Package: spider (via r-universe)

September 12, 2024

Type Package

Title Species Identity and Evolution in R

Version 1.5.0.9000

Date 2018-02-15

Author Samuel Brown, Rupert Collins, Stephane Boyer, Marie-Caroline Lefort, Jagoba Malumbres-Olarte, Cor Vink, Rob Cruickshank

Maintainer Rupert A. Collins <rupertcollins@gmail.com>

Description Analysis of species limits and DNA barcoding data. Included are functions for generating important summary statistics from DNA barcode data, assessing specimen identification efficacy, testing and optimizing divergence threshold limits, assessment of diagnostic nucleotides, and calculation of the probability of reciprocal monophyly. Additionally, a sliding window function offers opportunities to analyse information across a gene, often used for marker design in degraded DNA studies. Further information on the package has been published in Brown et al (2012) <doi:10.1111/j.1755-0998.2011.03108.x>.

License MIT + file LICENSE

LazyLoad yes

Imports ape, pegas, graphics, stats, utils

RoxygenNote 6.1.1

Suggests testthat

Repository https://boopsboops.r-universe.dev

RemoteUrl https://github.com/boopsboops/spider

RemoteRef HEAD

RemoteSha e93c5b4bc7f50168b8a155a6dca7c87dfbdef134

Contents

`spider-package` *Species Identity and Evolution in R*

Description

Spider: SPecies IDentity and Evolution in R, is an R package implementing a number of useful analyses for DNA barcoding studies and associated research into species delimitation and speciation. Included are functions for generating summary statistics from DNA barcode data, assessing specimen identification efficacy, and for testing and optimising divergence threshold limits. In terms of investigating evolutionary and taxonomic questions, techniques for sliding window, population aggregate, and nucleotide diagnostic analyses are also provided.

Details

The complete list of functions can be displayed with `library(help=spider)`.

More information, including a tutorial on the use of spider can be found at `http://spider.r-forge.r-project.org`.

Package:	spider
Type:	Package
Version:	1.4-2
Date:	2017-05-13
License:	GPL
LazyLoad:	yes

A few of the key functions provided by spider:

DNA barcoding: `bestCloseMatch`, `nearNeighbour`, `threshID`, `threshOpt`, `heatmapSpp`.

Sliding window: `slidingWindow`, `slideAnalyses`, `slideBoxplots`.

Nucleotide diagnostics: `nucDiag`, `rnucDiag`.

Morphological techniques: `paa`.

Author(s)

Samuel Brown, Rupert Collins, Stephane Boyer, Marie-Caroline Lefort, Jagoba Malumbres-Olarte, Cor Vink, Rob Cruickshank

Maintainer: Samuel Brown <s_d_j_brown@hotmail.com>

References

Brown S. D. J., Collins R. A., Boyer S., Lefort M.-C., Malumbres-Olarte J., Vink C. J., & Cruickshank R. H. 2012. SPIDER: an R package for the analysis of species identity and evolution, with particular reference to DNA barcoding. _Molecular Ecology Resources_ 12:562-565. doi: 10.1111/j.1755-0998.2011.03108.x

See Also

`ape-package`, `pegas-package`.

`anoteropsis` *Cytochrome oxidase I (COI) sequences of New Zealand _Anoteropsis_ species*

Description

A set of 33 sequences of the mitochondrial protein-coding gene cytochrome oxidase I from 20 species of the New Zealand wolf spider genus *Anoteropsis* (Lycosidae) and two species of *Artoria* as outgroups. The sequences are available on GenBank as accession numbers AY059961 through AY059993.

Format

A DNAbin object containing 33 sequences with a length of 409 base pairs stored as a matrix.

Source

Vink, C. J., and Paterson, A. M. (2003). Combined molecular and morphological phylogenetic analyses of the New Zealand wolf spider genus _Anoteropsis_ (Araneae: Lycosidae). _Molecular Phylogenetics and Evolution_ *28* 576-587.

`bestCloseMatch` *Measures of identification accuracy*

Description

Tests of barcoding efficacy using distance-based methods.

Usage

```
bestCloseMatch(distobj, sppVector, threshold = 0.01, names = FALSE)
```

Arguments

`distobj`	A distance object (usually from `dist.dna`).
`sppVector`	Vector of species names. See `sppVector`.
`threshold`	Distance cutoff for identifications. Default of 0.01 (1%).
`names`	Logical. Should the names of the nearest match be shown? Default of FALSE.

Details

These functions test barcoding efficacy. All sequences must be identified prior to testing. Each sequence is considered an unknown while the remaining sequences in the dataset constitute the DNA barcoding database that is used for identification. If the identification from the test is the same as the pre-considered identification, a correct result is returned.

`bestCloseMatch` conducts the "best close match" analysis of Meier et al. (2006), considering the closest individual unless it is further than the given threshold, which results in no identification. More than one species tied for closest match results in an assignment of "ambiguous". When the threshold is large, this analysis will return essentially the same result as `nearNeighbour`. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

`nearNeighbour` finds the closest individual and returns if their names are the same (TRUE) or different (FALSE). If `names = TRUE`, the name of the closest individual is returned. Ties are decided by majority rule.

`threshID` conducts a threshold-based analysis, similar to that conducted by the "Identify Specimen" tool provided by the Barcode of Life Database (`http://www.boldsystems.org/index.php/IDS_OpenIdEngine`). It is more inclusive than `bestCloseMatch`, considering ALL sequences within the given threshold. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

These functions are not recommended as identification tools, though they can be used as such when `names = TRUE`.

Value

`bestCloseMatch` and `threshID` return a character vector giving the identification status of each individual.

`"correct"`	The name of the closest match is the same
`"incorrect"`	The name of the closest match is different
`"ambiguous"`	More than one species is the closest match (`bestCloseMatch`), or is within the given threshold (`threshID`)
`"no id"`	No species are within the threshold distance

`nearNeighbour` returns a logical vector or (if `names = TRUE`) the name for the nearest individual.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Meier, R., Shiyang, K., Vaidya, G., & Ng, P. (2006). DNA barcoding and taxonomy in Diptera: a tale of high intraspecific variability and low identification success. _Systematic Biology_ *55* (5) 715-728.

See Also

`nearNeighbour`, `threshID`, `dist.dna`, `sppVector` Also as `help`, ~~~

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split = "_"),
    function(x) paste(x[1], x[2], sep = "_"))

bestCloseMatch(anoDist, anoSpp)
bestCloseMatch(anoDist, anoSpp, threshold = 0.005)
nearNeighbour(anoDist, anoSpp)
nearNeighbour(anoDist, anoSpp, names = TRUE)
threshID(anoDist, anoSpp)
threshID(anoDist, anoSpp, threshold = 0.003)

data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

bestCloseMatch(doloDist, doloSpp)
bestCloseMatch(doloDist, doloSpp, threshold = 0.005)
nearNeighbour(doloDist, doloSpp)
nearNeighbour(doloDist, doloSpp, names=TRUE)
threshID(doloDist, doloSpp)
threshID(doloDist, doloSpp, threshold = 0.003)
```

`blockAlignment` *Make all sequences the same length*

Description

Coerces all sequences in a DNAbin object to the same length.

Usage

```
blockAlignment(DNAbin, mode = "shortest", range = NULL, fill = "")
```

Arguments

`DNAbin`	An object of class `DNAbin`
`mode`	Character vector. Options of "shortest" or "longest"
`range`	Numeric vector of length 2. Index of the bases where the new alignment should begin and end
`fill`	Character to fill the extra bases in short sequences. Default of "" (blank). Recommend that only "-" (gap) or "?" be used

Details

When `mode = "shortest"`, the alignment is truncated at the length of the shortest sequence. When `mode = "longest"`, the alignment is extended to the end of the longest sequence, with shorter sequences filled in with `"fill"`s.

Value

A DNAbin object in matrix format.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(salticidae)
salticidae
blockAlignment(salticidae)
blockAlignment(salticidae, mode = "longest")
blockAlignment(salticidae, mode = NULL, range = c(200, 600))

graphics::image(blockAlignment(salticidae))
graphics::image(blockAlignment(salticidae, mode = "longest"))
graphics::image(blockAlignment(salticidae, mode = NULL, range = c(200, 600)))
```

`cgraph` *Complete graph*

Description

Creates a complete graph for the given cloud of vertices.

