Package 'rwty'

July 23, 2025

Type Package

Title R We There Yet? Visualizing MCMC Convergence in Phylogenetics

Version 1.0.3

Description Implements various tests, visualizations, and metrics for diagnosing convergence of MCMC chains in phylogenetics. It implements and automates many of the functions of the AWTY package in the R environment, as well as a host of other functions. Warren, Geneva, and Lanfear (2017), <doi:10.1093/molbev/msw279>.

License GPL-2

Depends R (>= 3.3.0), ape, ggplot2,

Imports reshape2, phangorn, coda, viridis, grid, plyr, ggdendro, GGally, parallel, TreeDist

Suggests knitr, testthat, markdown, rmarkdown, stringi

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Dan Warren [aut, cre],
Anthony Geneva [aut],
Rob Lanfear [aut],
Luke Harmon [ctb],
April Wright [ctb],
Diego Mallo [ctb],
Klaus Schliep [ctb],
Kelly Luke [ctb],
Kendall Michelle [ctb],
Smith Martin [ctb]

Maintainer Dan Warren <dan.l.warren@gmail.com>

Repository CRAN

Date/Publication 2025-03-27 11:50:02 UTC

Contents

analyze.rwty . . . 2
check.chains . . . 4
clade.freq . . . 5
combine.ptables . . . 6
cumulative.freq . . . 6
estimate.autocorr.m . . . 7
fungus . . . 8
load.multi . . . 8
load.trees . . . 9
makeplot.acsf.cumulative . . . 10
makeplot.acsf.sliding . . . 11
makeplot.all.params . . . 12
makeplot.asdsf . . . 13
makeplot.autocorr . . . 14
makeplot.pairs . . . 15
makeplot.param . . . 16
makeplot.pseudo.ess . . . 17
makeplot.splitfreq.matrix . . . 18
makeplot.splitfreqs.cumulative . . . 19
makeplot.splitfreqs.sliding . . . 20
makeplot.topology . . . 21
makeplot.treespace . . . 22
parse.clades . . . 23
parse.slide.clades . . . 24
print.rwty.chain . . . 24
salamanders . . . 25
slide.freq . . . 25
topological.approx.ess . . . 26
topological.autocorr . . . 27
topological.pseudo.ess . . . 29
tree.dist.matrix . . . 30
treespace . . . 32

Index **33**

`analyze.rwty` *analyze.rwty, the main interface for rwty analyses and plots.*

Description

This is the main user interface to rwty. It allows users to conduct simple visualizations of MCMC chain performance with very few arguments.

Usage

```
analyze.rwty(
  chains,
  burnin = 0,
  window.size = 20,
  treespace.points = 100,
  n.clades = 20,
  min.freq = 0,
  fill.color = NA,
  filename = NA,
  overwrite = FALSE,
  facet = TRUE,
  free_y = FALSE,
  autocorr.intervals = 100,
  ess.reps = 20,
  treedist = "PD",
  params = NA,
  max.sampling.interval = NA,
  ...
)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin. Default value is zero.
window.size	The number of trees to include in each windows of sliding window plots
treespace.points	The number of trees to plot in the treespace plot. Default is 100
n.clades	The number of clades to include in plots of split frequencies over the course of the MCMC
min.freq	The minimum frequency for a node to be used for calculating ASDSF. Default is 0.1
fill.color	The name of a column in your log file that you would like to use as the fill colour of points in the treespace plots
filename	Name of an output file (e.g., "output.pdf"). If none is supplied, rwty will not save outputs to file.
overwrite	Boolean variable saying whether output file should be overwritten, if it exists.
facet	A Boolean expression indicating whether multiple chains should be plotted as facet plots (default TRUE).
free_y	TRUE/FALSE to turn free y scales on the facetted plots on or off (default FALSE). Only works if facet = TRUE.
autocorr.intervals	The maximum number of intervals to use for autocorrelation plots.
ess.reps	The number of replicate analyses to do when calculating the pseudo ESS.

treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance.
params	A vector of parameters to use when making the parameter correlation plots. Defaults to the first two columns in the log table.
max.sampling.interval	The maximum sampling interval to use for generating autocorrelation plots
...	Extra arguments to be passed to plotting and analysis functions.

Value

output The output is a list containing the following plots:

Plots of likelihood, model parameters, and tree topologies as a function of chain length (the first two only when output from MCMC parameters has been loaded along with the tree topologies).

Plot of autocorrelation of tree topolgies at different sampling intervals along a chain

Plot of split frequencies calculated in sliding windows for the most variable clades

Plot of change in split frequencies between sliding windows for all clades

Plot of cumulative split frequencies as the MCMC progresses

Plot of change in cumulative split frequencies as the MCMC progresses

Heatmap and point depictions of chains in treespace.

Plot of the Average Standard Deviation of Split Frequencies (ASDSF) between chains as the MCMC progresses

Plot of pairwise correlations between split frequencies among chains

Plot of chains clustered by their pairwise ASDSF values

Examples

```
## Not run:
data(fungus)
p <- analyze.rwty(fungus, burnin = 50, window.num = 50)
p

## End(Not run)
```

---

check.chains *Function for checking suitability of chains for rwty analyses, auto-generating labels, etc*

Description

This function is automatically called by many other functions, but can be run manually as well. It performs a number of tests of chain format, labels, lengths, etc.

Usage

```
check.chains(chains)
```

Arguments

chains	A list of rwty.chain objects.

Value

chains A list of rwty.chain objects

Examples

```
## Not run:
data(fungus)
check.chains(fungus)

## End(Not run)
```

---

clade.freq *Returns clade names and frequencies*

Description

Uses ape functionality to get the frequencies and names of clades in an MCMC chain or subset thereof.

Usage

```
clade.freq(x, start, end, rooted = FALSE, ...)
```

Arguments

x	A multiPhylo or rwty.chain object
start	The index of the first tree to consider in calcuating frequencies
end	The index of the last tree to consider in calculating frequencies
rooted	(TRUE/FALSE). Tells RWTY whether your trees are rooted or not.
...	Arguments to be passed to ape's prop.part function

Value

clade.df A data froma containing clade names and frequencies

Examples

```
## Not run:
data(fungus)
clade.freq(fungus$Fungus.Run1, start=10, end=100)

## End(Not run)
```

combine.ptables *Function for merging p tables for multiple MCMC chains*

Description

This function is automatically called by some of the plot functions.

Usage

```
combine.ptables(chains, burnin)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin

Value

ptable A data frame of likelihood values and model parameters for the supplied rwty.chain objects

Examples

```
## Not run:
data(fungus)
combine.ptables(fungus, burnin=20)

## End(Not run)
```

cumulative.freq *Cumulative means of clade split frequencies.*

Description

This function calculates the cumulative mean split frequencies of clades as an MCMC progresses.

