Package 'tbea'

July 1, 2024

Title Pre- And Post-Processing in Bayesian Evolutionary Analyses

Version 1.5.0

Description Functions are provided for prior specification in divergence time estimation using fossils as well as other kinds of data. It provides tools for interacting with the input and output of Bayesian platforms in evolutionary biology such as 'BEAST2', 'MrBayes', 'RevBayes', or 'MCMCTree'. It Implements a simple measure similarity between probability density functions for comparing prior and posterior Bayesian densities, as well as code for calculating the combination of distributions using conflation of Hill (2008). Functions for estimating the origination time in collections of distributions using the x-intercept (e.g., Draper and Smith, 1998) and stratigraphic intervals (Marshall 2010) are also available. Hill, T. 2008. ``Conflations of probability distributions''. Transactions of the American Mathematical Society, 363:3351-3372. <doi:10.48550/arXiv.0808.1808>, Draper, N. R. and Smith, H. 1998. ``Applied Regression Analysis''. 1--706. Wiley Interscience, New York. <DOI:10.1002/9781118625590>, Marshall, C. R. 2010. ``Using confidence intervals to quantify the uncertainty in the end-points of stratigraphic ranges''. Quantitative Methods in Paleobiology, 291--316. <DOI:10.1017/S1089332600001911>.

Depends R (>= 3.5.0)

Imports ape, Rfit, boot, graphics, stats, utils, coda

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

URL https://github.com/gaballench/tbea

BugReports https://github.com/gaballench/tbea/issues

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Gustavo A. Ballen [aut, cre],
Sandra Reinales [aut]

Maintainer Gustavo A. Ballen <gustavo.a.ballen@gmail.com>

Repository CRAN

Date/Publication 2024-07-01 16:20:01 UTC

Contents

`andes` *Divergence-time estimation data for cis-trans-Andean pairs*

Description

A dataset containing point estimates and uncertainty intervals of divergence times for clade pairs east and west of the Andes, compiled by Ballen (2020).

Usage

```
data(andes)
```

Format

A data frame with three columns:

ages Estimated age (in Ma) from a given rock sample

min Standard deviation of the age estimate

max Sample code as in Table 3.2

References

Ballen, Gustavo A. 2020. Fossil freshwater fishes and the biogeography of northern South America. 2020. PhD thesis, Museu de Zoologia, Universidade de São Paulo, São Paulo, 2020. doi:10.11606/T.38.2020.tde-06052020-181631.

`concatNexus` *concatNexus: Function for concatenation of nexus matrices both morphological and molecular*

Description

concatNexus: Function for concatenation of nexus matrices both morphological and molecular

Usage

```
concatNexus(
  matrices = NULL,
  pattern,
  path,
  filename,
  morpho = FALSE,
  morphoFilename = NULL,
  sumFilename
)
```

Arguments

`matrices`	A vector of type 'character' with paths to the nexus alignments or their file names. If `morphoFilename` is non-null, either the path to the morphological partition or its file name must be included too. The default is NULL and it must be defined if none of `pattern` and `path` are included.
`pattern`	A vector of type 'character' and length one containing the text pattern to identify the alignments of interest. It would be tipically be some suffix and/or file extension (see examples).
`path`	A vector of type 'character' and length one pointing to the directory where the matrices are located. It is used in combination with `pattern` in order to build a path to each matrix file (see examples).

filename	A vector of type 'character' and length one with the file name (or path and file name) for the concatenated output matrix.
morpho	A vector of type 'logical' and length one indicating whether a morphological matrix is included in the concatenation.
morphoFilename	A vector of type 'character' and length one with the file name or path to the morphological nexus matrix. Needed if morpho = TRUE.
sumFilename	A vector of type 'character' and length one with the file name or path to the summary information of partition start and end positions. Useful for specifying concatenated analyses in MrBayes where each partition in the matrix might have its own substitution model.

Details

This function will concatenate matrices in nexus format (mandatory) and write to the disk the output and summary information on the partitions. It requires that the input matrices all share the same taxa in the same positions.

Value

This function writes to the disk two files, one with the concatenated matrix and one with the summary information on partition positions in the complete matrix.

