_to_relprob_matrix_sp, relative_probabilities_of_subsets

Examples

```
testval=1
# Examples
# Set up subplots
par(mfrow=c(3,2))

# Flat distribution (equal prob of any descendent size)
N = 6
# n = die_vals
die_vals = seq(1,N)
```

```
meanval = 3.5
probvals = get_probvals(die_vals, meanval)
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("Probabilities of each state, mean val=", meanval, sep=""))

# Descendents tend to have large ranges
N = 6
# n = die_vals
die_vals = seq(1,N)
meanval = 5.999
probvals = get_probvals(die_vals, meanval)
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("Probabilities of each state, mean val=", meanval, sep=""))

# Flat distribution (equal prob of any descendent size)
N = 6
# n = die_vals
die_vals = seq(1,N)
meanval = 5
probvals = get_probvals(die_vals, meanval)
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("Probabilities of each state, mean val=", meanval, sep=""))

# Flat distribution (equal prob of any descendent size)
N = 6
# n = die_vals
die_vals = seq(1,N)
meanval = 4
probvals = get_probvals(die_vals, meanval)
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("Probabilities of each state, mean val=", meanval, sep=""))

# Flat distribution (equal prob of any descendent size)
N = 6
# n = die_vals
die_vals = seq(1,N)
meanval = 2
probvals = get_probvals(die_vals, meanval)
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("Probabilities of each state, mean val=", meanval, sep=""))

# This produces the LAGRANGE default
# (all smaller descendents are of size 1)
N = 6
# n = die_vals
die_vals = seq(1,N)
meanval = 1.0001
probvals = get_probvals(die_vals, meanval)
```

```
probvals
barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
title(paste("LAGRANGE 'default', mean val=", meanval, sep=""))

# This is stopped by the error check
# (all smaller descendents are of size 1)
# N = 6
# # n = die_vals
# die_vals = seq(1,N)
# meanval = 0.5
# probvals = get_probvals(die_vals, meanval)
# probvals
# barplot(height=probvals, width=1, names.arg=die_vals, ylim=c(0,1))
# title(paste("Probabilities of each state, mean val=", meanval, sep=""))
```

`get_pruningwise_nodenums` *Get internal node numbers in pruningwise order*

Description

There are many ways of numbering nodes in a tree. This returns a matrix containing (column 1) R's native internal numbering scheme, and (column 2) the node numbers in a pruningwise downpass. Note that this is different from LAGRANGE's downpass ordering (see `get_lagrange_nodenums`).

Usage

```
get_pruningwise_nodenums(tr)
```

Arguments

`tr`	A `phylo` tree object

Value

`node_numbers_matrix` A matrix of node numbers

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_lagrange_nodenums, prt

Examples

```
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
tmpdir = paste(extdata_dir,
"/examples/Psychotria_M0/LGcpp/Psychotria_5.2.newick", sep="")
trfn = np(slashslash(tmpdir))
tr = read.tree(trfn)
node_numbers_matrix = get_pruningwise_nodenums(tr)
node_numbers_matrix
```

---

get_relative_prob_model1old

Calculate relative probability of model 1 (=Akaike Weight)

Description

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Usage

```
get_relative_prob_model1old(AICval_1, AICval_2)
```

Arguments

AICval_1	The AIC of the model of interest.
AICval_2	The AIC of another model of interest, for a pairwise comparison.

Value

relative_prob_model1 Akaike Weight of model 1.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

AkaikeWeights_on_summary_table

Examples

```
test=1

AICval_1 = 20
AICval_2 = 30
get_relative_prob_model1old(AICval_1, AICval_2)
```

`get_relative_prob_model2old` *Calculate relative probability of model 1 (Akaike Weight)*

Description

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Usage

```
get_relative_prob_model2old(AICval_1, AICval_2)
```

Arguments

`AICval_1`	The AIC of the model of interest.
`AICval_2`	The AIC of another model of interest, for a pairwise comparison.

Details

This is an older version of `get_relative_prob_model1old`, kept for back-compatibility.

Value

`relative_prob_model1` Akaike Weight of model 1.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

AkaikeWeights_on_summary_table, get_relative_prob_model1old

Examples

```
test=1

AICval_1 = 20
AICval_2 = 30
get_relative_prob_model1old(AICval_1, AICval_2)
```

---

get_rownum_ref_model *Get rownum of named model*

Description

Find the row number of the best model according to AIC, the worst model according to AIC, or just takes the row number if that is what was input.

Usage

```
get_rownum_ref_model(AICvals, ref_model = "best")
```

Arguments

AICvals	A vector of AIC values.
ref_model	What is the row of the reference model? "best", "worst", or a row number.

Value

ref_model_num The

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

FosterIdiots

See Also

convolve

Examples

```
test=1
```

---

get_simparams *Get the simulated model parameters from the row of a table*

Description

Basically this function assigns probability 1 to the simulated state/geographic range, and probability 0 for the other states/geographic ranges. These data – the simulated truth – can then be compared to the inferred probabilities for the states, from e.g. get_ML_probs.

Usage

```
get_simparams(simhist_row)
```

Arguments

simhist_row	A row from a table, which must have a column named simulated_states_by_node_txt.

Value

simparams A list of the parameter values.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

simulate_biogeog_history, infprobs_to_probs_of_each_area

Examples

```
testval=1
```

---

get_simstates *Load the simulation information from an underscore delimited text string.*

Description

If the simulated states are stored in a big text file, it can be useful to store them as a single string in a single cell per row, so that the number of columns doesn't have to change with each different-sized tree. This function extracts the simulated states from this format.

Usage

```
get_simstates(simhist_row)
```

Arguments

simhist_row	A row from a table, which must have a column named simulated_states_by_node_txt.

Value

simulated_states_by_node A numeric vector of 0-based state indices.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

read.table

Examples

```
testval=1
```

`get_sister_node` *Get the node sister to two nodes*

Description

Input two sister nodes, returns their "aunt". Assumes a binary tree.

Usage

```
get_sister_node(tr, nodepair)
```

Arguments

`tr`	A phylo tree object.
`nodepair`	A vector (length 2) with the node numbers of two nodes/tips.

Value

`moms_sister` The aunt node.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

add_to_downpass_labels

Examples

```
test=1
```

get_statesColors_table *Make a color table for each area and their combinations*

Description

Given a list of areas, make a color table for the various combinations.

Usage

```
get_statesColors_table(areanames = c("K", "O", "M", "H"))
```

Arguments

areanames	A list of the area names.

Value

statesColors_table A table giving the colors for each state.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

get_TF_tips *Get TRUE/FALSE for nodes being tips*

Description

A utility function that returns TRUE/FALSE for whether or not each node is a tip.

Usage

```
get_TF_tips(obj)
```

Arguments

obj	An ape phylo object

Value

TF_tips The TRUE/FALSE list for each tip.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2, match_list1_in_list2

Examples

```
test=1
```

get_tiplabel_ranges *For each tip, get a text string of the areas in a tipranges object.*

Description

This function extracts the names of the areas in a tipranges object. Just a shortcut for names(tipranges@df).

Usage

```
get_tiplabel_ranges(tipranges, tr, sep = "")
```

Arguments

tipranges	An object of class tipranges.
tr	An ape phylo object.
sep	Input to paste.

Value

areanames, a list of the names of the areas

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

define_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
testval=1
tipranges_object = define_tipranges_object()
tipranges_object

areanames = getareas_from_tipranges_object(tipranges_object)
areanames
```

given_a_starting_state_simulate_branch_end
Given the state at the start of a branch, simulate the state at the end of the branch

Description

This function simulates a biogeographical history, given a Q transition matrix, a starting state, and a branch length. All this involves is exponentiating the Q transition matrix, producing a P transition probability matrix, and then producing a random draw from this P matrix, conditional on the ancestor.

Usage

```
given_a_starting_state_simulate_branch_end(index_Qmat_0based_of_starting_state = 1,
  Qmat, branchlength = 1, all_tips_living = TRUE)
```

Arguments

index_Qmat_0based_of_starting_state	An integer index value, between 0 and (numstates-1), which specifies what state is the starting point for the branch.
Qmat	A (square, dense) Q transition matrix. Using a sparse matrix would require writing another function.
branchlength	The length of the branch, or branch segment if you are dealing with a stratified phylogeny.
all_tips_living	Currently this is the only assumption. If, hypothetically, you had a phylogeny with extinct tips (representing the ends of the ranges of fossil taxa), you might want to treat them differently, IF you think that the time-invariant geographic range addition/subtraction process is the same one that made lineages go extinct (it could be something else, e.g. mass extinction). False attribution of extinctions to the range loss process will dramatically elevate the rate of range loss, and also range expansion to compensate, and the resulting high rates can substantially degrade inference (*Matzke_Maguire_2011_SVP*).

Details

This could be sped up in various ways, if needed.

Value

state_desc 0-based index of the descendant state (just before cladogenesis, if below a node).

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

Matzke_Maguire_2011_SVP

See Also

rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

---

given_a_starting_state_simulate_split

Given the state just below a node, simulate the states after speciation

---

Description

This function simulates a biogeographical history during a speciation/cladogenesis range inheritance event, given a cladogenesis probability transition matrix and a starting state.

Usage

```
given_a_starting_state_simulate_split(index_Qmat_0based_of_starting_state = 1,
  COO_probs_columnar, numstates)
```

Arguments

index_Qmat_0based_of_starting_state	An integer index value, between 0 and (numstates-1), which specifies what state is the starting point for the branch.
COO_probs_columnar	A speciation/cladogenesis transition matrix, in COO-like form, as produced by rcpp_calc_anclikes_sp_COOweights_faster.
numstates	The number of states/geographic ranges.

Value

split_desc 0-based indices of the descendant states in the two daughters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

Matzke_Maguire_2011_SVP

See Also

rcpp_calc_anclikes_sp_COOweights_faster, rcpp_calc_rowsums_for_COOweights_columnar

Examples

```
testval=1
```

---

infprobs_to_probs_of_each_area

Convert probabilities of each state, to the probabilities of presence in each area

---

Description

Biogeographic inference in LAGRANGE and DIVA has focused heavily on inference of the exact ancestral state/geographic range. However, when the state space is large, there is often considerable uncertainty in the exact ancestral range. Even the ancestral state that confers the maximum likelihood on the data, and thus is the most probable ancestor, may have less than 50 probability, or even less (25 size of the state space. This function converts the probability of specific states/geographic ranges into the probability of presence/absence in each area. This can typically be inferred with much higher confidence.

Usage

```
infprobs_to_probs_of_each_area(relprobs_matrix,
  states_list)
```

Arguments

relprobs_matrix	A relative probabilities matrix returned by bears_2param_standard_fast or a similar function. The user should specify WHICH matrix in the results_object – i.e., scaled conditional likelihoods on downpass or uppass, or actual marginal probabilities of ancestral states. (The latter is the main thing of interest.) This specification is done via e.g. relprobs_matrix = results_object$relative_probs_of_each_stat
states_list	A list of the possible states/geographic ranges, in 0-based index form.

Value

area_probs The probability of presence in each area.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

bears_2param_standard_fast, get_ML_states, get_ML_probs, infprobs_to_probs_of_each_area_from_relprobs

Examples

```
testval=1
```

---

infprobs_to_probs_of_each_area_from_relprobs

Convert relative probabilities matrix to the probabilities of presence in each area

---

Description

Biogeographic inference in LAGRANGE and DIVA has focused heavily on inference of the exact ancestral state/geographic range. However, when the state space is large, there is often considerable uncertainty in the exact ancestral range. Even the ancestral state that confers the maximum likelihood on the data, and thus is the most probable ancestor, may have less than 50 probability, or even less (25 size of the state space. This function converts the probability of specific states/geographic ranges into the probability of presence/absence in each area. This can typically be inferred with much higher confidence.

Usage

```
infprobs_to_probs_of_each_area_from_relprobs(relprobs_matrix,
  states_list)
```

Arguments

relprobs_matrix	A matrix with nrows for nodes and columns for states, with each cell holding the relative probability of that state at that node.
states_list	A list of the possible states/geographic ranges, in 0-based index form.

Value

area_probs The probability of presence in each area.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

bears_2param_standard_fast, get_ML_states, get_ML_probs, infprobs_to_probs_of_each_area

Examples

```
testval=1
```

---

is.not.na *Check for not NA*

Description

A utility function.

Usage

```
is.not.na(x)
```

Arguments

x	Thing to check for NA

Value

TRUE or FALSE

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2

Examples

```
test=1
```

---

label_nodes_postorder_phylo3

Add postorder node number labels to a phylo3 tree object.

Description

Adds phylobase phylo4 postorder node number labels to a phylo tree object.

Usage

```
label_nodes_postorder_phylo3(tr2)
```

Arguments

tr2	phylo tree object.

Value

tr2 A phylo tree object with node labels added.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo, phylo4

Examples

```
test=1
```

---

letter_strings_to_tipranges_df

Convert ranges in the form of letters (A, AB, BFG, etc.) to a tipranges *object*

Description

This function converts ranges in the form of concatenated letters (A, AB, BFG, etc.) to binary state number codes. Via apply, this is done to each member of the entire input vector of strings. It outputs tipranges object.

Usage

```
letter_strings_to_tipranges_df(letter_strings,
  letter_codes_in_desired_order = "alphabet",
  tipnames_in_order = NULL)
```

Arguments

letter_strings	A list of ranges in concatenated letter form ("A", "AB", "BFG", etc.)
letter_codes_in_desired_order	The letter codes in the desired order. The default keyword, "alphabet", uses the standard 26 capital letters; the output binary codes will thus have 26 positions. If the user inputs fewer letters here, or puts them in another order, those will be used.
tipnames_in_order	If given, the input tipnames will be applied as rownames in the tipranges object. Default is NULL, which results in numbering the rows.

Value

tipranges An object of class tipranges.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

letter_string_to_binary, binary_range_to_letter_code_list, binary_ranges_to_letter_codes, getranges_from_LagrangePHYLIP

Examples

```
testval=1
letter_strings = c("A", "B", "C", "AB", "AC", "BC", "ABC")
letter_strings_to_tipranges_df(letter_strings)

letter_strings = c("A", "B", "C", "AB", "AC", "BC", "ABC")
letter_strings_to_tipranges_df(letter_strings,
tipnames_in_order=paste("tip", seq(1,7), sep=""))
```

---

letter_string_to_binary

Convert ranges in the form of letters (A, AB, BFG, etc.) to binary state number codes

---

Description

This function takes a letter string (e.g. ABD) and converts to binary encoding (e.g. 1101).

Usage

```
letter_string_to_binary(letter_string,
  letter_codes_in_desired_order = "alphabet")
```

Arguments

letter_string	A string of letters (e.g. "ABD")
letter_codes_in_desired_order	The letter codes in the desired order. The default keyword, "alphabet", uses the standard 26 capital letters; the output binary codes will thus have 26 positions. If the user inputs fewer letters here, or puts them in another order, those will be used.

Value

numcodes A list with the binary codes.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

binary_ranges_to_letter_codes, binary_range_to_letter_code_list, letter_strings_to_tipranges_df

Examples

```
testval=1
letter_string = "ABD"
letter_string_to_binary(letter_string, letter_codes_in_desired_order="alphabet")

letter_string = "ABD"
letter_string_to_binary(letter_string,
letter_codes_in_desired_order=c("A","B","C","D","E","F"))

letter_string = "ABD"
letter_string_to_binary(letter_string,
letter_codes_in_desired_order=strsplit("ABCDEF", split="")[[1]])
```

LGcpp_MLstate_per_node *Get the ML states per node, from a states table*

Description

Given a table of states probabilities from either LGcpp_states_fn_to_table or LGcpp_states_fn_to_table, get the ML state for each node.

Usage

```
LGcpp_MLstate_per_node(states)
```

Arguments

states	A data.frame containing the node numbers, states, and state probabilities.

Details

See get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

MLstates A data.frame containing the node numbers, ML states, and state probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGcpp_states_fn_to_table, LGcpp_states_fn_to_table

Examples

```
test=1
```

LGcpp_splits_fn_to_table *Get the ML splits per node, from C++ LAGRANGE output*

Description

C++ LAGRANGE outputs a list of splits and split probabilities for each node. This function converts them to a table.

Usage

```
LGcpp_splits_fn_to_table(splits_fn)
```

Arguments

splits_fn	The filename of a C++ LAGRANGE output file.

Details

LAGRANGE outputs just the splits making up the top 95 first.

See LGpy_MLsplit_per_node for choosing the single ML split at each node, and see get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

splits A data.frame containing the node numbers, splits, and split probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_MLsplit_per_node

Examples

```
test=1

# splits_fn = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/
# examples/Psychotria_M0/LAGRANGE_C++/Psychotria_M0_lgcpp_out_splits00001.txt"
# LGcpp_splits_fn_to_table(splits_fn)
```

LGcpp_splits_fn_to_table2

Get the ML splits per node, from Python LAGRANGE output

Description

Python LAGRANGE outputs a list of splits and split probabilities for each node. This function converts them to a table.

Usage

```
LGcpp_splits_fn_to_table2(splits_fn)
```

Arguments

splits_fn	The filename of a Python LAGRANGE output file.

Details

LAGRANGE outputs just the splits making up the top 95 the probability, or 15 states, whichever comes first.

See LGpy_MLsplit_per_node for choosing the single ML split at each node, and see get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

splits A data.frame containing the node numbers, splits, and split probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_MLsplit_per_node

Examples

```
test=1
```

LGcpp_states_fn_to_table *Get the ML states per node, from C++ LAGRANGE output*

Description

C++ LAGRANGE outputs a list of states and state probabilities for each node. This function converts them to a table.

Usage

```
LGcpp_states_fn_to_table(states_fn)
```

Arguments

states_fn	The filename of a C++ LAGRANGE output file.

Details

LAGRANGE outputs just the states making up the top 95 first.

See LGcpp_MLstate_per_node for choosing the single ML state at each node, and see get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

states A data.frame containing the node numbers, states, and state probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGcpp_MLstate_per_node

Examples

```
test=1

# states_fn = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/
# inst/extdata/examples/Psychotria_M0/LAGRANGE_C++/
# Psychotria_M0_lgcpp_out_states00001.txt"
# LGcpp_states_fn_to_table(states_fn)
```

LGpy_MLsplit_per_node *Get the ML splits per node, from a splits table*

Description

Given a table of splits probabilities from either LGpy_splits_fn_to_table or LGcpp_splits_fn_to_table, get the ML state for each node.

Usage

```
LGpy_MLsplit_per_node(splits)
```

Arguments

splits	A data.frame containing the node numbers, splits, and split probabilities.

Details

See get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

MLsplits A data.frame containing the node numbers, ML splits, and split probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

LGpy_splits_fn_to_table

Get the ML splits per node, from Python LAGRANGE output

Description

Python LAGRANGE outputs a list of splits and split probabilities for each node. This function converts them to a table.

Usage

```
LGpy_splits_fn_to_table(splits_fn)
```

Arguments

splits_fn	The filename of a Python LAGRANGE output file.

Details

LAGRANGE outputs just the splits making up the top 95 the probability, or 15 states, whichever comes first.

See LGpy_MLsplit_per_node for choosing the single ML split at each node, and see get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

splits A data.frame containing the node numbers, splits, and split probabilities.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_MLsplit_per_node

Examples

```
test=1
```

---

list2str *Convert a list of items to a string*

Description

This is a shortcut to save time when converting a list of items to a string.

Usage

```
list2str(list1, spacer = " ")
```

Arguments

list1	The list to convert.
spacer	The space between each item. Default " ".

Value

tmpstr The output string.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

FosterIdiots

See Also

paste, as.character

Examples

```
test=1
```

---

lrttest *Calculate Likelihood Ratio Test (LRT)*

Description

The Likelihood Ratio Test (LRT) is a standard method for testing whether or not the data likelihood conferred by a more complex is significantly better than the data likelihood conferred by the simpler model, given a certain number of extra free parameters for the complex model. The null hypothesis is that there is no difference; rejection means that there is a statistically significant improvement in the more complex model.

Usage