Usage

```
cumulative.freq(chains, burnin = 0, window.size = 20)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin. Defaults to zero.
window.size	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)

Value

A list of rwty.cumulative objects, one per chain in the input list of chains. Each rwty.cumulative object contains the cumulative mean split frequencies of clades at sp windows, and a translation table that converts clade groupings to factors.

Examples

```
## Not run:
data(fungus)
cumulative.data <- cumulative.freq(fungus, burnin=20)

## End(Not run)
```

`estimate.autocorr.m` *Calculate sampling interval based on exponential semivariogram model.*

Description

This function uses an exponential semivariogram model to estimate the asymptotic topological distance, and uses that to estimate the sampling interval at which topological distances have reached some fixed proportion of that value (default 0.95). It expects as input a data table output by rwty's topological.autocorr function

Usage

```
estimate.autocorr.m(dat, ac.cutoff = 0.95)
```

Arguments

`dat`	A data frame output from topological.autocorr.
`ac.cutoff`	Default 0.95. The proportion of the asymptotic topological distance to use as a cutoff for determining sampling interval. For example, if ac.cutoff = 0.9, then the minimum sampling interval returned is the one that guarantees a topological distance at least 0.9 times the asymptotic value.

Value

A data frame consisting of the value matching the ac.cutoff proportion of the asymptotic topological distance for each chain. This sampling interval estimates the interval at which topological distances are no longer autocorrelated. If the value is larger than the largest sampling distance, the table records this as a value of -1

Examples

```
data(fungus)
## Not run:
# To get a good estimate we need all sampling intervals
autocorr.intervals = as.integer(length(fungus[[1]]$trees)/21)
sampling.table <- topological.autocorr(fungus, burnin = 20, autocorr.intervals = autocorr.intervals)
estimate.autocorr.m(sampling.table)

## End(Not run)
```

`fungus` *MrBayes output from analysis of Hibbett et al. data*

Description

This is the output from a MrBayes run of 10,000,000 generations using the analysis settings from the original .nex file. Sampling is one tree per 40,000 generations. Four chains are included, each with its associated log file.

Usage

```
data(fungus)
```

Format

A data frame with four chains of 251 phylogenetic trees and associated likelihood and parameter values.

References

Study reference: Hibbett D., Pine E., Langer E., Langer G., & Donoghue M. 1997. Evolution of gilled mushrooms and puffballs inferred from ribosomal DNA sequences. Proceedings of the National Academy of Sciences of the United States of America, 94(22): 12002-12006.

http://treebase.org/treebase-web/search/study/summary.html?id=271

`load.multi` *Load all matching files from a directory into a list of rwty.chain objects*

Description

Finds trees and log files based on format definition, returns rwty.chain objects containing both

Usage

```
load.multi(path = ".", format = "mb", labels = NA, ...)
```

Arguments

path	The path to the directory containing tree and log files
format	File format, which is used to find tree and log files. Currently accepted values are "mb" for MrBayes, "beast" for BEAST, "*beast" for *BEAST, and "revbayes" for RevBayes. If you would like RWTY to understand additional formats, please contact the authors and send us some sample data.
labels	A vector of names to assign to chains as they are read in.
...	Further arguments to be passed to load.trees.

Value

output A list of rwty.chain objects containing the multiPhylos and the tables of values from the log files if available.

Examples

```
#load.multi(path = "~/my trees/", format = "*beast")
```

---

load.trees *Custom functions to load tree lists so that rwty can do basic processing on the way in.*

Description

Loads trees, looks for a log file of tree likelihoods and parameter values, returns an rwty.chain object containing both

Usage

```
load.trees(
  file,
  type = NA,
  format = "mb",
  gens.per.tree = NA,
  trim = 1,
  logfile = NA,
  skip = NA
)
```

Arguments

file	A path to a tree file containing an MCMC chain of trees
type	An argument that designates the type of tree file. If "nexus", trees are loaded using ape's read.nexus function. Otherwise, it's read.tree. If a "format" argument is passed, type will be determined from the format definition.

format	File format, which is used to find tree and log files. Currently accepted values are "mb" for MrBayes, "beast" for BEAST, "*beast" for *BEAST, and "revbayes" for RevBayes. If you would like RWTY to understand additional formats, please contact the authors and send us some sample data.
gens.per.tree	The number of generations separating trees. If not provided, RWTY will attempt to calculate it automatically.
trim	Used for thinning the chain. If a number N is provided, RWTY keeps every Nth tree.
logfile	A path to a file containing model parameters and likelihoods. If no path is provided but a "format" argument is supplied, RWTY will attempt to find the log file automatically based on the format definition.
skip	The number of lines that must be skipped to get to the header of the log file. MrBayes, for instance, prints a comment line at the top of the log file, so MrBayes files should be read in with a skip value of 1. If no "skip" value is provided but a "format" is supplied, RWTY will attempt to read logs using the skip value from the format definition.

Value

output An rwty.chain object containing the multiPhylo and the table of values from the log file if available.

See Also

read.tree, read.nexus

Examples