Author(s)

Gustavo A. Ballen

Examples

```
# Concatenate all the matrices in a given path,
# ending with the pattern 'aligned.nex', including a morphological matrix
# also defined with a pattern
## Not run:
path <- "sequences"
pattern <- "aligned.nex$"

concatNexus(matrices = NULL, pattern = pattern,
            filename = paste(path, "concatenatedMolmorph.nexus", sep = "/"),
            path = path,
            morpho = TRUE,
            morphoFilename = paste(path, grep(pattern = "morfologia",
                                              x = dir(path, pattern), value = TRUE),
                                   sep = "/"),
            sumFilename = "partitions.txt")

## End(Not run)
# Concatenate arbitrary matrices in the working directory,
# including a morphological matrix, return a concatenated file in the same dir
## Not run:
concatNexus(matrices = c("coi.nex", "rag1.nex", "cytb.nex", "morphology.nex"),
            filename = "concatenatedMolmorph.nexus",
```

```
            morpho = TRUE,
            morphoFilename = "morphology.nex",
            sumFilename = "partitions.txt")

## End(Not run)
```

`conflate` *conflate: Calculate the conflation of multiple distributions pdfs, plot = TRUE, from, to, n, add = FALSE*

Description

conflate: Calculate the conflation of multiple distributions pdfs, plot = TRUE, from, to, n, add = FALSE

Usage

```
conflate(pdfs, plot = TRUE, from, to, n, add = FALSE)
```

Arguments

`pdfs`	A vector of calls to density_fun for defininf each individual distribution.
`plot`	Whether to plot using curve
`from`, `to`, `n`	The appropriate values from and to which to calculate the conflation, and a number of points n. These are the same used by the function curve but are still necessary even if no plot is required.
`add`	Whether to add the curve to an existing plot.

Details

Produces either a plot or a data frame with the x and y values for the conflated PDF. It uses as input a vector of densities constructed with density_fun, and further parameters to be pased to curve if no plot is desired these are still used for returning a data frame with the x and y values from evaluation the conflated PDF on the sequence of x values determined by a number n of equidistant points between from and to.

Value

A tree of class phylo with summary branch lengths in tree$edge.length.

Examples

```
conflated_normals <- conflate(c("density_fun(x, 'dnorm', mean=0, sd=1)",
                                "density_fun(x, 'dnorm', mean=3, sd=1)"),
                              from=-4, to=4, n=101, plot=FALSE)
plot(conflated_normals)
```

crossplot *crossplot: Plot the median and HPD interval bars for pairs of distribution*

Description

crossplot: Plot the median and HPD interval bars for pairs of distribution

Usage

```
crossplot(
  log1Path,
  log2Path,
  skip.char = "#",
  pattern = NULL,
  idx.cols = NULL,
  bar.lty,
  bar.lwd,
  identity.lty,
  identity.lwd,
  extra.space = 0.5,
  ...
)
```

Arguments

log1Path	character vector of length 1. Path to the first log file.
log2Path	character vector of length 1. Path to the second log file.
skip.char	character vector of length 1, with '#' as default value. Which symbol is used as a comment. This will allow to ignore lines which start with the symbol when reading data.
pattern	character vector of length 1. the pattern for subsetting the columns containing the data to be plotted.
idx.cols	either an integer vector with the position of the columns to pick, or a character vector with the column names to pick. Defaults to NULL.
bar.lty	The line type to be used as error bars.
bar.lwd	As above but the width
identity.lty	The line type to be used in the identity y = x line
identity.lwd	As above but the width.
extra.space	numeric vector of length 1. How much space to be allowed in both xlim and ylim depending on the smallest value in highest density intervals plus or minus extra.space. A value of 0.5 units on the dimension of interest is used by default.
...	Optional arguments to be passed to 'plot' such as 'main', 'xlab', 'ylab', 'pch' and 'cex'.

Details

The function produces a crossplot, which is a scatterplot where we are comparing two distributions associated to each point by means of the medians as the points, and the highest density intervals as bars around the point. For instance, x may represent the prior of a set of parameters while y represents the posterior. Error bars on the x axis then are highest density intervals from the prior, and those on the y axis represent the interval for the posterior.

This function can also be used to compare two independent runs for (visual) convergence: If they are sampling the same posterior distribution, then they should fall on the identity y=x line.

Value

This function returns nothing, it plots to the graphical device.

