Package 'treespace'

August 27, 2025

Title Statistical Exploration of Landscapes of Phylogenetic Trees

Date 2025-08-26

Version 1.1.4.4

Description Tools for the exploration of distributions of phylogenetic trees.
This package includes a 'shiny' interface which can be started from R using
treespaceServer().
For further details see Jombart et al. (2017) <DOI:10.1111/1755-0998.12676>.

Depends R (>= 3.4.0), ape, ade4

Imports adegenet, adegraphics, combinat, compiler, distory, fields,
htmlwidgets, MASS, methods, parallel, phangorn, phytools, Rcpp,
rgl, RLumShiny, scatterD3, shiny, shinyBS, utils

LinkingTo Rcpp

Suggests ggplot2, igraph, knitr, pander, RColorBrewer, reshape2,
rmarkdown, sf, testthat

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Collate RcppExports.R metrics.R medTree.R treespace.R findGroves.R
plotGroves.R plotTreeDiff.R servers.R transmissionTrees.R
data.R makeCollapsedTree.R relatedTreeDist.R simulateIndTree.R
tipsMRCAdepths.R treeConcordance.R

RoxygenNote 7.3.2

VignetteBuilder knitr

URL https://cran.r-project.org/package=treespace,
https://github.com/thibautjombart/treespace

BugReports https://github.com/thibautjombart/treespace/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-08-27 16:30:02 UTC

Author Thibaut Jombart [aut] (ORCID: <https://orcid.org/0000-0003-2226-8692>),
Michelle Kendall [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7344-7071>),
Jacob Almagro-Garcia [aut] (ORCID:
<https://orcid.org/0000-0002-0595-7333>),
Caroline Colijn [aut] (ORCID: <https://orcid.org/0000-0001-6097-6708>)

Maintainer Michelle Kendall <michelle.kendall@ndm.ox.ac.uk>

Contents

.render.server.info . . . 2
DengueBEASTMCC . . . 3
DengueSeqs . . . 3
DengueTrees . . . 4
findGroves . . . 5
findMRCIs . . . 6
fluTrees . . . 7
linearMrca . . . 8
makeCollapsedTree . . . 8
medTree . . . 9
multiDist . . . 12
plotGroves . . . 13
plotGrovesD3 . . . 15
plotTreeDiff . . . 17
refTreeDist . . . 19
relatedTreeDist . . . 20
simulateIndTree . . . 21
tipDiff . . . 22
tipsMRCAdepths . . . 23
treeConcordance . . . 24
treeDist . . . 25
treespace . . . 26
treespaceServer . . . 28
treeVec . . . 29
wiwMedTree . . . 30
wiwTreeDist . . . 31
woodmiceTrees . . . 32

Index **33**

`.render.server.info` *Auxiliary functions*

Description

These functions are not supposed to be used by the user.

Usage

```
.render.server.info()
```

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

DengueBEASTMCC *Dengue fever BEAST MCC tree*

Description

The maximum clade credibility (MCC) tree from DengueTrees

Format

A phylo object

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Source

http://bmcevolbiol.biomedcentral.com/articles/10.1186/1471-2148-7-214

References

Drummond, A. J., and Rambaut, A. (2007) BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7(1), 214.

Lanciotti, R. S., Gubler, D. J., and Trent, D. W. (1997) Molecular evolution and phylogeny of dengue-4 viruses. *Journal of General Virology*, 78(9), 2279-2286.

DengueSeqs *Dengue fever sequences*

Description

17 dengue virus serotype 4 sequences from Lanciotti et al. (1997)

Format

A DNAbin object containing 17 DNA sequences, each of length 1485.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Source

http://bmcevolbiol.biomedcentral.com/articles/10.1186/1471-2148-7-214

References

Lanciotti, R. S., Gubler, D. J., and Trent, D. W. (1997) Molecular evolution and phylogeny of dengue-4 viruses. *Journal of General Virology*, 78(9), 2279-2286.

`DengueTrees` *BEAST analysis of Dengue fever*

Description

These trees were created using one of the `xml` files provided with the original BEAST paper by Drummond and Rambaut (2007). They provide an example of 17 dengue virus serotype 4 sequences from Lanciotti et al. (1997) (available as `DengueSeqs`) and `xml` files with varying priors for model and clock rate. Here we include a random sample of 500 of the trees (from the second half of the posterior) produced using BEAST v1.8 with the standard GTR + Gamma + I substitution model with uncorrelated lognormal-distributed relaxed molecular clock (file 4).

Format

A multiPhylo object containing 500 trees, each with 17 tips

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Source

http://bmcevolbiol.biomedcentral.com/articles/10.1186/1471-2148-7-214

References

Drummond, A. J., and Rambaut, A. (2007) BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7(1), 214.

Lanciotti, R. S., Gubler, D. J., and Trent, D. W. (1997) Molecular evolution and phylogeny of dengue-4 viruses. *Journal of General Virology*, 78(9), 2279-2286.

findGroves *Identify clusters of similar trees*

Description

This function uses hierarchical clustering on principal components output by `treespace` to identify groups of similar trees. Clustering relies on `hclust`, using Ward's method by default.