Usage

```
cgraph(x, y = NULL, ...)
```

Arguments

`x`	X values, or a matrix with two columns containing X and Y values.
`y`	Y values. Can be left empty if `x` is a matrix.
`...`	Other arguments to be passed to `segments`.

Details

If `y` is not given, `x` is required to be a matrix containing both x and y values.

Value

Plots a complete graph between the given vertices.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`plot.ordinDNA`.

Examples

```
x <- runif(15)
y <- runif(15)

graphics::plot(x, y)
cgraph(x, y)

M <- cbind(x, y)
cgraph(M[1:10,], col = "blue")
```

`chaoHaplo` *Chao estimator of haplotype number*

Description

Calculates the Chao1 estimate of the number of haplotypes in a population based on the total number of haplotypes present, and the number of singletons and doubletons in the dataset.

Usage

```
chaoHaplo(DNAbin)
```

Arguments

`DNAbin`	An object of class ‘DNAbin’.

Details

The function assumes a large number of specimens have been sampled and that duplicate haplotypes have not been removed. Interpretation becomes difficult when more than one species is included in the dataset.

Value

An vector of length three, giving the estimated total number of haplotypes in the population, and lower and upper 95% confidence limits.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Vink, C. J., McNeill, M. R., Winder, L. M., Kean, J. M., and Phillips, C. B. (2011). PCR analyses of gut contents of pasture arthropods. In: Paddock to PCR: Demystifying Molecular Technologies for Practical Plant Protection (eds. Ridgway, H. J., Glare, T. R., Wakelin, S. A., O'Callaghan, M.), pp. 125-134. New Zealand Plant Protection Society, Lincoln.

Chao, A. (1989). Estimating population size for sparse data in capture-recapture experimnets. _Biometrics_ *45* 427-438.

See Also

`haploAccum`

Examples

```
data(dolomedes)
#Create dataset with multiple copies of Dolomedes haplotypes
doloSamp <- dolomedes[sample(16, 100, replace=TRUE, prob=c(0.85, rep(0.01, 15))), ]

chaoHaplo(doloSamp)
```

`checkDNA` *Check a DNA alignment for missing data*

Description

This functions counts the number of bases in an alignment that are composed of missing data.

Usage

```
checkDNA(DNAbin, gapsAsMissing = TRUE)
```

Arguments

`DNAbin`	A DNA alignment of class ‘DNAbin’.
`gapsAsMissing`	Logical. Should gaps (coded as ’-’) be considered missing bases? Default of TRUE.

Details

This function considers bases coded as ’?’ and ’N’ as missing data. By default, gaps (coded as ’-’) are also considered missing.

Value

A numeric vector giving the number of missing bases in each sequence of the alignment.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
checkDNA(anoteropsis)
checkDNA(anoteropsis, gapsAsMissing=FALSE)
```

`dataStat` *Taxa statistics*

Description

Returns the numbers of species, genera and individuals in the dataset.

Usage

```
dataStat(sppVector, genVector, thresh = 5)
```

Arguments

`sppVector`	Species vector (see `sppVector`).
`genVector`	Genus vector that defines the genera of each individual, created in a similar manner to the species vector.
`thresh`	Threshold for adequate individual/species number. Default of 5.

Details

The value `NULL` can be passed to gen if genera are not of interest in the dataset.

Value

A table giving the number of genera and species in the dataset; giving the minimum, maximum, mean and median number of individuals per species, and the number of species below the given threshold.

Author(s)

Rupert Collins <rupertcollins@gmail.com>

Examples

```
data(anoteropsis)
#Species vector
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))
#Genus vector
anoGen <-  sapply(strsplit(anoSpp, split="_"), function(x) x[1])
dataStat(anoSpp, anoGen)
```

---

dolomedes *Cytochrome oxidase I (COI) sequences of New Zealand _Dolomedes_ species*

Description

A set of 37 sequences of the mitochondrial protein-coding gene cytochrome oxidase I from the 4 New Zealand species of the nursery-web spider genus *Dolomedes* (Pisauridae). These sequences are available on GenBank as accession numbers GQ337328 through GQ337385.

Format

A DNAbin object containing 37 sequences with a length of 850 base pairs stored as a matrix.

Source

Vink, C. J., and Duperre, N. (2010). Pisauridae (Arachnida: Araneae). _Fauna of New Zealand_ *64* 1-54.

haploAccum — *Haplotype accumulation curves*

Description

haploAccum identifies the different haplotypes represented in a set of DNA sequences and performs the calculations for plotting haplotype accumulations curves (see plot.haploAccum).

Usage

```
haploAccum(DNAbin, method = "random", permutations = 100, ...)
```

Arguments

DNAbin	A set of DNA sequences in an object of class 'DNAbin'.
method	Method for haplotype accumulation. Method "collector" enters the sequences in the order that they appear in the sequence alignment and "random" adds the sequences in a random order.
permutations	Number of permutations for method "random".
...	Other parameters to functions.

Details

Haplotype accumulation curves can be used to assess haplotype diversity in an area or compare different populations, or to evaluate sampling effort. ``random'' calculates the mean accumulated number of haplotypes and its standard deviation through random permutations (subsampling of sequences), similar to the method to produce rarefaction curves (Gotelli and Colwell 2001).

Value

An object of class 'haploAccum' with items:

call	Function call.
method	Method for accumulation.
sequences	Number of analysed sequences.
n.haplotypes	Accumulated number of haplotypes corresponding to each number of sequences.
sd	The standard deviation of the haplotype accumulation curve. Estimated through permutations for method = "random" and NULL for method = "collector".
perm	Results of the permutations for method = "random".

Note

This function is based on the functions haplotype (E. Paradis) from the package 'pegas' and specaccum (R. Kindt) from the package'vegan'. Missing or ambiguous data will be detected and indicated by a warning, as they may cause an overestimation of the number of haplotypes.

Author(s)

Jagoba Malumbres-Olarte <j.malumbres.olarte@gmail.com>.

References

Gotellli, N.J. & Colwell, R.K. (2001). Quantifying biodiversity: procedures and pitfalls in measurement and comparison of species richness. _Ecology Letters_ *4*, 379–391.

Examples

```
data(dolomedes)
#Generate multiple haplotypes
doloHaplo <- dolomedes[sample(37, size = 200, replace = TRUE), ]
dolocurv <- haploAccum(doloHaplo, method = "random", permutations = 100)
dolocurv
graphics::plot(dolocurv)
```

---

heatmapSpp *Visualise a distance matrix using a heatmap*

Description

This function plots a heatmap of the distance matrix, with shorter distances indicated by darker colours.

Usage

```
heatmapSpp(distObj, sppVector, col = NULL, axisLabels = NULL,
  triangle = "both", showData = FALSE, dataRound = 3, dataCEX = 1)
```

Arguments

`distObj`	A matrix or object of class `dist`.
`sppVector`	The species vector. See `sppVector`.
`col`	A vector giving the colours for the heatmap.
`axisLabels`	A character vector that provides the axis labels for the heatmap. By default the species vector is used.
`triangle`	Which triangle of the heatmap should be plotted. Possible values of "both", "upper" and "lower". Default of "both".
`showData`	Logical. Should the data be shown on the heatmap? Default of FALSE.
`dataRound`	The number of significant figures the printed data will show. Default of 3.
`dataCEX`	Size of text for printed data. Default of 1.

Details

The default palette has been taken from the `colorspace` package.

Value

Plots a heatmap of the distance matrix. Darker colours indicate shorter distances, lighter colours indicate greater distances.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes, model = "raw")
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)
heatmapSpp(doloDist, doloSpp)
heatmapSpp(doloDist, doloSpp, axisLabels = dimnames(dolomedes)[[1]])

data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis, model = "raw")
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))
heatmapSpp(anoDist, anoSpp)
heatmapSpp(anoDist, anoSpp, showData = TRUE)
heatmapSpp(anoDist, anoSpp, showData = TRUE, dataRound = 1, dataCEX = 0.4)
heatmapSpp(anoDist, anoSpp, triangle = "upper")
heatmapSpp(anoDist, anoSpp, triangle = "lower")
heatmapSpp(anoDist, anoSpp, triangle = "lower", showData = TRUE, dataRound = 1, dataCEX = 0.4)
```

`is.ambig` *Missing bases in alignments*

Description

Checks what columns in an alignment have ambiguous bases or missing data.