Author(s)

Gustavo A. Ballen

Examples

```
## Not run:
crossplot(log1Path="log1.tsv", log2Path="log2.tsv", skip.char="#",
          pattern="par", cols=NULL, bar.lty=1, bar.lwd=1,
          identity.lty=2, identity.lwd=1,
       extra.space=0.5, main="My plot", xlab="log 1 (prior)", ylab="log 2 (posterior)", pch=19)

## End(Not run)
```

`c_truncauchy` *c_truncauchy: Estimate the c parameter for the truncated cauchy L distribution to be used in MCMCTree*

Description

c_truncauchy: Estimate the c parameter for the truncated cauchy L distribution to be used in MCMCTree

Usage

```
c_truncauchy(tl, tr, p = 0.1, pr = 0.975, al = 0.025, output = "par")
```

Arguments

`tl`	minimum age.
`tr`	maximum age
`p`	constant p involved in Cauchy parameters location and scale. Set to 0.1 by default. It determines how close the mode of the distribution is to the tl min age.
`pr`	percentile to the right of the distribution (0.975 by default)

al	alpha to the right of the minimum on x. Set it to zero if a hard minimum is desired, otherwise the random variable can take values below t_L with probability al. Set to 0.025 by default.
output	Whether to return just the parameters or all of the optimisation output. Defaults to "par". Leave it blank "" or with different text in order to return all of the optimisation output.

Details

We solve for c while fixing p=0.1 so that the mode of the distribution is closer to the t_L and then we calculate c so that t_R is at the desired max age. note that ar and al are NOT complements, thus both can be 0.025. Optimisation proceeds by fixing p in t_L(1-p) and then using numerical optimisation to find c in c*t_L.

Value

Either the parameter optimisation value as a numeric vector of length one (when output="par") or the complete optimisation output as a list (otherwise)

Author(s)

Gustavo A. Ballen

Examples

```
testValues.tr <- c(4.93, 12.12, 24.43, 49.20)
# the values below should be approx. c = 0.2, 0.5, 1, 2
# according to the paml documentation
for (i in testValues.tr) {
    print(c_truncauchy(tl=1, tr=i, p=0.1, pr=0.975, al=0.025))
}
```

---

density_fun *density_fun: A way to represent distributions to be conflated*

Description

density_fun: A way to represent distributions to be conflated

Usage

```
density_fun(x, dist, ...)
```

Arguments

x	a symbol which needs to be present in order to allow passing values towards the distruibution generator. It should be just x, without quotation marks (see examples).
dist	A character (using single quotes!) with the name of the distribution to use.
...	Parameters to be passed on to dist. See examples

Details

Produces a definition of each individual distribution to be conflated and provides the symbol for non-standard evaluation (x). Single quotes in dist are mandatory in order to avoid issues when calling expressions under the hood. See the documentation for each individual distribution to call their parameters adequately. Argument names and values should be used.

Value

A call from the elements of the distribution to be used.

Examples

```
c("density_fun(x, 'dnorm', mean=0, sd=1)",
  "density_fun(x, 'dnorm', mean=-1, sd=1)",
  "density_fun(x, 'dnorm', mean=1, sd=1)")
```

---

fasta2nexus *fasta2nexus (deprecated): Function for converting molecular alignments from fasta to nexus format*

Description

fasta2nexus (deprecated): Function for converting molecular alignments from fasta to nexus format

Usage

```
fasta2nexus(...)
```

Arguments

...	A placeholder for any argument that used to be in this function.

Details

This function was deprecated from tbea v1.0.0 onwards. For using this function please install tbea v0.5.0.

Value

This function returns an error because it has been deprecated.

Author(s)

Gustavo A. Ballen

`findParams` *Function for estimation of probability density function parameters through quadratic optimization*

Description

Function for estimation of probability density function parameters through quadratic optimization

Usage

```
findParams(q, p, output = "complete", pdfunction, params, initVals = NULL)
```

Arguments

`q`	A numeric vector of observed quantiles, might come from a HPD from a previous study (along with a median), or from other sources of prior information. See Details.
`p`	A numeric vector of percentiles.
`output`	One of two possible values: `"complete"` and `"parameters"`. For the latter the complete output of the `optim` function is returned with information on convergence and squared errors (that might be useless for simple cases) or just the parameters.
`pdfunction`	A character vector (of length one) with the name of the PDF function of interest. Technically this argument supports any PDF function of the form pDIST (e.g., `pnorm`, `ppois`, `pexp`).
`params`	A character vector with the name of the parameter(s) to optimize in the probability density function. These should match the parameter names of the respective PDF function, e.g., `"lambda"` in the function `ppois`
`initVals`	A numeric vector with default value `NULL`. It allows the user to provide initial values, althought this is discouraged in most cases.