```
lrttest(LnL_1, LnL_2, numparams1, numparams2,
  returnwhat = "pval")
```

Arguments

LnL_1	Log-likelihood of more complex model.
LnL_2	Log-likelihood of simpler complex model.
numparams1	Number of free parameters of the more complex model.
numparams2	Number of free parameters of the less complex model.
returnwhat	If "pval", just return the p-value. If "all", return all of the intermediate outputs.

Details

The LRT only works for situations in which the simpler model is nested within the more complex model (i.e., by taking some parameters of the more complex model and forcing them to be fixed to a specific value). In addition, the LRT may be unreliable in data-poor situations, and inherits whatever difficulties there may be in ML searches. See *Burnham et al. (2002)* for discussion.

This function assumes that LnL_1 and numparams1 refer to the more complex model, and that LnL_2 and numparams2 refer to the simpler model nested within the more complex one.

Value

pval or LRT_result2. Depends on returnwhat.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

lrttest_on_summary_table

Examples

```
test=1
```

---

lrttest_on_summary_table

Calculate Likelihood Ratio Test (LRT) results, and add to table

---

Description

The Likelihood Ratio Test (LRT) is a standard method for testing whether or not the data likelihood conferred by a more complex is significantly better than the data likelihood conferred by the simpler model, given a certain number of extra free parameters for the complex model. The null hypothesis is that there is no difference; rejection means that there is a statistically significant improvement in the more complex model.

Usage

```
lrttest_on_summary_table(restable, row_to_use_as_null,
  rows_to_exclude, returnwhat = "pval",
  add_to_table = TRUE)
```

Arguments

restable	A data.frame with at least columns named "LnL" and "nparams".
row_to_use_as_null	This is the row specifying the model to which the others will be compared in pairwise fashion.
rows_to_exclude	Some rows may have models that the simpler model cannot nest within. These should be excluded.
returnwhat	If "pval", just return the p-value. If "all", return all of the intermediate outputs.
add_to_table	If TRUE, add to the main table and return the main table. If FALSE, return just the Akaike Weights results.

Details

The LRT only works for situations in which the simpler model is nested within the more complex model (i.e., by taking some parameters of the more complex model and forcing them to be fixed to a specific value). In addition, the LRT may be unreliable in data-poor situations, and inherits whatever difficulties there may be in ML searches. See *Burnham et al. (2002)* for discussion.

This function assumes that the log-likelihoods are in the column "LnL", and the number of parameters is specified in "nparams"

Value

pval or LRTrow, both data.frame. Depends on returnwhat.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

lrttest

Examples

```
test=1
```

`make_dispersal_multiplier_matrix` *Make a default matrix of relative dispersal probabilities between areas*

Description

Given either a list of areas, or a list of states, this function provides a square dispersal matrix giving the relative probability of dispersal between areas. The function fills in these dispersals probabilities with the value 1. The user can then modify this as desired. dispersal_multipliers_matrix Default NULL distances_mat Default NULL x_exponent Default 0

Usage

```
make_dispersal_multiplier_matrix(areas = NULL,
  states_list = default_states_list(),
  dispersal_multipliers_matrix = NULL,
  distances_mat = NULL, x_exponent = 0)
```

Arguments

`areas`	A list of areas; if `NULL`, the states list will be used.
`states_list`	A list of states, where each state consists of a list of areas. A default example list is provided.
`dispersal_multipliers_matrix`	Default NULL.
`distances_mat`	Default NULL.
`x_exponent`	Default 0.

Details

If only a states list is given, the list of areas is calculated by getting `unique` values from the concatenated states list.

Value

`dispersal_multiplier_matrix` A square matrix, with 1s for all cells.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

FosterIdiots

See Also

make_relprob_matrix_de

Examples

```
testval=1
make_dispersal_multiplier_matrix(areas=NULL,
states_list=list("_", c("A"), c("B"), c("C"),
c("A","B"), c("B","C"), c("A","C"), c("A","B","C")))
make_dispersal_multiplier_matrix(areas=c("A","B","C","D"))
```

---

make_relprob_matrix_bi

Make a relative probability matrix for a single speciation (bifurcation) event

---

Description

Given the identity of the states/geographic ranges on the left branch (Lstates), right branch (Rstates), and ancestral areas (ancareas_txt_tmp), construct the (text version) of the row of transition probabilities. This means that each nonzero cell gets a *v* for a vicariance event, a *y* for a sympatric speciation/range-copying event, a *j* for a founder-event/jump speciation event, and an *s* for a sympatric-subset event.

Usage

```
make_relprob_matrix_bi(states_list = default_states_list(),
  split_ABC = FALSE, splitval = "",
  code_for_overlapping_subsets = NA, printwarn = 1)
```

Arguments

states_list	A list of states, where each state consists of a list of areas. A default example list is provided.
split_ABC	TRUE or FALSE If TRUE then each state/range in the input geographic ranges (states_list) will be split on the argument contained in split.
splitval	The character to split on.

code_for_overlapping_subsets
: Hypothetically, there is no reason that a vicariance event could happen, e.g. ABC–>AB, BC. This is disallowed in LAGRANGE BioGeoBEARS defaults, and, if one is going to employ the construct of discrete areas in the first place, overlaps should probably be avoided. But this parameter will allow experimentation. Here, code_for_overlapping_subsets=NA equals the default, and any other value means that overlapping vicariance events are included, with a number describing the number of areas in the overlap. Users could then manually convert this to a probability according to some function.

printwarn
: If printwarn>0 (printwarn=1 by default), then print to screen a message describing the size of the cladogenesis matrix.

Details

This function is utilized by apply in other functions (e.g.) in an attempt to speed up calculation over rows. However, processing of text formulas via apply will never be fast enough for large matrices; see cladoRcpp for optimized functions.

This text-based matrix later gets evaluated by other functions to calculate the numerical probabilities. I.e., if j=0 and the other forms of speciation have weights equal to each other, this is the LAGRANGE cladogenesis model.

NOTE: This function is veeeeeeery slow, even for only 3 areas (i.e. 2^3=8 geographic ranges). It is mostly useful for illustration. See cladoRcpp for drastic improvements in calculating cladogenesis models.

Value

probmat A matrix of strings, where each cell contains the parameters describing the conditional probability of that ancestor–>(Left descendant,Right descendant) range inheritance scenario.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

size_species_matrix, make_spmat_row

Examples

```
testval=1
probmat = make_relprob_matrix_bi(states_list=list("_",
c("A"), c("B"), c("C"), c("A","B"), c("B","C"), c("A","C"),
c("A","B","C")), split_ABC=FALSE, splitval="",
code_for_overlapping_subsets=NA, printwarn=1)
probmat
```

`make_relprob_matrix_de` *Make a relative dispersal probability matrix (in text form)*

Description

This function takes a list of states/geographic ranges, and makes a relative probability matrix describing the probability of transition between each state. These probabilities are described in terms of d, "dispersal" (actually range expansion) and "extinction" (actually local extirpation, or range contraction), as done in the program LAGRANGE (*Ree et al. (2008)*, *Smith et al. (2010)*).

Usage

```
make_relprob_matrix_de(states_list = default_states_list(),
  split_ABC = FALSE, split = "",
  remove_simultaneous_events = TRUE,
  add_multiple_Ds = TRUE,
 dispersal_multiplier_matrix = make_dispersal_multiplier_matrix(states_list))
```

Arguments

`states_list`	A list of states, where each state consists of a list of areas. A default example list is provided.
`split_ABC`	TRUE or FALSE If TRUE then each state/range in the input geographic ranges (`states_list`) will be split on the argument contained in `split`.
`split`	The character to split on.
`remove_simultaneous_events`	If TRUE (default, as in LAGRANGE and almost all phylogenetic Markov models), then it is assumed that all changes in geographic range along branches must happen one event at a time. If FALSE, simultaneous events are not excluded; this is not recommended. However, notably, a commonly-used biogeographic model (treating biogeography as a multistate discrete character in an ML framework, where every species/lineage inhabits one and only one area at any point in time) effectively is invoking a simultaneous event: e.g., A->B is a simultaneous range gain and range loss, from the perspective of the dispersal-extinction framework.
`add_multiple_Ds`	If TRUE (default, as in LAGRANGE), the probabilities of dispersal from each possible source area are added together.

dispersal_multiplier_matrix
: A user-provided dispersal multiplier matrix; the default is a matrix of 1s from make_dispersal_multiplier_matrix(states_list=states_list).

Details

The output data.frame, termed dedf (dedf=dispersal-extinction data.frame), contains the actual text of the formulas by which the transition probability matrix would be calculated. E.g., the example calculates the matrix corresponding to Equation 1 on p. 6 of Ree & Smith (2008).

Note that the geographic range-change process described here is a continuous-time process, where the probability of change is a function of branch length, and all transitions occur because of dispersal and extinction. LAGRANGE also implements a cladogenesis model (thus DEC – dispersal-extinction-cladogenesis) which describes an "instantaneous" process of geographic range change at speciation/lineage-splitting events. BioGeoBEARS allows users to turn on, turn off, or otherwise customize both the continuous-time model and the cladogenesis model.

Value

dedf The output data.frame, termed dedf (dedf=dispersal-extinction data.frame), contains the actual text of the formulas by which the transition probability matrix would be calculated.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

ReeSmith2008

SmithRee2010_CPPversion

Matzke_2012_IBS

FosterIdiots

See Also

make_dispersal_multiplier_matrix

Examples

```
testval=1

states_list = list("_", c("A"), c("B"), c("C"), c
("A","B"), c("B","C"), c("A","C"), c("A","B","C"))
```

```
states_list = areas_list_to_states_list_new(
areas=c("A","B","C"), include_null_range=TRUE, split_ABC=TRUE)
states_list

dedf = make_relprob_matrix_de(states_list=states_list,
split_ABC=FALSE, split="", remove_simultaneous_events=TRUE,
add_multiple_Ds=TRUE,
dispersal_multiplier_matrix=make_dispersal_multiplier_matrix(states_list=states_list))

dedf
```

`make_relprob_nummatrix_sp1` *Convert a observed-speciation transition matrix to an unobserved-speciation transition matrix (numeric version)*

Description

Convert a cladogenesis/speciation transition matrix (specifying the probability of each Left/Right descendant range pair, conditional on each ancestral state) of dimensions `numstates` by `numstates^2` to a square transition matrix of dimensions `numstates` by `numstates`, representing the probability of a transition when only one daughter survives in the tree.

Usage

```
make_relprob_nummatrix_sp1(probmat, spPmat,
  split = "\\|")
```

Arguments

`probmat`	A matrix of text, describing each of the allowed range-inheritance events. Assumes that column names are in the "A|B" format.
`spPmat`	A matrix of numbers, where each cell contains the conditional probability of that ancestor–>(Left descendant,Right descendant) range inheritance scenario.
`split`	The value to split Left/Right pairs on (e.g., "A|B" –> "A", "B")

Details

This matrix could be used to quantify the probability of range-change along a branch due to unobserved speciation events; all that would be required would be an estimate of the number of unobserved speciation events on the branch, and treating this as a Poisson process. (Note: this assumes that the probability of either branch surviving is identical, which might not be the case. See the GeoSSE (*Goldberg et al. (2011)*) and ClaSSE (*"Goldberg et al. (2012)*) for the beginnings of work on this, with 2 and 3 geographic areas, respectively.

Value

`newmat` A new square matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/ http://tigger.uic.edu/~eeg/code/code.html

Goldberg_etal_2011_GeoSSE

Goldberg_Igic_2012_ClaSSE

Matzke_2012_IBS

ReeSmith2008

See Also

make_relprob_matrix_bi, make_relprob_txtmatrix_sp1, paste_rows_without_zeros

Examples

```
testval=1
spmat = make_relprob_matrix_bi(states_list=list("_", c("A"),
c("B"), c("C"), c("A","B"), c("B","C"), c("A","C"), c("A","B","C")),
split_ABC=FALSE, splitval="", code_for_overlapping_subsets=NA, printwarn=1)
spmat

spPmat = symbolic_to_relprob_matrix_sp(spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=0, v=1, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat

newmat = make_relprob_nummatrix_sp1(probmat=spmat, spPmat=spPmat, split="\\\\|")
newmat
```

---

make_relprob_txtmatrix_sp1

Convert a observed-speciation transition matrix to an unobserved-speciation transition matrix (text version)

Description

Convert a cladogenesis/speciation transition matrix (specifying the probability of each Left/Right descendant range pair, conditional on each ancestral state) of dimensions `numstates` by `numstates^2` to a square transition matrix of dimensions `numstates` by `numstates`, representing the probability of a transition when only one daughter survives in the tree.

Usage

```
make_relprob_txtmatrix_sp1(probmat, split = "\\|")
```

Arguments

probmat	A matrix of text, describing each of the allowed range-inheritance events. Assumes that column names are in the "A|B" format.
split	The value to split Left/Right pairs on (e.g., "A|B" –> "A", "B")

Details

This matrix could be used to quantify the probability of range-change along a branch due to unobserved speciation events; all that would be required would be an estimate of the number of unobserved speciation events on the branch, and treating this as a Poisson process. (Note: this assumes that the probability of either branch surviving is identical, which might not be the case. See the GeoSSE (*Goldberg et al. (2011)*) and ClaSSE (*"Goldberg et al. (2012)*) for the beginnings of work on this, with 2 and 3 geographic areas, respectively.

Value

newmat A new square matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/ http://tigger.uic.edu/~eeg/code/code.html

Goldberg_etal_2011_GeoSSE

Goldberg_Igic_2012_ClaSSE

Matzke_2012_IBS

ReeSmith2008

See Also

make_relprob_matrix_bi, make_relprob_nummatrix_sp1

Examples

```
testval=1
probmat = make_relprob_matrix_bi(states_list=list("_", c("A"),
c("B"), c("C"), c("A","B"), c("B","C"), c("A","C"), c("A","B","C")),
split_ABC=FALSE, splitval="", code_for_overlapping_subsets=NA, printwarn=1)
probmat

newmat = make_relprob_txtmatrix_sp1(probmat=probmat, split="\\\\|")
newmat
```

`make_spmat_row` *Construct a (text) cell of the cladogenesis/speciation matrix*

Description

Given the identity of the states/geographic ranges on the left branch (`Lstates`), right branch (`Rstates`), and ancestral areas (`ancareas_txt_tmp`), construct the (text version) of the row of transition probabilities. This means that each nonzero cell gets a *v* for a vicariance event, a *y* for a sympatric speciation/range-copying event, a *j* for a founder-event/jump speciation event, and an *s* for a sympatric-subset event.

Usage

```
make_spmat_row(Lstates, Rstates, ancareas_txt_tmp,
  splitval = "", code_for_overlapping_subsets = NA)
```

Arguments

`Lstates`	A string listing the possible left states, which will be split by `splitval`.
`Rstates`	A string listing the possible right states, which will be split by `splitval`.
`ancareas_txt_tmp`	A string listing the possible ancestral states, which will be split by `splitval`.
`splitval`	The character to split on.
`code_for_overlapping_subsets`	Hypothetically, there is no reason that a vicariance event could happen, e.g. ABC–>AB, BC. This is disallowed in `LAGRANGE` BioGeoBEARS defaults, and, if one is going to employ the construct of discrete areas in the first place, overlaps should probably be avoided. But this parameter will allow experimentation. Here, `code_for_overlapping_subsets=NA` equals the default, and any other value means that overlapping vicariance events are included, with a number describing the number of areas in the overlap. Users could then manually convert this to a probability according to some function.

Details

This function is utilized by apply in other functions (e.g.) in an attempt to speed up calculation over rows. However, processing of text formulas via apply will never be fast enough for large matrices; see cladoRcpp for optimized functions.

This text-based matrix later gets evaluated by other functions to calculate the numerical probabilities. I.e., if j=0 and the other forms of speciation have weights equal to each other, this is the LAGRANGE cladogenesis model.

Value

returncell The text specifying the type of transition.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

size_species_matrix, make_relprob_matrix_bi

Examples

```
testval=1
```

---

mapply_calc_obs_like *Mapply version of calc_obs_like()*

Description

This function applies calc_obs_like to all cells of the input matrices obs_target_species and obs_all_species. These matrices obviously must have the same dimensions.

Usage

```
mapply_calc_obs_like(truly_present = TRUE,
  obs_target_species, obs_all_species,
  mean_frequency = 0.1, dp = 1, fdp = 0)
```

Arguments

truly_present	Is the OTU of interest known/conditionally assumed to be truly present (TRUE) or truly absent (FALSE)?
obs_target_species	A scalar or column/vector/matrix of detection counts, e.g. as produced from the output from read_detections.
obs_all_species	A scalar or column/vector/matrix of detection counts, e.g. as produced from the output from read_controls.
mean_frequency	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.
dp	The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.
fdp	The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer mean_frequency, dp and fdp all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.

Details

The inputs are the same as for calc_obs_like, except that obs_target_species and obs_all_species can be matrices.

Value

pp_df A matrix of the natural log-likelihood of the data, given the model & assumption of true presence or absence.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Bayes'_theorem

Matzke_2012_IBS

Bottjer_Jablonski_1988

Bayes_1763

See Also

calc_obs_like, calc_post_prob_presence, mapply_calc_post_prob_presence, Pdata_given_rangerow, mapply, tiplikes_wDetectionModel

Examples

```
test=1
# Calculate likelihood of data, given presence in an area,
# given a dp (detection probability) and detection model.

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

OTUnames=NULL
areanames=NULL
tmpskip=0

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)

detects_df
controls_df
detects_df / controls_df


# Calculate data likelihoods, and posterior probability of presence=TRUE
mean_frequency=0.1
dp=1
fdp=0

mapply_calc_obs_like(truly_present=TRUE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)

mapply_calc_obs_like(truly_present=FALSE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)
```

```
mapply_calc_post_prob_presence(prior_prob_presence=0.01,
obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)
```

`mapply_calc_post_prob_presence` *Mapply version of calc_post_prob_presence()*

Description

This function applies `calc_post_prob_presence` to all cells of the input matrices `obs_target_species` and `obs_all_species`. These matrices obviously must have the same dimensions.

Usage

```
mapply_calc_post_prob_presence(prior_prob_presence = 0.01,
  obs_target_species, obs_all_species,
  mean_frequency = 0.1, dp = 1, fdp = 0,
  print_progress = "")
```

Arguments

`prior_prob_presence`	The prior probability of presence, i.e. when no detection or taphonomic control data whatsoever is available. Default is set to 0.01 which expresses my totally uninformed bias that in whatever your data is, your species of interest probably doesn't live in the typical area you are looking at.
`obs_target_species`	A scalar or column/vector/matrix of detection counts, e.g. as produced from the output from `read_detections`.
`obs_all_species`	A scalar or column/vector/matrix of detection counts, e.g. as produced from the output from `read_controls`.
`mean_frequency`	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.
`dp`	The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.

`fdp` The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer `mean_frequency`, `dp` and `fdp` all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.

`print_progress` If not the default (`""`), print whatever is in print_progress, followed by a space (for error checking/surveying results).

Details

The inputs are the same as for `calc_post_prob_presence`, except that `obs_target_species` and `obs_all_species` can be matrices.

Value

`pp_df` A matrix of the posterior probability of presence, given the prior probability, the model parameters, and the data.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Bayes'_theorem

Matzke_2012_IBS

Bottjer_Jablonski_1988

Bayes_1763

See Also

`calc_obs_like`, `calc_post_prob_presence`, `mapply_calc_obs_like` `Pdata_given_rangerow`, `mapply`, `tiplikes_wDetectionModel`

Examples

```
# Calculate posterior probability of presence in an area,
# given a dp (detection probability) and detection model.