Usage

```
is.ambig(DNAbin)
```

Arguments

`DNAbin`	A DNA alignment of class 'DNAbin'.

Details

Ambiguous bases are bases that have been coded with any of the Union of Pure and Applied Chemistry (IUPAC) DNA codes that are not A, C, G, or T. Missing data are bases that have been coded with "-", "?" or "N".

Value

A logical vector containing TRUE if ambiguous bases or missing data are present, FALSE if not. Does not differentiate between the two classes of data.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`checkDNA`

Examples

```
data(woodmouse)
is.ambig(woodmouse)
#Columns with ambiguous bases
which(is.ambig(woodmouse))
```

`localMinima` *Determine thresholds from a density plot*

Description

This function determines possible thresholds from the distance matrix for an alignment.

Usage

```
localMinima(distobj)
```

Arguments

`distobj`	A distance object (usually from `dist.dna`).

Details

This function is based on the concept of the barcoding gap, where a dip in the density of genetic distances indicates the transition between intra- and inter-specific distances. Understanding your data is vital to correctly interpreting the output of this function, but as a start, the first local minimum is often a good place to start.

The value of this function is that it does not require prior knowledge of species identity to get an indication of potential threshold values.

Value

An object of class 'density', which is a list containing the values calculated by density. The element localMinima has been added, which contains the values of the local minima of the density plot.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

dist.dna, density. Also as help, ~~~

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)

anoThresh <- localMinima(anoDist)
graphics::plot(anoThresh)
anoThresh$localMinima
#Often the first value is the one to go for:
anoThresh$localMinima[1]
```

---

maxInDist *Nearest non-conspecific and maximum intra-specific distances*

Description

These functions give the distances to the nearest non-conspecific and furthest conspecific representatives for each individual in the dataset.

Usage

```
maxInDist(distobj, sppVector = NULL, propZero = FALSE, rmNA = FALSE)
```

Arguments

distobj	Distance matrix.
sppVector	Species vector (see sppVector). Default of NULL.
propZero	Logical. TRUE gives the proportion of zero distances.
rmNA	Logical. TRUE ignores missing values in the distance matrix. Default of FALSE

Details

`nonConDist` returns the minimum inter-specific distance for each individual.

`maxInDist` returns the maximum intra-specific distance for each individual.

These two functions can be used to create a version of the barcoding gap.

`minInDist` returns the minimum intra-specific distance for each individual.

Value

If `propZero=FALSE`, a numeric vector giving the distance of the closest non-conspecific individual (`nonConDist`) or the most distant conspecific individual (`maxInDist`).

If `propZero=TRUE`, a single number giving the proportion of zero distances.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

nonConDist(anoDist, anoSpp)
nonConDist(anoDist, anoSpp, propZero=TRUE)

maxInDist(anoDist, anoSpp)
maxInDist(anoDist, anoSpp, propZero=TRUE)

#Barcoding gap
inter <- nonConDist(anoDist, anoSpp)
intra <- maxInDist(anoDist, anoSpp)
graphics::hist(inter-intra)

#An alternative way of plotting the gap
bnd <- cbind(data.frame(inter, intra))
ord <- bnd[order(bnd$inter),]
graphics::plot(ord$inter, type="n", ylab="Percent K2P distance", xlab="Individual")
segCol <- rep("gray50", length(ord$inter))
segCol[ord$inter-ord$intra < 0] <- "red"
graphics::segments(x0=1:length(ord$inter), y0=ord$inter, y1=ord$intra, col=segCol, lwd=6)
```

minInDist — *Nearest non-conspecific and maximum intra-specific distances*

Description

These functions give the distances to the nearest non-conspecific and furthest conspecific representatives for each individual in the dataset.

Usage

```
minInDist(distobj, sppVector = NULL, propZero = FALSE, rmNA = FALSE)
```

Arguments

`distobj`	Distance matrix.
`sppVector`	Species vector (see `sppVector`). Default of NULL.
`propZero`	Logical. TRUE gives the proportion of zero distances.
`rmNA`	Logical. TRUE ignores missing values in the distance matrix. Default of FALSE

Details

`nonConDist` returns the minimum inter-specific distance for each individual.

`maxInDist` returns the maximum intra-specific distance for each individual.

These two functions can be used to create a version of the barcoding gap.

`minInDist` returns the minimum intra-specific distance for each individual.

Value

If `propZero=FALSE`, a numeric vector giving the distance of the closest non-conspecific individual (`nonConDist`) or the most distant conspecific individual (`maxInDist`).

If `propZero=TRUE`, a single number giving the proportion of zero distances.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

nonConDist(anoDist, anoSpp)
nonConDist(anoDist, anoSpp, propZero=TRUE)
```

```
maxInDist(anoDist, anoSpp)
maxInDist(anoDist, anoSpp, propZero=TRUE)

#Barcoding gap
inter <- nonConDist(anoDist, anoSpp)
intra <- maxInDist(anoDist, anoSpp)
graphics::hist(inter-intra)

#An alternative way of plotting the gap
bnd <- cbind(data.frame(inter, intra))
ord <- bnd[order(bnd$inter),]
graphics::plot(ord$inter, type="n", ylab="Percent K2P distance", xlab="Individual")
segCol <- rep("gray50", length(ord$inter))
segCol[ord$inter-ord$intra < 0] <- "red"
graphics::segments(x0=1:length(ord$inter), y0=ord$inter, y1=ord$intra, col=segCol, lwd=6)
```

`monophyly` *Species monophyly over a tree*

Description

Determines if the species given in `sppVector` form monophyletic groups on a given tree.

Usage

```
monophyly(phy, sppVector, pp = NA, singletonsMono = TRUE)
```

Arguments

`phy`	A tree of class ‘phylo’.
`sppVector`	Species vector. See `sppVector`
`pp`	Object of class ‘prop.part’. Assists in speeding up the function, if it has been called already. Default of NA, calling `prop.part` internally.
`singletonsMono`	Logical. Should singletons (i.e. only a single specimen representing that species) be treated as monophyletic? Default of TRUE. Possible values of FALSE and NA.

Details

`monophyly` determines if each species is monophyletic. `monophylyBoot` incorporates a bootstrap test to determine the support for this monophyly. Species with a bootstrap support lower than `"thresh"` are recorded as FALSE.

Rerooting is done on the longest internal edge in the tree returned by `nj(dist.dna(DNAbin))`.

Value

`monophyly` returns a logical vector, stating if each species is monophyletic. Values correspond to the species order given by `unique(sppVector)`.

`monophylyBoot` returns a list with the following elements:

`results`	A logical vector, stating if each species is monophyletic with a bootstrap support higher than the given threshold.
`BSvalues`	A numeric vector giving the bootstrap proportions for each node of `phy`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`prop.part`, `root`, `boot.phylo`.

Examples

```
#Random trees
set.seed(16)
tr <- ape::rtree(15)
spp <- rep(LETTERS[1:5], rep(3,5))
monophyly(tr, spp)

tr2 <- tr
spp2 <- c(rep(LETTERS[1:4], rep(3,4)), LETTERS[5:7])
monophyly(tr2, spp2)

#Empirical data
## Not run:
data(anoteropsis)
anoTree <- ape::nj(ape::dist.dna(anoteropsis))
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

monophyly(anoTree, anoSpp)
monophyly(anoTree, anoSpp, singletonsMono=FALSE)
unique(anoSpp)