Usage

```
findGroves(
  x,
  method = "treeVec",
  nf = NULL,
  clustering = "ward.D2",
  nclust = NULL,
  ...
)
```

Arguments

`x`	an object of the class multiPhylo or the output of the function `treespace`
`method`	(ignored if x is from `treespace`) this specifies a function which outputs the summary of a tree in the form of a vector. Defaults to `treeVec`.
`nf`	(ignored if x is from `treespace`) the number of principal components to retain
`clustering`	a character string indicating the clustering method to be used; defaults to Ward's method; see argument `method` in `?hclust` for more details.
`nclust`	an integer indicating the number of clusters to find; if not provided, an interactive process based on cutoff threshold selection is used.
`...`	further arguments to be passed to `treespace`

Value

A list containing:

- groups: a factor defining groups of trees
- treespace: the output of treespace

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

See Also

`plotGroves` to display results

Examples

```
if(require("adegenet") && require("adegraphics")){
## load data
data(woodmiceTrees)

## run findGroves: treespace+clustering
res <- findGroves(woodmiceTrees, nf=5, nclust=6)

## plot results on first 2 axes
PCs <- res$treespace$pco$li
s.class(PCs, fac=res$groups, col=funky(6))

## using plotGroves
plotGroves(res)
}
```

`findMRCIs` *Find MRCIs*

Description

Function to find the most recent common infector (MRCI) matrix from "who infected whom" information.

Usage

```
findMRCIs(wiw)
```

Arguments

`wiw`	a two-column matrix where the first column gives the infectors and the second column gives the infectees; each row corresponds to a transmission event from an infector to an infectee.

Value

Returns three objects:

- `sourceCase`: the number of the node which is the source case, i.e. the common infector of all cases (outputs a warning if there is more than one source case).
- `mrcis`: a matrix where, for each pair of individuals i and j, the entry (i,j) is the node number of their MRCI. Note that if i infected j then this entry is i itself.
- `mrciDepths`: a matrix where, for each pair of individuals i and j, the entry (i,j) is the depth of their MRCI, defined as the number of edges from the source case. The source case has depth zero, its direct infectees have depth 1, and so on.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## a simple who infected whom matrix:
tree1 <- cbind(Infector=1:5,Infectee=2:6)
findMRCIs(tree1)
```

`fluTrees` *BEAST analysis of seasonal influenza (A/H3N2)*

Description

These trees were created using BEAST on hemagglutinin (HA) segments of seasonal influenza A/H3N2 samples collected in New-York city (US) between 2000 and 2003. This data comes from the influenza BEAST tutorial distributed at: http://beast.bio.ed.ac.uk/tutorials

Format

A multiPhylo object containing 200 trees, each with 165 tips

Details

Only the first 200 trees (out of 10,000) were retained.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Source

http://beast.bio.ed.ac.uk/tutorials

References

http://beast.bio.ed.ac.uk/tutorials

linearMrca *Linear MRCA function*

Description

Function to make the most recent common ancestor (MRCA) matrix of a tree, where entry (i,j) gives the MRCA of tips i and j. The function is linear, exploiting the fact that the tree is rooted.

Usage

```
linearMrca(tree, k = 0)
```

Arguments

tree	an object of the class phylo which should be rooted.
k	(optional) number of tips in tree, for faster computation

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate a random tree
x <- rtree(6)

## create matrix of MRCAs: entry (i,j) is the node number of the MRCA of tips i and j
linearMrca(x,6)
```

makeCollapsedTree *Collapse a tree into a single tip per category*

Description

Reduce a tree with many tips into a tree with a single tip per category. Where a category's tips form a monophyletic clade, the clade is replaced by a single tip labelled by that category. Where a category's tips are paraphyletic, the largest clade for that category is treated as above, and all other tips pruned.

Usage

```
makeCollapsedTree(tree, df, warnings = TRUE)
```

Arguments

tree	an object of the class phylo: the tree to collapse.
df	a two-column data frame linking tip labels (column 2) with their corresponding categories (column 1).
warnings	a logical determining whether a warning should be given if there are paraphyletic categories (default TRUE)

Value

A tree (class phylo) whose tip labels are exactly the set of unique categories from df.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

See Also

treeConcordance simulateIndTree

Examples

```
# simulate a tree which is monophyletic per category
tree <- simulateIndTree(rtree(5), permuteTips=FALSE)

df <- cbind(sort(rep(rtree(5)$tip.label,5)),sort(tree$tip.label))
palette <- c("red","blue","black","green","purple")#'
tipCols <- palette[as.factor(sapply(tree$tip.label, function(l) df[which(df[,2]==l),1]))]
plot(tree, tip.color=tipCols)
collapsedTree <- makeCollapsedTree(tree,df)
plot(collapsedTree, tip.color=palette[as.factor(collapsedTree$tip.label)])

# simulate a tree which is paraphyletic per category
tree <- simulateIndTree(rtree(5), tipPercent=20)
tipCols <- palette[as.factor(sapply(tree$tip.label, function(l) df[which(df[,2]==l),1]))]
plot(tree, tip.color=tipCols)
collapsedTree <- makeCollapsedTree(tree,df)
plot(collapsedTree, tip.color=palette[as.factor(collapsedTree$tip.label)])
```

---

medTree *Geometric median tree function*

Description

Finds the geometric median of a set of trees according to the Kendall Colijn metric.