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

OTUnames=NULL
areanames=NULL
tmpskip=0

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)

detects_df
controls_df
detects_df / controls_df


# Calculate data likelihoods, and posterior probability of presence=TRUE
mean_frequency=0.1
dp=1
fdp=0

mapply_calc_obs_like(truly_present=TRUE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)

mapply_calc_obs_like(truly_present=FALSE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)

mapply_calc_post_prob_presence(prior_prob_presence=0.01,
obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)
```

`mapply_likelihoods` *Use mapply on matrix exponentiations – post-byte-compiling*

Description

During the likelihood calculations from the tips to the root of a tree, the transition matrix Qmat needs to be exponentiated for each branch length in the tree. This is the slowest step of the likelihood calculation, especially for large matrices. This function performs this with mapply.

Usage

```
mapply_likelihoods(Qmat, phy2, transpose_needed)
```

Arguments

Qmat	an input Q transition matrix.
phy2	A phylogenetic tree.
transpose_needed	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal).

Details

Byte-compiling is supposed to speed up functions; this is an attempt to do this on the rexpokit function expokit_dgpadm_Qmat. It is also possible to byte-compile everything during package installation (via ByteCompile: true in the DESCRIPTION file), which is implemented in BioGeoBEARS, so this may be redundant.

mapply_likelihoods_prebyte gets byte-compiled into mapply_likelihoods.

See http://dirk.eddelbuettel.com/blog/2011/04/12/ for discussion of the compile package.

Value

independent_likelihoods_on_each_branch The output matrix of the likelihoods for each state on each branch.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

mapply, expokit_dgpadm_Qmat, expokit_dgpadm_Qmat2, compile, cmpfun

Examples

```
testval=1
```

mapply_likelihoods_prebyte *Use mapply on matrix exponentiations – pre-byte-compiling*

Description

During the likelihood calculations from the tips to the root of a tree, the transition matrix Qmat needs to be exponentiated for each branch length in the tree. This is the slowest step of the likelihood calculation, especially for large matrices. This function performs this with mapply.

Usage

```
mapply_likelihoods_prebyte(Qmat, phy2, transpose_needed)
```

Arguments

Qmat	an input Q transition matrix.
phy2	A phylogenetic tree.
transpose_needed	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal).

Details

Byte-compiling is supposed to speed up functions; this is an attempt to do this on the rexpokit function expokit_dgpadm_Qmat. It is also possible to byte-compile everything during package installation (via ByteCompile: true in the DESCRIPTION file), which is implemented in BioGeoBEARS, so this may be redundant.

mapply_likelihoods_prebyte gets byte-compiled into mapply_likelihoods.

See http://dirk.eddelbuettel.com/blog/2011/04/12/ for discussion of the compile package.

Value

independent_likelihoods_on_each_branch The output matrix of the likelihoods for each state on each branch.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

mapply, expokit_dgpadm_Qmat, expokit_dgpadm_Qmat2, compile, cmpfun

Examples

```
testval=1
```

`map_LGpy_MLsplits_to_tree` *Take the table of ML splits and node number and map on tree (Python version)*

Description

Given a table of splits probabilities from LGpy_splits_fn_to_table, map the splits on the tree.

Usage

```
map_LGpy_MLsplits_to_tree(MLsplits_LGpy, tr,
  tiprange_names)
```

Arguments

MLsplits_LGpy	A data.frame containing the node numbers, splits, and split probabilities.
tr	An ape phylo object
tiprange_names	The geographic ranges at the tips (i.e. the input data)

Details

See get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

MLsplits_LGpy A data.frame containing the node numbers, ML splits, and split probabilities; re-ordered for this plot

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

`map_LG_MLsplits_to_tree` *Take the table of ML splits and node number and map on tree (C++ LAGRANGE version)*

Description

Given a table of splits probabilities from LGcpp_splits_fn_to_table, map the splits on the tree.

Usage

```
map_LG_MLsplits_to_tree(MLsplits_LGcpp, tr,
  tiprange_names, removechar = NULL, type = "C++")
```

Arguments

MLsplits_LGcpp	A data.frame containing the node numbers, splits, and split probabilities.
tr	An ape phylo object
tiprange_names	The geographic ranges at the tips (i.e. the input data)
removechar	The character to remove, if needed.
type	The type of LAGRANGE input (default C++)

Details

See get_lagrange_nodenums for connecting these node numbers to APE node numbers.

Value

MLsplits_LGcpp A data.frame containing the node numbers, ML splits, and split probabilities; reordered for this plot.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

---

map_LG_MLsplits_to_tree_corners

Map splits to the corners on a phylogeny

Description

What it says.

Usage

```
map_LG_MLsplits_to_tree_corners(MLsplits, tr, tipranges,
  removechar = NULL, type = "C++",
  statesColors_table = "default", bgcol = "green3",
  areanames = "default", newplot = TRUE, ...)
```

Arguments

MLsplits	A data.frame containing the node numbers, splits, and split probabilities.
tr	An ape phylo object
tipranges	Tipranges object
removechar	The character to remove, if needed.
type	The type of LAGRANGE input (default C++)
statesColors_table	If not default, a table with a color for each area combination.
bgcol	The background color
areanames	The area names, if different from those in the tipranges object
newplot	Default TRUE; should there be a new plot, or should the splits be added to another plot?
...	Additional arguments to standard functions

Value

MLsplits The splits table, ordered appropriately.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

map_LG_MLstates_to_tree *Map states to the nodes on a phylogeny*

Description

What it says.

Usage

```
map_LG_MLstates_to_tree(MLstates_LGcpp, tr, tipranges,
  removechar = NULL, type = "C++",
  statesColors_table = "default", bgcol = "green3",
  areanames = "default", newplot = TRUE, ...)
```

Arguments

MLstates_LGcpp	A data.frame containing the node numbers, states, and states probabilities.
tr	An ape phylo object
tipranges	Tipranges object
removechar	The character to remove, if needed.
type	The type of LAGRANGE input (default C++)
statesColors_table	If not default, a table with a color for each area combination.
bgcol	The background color
areanames	The area names, if different from those in the tipranges object
newplot	Default TRUE; should there be a new plot, or should the splits be added to another plot?
...	Additional arguments to standard functions

Value

MLstates_LGcpp The states table, ordered appropriately.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

---

match_list1_in_list2 *Return TRUE for list1 items when they occur in list2*

Description

Return matching TRUE/FALSE values. E.g. list1 (e.g. a big list) TRUE if it is found in list2 (e.g. a smaller list)

Usage

```
match_list1_in_list2(list1, list2)
```

Arguments

list1	The list of things you want to check
list2	The list of things you want to check against

Details

Utility function for confused.

Value

matchlist The TRUE/FALSE list for list1

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

match

Examples

```
test=1
```

---

maxsize *Get the maximum rangesize for a given ancestral rangesize*

Description

This function returns the maximum descendant rangesize for a given ancestral rangesize, given a list of 0/1 values specifying the possibility of each descendant rangesizes.

Usage

```
maxsize(areasizes_possible_01)
```

Arguments

areasizes_possible_01
: A list of 0/1 values, indicating whether an range of that size (rangesize = 1-based index = 1, 2, 3...) is possible (1) or not (0).

Details

This is mostly a utility function used within apply within other functions.

Value

max_number_of_areas The maximum number of areas

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

apply

Examples

```
testval=1
areasizes_possible_01 = c(1,1,1,0,0)
maxsize(areasizes_possible_01)
```

---

merge_words_nonwords *Merge lists of words and nonwords (numbers) that may be of different length*

Description

Utility function.

Usage

```
merge_words_nonwords(words, nonwords)
```

Arguments

words	A list of words
nonwords	A list of nonwords

Value

sentence A text string.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_model_object

Examples

```
test=1
```

---

meval eval() *function for use in* sapply

Description

meval is a wrapper for eval, to allow use in sapply.

Usage

```
meval(equation_txt)
```

Arguments

equation_txt	The text of the equation to run eval on – e.g., from a cell of a text-based transition matrix.

Details

This is an attempt to speed up the use of eval; in general use of eval to convert a text version of a transition matrix to a numeric version with probabilities is a poor, slow choice; but it can be useful for examples and display purposes.

See cladoRcpp for fast C++ implementations of transition matrix setup.

Value

outval The numeric result of eval.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve

Examples

```
testval=1
d = 0.1
equation_txt = "1*d+1*d"
meval(equation_txt)
```

---

mix_colors_for_states *Mix colors logically to produce colors for multi-area ranges*

Description

Like it says.

Usage

```
mix_colors_for_states(colors_matrix,
  states_list_0based_index, exclude_null = TRUE)
```

Arguments

colors_matrix	A column with a color for each single area
states_list_0based_index	States list giving areas, 0-based
exclude_null	If TRUE, null ranges are excluded (however coded). Default TRUE.

Value

colors_list_for_states The colors for the ML states

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

optim

Examples

```
testval=1
```

---

moref *print to screen the header of a file*

Description

This does the rough equivalent of the UNIX function more, but within R.

Usage

```
moref(fn, printnotcat = FALSE)
```

Arguments

fn	A filename.
printnotcat	If TRUE, use print instead of cat. Default FALSE.

Value

Nothing returned.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster
Matzke_2012_IBS

See Also

scan

Examples

```
test=1
```

nodenums_bottom_up *Assign node labels in bottom-up, left-first format (as in e.g. r8s)*

Description

This function assigns node numbers by tracing up from the root. This corresponds to the node numbers in e.g. r8s (*Sanderson (2003)*).

Usage

```
nodenums_bottom_up(tr)
```

Arguments

tr	A tree object in phylo format.

Value

traverse_records

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Sanderson_2003_r8s

Matzke_2012_IBS

r8s

Marazzi_etal_Sanderson_2012_r8s_morph

See Also

phylo4,

Examples

```
test=1
```

normat *Utility functions to help deal with matrices Normalize a transition matrix*

Description

normat normalizes a square transition matrix, such that each row sums to 0, and the diagonal equals the negative of the sum of the rest of the cells in the row. This matrix can then be exponentiated by values of t (time or another measure of branch length) to produce transition probabilities for any given value of t.

Usage

```
normat(relative_matrix)
```

Arguments

relative_matrix
: A square matrix giving the relative probabilities/weights of transitions.

Details

See *Foster (2001)* for a succinct summary of transition matrices and their exponentiation.

Value

m A Q matrix, i.e. normalized transition matrix (Qmat)

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

FosterIdiots

Examples

```
testval=1
```

np *normalizePath shortcut*

Description

Utility function that runs normalizePath. Useful for running on Mac vs. Windows.

Usage

```
np(path = path, ...)
```

Arguments

path	The path to run normalizePath on.
...	Additional arguments to normalizePath.

Value

path The path that was normalized.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

normalizePath

Examples

```
# Get a path
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
extdata_dir

path = paste(extdata_dir, "//", "Psychotria_5.2.newick", sep="")
path

path = np(path)
path
```

nullsym_to_NA *Convert a specified null range code to NA*

Description

Takes a matrix mat, converts any instances of the nullsym symbol to NA.

Usage

```
nullsym_to_NA(mat, nullsym = "-")
```

Arguments

mat	A matrix.
nullsym	A character specifying the null symbol.

Value

mat The revised matrix

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

remove_null_rowcols_from_mat

Examples

```
testval=1
mat = matrix(c("-",1,1,1,"-",1,1,1,"-"), nrow=3)
mat
mat2 = nullsym_to_NA(mat, nullsym="-")
mat2
```

order_LGnodes *Order LAGRANGE-numbered nodes so that they can be plotted in R*

Description

What it says.

Usage

```
order_LGnodes(MLsplits_LGcpp, tr = NULL,
  removechar = NULL, type = "C++", type2 = "splits")
```

Arguments

MLsplits_LGcpp	A data.frame containing the node numbers, splits, and split probabilities.
tr	An ape phylo object
removechar	The character to remove, if needed.
type	The type of LAGRANGE input (default C++)
type2	"splits" or "states"

Value

MLsplits The splits table, ordered appropriately.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table, LGcpp_splits_fn_to_table

Examples

```
test=1
```

`order_tipranges_by_tr` *Order the tipranges in a tipranges object so they match the order of tips in a tree*

Description

Utility function. What it says. Life can get very confusing if you don't do this before plotting.

Usage

```
order_tipranges_by_tr(tipranges, tr)
```

Arguments

`tipranges`	A tipranges object.
`tr`	An ape tree object.

Value

`tipranges` The reordered data.frame

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

unlist

Examples

```
test=1
```

order_tipranges_by_tree_tips *Reorder the rows in a* tipranges *object, to correspond to tree tips*

Description

The tipranges object, as read from a LAGRANGE/PHYLIP-style geography file, may not have the species names as the same order as they are in the tips of the tree. This function allows the user to reorder them to match the tree

Usage

```
order_tipranges_by_tree_tips(tipranges, tr)
```

Arguments

tipranges	An object of class tipranges.
tr	A phylo tree object.

Value

tipranges An object of class tipranges

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

SmithRee2010_CPPversion

See Also

tipranges_to_area_strings, define_tipranges_object, save_tipranges_to_LagrangePHYLIP

Examples

```
testval=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
# Set the filename (Hawaiian Psychotria from Ree & Smith 2008)

trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(trfn)

fn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))
tipranges1 = getranges_from_LagrangePHYLIP(lgdata_fn=fn)
tipranges1

# Reorder the tipranges object
tipranges2 = order_tipranges_by_tree_tips(tipranges1, tr)
tipranges2
```

---

params_into_BioGeoBEARS_model_object

Feed modified parameters back into a BioGeoBEARS model object

---

Description

What it says.

Usage

```
params_into_BioGeoBEARS_model_object(BioGeoBEARS_model_object,
  params)
```

Arguments

BioGeoBEARS_model_object	The BioGeoBEARS_model object, of class BioGeoBEARS_model
params	parameter vector

Value

BioGeoBEARS_model_object The BioGeoBEARS_model object, of class BioGeoBEARS_model

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_model_object

Examples

```
test=1
```

---

parse_lagrange_output *Parse the output file from python* LAGRANGE

Description

Parse the output of a C++ LAGRANGE run.

Usage

```
parse_lagrange_output(outfn, outputfiles = FALSE,
  results_dir = getwd(), new_splits_fn = FALSE,
  new_states_fn = TRUE, filecount = 0)
```

Arguments

outfn	The C++ LAGRANGE output text file.
outputfiles	Should parsed output be written to files? Default FALSE.
results_dir	The directory outfn is in.
new_splits_fn	Should a text file containing a table of the splits and their probabilities be output? Default FALSE.
new_states_fn	Should a text file containing a table of the states and their probabilities be output? Default TRUE, unlike python LAGRANGE, C++ LAGRANGE *will* output the states at the nodes.
filecount	The starting number for the filecount (relevant if one is processing many files).

Details

This function parses the output of LAGRANGE, obtained by a command such as the following, run at a UNIX/Mac Terminal command line.

```
cd   /Users/nick/Desktop/__projects/_2011-07-15_Hannah_spider_fossils/_data/lagrange_for_nick
./lagrange_cpp palp_no_Lacun_v1_2nd387.lg >   lagrange_results_v1_2nd387.txt
```

C++ LAGRANGE can be obtained at https://code.google.com/p/lagrange/

Value

sumstats A data.frame containing the summary statistics (LnL, d and e rates, etc.) The splits filename is output to screen.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table

Examples

```
test=1
```

---

parse_lagrange_output_old

Parse the output file from python LAGRANGE *– older version*

Description

Parse the output of a C++ LAGRANGE run.

Usage

```
parse_lagrange_output_old(outfn, results_dir = getwd(),
  new_splits_fn = TRUE, new_states_fn = TRUE,
  filecount = 0)
```

Arguments

outfn	The C++ LAGRANGE output text file.
results_dir	The directory outfn is in.
new_splits_fn	Should a text file containing a table of the splits and their probabilities be output? Default TRUE.
new_states_fn	Should a text file containing a table of the splits and their probabilities be output? Default TRUE, unlike python LAGRANGE, C++ LAGRANGE *will* output the states at the nodes.
filecount	The starting number for the filecount (relevant if one is processing many files).

Details

This function parses the output of LAGRANGE, obtained by a command such as the following, run at a UNIX/Mac Terminal command line. This is an older version useful for automating processing of many files.

```
cd   /Users/nick/Desktop/__projects/_2011-07-15_Hannah_spider_fossils/_data/lagrange_for_nick
./lagrange_cpp palp_no_Lacun_v1_2nd387.lg >   lagrange_results_v1_2nd387.txt
```

C++ LAGRANGE can be obtained at https://code.google.com/p/lagrange/

Value

sumstats A data.frame containing the summary statistics (LnL, d and e rates, etc.) The splits filename is output to screen.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table

Examples

```
test=1
```

parse_lagrange_python_output
Parse the output file from python LAGRANGE

Description

Parse the output of a python LAGRANGE.

Usage

```
parse_lagrange_python_output(outfn = "output.results.txt",
  outputfiles = FALSE, results_dir = getwd(),
  new_splits_fn = TRUE, new_states_fn = FALSE,
  filecount = 0, append = FALSE)
```

Arguments

outfn	The python LAGRANGE output text file.
outputfiles	Should parsed output be written to files? Default FALSE.
results_dir	The directory outfn is in.
new_splits_fn	Should a text file containing a table of the splits and their probabilities be output? Default TRUE.
new_states_fn	Should a text file containing a table of the states and their probabilities be output? Default FALSE, as I don't believe python LAGRANGE will output the states at the nodes (C++ LAGRANGE will, however).
filecount	The starting number for the filecount (relevant if one is processing many files).
append	Should results be appended to preexisting file? (default FALSE)

Details

Python LAGRANGE is run from a UNIX/Terminal command-line with a command such as "python lagrangefilename.py". You will need to have the "lagrange" python directory in your working directory.

The input file can be obtained from http://www.reelab.net/lagrange/configurator/index (*Ree (2009)*).

Python comes installed on many machines, or can be downloaded from the Enthought Python Distribution (https://www.enthought.com/products/epd/).

Value

sumstats A data.frame containing the summary statistics (LnL, d and e rates, etc.) The splits filename is output to screen.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://code.google.com/p/lagrange/ https://www.enthought.com/products/epd/ http://www.reelab.net/lagrange/configurator/index

Ree2009configurator

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table

Examples

```
test=1
```

---

parse_lagrange_python_output_old

Parse the output file from python LAGRANGE *– old version*

Description

Parse the output of a python LAGRANGE output file. This is an older version useful for automating the parsing of a large number of files.

Usage

```
parse_lagrange_python_output_old(outfn = "output.results.txt",
  results_dir = getwd(), new_splits_fn = TRUE,
  new_states_fn = FALSE, filecount = 0)
```

Arguments

outfn	The python LAGRANGE output text file.
results_dir	The directory outfn is in.
new_splits_fn	Should a text file containing a table of the splits and their probabilities be output? Default TRUE.
new_states_fn	Should a text file containing a table of the splits and their probabilities be output? Default FALSE, as I don't believe python LAGRANGE will output the states at the nodes (C++ LAGRANGE will, however).
filecount	The starting number for the filecount (relevant if one is processing many files).

Details

Python LAGRANGE is run from a UNIX/Terminal command-line with a command such as "python lagrangefilename.py". You will need to have the "lagrange" python directory in your working directory.

The input file can be obtained from http://www.reelab.net/lagrange/configurator/index (*Ree (2009)*).

Python comes installed on many machines, or can be downloaded from the Enthought Python Distribution (https://www.enthought.com/products/epd/).

Value

sumstats A data.frame containing the summary statistics (LnL, d and e rates, etc.) The splits filename is output to screen.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://code.google.com/p/lagrange/ https://www.enthought.com/products/epd/ http://www.reelab.net/lagrange/configurator/index

Ree2009configurator

Matzke_2012_IBS

ReeSmith2008

See Also

get_lagrange_nodenums, LGpy_splits_fn_to_table

Examples

```
test=1
```

paste_rows_without_zeros *Concatenate cells in each row of a text-based transition matrix, excluding zeros*

Description

This is a utility function for make_relprob_txtmatrix_sp1.

Usage

```
paste_rows_without_zeros(tmpmat)
```

Arguments

tmpmat	A cladogenesis/speciation probability matrix (text-based) to collapse each row of.

Details

Convert e.g.:

```
A|A A|B A|C A|A,B A|B,C A|A,C   A|A,B,C
A s j j 0 0 0 0
B 0 j 0 0 0   0 0
C 0 0 j 0 0 0 0
A,B 0 v 0 b1 0   0 0
B,C 0 0 0 0 j 0 0
A,C 0 0 v 0 0   b1 0
A,B,C 0 0 0 0 v 0 b1
```

...to...

