Package ‘distory’

April 19, 2020

Version 1.4.4
Date 2020-04-19
Title Distance Between Phylogenetic Histories
Author John Chakerian [aut, cre], Susan Holmes [aut, cre]
Maintainer Emmanuel Paradis <Emmanuel.Paradis@ird.fr>
Depends ape (>= 5.0)
Imports graphics, stats
License BSD_3_clause + file LICENSE
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-04-19 09:00:02 UTC

R topics documented:

  distory-package .................................................. 2
  bethe.tree ....................................................... 2
  bin.multiPhylo ................................................... 3
  dist.multiPhylo ................................................... 4
  gromov.hyperbolicity ........................................... 6
  mcmc.target.seq ............................................... 8
  orthant.boundary.tree ......................................... 10
  phylo.diff ....................................................... 11
  position.leverage ............................................. 13

Index 15
distory-package  Distance Between Phylogenetic Histories

Description

The distory package provides functions for computing geodesic distances between phylogenetic trees, as well as functions which use this distance. Methods for computing Gromov delta-hyperbolicity, Markov Chain Monte Carlo routines in tree space, and per-position leverage for DNA sequences are included.

Details

A description of the algorithm used for the distance computation can be found in dist.multiPhylo.

Author(s)

John Chakerian <chakj@stanford.edu> and Susan Holmes <susan@stat.stanford.edu>

References


bethe.tree  Bethe Tree

Description

Generates a Bethe tree with given tips, inner edge lengths, and outgroup.

Usage

bethe.tree(tips, level.lengths = NULL, outgroup="O", outgroup.dist=1)

Arguments

tips  A list of tip names as a character vector. Should be a power of 2. All tip names must be distinct.
level.lengths  Edge lengths for each level, counted from the bottom up. NULL means a default of 1. If the vector isn’t long enough, the last value will be repeated as necessary.
outgroup  The tip label for the outgroup.
outgroup.dist  The distance of the outgroup from the root.
bin.multiPhylo

Details
Generates a Bethe tree with specified internal edge lengths.

Value
A class of type phylo representing the tree.

Author(s)
John Chakerian

References

See Also
dist.multiPhylo

Examples
plot(bethe.tree(as.character(1:16), 1:4, "17", 14))

bin.multiPhylo

Description
Bins trees according to branching topology.

Usage
bin.multiPhylo(treelist)

Arguments
treelist A list of trees that can be passed to dist.phylo (see the help for dist.phylo for acceptable formats).

Details
Bins trees according to branching topology. Two trees are considered to have the same topology if the same set of partitions of tips are produced by the edges, which corresponds to the same branching up to rearrangement of tips.
dist.multiPhylo

Value

Returns a numeric vector of bin ids. Bin ids are assigned in order of the first tree in that bin, that is, the first k unique trees in the list passed will be assigned bins 1..k in order of appearance.

Author(s)

John Chakerian

References


See Also

dist.multiPhylo

dist.multiPhylo(combined.trees)

Examples

data(woodmouse)
mtree <- root(fastme.ols(dist.dna(woodmouse)), "No305", resolve.root=TRUE)
breps <- 500

trees <- boot.phylo(otree, woodmouse, B=breps, function(x)
    root(fastme.ols(dist.dna(x)), "No305", resolve.root=TRUE),trees=TRUE)
combined.trees <- c(list(otree), trees$trees)

bin.multiPhylo(combined.trees)

Description

Computes the geodesic distance of a list of phylogenetic trees using a polynomial algorithm.

Usage

dist.multiPhylo(x, method = "geodesic", force.multi2di = FALSE,
    outgroup = NULL, convert.multifurcating = FALSE,
    use.random.resolution = FALSE, scale = NULL,
    verbose = FALSE)
**Arguments**

- **x**: A list of ape trees (class 'phylo'). The list does not have to be of class 'multiPhylo'. The function will also accept a list of strings of trees in Newick format, or a single string with trees in Newick format separated by semicolons. All the trees must have the same tip labels.

- **method**: Determines which distance method is used. Options are 'geodesic' for the tree space geodesic distance, or 'edgeset' for the number of edges (defined by splits of tips) that are different.