Usage

```
medTree(
  x,
  groups = NULL,
  lambda = 0,
  weights = NULL,
  emphasise.tips = NULL,
  emphasise.weight = 2,
  return.lambda.function = FALSE,
  save.memory = FALSE
)
```

Arguments

x	A list of trees of the class multiPhylo, for which the median tree will be computed, OR a matrix of tree vectors as given by `treespace$vectors`.
groups	an optional factor defining groups of trees; if provided, one median tree will be found for each group.
lambda	a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised. This argument is ignored if `return.lambda.function=TRUE` or if the vectors are already supplied as the object `x`.
weights	A vector of weights for the trees. Defaults to a vector of 1's so that all trees are equally weighted, but can be used to encode likelihood, posterior probabilities or other characteristics.
emphasise.tips	an optional list of tips whose entries in the tree vectors should be emphasised. Defaults to `NULL`.
emphasise.weight	applicable only if a list is supplied to `emphasise.tips`, this value (default 2) is the number by which vector entries corresponding to those tips are emphasised.
return.lambda.function	If true, a function that can be invoked with different lambda values is returned. This function returns the vector of metric values for the given lambda. Ignored if the tree vectors are already supplied as the object `x`.
save.memory	A flag that saves a lot of memory but increases the execution time (not compatible with return.lambda.function=TRUE). Ignored if the tree vectors are already supplied as the object `x`.

Value

A list of five objects:

- $centre is the "central vector", that is, the (weighted) mean of the tree vectors (which typically does not correspond to a tree itself);
- $distances gives the distance of each tree from the central vector;

- $mindist is the minimum of these distances;
- $treenumbers gives the numbers (and, if supplied, names) of the "median tree(s)", that is, the tree(s) which achieve this minimum distance to the centre;
- $trees if trees were supplied then this returns the median trees as a multiPhylo object.

If groups are provided, then one list is returned for each group. If `return.lambda.function=TRUE` then a function is returned that produces this list for a given value of lambda.

Author(s)

Jacob Almagro-Garcia <nativecoder@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

Thibaut Jombart <thibautjombart@gmail.com>

Examples

```
## EXAMPLE WITH WOODMICE DATA
data(woodmiceTrees)

## LOOKING FOR A SINGLE MEDIAN
## get median tree(s)
res <- medTree(woodmiceTrees)
res

## plot first tree
med.tree <- res$trees[[1]]
plot(med.tree)

## LOOKING FOR MEDIANS IN SEVERAL CLUSTERS
## identify 6 clusters
groves <- findGroves(woodmiceTrees, nf=3, nclust=6)

## find median trees
res.with.grp <- medTree(woodmiceTrees, groves$groups)

## there is one output per cluster
names(res.with.grp)

## get the first median of each
med.trees <- lapply(res.with.grp, function(e) ladderize(e$trees[[1]]))

## plot trees
par(mfrow=c(2,3))
for(i in 1:length(med.trees)) plot(med.trees[[i]], main=paste("cluster",i))

## highlight the differences between a pair of median trees
plotTreeDiff(med.trees[[1]],med.trees[[5]])
```

multiDist *Metric function for* multiPhylo *input*

Description

Comparison of a list of trees using the Kendall Colijn metric. Output is given as a pairwise distance matrix. This is equivalent to the $D output from treespace but may be preferable for large datasets, and when principal co-ordinate analysis is not required. It includes an option to save memory at the expense of computation time.

Usage

```
multiDist(
  trees,
  lambda = 0,
  return.lambda.function = FALSE,
  save.memory = FALSE,
  emphasise.tips = NULL,
  emphasise.weight = 2
)
```

Arguments

trees	an object of the class multiPhylo containing the trees to be compared
lambda	a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised. This argument is ignored if return.lambda.function=TRUE.
return.lambda.function	If true, a function that can be invoked with different lambda values is returned. This function returns the matrix of metric values for the given lambda.
save.memory	A flag that saves a lot of memory but increases the execution time (not compatible with return.lambda.function=TRUE).
emphasise.tips	an optional list of tips whose entries in the tree vectors should be emphasised. Defaults to NULL.
emphasise.weight	applicable only if a list is supplied to emphasise.tips, this value (default 2) is the number by which vector entries corresponding to those tips are emphasised.

Value

The pairwise tree distance matrix or a function that produces the distance matrix given a value for lambda.

Author(s)

Jacob Almagro-Garcia <nativecoder@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate 10 random trees, each with 6 tips
trees <- rmtree(10,6)

## pairwise distance matrix when lambda=0
multiDist(trees)

## pairwise distance matrix as a function of lambda:
m <- multiDist(trees, return.lambda.function=TRUE)

## evaluate at lambda=0. Equivalent to multiDist(trees).
m0 <- m(0)

## save memory by recomputing each tree vector for each pairwise tree comparison (for fixed lambda):
m0.5 <- multiDist(trees,0.5,save.memory=TRUE)
```

---

plotGroves *Scatterplot of groups of trees*

Description

This function displays the scatterplot of the Multidimensional Scaling (MDS) output by treespace, superimposing group information (derived by `findGroves`) using colors.

Usage

```
plotGroves(
  x,
  groups = NULL,
  xax = 1,
  yax = 2,
  type = c("chull", "ellipse"),
  col.pal = funky,
  bg = "white",
  lab.show = FALSE,
  lab.col = "black",
  lab.cex = 1,
  lab.optim = TRUE,
  point.cex = 1,
  scree.pal = NULL,
  scree.size = 0.2,
  scree.posi = c(0.02, 0.02),
  ...
)
```

Arguments

x	a list returned by findGroves or a MDS with class dudi
groups	a factor defining groups of trees
xax	a number indicating which principal component to be used as 'x' axis
yax	a number indicating which principal component to be used as 'y' axis
type	a character string indicating which type of graph to use
col.pal	a color palette to be used for the groups
bg	the background color
lab.show	a logical indicating whether labels should be displayed
lab.col	a color for the labels
lab.cex	the size of the labels
lab.optim	a logical indicating whether label positions should be optimized to avoid overlap; better display but time-consuming for large datasets
point.cex	the size of the points
scree.pal	a color palette for the screeplot
scree.size	a size factor for the screeplot, between 0 and 1
scree.posi	either a character string or xy coordinates indicating the position of the screeplot.
...	further arguments passed to s.class

Details

This function relies on s.class from the adegraphics package.

Value

An adegraphics object (class: ADEgS)

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

See Also

findGroves to find any clusters in the tree landscape s.class

Examples