#To get score for each individual
anoMono <- monophyly(anoTree, anoSpp)
anoMono[match(anoSpp, unique(anoSpp))]

data(woodmouse)
woodTree <- ape::nj(ape::dist.dna(woodmouse))
woodSpp <- c("D", "C", "C", "A", "A", "E", "A", "F", "C", "F", "E", "D", "A", "A", "E")
unique(woodSpp)
monophyly(woodTree, woodSpp)
woodMono <- monophylyBoot(woodTree, woodSpp, woodmouse)
woodMono$results
```

```
woodMono$BSvalues

monophylyBoot(woodTree, woodSpp, woodmouse, reroot = FALSE)
monophylyBoot(woodTree, woodSpp, woodmouse, thresh = 0.9, reroot = FALSE)

## End(Not run)
```

---

monophylyBoot *Species monophyly over a tree*

Description

Determines if the species given in `sppVector` form monophyletic groups on a given tree.

Usage

```
monophylyBoot(phy, sppVector, DNAbin, thresh = 0.7, reroot = TRUE,
  pp = NA, singletonsMono = TRUE, reps = 1000, block = 3)
```

Arguments

`phy`	A tree of class ‘phylo’.
`sppVector`	Species vector. See `sppVector`
`DNAbin`	An object of class ’DNAbin’. Required for calculating bootstrap values.
`thresh`	Numeric between 0 and 1. Bootstrap threshold under which potentially monophyletic species are negated. Default of 0.7.
`reroot`	Logical. Should the bootstrap replicates be rerooted on the longest edge? Default of TRUE.
`pp`	Object of class ‘prop.part’. Assists in speeding up the function, if it has been called already. Default of NA, calling `prop.part` internally.
`singletonsMono`	Logical. Should singletons (i.e. only a single specimen representing that species) be treated as monophyletic? Default of TRUE. Possible values of FALSE and NA.
`reps`	Numeric. Number of bootstrap replications. Default of 1000.
`block`	The number of nucleotides that will be resampled together. Default of 3 to resample on the codon level.

Details

`monophyly` determines if each species is monophyletic. `monophylyBoot` incorporates a bootstrap test to determine the support for this monophyly. Species with a bootstrap support lower than `"thresh"` are recorded as FALSE.

Rerooting is done on the longest internal edge in the tree returned by `nj(dist.dna(DNAbin))`.

Value

monophyly returns a logical vector, stating if each species is monophyletic. Values correspond to the species order given by unique(sppVector).

monophylyBoot returns a list with the following elements:

results	A logical vector, stating if each species is monophyletic with a bootstrap support higher than the given threshold.
BSvalues	A numeric vector giving the bootstrap proportions for each node of phy.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

prop.part, root, boot.phylo, monophyly.

Examples

```
#Random trees
set.seed(16)
tr <- ape::rtree(15)
spp <- rep(LETTERS[1:5], rep(3,5))
monophyly(tr, spp)

tr2 <- tr
spp2 <- c(rep(LETTERS[1:4], rep(3,4)), LETTERS[5:7])
monophyly(tr2, spp2)

#Empirical data
## Not run:
data(anoteropsis)
anoTree <- ape::nj(ape::dist.dna(anoteropsis))
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

monophyly(anoTree, anoSpp)
monophyly(anoTree, anoSpp, singletonsMono=FALSE)
unique(anoSpp)

#To get score for each individual
anoMono <- monophyly(anoTree, anoSpp)
anoMono[match(anoSpp, unique(anoSpp))]

data(woodmouse)
woodTree <- ape::nj(ape::dist.dna(woodmouse))
woodSpp <- c("D", "C", "C", "A", "A", "E", "A", "F", "C", "F", "E", "D", "A", "A", "E")
unique(woodSpp)
monophyly(woodTree, woodSpp)
woodMono <- monophylyBoot(woodTree, woodSpp, woodmouse)
woodMono$results
```

```
woodMono$BSvalues

monophylyBoot(woodTree, woodSpp, woodmouse, reroot = FALSE)
monophylyBoot(woodTree, woodSpp, woodmouse, thresh = 0.9, reroot = FALSE)

## End(Not run)
```

`nearNeighbour` *Measures of identification accuracy*

Description

Tests of barcoding efficacy using distance-based methods.

Usage

```
nearNeighbour(distobj, sppVector, names = FALSE)
```

Arguments

`distobj`	A distance object (usually from `dist.dna`).
`sppVector`	Vector of species names. See `sppVector`.
`names`	Logical. Should the names of the nearest match be shown? Default of FALSE.

Details

These functions test barcoding efficacy. All sequences must be identified prior to testing. Each sequence is considered an unknown while the remaining sequences in the dataset constitute the DNA barcoding database that is used for identification. If the identification from the test is the same as the pre-considered identification, a correct result is returned.

`bestCloseMatch` conducts the "best close match" analysis of Meier et al. (2006), considering the closest individual unless it is further than the given threshold, which results in no identification. More than one species tied for closest match results in an assignment of "ambiguous". When the threshold is large, this analysis will return essentially the same result as `nearNeighbour`. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

`nearNeighbour` finds the closest individual and returns if their names are the same (TRUE) or different (FALSE). If `names = TRUE`, the name of the closest individual is returned. Ties are decided by majority rule.

`threshID` conducts a threshold-based analysis, similar to that conducted by the "Identify Specimen" tool provided by the Barcode of Life Database (`http://www.boldsystems.org/index.php/IDS_OpenIdEngine`). It is more inclusive than `bestCloseMatch`, considering ALL sequences within the given threshold. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

These functions are not recommended as identification tools, though they can be used as such when `names = TRUE`.

Value

bestCloseMatch and threshID return a character vector giving the identification status of each individual.

"correct"	The name of the closest match is the same
"incorrect"	The name of the closest match is different
"ambiguous"	More than one species is the closest match (bestCloseMatch), or is within the given threshold (threshID)
"no id"	No species are within the threshold distance

nearNeighbour returns a logical vector or (if names = TRUE) the name for the nearest individual.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Meier, R., Shiyang, K., Vaidya, G., & Ng, P. (2006). DNA barcoding and taxonomy in Diptera: a tale of high intraspecific variability and low identification success. _Systematic Biology_ *55* (5) 715-728.

See Also

nearNeighbour, threshID, dist.dna, sppVector Also as help, ~~~

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split = "_"),
    function(x) paste(x[1], x[2], sep = "_"))

bestCloseMatch(anoDist, anoSpp)
bestCloseMatch(anoDist, anoSpp, threshold = 0.005)
nearNeighbour(anoDist, anoSpp)
nearNeighbour(anoDist, anoSpp, names = TRUE)
threshID(anoDist, anoSpp)
threshID(anoDist, anoSpp, threshold = 0.003)

data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

bestCloseMatch(doloDist, doloSpp)
bestCloseMatch(doloDist, doloSpp, threshold = 0.005)
nearNeighbour(doloDist, doloSpp)
nearNeighbour(doloDist, doloSpp, names=TRUE)
threshID(doloDist, doloSpp)
threshID(doloDist, doloSpp, threshold = 0.003)
```

nonConDist — *Nearest non-conspecific and maximum intra-specific distances*

Description

These functions give the distances to the nearest non-conspecific and furthest conspecific representatives for each individual in the dataset.

Usage

```
nonConDist(distobj, sppVector = NULL, propZero = FALSE, rmNA = FALSE)
```

Arguments

`distobj`	Distance matrix.
`sppVector`	Species vector (see `sppVector`). Default of NULL.
`propZero`	Logical. TRUE gives the proportion of zero distances.
`rmNA`	Logical. TRUE ignores missing values in the distance matrix. Default of FALSE

Details

`nonConDist` returns the minimum inter-specific distance for each individual.

`maxInDist` returns the maximum intra-specific distance for each individual.

These two functions can be used to create a version of the barcoding gap.