Details

This function comes handy whenever we have some values of uncertainty, (e.g., confidence intervals, HPDs, biostratigraphic age constrains) and want to express it in the form of a probability density function of the form $P(x; \theta)$. As we have some values (the quantiles) already and their corresponding percentiles, all we need is a way to approximate the parameters θ that produce the same combination of quantiles for the given percentiles under a given PDF. This is carried out through optimization of a quadratic error function. This is accomplished through the function `optim`. For instance, if the estimated age of a fossil is Lutetian, in the Eocene (41.2 to 47.8 Ma), and we want to model such uncertainty through a normal distribution, we could assume that these age boundaries are the quantiles for percentiles 0.025 and 0.975 respectively, and add a thir pair with the midpoint corresponding to the percentile 0.5. This is all the information needed in order to estimate the parameters `mean` and `sd` in the functiono `pnorm`.

Value

Either a list with the complete output of convergence, squared errors and parameter values, or just a vector of parameter values. Depends on the value of `output`. Warnings may be triggered by the function `optim` since the optimization is a heuristic process, whenever a given iteration results in an invalid value for a given combination of parameters, the `optim` function tries another combination of values but inform the user about the problem through a warning. In general these can be safely disregarded.

Author(s)

Main code by Gustavo A. Ballen with important contributions in expression call structure and vectorized design by Klaus Schliep (<Klaus.Schliep@umb.edu>).

Examples

```
# Find the best parameters for a standard normal density that fit the observed quantiles
# -1.644854, 0, and 1.644854, providing full output for the calculations in the form of
# a list
findParams(q = c(-1.959964, 0.000000, 1.959964),
           p = c(0.025, 0.50, 0.975),
           output = "complete",
           pdfunction = "pnorm",
           params = c("mean", "sd"))

# Given that we have prior on the age of a fossil to be 1 - 10 Ma and that we want to
# model it with a lognormal distribution, fin the parameters of the PDF that best reflect
# the uncertainty in question (i.e., the parameters  for which the observed quantiles are
# 1, 5.5, and 10, assuming that we want the midpoint to reflect the mean of the PDF.
findParams(q = c(1, 5.5, 10),
           p = c(0.025,  0.50, 0.975),
           output = "complete",
           pdfunction = "plnorm",
           params = c("meanlog", "sdlog"))
```

---

laventa *Geochronology samples from the Honda Group in Colombia*

Description

A dataset containing geochronology data from several samples along the stratigraphic column of the Honda and Huila groups in the Tatacoa Desert area. The dataset was compiled from the Table 3.2 in Flynn et al. (1997).

Usage

```
data(laventa)
```

Format

A data frame with 87 rows and 7 variables:

age Estimated age (in Ma) from a given rock sample

one_sigma Standard deviation of the age estimate

sample Sample code as in Table 3.2

unit Stratigraphic unit in either the Honda Group or the Huila Group

elevation Position in the stratigraphic column, in meters

mineral The mineral used for dating the sample

comments Comments from footnotes in the original table

References

Flynn, J.J., Guerrero, J. & Swisher III, C.C. (1997) Geochronology of the Honda Group. In: R. F. Kay, R. H. Madden, R. L. Cifelli, and J. J. Flynn (Eds), Vertebrate Paleontology in the Neotropics: the Miocene Fauna of La Venta, Colombia. Smithsonian Institution Press, pp. 44–60.

`lognormalBeast` *Constructing a curve for the user-specified lognormal prior using Beast2 parameters*

Description

Constructing a curve for the user-specified lognormal prior using Beast2 parameters

Usage

```
lognormalBeast(
  M,
  S,
  meanInRealSpace = TRUE,
  offset = 0,
  from = NULL,
  to = NULL,
  by = 0.05
)
```

Arguments

`M`	Mean of the lognormal density in Beast2.
`S`	Standard deviation of the lognormal density in Beast2.
`meanInRealSpace`	Whether to plot the mean on the real- or log-space (i.e., apply log(M) before plotting). Please see under details.

offset	Hard lower bound.
from, to, by	Starting and ending point to calculate considering the offset as zero. That is, from will affect produce a starting point of (offset + from) and an ending point of (offset + to). By sets the step size of the sequence from 'from' to 'to' each 'by' steps.