```
## Not run:
if(require("adegenet") && require("adegraphics")){
## load data
data(woodmiceTrees)

## run findGroves: treespace+clustering
res <- findGroves(woodmiceTrees, nf=5, nclust=6)
```

```
## basic plot
plotGroves(res)

## adding labels
plotGroves(res, lab.show=TRUE)

## customizing
plotGroves(res, lab.show=TRUE,
bg="black", lab.col="white", scree.size=.35)

## customizing
plotGroves(res, type="ellipse", lab.show=TRUE,
lab.optim=FALSE, scree.size=.35)

## example with no group information
plotGroves(res$treespace$pco)

## adding labels
plotGroves(res$treespace$pco, lab.show=TRUE, lab.cex=2)

}

## End(Not run)
```

---

plotGrovesD3 *Scatterplot of groups of trees using* scatterD3

Description

This function displays the scatterplot of the Multidimensional Scaling (MDS) output by treespace, superimposing group information (derived by findGroves) using colors. scatterD3 enables interactive plotting based on d3.js, including zooming, panning and fading effects in the legend.

Usage

```
plotGrovesD3(
  x,
  groups = NULL,
  xax = 1,
  yax = 2,
  treeNames = NULL,
  symbol_var = NULL,
  xlab = paste0("Axis ", xax),
  ylab = paste0("Axis ", yax),
  ...
)
```

Arguments

x	a list returned by findGroves or a MDS with class dudi
groups	a factor defining groups of trees. If x is a list returned by findGroves these will be detected automatically.
xax	a number indicating which principal component to be used as 'x' axis
yax	a number indicating which principal component to be used as 'y' axis
treeNames	if a list of tree names or labels are given, these will be plotted alongside the points. Their size can be altered using labels_size - see ?scatterD3 for more information.
symbol_var	a factor by which to vary the symbols in the plot
xlab	the label for the 'x' axis. Defaults to use the value of 'xax'
ylab	the label for the 'y' axis. Defaults to use the value of 'yax'
...	further arguments passed to scatterD3

Value

A scatterD3 plot

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

See Also

findGroves to find any clusters in the tree landscape

Examples

```
## Not run:
if(require("adegenet") && require("scatterD3")){
## load data
data(woodmiceTrees)

## run findGroves: treespace+clustering
res <- findGroves(woodmiceTrees, nf=5, nclust=6)

## basic plot
plotGrovesD3(res)

## adding tree labels
plotGrovesD3(res, treeNames=1:201)

## customizing: vary the colour and the symbol by group
plotGrovesD3(res, symbol_var=res$groups)

## example with no group information
plotGrovesD3(res$treespace$pco)
}
```

```
## End(Not run)
```

---

plotTreeDiff *Plot tree differences*

Description

Highlight the topological differences between two trees, plotted side by side. This function is useful for comparing representative "median" trees - see medTree. It relies on the function tipDiff.

Usage

```
plotTreeDiff(
  tr1,
  tr2,
  tipDiff = NULL,
  vec1 = NULL,
  vec2 = NULL,
  sizeOfDifferences = FALSE,
  tipMatch = TRUE,
  treesFacing = FALSE,
  baseCol = "grey",
  col1 = "peachpuff",
  col2 = "red2",
  colourMethod = "ramp",
  palette = lightseasun,
  ...
)
```

Arguments

tr1	an object of the class phylo: the first tree to plot.
tr2	an object of the class phylo: the second tree to plot.
tipDiff	an optional input, the result of tipDiff. Supplying this will save time if calling plotTreeDiff repeatedly, for example with different aesthetics.
vec1	an optional input, the result of treeVec(tr1, lambda=0). This argument is ignored if tipDiff is supplied; otherwise supplying this will save time if calling plotTreeDiff repeatedly, for example with different aesthetics.
vec2	an optional input, the result of treeVec(tr2, lambda=0). This argument is ignored if tipDiff is supplied; otherwise supplying this will save time if calling plotTreeDiff repeatedly, for example with different aesthetics.
sizeOfDifferences	a logical (default FALSE) specifying whether the size of the tip differences should be used, or just a count of the number of differences (see tipDiff)

tipMatch	a logical (default TRUE) specifying whether the second tree should be rotated so that, as far as possible, each of its tips lies opposite its equivalent in the first tree
treesFacing	a logical (default FALSE) specifying whether the trees should be plotted facing each other - that is, with the second tree plotted "leftwards".
baseCol	the colour used for tips with identical ancestry in the two trees. Defaults to "grey".
col1	the first colour used to define the colour spectrum for tips with differences. This colour will be used for tips with minor differences. Defaults to "peachpuff". Ignored if colourMethod="palette".
col2	the second colour used to define the colour spectrum for tips with differences. This colour will be used for tips with major differences. Defaults to "red2". Ignored if colourMethod="palette".
colourMethod	the method to use for colouring. Default is "ramp", corresponding to the original implementation, where the function colorRampPalette is used to create a palette which ranges from col1 to col2. For large trees this can be hard to interpret, and method palette may be preferred, which permits the selection of a palette to use in adegenet's function num2col.
palette	the colour palette to be used if colourMethod="palette". For a list of available palettes see ?num2col.
...	further arguments passed to plot.phylo

Value

A plot of the two trees side by side. Tips are coloured in the following way:

- if each ancestor of a tip in tree 1 occurs in tree 2 with the same partition of tip descendants, then the tip is coloured grey (or supplied "baseCol")
- if not, the tip gets coloured pale orange to red on a scale according to how many differences there are amongst its most recent common ancestors with other tips. The colour spectrum can be changed according to preference.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

See Also

medTree, tipDiff

Examples