- **force.multi2di**: Force conversion of every tree to strict bifurcating through the ape function 'multi2di', using the use.random.resolution as its parameter. This option should not be used in conjunction with specification of an outgroup.

- **outgroup**: Specifies an outgroup to root each tree with respect to. This calls the ape function 'root' on every tree in the list.

- **convert.multifurcating**: Setting this option will check every tree for multifurcations using the ape function 'is.binary.tree' - if it returns FALSE, the ape function 'multi2di' will be called on it. Note that this does not ensure a tree is strictly binary, since ape considers an unrooted tree binary even if the root node is trifurcating. This option can be used in conjunction with specification of an outgroup.

- **use.random.resolution**: Specifies the parameter to 'multi2di' if needed.

- **scale**: Specifies a scale to make all trees uniformly scaled (that is, the sum of all edges will be uniform).

- **verbose**: Turns on incremental status updates and more warnings. Helpful for large computations.

**Details**

This function computes the geodesic distance according to Billera et. al. using an algorithm based off of the polynomial time algorithm of Owen and Provan. Since it corresponds to a formal definition of tree-space as a space of strictly binary trees, no multifurcations are allowed, including on the root node. In addition, negative and 0-lengthed edges are clamped to a very small value (DBL_MIN) for technical reasons.

The Newick parser supports only a subset of the Newick format. In particular, it does not at the moment allow for internal node labels, only weights. Weights will be automatically set to 1 if not specified. It may be necessary to clean data in ape to make the trees conform to this.

**Value**

Returns a distance matrix of class 'dist' representing the pairwise geodesic distances between all input trees. Keep in mind this distance matrix is not Euclidean. N/A values are provided in the case of an error in determining the distance.
Author(s)

John Chakerian

References


See Also

dist.dna, boot.phylo, cmdscale

Examples

data(woodmouse)
oTree <- root(nj(dist.dna(woodmouse)), "No305", resolve.root=TRUE)
breps <- 250

trees <- boot.phylo(oTree, woodmouse, B=breps, function(x)
  root(nj(dist.dna(x)), "No305", resolve.root=TRUE), trees = TRUE)

combined.trees <- c(list(oTree), trees$trees)
tree.dists <- dist.multiPhylo(combined.trees)
mdres <- cmdscale(tree.dists, k=breps, add=TRUE)
plot(mdres$points[,1], mdres$points[,2], col = c("red", rep("black", breps)))
text(mdres$points[,1], mdres$points[,2], labels = 1:(breps + 1),
  cex = 0.7, adj = c(0, 2))

gromov.hyperbolicity  Gromov Hyperbolicity Constant

Description

Computes the Gromov Hyperbolicity Constant of a distance matrix.

Usage

gromov.hyperbolicity(d, deltas = FALSE, scale = NA)
Arguments

\(d\) A distance matrix of type \texttt{dist} or \texttt{matrix}, or anything that can be coerced into \texttt{dist} by \texttt{as.dist}. Must have at least 4 points.

\(\text{deltas}\) A logical value specifying whether to return the vector of delta values. Default is \texttt{FALSE}.

\(\text{scale}\) Specifies a scaling method for each delta. Default is no scaling (\texttt{NA} or "none"). Available methods are "max" which scales deltas by the max of the sums computed, and "perimeter" for the largest perimeter of the four points.

Details

This computes a constant that represents the relaxation of a 4-point condition for delta-hyperbolicity. See (Gromov 1987) for details.

Value

The Gromov hyperbolicity constant of the given distance matrix.

Author(s)

John Chakerian

References


See Also
dist.multiPhylo

Examples

# scale final delta by max distance
points <- cbind(runif(100), runif(100))
d <- dist(points)
gromov.hyperbolicity(d)/max(d)

# scale each delta by max distance for the 4 points
points <- cbind(runif(100), runif(100))
d <- dist(points)
gromov.hyperbolicity(d, scale="max")

# scale each delta by the max perimeter for the 4 points
points <- cbind(runif(100), runif(100))
d <- dist(points)
gromov.hyperbolicity(d, scale="max")
mcmc.target.seq  

Find MCMC Target Sequence

Description

mcmc.target.seq uses MCMC to find a configuration of DNA positions to get as close as possible to a given tree.

boot.samples.idxs bootstraps over indices into a DNA matrix.

lookup.samples goes from an index representation of a configuration of DNA to the actual DNAbin format.

convert.table.to.idx converts a table of counts for positions 1..n into a list of indices corresponding to positions (i.e. goes from the tabled form to a vector whose tabling matches the input).