```
A B C A,B B,C A,C A,B,C
"s+j+j" "j"   "j" "v+b1" "j" "v+b1" "v+b1"
```

Value

tmpcol A list containing each row, concatenated

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

make_relprob_txtmatrix_sp1

Examples

```
testval=1

spmat = make_relprob_matrix_bi(states_list=list("_", c("A"),
c("B"), c("C"), c("A","B"), c("B","C"), c("A","C"), c("A","B","C")),
split_ABC=FALSE, splitval="", code_for_overlapping_subsets=NA, printwarn=1)
spmat
tmpcol = paste_rows_without_zeros(tmpmat=spmat)
tmpcol
```

Pdata_given_rangerow *Calculate probability of detection data given a true geographic range and a detection probability*

Description

This function calculates P(data|range,dp), i.e. the probability of some detection and taphonomic control counts, given the true geographic range/state, and dp, a detection probability (and, optionally, a false detection probability, fdp).

Usage

```
Pdata_given_rangerow(range_as_areas_TF, detects_df_row,
  controls_df_row, mean_frequency = 0.1, dp = 1, fdp = 0,
  return_LnLs = FALSE)
```

Arguments

range_as_areas_TF	The list of areas (as TRUE/FALSE) in this geographic range/state.
detects_df_row	A column/vector of detection counts, as produced from a row of the output from read_detections.
controls_df_row	A column/vector of detection counts, as produced from a row of the output from read_controls.

`mean_frequency` This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.

`dp` The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.

`fdp` The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer `mean_frequency`, `dp` and `fdp` all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.

`return_LnLs` If `FALSE` (default), return exp(sum(LnLs of data in each area)), i.e. the likelihood of the data, non-logged. If `TRUE`, return the LnLs of the data in each area.

Details

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

`likelihood_of_data_given_range` The (non-logged!) likelihood of the data given the input range, and the detection model parameters. If `return_LnLs=TRUE`, returns `LnLs_of_data_in_each_area`, the LnLs of the data in each area.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

calc_obs_like, mapply, tiplikes_wDetectionModel

Examples

```
testval=1

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

OTUnames=NULL
areanames=NULL
tmpskip=0

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)

detects_df
controls_df
detects_df / controls_df

mean_frequency=0.1
dp=1
fdp=0

mapply_calc_obs_like(truly_present=TRUE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)

mapply_calc_obs_like(truly_present=FALSE, obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)

mapply_calc_post_prob_presence(prior_prob_presence=0.01,
obs_target_species=detects_df,
obs_all_species=controls_df, mean_frequency, dp, fdp)
```

```
# Now, calculate the likelihood of the data given a geographic range
numareas = 4
tmpranges = list(c(0), c(1), c(0,1))
truerange_areas = tmpranges[[3]]
truerange_areas


# Build a TRUE/FALSE row specifying the ranges in this assumed true
# state/geographic range
range_as_areas_TF = matrix(data=FALSE, nrow=1, ncol=numareas)
range_as_areas_TF[truerange_areas+1] = TRUE
range_as_areas_TF

detects_df_row = detects_df[1,]
controls_df_row = controls_df[1,]

# Manual method, superceded by Pdata_given_rangerow():
# LnLs_of_data_in_each_area = mapply(FUN=calc_obs_like,
# obs_target_species=detects_df_row,
# obs_all_species=controls_df_row, truly_present=range_as_areas_TF,
# MoreArgs=list(mean_frequency=mean_frequency, dp=dp, fdp=fdp),
# USE.NAMES=TRUE)


# Calculate data likelihoods on for this geographic range
mean_frequency=0.1
dp=1
fdp=0

# Get the likelihood (the probability of the data, given this range)
likelihood_of_data_given_range = Pdata_given_rangerow(
range_as_areas_TF=range_as_areas_TF,
detects_df_row=detects_df_row,
controls_df_row=controls_df_row, mean_frequency=mean_frequency, dp=dp, fdp=fdp)
likelihood_of_data_given_range

# Return the raw log-likelihoods:
LnLs_of_data_in_each_area = Pdata_given_rangerow(range_as_areas_TF=range_as_areas_TF,
detects_df_row=detects_df_row,
controls_df_row=controls_df_row, mean_frequency=mean_frequency, dp=dp, fdp=fdp,
return_LnLs=TRUE)

detects_df_row
controls_df_row
LnLs_of_data_in_each_area

# The likelihood: the probability of the data in each area:
exp(LnLs_of_data_in_each_area)
```

`Pdata_given_rangerow_dp` *Calculate probability of detection data given a true geographic range and a detection probability*

Description

This function calculates P(data|range,dp), i.e. the probability of some detection and taphonomic control counts, given the true geographic range/state, and `dp`, a detection probability (and, optionally, a false detection probability, `fdp`).

Usage

```
Pdata_given_rangerow_dp(truerange_areas, numareas,
  detects_df, controls_df, mean_frequency = 0.1, dp = 1,
  fdp = 0)
```

Arguments

`truerange_areas`	The list of areas (as 0-based numbers) in this geographic range/state.
`numareas`	The function needs to know the total number of areas in the analysis.
`detects_df`	A column/vector of detection counts, as produced from a column of the output from `read_detections`.
`controls_df`	A column/vector of detection counts, as produced from a column of the output from `read_controls`.
`dp`	The detection probability. This is the per-sample probability that you will detect the OTU in question. In other words, the model assumes that each specimen from the taphonomic control group has a chance of being a representative of the OTU you are looking for. The default is 1, which assumes perfect detection, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. A value of 1 will only work when the taphonomic control count equals the detection count; any other data would have likelihood=0.
`fdp`	The false detection probability. This is probability of falsely concluding a detection occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. This option is being included for completeness, but it may not be wise to try to infer both `dp` and `fdp` at once due to identifiability issues (and estimation of fdp may take a very large amount of data).
`mean_frequency`	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be

present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.

Details

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

`dtf`

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

pdfit *Print a table to LaTeX format*

Description

This function prints a table to PDF via xtable and the LaTeX pdflatex function. It will only work if you have command-line LaTeX installed.

Usage

```
pdfit(table_vals, file_prefix = "tmptable",
  size = "\\tiny", gettex = FALSE, caption = NULL)
```

Arguments

table_vals	A table, hopefully produced by conditional_format_table.
file_prefix	The prefix for the output PDF and the intermediate files.
size	Font size, overriding getOption("xtable.size"). Default is "tiny" (with backslashes). You can also try "small". Input NULL (without quotes or backslashes) for medium. (NULL is the options default.)
gettex	If TRUE, the tex code for the table is returned.
caption	A caption, if desired.

Details

This function was inspired by http://tex.stackexchange.com/questions/15013/generate-a-pdf-containing-r-ou

Value

texfile The filename of the tex file.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

pdftable

Examples

```
test=1

# Setup data
## Not run:
data = c(2.768443, 1.869964, 5.303702, 4.733483,  2.123816,
18.551051, 5.483625,  3.590745,  18.772389)
result = matrix(data, nrow=3, byrow=TRUE)
result = as.data.frame(result)
names(result) = c("CV", "LCB", "UCB")
rownames(result) = c("within", "between", "total")
result
pdfit(table_vals=result)#'
## End(Not run)
```

---

pdftable *Print a table to LaTeX format*

Description

This function prints a table to PDF via pdfit, which calls xtable and the LaTeX pdflatex function. It will only work if you have command-line LaTeX installed.

Usage

```
pdftable(table_vals, pdffn = "tmptable.pdf",
  size = "\\tiny", tmpdir = "~", openPDF = TRUE,
  caption = NULL)
```

Arguments

table_vals	A table, hopefully produced by conditional_format_table.
pdffn	The filename for the output PDF (and the prefix for the intermediate files).
size	Font size, overriding getOption("xtable.size"). Default is "tiny" (with backslashes). You can also try "small". Input NULL (without quotes or backslashes) for medium. (NULL is the options default.)
tmpdir	The location for the temporary files.
openPDF	If TRUE, open the PDF via a system command.
caption	A caption, if desired.

Details

This function was inspired by http://tex.stackexchange.com/questions/15013/generate-a-pdf-containing-r-ou

Value

pdffn The filename of the PDF file.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

pdfit

Examples

```
test=1

# Setup data
## Not run:
data = c(2.768443, 1.869964, 5.303702, 4.733483,  2.123816,
18.551051, 5.483625,  3.590745,  18.772389)
result = matrix(data, nrow=3, byrow=TRUE)
result = as.data.frame(result)
names(result) = c("CV", "LCB", "UCB")
rownames(result) = c("within", "between", "total")
result
pdftable(table_vals=result)#'
## End(Not run)
```

---

plot_BioGeoBEARS_model

Graphical display of your anagenetic and cladogenetic biogeography models

---

Description

This function produces a graphical summary of the model stored in a BioGeoBEARS_run_object. This could be either an input model, or the result of the ML parameter search.

Usage

```
plot_BioGeoBEARS_model(obj, obj_is_run_or_results = NULL,
  plotwhat = "init", titletxt = "", statenames = NULL)
```

Arguments

obj	The input object, either a BioGeoBEARS_run_object (if so, set obj_is_run_or_results="run" or an output object from bears_optim_run (if so, specify obj_is_run_or_results="results".
obj_is_run_or_results	Specify "run" or "results", as described above for parameter obj.
plotwhat	Default is "init", which means plotting the starting model parameters. "est" plots the estimated model parameters.
titletxt	Additional text for the title of the plot
statenames	State names to pass to plot_cladogenesis_size_probabilities. If NULL (default), these are auto-generated assuming all areas up to the maximum number are allowed.

Details

Understanding of phylogenetic methoods in historical biogeography methods is hampered by the difficulty of displaying the models the computer is using. This function is one attempt to improve the situation, by plotting the relative weights of the various parameters.

Value

nada

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

plot_cladogenesis_size_probabilities, define_BioGeoBEARS_run, define_BioGeoBEARS_model_object

Examples

```
blah=1
```

plot_BioGeoBEARS_results *Plot the results of a BioGeoBEARS run*

Description

This function plots on a tree the highest-probability ancestral states (ranges), splits if desired (these are the ranges/states just after cladogenesis, and are plotted on the corners of a tree), and/or pie charts at nodes. A legend tying the relationship between colors and states/ranges is also optionally plotted.

Usage

```
plot_BioGeoBEARS_results(results_object,
  analysis_titletxt = NULL, addl_params = list(),
  plotwhat = "text", label.offset = NULL, tipcex = 0.8,
  statecex = 0.7, splitcex = 0.6, titlecex = 0.8,
  plotsplits = TRUE, plotlegend = FALSE,
  legend_ncol = NULL, legend_cex = 1,
  cornercoords_loc = "manual", include_null_range = TRUE,
  tr = NULL, tipranges = NULL)
```

Arguments

results_object	The results object from bears_optim_run (with ancestral states on).
analysis_titletxt	The main title of the plot. If NULL, results_object$inputs$description is checked.
addl_params	The function will plot the log-likelihood (LnL) and the ML values of the free parameters. If you want additional parameters plotted, list them here.
plotwhat	To plot the ML discrete states, "text". To plot a piechart of the relative probability of all the states, "pie".
label.offset	Offset for the tree tip labels. If NULL, program chooses 0.05 x tree height.
tipcex	cex value for the tiplabels (scaling factor, i.e. 0.5 is half size)
statecex	cex value for the states (scaling factor, i.e. 0.5 is half size). Used on piecharts if plotwhat="pie".
splitcex	cex value for the splits (scaling factor, i.e. 0.5 is half size). Used on piecharts if plotwhat="pie".
titlecex	cex value for the title (scaling factor, i.e. 0.5 is half size).
plotsplits	If TRUE, plot states on the corners – text or pie charts, depending on plotwhat.
plotlegend	If TRUE, make a (separate) plot with a legend giving the colors for each state/range, using colors_legend.

`legend_ncol` The number of columns in the legend. If `NULL` (default), the function calculates `floor(sqrt(length(possible_ranges_list_txt) / 2))` when the number of states is <=64, and `sqrt(ceiling(length(possible_ranges_list_txt)))` when > 64. Note that when you have hundreds of states, there is probably no good way to have a readable legend, and it is easier to just rely upon printing the character codes for the ML states in the plots, with the colors, and users can then see and trace the common colors/states by eye.

`legend_cex` The cex (character expansion size) for the legend. Defaults to 1, which means the `legend` function determines the size. The value 2.5 works well for 15 or 16 states/ranges.

`cornercoords_loc`
The directory location containing the R script `plot_phylo3_nodecoords.R`. This function, modified from the APE function `plot.phylo`, cannot be included directly in the R package as it contains C code that does not pass CRAN's R CMD check. The default, cornercoords_loc="manual", will not allow split states to be plot. The R script `plot_phylo3_nodecoords.R` is located in the BioGeoBEARS extension data directory, `extdata/a_scripts`. You should be able to get the full path with `list.files(system.file("extdata/a_scripts", package="BioGeoBEARS'`

`include_null_range`
If `TRUE` (default), the null range is included in calculation of colors. (Safest for now.)

`tr` Tree to plot on. Default `NULL`, which means the tree will be read from the file at `results_object$inputs$trfn`.

`tipranges` Tip geography data. Default `NULL`, which means the tree will be read from the file at `results_object$inputs$geogfn`.

Details

The legend is plotted on a separate plot, as it is very difficult to predict whether or not there will be space on any given tree plot. The utility of the legend is also debatable, as `plot_BioGeoBEARS_results` plots the colors and state/range names directly onto the plot. Any legend will get unwieldy above perhaps 32 states, which is just 5 areas with no constraints (see `numstates_from_numareas`, or type `numstates_from_numareas(numareas5, maxareas5, include_null_range=TRUE)`.

Note that this assumes that the ancestral states were calculated under the global optimum model (rather than the local optimum, with the model re-optimized for each possible state at each possible node, as done in e.g. `LAGRANGE`), and that these are marginal probabilities, i.e. this is not a joint reconstruction, instead it gives the probabilities of states at each node. This will not always be readable as a joint reconstruction (it could depict split scenarios that are not possible, for instance.)

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

get_leftright_nodes_matrix_from_results, corner_coords, plot.phylo, plot.phylo, tiplabels, legend, floor, ceiling, floor, numstates_from_numareas, system.file, list.files

Examples

```
test=1
```

---

plot_cladogenesis_size_probabilities

Graphical display of P(daughter rangesize) for your input or inferred speciation model

---

Description

This function produces a graphical summary of the daughter rangesize aspect of the cladogenesis model stored in a BioGeoBEARS_run_object. This could be either an input model, or the result of the ML parameter search.

Usage

```
plot_cladogenesis_size_probabilities(BioGeoBEARS_run_object,
  plotwhat = "est", statenames = NULL)
```

Arguments

BioGeoBEARS_run_object	The input run object.
plotwhat	Default is "input", which means plotting the starting model.
statenames	State names to pass to plot_cladogenesis_size_probabilities. If NULL (default), these are auto-generated assuming all areas up to the maximum number are allowed.

Details

The LAGRANGE DEC model assumes that at cladogenesis events, one daughter species has a range size of 1 area, and the other daughter either inherits the full ancestral range (sympatric-subset speciation), inherits the remainder of the ancestral range (vicariance), or as the same range (sympatric-range copying, which is the only option when the ancestor range is of size 1 area.

BioGeoBEARS enables numerous additional models. To see how these are similar or different from the LAGRANGE DEC cladogenesis model, this function can be used. E.g., comparison of LAGRANGE DEC to a DIVA-like model is instructive: see examples. DIVA disallows sympatric-subset speciation (probability 0 under this model), but allows classic vicariance (a species with 4 areas splitting into 2 daughters, each occupying 2 areas). LAGRANGE DEC gives 0 probability to a 4->(2,2) history, allowing only 4->(3,1) or 4->(1,3) histories.

Several additional plots relating to the cladogenesis model are also produced. Best used via plot_BioGeoBEARS_model.

Value

Nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

plot_BioGeoBEARS_model, define_BioGeoBEARS_run, define_BioGeoBEARS_model_object

Examples

```
blah=1
```

postorder_nodes_phylo4_return_table
Get a table of node numbers, including DIVA node numbers

Description

Various programs (annoyingly) label internal nodes in different ways. This function shows the corresponding node numbers for several different systems. This table can then be used to translate, when the user wishes to plot the output from various programs on the nodes of a tree. In particular, the last column contains the DIVA node-numbering scheme (*Ronquist (1996)*, *Ronquist (1997)*).

Usage

```
postorder_nodes_phylo4_return_table(tr4)
```

Arguments

tr4	A tree object in phylo or phylo4 format.

Details

There are many ways of numbering nodes in a tree. This returns a matrix containing (column 1) R's native internal numbering scheme, and (column 2) the node numbers in the downpass numbering used by C++ LAGRANGE, in particular in their .bgkey output file. Note that this is different from ape's pruningwise downpass ordering (see get_pruningwise_nodenums).

The python version of LAGRANGE labels internal nodes differently (sigh), but they are in the same order at least, so can just be renumbered from 1 to tr$Nnode to get them to match the C++ LAGRANGE node numbering.

DIVA has yet a different node numbering scheme; see postorder_nodes_phylo4_return_table

Value

postorder_table A data.frame showing the various corresponding node numbers.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Ronquist1996_DIVA

Ronquist_1997_DIVA

Matzke_2012_IBS

ReeSmith2008

See Also

get_pruningwise_nodenums, get_lagrange_nodenums, prt

Examples

```
test=1
```

---

post_prob_states *Calculate posterior probability of each states/geographic ranges, given prior probabilities and data likelihoods*

Description

This function calculates P(range|data,detection model), i.e. the probability of each possible range, given a prior probability of each range, and the likelihood of each range.

Usage

```
post_prob_states(prob_of_each_range,
  condlikes_of_data_on_each_range)
```

Arguments

prob_of_each_range	The probability of each range, given the prior probability of presence in each area.
condlikes_of_data_on_each_range	The probability of the data, conditional on each range (i.e., the likelihood), as found in e.g. a row of the output from tiplikes_wDetectionModel.

Details

The prior probability of each range should be considered by the user. Note that putting the same prior on the probability of occurrence in each individual range does NOT mean a flat prior on each state/geographic range. This fact is demonstrated in the function prob_of_states_from_prior_prob_areas.

Value

posterior_probs The posterior probability of each range.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Log_probability

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

prob_of_states_from_prior_prob_areas, tiplikes_wDetectionModel, rcpp_areas_list_to_states_list, Pdata_given_rangerow, calc_obs_like, mapply, read_detections, read_controls

Examples

```
testval=1

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)


# Setup
prior_prob_presence = 0.01
areas = c("K", "O", "M", "H")
numareas = length(areas)
maxareas = length(areas)
states_list_0based_index =
rcpp_areas_list_to_states_list(areas=areas, maxareas=maxareas,
                               include_null_range=TRUE)
states_list_0based_index

mean_frequency=0.1
dp=1
fdp=0

tip_condlikes_of_data_on_each_state =
```

```
tiplikes_wDetectionModel(states_list_0based_index, numareas=numareas,
detects_df, controls_df, mean_frequency=mean_frequency, dp=dp, fdp=fdp,
null_range_gets_0_like=TRUE)

tip_condlikes_of_data_on_each_state



# To get denominator, just iterate over all the states
# Prior probability
prob_of_each_range = prob_of_states_from_prior_prob_areas(states_list_0based_index,
numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=TRUE,
normalize_probs=TRUE)

# Likelihoods of the data on each range
condlikes_of_data_on_each_range = tip_condlikes_of_data_on_each_state[1,]

posterior_probs = post_prob_states(prob_of_each_range,
                  condlikes_of_data_on_each_range)
posterior_probs

# Should sum to 1
sum(posterior_probs)
```

`post_prob_states_matrix` *Calculate posterior probability of each states/geographic ranges, given prior probabilities and data likelihoods*

Description

This function calculates P(range|data,detection model), i.e. the probability of each possible range, given a prior probability of each range, and the likelihood of each range.

Usage