```
## simple example on trees with five tips:
tr1 <- read.tree(text="((A:1,B:1):1,((C:1,D:1):1,E:1):1):1;")
tr2 <- read.tree(text="((A:1,B:1):1,(C:1,(D:1,E:1):1):1):1;")
plotTreeDiff(tr1,tr2)

## example on larger woodmice trees
```

```
data(woodmiceTrees)
tr1 <- woodmiceTrees[[1]]
tr2 <- woodmiceTrees[[57]] # for example

# find the tip differences in advance, to avoid recalculating with each plot
wmTipDiff <- tipDiff(tr1,tr2, sizeOfDifferences=TRUE)
plotTreeDiff(tr1,tr2, tipDiff=wmTipDiff, tipMatch=TRUE)

## change aesthetics:
# trees facing each other:
plotTreeDiff(tr1,tr2, tipDiff=wmTipDiff, treesFacing=TRUE)

# radial plots, and change colours:
plotTreeDiff(tr1,tr2, tipDiff=wmTipDiff,
   baseCol="grey2", col1="cyan", col2="navy",
   edge.width=2, type="radial", cex=0.5, font=2)
# cladogram plots, and use colour palette from adegenet to see differences more clearly:
plotTreeDiff(tr1,tr2, tipDiff=wmTipDiff,
   treesFacing=TRUE, baseCol="black", colourMethod="palette",
   edge.width=2, type="cladogram", cex=0.5, font=2)

# including the size of the differences highlights tip "No0906s" a little more:
# (this is typically a more informative plot in cases where many tips have the
# same difference count, for example when a whole clade has been shifted "up"
# or "down" the tree but its internal topology remains the same.)

plotTreeDiff(tr1,tr2, tipDiff=wmTipDiff, sizeOfDifferences=TRUE,
   treesFacing=TRUE, baseCol="black", colourMethod="palette",
   edge.width=2, type="cladogram", cex=0.5, font=2)
```

`refTreeDist` *Metric function for comparing a reference* `phylo` *to* `multiPhylo` *input*

Description

Comparison of a single reference tree to a list of trees using the Kendall Colijn metric. Output is given as a vector of distances from the reference tree.

Usage

```
refTreeDist(
  refTree,
  trees,
  lambda = 0,
  return.lambda.function = FALSE,
  emphasise.tips = NULL,
  emphasise.weight = 2
)
```

Arguments

refTree	a tree of class phylo, the "reference tree".
trees	an object of the class multiPhylo containing the trees to be compared to the reference tree
lambda	a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised. This argument is ignored if return.lambda.function=TRUE.
return.lambda.function	If true, a function that can be invoked with different lambda values is returned. This function returns the vector of metric values for the given lambda.
emphasise.tips	an optional list of tips whose entries in the tree vectors should be emphasised. Defaults to NULL.
emphasise.weight	applicable only if a list is supplied to emphasise.tips, this value (default 2) is the number by which vector entries corresponding to those tips are emphasised.

Value

The vector of distances, where entry i corresponds to the distance between the i'th tree and the reference tree, or a function that produces the vector of distances given a value for lambda.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate a single reference tree with 6 tips
refTree <- rtree(6)

## generate 10 random trees, each with 6 tips
trees <- rmtree(10,6)

## find the distances from each of the 10 random trees to the single reference tree
refTreeDist(refTree,trees)
```

---

relatedTreeDist *Tree distance when trees have "related" tips*

Description

This function calculates the distances between trees whose tips belong to the same categories but are not necessarily identically labelled

Usage

```
relatedTreeDist(trees, df, checkTrees = TRUE)
```

Arguments

trees	a list of trees or multiphylo object
df	a data frame specifying to which category each individual (from all the trees) belongs. Each row gives: an individual (column 2) and its corresponding category (column 1)
checkTrees	a logical (default TRUE) specifying whether the trees should be checked. When TRUE, error messages will be helpful in locating problematic trees, that is, any trees with repeated tip labels, or any trees with missing categories.

Examples

```
# we will simulate some trees as an example, each "based" on the same tree:
baseTree <- rtree(5)
baseTree$tip.label <- letters[5:1]
plot(baseTree)

tree1 <- simulateIndTree(baseTree, itips=3, permuteTips=FALSE)
tree2 <- simulateIndTree(baseTree, itips=4, permuteTips=FALSE)
tree3 <- simulateIndTree(baseTree, itips=4, permuteTips=TRUE, tipPercent=20)
tree4 <- simulateIndTree(baseTree, itips=4, permuteTips=TRUE, tipPercent=60)
tree5 <- simulateIndTree(baseTree, itips=4, permuteTips=TRUE, tipPercent=100)
# combine:
trees <- list(tree1,tree2,tree3,tree4,tree5)

df <- cbind(sort(rep(letters[1:5],4)),sort(paste0(letters[1:5],"_",rep(1:4,5))))
head(df)

# Find distances:
relatedTreeDist(trees,df)

# Note that trees 1 and 2 have different numbers of tips but the relationships between those tips
# are identical at the category level, hence the related tree distance is 0.
# We can see that the distances between trees increase the more the trees are permuted.
```

---

simulateIndTree *Simulate randomised "individuals" tree*

Description

This function takes in a "category" tree and outputs a simulated corresponding "individuals" tree, for testing the concordance measure.