```
#load.trees(file="mytrees.t", format = "mb")
```

---

makeplot.acsf.cumulative

Plot the Change in Split Frequencies (CSF) in sliding windows over the course of an MCMC.

Description

This function takes one or more rwty.chain ojects and returns a plot of CSF within each chain as the MCMC progresses. The solid line with points shows the Average Change in Split Frequencies (ACSF; it is average across the changes in split frequencies from all clades in the analysis) between this window and the previous window The grey ribbon shows the upper and lower 95

Usage

```
makeplot.acsf.cumulative(chains, burnin = 0, window.size = 20, facet = TRUE)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin. Defaults to zero.
window.size	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)
facet	(TRUE/FALSE). TRUE: return a single plot with one facet per chain; FALSE: return a list of individual plots with one plot per chain

Value

output A plof of the CSF between sliding windows over all chains

acsf.plot A ggplot object, or list of ggplot objects

Examples

```
## Not run:
data(fungus)
makeplot.acsf.cumulative(fungus, burnin=20)

## End(Not run)
```

---

makeplot.acsf.sliding *Plot the Chaing in Split Frequencies (CSF) in sliding windows over the course of an MCMC.*

Description

This function takes one or more rwty.chain ojects and returns a plot of CSF within each chain as the MCMC progresses. The solid line with points shows the Average Change in Split Frequencies (ACSF) between this window and the previous window The grey ribbon shows the upper and lower 95

Usage

```
makeplot.acsf.sliding(chains, burnin = 0, window.size = 20, facet = TRUE)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin. Defaults to zero.
window.size	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)
facet	(TRUE/FALSE). TRUE: return a single plot with one facet per chain; FALSE: return a list of individual plots with one plot per chain

Value

output A plof of the CSF between sliding windows over all chains

acsf.plot A ggplot object, or list of ggplot objects

Examples

```
## Not run:
data(fungus)
makeplot.acsf.sliding(fungus, burnin=20)

## End(Not run)
```

`makeplot.all.params` *Plotting all parameters*

Description

Plots all parameter values, including tree topologies (see makeplot.topology) over the length of the MCMC chain

Usage

```
makeplot.all.params(
  chains,
  burnin = 0,
  facet = TRUE,
  free_y = FALSE,
  strip = 1
)
```

Arguments

`chains`	A set of rwty.chain objects
`burnin`	The number of trees to omit as burnin.
`facet`	Boolean denoting whether to make a facet plot.
`free_y`	TRUE/FALSE to turn free y scales on the facetted plots on or off (default FALSE). Only works if facet = TRUE.
`strip`	Number indicating which column to strip off (i.e., strip=1 removes first column, which is necessary for most MCMC outputs in which the first column is just the generation). You can skip multiple columns by passing a vector of columns to skip, e.g., strip=c(1,4,6).

Value

param.plot Returns a list of ggplot objects.

Examples

```
## Not run:
data(fungus)
makeplot.all.params(fungus, burnin=20)

## End(Not run)
```

`makeplot.asdsf` *Plot the Standard Deviation of Split Frequencies over the course of an MCMC.*

Description

This function takes two or more rwty.chain ojects and returns a plot of ASDSF as the run progresses. The solid line with points shows the Average Standard Deviation of Split Frequences at the current generation The grey ribbon shows the upper and lower 95

Usage

```
makeplot.asdsf(
  chains,
  burnin = 0,
  window.size = 20,
  min.freq = 0,
  log.y = TRUE
)
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin. Defaults to zero.
`window.size`	The number of trees between each point at which the ASDSFs is calculated (note, specified as a number of sampled trees, not a number of generations)
`min.freq`	The minimum frequency for a node to be used for calculating ASDSF.
`log.y`	Controls whether they Y axis is plotted on a log scale or not. Which scale is more useful depends largely on the amount of disagreement between your chains. Attempting to make an asdsf plot with a log Y axis for chains that include standard deviations of zero will result in warning messages.

Value

output A cumulative plot of ASDSF across all chains

Examples

```
## Not run:
data(fungus)
p <- makeplot.asdsf(fungus, burnin = 20)
p

## End(Not run)
```

`makeplot.autocorr` *Make autocorrelation plots of tree topologies from MCMC analyses*

Description

This function takes a list of rwty.chain objects, and makes an autocorrelation plot for each chain. Each plot shows the mean phylogenetic distance at a series of sampling intervals. In really well behaved MCMC analyses, the mean distance will stay constant as the sampling interval increases. If there is autocorrelation, the mean distance will increase as the sampling interval increases, and is expected to level off when the autocorrelation decreases to zero. The function calculates path distances, though other distances could also be employed.

Usage

```
makeplot.autocorr(
  chains,
  burnin = 0,
  max.sampling.interval = NA,
  autocorr.intervals = 40,
  squared = FALSE,
  facet = FALSE,
  free_y = FALSE,
  treedist = "PD",
  use.all.samples = FALSE
)
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin.
`max.sampling.interval`	The largest sampling interval for which you want to calculate the mean distance between pairs of trees (default is 10 percent of the length of the chain).
`autocorr.intervals`	The number of sampling intervals to use. These will be spaced evenly between 1 and the max.sampling.interval
`squared`	TRUE/FALSE use squared tree distances (necessary to calculate approximate ESS; default FALSE)

facet	TRUE/FALSE to turn facetting of the plot on or off (default FALSE)
free_y	TRUE/FALSE to turn free y scales on the facetted plots on or off (default FALSE). Only works if facet = TRUE.
treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance
use.all.samples	(TRUE/FALSE). Whether to calculate autocorrelation from all possible pairs of trees in your chain. The default is FALSE, in which case 500 samples are taken at each sampling interval. This is sufficient to get reasonably accurate estimates of the approximate ESS. Setting this to TRUE will give you slightly more accurate ESS estimates, at the cost of potentially much longer execution times.

Value

A ggplot2 plot object, with one line (facetting off) or facet (facetting on) per rwty.chain object.

Examples

```
## Not run:
data(fungus)
makeplot.autocorr(fungus, burnin = 20)