```
post_prob_states_matrix(prob_of_each_range,
  tip_condlikes_of_data_on_each_state)
```

Arguments

`prob_of_each_range`
: The probability of each range, given the prior probability of presence in each area.

`tip_condlikes_of_data_on_each_state`
: The probability of the data, conditional on each range (i.e., the likelihood), as found in e.g. a row of the output from `tiplikes_wDetectionModel`.

Details

The prior probability of each range should be considered by the user. Note that putting the same prior on the probability of occurrence in each individual range does NOT mean a flat prior on each state/geographic range. This fact is demonstrated in the function prob_of_states_from_prior_prob_areas.

Value

posterior_probs The posterior probability of each range.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Log_probability

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

prob_of_states_from_prior_prob_areas, tiplikes_wDetectionModel, rcpp_areas_list_to_states_list, Pdata_given_rangerow, calc_obs_like, mapply, read_detections, read_controls

Examples

```
testval=1

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
#extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)


# Setup
prior_prob_presence = 0.01
areas = c("K", "O", "M", "H")
numareas = length(areas)
maxareas = length(areas)
states_list_0based_index =
rcpp_areas_list_to_states_list(areas=areas, maxareas=maxareas,
```

```
                                include_null_range=TRUE)
states_list_0based_index

mean_frequency=0.1
dp=1
fdp=0

tip_condlikes_of_data_on_each_state =
tiplikes_wDetectionModel(states_list_0based_index, numareas=numareas,
detects_df, controls_df, mean_frequency=mean_frequency, dp=dp, fdp=fdp,
null_range_gets_0_like=TRUE)

tip_condlikes_of_data_on_each_state



# To get denominator, just iterate over all the states
# Prior probability
prob_of_each_range = prob_of_states_from_prior_prob_areas(states_list_0based_index,
numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=TRUE,
normalize_probs=TRUE)

posterior_probs_matrix = post_prob_states_matrix(prob_of_each_range,
                  tip_condlikes_of_data_on_each_state)
posterior_probs_matrix

# Should sum to 1
rowSums(posterior_probs_matrix)

# How does posterior probability correlate with likelihood and prior probability?
par(mfrow=c(1,2))
plot(x=jitter(log(tip_condlikes_of_data_on_each_state)),
y=jitter(log(posterior_probs_matrix)))
title("Correlation of data likelihoods\nand posterior probabilities")

prob_of_each_range_matrix = matrix(data=prob_of_each_range,
nrow=nrow(posterior_probs_matrix), ncol=length(prob_of_each_range))
plot(x=jitter(log(prob_of_each_range_matrix)),
y=jitter(log(posterior_probs_matrix)))
title("Correlation of prior probability\nand posterior probabilities")
```

`prflag` *Utility function to conditionally print intermediate results*

Description

Just a handy shortcut function, allowing other functions to optionally print, depending on the value of `printflag`.

Usage

```
prflag(x, printflag = TRUE)
```

Arguments

x	What to print.
printflag	If TRUE, do the printing

Value

nothing

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

get_daughters, chainsaw2

Examples

```
test=1
```

---

printall *Print an entire table to screen*

Description

Utility function. This prints a table to screen in chunks of chunksize_toprint (default=40). This avoids the annoying situation of not being able to see the bottom of a table. Note that if you print something huge, you will be waiting for awhile (try ESC or CTRL-C to cancel such an operation).

Usage

```
printall(dtf, chunksize_toprint = 40, printflag = TRUE)
```

Arguments

dtf	The data.frame to print.
chunksize_toprint	Number of lines to print. Default 50.
printflag	For optional printing. Passed to prflag.

Details

Another option is to reset options to something like: options(max.print=99999), but this is hard to remember. Your current setting is getOption("max.print").

Value

NULL

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

print, prflag

Examples

```
test=1
```

---

prob_of_states_from_prior_prob_areas

Calculate probability of detection data for each OTU at each range in a list of states/geographic ranges

---

Description

This function calculates P(data|range,dp), i.e. the probability of some detection and taphonomic control counts, given the true geographic range/state, and dp, a detection probability (and, optionally, a false detection probability, fdp).

Usage

```
prob_of_states_from_prior_prob_areas(states_list_0based_index,
  numareas = NULL, prior_prob_presence = 0.01,
  null_range_gets_0_prob = TRUE, normalize_probs = TRUE)
```

Arguments

states_list_0based_index	A states_list, 0-based, e.g. from rcpp_areas_list_to_states_list.
numareas	The number of areas being considered in the analysis. If NULL (default), this is calculated to be the maximum range length, or one plus the maximum 0-based index in any of the ranges.
prior_prob_presence	The prior probability of presence, i.e. when no detection or taphonomic control data whatsoever is available. Default is set to 0.01 which expresses my totally uninformed bias that in whatever your data is, your species of interest probably doesn't live in the typical area you are looking at.
null_range_gets_0_prob	If TRUE (default), then the null range is given zero probability. A null range has no areas occupied. This is equivalent to saying that you are sure/are willing to assume that the OTU exists somewhere in your study area, at the timepoint being considered. Null ranges are identified by length=1, containing NULL, NA, "", "_", etc.
normalize_probs	If TRUE, the probabilities of each range will be normalized so that they sum to 1. Otherwise, they won't.

Details

This function performs the operation for all states/ranges for all tips.

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

prob_of_each_range The probability of each range, given the prior probability of presence in each area.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

rcpp_areas_list_to_states_list, Pdata_given_rangerow, calc_obs_like, mapply, read_detections, read_controls

Examples

```
testval=1

prior_prob_presence = 0.01

areas = c("K", "O", "M", "H")
numareas = length(areas)
states_list_0based_index =
rcpp_areas_list_to_states_list(areas=areas, maxareas=4, include_null_range=TRUE)
states_list_0based_index
numareas = 4

mean_frequency=0.1
dp=1
fdp=0

prob_of_states_from_prior_prob_areas(states_list_0based_index, numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=TRUE,
normalize_probs=TRUE)

prob_of_states_from_prior_prob_areas(states_list_0based_index, numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=TRUE,
normalize_probs=FALSE)

prob_of_states_from_prior_prob_areas(states_list_0based_index, numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=FALSE,
normalize_probs=TRUE)

prob_of_states_from_prior_prob_areas(states_list_0based_index, numareas=numareas,
prior_prob_presence=prior_prob_presence, null_range_gets_0_prob=FALSE,
normalize_probs=FALSE)
```

process_optim *Extract* optim *results to a row*

Description

After running an ML (maximum likelihood) search with optim, optim returns a list with a variety of objects. It is often handy to have the parameter values, log-likelihood, etc., extracted to a table for comparison with other optimization runs. process_optim does this.

Usage

```
process_optim(optim_results, max_num_params = NULL)
```

Arguments

optim_results	A results object from optim
max_num_params	Specify the number of parameters, if known. If NULL, the code will try to guess.

Value

tmprow3 A row holding the optim results, which can then be added to a table with rbind.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

optim

Examples

```
testval=1
# Any optim() for a biogeography scenario would take too long to run for R CMD check.
```

prt *Print tree in table format*

Description

Learning and using APE's tree structure can be difficult and confusing because much of the information is implicit. This function prints the entire tree to a table, and makes much of the implicit information explicit. It is not particularly fast, but it is useful.

Usage

```
prt(t, printflag = TRUE, relabel_nodes = FALSE,
  time_bp_digits = 7, add_root_edge = TRUE,
  get_tipnames = FALSE, fossils_older_than = 0.6)
```

Arguments

t	A phylo tree object.
printflag	Should the table be printed to screen? Default TRUE.
relabel_nodes	Manually renumber the internal nodes, if desired. Default FALSE.
time_bp_digits	The number of digits to print in the time_bp (time before present) column. Default=7.
add_root_edge	Should a root edge be added? Default TRUE.
get_tipnames	Should the list of tipnames descending from each node be printed as a string in another column? This is slow-ish, but useful for matching up nodes between differing trees. Default FALSE.
fossils_older_than	Tips that are older than fossils_older_than will be marked as TRUE in a column called fossil. This is not currently set to 0, because Newick files can have slight precision issues etc. that mean not all tips quite come to zero. You can attempt to fix this with average_tr_tips (but make sure you do not inappropriately average in fossils!!).

Details

See http://ape.mpl.ird.fr/ape_development.html for the official documentation of R tree objects.

Value

dtf A data.frame holding the table. (Similar to the printout of a phylo4 object.)

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://ape.mpl.ird.fr/ape_development.html

Matzke_2012_IBS

See Also

phylo, average_tr_tips

Examples

```
test=1
```

---

prt_tree_to_phylo4 *prt_tree_to_phylo4*

Description

Converts a tree table (a prt_tree from the function prt, which prints trees to tables) to a phylobase phylo4 tree object.

Usage

```
prt_tree_to_phylo4(prt_tr)
```

Arguments

prt_tr	A prt_tree from the function prt.

Value

newtr A phylobase phylo4 tree object.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo4, prt

Examples

```
test=1
```

`prune_specimens_to_species` *Take a tree and species names/geography table and produce a pruned tree and tipranges object*

Description

This function takes a tree and species names/geography table and produces a pruned tree and (optionally) a tipranges object.

Usage

```
prune_specimens_to_species(original_tr, xls,
  group_name = "default", titletxt = "",
  areas_abbr = NULL, plot_intermediate = TRUE)
```

Arguments

`original_tr`	The input tree (an ape phylo object).
`xls`	The input table (a data.frame)
`group_name`	The name of the clade in the tree. For use in plots and output files. Default="default".
`titletxt`	Additional text for the plots. Default "".
`areas_abbr`	An optional table, containing the abbreviations (e.g. letters) corresponding to each region in xls$region. Default is NULL, in which case the program imposes A, B, C, D, etc. `areas_abbr` must have column headings `abbr` and `letter`.
`plot_intermediate`	If TRUE, the starting, ending, and intermediate stages of tree pruning are plotted.

Details

Often, users will have an phylogeny where the tips/OTUs (operational taxonomic units) are specimens rather than species. The analyses done by models like DEC, DEC+J, etc., in programs like LAGRANGE and BioGeoBEARS, assume as a core part of the model that species might occupy more than one areas. A phylogeny of specimens, then, would not be an appropriate input to these programs, as each single specimen can only be found in one region. The exception would occur when the researcher is confident that each species lives in only one region; in that case, the specimen geography is representative of the species geography.

This function requires a table containing

(1) Column "OTUs": all tipnames in the input tree (often, original specimen/original OTU names));

(2) Column "species": the corresponding species names;

(3) optionally, the geographic range inhabited by each specimen (column "region"). If an OTU has more than one geographic range in the original table, these should be split by "|".

When the pruning occurs, all tips belonging to the same species are cut, except the first.

NOTE: Tips that should be cut because they are outgroups, or because they are geographically outside of your domain of analysis, should be represented in xls$region by "out_group" or "Out". These will be cut from the final tree/geography table.

Value

The outputs are a `list` with a pruned tree and, optionally, a tipranges object.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`drop.tip`, `define_tipranges_object`,

Examples

```
testval=1
tipranges_object = define_tipranges_object()
tipranges_object

areanames = getareas_from_tipranges_object(tipranges_object)
areanames
```

`prune_states_list` *Cut down the states list according to areas_allowed_mat*

Description

Go through a list of states. Remove states that represent areas disallowed according to areas_allowed_mat. It is assumed (crucial!) that the areas in the `states_list`, and in the `areas_allowed_mat`, have the same order.

Usage

```
prune_states_list(states_list_0based_index,
  areas_allowed_mat)
```

Arguments

states_list_0based_index	A states_list, 0-based, e.g. from rcpp_areas_list_to_states_list
areas_allowed_mat	The matrix of area combinations allowed (represented by 1s)

Value

states_list_0based_index_new A 0-based list of allowed states/ranges

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

rcpp_areas_list_to_states_list

Examples

```
test=1
```

---

rangestxt_to_colors *Convert a list of ranges text (KOM, MH, KOMIH, etc.)*

Description

Like it says.

Usage

```
rangestxt_to_colors(possible_ranges_list_txt,
  colors_list_for_states, MLstates)
```

Arguments

possible_ranges_list_txt	A list of the allowed ranges/states
colors_list_for_states	The corresponding colors
MLstates	The ML states for the internal nodes

Value

MLcolors The colors for the ML states

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

optim

Examples

```
testval=1
```

---

readfiles_BioGeoBEARS_run

Read in the extra input files, if any

Description

This function reads input files for stratification, constraints, and detection, i.e., everything except the tree and geography files. E.g., areas_allowed_fn file is just a list of distance matrices, separated by blank lines, from youngest to oldest.

Usage

```
readfiles_BioGeoBEARS_run(inputs)
```

Arguments

inputs	The inputs list

Value

inputs The modified inputs list

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_BioGeoBEARS_run, read_times_fn, read_distances_fn, read_dispersal_multipliers_fn, read_area_of_areas_fn, read_areas_allowed_fn, read_detections, read_controls

Examples

```
test=1
```

---

read_areas_allowed_fn *Read in the area areas by time*

Description

areas_allowed file is just a list of 1/0 matrices, separated by blank lines, from youngest to oldest. 1s represent allowed combinations of areas

Usage

```
read_areas_allowed_fn(inputs = NULL,
  areas_allowed_fn = NULL)
```

Arguments

inputs	The inputs list
areas_allowed_fn	The areas-allowed filename.

Value

`list_of_areas_allowed_mats` A list object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve

Examples

```
test=1
```

---

read_area_of_areas_fn *Read in the area areas by time*

Description

area_areas file is just a list of distance matrices, separated by blank lines, from youngest to oldest.

Usage

```
read_area_of_areas_fn(inputs = NULL,
  area_of_areas_fn = NULL)
```

Arguments

`inputs`	The inputs list
`area_of_areas_fn`	The area-of-areas filename.

Value

`list_of_area_areas_mats` A list object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve

Examples

```
test=1
```

`read_controls` *Read a file with the total number of detections in a taphonomic control*

Description

This function reads in a tab-delimited text file containing counts of detections of the taphonomic controls in each region. These numbers should always be equal to or larger than the counts in the detections file.

Usage

```
read_controls(controls_fn, OTUnames = NULL,
  areanames = NULL, tmpskip = 0, phy = NULL)
```

Arguments

`controls_fn`	The filename of the file containing the counts of taphonomic control detections.
`OTUnames`	Default `NULL`, in which case the first column of the text file is used as row names/OTU names.
`areanames`	Default `NULL`, in which case the text file column headings are used.
`tmpskip`	How many lines should be skipped before reading the text file? Default `0`.
`phy`	An ape phylo object. If included, the rows will be sorted to match the order of tree tip labels.

Details

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function implements (a). Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

`dtf`

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

read_detections *Read a file with detection counts per area*

Description

This function reads in a tab-delimited text file containing counts of detections of each OTU in each region. These could be from database records or some other source.

Usage

```
read_detections(detects_fn, OTUnames = NULL,
  areanames = NULL, tmpskip = 0, phy = NULL)
```

Arguments

detects_fn	The filename of the detections file.
OTUnames	Default NULL, in which case the first column of the text file is used as row names/OTU names.
areanames	Default NULL, in which case the text file column headings are used.
tmpskip	How many lines should be skipped before reading the text file? Default 0.
phy	An ape phylo object. If included, the rows will be sorted to match the order of tree tip labels.

Value

dtf

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

`read_dispersal_multipliers_fn` *Read in the hard-coded dispersal multipliers from file*

Description

dispersal_multipliers file is just a list of distance matrices, separated by blank lines, from youngest to oldest

Usage

```
read_dispersal_multipliers_fn(inputs = NULL,
  dispersal_multipliers_fn = NULL)
```

Arguments

`inputs`	The inputs list
`dispersal_multipliers_fn`	The dispersal multipliers filename.

Value

`list_of_dispersal_multipliers_mats` A list object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`convolve`

Examples

```
test=1
```

`read_distances_fn` *Read in the distances by time*

Description

Distances file is just a list of distance matrices, separated by blank lines, from youngest to oldest.

Usage

```
read_distances_fn(inputs = NULL, distsfn = NULL)
```

Arguments

`inputs`	The inputs list
`distsfn`	The distances filename.

Value

`list_of_distances_mats` A list object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

`convolve`

Examples

```
test=1
```

read_PHYLIP_data *Read a PHYLIP-format file*

Description

This assumes data are interleaved, and that names are separated from data by a tab character; there is no 10-character limit on names.

Usage

```
read_PHYLIP_data(lgdata_fn = "lagrange_area_data_file.data",
  regionnames = NULL)
```

Arguments

lgdata_fn	The filename to read.
regionnames	A list of the names of the areas. Only used if the names are NOT specified in the file.

Details

This function is a precursor to getranges_from_LagrangePHYLIP.

Value

tmpdf A data.frame containing the data.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

getranges_from_LagrangePHYLIP

Examples

```
testval=1

# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code: extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
# Set the filename (Hawaiian Psychotria from Ree & Smith 2008)
fn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))

# Read in the file
tmpdf = read_PHYLIP_data(lgdata_fn=fn, regionnames=NULL)
tmpdf

# Read in the file
tmpdf = read_PHYLIP_data(lgdata_fn=fn,
regionnames=c("Kauai", "Oahu", "Maui-Nui","Big Island"))
tmpdf # Note that regionnames are only
# used if they are NOT specified in the file.
# But, you could put them on manually
names(tmpdf) = c("Kauai", "Oahu", "Maui-Nui","Big Island")
tmpdf

# This one has no area names
fn = np(paste(extdata_dir, "/Psychotria_geog_noAreaNames.data", sep=""))
tmpdf = read_PHYLIP_data(lgdata_fn=fn,
regionnames=c("Kauai", "Oahu", "Maui-Nui","Big Island"))
tmpdf # Note that regionnames are only
# used if they are NOT specified in the file.
```

---

read_times_fn *Read in the stratification time breakpoints*

Description

The timeperiods file is just a list of times, 1 per line, from youngest to oldest.

Usage

```
read_times_fn(inputs = NULL, timesfn = NULL)
```

Arguments

inputs	The inputs list
timesfn	The times filename.

Value

timeperiods A list object

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

convolve

Examples

```
test=1
```

---

relative_probabilities_of_subsets

Calculate probability of different descendant rangesizes, for the smaller descendant, in subset speciation

---

Description

"Rangesize" here means "number of areas in a geographic range". The LAGRANGE cladogenesis model requires that, during cladogenesis events, one daughter lineage will ALWAYS have a geographic range of size 1. This is argued for in *Ree et al. (2008)* on the grounds that new species usually get isolated and start in a new area. This is a reasonable proposition, but still, it would be nice to test the assumption. In addition, it could be that some speciation modes, especially vicariance, obey different rules. E.g., DIVA (*Ronquist (1996)*, *Ronquist (1997)*) allows vicariant speciation to divide up the ancestral range in every possible way (e.g., ABCD–>AB|CD, or AC|BD, or A|BCD, or D|ABC, etc.), but LAGRANGE would only allow vicariance to split off areas of size 1: (ABCD–>A|BCD, B|ACD, etc.) (Ronquist_Sanmartin_2011).

Usage