Usage

```
simulateIndTree(
  catTree,
```

```
  itips = 5,
  permuteCat = FALSE,
  permuteTips = TRUE,
  tipPercent = 100
)
```

Arguments

catTree	object of class phylo, the category-level tree
itips	number of individual tips to assign per category
permuteCat	logical specifying whether to permute the category labels on the category tree before grafting on individual tips. Defaults to FALSE.
permuteTips	logical specifying whether to permute the individual tip labels after building the individual level tree based on the category tree. Defaults to TRUE.
tipPercent	number specifying the percentage of tips to be permuted. Defaults to 100, ignored if permuteTips=FALSE.

See Also

treeConcordance makeCollapsedTree

Examples

```
tree <- simulateIndTree(rtree(3))
plot(tree)
```

---

tipDiff *Find tip position differences*

Description

Find the topologicial differences between two trees with the same tip labels. The function returns a data frame of the tips and the number of differences in their ancestry between the two trees. Called by plotTreeDiff, which highlights the differing tips in a plot of the two trees.

Usage

```
tipDiff(tr1, tr2, vec1 = NULL, vec2 = NULL, sizeOfDifferences = FALSE)
```

Arguments

tr1	an object of the class phylo: the first tree to compare.
tr2	an object of the class phylo: the second tree to compare.
vec1	an optional input, the result of treeVec(tr1, lambda=0), to speed up the computation.

vec2 an optional input, the result of `treeVec(tr2, lambda=0)`, to speed up the computation.

sizeOfDifferences a logical (default FALSE) specifying whether the size of the differences in the vectors per tip is also computed

Value

A data frame of the tree tips and the number of ancestral differences between them in the two trees, in order of increasing difference. A tip is said to have zero difference if each of its ancestral nodes admits the same tip partition in the two trees.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

See Also

medTree plotTreeDiff

Examples

```
## simple example on trees with five tips:
tr1 <- read.tree(text="((A:1,B:1):1,((C:1,D:1):1,E:1):1):1;")
tr2 <- read.tree(text="((A:1,B:1):1,(C:1,(D:1,E:1):1):1):1;")
tipDiff(tr1,tr2)

## example on larger woodmice trees
data(woodmiceTrees)
tipDiff(woodmiceTrees[[1]],woodmiceTrees[[2]])
```

---

tipsMRCAdepths *Tip-tip MRCA depths*

Description

This function creates a matrix where columns 1 and 2 correspond to tip labels and column 3 gives the depth of the MRCA of that pair of tips. It is strongly based on `treeVec` and is used by `relatedTreeDist` and `treeConcordance` where tip labels belong to "categories".

Usage

```
tipsMRCAdepths(tree)
```

Arguments

tree An object of class phylo

Examples

```
tree <- rtree(10)
plot(tree)
tipsMRCAdepths(tree)
```

`treeConcordance` *Tree concordance*

Description

This function calculates the concordance between a category tree and an individuals tree

Usage

```
treeConcordance(catTree, indTree, df)
```

Arguments

`catTree`	object of class phylo
`indTree`	object of class phylo
`df`	data frame specifying to which category each individual belongs. Each row gives an individual (column 2) and its corresponding category (column 1)

See Also

`simulateIndTree` `makeCollapsedTree`

Examples

```
# create an example category tree
catTree <- read.tree(text="(C:1,(B:1,A:1):1);")
plot(catTree)

# make individuals tree with complete concordance:
indTree1 <- read.tree(text="(((c4,c3),(c2,c1)),((b1,b2),((a3,a2),a1)));")
plot(indTree1)

# create data frame linking categories with individuals
df <- cbind(c(rep("A",3),rep("B",2),rep("C",4)),sort(indTree1$tip.label))

treeConcordance(catTree,indTree1,df)

# make a less concordant tree:
indTree2 <- read.tree(text="((((c4,c3),(c2,c1)),b2),(b1,((a3,a2),a1)));")
plot(indTree2)
treeConcordance(catTree,indTree2,df)
```

```
# simulate larger example:
catTree <- rtree(10)
indTree3 <- simulateIndTree(catTree, tipPercent=10)
df <- cbind(sort(rep(catTree$tip.label,5)),sort(indTree3$tip.label))
plot(indTree3)
treeConcordance(catTree,indTree3,df)
```

`treeDist` *Metric function*

Description

Comparison of two trees using the Kendall Colijn metric

Usage

```
treeDist(
  tree.a,
  tree.b,
  lambda = 0,
  return.lambda.function = FALSE,
  emphasise.tips = NULL,
  emphasise.weight = 2
)
```

Arguments

`tree.a`	an object of the class `phylo`
`tree.b`	an object of the class `phylo` (with the same tip labels as tree.a)
`lambda`	a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised. This argument is ignored if `return.lambda.function=TRUE`.
`return.lambda.function`	If true, a function that can be invoked with different lambda values is returned. This function returns the vector of metric values for the given lambda.
`emphasise.tips`	an optional list of tips whose entries in the tree vectors should be emphasised. Defaults to `NULL`.
`emphasise.weight`	applicable only if a list is supplied to `emphasise.tips`, this value (default 2) is the number by which vector entries corresponding to those tips are emphasised.