## End(Not run)
```

---

makeplot.pairs *Plotting parameters against each other*

Description

Makes a plot matrix of each parameter against each other (including the topology) in your analysis. The default behaviour is to just plot the first two columns of your parameter file (after removing the column for the generation number) as well as the topological distance. This usually means that you see a pairs plot with the likelihood, the tree length, and the tree toppology. We do this because some parameter files contain so many columns that the plot matrix becomes too busy. To include parameters of your choice, use the 'parameters' argument. In this function, the topological distance is calculate from the first tree in every chain.

Usage

```
makeplot.pairs(chains, burnin = 0, treedist = "PD", params = NA, strip = 1)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to omit as burnin.
treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance
params	'NA', 'all', or a vector of column names to include in the plot. 'NA' gives the default behaviour (see above). 'all' plots all columns (watch out!). Choose specific columns by name with a vector.
strip	Number indicating which column to strip off (i.e., strip=1 removes first column, which is necessary for most MCMC outputs in which the first column is just the generation).

Value

pairs.plot Returns a ggplot object.

Examples

```
## Not run:
data(salamanders)
makeplot.pairs(salamanders[1], burnin=20)

# plot all the variables
makeplot.pairs(salamanders[1], burnin=20, params = 'all')

# plot specific the variables (note: you always get the topological distance)
makeplot.pairs(salamanders[1], burnin=20, params = c('pi.A.', 'pi.C.', 'pi.G.', 'pi.T.'))

## End(Not run)
```

---

makeplot.param *Plotting parameters*

Description

Plots parameter values over the length of the MCMC chain

Usage

```
makeplot.param(
  chains,
  burnin = 0,
  parameter = "LnL",
  facet = TRUE,
  free_y = FALSE
)
```

Arguments

chains	A set of rwty.chain objects.
burnin	The number of trees to omit as burnin.
parameter	The column name of the parameter to plot.
facet	Boolean denoting whether to make a facet plot.
free_y	TRUE/FALSE to turn free y scales on the facetted plots on or off (default FALSE). Only works if facet = TRUE.

Value

param.plot Returns a ggplot object.

Examples

```
## Not run:
data(fungus)
makeplot.param(fungus, burnin=20, parameter="pi.A.")

## End(Not run)
```

---

makeplot.pseudo.ess *Plot the pseudo ESS of tree topologies from MCMC chains.*

Description

This function takes a list of rwty.chain objects, and plots the pseudo ESS of the tree topologies from each chain, after removing burnin. Each caulcation is repeated n times, where in each replicate a random tree from the chain is chosen as a 'focal' tree. The calculation works by calculating the path distance of each tree in the chain from the focal tree, and calculating the ESS of the resulting vector of phylogenetic distances using the effectiveSize function from the coda package. NB this function requires the calculation of many tree distances, so can take some time.

Usage

```
makeplot.pseudo.ess(chains, burnin = 0, n = 20)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin
n	The number of replicate analyses to do

Value

pseudo.ess.plot A ggplot2 plot object, in which each chain is represented by a point which represents the median pseudo ESS from the n replicates, and whiskers representing the upper and lower 95

Examples

```
## Not run:
data(fungus)
makeplot.pseudo.ess(fungus, burnin = 20, n = 10)

## End(Not run)
```

`makeplot.splitfreq.matrix` *Plots a matrix of split frequency comparisons between multiple MCMC chains.*

Description

This function takes list of rwty.chain objects, and returns a scatterplot matrix in which each plot shows the split frequencies of all clades that appear in one or both MCMC chains at least once. In the upper diagonal, we show the correlation between the split frequencies (Pearson's R), and the Average Standard Deviation of the split frequencies.

Usage

```
makeplot.splitfreq.matrix(chains, burnin = 0)
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin

Value

output A list of two plots: the first is a matrix of scatterplots, where each point is a clade, and the values are the split frequencies of that clade in the post-burnin trees of each chain. The second plot is a tree of the chains clustered by their ASDSFs.

Examples

```
## Not run:
data(salamanders)
makeplot.splitfreq.matrix(salamanders[1:4], burnin = 20)

## End(Not run)
```

`makeplot.splitfreqs.cumulative` *Plot cumulative split frequencies over the course of an MCMC*

Description

Takes a list of rwty.chain objects. Plots the cumulative split frequencies of clades over the course of the MCMC. Stationarity is indicated by split frequencies levelling out. Only plots the n.clades most variable clades, as measured by the standard deviation of the split frequencies of each clade across all windows. Each line in the plot represents a single clade. The colour of the line represents the standard deviation of the split frequencies of that clade across all sliding windows.

Usage

```
makeplot.splitfreqs.cumulative(
  chains,
  burnin = 0,
  n.clades = 20,
  window.size = 20,
  facet = TRUE,
  rank = "wcsf"
)
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin
`n.clades`	The number of clades to plot
`window.size`	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)
`facet`	(TRUE/FALSE). TRUE: return a single plot with one facet per chain; FALSE: return a list of individual plots with one plot per chain
`rank`	('wcsf', 'sd'). How to rank the clades? By default, we plot the 20 'worst' clades. This parameter sets the definition of 'worst'. The default is to rank the by the weighted change in split frequencies (rank = 'wcsf'). This works by looking at the change in the cumulative split frequency over the course of the MCMC, and ranks the worst chains as those that do not level off (i.e. those that have changes near the end). We do this because in a well-behaved chain, we expect the cumulative split frequencies to level off once the chain has been run for long enough. So, any cumulative split frequencies which are still changing towards the end of your run are likely to indicate problematic clades. Specifically, we multiply the absolute change in split frequencies for each clade by a set of weights that increase linearly towards the end of the chain (the first observation gets a weight of zero, the final observation gets a weight of one). The original AWTY ranked clades by their standard deviations (higher SD = worse), so we include this as an option too. To do this, just set rank = 'sd'.

Value

splitfreqs.plot Either a single ggplot2 object or a list of ggplot2 objects.

Examples