```
relative_probabilities_of_subsets(max_numareas = 6,
  maxent_constraint_01 = 0.5, NA_val = NA)
```

Arguments

max_numareas	The maximum number of areas possible allowed for the smaller-ranged-daughter in this type of cladogenesis/speciation.
maxent_constraint_01	The parameter describing the probability distribution on descendant rangesizes for the smaller descendant. See above.
NA_val	The output matrix consists of ancestral rangesizes and rangesizes of the smaller descendant. Some values are disallowed – e.g. descendant ranges larger than the ancestor; or, in subset speciation, descendant ranges the same size as the ancestor are disallowed. All disallowed descendant rangesizes get NA_val.

Details

To test different models, the user has to have control of the relative probability of different descendant rangesizes. The probability of each descendant rangesize could be parameterized individually, but we have a limited amount of observational data (essentially one character), so efficient parameterizations should be sought.

One way to do this is with the Maximum Entropy (*Harte (2011)*) discrete probability distribution of a number of ordered states. Normally this is applied (in examples) to the problem of estimation of the relative probability of the different faces of a 6-sided die. The input "knowledge" is the true mean of the dice rolls. If the mean value is 3.5, then each face of the die will have probability 1/6. If the mean value is close to 1, then the die is severely skewed such that the probability of rolling 1 is 99 other die rolls is very small. If the mean value is close to 6, then the probability distribution is skewed towards higher numbers.

Here in BioGeoBEARS, we use the same Maximum Entropy function to specify the relative probability of geographic ranges of a number of different rangesizes. This is merely used so that a single parameter can control the probability distribution – there is no MaxEnt estimation going on here. The user specifies a value for the parameter maxent_constraint_01 between 0.0001 and 0.9999. This can then be applied to all of the different ancestor-descendant range combinations in the cladogenesis/speciation matrix.

Example values of maxent_constraint_01 would give the following results:

maxent_constraint_01 = 0.0001 – The smaller descendant has rangesize 1 with 100 LAGRANGE)
maxent_constraint_01 = 0.5 – The smaller descendant can be any rangesize equal probability. This is effectively what happens in DIVA's version of vicariance speciation
maxent_constraint_01 = 0.9999 – The smaller descendant will take the largest possible rangesize for a given type of speciation, and a given ancestral rangesize. E.g., for sympatric/range-copying speciation (the ancestor is simply copied to both descendants, as in a continuous-time model with no cladogenesis effect), an ancestor of size 3 would product two descendant lineages of size 3. Such a model is implemented in the program BayArea (*Landis et al. (2013)*). LAGRANGE, on the other hand, would only allow range-copying for ancestral ranges of size 1.

Note: In LAGRANGE-type models, at speciation/cladogenesis events, one descendant daughter branch ALWAYS has size 1, whereas the other descendant daughter branch either (a) is the same (in sympatric/range-copying speciation), (b) inherits the complete ancestral range (in sympatric/subset speciation) or (c) inherits the remainder of the range (in vicariant/range-division speciation). LAGRANGE-type behavior (the smaller descendant has rangesize 1 with 100 rangesize) can be achieved by setting the maxent_constraint_01 parameter to 0.0001.

See also: Maximum Entropy probability distribution for discrete variable with given mean (and discrete uniform flat prior) http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Currently, the function maxent from the FD package is used to get the discrete probability distribution, given the number of states and the maxent_constraint_01 parameter. This could also be done with get_probvals, which uses calcZ_part, calcP_n, following equations 6.3-6.4 of *Harte (2011)*, although this is not yet implemented.

Value

relative_probabilities_of_vicariants, relprob_subsets_matrix, a numeric matrix giving the relative probability of each rangesize for the smaller descendant of an ancestral range, conditional on the ancestral rangesize.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

ReeSmith2008

Ronquist1996_DIVA

Ronquist_1997_DIVA

Harte2011

Landis_Matzke_etal_2013_BayArea

Matzke_2012_IBS

Ronquist_Sanmartin_2011

See Also

symbolic_to_relprob_matrix_sp, get_probvals, maxent, calcZ_part, calcP_n

Examples

```
testval=1
# Examples

# Probabilities of different descendant rangesizes, for the smaller
# descendant, under sympatric/subset speciation
# (plus sympatric/range-copying, which is folded in):
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.0001,
NA_val=NA)
```

```
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.5,
NA_val=NA)
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.9999,
NA_val=NA)

# Probabilities of different descendant rangesizes, for the smaller descendant,
# under vicariant speciation
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.0001,
NA_val=NA)
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.5,
NA_val=NA)
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.9999,
NA_val=NA)
```

`relative_probabilities_of_vicariants` *Calculate probability of different descendant rangesizes, for the smaller descendant, in vicariant speciation*

Description

"Rangesize" here means "number of areas in a geographic range". The LAGRANGE cladogenesis model requires that, during cladogenesis events, one daughter lineage will ALWAYS have a geographic range of size 1. This is argued for in *Ree et al. (2008)* on the grounds that new species usually get isolated and start in a new area. This is a reasonable proposition, but still, it would be nice to test the assumption. In addition, it could be that some speciation modes, especially vicariance, obey different rules. E.g., DIVA (*Ronquist (1996)*, *Ronquist (1997)*) allows vicariant speciation to divide up the ancestral range in every possible way (e.g., ABCD–>AB|CD, or AC|BD, or A|BCD, or D|ABC, etc.), but LAGRANGE would only allow vicariance to split off areas of size 1: (ABCD–>A|BCD, B|ACD, etc.) (Ronquist_Sanmartin_2011).

Usage

```
relative_probabilities_of_vicariants(max_numareas = 6,
  maxent_constraint_01v = 1e-04, NA_val = NA)
```

Arguments

`max_numareas`	The maximum number of areas possible allowed for the smaller-ranged-daughter in this type of cladogenesis/speciation.
`maxent_constraint_01v`	The parameter describing the probability distribution on descendant rangesizes for the smaller descendant, in a vicariance event (where the maximum size of the smaller range is numareas/2, rounded down). See above.
`NA_val`	The output matrix consists of ancestral rangesizes and rangesizes of the smaller descendant. Some values are disallowed – e.g. descendant ranges larger than the ancestor; or, in subset speciation, descendant ranges the same size as the ancestor are disallowed. All disallowed descendant rangesizes get `NA_val`.

Details

To test different models, the user has to have control of the relative probability of different descendant rangesizes. The probability of each descendant rangesize could be parameterized individually, but we have a limited amount of observational data (essentially one character), so efficient parameterizations should be sought.

One way to do this is with the Maximum Entropy (*Harte (2011)*) discrete probability distribution of a number of ordered states. Normally this is applied (in examples) to the problem of estimation of the relative probability of the different faces of a 6-sided die. The input "knowledge" is the true mean of the dice rolls. If the mean value is 3.5, then each face of the die will have probability 1/6. If the mean value is close to 1, then the die is severely skewed such that the probability of rolling 1 is 99 other die rolls is very small. If the mean value is close to 6, then the probability distribution is skewed towards higher numbers.

Here in `BioGeoBEARS`, we use the same Maximum Entropy function to specify the relative probability of geographic ranges of a number of different rangesizes. This is merely used so that a single parameter can control the probability distribution – there is no MaxEnt estimation going on here. The user specifies a value for the parameter `maxent_constraint_01` between 0.0001 and 0.9999. This can then be applied to all of the different ancestor-descendant range combinations in the cladogenesis/speciation matrix.

Example values of `maxent_constraint_01` would give the following results:

`maxent_constraint_01 = 0.0001` – The smaller descendant has rangesize 1 with 100 `LAGRANGE`)
`maxent_constraint_01 = 0.5` – The smaller descendant can be any rangesize equal probability. This is effectively what happens in `DIVA`'s version of vicariance speciation
`maxent_constraint_01 = 0.9999` – The smaller descendant will take the largest possible rangesize for a given type of speciation, and a given ancestral rangesize. E.g., for sympatric/range-copying speciation (the ancestor is simply copied to both descendants, as in a continuous-time model with no cladogenesis effect), an ancestor of size 3 would product two descendant lineages of size 3. Such a model is implemented in the program `BayArea` (*Landis et al. (2013)*). `LAGRANGE`, on the other hand, would only allow range-copying for ancestral ranges of size 1.

Note: In `LAGRANGE`-type models, at speciation/cladogenesis events, one descendant daughter branch ALWAYS has size 1, whereas the other descendant daughter branch either (a) is the same (in sympatric/range-copying speciation), (b) inherits the complete ancestral range (in sympatric/subset speciation) or (c) inherits the remainder of the range (in vicariant/range-division speciation). LAGRANGE-type behavior (the smaller descendant has rangesize 1 with 100 rangesize) can be achieved by setting the `maxent_constraint_01` parameter to 0.0001.

`See also:` Maximum Entropy probability distribution for discrete variable with given mean (and discrete uniform flat prior) http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Currently, the function `maxent` from the `FD` package is used to get the discrete probability distribution, given the number of states and the `maxent_constraint_01` parameter. This could also be done with `get_probvals`, which uses `calcZ_part`, `calcP_n`, following equations 6.3-6.4 of *Harte (2011)*, although this is not yet implemented.

Value

`relprob_subsets_matrix`, a numeric matrix giving the relative probability of each rangesize for

the smaller descendant of an ancestral range, conditional on the ancestral rangesize.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

ReeSmith2008

Ronquist1996_DIVA

Ronquist_1997_DIVA

Harte2011

Landis_Matzke_etal_2013_BayArea

Matzke_2012_IBS

Ronquist_Sanmartin_2011

See Also

relative_probabilities_of_subsets, symbolic_to_relprob_matrix_sp, get_probvals, maxent, calcZ_part, calcP_n

Examples

```
testval=1
# Examples

# Probabilities of different descendant rangesizes, for the smaller descendant,
# under sympatric/subset speciation
# (plus sympatric/range-copying, which is folded in):
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.0001,
NA_val=NA)
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.5,
NA_val=NA)
relative_probabilities_of_subsets(max_numareas=6, maxent_constraint_01=0.9999,
NA_val=NA)

# Probabilities of different descendant rangesizes, for the smaller descendant,
# under vicariant speciation
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.0001,
NA_val=NA)
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.5,
NA_val=NA)
relative_probabilities_of_vicariants(max_numareas=6, maxent_constraint_01v=0.9999,
NA_val=NA)
```

`rel_likes_from_deltaAICs` *Calculate the relative likelihoods of the models, from the deltaAIC*

Description

Given deltaAIC (Akaike Information Criterion), the absolute difference between the best model (lowest AIC) and other models, calculate the relative likelihoods of the models.

Usage

```
rel_likes_from_deltaAICs(deltaAICs)
```

Arguments

`deltaAICs`	A vector of deltaAIC values.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`rel_likes_AIC` A vector of relative likelihoods.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_Akaike_weights_from_rel_likes, rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
deltaAICs = get_deltaAIC(AICvals)
deltaAICs

rel_likes_AIC = rel_likes_from_deltaAICs(deltaAICs)
rel_likes_AIC
```

`rel_likes_from_deltaAICs_pairwise` *Calculate the relative likelihoods of the models, from the deltaAICs, pairwise*

Description

Given deltaAIC (Akaike Information Criterion), the absolute difference between the best model (lowest AIC) and other models, calculate the relative likelihoods of the models.

Usage

```
rel_likes_from_deltaAICs_pairwise(deltaAICs_pairwise)
```

Arguments

`deltaAICs_pairwise` A vector of AIC values.

Details

See *Burnham et al. (2002)* and http://www.brianomeara.info/tutorials/aic for discussion of AIC and its uses.

Value

`rel_likes_AIC_pairwise` A `data.frame` of relative likelihoods for each row (column 1) and the reference model (column 2).

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://www.brianomeara.info/tutorials/aic

Burnham_Anderson_2002

Matzke_2012_IBS

See Also

get_Akaike_weights_from_rel_likes, rel_likes_from_deltaAICs, getAIC

Examples

```
test=1

AICvals = c(40, 50, 60)
deltaAICs = get_deltaAIC_pairwise_w_ref_model(AICvals, ref_model="best")
deltaAICs

rel_likes_AIC_pairwise = rel_likes_from_deltaAICs_pairwise(deltaAICs)
rel_likes_AIC_pairwise
```

---

remove_null_rowcols_from_mat

Remove rows or columns representing a null geographic range from a matrix

---

Description

This function removes rows or columns representing a null geographic range from a matrix.

Usage

```
remove_null_rowcols_from_mat(tmpmat, null_sym = "()")
```

Arguments

tmpmat	The matrix to check for null ranges. Function will only work if rows and columns have names, and one of the names matches null_sym.
null_sym	The character(s) denoting a null range.

Details

LAGRANGE (*Ree et al. (2008)*) and other models often assume that a null geographic range (the lineage inhabits no areas, i.e. is extinct) is a possible state. However, this is never a possible ancestral state (since an extinct lineage will never have descendants) so sometimes we must remove it.

Value

`tmpmat3` The revised matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

ReeSmith2008

Matzke_2012_IBS

See Also

areas_list_to_states_list_new, areas_list_to_states_list_old, make_relprob_matrix_de

Examples

```
testval=1
states_list = list("_", c("A"), c("B"), c("C"), c("A","B"),
c("B","C"), c("A","C"), c("A","B","C"))

states_list = areas_list_to_states_list_new(areas=c("A","B","C"),
include_null_range=TRUE, split_ABC=TRUE)
states_list

dedf = make_relprob_matrix_de(states_list=states_list,
split_ABC=FALSE, split="", remove_simultaneous_events=TRUE,
add_multiple_Ds=TRUE,
dispersal_multiplier_matrix=make_dispersal_multiplier_matrix(states_list=states_list))

spmat_noNulls = remove_null_rowcols_from_mat(tmpmat=dedf, null_sym="()")
spmat_noNulls

spmat_noNulls = remove_null_rowcols_from_mat(tmpmat=dedf, null_sym="_")
spmat_noNulls
```

return_items_not_NA *Remove NAs from a vector/list*

Description

Utility function. This function returns the non-NA values from a vector.

Usage

```
return_items_not_NA(x)
```

Arguments

x	The vector of items to check for being not NA.

Details

This is used by get_indices_where_list1_occurs_in_list2_noNA, which is used by get_indices_of_branches_under which is used by extend_tips_to_ultrametricize, which can be used by section_the_tree.

Value

y The surviving, non-NA cells of a vector.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, LETTERS, get_indices_where_list1_occurs_in_list2_noNA, get_indices_where_list1_occurs_in_list2, extend_tips_to_ultrametricize, section_the_tree

Examples

```
list1 = c("N", "I", NA, "C", "K")
return_items_not_NA(list1)
```

save_tipranges_to_LagrangePHYLIP *Save a tipranges object to a LAGRANGE PHYLIP-style file containing binary-encoded geographic ranges*

Description

Given some geographic range data for tips in the `tipranges` object, this function exports them to an ASCII text file in the Lagrange C++/PHYLIP format (*Smith et al. (2010)*). This file can then be read by `getranges_from_LagrangePHYLIP`.

Usage

```
save_tipranges_to_LagrangePHYLIP(tipranges_object,
  lgdata_fn = "lagrange_area_data_file.data",
  areanames = colnames(tipranges_object@df))
```

Arguments

`tipranges_object`	An object of class `tipranges`.
`lgdata_fn`	The LAGRANGE geographic data file to be output.
`areanames`	A list of the names of the areas.

Details

LAGRANGE C++ geographic range files are ASCII text files with the format:

```
19 4 (A B C D)
P_mariniana_Kokee2   1000
P_mariniana_Oahu 0100
P_mariniana_MauiNui 0010
P_hawaiiensis_Makaopuhi 0001
P_wawraeDL7428 1000
[...]
```

The first row specifies the number of taxa (here, 19), the number of areas (here, 4), and finally, the names/abbreviations of the areas. The rest of the rows give the taxon names, followed by a tab and then the presence/absence in each range with 1s/0s.

The file above is part of the geographic range data for the Hawaiian *Psychotria* dataset used by *Ree et al. (2008)*.

Value

`tipranges_object` An object of class `tipranges`

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

SmithRee2010_CPPversion

ReeSmith2008

Matzke_2012_IBS

See Also

define_tipranges_object, getranges_from_LagrangePHYLIP

Examples

```
testval=1

# Create an example tipranges object
tipranges = define_tipranges_object()

# See current directory
getwd()

## Not run:
# Save the file
# Set the filename
fn = "example_tipranges.data"
save_tipranges_to_LagrangePHYLIP(tipranges_object=tipranges, lgdata_fn=fn)

# Show the file
tmplines = scan(file=fn, what="character", sep="\n")
cat(tmplines, sep="\n")

# Again, with areanames
save_tipranges_to_LagrangePHYLIP(tipranges_object=tipranges,
lgdata_fn=fn, areanames=c("area1","area2","area3"))

# Show the file
tmplines = scan(file=fn, what="character", sep="\n")
cat(tmplines, sep="\n")

## End(Not run) # End dontrun
```

`section_the_tree` *Section a tree for stratified analysis*

Description

A utility function for stratified analysis. Sections the tree into a series of strata. Each stratum may have one or more subtrees (APE phylo3 objects, *WITH* root edges) and/or branch segments (which are just represented as numeric values, indicating the length of the sub-branch, i.e. the time-width of the stratum, if the branch crosses the whole stratum.

Usage

```
section_the_tree(inputs, make_master_table = FALSE,
  plot_pieces = TRUE, cut_fossils = TRUE,
  fossils_older_than = 0.6)
```

Arguments

`inputs`	The list of inputs for stratified analysis
`make_master_table`	If desired, make an `inputs$master_table` containing the correspondance between the original tree and the sectioned pieces.
`plot_pieces`	If `TRUE`, plot the tree chunks (but not isolated branch segments) as they are created.
`cut_fossils`	If `TRUE` (default), the program is stopped if there are fossils, i.e. tips older than 0.6 my (default). Users should use codedrop.tip or an external program to clip fossils out of the tree. PLEASE NOTE that several times I have experienced miserable long nights due, apparently, to `drop.tip` producing weird tree structures, resulting in weird Newick files, without me realizing it. The solution is usually to open the Newick file in something like `FigTree`, resort the branches, and save to a new Newick file. Fossils have now been implemented in stratified analysis; this was complicated, as it involves inserting new branches in chopped trees.
`fossils_older_than`	Tips that are older than `fossils_older_than` will be marked as `TRUE` in a column called `fossil`. This is not currently set to 0, because Newick files can have slight precision issues etc. that mean not all tips quite come to zero. You can attempt to fix this with `extend_tips_to_ultrametricize` (but make sure you do not inappropriately average in fossils!!).

Value

`inputs` with `inputs$tree_sections_list` added.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

prt, chainsaw2, drop.tip

Examples

```
test=1
```

`sfunc` *Extract the appropriate probability for a subset speciation event, given text code for rangesize of smaller descendant, and ancestor*

Description

Extract the appropriate probability for a subset speciation event, given text code for rangesize of smaller descendant, and ancestor

Usage

```
sfunc(charcell, relprob_subsets_matrix)
```

Arguments

`charcell`	The text in the cell, indicating the type of speciation/cladogenesis range inheritance event.
`relprob_subsets_matrix`	A numeric matrix describing the relative probability of each smaller daughter range, conditional on the ancestral rangesize.

Value

`prob_of_this_b`, a numeric value giving the relative probability of that descendent-ancestor rangesize pair.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Matzke_2012_IBS

Harte2011

ReeSmith2008

Ronquist1996_DIVA

Ronquist_1997_DIVA

Ronquist_Sanmartin_2011

Landis_Matzke_etal_2013_BayArea

See Also

yfunc, vfunc, relative_probabilities_of_subsets, symbolic_to_relprob_matrix_sp, get_probvals, maxent, calcZ_part, calcP_n

Examples

```
testval=1
# Examples

# Probabilities of different descendant rangesizes, for the smaller descendant,
# under sympatric/subset speciation
# (plus sympatric/range-copying, which is folded in):
relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
```

```
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)


# Probabilities of different descendant rangesizes, for the smaller descendant,
# under vicariant speciation
relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.0001, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.5, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
```

```
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.9999, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)
```

`simstates_to_probs_of_each_area` *Convert simulated states to probabilities of each area*

Description

Basically this function assigns probability 1 to occupied areas according to the simulated state for a node, and probability 0 for the other areas. These data – the simulated truth – can then be compared to the inferred probabilities of presence in each area, from `infprobs_to_probs_of_each_area`.

Usage

```
simstates_to_probs_of_each_area(simulated_states_by_node,
  states_list, relprobs_matrix)
```

Arguments

`simulated_states_by_node`	The simulated states by node (0-based indices).
`states_list`	A list of the possible states/geographic ranges, in 0-based index form.
`relprobs_matrix`	A relative probabilities matrix returned by `bears_2param_standard_fast` or a similar function. The user should specify WHICH matrix in the results_object – i.e., scaled conditional likelihoods on downpass or uppass, or actual marginal probabilities of ancestral states. (The latter is the main thing of interest.) This specification is done via e.g. `relprobs_matrix = results_object$relative_probs_of_each_stat`

Value

`area_probs` The probability of presence in each area.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

simulate_biogeog_history, infprobs_to_probs_of_each_area

Examples

```
testval=1
```

---

simulated_indexes_to_tipranges_file

Convert simulated Qmat 0-based indexes to a tipranges file

Description

This function takes simulated state indices (ranging from 0 to numstates-1, i.e. number of possible geographic ranges-1) and converts them to a C++-LAGRANGE-style PHYLIP geographic ranges file.