`minInDist` returns the minimum intra-specific distance for each individual.

Value

If `propZero=FALSE`, a numeric vector giving the distance of the closest non-conspecific individual (`nonConDist`) or the most distant conspecific individual (`maxInDist`).

If `propZero=TRUE`, a single number giving the proportion of zero distances.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

nonConDist(anoDist, anoSpp)
nonConDist(anoDist, anoSpp, propZero=TRUE)

maxInDist(anoDist, anoSpp)
maxInDist(anoDist, anoSpp, propZero=TRUE)

#Barcoding gap
inter <- nonConDist(anoDist, anoSpp)
intra <- maxInDist(anoDist, anoSpp)
graphics::hist(inter-intra)

#An alternative way of plotting the gap
bnd <- cbind(data.frame(inter, intra))
ord <- bnd[order(bnd$inter),]
graphics::plot(ord$inter, type="n", ylab="Percent K2P distance", xlab="Individual")
segCol <- rep("gray50", length(ord$inter))
segCol[ord$inter-ord$intra < 0] <- "red"
graphics::segments(x0=1:length(ord$inter), y0=ord$inter, y1=ord$intra, col=segCol, lwd=6)
```

`nucDiag` *Nucleotide diagnostics for species alignments*

Description

Determines the diagnostic nucleotides for each species given in `sppVector`.

Usage

```
nucDiag(DNAbin, sppVector)
```

Arguments

`DNAbin`	An object of class 'DNAbin'.
`sppVector`	The species vector (see `sppVector`).

Details

These functions provide a means for evaluating the presence of diagnostic nucleotides that distinguish species within an alignment. `nucDiag` returns the positions of bases corresponding to the definition of pure, simple diagnostic nucleotides given by Sarkar et al (2008).

`rnucDiag` runs a bootstrapping-style resampling test to evaluate the numbers of diagnostic nucleotides that might be expected by random assortment of specimens.

Value

`nucDiag` returns a list giving the pure, simple diagnostic nucleotides (i.e. those nucleotides that are fixed within species and different from all other species) for each species in the species vector. A result of `integer(0)` indicates there are no diagnostic nucleotides for those species.

`rnucDiag` returns a list containing the following elements:

`min`	The minimum number of diagnostic nucleotides in the sample.
`mean`	The mean number of diagnostic nucleotides in the sample.
`median`	The median number of diagnostic nucleotides in the sample.
`max`	The maximum number of diagnostic nucleotides in the sample.
`rndFreq`	A list of frequency distributions of the number of diagnostic nucleotides in groups formed by 1 sequence, 2 sequences, etc.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Sarkar, I., Planet, P., & DeSalle, R. (2008). CAOS software for use in character- based DNA barcoding. _Molecular Ecology Resources_ *8* 1256-1259

See Also

`slideNucDiag`, `rnucDiag`

Examples

```
data(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
 function(x) paste(x[1], x[2], sep="_"))

nucDiag(anoteropsis, anoSpp)


#To view the nucleotide values
anoNuc <- nucDiag(anoteropsis, anoSpp)
as.character(anoteropsis[ ,anoNuc[[1]][1] ])



data(sarkar)
sarkarSpp <- substr(dimnames(sarkar)[[1]], 1, 3)
nucDiag(sarkar, sarkarSpp)

## Not run:
rnucDiag(anoteropsis, anoSpp, n = 100)

## End(Not run)
```

`ordinDNA`	*Calculates a Principal Components Ordination of genetic distances*

Description

Calculates Principical Coonrdinates Analysis on a matrix of genetic distances and plots an ordination of the first two major axes.

Usage

```
ordinDNA(distobj, sppVector, ...)
```

Arguments

`distobj`	A distance matrix.
`sppVector`	The species vector (see `sppVector`).
`...`	Other arguments to be passed to `plot.ordinDNA`.

Details

This function is a wrapper for `cmdscale`, which performs a Principal Coordinates Analysis on the distance matrix given. In addition, it plots an ordination of the genetic distance matrix given, showing the relative distance between each of the species in the dataset. It is presented as an alternative to the neighbour-joining trees which are frequently used for the visualisation of DNA barcoding data. NJ trees show hypotheses of relationships, which are inappropriate for the questions usally asked in DNA barcoding studies.

The distance between the centroids of the clusters are roughly proportional to the genetic distances between the species. NOTE: it is important to remember that the plot shows only one plane of a multi-dimensional space. Species with overlapping circles are not necessarily conspecific. Further exploration is required.

Value

Plots an ordination of the first two major axes showing the positions of each individual (squares), the centroid of each species (circular bullet and name of species), and the variation in the species (large circle, the radius of which is the distance to the furthest individual from the centroid).

Additionally returns a list of class `"ordinDNA"` with the following elements:

`pco`	Output of the Principal Coordinates Analysis.
`sppVector`	Character vector giving the species vector.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`cmdscale`, `plot.ordinDNA`

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloOrd <- ordinDNA(doloDist, doloSpp)
doloOrd
```

`paa` *Population Aggregate Analysis*

Description

Conducts population aggregate analysis over a matrix of characters of interest.

Usage

```
paa(data, sppVector)
```

Arguments

`data`	A data matrix with columns as characters and rows as individuals.
`sppVector`	The species vector. See `sppVector`.

Details

When used on DNA sequences, the function treats gaps as seperate characters.

Value

A matrix with species as rows and characters as columns. Cells give the character state of each species if fixed, or "poly" if the character is polymorphic.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Sites, J. W. J., & Marshall, J. C. (2003). Delimiting species: a Renaissance issue in systematic biology. _Trends in Ecology and Evolution_ *18* (9), 462-470.

Examples

```
#Create some exemplar data
u <- sample(c(0,1), 16, replace=TRUE)
v <- rep(c(0,1), rep(8,2))
x <- rep(c(1,0), rep(8,2))
y <- sample(c(0,1), 16, replace=TRUE)
z <- rep(c(1,0), rep(8,2))

dat <- cbind(u,v,x,y,z)
popn <- rep(c("A","B", "C", "D"), rep(4,4))

paa(dat, popn)

#Use on DNA sequences
data(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
 function(x) paste(x[1], x[2], sep="_"))

paa(as.character(anoteropsis), anoSpp)
```

`plot.haploAccum` *Plotting haplotype accumulation curves*

Description

Plots the accumulation curves calculated by `haploAccum`.

Usage

```
## S3 method for class 'haploAccum'
plot(x, add = FALSE, ci = 2, ci.type = c("bar",
  "line", "polygon"), col = par("fg"), ci.col = col, ci.lty = 1,
  xlab, ylab = "Haplotypes", ylim, main = paste(x$method,
  "method of haplotype accumulation", sep = " "), ...)
```

Arguments

`x`	A ‘haploAccum’ object obtained from `haploAccum`.
`add`	Add graph to an existing graph.
`ci`	Multiplier for the calculation of confidence intervals from standard deviation. `ci = 0` prevents the drawing of confidence intervals.
`ci.type`	Type of confidence intervals: `"bar"` for vertical bars, `"line"` for lines, and `"polygon"` for a shaded area.
`col`	Colour for curve line.
`ci.col`	Colour for lines or shaded area when `"polygon"`.

`ci.lty`	Line type for confidence interval lines or border of the `"polygon"`.
`xlab`	Label for the X-axis.
`ylab`	Label for the Y-axis.
`ylim`	Y-axis limits.
`main`	Title of the plot.
`...`	Other parameters to pass to plot.

Value

Plots a haplotype accumulation curve and confidence intervals depending on the options given to `haploAccum`.

Author(s)

Jagoba Malumbres-Olarte <j.malumbres.olarte@gmail.com>.

References

Gotellli, N.J. & Colwell, R.K. (2001). Quantifying biodiversity: procedures and pitfalls in measurement and comparison of species richness. _Ecology Letters_ *4* 379–391.