```
## Not run:
data(fungus)
makeplot.splitfreqs.cumulative(fungus, burnin = 20, n.clades=25)

## End(Not run)
```

`makeplot.splitfreqs.sliding`
Plot split frequencies in sliding windows over the course of an MCMC

Description

Takes a list of rwty.chain objects. Plots the split frequencies of clades over the course of the MCMC, calculated from windows of a specified size. Only plots the n.clades most variable clades, as measured by the standard deviation of the split frequencies of each clade across the MCMC. Each line in the plot represents a single clade. The colour of the line represents the standard deviation of the split frequencies of that clade across the MCMC.

Usage

```
makeplot.splitfreqs.sliding(
  chains,
  burnin = 0,
  n.clades = 20,
  window.size = 20,
  facet = TRUE,
  rank = "ess"
)
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin
`n.clades`	The number of clades to plot
`window.size`	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)
`facet`	(TRUE/FALSE). TRUE: return a single plot with one facet per chain; FALSE: return a list of individual plots with one plot per chain

rank ('ess', 'sd'). How to rank the clades? By default, we plot the 20 'worst' clades. This parameter sets the definition of 'worst'. The default is to rank the clades by increasing Effective Sample Size (i.e. the 20 worst clades are those with the lowest ESS), since in a sliding window plot we expect well-sampled splits to have a high value (rank = "ess"). The original AWTY ranked clades by their standard deviations. To do this, just set rank = 'sd'.

Value

splitfreqs.plot Either a single ggplot2 object or a list of ggplot2 objects.

Examples

```
## Not run:
data(fungus)
makeplot.splitfreqs.sliding(fungus, burnin = 20, n.clades=25)

## End(Not run)
```

---

makeplot.topology *Plotting parameters*

Description

Plots a trace of topological distances of trees over the length of the MCMC chain. The plot shows the path distance of each tree in each chain from the last tree of the burnin of the first chain. If burnin is set to zero, then distances are calculated from the first tree of the first chain. If required, the behaviour can be changed to plot the path distance of each tree from the last tree of the burnin of each chain, using the independent.chains option. This is not recommended in most cases.

Usage

```
makeplot.topology(
  chains,
  burnin = 0,
  facet = TRUE,
  free_y = FALSE,
  independent.chains = FALSE,
  treedist = "PD",
  approx.ess = TRUE
)
```

Arguments

chains	A set of rwty.chain objects.
burnin	The number of trees to omit as burnin.
facet	TRUE/FALSE denoting whether to make a facet plot (default TRUE)

free_y	TRUE/FALSE to turn free y scales on the facetted plots on or off (default FALSE). Only works if facet = TRUE.
independent.chains	TRUE/FALSE if FALSE (the default) then the plots show the distance of each tree from the last tree of the burnin of the first chain. If TRUE, the plots show the distance of each tree from the first tree of the chain in which that tree appears. The TRUE option should only be used in the case that different chains represent analyses of different genes or datasets.
treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance
approx.ess	TRUE/FALSE do you want the approximate topological ess to be calculated and displayed for each chain?

Value

topology.trace.plot Returns a ggplot object.

Examples

```
## Not run:
data(fungus)
makeplot.topology(fungus, burnin=20)

## End(Not run)
```

---

makeplot.treespace *Plot chains in treespace.*

Description

This function will take list of rwty.chains objects and produce plots of chains in treespace.

Usage

```
makeplot.treespace(chains, burnin = 0, n.points = 100, fill.color = NA)
```

Arguments

chains	A list of one or more rwty.chain objects
burnin	The number of samples to remove from the start of the chain as burnin
n.points	The number of points on each plot
fill.color	The name of any column in your parameter file that you would like to use as a fill colour for the points of the plot.

Value

A list of two ggplot objects: one plots the points in treespace, the other shows a heatmap of the same points

Examples

```
## Not run:
data(fungus)

p <- makeplot.treespace(fungus, burnin = 20, fill.color = 'LnL')
# Treespace plot for all the fungus data

# NB: these data indicate significant problems: the chains are sampling very
# different parts of tree space.
#
# View the points plotted in treespace (these data indicate significant problems)
p$treespace.points.plot

# View the heatmap of the same data
# Note that this data is so pathologically bad that the heatmap is not
# very useful. It is more useful on better behaved datasets
p$treespace.heatmap

# we can also plot different parameters as the fill colour.
# e.g. we can plot the first two fungus chains with likelihood as the fill
makeplot.treespace(fungus[1:2], burnin = 100, fill.color = 'LnL')

# or with tree length as the fill
makeplot.treespace(fungus[1:2], burnin = 100, fill.color = 'TL')

# you can colour the plot with any parameter in your ptable
# to see which parameters you have you can simply do this:
names(fungus[[1]]$ptable)

## End(Not run)
```

`parse.clades` *Rename clades for easy recall*

Description

Converts a list of clades (e.g., "1 2 3 4" as a clade) and returns a list of parsed clades, converting numbers to names using a set of trees. Called internally by the slide and cumulative analyses, not user-facing.

Usage

```
parse.clades(clades, treelist)
```

Arguments

`clades`	A list of clades, as in the first column of a cladetable in an rwty.slide or rwty.cumulative object.
`treelist`	A list of trees, used for getting tip names.

Value

output A list of clades with parsed tip names

`parse.slide.clades` *Rename clades for easy recall*

Description

Converts a list of clades (e.g., "1 2 3 4" as a clade) and returns a list of parsed clades, converting numbers to names using a set of trees. Called internally by the slide and cumulative analyses, not user-facing.

Usage

```
parse.slide.clades(clades, treelist)
```

Arguments

`clades`	A list of clades, as in the first column of a cladetable in an rwty.slide or rwty.cumulative object.
`treelist`	A list of trees, used for getting tip names.