Details

This function creates a matrix of x,y values given parameters of a lognormal density as specified in the program Beast2. It's main purpose is for plotting but other uses such as similarity quantification are available. Please note that the value of mean depends on whether we expect it to be in real or log space. Please refer to Heath (2015) for more info: Heath, T. A. (2015). Divergence Time Estimation using BEAST v2.

Value

A matrix of two columns consisting of the x and y values of the lognormal density.

Examples

```
# Generate a matrix for the lognormal density with mean 1 and standard deviation 1, with mean
# in real space, and spanning values in x from 0 to 10
lognormalBeast(M = 1, S = 1, meanInRealSpace = TRUE, from = 0, to = 10)
# The same as above but with an offset of 10, that is, the curve starts at 10 as if it was 0
# to values will start in (offset + from) and finish in (offset + to)
lognormalBeast(M = 1, S = 1, meanInRealSpace = TRUE, offset = 10, from = 0, to = 10)
```

---

measureSimil *Calculate the Intersection Between Two Densities*

Description

Calculate the Intersection Between Two Densities

Usage

```
measureSimil(
  d1,
  d2,
  splits = 500,
  rawData = c(TRUE, TRUE),
  plot = TRUE,
  x_limit = "auto",
  colors = c("red", "blue", "gray"),
  ...
)
```

Arguments

d1, d2	Either two vectors of empirical (i.e., MCMC-produced) values OR a data.frame/matrix with columns x and y for values fitted to a density from which to calculate areas. If rawData is set to TRUE in any instance, the data must be placed in vectors and not multidimensional objects.
splits	A numerical argument controling the number of subdivisions of the intersection area for numerical integration
rawData	Are d1 and/or d2 raw data for which a density should be calculated? A vector of length two containing logical values indicating whenther any of the arguments d1 or d2 are raw data or whether the user is inputing already calculated densities (e.g., the output from the density, curve, or dDIST functions, or any two-dimension object with x and y values)
plot	Should a plot be produced?
x_limit	Whether to define the xlim form the min-max of the combined density x-values
colors	A vector of three colors, namely, color of the d1 density (e.g., the prior), color of the d2 density e.g., the posterior), and color of the intersection.
...	Further arguments to pass to the graphical functions such as lines and plot internally (e.g., main, xlim, ylim, xlab, ylab, etc.).

Details

Similarity is measured as the overlapping portion between two densities. It has a value between 0 and 1. The values of the vector rawData determine the behavior of the function and therefore attention must be paid to their consistence with the nature of arguments d1 and d2. Despite the function was designed in order to allow to quantify similarity between the posterior and the prior, this can be used to quantify any overlap between two given densities and for any other purpose.

Value

A numeric vector with the value of the intersection between two densities. As a side effect, a plot is produced to an active (or new) graphical device.

Examples

```
# Set seed and colors to use in plots in the order: Prior, posterior, and intersection
set.seed(1985)
colors <- c("red", "blue", "lightgray")
# Similarity between two identical distributions
below <- measureSimil(d1 = rnorm(1000000, mean = 0, 1),
                      d2 = rnorm(1000000, mean = 0, 1),
                      main = "Comp. similarity",
                      colors = colors)
legend(x = "topright", legend = round(below, digits = 2))
# Similarity in two distributions partially overlapping
below <- measureSimil(d1 = rnorm(1000000, mean = 3, 1),
                      d2 = rnorm(1000000, mean = 0, 1),
                      main = "Partial similarity",
                      colors = colors)
```

```
legend(x = "topright", legend = round(below, digits = 2))
# Similarity in two completely-different distributions
below <- measureSimil(d1 = rnorm(1000000, mean = 8, 1),
                      d2 = rnorm(1000000, mean = 0, 1),
                      main = "Comp. dissimilarity",
                      colors = colors)
legend(x = "topright", legend = round(below, digits = 2))
# Don't plot, just return the intersection
measureSimil(d1 = rnorm(1000000, mean = 3, 1),
             d2 = rnorm(1000000, mean = 0, 1),
             plot = FALSE)
```

`mswd.test` *Reduced chi-square test or mean square weighted deviation (mswd) test*

Description

Reduced chi-square test or mean square weighted deviation (mswd) test

Usage

```
mswd.test(age, sd)
```

Arguments

age	A vector of age radiometric age estimates
sd	A vector of the standard deviation corresponding to each element in age

Details

From Ludwig (2003:646): "By convention, probabilities of fit greater than 0.05 are generally considered as arguably satisfying the mathematical assumptions of an isochron, while lower probabilities are generally taken as indicating the presence of "geological" scatter, and hence a significant possibility of bias in the isochron age.". The null hypothesis is that the isochron conditions hold.