Examples

```
data(dolomedes)
#Generate multiple haplotypes
doloHaplo <- dolomedes[sample(37, size = 200, replace = TRUE), ]
dolocurv <- haploAccum(doloHaplo, method = "random", permutations = 100)

graphics::plot(dolocurv)
graphics::plot(dolocurv, add = FALSE, ci = 2, ci.type = "polygon", col = "blue", ci.col = "red",
    ci.lty = 1)
```

`plot.ordinDNA` *Plot an 'ordinDNA' object*

Description

Plots an ordination of the Principal Components Analysis conducted by `ordinDNA`.

Usage

```
## S3 method for class 'ordinDNA'
plot(x, majorAxes = c(1, 2), plotCol = "default",
  trans = "CC", textcex = 0.7, pchCentroid = FALSE,
  sppBounds = "net", sppNames = TRUE, namePos = "top", ptPch = 21,
  ptCex = 0.5, netWd = 1, ...)
```

Arguments

`x`	An object of class ‘ordinDNA’.
`majorAxes`	Numeric. Gives the numbers of the major axes that should be plotted. Default of the first two major axes (`majorAxes = c(1,2)`)
`plotCol`	A vector of RGB colours giving the colours of the points and circles. Must be in the form of a character vector with elements "#XXXXXX" where XXXXXX gives the hexadecimal value for the colours desired. Default of `"default"`. Colours are recycled if necessary.
`trans`	A character vector giving the hexadecimal value for the transparency of the circles. Default of "CC".
`textcex`	Numeric. Controls the size of the text giving the species value of the circles.
`pchCentroid`	Numeric. Controls the shape of the point showing the centroid of the circle for each species. Default of FALSE, no plotting of centroid position.
`sppBounds`	Option to determine the method of visualising conspecific points. Options of `"net"` (the default), `"none"` or `"circles"`.
`sppNames`	Logical. Should species names be plotted? Default of TRUE.
`namePos`	Character vector of length 1 giving the position where the species names should be plotted. Possible values are: "top" and "bottom", anything else plots the names at the centroid.
`ptPch`	Numeric. Number of the symbol to be used for plotting. see `points`. Default of 21.
`ptCex`	Numeric. Number governing the size of the points. Default of 0.5.
`netWd`	Numeric. Number governing the width of the lines in the netowk. Default of 1.
`...`	Other arguments to be passed to `plot`.

Details

`plot.ordinDNA` calculates the centroid and radius of the most variable individual for each species in the multivariate space of the Principal Components Analysis object given.

`majorAxes` plots the axes in the form `c(x, y)`. The maximum number of axes calculated is the number of specimens in the dataset minus one.

`sppBounds` has the following options: `"net"` (the default) creates a complete graph between all individuals within a species. If `"circles"` is specified, a circle is drawn with a center fixed on the centroid, and a radius of the length to the maximally distant individual. Selecting the option of `"none"` means the individuals are not connected in any way.

Value

Plots an ordination of the first two major axes showing the positions of each individual (squares), the centroid of each species (circular bullet and name of species), and the variation in the species (large circle, the radius of which is the distance to the furthest individual from the centroid).

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`ordinDNA`, `cgraph`.

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloOrd <- ordinDNA(doloDist, doloSpp)

graphics::plot(doloOrd)
graphics::plot(doloOrd, majorAxes = c(1,3))
graphics::plot(doloOrd, textcex = 0.001)
graphics::plot(doloOrd, plotCol = c("#FF0000", "#00FF00", "#0000FF"))
graphics::plot(doloOrd, namesPos = "bottom")
graphics::plot(doloOrd, namesPos = "centre")

data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

anoOrd <- ordinDNA(anoDist, anoSpp)

plot(anoOrd, sppBounds = "circles")
```

`plot.slidWin` *Plot a 'slidWin' object*

Description

Graphical representation of the summary statistics derived from `slideAnalyses` and `slideBoxplots`

Usage

```
## S3 method for class 'slidWin'
plot(x, outliers = FALSE, ...)
```

Arguments

`x`	An object of class 'slidWin'.
`outliers`	Logical. When the results of `slideBoxplots` are being called, should the outliers be plotted? Default of FALSE.
`...`	Other arguments to be passed to `plot`.

Details

When boxplots of methods nonCon and interAll, the y-axis limits are constrained to the midpoint of the range covered by the boxplots, so that the intra-specific variation can be seen.

Value

Plots graphs depending on the options given to slideAnalyses or slideBoxplots.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

slideAnalyses, slideBoxplots.

Examples

```
data(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloSlide <- slideAnalyses(dolomedes,  doloSpp, 200, interval=10, treeMeasures=TRUE)

graphics::plot(doloSlide)

doloBox <- slideBoxplots(dolomedes,  doloSpp, 200, interval=10, method="overall")

graphics::plot(doloBox)


data(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

anoBox <- slideBoxplots(anoteropsis,  anoSpp, 200, interval=10, method="interAll")

graphics::plot(anoBox)
graphics::plot(anoBox, outliers=TRUE)
```

---

polyBalance *Balance of a phylogenetic tree with polytomies*

Description

This function computes the numbers of descendants for each dichotomous branch of a phylogenetic tree.

Usage

```
polyBalance(phy)
```

Arguments

`phy`	A tree of class 'phylo'.

Details

The function extends `balance` to allow the balance of a tree with polytomies to be calculated. When the tree is fully dichotomous, the result is identical to `balance`.

Value

A numeric matrix with two columns and one row for each node of the tree. The columns give the numbers of descendants on each node. Non-dichotomous nodes are reported as 'NA'.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`balance`.

Examples

```
set.seed(55)
tr <- ape::rtree(15)
tr2 <- ape::di2multi(tr, tol=0.02)
polyBalance(tr)
polyBalance(tr2)
```

`rankSlidWin` *Rank a 'slidWin' object.*

Description

Display the highest ranking windows measured by `slideAnalyses`.

Usage

```
rankSlidWin(slidWin, criteria = "mean_distance", num = 10)
```

Arguments

`slidWin`	An object of class 'slidWin', made using `slideAnalyses`.
`criteria`	Name of criteria to sort by. Can be any of the following: `"mean_distance"`, `"monophyly"`, `"clade_comparison"`,`"clade_comp_shallow"`, `"zero_noncon"`, `"zero_distances"`, `"diag_nuc"` or `"all"`. Default of `"mean_distance"` if distance measures have been calculated, otherwise `"monophyly"`.
`num`	Number of windows to return. Default of 10.

Details

The criteria for `rankSlidWin` correspond to the variables outputted by `slideAnalyses` and are sorted in the following manner:

`rankSlidWin` criterion:	`slideAnalyses` output:	Sorting method:
`"mean_distance"`	`"dist_mean_out"`	Ascending
`"monophyly"`	`"win_mono_out"`	Ascending
`"clade_comparison"`	`"comp_out"`	Ascending
`"clade_comp_shallow"`	`"comp_depth_out"`	Ascending
`"zero_noncon"`	`"noncon_out"`	Descending
`"zero_distances"`	`"zero_out"`	Descending
`"diag_nuc"`	`"nd_out"`	Ascending

Given a sequence of 1:10, the ascending method of sorting considers 10 as high. The descending method considers 1 as high.

The `"all"` criterion returns the windows that have the highest cumulative total score over all criteria.

Value

A data frame giving the values of the measures calculated by `slideAnalyses`, ranked to show the top 10 positions based on the criterion given.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`slideAnalyses`.

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloSlide <- slideAnalyses(dolomedes, doloSpp, 200, interval = 10, treeMeasures = TRUE)
```

```
rankSlidWin(doloSlide)
rankSlidWin(doloSlide, criteria = "zero_distances")

doloSlide2 <- slideAnalyses(dolomedes, doloSpp, 200, interval = 10, treeMeasures = FALSE)
rankSlidWin(doloSlide2)

doloSlide3 <- slideAnalyses(dolomedes, doloSpp, 200, interval = 10, distMeasures = FALSE,
    treeMeasures = TRUE)
rankSlidWin(doloSlide3)
```

read.BOLD *Downloads DNA sequences from the Barcode of Life Database (BOLD)*

Description

These functions allow DNA sequences to be downloaded from the Barcode of Life Database (BOLD).