Value

The distance between the two trees according to the metric for the given value of lambda, or a function that produces the distance given a value of lambda.

Author(s)

Jacob Almagro-Garcia <nativecoder@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate random trees
tree.a <- rtree(6)
tree.b <- rtree(6)
treeDist(tree.a,tree.b) # lambda=0
treeDist(tree.a,tree.b,1)  # lambda=1
dist.func <- treeDist(tree.a,tree.b,return.lambda.function=TRUE) # distance as a function of lambda
dist.func(0) # evaluate at lambda=0. Equivalent to treeDist(tree.a,tree.b).
## We can see how the distance changes when moving from focusing on topology to length:
plot(sapply(seq(0,1,length.out=100), function(x) dist.func(x)), type="l",ylab="",xlab="")

## The distance may also change if we emphasise the position of certain tips:
plot(sapply(tree.a$tip.label, function(x) treeDist(tree.a,tree.b,emphasise.tips=x)),
    xlab="Tip number",ylab="Distance when  vector entries corresponding to tip are doubled")
```

`treespace` *Phylogenetic tree exploration*

Description

Compares phylogenetic trees using a choice of metrics / measures, and maps their pairwise distances into a small number of dimensions for easy visualisation and identification of clusters.

Usage

```
treespace(
  x,
  method = "treeVec",
  nf = NULL,
  lambda = 0,
  return.tree.vectors = FALSE,
  processors = 1,
  ...
)
```

Arguments

`x`	an object of the class multiPhylo
`method`	the method for summarising the tree as a vector. Choose from:

- `treeVec` (default) the Kendall Colijn metric vector (for rooted trees)

- BHV the Billera, Holmes Vogtmann metric using dist.multiPhylo from package distory (for rooted trees)
- KF the Kuhner Felsenstein metric (branch score distance) using KF.dist from package phangorn (considers the trees unrooted)
- RF the Robinson Foulds metric using RF.dist from package phangorn (considers the trees unrooted)
- wRF the weighted Robinson Foulds metric using wRF.dist from package phangorn (considers the trees unrooted)
- nNodes the Steel & Penny tip-tip path difference metric, (topological, ignoring branch lengths), using path.dist from package phangorn (considers the trees unrooted)
- patristic the Steel & Penny tip-tip path difference metric, using branch lengths, calling path.dist from package phangorn (considers the trees unrooted)
- Abouheif: performs Abouheif's test, inherited from distTips in adephylo. See Pavoine et al. (2008) and adephylo.
- sumDD: sum of direct descendants of all nodes on the path, related to Abouheif's test, inherited from distTips in adephylo.

nf — the number of principal components to retain

lambda — a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised in the Kendall Colijn metric.

return.tree.vectors — if using the Kendall Colijn metric, this option will return the tree vectors as part of the output. Note that this can use a lot of memory so defaults to FALSE.

processors — value (default 1) to be passed to mcmapply specifying the number of cores to use. Must be 1 on Windows (see mcmapply for more details).

... — further arguments to be passed to method.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate list of trees
x <- rmtree(10, 20)
names(x) <- paste("tree", 1:10, sep = "")

## use treespace
res <- treespace(x, nf=3)
table.paint(as.matrix(res$D))
scatter(res$pco)

data(woodmiceTrees)
woodmiceDists <- treespace(woodmiceTrees,nf=3)
```

```
plot(woodmiceDists$pco$li[,1],woodmiceDists$pco$li[,2])
woodmicedf <- woodmiceDists$pco$li
if(require(ggplot2)){
woodmiceplot <- ggplot(woodmicedf, aes(x=A1, y=A2)) # create plot
woodmiceplot + geom_density2d(colour="gray80") + # contour lines
geom_point(size=6, shape=1, colour="gray50") + # grey edges
geom_point(size=6, alpha=0.2, colour="navy") + # transparent blue points
xlab("") + ylab("") + theme_bw(base_family="") # remove axis labels and grey background
}

## Not run:
if(require(rgl)){
plot3d(woodmicedf[,1], woodmicedf[,2], woodmicedf[,3], type="s", size=1.5,
col="navy", alpha=0.5, xlab="", ylab="", zlab="")
}

## End(Not run)
```

`treespaceServer` *Web-based tree explorer*

Description

This function opens up an application in a web browser for an interactive exploration of the diversity in a set of trees. For further details please see the "help" tab within the application.

Usage

```
treespaceServer()
```

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

See Also

For convenience, `treespaceServer` is also available as a separate web app which can be used from any browser (it is not necessary to have R installed): https://mkendall.shinyapps.io/treespace/

treeVec *Tree vector function*

Description

Function which takes an object of class phylo and outputs the vector for the Kendall Colijn metric. The elements of the vector are numeric if `return.lambda.function=FALSE` (default), and otherwise they are functions of lambda.

Usage

```
treeVec(
  tree,
  lambda = 0,
  return.lambda.function = FALSE,
  emphasise.tips = NULL,
  emphasise.weight = 2
)
```

Arguments

`tree`	an object of the class `phylo`
`lambda`	a number in [0,1] which specifies the extent to which topology (default, with lambda=0) or branch lengths (lambda=1) are emphasised. This argument is ignored if `return.lambda.function=TRUE`.
`return.lambda.function`	If true, a function that can be invoked with different lambda values is returned. This function returns the vector of metric values for the given lambda.
`emphasise.tips`	an optional list of tips whose entries in the tree vector should be emphasised. Defaults to `NULL`.
`emphasise.weight`	applicable only if a list is supplied to `emphasise.tips`, this value (default 2) is the number by which vector entries corresponding to those tips are emphasised.