Value

A numeric vector of length one with the p-value corresponding to the test.

Examples

```
data(laventa)
# Do the age estimates for the boundaries of the Honda Group (i.e., samples at meters 56.4
# and 675.0) conform to the isochron hypothesis?
hondaIndex <- which(laventa$elevation == 56.4 | laventa$elevation == 675.0)
mswd.test(age = laventa$age[hondaIndex], sd = laventa$one_sigma[hondaIndex])
# The p-value is smaller than the nominal alpha of 0.05, so we can reject the null
```

```
# hypothesis of isochron conditions

# Do the age estimates for the samples JG-R 88-2 and JG-R 89-2 conform to the isochron hypothesis?
twoLevelsIndex <- which(laventa$sample == "JG-R 89-2" | laventa$sample == "JG-R 88-2")
dataset <- laventa[twoLevelsIndex, ]
# Remove the values 21 and 23 because of their abnormally large standard deviations
mswd.test(age = dataset$age[c(-21, -23)], sd = dataset$one_sigma[c(-21, -23)])
# The p-value is larger than the nominal alpha of 0.05, so we can
# not reject the null hypothesis of isochron conditions
```

`stratCI` *stratCI: Estimate the confidence intervals of endpoints in stratigraphic intervals*

Description

stratCI: Estimate the confidence intervals of endpoints in stratigraphic intervals

Usage

```
stratCI(times, method, nparams, C, endpoint, confidence, quantile)
```

Arguments

`times`	a vector of occurrences in time for which we want to estimate the endpoints of the stratigraphic interval
`method`	a character describing which method to use, either 'Strauss-Sadler89' or 'Marshall94'.
`nparams`	A character indicating whether to estimate one or two parameters, possible values are 'one.par' and 'two.par'.
`C`	numeric indicating the confidence level, e.g. 0.95
`endpoint`	used only for nparams = 'one.par'. Possible values are 'first' and 'last'.
`confidence`	the confidence interval level, usually 0.95.
`quantile`	the desired confidence level for a quantile representing the brackets around the confidence interval.

Details

For method='Strauss-Sadler89' we need to provide 'nparams', 'C', and 'endpoint'. For method='Marshall94' we need to provide 'confidence' and 'quantile'.

Value

A named vector when using the 'Strauss-Sadler89' method, or an unnamed vector when using the 'Marshall94' method.

Author(s)

Gustavo A. Ballen.

Examples

```
data(andes)
andes <- andes$ages
# remove missing data
andes <- andes[complete.cases(andes)]
# remove outliers
andes <- sort(andes[which(andes < 10)])
stratCI(andes, method="Strauss-Sadler89",
        nparams="one.par", C=0.95, endpoint="first")
stratCI(andes, method="Strauss-Sadler89",
        nparams="one.par", C=0.95, endpoint="last")
stratCI(andes, method="Strauss-Sadler89",
        nparams="two.par", C=0.95)
stratCI(andes, method="Marshall94", confidence = 0.95,
        quantile = 0.8)
stratCI(andes, method="Marshall94", confidence = 0.95,
        quantile = 0.95)
```

---

summaryBrlen — *summaryBrlen: Summarise branch lengths on trees with identical topology*

Description

summaryBrlen: Summarise branch lengths on trees with identical topology

Usage

```
summaryBrlen(mphy, method)
```

Arguments

mphy	An list of objects of class multiPhylo. If a single object in this argument is of class multiPhylo, it is first enclosed in a list.
method	A character with the function name for the summary to be applied

Details

This function can be used on the output of topofreq from the $trees element in order to summarise the branch length on each topology set so that we have a single tree summarising both topology and branch lengths. Useful for depicting posterior tree density. Alternatively, it can be used with a single element provided that it is first enclosed in a list

Value

A tree of class phylo with summary branch lengths in tree$edge.length.

