Package ‘tbea’

September 8, 2021

Title Tools for Pre- and Post-processing in Bayesian Evolutionary Analyses

Version 0.3.2

Description Package for bayesian inference in phylogenetics and evolution.
It provides functions for prior specification in divergence time
estimation using fossils as well as other kinds of data. The package
provides tools for interacticng with the input and output of bayesian
platforms in evolutionary biology such as BEAST2. The package implements
a way to measure interdependence between probability density functions
in the context of comparisons between prior and posterior bayesian
densities. It also provides functions for concatenating molecular and
morphological data for standard tree estimation (e.g., MrBayes) or total
evidence FBD divergence time estimation (e.g., Beast or RevBayes).

Depends R (>= 3.1.0)

Imports ape, apex

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

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

R topics documented:

`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. The former is guaranteed by the function `fasta2nexus`, otherwise need to be carried out by the user, e.g., manually. The latter is kind of manual too as the morphological matrix is also expected to be manually generated by the user. This can be achieved by manually modifying the `wholeAlign.nex` file generated by `fasta2nexus`.

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"),
            filename = "concatenatedMolmorph.nexus",
            morpho = TRUE,
            morphoFilename = "morphology.nex",
            sumFilename = "partitions.txt")

## End(Not run)
```

`fasta2nexus` *fasta2nexus: Function for converting molecular alignments from fasta to nexus format*

Description

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

Usage

```
fasta2nexus(path, outpath = NULL, pattern, wholeAlign = TRUE)
```

Arguments

`path`	A vector of type 'character' with the path to the fasta alignments.
`outpath`	A vector of type 'character' with the path to the nexus matrices. Defaults to NULL, so that the output files are written into the same directory declared in `path`
`pattern`	A vector of type 'character' with the string (also supports regular expressions) to be used as keyword for selecting the fasta files. The most basic case is to use ".fasta$" for a file ending with the extension ".fasta".
`wholeAlign`	Whether to fuse the fasta alignments into a concatenated molecular-only, continuous nexus matrix. Defaults to TRUE.

Details

This function will convert from fasta to nexus, and optionally concatenate a single nexus with the content of all fasta files.

Value

This function writes to the disk several files, at least one nexus originally from a fasta file, and potentially a concatenated file if several fasta are provided.

Author(s)

Gustavo A. Ballen

Examples

```
# Convert all fasta alignments into nexus matrices in a given path,
# with the output files in the same directory, for files
# ending with the pattern 'trimmed.nex'.
## Not run:
fasta2nexus(path = "sequences", outpath = NULL, pattern = "trimmed.fasta$", wholeAlign = TRU

## End(Not run)
```

`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, to, 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 mea
# 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

```
## Not run:
# 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)

## End(Not run)
```

`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 hypot
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
```

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, subsetting = TRUE)
```

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.
`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.

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.

Author(s)

Gustavo A. Ballen

Examples

```
# Convert a tree in TNT tree format to newick format
## Not run:
tnt2newick(file = "my_TNT_tree.tre", output = "my_TNT_tree.newick")

## End(Not run)
```

Index