Usage

```
read.BOLD(IDs)
```

Arguments

IDs	A character vector containing BOLD process ID numbers.

Details

`search.BOLD` retrieves BOLD process identification numbers for any given taxon using the API for BOLD version 3.0. By default, it only returns the first 500 process IDs for the given taxon. By selecting the option `exhaustive = TRUE`, the function can be made to search for more than 500 process IDs, but is much slower.

`stats.BOLD` retrieves the total number of records for the given taxon.

`read.BOLD` downloads the sequences associated with the process identification numbers using a brute force method of downloading the specimen record, then searching and splitting the HTML code to remove the relevant information. This process is likely to make the function fairly unstable if BOLD make any changes to their website.

Previous versions of `read.BOLD` used the eFetch web service offered by BOLD to enable batch retrieval of records, however from October 2012 BOLD deprecated eFetch without providing a replacement service.

Value

`search.BOLD` returns a character vector giving the process identification numbers of the specimens found by the search.

`read.BOLD` returns an object of class ‘DNAbin’. This object has the attributes "species", "accession_num", and "gene".

Warning

On 26 Oct 2011, attempts to access records using the eFetch system through a web browser resulted in an error, saying that eFetch and eSearch are offline for maintainance.

As of 7 March 2012, both functions have been modified to interface with the new BOLD architecture, and work as expected.

29 Oct 2012: It appears that BOLD has taken eFetch offline permanently, rendering `read.BOLD` as it currently stands useless. While we may be able to work out something, this will require a complete rewrite of the function. `search.BOLD` continues to work as intended.

17 Dec 2012: A new version of `read.BOLD` has been released that appears to work (for the time being).

15 Feb 2018: 'read.BOLD' is deprecated. Please use the rOpenSci 'bold' package for better functionality.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

BOLD web services: `http://www.boldsystems.org/index.php/resources/api?type=webservices`.

BOLD version 3.0 `http://v3.boldsystems.org/`.

See Also

`stats.BOLD`, `search.BOLD`, `read.GB`. `help`, ~~~

Examples

```
## Not run:
stats.BOLD("Pisauridae")

search.BOLD(c("Danio kyathit", "Dolomedes", "Sitona discoideus"))

nn <- search.BOLD("Pisauridae")
pisaurid <- read.BOLD(nn)

ape::write.dna(pisaurid, "filename.fas", format="fasta")
## End(Not run)
```

read.GB *Download sequences from Genbank with metadata.*

Description

Downloads sequences associated with the given accession numbers into a ‘DNAbin’ class.

Usage

```
read.GB(access.nb, seq.names = access.nb, species.names = TRUE,
  gene = TRUE, access = TRUE, as.character = FALSE)
```

Arguments

`access.nb`	A character vector giving the GenBank accession numbers to download.
`seq.names`	A character vector giving the names to give to each sequence. Defaults to "accession number | species name".
`species.names`	Logical. Should species names be downloaded? Default of TRUE.
`gene`	Logical. Should the name of the gene region be downloaded? Default of TRUE.
`access`	Logical. Should the accession number be downloaded? Default of TRUE.
`as.character`	Logical. Should the sequences be returned as character vector? Default of FALSE, function returns sequences as a ‘DNAbin’ object.

Details

This function is a modification of `read.GenBank` to include metadata with each sequence. Additional data currently implemented are the species names and the gene region from which sequences were derived.

Value

A ’DNAbin’ object with the following attributes: `"species"`, `"gene"`, and `"accession_num"`.

Warning

15 Feb 2018: ’read.GB’ is deprecated. Please use the rOpenSci packages ’rentrez’ and ’traits’, or ’ape’ for better functionality.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`read.GenBank`.

Examples

```
## Not run:
read.GB("AY059961")

#Download the sequences making data(anoteropsis) from Genbank
nums <- 59961:59993
seqs <- paste("AY0", nums, sep="")
dat <- read.GB(seqs)

attr(dat, "species")
attr(dat, "gene")
attr(dat, "accession_num")
## End(Not run)
```

`rmSingletons` *Detect and remove singletons*

Description

A utility to detect and remove species represented only by singletons.

Usage

```
rmSingletons(sppVector, exclude = TRUE)
```

Arguments

`sppVector`	Vector of species names. (see `sppVector`).
`exclude`	Logical. Should singletons be removed? Default of TRUE.

Details

When `exclude = TRUE` (the default), singletons are excluded and the vector returns the index of all non-singletons in the dataset. When `exclude = FALSE`, the indices of the singletons are presented.

Value

Returns a numeric vector giving the indices of the selected individuals.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))

rmSingletons(anoSpp)
rmSingletons(anoSpp, exclude=FALSE)

data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

rmSingletons(doloSpp)
rmSingletons(doloSpp, exclude=FALSE)
```

`rnucDiag` *Nucleotide diagnostics for species alignments*

Description

Determines the diagnostic nucleotides for each species given in `sppVector`.

Usage

```
rnucDiag(DNAbin, sppVector, n = 100)
```

Arguments

`DNAbin`	An object of class 'DNAbin'.
`sppVector`	The species vector (see `sppVector`).
`n`	The number of pseudoreplicates to perform. Default of 100

Details

These functions provide a means for evaluating the presence of diagnostic nucleotides that distinguish species within an alignment. `nucDiag` returns the positions of bases corresponding to the definition of pure, simple diagnostic nucleotides given by Sarkar et al (2008).

`rnucDiag` runs a bootstrapping-style resampling test to evaluate the numbers of diagnostic nucleotides that might be expected by random assortment of specimens.

Value

`nucDiag` returns a list giving the pure, simple diagnostic nucleotides (i.e. those nucleotides that are fixed within species and different from all other species) for each species in the species vector. A result of `integer(0)` indicates there are no diagnostic nucleotides for those species.

`rnucDiag` returns a list containing the following elements:

`min`	The minimum number of diagnostic nucleotides in the sample.
`mean`	The mean number of diagnostic nucleotides in the sample.
`median`	The median number of diagnostic nucleotides in the sample.
`max`	The maximum number of diagnostic nucleotides in the sample.
`rndFreq`	A list of frequency distributions of the number of diagnostic nucleotides in groups formed by 1 sequence, 2 sequences, etc.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Sarkar, I., Planet, P., & DeSalle, R. (2008). CAOS software for use in character- based DNA barcoding. _Molecular Ecology Resources_ *8* 1256-1259

See Also

`slideNucDiag`, `rnucDiag`

Examples

```
data(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
 function(x) paste(x[1], x[2], sep="_"))

nucDiag(anoteropsis, anoSpp)


#To view the nucleotide values
anoNuc <- nucDiag(anoteropsis, anoSpp)
as.character(anoteropsis[ ,anoNuc[[1]][1] ])



data(sarkar)
sarkarSpp <- substr(dimnames(sarkar)[[1]], 1, 3)
nucDiag(sarkar, sarkarSpp)

## Not run:
rnucDiag(anoteropsis, anoSpp, n = 100)

## End(Not run)
```

rosenberg *Rosenberg's probability of reciprocal monophyly*

Description

This function computes Rosenberg's probability of reciprocal monophyly for each dichotomous node of a phylogenetic tree.

Usage

```
rosenberg(phy)
```

Arguments

phy	A tree of class 'phylo'.

Details

Because ape plots node labels in a different manner to the method in which they are stored, when plotting the node labels made by `rosenberg`, make sure the node argument is given as shown in the examples below.

Value

A numeric vector with names giving the node numbers of `phy`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Rosenberg, N. A. (2007). Statistical tests for taxonomic distinctiveness from observations of monophyly. _Evolution_ *61* (2), 317-323.

See Also

`nodelabels`.

Examples