Examples

```
set.seed(1)
library(ape)
trl <- ape::rmtree(10, 4)
tpf <- topoFreq(unroot(trl), output="trees")
sumtrees <- summaryBrlen(tpf$trees, method = "median")
oldpar <- par(no.readonly = TRUE)
par(mfrow=c(1,3))
plot(sumtrees[[1]], type="unrooted", show.node.label=FALSE, cex=1.5)
plot(sumtrees[[2]], type="unrooted", show.node.label=FALSE, cex=1.5)
plot(sumtrees[[3]], type="unrooted", show.node.label=FALSE, cex=1.5)
par(oldpar)
```

---

table2nexus *table2nexus: Read a data matrix in delimited format and convert into a data matrix in nexus format*

Description

table2nexus: Read a data matrix in delimited format and convert into a data matrix in nexus format

Usage

```
table2nexus(
  path,
  datatype = c("standard", "dna", "rna", "protein"),
  header = FALSE,
  sep = ",",
  con = stdout()
)
```

Arguments

path	a character vector of length 1 with the path to the table file.
datatype	a character vector of length 1 with the desired datatype. Possible values are STANDARD, DNA, RNA, or PROTEIN. Multicharacter types such as continuous or nucleotide are not supported.
header	a logical vector of length 1 indicating whether the table file has a header. Defaults to FALSE.
sep	a character vector of length 1 telling the kind of separator in the table file. Defaults to comma ",".

con — the connection to which the matrix should be returned. Defaults to stdout(), that is, return the text to the console. If writing to a file, then this should be the path to the output file.

Details

This function will concatenate matrices in nexus format (mandatory) and write to the disk the output and summary information on the partitions. It requires that the input matrices all share the same taxa in the same positions.

Value

This function writes to the connected required a matrix in nexus format for a morphological dataset (that is, datatype=standard).

Author(s)

Gustavo A. Ballen

Examples

```
## Not run:
# this will return the matrix to the console rather than to a file
table2nexus(path="morpho.csv", datatype="standard", header=FALSE, sep=",")

## End(Not run)
```

---

tnt2newick — *tnt2newick: Function for converting from TNT tree format to newick parenthetical format*

Description

tnt2newick: Function for converting from TNT tree format to newick parenthetical format

Usage

```
tnt2newick(
  file,
  output = NULL,
  string = NULL,
  return = FALSE,
  subsetting = FALSE,
  name.sep = NULL
)
```

Arguments

file	A vector of type 'character' with the path to the original TNT tree file.
output	A vector of type 'character' with the path to output files to contain the tree in newick format.
string	A vector of type 'character' which can be either an object in memory or a string for interactive transformation, in TNT format. Use file in case your tree(s) are stored in a file instead.
return	A 'logical' expression indicating whether to write the newick tree(s) to a file in 'output' (if FALSE, the default), or whether to return to the screen (if TRUE), potentially to be stored in a vector via the '<-' operator.
subsetting	A vector of type 'logical' indicating whether subsetting (i.e., chopping at once the first and last line of the TNT tree file) should be done. Otherwise, explicit text replacements removing such lines are used. The default is false because it does not play well with multi-tree TNT files. Only use subsetting = TRUE if you are sure that there is only one tree in the file with the commands, tread and proc as first and last lines.
name.sep	A vector of length 2 and type 'character' for carrying out separator conversion in the names of terminals. For instance, if the terminals have names composed of two words separated by an underscore (_) and you want them to be separated by space ( ) then use name.sep = c("_", " "). This does not support regular expressions.

Details

This function has been tested for cases where only one tree is in the original tnt tree file. Please be careful with files containing multiple trees.

Value

This function writes to the disk a text file containing the tree converted to newick format. Alternatively, it returns the output to the screen or writes it to an object in memory thanks to the argument 'string'.

Author(s)

Gustavo A. Ballen

Examples

```
## Not run:
tnt2newick(file = "someTrees.tre", return = TRUE)

## End(Not run)
```

topoFreq *Frequency of topologies in a tree sample*

Description

Frequency of topologies in a tree sample

Usage

```
topoFreq(mphy, output = "index", maxtrees = 10000)
```

Arguments

mphy	An object of class multiPhylo
output	A character indicating whether the tree indices or the actual trees should be returned. Defaults to "index"
maxtrees	A numeric indicating whether to warn about having more trees than the arbitrary threshold

Details

This function can be used e.w. with a posterior sample of trees from a Bayesian analysis where we want to explore the distribution of topologies in the posterior of trees. This way we can assess topological uncertainty in a more meaningful way than using a majority-rule consensus.