Value

output A list of clades with parsed tip names

`print.rwty.chain` *Function for printing rwty.chain objects*

Description

This function is automatically called when printing a rwty.chain object

Usage

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

Arguments

`x`	A rwty.chain object
`...`	Other arguments to be passed on to next function

Value

A summary of the contents of the chain

Examples

```
data(fungus)
fungus$Fungus.Run1
```

`salamanders` *MrBayes output from analysis of Williams et al. data*

Description

This is the output from a MrBayes run of 25,000,000 generations using the analysis settings from the original .nex files. Sampling is one tree per 100,000 generations. Data is from alignments of three separate sequences, two chains per alignment, each with its associated log file.

Usage

```
data(salamanders)
```

Format

A data frame with six chains (two each from three separate alignments) of 251 phylogenetic trees and associated likelihood and parameter values.

References

Study reference: Williams JS, Niedzwiecki JH, Weisrock DW (2013) Species tree reconstruction of a poorly resolved clade of salamanders (Ambystomatidae) using multiple nuclear loci. Molecular Phylogenetics and Evolution 68(3): 671-682. http://dx.doi.org/10.1016/j.ympev.2013.04.013

Dryad reference: Williams JS, Niedzwiecki JH, Weisrock DW (2013) Data from: Species tree reconstruction of a poorly resolved clade of salamanders (Ambystomatidae) using multiple nuclear loci. Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.2gq14

http://datadryad.org/resource/doi:10.5061/dryad.2gq14

`slide.freq` *Sliding window measurements of clade split frequencies.*

Description

This function takes sliding windows of a specified length over an MCMC chain and calculates the split frequency of clades within that window. It allows users to see whether the chain is visiting different areas of treespace.

Usage

```
slide.freq(chains, burnin = 0, window.size = 20)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin. Defaults to zero.
window.size	The number of trees to include in each window (note, specified as a number of sampled trees, not a number of generations)

Value

A list of rwty.slide objects, one per chain in the input list of chains. Each rwty.slide object contains the frequencies of clades in the sliding windows, and a translation table that converts clade groupings to factors.

Examples

```
## Not run:
data(fungus)
slide.data <- slide.freq(fungus, burnin=20)\

## End(Not run)
```

---

topological.approx.ess

Calculate the approximate Effective Sample Size (ESS) of tree topologies

---

Description

This function takes a list of rwty.chain objects, and calculates the pseudo ESS of the trees from each chain, after removing burnin. The calculation uses the autocorrelation among squared topological distances between trees to calculate an approximate ESS of tree topologies for each chain. NB this function requires the calculation of many many tree distances, so can take some time.

Usage

```
topological.approx.ess(
  chains,
  burnin = 0,
  max.sampling.interval = 100,
  treedist = "PD",
  use.all.samples = FALSE
)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin
max.sampling.interval	The largest sampling interval you want to use to calculate the ESS. Every sampling interval up to and including this number will be sampled. Higher is better, but also slower. In general, setting this number to 100 (the default) should be fine for most cases. However, if you get an upper bound on the ESS estimate (i.e. ESS<x) rather than a point estimate (i.e. ESS = x) then that indicates a higher max.sampling.interval would be better, because the algorithm could not find the asymptote on the autocorrelation plot with the current max.sampling.interval.
treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance
use.all.samples	(TRUE/FALSE). Whether to calculate autocorrelation from all possible pairs of trees in your chain. The default is FALSE, in which case 500 samples are taken at each sampling interval. Setting this to TRUE will give you slightly more accurate ESS estimates, at the cost of potentially much longer execution times.

Value

A data frame with one row per chain, and columns describing the approximate ESS and the name of the chain.

Examples

```
## Not run:
data(fungus)
topological.approx.ess(fungus, burnin = 20)

## End(Not run)
```

---

topological.autocorr *Calculate data for autocorrelation plots of tree topologies from MCMC analyses*

Description

This function takes a list of rwty.chain objects, and calculates the mean phylogenetic distance at a series of roughly even sampling intervals. In really well behaved MCMC analyses, the mean distance will stay constant as the sampling interval increases. If there is autocorrelation, it will increase as the sampling interval increases, and is expected to level off when the autocorrelation decreases to zero. The function calculates path distances, though other distances could also be employed.

Usage

```
topological.autocorr(
  chains,
  burnin = 0,
  max.sampling.interval = NA,
  autocorr.intervals = 100,
  squared = FALSE,
  treedist = "PD",
  use.all.samples = FALSE
)
```

Arguments

chains	A list of rwty.chain objects.
burnin	The number of trees to eliminate as burnin
max.sampling.interval	The largest sampling interval for which you want to calculate the mean distance between pairs of trees (default is 10 percent of the length of the list of trees).
autocorr.intervals	The number of sampling intervals to use. These will be spaced evenly between 1 and the max.sampling.interval
squared	TRUE/FALSE use squared tree distances (necessary to calculate approximate ESS)
treedist	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance
use.all.samples	(TRUE/FALSE). Whether to calculate autocorrelation from all possible pairs of trees in your chain. The default is FALSE, in which case 500 samples are taken at each sampling interval. This is sufficient to get reasonably accurate estimates of the approximate ESS. Setting this to TRUE will give you slightly more accurate ESS estimates, at the cost of potentially much longer execution times.

Value

A data frame with one row per sampling interval, per chain. The first column is the sampling interval. The second column is the mean path distance between pairs of trees from that sampling interval. The third column is the chain ID.

Examples

```
## Not run:
data(fungus)
topological.autocorr(fungus, burnin = 20)

## End(Not run)
```

`topological.pseudo.ess` *Calculate the pseudo Effective Sample Size (ESS) of tree topologies*

Description

This function takes a list of rwty.chain objects, and calculates the pseudo ESS of the trees from each chain, after removing burnin. Each caulcation is repeated n times, where in each replicate a random tree from the chain is chosen as a 'focal' tree. The calculation works by calculating the path distance of each tree in the chain from the focal tree, and calculating the ESS of the resulting vector of phylogenetic distances using the effectiveSize function from the coda package. NB this function requires the calculation of many many tree distances, so can take some time.