Value

The vector of values according to the metric, or a function that produces the vector given a value of lambda.

Author(s)

Jacob Almagro-Garcia <nativecoder@gmail.com>

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
## generate a random tree
tree <- rtree(6)
## topological vector of mrca distances from root:
treeVec(tree)
## vector of mrca distances from root when lambda=0.5:
treeVec(tree,0.5)
## vector of mrca distances as a function of lambda:
vecAsFunction <- treeVec(tree,return.lambda.function=TRUE)
## evaluate the vector at lambda=0.5:
vecAsFunction(0.5)
```

---

wiwMedTree *Median transmission tree*

Description

Function to find the median of a list of transmission scenarios

Usage

```
wiwMedTree(matList, sampled = NULL, weights = NULL)
```

Arguments

matList	a list of matrices, each of which is the output of findMRCIs$mrciDepths
sampled	a vector of node IDs which corresponds to those nodes which are sampled cases
weights	optional vector of weights to correspond to the entries of matList

Value

Returns three objects:

- centre: the mean of the matList entries, restricted to the sampled cases
- distances: for each entry of matList, its distance from centre
- mindist: the minimum of distances
- median: the number of the median entry of matList, i.e. the one(s) which achieve the mindist from the centre.

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
# create some simple "who infected whom" scenarios:
tree1 <- cbind(Infector=1:5,Infectee=2:6)
tree2 <- cbind(Infector=c(1,5,2,2,3),Infectee=2:6)
tree3 <- cbind(Infector=c(2,2,3,4,5),Infectee=c(1,3,4,5,6))
# create list of the MRCI depth matrices:
matList <- lapply(list(tree1,tree2,tree3), function(x) findMRCIs(x)$mrciDepths)

# median tree, assuming all cases are sampled:
wiwMedTree(matList)
# median tree when cases 1, 2 and 4 are sampled:
wiwMedTree(matList, sampled=c(1,2,4))
```

`wiwTreeDist` *Transmission tree distance*

Description

Function to find the distance between transmission trees by comparing their MRCI depth matrices; to be precise, by finding the Euclidean distance between the tree vectors, restricted to their sampled node entries.

Usage

```
wiwTreeDist(matList, sampled = NULL)
```

Arguments

`matList`	a list of matrices, each of which is the output of `findMRCIs$mrciDepths`
`sampled`	a vector of node IDs which corresponds to those nodes which are sampled cases. Default is to treat all nodes as sampled cases.

Value

Returns a distance matrix, where entry (i,j) is the transmission tree distance between matrices i and j in `matList`

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Examples

```
# create some simple "who infected whom" scenarios:
tree1 <- cbind(Infector=1:5,Infectee=2:6)
tree2 <- cbind(Infector=c(1,5,2,2,3),Infectee=2:6)
tree3 <- cbind(Infector=c(2,2,3,4,5),Infectee=c(1,3,4,5,6))
# create list of the MRCI depth matrices:
matList <- lapply(list(tree1,tree2,tree3), function(x) findMRCIs(x)$mrciDepths)

# transmission tree distance, assuming all cases are sampled:
wiwTreeDist(matList)
# transmission tree distance when cases 1, 2 and 4 are sampled:
wiwTreeDist(matList, sampled=c(1,2,4))
```

`woodmiceTrees` *Bootstrap trees from woodmouse dataset*

Description

These trees were created using the neighbour-joining and bootstrapping example from the ape documentation.

Format

A multiPhylo object containing 201 trees, each with 15 tips

Author(s)

Michelle Kendall <michelle.louise.kendall@gmail.com>

Source

A set of 15 sequences of the mitochondrial gene cytochrome b of the woodmouse (Apodemus sylvaticus) which is a subset of the data analysed by Michaux et al. (2003). The full data set is available through GenBank (accession numbers AJ511877 to AJ511987)

References

Michaux, J. R., Magnanou, E., Paradis, E., Nieberding, C. and Libois, R. (2003) Mitochondrial phylogeography of the Woodmouse (Apodemus sylvaticus) in the Western Palearctic region. *Molecular Ecology*, 12, 685-697

Index

∗ **datasets**
  DengueBEASTMCC, 3
  DengueSeqs, 3
  DengueTrees, 4
  fluTrees, 7
  woodmiceTrees, 32
.render.server.info, 2

DengueBEASTMCC, 3
DengueSeqs, 3, *4*
DengueTrees, *3*, 4

findGroves, 5, *13–16*
findMRCIs, 6
fluTrees, 7

hclust, *5*

linearMrca, 8

makeCollapsedTree, 8, *22*, *24*
medTree, 9, *17, 18*, *23*
multiDist, 12

plot.phylo, *18*
plotGroves, *5*, 13
plotGrovesD3, 15
plotTreeDiff, 17, *22, 23*

refTreeDist, 19
relatedTreeDist, 20

s.class, *14*
scatterD3, *16*
simulateIndTree, *9*, 21, *24*

tipDiff, *17, 18*, 22
tipsMRCAdepths, 23
treeConcordance, *9*, *22*, 24
treeDist, 25
treespace, *5*, 26
treespaceServer, 28
treeVec, 29

wiwMedTree, 30
wiwTreeDist, 31
woodmiceTrees, 32