The use of ‘maxtrees‘ is actually a convenience for keeping in mind that large amounts of trees can cause memory issues. This can end up in situations which are difficult to debug but that from personal experience have come from exactly that: More trees than memory can fit or which can be processed for calculating similarity. This number will _not_ break the function call but will return a warning. Try to avoid modifying its default value unless you are sure it will not cause any issues under your computing conditions (e.g. when lots of trees are being processed but also large RAM is available).

Value

A list with an element containing the the different tree clusters (as multiPhylo) and the absolute, cumulative, and relative frequencies of each topology in the tree sample.

Examples

```
# tests
set.seed(1)
library(ape)
trl <- ape::rmtree(10, 4)
tpf <- topoFreq(ape::unroot(trl), output="trees")
```

`xintercept` *xintercept: Estimate the x-intercept of an empirical cdf*

Description

xintercept: Estimate the x-intercept of an empirical cdf

Usage

```
xintercept(x, method, alpha = 0.05, p = c(0.025, 0.975), R = 1000, robust)
```

Arguments

`x`	A vector of type numeric with time data points.
`method`	Either "Draper-Smith" or "Bootstrap". The function will fail otherwise.
`alpha`	A vector of length one and type numeric with the nominal alpha value for the Draper-Smith method, defaults to 0.05.
`p`	A vector of length two and type numeric with the two-tail probability values for the CI. Defaults to 0.025 and 0.975.
`R`	The number of iterations to be used in the Bootstrap method.
`robust`	Logical value indicating whether to use robust regression using 'Rfit::rfit' ('robust = TRUE') or ordinary least squares 'lm' ('robust = FALSE').

Details

This function will take a vector of time points, calculate the empirical cumulative density function, and regress its values in order to infer the x-intercept and its confidence interval. For plotting purposes, it will also return the x-y empirical cumulative density values.

Value

A named list with three elements: 'param', the value of x_hat; 'ci', the lower and upper values of the confidence interval on x; 'ecdfxy', the x and y points for the empirical cumulative density curve.

Author(s)

Gustavo A. Ballen.

Examples

```
data(andes)
ages <- andes$ages
ages <- ages[complete.cases(ages)] # remove NAs
ages <- ages[which(ages < 10)] # remove outliers


# Draper-Smith, OLS
```

```
draperSmithNormalX0 <- xintercept(x = ages, method = "Draper-Smith", alpha = 0.05, robust = FALSE)
# Draper-Smith, Robust fit
draperSmithRobustX0 <- xintercept(x = ages, method = "Draper-Smith", alpha = 0.05, robust = TRUE)
# Bootstrap, OLS
bootstrapNormalX0 <- xintercept(x = ages, method = "Bootstrap", p = c(0.025, 0.975), robust = FALSE)
# Bootstrap, Robust fit
bootstrapRobustX0 <- xintercept(x = ages, method = "Bootstrap", p = c(0.025, 0.975), robust = TRUE)
# plot the estimations
hist(ages, probability = TRUE, col = rgb(red = 0, green = 0, blue = 1, alpha = 0.3),
     xlim = c(0, 10), main = "CDF-based on confidence intervals", xlab = "Age (Ma)")
# plot the lines for the estimator of Draper and Smith using lm
arrows(x0 = draperSmithNormalX0$ci["upper"], y0 = 0.025, x1 = draperSmithNormalX0$ci["lower"],
       y1 = 0.025, code = 3, angle = 90, length = 0.1, lwd = 3, col = "darkblue")
# plot the lines for the estimator of Draper and Smith using rfit
arrows(x0 = draperSmithRobustX0$ci["upper"], y0 = 0.05, x1 = draperSmithRobustX0$ci["lower"],
       y1 = 0.05, code = 3, angle = 90, length = 0.1, lwd = 3, col = "darkgreen")
# plot the lines for the estimator based on bootstrap
arrows(x0 = bootstrapRobustX0$ci["upper"], y0 = 0.075, x1 = bootstrapRobustX0$ci["lower"],
       y1 = 0.075, code = 3, angle = 90, length = 0.1, lwd = 3, col = "darkred")
# plot a legend
legend(x = "topright", legend = c("Draper and Smith with lm", "Draper and Smith with rfit",
                                  "Bootstrap on x0"),
       col = c("darkblue", "darkgreen", "darkred"),  lty = 1, lwd = 3)
```

Index