Usage

```
topological.pseudo.ess(chains, burnin = 0, n = 20, treedist = "PD")
```

Arguments

`chains`	A list of rwty.chain objects.
`burnin`	The number of trees to eliminate as burnin
`n`	The number of replicate analyses to do
`treedist`	the type of tree distance metric to use, can be 'PD' for path distance or 'RF' for Robinson Foulds distance

Value

A data frame with one row per chain, and columns describing the median ESS, the upper and lower 95 replicates performed, and the name of the chain.

Examples

```
## Not run:
data(fungus)
topological.pseudo.ess(fungus, burnin = 20, n = 20)

## End(Not run)
```

`tree.dist.matrix` *Tree distance matrix calculation*

Description

This function takes a list of trees and returns a distance matrix populated with distances between all trees in the list.

Usage

```
tree.dist.matrix(trees, treedist = "rf", ...)
```

Arguments

`trees`	an object of class 'multiPhylo'.
`treedist`	acronym of distance method to employ: one of `cid`, `icrf`, `jrf`, `mast`, `masti`, `ms`, `msid`, `nni`, `pd`, `pid`, `rf` (default), or `spr`. See below for details.
`...`	additional parameters sent to distance functions.

Value

a matrix of distances between each pair of trees

Recommended methods

A suite of distance metrics are implemented, offering a trade-off between running time and suitability of metric. Ranked according to their running time with 251 85-tip trees on a low-spec desktop computer, recommended distance metrics are:

- `rf` (0.4 seconds): Robinson-Foulds distance (Robinson & Foulds, 1981): although widely used, the RF metric has a series of theoretical shortcomings that give rise to bias and artefacts, translating to poor performance in a suite of practical applications. Its low resolution and rapid saturation make it particularly unsuitable for operations in tree space. Nevertheless, its speed is hard to match.

- `pd` (1 s): The path (= cladistic / nodal / patristic / tip) distance (Farris 1973) can also be calculated rapidly, but is heavily influenced by the shape (e.g. balanced / unbalanced) of trees, meaning that similar-looking trees that nevertheless denote very different sets of relationships will have shorter distances than may be anticipated. Consequently, the path metric does a poor job of identifying clusters of similar trees.

- `icrf` (1.4 s): Robinson-Foulds distance, corrected for split size using information theory (Smith 2020). This measure adjusts the Robinson-Foulds distance to account for the different significance of different partitions: partitions that evenly divide taxa contain more information, and thus should contribute more to a distance score if they are not shared between trees. This adjustment improves the resolution and sensitivity of the metric, but does not correct for a number of arguably more significant biases.

- `pid` (4 s); `msid` (8 s); `cid` (30 s): Phylogenetic information distance, matching split information distance and clustering information distance (Smith 2020). These information-theoretic methods belong to the class of Generalized Robinson-Foulds distances: by recognizing similarities between

non-identical splits, they overcome many of the artefacts that affect the RF distance, providing a more representative measure of tree distances; whereas their information-theoretic basis affords them a natural unit (the bit), providing a measurable dimension to tree space. Whilst the CID performs the best against a suite of theoretical and practical criteria, the MSID comes a very close second and is somewhat quicker to calculate. The PID will provide unexpectedly large distances in a subset of the cases that distort the RF metric, which may result in undesirable distortions of an accompanying tree space.

Detailed analysis of the behaviour of these and other tree distance methods against a suite of criteria is available in Smith (2020); implementation details are provided in the R package 'TreeDist'.

Further methods

A further set of methods that underperform methods with similar running time listed above are also implemented for comparative purposes:

- `mast`, `masti` (30 minutes): size / information content of the maximum agreement forest, subtracted from its maximum possible value to create a distance. Specify 'rooted = FALSE' if trees are unrooted.

- `jrf` (1 min, k = 2; 25 min, conflict-ok; 4 h, k = 4, no-conflict): Jaccard Robinson-Foulds metric (Böcker et al. 2013); specify a value of `k` and `allowConflict` using ....

- `ms` (5 s): Matching split distance (Bogdanowicz and Giaro 2012; Lin et al. 2012.

- `nye` (65 s): The generalized RF distance of Nye et al. (2006).

- `nni` (65 s): Approximate Nearest Neighbour Interchance (rotation) distance.

- `spr` (0.4 s): Approximate Subtree Prune and Regraft distance.

References

Böcker S, Canzar S, Klau GW (2013). "The generalized Robinson-Foulds metric." In Darling A, Stoye J (eds.), Algorithms in Bioinformatics. WABI 2013. Lecture Notes in Computer Science, vol 8126, 156–169. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-40453-5_13.

Bogdanowicz D, Giaro K (2012). "Matching split distance for unrooted binary phylogenetic trees." IEEE/ACM Transactions on Computational Biology and Bioinformatics, 9(1), 150–160. doi: 10.1109/TCBB.2011.48.

Farris JS (1973). "On comparing the shapes of taxonomic trees." Systematic Zoology, 22(1), 50–54. doi: 10.2307/2412378.

Lin Y, Rajan V, Moret BME (2012). "A metric for phylogenetic trees based on matching." IEEE/ACM Transactions on Computational Biology and Bioinformatics, 4(9), 1014–1022. doi: 10.1109/TCBB.2011.157.

Nye TMW, Liò P, Gilks WR (2006). "A novel algorithm and web-based tool for comparing two alternative phylogenetic trees." Bioinformatics, 22(1), 117–119. doi: 10.1093/bioinformatics/bti720.

Robinson DF, Foulds LR (1981). "Comparison of phylogenetic trees." Mathematical Biosciences, 53(1-2), 131–147. doi: 10.1016/0025-5564(81)90043-2.

Smith MR (2020). "Information theoretic Generalized Robinson-Foulds metrics for comparing phylogenetic trees." Bioinformatics, in production. doi: 10.1093/bioinformatics/btaa614.