Usage

```
simulated_indexes_to_tipranges_file(simulated_states_by_node,
  areas_list, states_list, trfn,
  out_geogfn = "lagrange_area_data_file.data")
```

Arguments

simulated_states_by_node	The simulated states/geographic ranges, in 0-based index form, ordered as the tips & nodes are ordered in a pruningwise-ordered phylo object in APE.
areas_list	A list of the desired area names/abbreviations/letters.
states_list	A list of the possible states/geographic ranges, in 0-based index form.
trfn	The filename of the source Newick tree.
out_geogfn	The output filename.

Value

out_geogfn The output filename.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

SmithRee2010_CPPversion

See Also

define_tipranges_object, getareas_from_tipranges_object, simulated_indexes_to_tipranges_object

Examples

```
testval=1
```

---

simulated_indexes_to_tipranges_object

Convert simulated Qmat 0-based indexes to a tipranges object

Description

This function takes simulated state indices (ranging from 0 to numstates-1, i.e. number of possible geographic ranges-1) and converts them to a tipranges object. This can then be converted into a C++-LAGRANGE-style PHYLIP geographic ranges file.

Usage

```
simulated_indexes_to_tipranges_object(simulated_states_by_node,
  areas_list, states_list, trfn)
```

Arguments

simulated_states_by_node	The simulated states/geographic ranges, in 0-based index form, ordered as the tips & nodes are ordered in a pruningwise-ordered phylo object in APE.
areas_list	A list of the desired area names/abbreviations/letters.
states_list	A list of the possible states/geographic ranges, in 0-based index form.
trfn	The filename of the source Newick tree.

Value

tipranges_object An object of class tipranges.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

SmithRee2010_CPPversion

See Also

define_tipranges_object, getareas_from_tipranges_object, simulated_indexes_to_tipranges_file

Examples

```
testval=1
```

---

simulate_biogeog_history

Simulate a biogeographical history, given a transition matrix and cladogenesis model

---

Description

This function simulates a biogeographical history, given a Q transition matrix, a cladogenesis model giving the relative probability of different range inheritance scenarios, a phylogeny, and a 0-based index value deciding the starting state (which could be randomly generated according to a prior distribution of states).

Usage

```
simulate_biogeog_history(phy, Qmat, COO_probs_columnar,
  index_Qmat_0based_of_starting_state)
```

Arguments

phy	An R phylo object.
Qmat	A (square, dense) Q transition matrix. Using a sparse matrix would require writing another function.
COO_probs_columnar	A speciation/cladogenesis transition matrix, in COO-like form, as produced by rcpp_calc_anclikes_sp_COOweights_faster.
index_Qmat_0based_of_starting_state	An integer index value, between 0 and (numstates-1), which specifies what state will be the starting point for the simulation.

Value

simulated_states_by_node A numeric matrix, giving the 0-based index of the state at each node and tip in the simulated history. Getting a more detailed history would require a version of stochastic mapping (*Huelsenbeck et al. (2003)*, *Bollback (2005)*, *Bollback (2006)*), but customized for the nonreversible and cladogenic aspects of biogeographical range evolution models.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Huelsenbeck_etal_2003_stochastic_mapping

Bollback_2005

Bollback_2006_SIMMAP

Matzke_2012_IBS

See Also

rcpp_calc_anclikes_sp_COOweights_faster

Examples

```
testval=1
```

`size_species_matrix` *Calculate the dimensions of the cladogenesis/speciation matrix*

Description

This function calculates the dimensions of the cladogenesis/speciation matrix describing the transition probabilities between ancestral geographic ranges and descendant geographic range pairs on Left (L) and Right (R) branches.

Usage

```
size_species_matrix(states_list = default_states_list(),
  printwarn = 1)
```

Arguments

`states_list`	A list of states, where each state consists of a list of areas. A default example list is provided.
`printwarn`	If `printwarn>0` (`printwarn=1` by default), then print to screen a message describing the size of the cladogenesis matrix.

Details

Under a cladogenesis model of geographic range change, the model will give the conditional probability of each possible combination of geographic ranges on the Left (L) and Right (R) descendant branches, conditional on a particular ancestral state. A matrix representing these transitions will have `numstates` ancestral states, and `numstates*numstates` possible descendant pairs. Many of these will have 0 conditional probability under the model, but, for visualization or experimental purposes it can be useful to display them all.

However, because `numstates = 2^numareas` under default conditions, and the number of cells the processor has to consider (without optimization tricks) is `numstates^3`, this transition matrix can very quickly become cumbersome to explicitly calculate or display. `size_species_matrix` allows the user to check this ahead of time.

See `numstates_from_numareas` for the details of calculating `numstates`.

At various points in `BioGeoBEARS` code, the text and numeric versions of the cladogenesis matrix are named `spmat` and `spPmat`, respectively.

Value

`spmat_dimensions` The dimensions of the cladogenesis matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

make_relprob_matrix_de, make_spmat_row

Examples

```
testval=1
spmat_dimensions = size_species_matrix(
states_list=list("_", c("A"), c("B"), c("C"), c("A","B"),
c("B","C"), c("A","C"), c("A","B","C")), printwarn=1)
spmat_dimensions
```

---

slashslash *Remove double slash (slash a slash)*

Description

Shortcut for: gsub(pattern="//", replacement="/", x=tmpstr)

Usage

```
slashslash(tmpstr)
```

Arguments

tmpstr	a path that you want to remove double slashes from

Details

This function is useful for removing double slashes that can appear in full pathnames due to inconsistencies in trailing slashes in working directories etc.

Value

outstr a string of the fixed path

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

getwd, setwd, gsub

Examples

```
tmpstr = "/Library/Frameworks//R.framework/Versions/"

outstr = slashslash(tmpstr)
outstr
```

---

sourceall *Source all .R files in a directory, except "compile" and "package" files*

Description

Utility function.

Usage

```
sourceall(path = path, pattern = "\\.R", ...)
```

Arguments

path	The path to source
pattern	Default is .R
...	Additional arguments to source

Value

path The path that was sourced.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

source

Examples

```
test=1
```

`states_list_indexes_to_areastxt`
States (ranges) lists to txt string of the areas

Description

This is a utility function.

Usage

```
states_list_indexes_to_areastxt(states_list, areanames,
  counting_base = 0, concat = TRUE, sep = "")
```

Arguments

`states_list`	A list of states, where each state consists of a list of areas.
`areanames`	A list of areanamess.
`counting_base`	Does states_list start indexing areas from 0 (default) or 1?
`concat`	If TRUE (default), merge the areas in a state into a single string.
`sep`	Character to merge on, as in paste. Default "".

Value

`tiparea` A string.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

getname, order_tipranges_by_tree_tips, define_tipranges_object, save_tipranges_to_LagrangePHYLIP

Examples

```
test=1
```

---

strsplit2 *String splitting shortcut*

Description

strsplit returns the results inside a list, which is annoying. strsplit2 shortens the process.

Usage

```
strsplit2(x, ...)
```

Arguments

x	A string to split
...	Other arguments to strsplit. The argument split is *required*.

Value

out The output from inside the list.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

strsplit

Examples

```
test=1

# strsplit returns the results inside a list element
out = strsplit("ABC", split="")
out
# I.e....
out[[1]]

# If this is annoying/ugly in the code, use strsplit2:
out = strsplit2("ABC", split="")
out
```

strsplit_whitespace *Split strings on whitespace*

Description

This function splits strings on whitespace (spaces and tabs), so you don't have to remember the regexp/grep format codes.

Usage

```
strsplit_whitespace(tmpline)
```

Arguments

tmpline	A string containing text.

Value

list_of_strs

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

strsplit

Examples

```
tmpline = "Hello world see my tabs."
strsplit_whitespace(tmpline)
```

symbolic_cell_to_relprob_cell *Convert symbolic cell (a text equation) to relprob matrix (a numeric value).*

Description

This is a utility function for symbolic_to_P_matrix and symbolic_to_Q_matrix.

Usage

```
symbolic_cell_to_relprob_cell(charcell, cellsplit = "",
  mergesym = "*", d = 0.1, e = 0.01, ...)
```

Arguments

charcell	The text formula.
cellsplit	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).
mergesym	The symbol to merge the formulas with. Default "+".
d	The dispersal/range expansion rate. Default d=0.1.
e	The extinction/range contraction rate. Default e=0.01.
...	Additional arguments to pass to strsplit.

Details

This function can be used in sapply. It still will not be very fast compared to the calculations in cladoRcpp, but can be useful for demonstrative purposes.

Value

cellval The output cell value.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

symbolic_to_P_matrix

Examples

```
testval=1

charcell = "1*d+1*d"

# Right
cellval = symbolic_cell_to_relprob_cell(charcell, cellsplit="yadda",
mergesym="", d=0.1, e=0.01)
cellval

# Wrong
cellval = symbolic_cell_to_relprob_cell(charcell, cellsplit="\\+",
mergesym="*", d=0.1, e=0.01)
cellval

# Right
cellval = symbolic_cell_to_relprob_cell(charcell, cellsplit="\\+",
mergesym="+", d=0.1, e=0.01)
cellval
```

---

symbolic_cell_to_relprob_cell_sp

Convert symbolic cell (a text equation) to relprob cell (a numeric value) – speciation matrix version

---

Description

This does the equivalent of symbolic_to_P_matrix, but for a speciation/cladogenesis matrix.

Usage

```
symbolic_cell_to_relprob_cell_sp(charcell,
  cellsplit = "\\+", mergesym = "*", ys = 1, j = 0,
  v = 1,
  relprob_subsets_matrix = relative_probabilities_of_subsets(6, 1e-04),
  relprob_vicar_matrix = relative_probabilities_of_vicariants(6, 1e-04),
  ...)
```

Arguments

charcell	The text formula.
cellsplit	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).
mergesym	The symbol to merge the formulas with. Default "+".

ys Relative weight of fully sympatric speciation (range-copying) and sympatric "subset" speciation. Default s=1 mimics LAGRANGE model.

v Relative weight of vicariant speciation. Default v=1 mimics LAGRANGE model.

j Relative weight of "founder event speciation"/jump speciation. Default j=0 mimics LAGRANGE model.

relprob_subsets_matrix
A numeric matrix describing the relative probability of each smaller daughter range, conditional on the ancestral rangesize.

relprob_vicar_matrix
A numeric matrix describing the relative probability of each smaller daughter range, conditional on the ancestral rangesize.

... Additional arguments to pass to relative_probabilities_of_subsets and relative_probabilities_of_vicariants, and thence to strsplit.

Details

These are 1-event probability matrices, not instantaneous rate matrices.

This function can be used in sapply. It still will not be very fast compared to the calculations in cladoRcpp, but can be useful for demonstrative purposes.

Value

cellval The output cell value.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

symbolic_to_relprob_matrix_sp, make_relprob_matrix_de

Examples

```
testval=1

charcell = "y1"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=0, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
```

```
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "y1"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=1, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "j"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=0, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "j"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=1, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "v1_2"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=0, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "v1_2"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
 j=1, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "s1_2"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*", ys=1,
j=0, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(max_numareas=3,
maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(
max_numareas=3,
maxent_constraint_01v=0.0001))

charcell = "s1_2"
symbolic_cell_to_relprob_cell_sp(charcell, cellsplit="\\\\+", mergesym="*",
ys=1, j=1, v=1, relprob_subsets_matrix=relative_probabilities_of_subsets(
max_numareas=3, maxent_constraint_01=0.0001),
relprob_vicar_matrix=relative_probabilities_of_vicariants(
max_numareas=3,
maxent_constraint_01v=0.0001))
```

`symbolic_to_P_matrix` *Convert symbolic matrix to relprob matrix*

Description

This function takes a transition probability matrix (in text form) and converts to numeric form, given values for *d*, *e*, or other parameters in the text formulas.

Usage

```
symbolic_to_P_matrix(dedf, cellsplit = "\\+",
  mergesym = "+", diags_sum_to_1 = FALSE, d = 0.1,
  e = 0.01, ...)
```

Arguments

`dedf`	The transition matrix or dispersal-extinction data.frame (dedf), contains the actual text of the formulas by which the transition probability matrix would be calculated.
`cellsplit`	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).
`mergesym`	The symbol to merge the formulas with. Default "+".
`diags_sum_to_1`	Calculate the diagonals such that, when added to the sum of the off-diagonals in a row, the entire row sums to 1. This creates a transition probability matrix where each row sums to 1, i.e. each cell represents the conditional probability of the column state, given the ancestral row state. The diagonal values represent the probability of staying the same.
`d`	The dispersal/range expansion rate. Default `d=0.1`.
`e`	The extinction/range contraction rate. Default `e=0.01`.
`...`	Additional arguments to pass to `symbolic_cell_to_relprob_cell` via `sapply`, and thence to cell`strsplit`.

Details

This is not particularly fast, but good for illustrative purposes.

Value

`dedf_vals` The output `data.frame`, contains the numeric results of the formulas calculating the transition probability matrix.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

areas_list_to_states_list_new, areas_list_to_states_list_old, make_relprob_matrix_de

Examples

```
testval=1

states_list = list("_", c("A"), c("B"), c("C"), c("A","B"),
c("B","C"), c("A","C"), c("A","B","C"))

states_list = areas_list_to_states_list_new(areas=c("A","B","C"),
include_null_range=TRUE, split_ABC=TRUE)
states_list

dedf = make_relprob_matrix_de(states_list=states_list,
 split_ABC=FALSE, split="", remove_simultaneous_events=TRUE,
add_multiple_Ds=TRUE,
dispersal_multiplier_matrix=make_dispersal_multiplier_matrix(states_list=states_list))
dedf

# Defaults
Pmat = symbolic_to_P_matrix(dedf, cellsplit="\\\\+", mergesym="+",
diags_sum_to_1=FALSE, d=0.1, e=0.01)
Pmat

# Calculate diagonal
Pmat = symbolic_to_P_matrix(dedf, cellsplit="\\\\+", mergesym="+",
diags_sum_to_1=TRUE, d=0.1, e=0.01)
Pmat

# You don't have to split, if the formulas are directly parsable
Pmat = symbolic_to_P_matrix(dedf, cellsplit="yadda", mergesym="",
diags_sum_to_1=FALSE, d=0.1, e=0.01)
Pmat
```

---

symbolic_to_Q_matrix *Convert symbolic matrix to an instantaneous rate matrix (Q matrix)*

Description

This function takes a transition probability matrix (in text form) and converts it to an instantaneous rate matrix (Q matrix), given values for *d*, *e*, or other parameters in the text formulas.

Usage

```
symbolic_to_Q_matrix(dedf, cellsplit = "\\+",
  mergesym = "*", d = 0.1, e = 0.01, ...)
```

Arguments

dedf	The transition matrix or dispersal-extinction data.frame (dedf), contains the actual text of the formulas by which the transition probability matrix would be calculated.
cellsplit	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).
mergesym	The symbol to merge the formulas with. Default "+".
d	The dispersal/range expansion rate. Default d=0.1.
e	The extinction/range contraction rate. Default e=0.01.
...	Additional arguments to pass to symbolic_cell_to_relprob_cell via sapply, and thence to cellstrsplit.

Details

This is not particularly fast, but good for illustrative purposes.

Value

dedf_vals The output data.frame, contains the Q matrix

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

FosterIdiots

See Also

areas_list_to_states_list_new, areas_list_to_states_list_old, make_relprob_matrix_de

Examples

```
testval=1

states_list = list("_", c("A"), c("B"), c("C"), c("A","B"),
c("B","C"), c("A","C"), c("A","B","C"))

states_list = areas_list_to_states_list_new(areas=c("A","B","C"),
include_null_range=TRUE, split_ABC=TRUE)
states_list

dedf = make_relprob_matrix_de(states_list=states_list, split_ABC=FALSE,
split="", remove_simultaneous_events=TRUE, add_multiple_Ds=TRUE,
dispersal_multiplier_matrix=make_dispersal_multiplier_matrix(states_list=states_list))
dedf

# Right
Qmat = symbolic_to_Q_matrix(dedf, cellsplit="\\\\+", mergesym="+", d=0.1, e=0.01)
Qmat

# Wrong
Qmat = symbolic_to_Q_matrix(dedf, cellsplit="\\\\+", mergesym="*", d=0.1, e=0.01)
Qmat

# You don't have to split, if the formulas are directly parsable
Qmat = symbolic_to_Q_matrix(dedf, cellsplit="yadda", mergesym="", d=0.1, e=0.01)
Qmat
```

---

symbolic_to_Q_matrix_exper

Experimental version of `symbolic_to_Q_matrix_exper`*, including base frequencies*

Description

Still experimental.

Usage

```
symbolic_to_Q_matrix_exper(dedf, cellsplit = "\\+",
  mergesym = "*", d = 0.1, e = 0.01,
  basefreqs = rep(1, nrow(dedf))/nrow(dedf), ...)
```

Arguments

`dedf`	The transition matrix or dispersal-extinction data.frame (dedf), contains the actual text of the formulas by which the transition probability matrix would be calculated.
`cellsplit`	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).

mergesym	The symbol to merge the formulas with. Default "+".
d	The dispersal/range expansion rate. Default d=0.1.
e	The extinction/range contraction rate. Default e=0.01.
basefreqs	Base frequencies, i.e. the equilibrium probabilities of the different states; the meaning of such an idea is debatable in the context of a LAGRANGE-like model where the null range (extinct everywhere) is included in the matrix and is a nonreversible absorbing state. Default is rep(1,nrow(dedf))/nrow(dedf).
...	Additional arguments to pass to symbolic_cell_to_relprob_cell via sapply, and thence to cellstrsplit.

Details

This function takes a transition probability matrix (in text form) and converts it to an instantaneous rate matrix (Q matrix), given values for *d*, *e*, or other parameters in the text formulas.

This is not particularly fast, but good for illustrative purposes.

Value

dedf_vals The output data.frame, contains the Q matrix

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

FosterIdiots

See Also

areas_list_to_states_list_old, make_relprob_matrix_de areas_list_to_states_list_new

Examples

```
testval=1

states_list = list("_", c("A"), c("B"), c("C"), c("A","B"),
c("B","C"), c("A","C"), c("A","B","C"))

states_list = areas_list_to_states_list_new(areas=c("A","B","C"),
include_null_range=TRUE, split_ABC=TRUE)
states_list
```

```
dedf = make_relprob_matrix_de(states_list=states_list, split_ABC=FALSE,
split="", remove_simultaneous_events=TRUE, add_multiple_Ds=TRUE,
dispersal_multiplier_matrix=make_dispersal_multiplier_matrix(states_list=states_list))
dedf

# Right
Qmat = symbolic_to_Q_matrix_exper(dedf, cellsplit="\\\\+", mergesym="+", d=0.1, e=0.01)
Qmat

# Wrong
Qmat = symbolic_to_Q_matrix_exper(dedf, cellsplit="\\\\+", mergesym="*", d=0.1, e=0.01)
Qmat

# You don't have to split, if the formulas are directly parsable
Qmat = symbolic_to_Q_matrix_exper(dedf, cellsplit="yadda", mergesym="", d=0.1, e=0.01)
Qmat

# Compare to symbolic_to_Q_matrix
Qmat = symbolic_to_Q_matrix(dedf, cellsplit="yadda", mergesym="", d=0.1, e=0.01)
Qmat
```

`symbolic_to_relprob_matrix_sp` *Convert symbolic matrix (with text equations) to relprob matrix (numeric values) – speciation matrix version*

Description

This does the equivalent of `symbolic_to_P_matrix`, but for a speciation/cladogenesis matrix.