Usage

mcmc.target.seq(data, x, F, n)

boot.samples.idxs(data, B = 100, block = 1)

lookup.samples(data, idxs)

convert.table.to.idx(T)

Arguments

data A DNA matrix in DNAbin format.

x A tree of class 'phylo' to estimate.

F A tree estimation function, accepting a DNA matrix in DNAbin format and returning a tree of class 'phylo.'

n The number of MCMC iterations to perform.

B The number of bootstrap replicates.

block The block size to use during bootstrapping.

idxs A list of numeric vectors of indices to use for lookup.

T A table or table-like vector to convert.

Details

mcmc.target.seq performs an MCMC with simulated annealing to locate a configuration of DNA positions from the original matrix that gets as close as possible to a target tree. Propositions for the MCMC replacing one character with another uniformly at random.

The remaining functions are intended to be used as support functions.
mcmc.target.seq returns a list of 4 elements: a numeric vector of counts of each position in the
original matrix, the best estimated tree, the best distance from the estimated tree to the target tree,
and a numeric vector of the distances for every iteration of the simulation.

boot.samples.idxs returns a numeric vector representing the bootstrapped indices.

lookup.samples returns a list of objects of class DNAbin corresponding to the DNA sequences
generated from indices into the original DNA matrix.

convert.table.to.idx returns a numeric vector of indices based on the table counts.

Author(s)

John Chakerian

References

Chakerian, J. and Holmes, S. P. Computational Tools for Evaluating Phylogenetic and Hierarchical

See Also

dist.multiPhylo, orthant.boundary.tree

Examples

## Not run:
## This example has been excluded from checks:
## copy/paste the code to try it

data(woodmouse)
otree <- root(fastme.ols(dist.dna(woodmouse)), "No305", resolve.root=TRUE)
breps <- 200
trees <- boot.phylo(otree, woodmouse, B=breps, function(x)
  root(fastme.ols(dist.dna(x)), "No305", resolve.root=TRUE),
trees = TRUE)
combined.trees <- c(list(otree), trees$trees)
binning <- bin.multiPhylo(combined.trees)
tree.a <- combined.trees[[match(1, binning)]]
i <- 2
max.bin <- max(binning)
tree.b <- combined.trees[[match(2, binning)]]
while(length(distinct.edges(tree.a,tree.b)) > 1 && i < max.bin)
{
  i = i + 1
  tree.b = combined.trees[[match(i, binning)]]
}
orthant.boundary.tree

Orthant Boundary Tree

Description

Produces a degenerate tree on the boundary between trees that differ by one split.

Usage

orthant.boundary.tree(x, y)

Arguments

x The tree in the first orthant.
y The tree in the second orthant.

Details

The tree found is the tree on the boundary between the two orthants such that it is on the straight line connecting the two trees when one orthant is thought of as being the (-,+), quadrant and the second orthant as being the (+,+), quadrant, where the (0,y) line is the particular boundary in question.

Value

Returns an object of class 'phylo' representing the boundary tree.

Author(s)

John Chakerian

References

phylo.diff

See Also

mcmc.target.seq

Examples

data(woodmouse)
otree <- root(fastme.ols(dist.dna(woodmouse)), "No305", resolve.root=TRUE)
breps <- 200

trees <- boot.phylo(otree, woodmouse, B=breps, function(x)
  root(fastme.ols(dist.dna(x)), "No305", resolve.root=TRUE),
  trees = TRUE)
combined.trees <- c(list(otree), trees$trees)
binning <- bin.multiPhylo(combined.trees)
tree.a <- combined.trees[[match(1, binning)]]
i <- 2
max.bin <- max(binning)
tree.b <- combined.trees[[match(2, binning)]]
while(length(distinct.edges(tree.a,tree.b)) > 1 && i < max.bin)
{
  i = i + 1
  tree.b = combined.trees[[match(i, binning)]]
}
plot(orthant.boundary.tree(tree.a, tree.b))
phylo.diff

Usage

phylo.diff(x, y, ...)
distinct.edges(x, y)
edge.from.split(x, split)
get.bipartition(x, e)
partition.leaves(x)

Arguments

x The first (or only) tree.
y The second tree, for the functions that accept two trees.
split A list of bipartitions, probably from partition.leaves.
e An edge for a particular tree, given as an id.
... Additional arguments to pass to the plot.phylo function.