Examples

```
## Not run:
data(fungus)
tree.dist.matrix(fungus$Fungus.Run1$trees)

## End(Not run)
```

`treespace` *MDS scaling of treespace for a single tree list.*

Description

This function constructs a distance matrix from a list of trees and uses multi-dimensional scaling to collapse it to a two- dimensional tree space for plotting.

Usage

```
treespace(chains, n.points = 100, burnin = 0, fill.color = NA)
```

Arguments

`chains`	A list of 1 or more rwty.chain objects.
`n.points`	The minimum number of points you want in your plot.
`burnin`	The number of trees to eliminate as burnin. Default is zero.
`fill.color`	The name of the column from the log table that that you would like to use to colour the points in the plot.

Value

Returns a list containing the points and a plot.

Examples

```
## Not run:
data(fungus)
treespace(fungus, n.points=50, burnin=20, fill.color="LnL")

## End(Not run)
```

Index

∗ **ASDSF**
  makeplot.asdsf, 13
  makeplot.splitfreq.matrix, 18
∗ **Clade**
  clade.freq, 5
∗ **ESS**
  analyze.rwty, 2
  makeplot.pseudo.ess, 17
∗ **MCMC**
  analyze.rwty, 2
  check.chains, 4
  combine.ptables, 6
  cumulative.freq, 6
  load.multi, 8
  load.trees, 9
  makeplot.asdsf, 13
  makeplot.splitfreq.matrix, 18
  print.rwty.chain, 24
  slide.freq, 25
∗ **Phylogenetics**
  load.multi, 8
  load.trees, 9
∗ **autocorrelation**
  estimate.autocorr.m, 7
  makeplot.autocorr, 14
  topological.autocorr, 27
∗ **awty**
  check.chains, 4
  combine.ptables, 6
  print.rwty.chain, 24
∗ **clade**
  makeplot.splitfreq.matrix, 18
∗ **consensus**
  clade.freq, 5
  makeplot.splitfreq.matrix, 18
∗ **convergence**
  check.chains, 4
  combine.ptables, 6
  cumulative.freq, 6
  makeplot.acsf.cumulative, 10
  makeplot.acsf.sliding, 11
  makeplot.all.params, 12
  makeplot.pairs, 15
  makeplot.param, 16
  makeplot.splitfreq.matrix, 18
  makeplot.topology, 21
  print.rwty.chain, 24
  slide.freq, 25
∗ **cumulative**
  makeplot.asdsf, 13
∗ **datasets**
  fungus, 8
  salamanders, 25
∗ **distance**
  estimate.autocorr.m, 7
  makeplot.autocorr, 14
  makeplot.pseudo.ess, 17
  topological.approx.ess, 26
  topological.autocorr, 27
  topological.pseudo.ess, 29
∗ **frequency**
  clade.freq, 5
  cumulative.freq, 6
  makeplot.splitfreq.matrix, 18
∗ **load**
  load.multi, 8
  load.trees, 9
∗ **mcmc**
  clade.freq, 5
  makeplot.acsf.cumulative, 10
  makeplot.acsf.sliding, 11
  makeplot.all.params, 12
  makeplot.pairs, 15
  makeplot.param, 16
  makeplot.splitfreqs.cumulative, 19
  makeplot.splitfreqs.sliding, 20
  makeplot.topology, 21
∗ **mds**

treespace, 32
* **parameter**
makeplot.all.params, 12
makeplot.pairs, 15
makeplot.param, 16
makeplot.topology, 21
* **path**
estimate.autocorr.m, 7
makeplot.autocorr, 14
* **phylogenetics**
check.chains, 4
clade.freq, 5
combine.ptables, 6
makeplot.acsf.cumulative, 10
makeplot.acsf.sliding, 11
makeplot.all.params, 12
makeplot.asdsf, 13
makeplot.pairs, 15
makeplot.param, 16
makeplot.splitfreq.matrix, 18
makeplot.splitfreqs.cumulative, 19
makeplot.splitfreqs.sliding, 20
makeplot.topology, 21
print.rwty.chain, 24
* **plots**
analyze.rwty, 2
* **plot**
makeplot.all.params, 12
makeplot.pairs, 15
makeplot.param, 16
makeplot.splitfreqs.cumulative, 19
makeplot.splitfreqs.sliding, 20
makeplot.topology, 21
makeplot.treespace, 22
print.rwty.chain, 24
* **robinson-foulds**
tree.dist.matrix, 30
* **rwty**
analyze.rwty, 2
check.chains, 4
combine.ptables, 6
makeplot.treespace, 22
print.rwty.chain, 24
* **split**
cumulative.freq, 6
* **topology**
analyze.rwty, 2
* **tree-distance**
tree.dist.matrix, 30
* **treespace**
makeplot.treespace, 22
topological.approx.ess, 26
topological.pseudo.ess, 29
tree.dist.matrix, 30
treespace, 32
* **trees**
load.multi, 8
* **uncertainty**
makeplot.acsf.cumulative, 10
makeplot.acsf.sliding, 11

analyze.rwty, 2

check.chains, 4
clade.freq, 5
combine.ptables, 6
cumulative.freq, 6

estimate.autocorr.m, 7

fungus, 8

load.multi, 8
load.trees, 9

makeplot.acsf.cumulative, 10
makeplot.acsf.sliding, 11
makeplot.all.params, 12
makeplot.asdsf, 13
makeplot.autocorr, 14
makeplot.pairs, 15
makeplot.param, 16
makeplot.pseudo.ess, 17
makeplot.splitfreq.matrix, 18
makeplot.splitfreqs.cumulative, 19
makeplot.splitfreqs.sliding, 20
makeplot.topology, 21
makeplot.treespace, 22

parse.clades, 23
parse.slide.clades, 24
print.rwty.chain, 24

read.nexus, *9, 10*
read.tree, *9, 10*

salamanders, 25
slide.freq, 25

topological.approx.ess, 26
topological.autocorr, 27
topological.pseudo.ess, 29
tree.dist.matrix, 30
treespace, 32