Usage

```
symbolic_to_relprob_matrix_sp(spmat, cellsplit = "\\+",
  mergesym = "*", ys = 1, j = 0, v = 1,
  maxent_constraint_01 = 1e-04,
  maxent_constraint_01v = 1e-04, max_numareas = 6, ...)
```

Arguments

`spmat`	The speciation/cladogenesis matrix, with text formula.
`cellsplit`	The symbol to split the formulas on. Default "\\+" (plus symbol, with escape code).
`mergesym`	The symbol to merge the formulas with. Default "+".
`ys`	Relative weight of fully sympatric speciation (range-copying) and sympatric "subset" speciation. Default `s=1` mimics LAGRANGE model.
`v`	Relative weight of vicariant speciation. Default `v=1` mimics LAGRANGE model.
`j`	Relative weight of "founder event speciation"/jump speciation. Default `j=0` mimics LAGRANGE model.

maxent_constraint_01
: Parameter which assigns relative probabilities to different descendants range sizes, for the smaller descendant. Values can range from 0.0001 to 1. If `maxent_constraint_01=0.0001`, then the smaller descendant has a range size of 1 with probability 1 (i.e., the `LAGRANGE` default). If `maxent_constraint_01=0.5`, then all range sizes are equally weighted. If `maxent_constraint_01=1`, then the largest possible smaller descendant gets probability 1. The reference to "maxent" derives from the fact that the maxent probability distribution on a multistate, ordered, discrete variable – e.g. a die roll – can be calculated given just the mean value. Here, the `maxent_constraint_01` parameter is multiplied by the (maximum rangesize + 1). Thus, when `maxent_constraint_01=0.5`, if there are 6 possible states, then the parameter becomes 3.5, which sets equal probabilities of all possible descendant ranges sizes, when range size can range from 1 to 6.

maxent_constraint_01v
: Works the same as `maxent_constraint_01`, but just for descendants of vicariant events.

max_numareas
: The maximum number of areas possible allowed for the smaller-ranged-daughter in either vicariant or sympatric types of cladogenesis/speciation.

...
: Additional arguments to pass to `relative_probabilities_of_subsets` and `relative_probabilities_of_vicariants`, and thence to `strsplit`.

Details

These are 1-event probability matrices, not instantaneous rate matrices.

This function uses `symbolic_cell_to_relprob_cell_sp` in an `sapply` call. It still will not be very fast compared to the calculations in `cladoRcpp`, but can be useful for demonstrative purposes.

Value

`cellval` The output cell value.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

See Also

`symbolic_cell_to_relprob_cell_sp`, `make_relprob_matrix_de`

Examples

```
testval=1
# Generate the text version of the speciation/cladogenesis probability matrix
# (actually a relative weights matrix
# until the rows are normalized so that each sums to 1).
spmat = make_relprob_matrix_bi(states_list=list("_", c("A"), c("B"), c("C"),
c("A","B"), c("B","C"), c("A","C"), c("A","B","C")), split_ABC=FALSE, splitval="",
code_for_overlapping_subsets=NA, printwarn=1)
spmat

# Look at the conditional probabilities generated by a variety of models
spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+", mergesym="*",
ys=1, j=0, v=1, maxent_constraint_01=0.0001, maxent_constraint_01v=0.0001,
max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=0.5, j=0, v=0.5, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=1, v=1, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=0.25, j=0.25, v=0.25, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=1, v=0, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=1, v=0, maxent_constraint_01=0.5,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat

spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=0, v=0, maxent_constraint_01=0.5,
maxent_constraint_01v=0.0001, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat
```

```
spPmat = symbolic_to_relprob_matrix_sp(spmat=spmat, cellsplit="\\\\+",
mergesym="*", ys=1, j=0, v=1, maxent_constraint_01=0.0001,
maxent_constraint_01v=0.5, max_numareas=3)
spPmat = adf(spPmat); names(spPmat) = names(spmat); rownames(spPmat) = rownames(spmat)
spPmat
```

`tiplikes_wDetectionModel` *Calculate probability of detection data for each OTU at each range in a list of states/geographic ranges*

Description

This function calculates P(data|range,dp), i.e. the probability of some detection and taphonomic control counts, given the true geographic range/state, and `dp`, a detection probability (and, optionally, a false detection probability, `fdp`).

Usage

```
tiplikes_wDetectionModel(states_list_0based_index,
  numareas = NULL, detects_df, controls_df,
  mean_frequency = 0.1, dp = 1, fdp = 0,
  null_range_gets_0_like = TRUE)
```

Arguments

`states_list_0based_index`	A states_list, 0-based, e.g. from `rcpp_areas_list_to_states_list`.
`numareas`	The number of areas being considered in the analysis. If `NULL` (default), this is calculated to be the maximum range length, or one plus the maximum 0-based index in any of the ranges.
`detects_df`	A matrix/data.frame of detection counts, as produced from the output from `read_detections`.
`controls_df`	A matrix/data.frame of detection counts, as produced from the output from `read_controls`.
`mean_frequency`	This is the proportion of samples from the taphonomic control group that will truly be from this OTU, GIVEN that the OTU is present. This could be estimated, but a decent first guess is (total # samples of OTU of interest / total # of samples in the taphonomic control group where the OTU is known to be present). All that is really needed is some reasonable value, such that more sampling without detection lowers the likelihood of the data on the hypothesis of true presence, and vice versa. This value can only be 1 when the number of detections = the number of taphonomic control detections, for every OTU and area. This is the implicit assumption in e.g. standard historical biogeography analyses in LAGRANGE or BioGeoBEARS.

dp — The detection probability. This is the per-sample probability that you will correctly detect the OTU in question, when you are looking at it. Default is 1, which is the implicit assumption in standard analyses.

fdp — The false detection probability. This is probability of falsely concluding a detection of the OTU of interest occurred, when in fact the specimen was of something else. The default is 0, which assumes zero error rate, i.e. the assumption being made in all historical biogeography analyses that do not take into account detection probability. These options are being included for completeness, but it may not be wise to try to infer `mean_frequency`, `dp` and `fdp` all at once due to identifiability issues (and estimation of fdp may take a very large amount of data). However, fixing some of these parameters to reasonable values can allow the user to effectively include beliefs about the uncertainty of the input data into the analysis, if desired.

`null_range_gets_0_like` — If `TRUE` (default), then the data is given zero probability on the hypothesis that the range is a null range (i.e., no areas occupied). This is equivalent to saying that you are sure/are willing to assume that the OTU exists somewhere in your study area, at the timepoint being considered. Null ranges are identified by length=1, containing NULL, NA, "", "_", etc.

Details

This function performs the operation for all states/ranges for all tips.

The idea of taphonomic controls dates back at least to work of Bottjer & Jablonski (1988). The basic idea is that if you have taxa of roughly similar detectability, then detections of other taxa give some idea of overall detection effort. Obviously this is a very simple model that can be criticized in any number of ways (different alpha diversity in each region, different detectability of individual taxa, etc.), but it is a useful starting point as there has been no implementation of any detection model in historical/phylogenetic biogeography to date.

One could imagine (a) every OTU and area has a different count of detections and taphonomic control detections, or (b) the taphonomic control detections are specified by area, and shared across all OTUs. Situation (b) is likely more common, but this function assumes (a) as this is the more thorough case. Behavior (b) could be reproduced by summing each column, and/or copying this sum to all cells for a particular area.

Value

`tip_condlikes_of_data_on_each_state` The (non-logged!) likelihood of the data for each tip, given each possible range, and the detection model parameters.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

Bottjer_Jablonski_1988

See Also

Pdata_given_rangerow, calc_obs_like, mapply, read_detections, read_controls

Examples

```
testval=1

# soft-coded input files
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
detects_fn = np(paste(extdata_dir, "/Psychotria_detections_v1.txt", sep=""))
controls_fn = np(paste(extdata_dir, "/Psychotria_controls_v1.txt", sep=""))

detects_df = read_detections(detects_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)
controls_df = read_controls(controls_fn, OTUnames=NULL, areanames=NULL, tmpskip=0)

# Calculate the likelihood of the data at each tip, for each possible geographic range
numareas = 4
tmpranges = list(c(0), c(1), c(0,1))

mean_frequency=0.1
dp=1
fdp=0

tip_condlikes_of_data_on_each_state =
tiplikes_wDetectionModel(states_list_0based_index=tmpranges, numareas=numareas,
detects_df, controls_df, mean_frequency=mean_frequency, dp=dp, fdp=fdp,
null_range_gets_0_like=TRUE)

tip_condlikes_of_data_on_each_state
```

---

tipranges *The tipranges class*

Description

This class holds geographic range data for each tip in a phylogeny.

Details

Geographic range data can be read into a tipranges class object with BioGeoBEARS functions, e.g. `define_tipranges_object` or `getareas_from_tipranges_object`.

Class tipranges is an extension of the data.frame class. It is used for holding discrete geographic range data for the tips on a phylogeny. Geographic ranges are represented with bit encoding (0/1) indicating absence or presence in each possible area.

This is just a data.frame with: rows = taxanames
columns = area names
cells = 0/1 representing empty/occupied

Slots

df: Data.frame of class "numeric", containing data from df

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

define_tipranges_object, getareas_from_tipranges_object, areas_list_to_states_list_old, areas_list_to_states_list_new, tipranges_to_tip_condlikes_of_data_on_each_state

Examples

```
tipranges_object = define_tipranges_object()
tipranges_object
```

---

tipranges_to_area_strings

Convert tipranges binary coding to range strings

Description

This function converts the 0110-type format of the tipranges object into a list of strings describing the geographic ranges. E.g., 1100 becomes AB, 0111 become BCD (assuming the regions are abbreviated A, B, C...). Users can input their preferred abbreviations with areaabbr.

Usage

```
tipranges_to_area_strings(tipranges, areaabbr = NULL)
```

Arguments

tipranges	An object of class tipranges.
areaabbr	A vector of the abbreviations (preferably 1 character each).

Details

Note that you will HAVE to use order_tipranges_by_tree_tips on the tipranges object first, to make sure the tipranges are in the correct order on the tree tips.

Value

tiprange_names A vector of strings.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

ReeSmith2008

SmithRee2010_CPPversion

See Also

order_tipranges_by_tree_tips, define_tipranges_object, save_tipranges_to_LagrangePHYLIP

Examples

```
# Get the example files directory
extdata_dir = np(system.file("extdata", package="BioGeoBEARS"))
# tmp hard code:
# extdata_dir = "/Dropbox/_njm/__packages/BioGeoBEARS_setup/inst/extdata/"
# Set the filename (Hawaiian Psychotria from Ree & Smith 2008)

trfn = np(paste(extdata_dir, "/Psychotria_5.2.newick", sep=""))
tr = read.tree(trfn)

fn = np(paste(extdata_dir, "/Psychotria_geog.data", sep=""))
tipranges1 = getranges_from_LagrangePHYLIP(lgdata_fn=fn)
tipranges1
tipranges_to_area_strings(tipranges=tipranges1, areaabbr=NULL)
tipranges_to_area_strings(tipranges=tipranges1, areaabbr=c("K", "O", "M", "H"))
```

```
# Reorder the tipranges object
tipranges2 = order_tipranges_by_tree_tips(tipranges1, tr)
tipranges2
tipranges_to_area_strings(tipranges=tipranges2, areaabbr=NULL)
tipranges_to_area_strings(tipranges=tipranges2, areaabbr=c("K", "O", "M", "H"))
```

`tipranges_to_tip_condlikes_of_data_on_each_state`
Convert a tipranges object to the tip likelihoods

Description

This function takes a tipranges object, and converts it to tip likelihoods for input into the likelihood calculations of `calc_loglike_sp`.

Usage

```
tipranges_to_tip_condlikes_of_data_on_each_state(tipranges,
  phy, states_list = NULL,
  maxareas = length(getareas_from_tipranges_object(tipranges)))
```

Arguments

`tipranges`	An object of class `tipranges`.
`phy`	A phylogenetic tree (`ape` object of class `phylo`)
`states_list`	A complete list of the different states, of class `list` form
`maxareas`	The maximum number of areas in a geographic range, if the user does

Details

This (like LAGRANGE (*Ree et al. (2008)*) and every other available program) assumes that the geographic ranges at the tips are known with certainty. Reality may be different, particularly for sparsely-studied, scarce, or fossil taxa. In such a case, a detection model is needed to specify the likelihood of the observation data under each possible geographic range at the tips.

Note that data likelihoods under this or that hypothesis are not the same thing as probabilities. E.g., with DNA, if sequencing machine says that the base could be either A or C, but not G or T, then the likelihood of the data for that nucleotide position for that species would be 1 1 0 0, not 0.5 0.5 0 0. See Felsenstein (2004), p. 255, for more.

Value

`tip_condlikes_of_data_on_each_state` For each tip/row, likelihood of that tip's data under each possible true geographic range (columns)

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

ReeSmith2008

Matzke_2012_IBS

Felsenstein2004

See Also

define_tipranges_object, getareas_from_tipranges_object, areas_list_to_states_list_new, areas_list_to_states_list_old, binary_ranges_to_letter_codes

Examples

```
testval=1
# Define a tipranges object
tipranges_object = define_tipranges_object()
tipranges_object

areanames = getareas_from_tipranges_object(tipranges_object)
areanames

# Specify phylogeny to go with default tipranges object
newick_str = "((tip1:1,tip2:1):1,tip3:2):1;"
phy = read.tree(file="", text=newick_str)

# Here, we will assume the maximum range size is all areas, but it could be smaller
maxareas = length(areanames)
## Not run:
states_list = areas_list_to_states_list_old(areas=areanames, include_null_range=TRUE,
maxareas=maxareas)
states_list

## End(Not run)

states_list = areas_list_to_states_list_new(areas=areanames, include_null_range=TRUE,
maxareas=maxareas)
states_list

tip_condlikes_of_data_on_each_state = tipranges_to_tip_condlikes_of_data_on_each_state(
tipranges=tipranges_object, phy=phy, states_list=states_list, maxareas=maxareas )
tip_condlikes_of_data_on_each_state
```

traverse_up *Traverse the tree from node up to the tips*

Description

This is a utility function for nodenums_bottom_up.

Usage

```
traverse_up(tr4, startnode, traverse_records)
```

Arguments

tr4	A tree object in phylo4 format.
startnode	The node number to start the uppass at.
traverse_records	A list of the nodes visited.

Value

traverse_records

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Matzke_2012_IBS

See Also

phylo4,

Examples

```
test=1
```

unlist_df *Unlist the columns in a data.frame*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This is a shortcut for `data.frame(lapply(df, function(x) unlist(x)))`.

Usage

```
unlist_df(df)
```

Arguments

df	matrix or other object transformable to data.frame

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

unlist_df2

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
unlist_df2(x)
```

unlist_df2 *Unlist the columns in a data.frame, with more checks*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This runs unlist and additional checks.

Usage

```
unlist_df2(df)
```

Arguments

df	matrix or other object transformable to data.frame

Value

outdf A matrix.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

unlist_df

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
unlist_df2(x)
```

---

unlist_df3 *Unlist the columns in a data.frame, with more checks and adf*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This runs unlist and additional checks, and forces conversion to a data.frame at the end.

Usage

```
unlist_df3(df)
```

Arguments

df	matrix or other object transformable to data.frame

Value

outdf data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

unlist_df

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
unlist_df3(x)
```

`unlist_df4` *Unlist the columns in a data.frame, with more checks, adf, and dfnums_to_numeric*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This runs `unlist` and additional checks, and forces conversion to a `data.frame` at the end. It also adds `dfnums_to_numeric` which should remove the problem of numbers columns being of class `character`.

Usage

```
unlist_df4(df, ...)
```

Arguments

`df`	matrix or other object transformable to data.frame
`...`	Additional options passed to `dfnums_to_numeric`.

Details

See especially `data.matrix` for a possibly simpler alternative.

Value

`outdf` data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`unlist_df`, `dfnums_to_numeric`, `cls.df`, `data.matrix`

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
cls.df(x)
unlist_df4(x)

x = matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2)
cls.df(x)
unlist_df4(x)

x = adf(matrix(c(1,2,3,4,5,"A"), nrow=3, ncol=2))
names(x) = c("A","B")
cls.df(x)
unlist_df4(x)
```

---

unlist_dtf_cols *Unlist the columns in a data.frame*

Description

Utility function. What it says.

Usage

```
unlist_dtf_cols(dtf, printflag = FALSE)
```

Arguments

dtf	Input data.frame
printflag	Print the results if TRUE.

Value

dtf The data.frame, hopefully without lists for columns

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster https://code.google.com/p/lagrange/

Matzke_2012_IBS

See Also

unlist

Examples

```
test=1
```

---

vfunc *Extract the appropriate probability for a vicariant speciation event, given text code for rangesize of smaller descendant, and ancestor*

Description

Extract the appropriate probability for a vicariant speciation event, given text code for rangesize of smaller descendant, and ancestor

Usage

```
vfunc(charcell, relprob_vicar_matrix)
```

Arguments

charcell	The text in the cell, indicating the type of speciation/cladogenesis range inheritance event.
relprob_vicar_matrix	A numeric matrix describing the relative probability of each smaller daughter range, conditional on the ancestral rangesize.

Value

prob_of_this_v, a numeric value giving the relative probability of that descendent-ancestor rangesize pair.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Matzke_2012_IBS

Harte2011

ReeSmith2008

Ronquist1996_DIVA

Ronquist_1997_DIVA

Ronquist_Sanmartin_2011

Landis_Matzke_etal_2013_BayArea

See Also

sfunc, vfunc, relative_probabilities_of_subsets, symbolic_to_relprob_matrix_sp, get_probvals, maxent, calcZ_part, calcP_n

Examples

```
testval=1
# Examples

# Probabilities of different descendant rangesizes, for the smaller descendant,
# under sympatric/subset speciation
# (plus sympatric/range-copying, which is folded in):
relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
```

```
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)


# Probabilities of different descendant rangesizes, for the smaller descendant,
# under vicariant speciation
relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.0001, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.5, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
```

```
maxent_constraint_01v=0.9999, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)
```

---

yfunc *Extract the appropriate probability for a sympatric/range-copying speciation event, given text code for rangesize of smaller descendant, and ancestor*

Description

Extract the appropriate probability for a sympatric/range-copying speciation event, given text code for rangesize of smaller descendant, and ancestor

Usage

```
yfunc(charcell, relprob_subsets_matrix)
```

Arguments

charcell	The text in the cell, indicating the type of speciation/cladogenesis range inheritance event.
relprob_subsets_matrix	A numeric matrix describing the relative probability of each smaller daughter range, conditional on the ancestral rangesize.

Value

prob_of_this_s, a numeric value giving the relative probability of that descendent-ancestor rangesize pair.

Note

Go BEARS!

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster http://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution

Matzke_2012_IBS

Harte2011

ReeSmith2008

Ronquist1996_DIVA

Ronquist_1997_DIVA

Ronquist_Sanmartin_2011

Landis_Matzke_etal_2013_BayArea

See Also

sfunc, vfunc, relative_probabilities_of_subsets, symbolic_to_relprob_matrix_sp, get_probvals, maxent, calcZ_part, calcP_n

Examples

```
testval=1
# Examples

# Probabilities of different descendant rangesizes, for the smaller
# descendant, under sympatric/subset speciation
# (plus sympatric/range-copying, which is folded in):
relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
sfunc(charcell="s1_1", relprob_subsets_matrix)
sfunc(charcell="s1_2", relprob_subsets_matrix)
sfunc(charcell="s1_3", relprob_subsets_matrix)
sfunc(charcell="s2_3", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
```

```
maxent_constraint_01=0.0001, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.5, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_subsets(max_numareas=6,
maxent_constraint_01=0.9999, NA_val=NA)
relprob_subsets_matrix
yfunc(charcell="y1", relprob_subsets_matrix)
yfunc(charcell="y2", relprob_subsets_matrix)
yfunc(charcell="y3", relprob_subsets_matrix)
yfunc(charcell="y4", relprob_subsets_matrix)


# Probabilities of different descendant rangesizes, for the smaller descendant,
# under vicariant speciation
relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.0001, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
maxent_constraint_01v=0.5, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)

relprob_subsets_matrix = relative_probabilities_of_vicariants(max_numareas=6,
```

```
maxent_constraint_01v=0.9999, NA_val=NA)
relprob_subsets_matrix
vfunc(charcell="v1_1", relprob_subsets_matrix)
vfunc(charcell="v1_2", relprob_subsets_matrix)
vfunc(charcell="v1_3", relprob_subsets_matrix)
vfunc(charcell="v1_4", relprob_subsets_matrix)
vfunc(charcell="v2_4", relprob_subsets_matrix)
vfunc(charcell="v2_2", relprob_subsets_matrix)
vfunc(charcell="v1_6", relprob_subsets_matrix)
vfunc(charcell="v2_6", relprob_subsets_matrix)
vfunc(charcell="v3_6", relprob_subsets_matrix)
```

Index