Details

phylo.diff uses the ape tree plotting function. The other functions are mostly meant as support functions.

Value

phylo.diff returns invisible.
distinct.edges returns a numeric vector of edge ids for the first tree.
edge.from.split returns an edge id for a particular tree corresponding to a given bipartition and NA if none such edge exists.
get.bipartition returns a character vector of the tips below that edge in the given tree.
partition.leaves returns a list of partitions (themselves character vectors) of the given tree.

Author(s)

John Chakerian

References


See Also

dist.multiPhylo
Examples

```r
data(woodmouse)
mtree <- root(fastme.ols(dist.dna(woodmouse)), "No305", resolve.root=TRUE)
breps <- 10

trees <- boot.phylo(otree, woodmouse, B=breps, function(x)
root(fastme.ols(dist.dna(x)), "No305", resolve.root=TRUE),
trees = TRUE)

combined.trees <- c(list(otree), trees$trees)

binning <- bin.multiPhylo(combined.trees)

phylo.diff(combined.trees[[match(1, binning)]], combined.trees[[match(2, binning)]])
```

---

**position.leverage**

**Position Leverage**

**Description**

Provides a rough heuristic for determining the degree to which each position in the DNA matrix affects the resulting tree.

**Usage**

```r
position.leverage(data, F, to = NULL, rep = 50, by = 1)
```

**Arguments**

- `data`: A DNA matrix in DNAbin format.
- `F`: A tree estimation function, accepting a DNA matrix of class DNAbin and returning a tree of class phylo.
- `to`: The tree with which distances are measured in respect to, or NULL to indicate the tree estimated by `F` for the starting DNA matrix.
- `rep`: The number of times to replicate the position in question.
- `by`: The function will perform the calculation on every by-th position (that is, on seq(1, N, by)).

**Details**

This function takes a DNA matrix and, for every by-th position, replicates that position `rep` times, randomly removing `rep` other positions to keep all sequences the same length other positions to keep all sequences the same length. For each new DNA matrix created in this way, `F` is used to estimate the corresponding tree, and the distance to tree `to` is computed and stored. This distance can be thought of as somewhat analogous to the leverage of that position.
**Value**

Returns a numeric vector of distances from tree to for each position sampled.

**Author(s)**

John Chakerian

**References**


**See Also**

dist.multiPhylo

**Examples**

data(woodmouse)
f.est <- function(x) root(nj(dist.dna(x)), "No305", resolve.root = TRUE)
position.leverage(woodmouse, f.est, by = 10)
Index

*Topic **datagen**
  bethe.tree, 2

*Topic **manip**
  bin.multiPhylo, 3
  dist.multiPhylo, 4
  gromov.hyperbolicity, 6
  mcmc.target.seq, 8
  orthant.boundary.tree, 10
  phylo.diff, 11
  position.leverage, 13

*Topic **package**
  distory-package, 2

bethe.tree, 2
bin.multiPhylo, 3
boot.phylo, 6
boot.samples.idxs (mcmc.target.seq), 8
cmdscale, 6
convert.table.to.idx (mcmc.target.seq), 8
dist.dna, 6
dist.multiPhylo, 2–4, 4, 7, 9, 12, 14
distinct.edges (phylo.diff), 11
distory (distory-package), 2
distory-package, 2
edge.from.split (phylo.diff), 11
get.bipartition (phylo.diff), 11
gromov.hyperbolicity, 6
lookup.samples (mcmc.target.seq), 8
mcmc.target.seq, 8, 11
orthant.boundary.tree, 9, 10
partition.leaves (phylo.diff), 11
phylo.diff, 11
position.leverage, 13