```
data(anoteropsis)
anoTr <- ape::nj(ape::dist.dna(anoteropsis))
anoLab <- rosenberg(anoTr)
ape::plot.phylo(anoTr)
ape::nodelabels(round(anoLab,3), node=as.numeric(names(anoLab)))

data(dolomedes)
doloTr <- ape::nj(ape::dist.dna(dolomedes))
```

```
doloRose <- rosenberg(doloTr)
ape::plot.phylo(doloTr)
ape::nodelabels(round(doloRose, 3))

#Colour circles for nodes with a probability < 0.005
doloNodes <- doloRose < 0.005
doloLabs <- doloRose
doloLabs[doloNodes] <- "blue"
doloLabs[!doloNodes] <- "red"
ape::plot.phylo(doloTr, cex=0.7)
ape::nodelabels(pch=21, bg=doloLabs, node=as.numeric(names(doloLabs)), cex=2)
graphics::legend(x=0.015, y=16.13, legend=c("significant", "not significant"), pch=21,
    pt.bg=c("blue", "red"), bty="n", pt.cex=2)
```

salticidae *Cytochrome oxidase I (COI) sequences of world-wide species of Salticidae*

Description

A set of 41 sequences of the mitochondrial protein-coding gene cytochrome oxidase I from 41 species of the jumping spider family Salticidae.The sequences are available on GenBank as accession numbers AY297360 through AY297400.

Format

A DNAbin object containing 41 sequences with a length of 409 base pairs stored as a list.

Source

Maddison, W. P., and Hedin, M. C. (2003). Jumping spider phylogeny (Araneae: Salticidae). _Invertebrate Systematics_ *17* 529-549.

sarkar *Dummy sequences illustrating the categories of diagnostic nucleotides*

Description

A set of 8 dummy sequences published in Sarkar et al 2008 to illustrate the different categories of diagnostic nucleotides.

Format

A DNAbin object containing 8 sequences with a length of 18 base pairs stored as a matrix.

Source

Sarkar, I., Planet, P., & DeSalle, R. (2008). CAOS software for use in character- based DNA barcoding. _Molecular Ecology Resources_ *8* 1256-1259

`search.BOLD`	*Downloads DNA sequences from the Barcode of Life Database (BOLD)*

Description

These functions allow DNA sequences to be downloaded from the Barcode of Life Database (BOLD).

Usage

```
search.BOLD(taxon, exhaustive = FALSE)
```

Arguments

`taxon`	A character vector of the names of the taxa of interest.
`exhaustive`	Logical. Should the function search for more than 500 process IDs? Default of FALSE.

Details

`search.BOLD` retrieves BOLD process identification numbers for any given taxon using the API for BOLD version 3.0. By default, it only returns the first 500 process IDs for the given taxon. By selecting the option `exhaustive = TRUE`, the function can be made to search for more than 500 process IDs, but is much slower.

`stats.BOLD` retrieves the total number of records for the given taxon.

`read.BOLD` downloads the sequences associated with the process identification numbers using a brute force method of downloading the specimen record, then searching and splitting the HTML code to remove the relevant information. This process is likely to make the function fairly unstable if BOLD make any changes to their website.

Previous versions of `read.BOLD` used the eFetch web service offered by BOLD to enable batch retrieval of records, however from October 2012 BOLD deprecated eFetch without providing a replacement service.

Value

`search.BOLD` returns a character vector giving the process identification numbers of the specimens found by the search.

`read.BOLD` returns an object of class 'DNAbin'. This object has the attributes "species", "accession_num", and "gene".

Warning

On 26 Oct 2011, attempts to access records using the eFetch system through a web browser resulted in an error, saying that eFetch and eSearch are offline for maintainance.

As of 7 March 2012, both functions have been modified to interface with the new BOLD architecture, and work as expected.

29 Oct 2012: It appears that BOLD has taken eFetch offline permanently, rendering `read.BOLD` as it currently stands useless. While we may be able to work out something, this will require a complete rewrite of the function. `search.BOLD` continues to work as intended.

17 Dec 2012: A new version of `read.BOLD` has been released that appears to work (for the time being). 15 Feb 2018: 'search.BOLD' is deprecated. Please use the rOpenSci 'bold' package for better functionality.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

BOLD web services: `http://www.boldsystems.org/index.php/resources/api?type=webservices`.

BOLD version 3.0 `http://v3.boldsystems.org/`.

See Also

`stats.BOLD`, `search.BOLD`, `read.GB`. `help`, ~~~

Examples

```
## Not run:
stats.BOLD("Pisauridae")

search.BOLD(c("Danio kyathit", "Dolomedes", "Sitona discoideus"))

nn <- search.BOLD("Pisauridae")
pisaurid <- read.BOLD(nn)

ape::write.dna(pisaurid, "filename.fas", format="fasta")
## End(Not run)
```

`seeBarcode` *Create illustrative barcodes*

Description

This function plots an illustrative barcode consisting of vertical bands in four colours corresponding to the DNA bases adenine (A), cytosine (C), guanine (G) and thiamine (T).

Usage

```
seeBarcode(seq, col = c("green", "blue", "black", "red"))
```

Arguments

`seq`	A single sequence of class 'DNAbin'.
`col`	A character vector of length 4 giving colours to represent A, G, C and T respectively.

Details

Green, blue, black and red are the standard colours representing A, G, C and T respectively.

Value

Plots an illustrative barcode.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
graphics::layout(matrix(1:6, ncol=1))
graphics::par(mar=c(0.5, 0, 0.5, 0))
data(woodmouse)
seeBarcode(woodmouse[1,])
seeBarcode(woodmouse[1,], col=c("pink", "orange", "steelblue", "yellow"))
seeBarcode(woodmouse[1,], col=c("black", "white", "white", "black"))
apply(woodmouse[1:3,], MARGIN=1, FUN=seeBarcode)
```

`seqStat` *Sequence statistics*

Description

Utility that produces a table giving summary statistics for a 'DNAbin' object.

Usage

```
seqStat(DNAbin, thresh = 500)
```

Arguments

`DNAbin`	Alignment of class 'DNAbin'.
`thresh`	Threshold sequence length. Default of 500 (minimum length for official DNA barcodes).

Details

This function considers bases coded as '?', 'N' and '-' as missing data.

Value

A table giving the minimum, maximum, mean and median sequence lengths, and the number of sequences with lengths below the given threshold.

Author(s)

Rupert Collins <rupertcollins@gmail.com>

Examples

```
data(anoteropsis)
seqStat(anoteropsis)
```

`slideAnalyses` *Sliding window analyses*

Description

Wraps a number of measures used in sliding window analyses into one easy-to-use function.

Usage

```
slideAnalyses(DNAbin, sppVector, width, interval = 1,
  distMeasures = TRUE, treeMeasures = FALSE)
```

Arguments

`DNAbin`	A DNA alignment of class 'DNAbin'.
`sppVector`	Species vector (see `sppVector`).
`width`	Desired width of windows in number of nucleotides.
`interval`	Distance between each window in number of nucleotides. Default of 1. Giving the option of 'codons' sets the size to 3.
`distMeasures`	Logical. Should distance measures be calculated? Default of TRUE.
`treeMeasures`	Logical. Should tree-based measures be calculated? Default of FALSE.

Details

Distance measures include the following: proportion of zero non-conspecific distances, number of diagnostic nucleotides, number of zero-length distances, and overall mean distance.

Tree-based measures include the following: proportion of species that are monophyletic, proportion of clades that are identical between the neighbour joining tree calculated for the window and the tree calculated for the full dataset, and the latter with `method="shallow"`.

Tree-based measures are a lot more time-intensive than distance measures. When dealing with lots of taxa and short windows, this part of the function can take hours.

Both distance and tree measures are calculated from a K2P distance matrix created from the data with the option `pairwise.deletion = TRUE`. When sequences with missing data are compared with other sequences, a `NA` distance results. These are ignored in the calculation of `slideAnalyses` distance metrics. However, the tree measures cannot cope with this missing data, and so no result is returned for windows where some sequences solely contain missing data.

Value

An object of class 'slidWin' which is a list containing the following elements:

`win_mono_out`	Proportion of species that are monophyletic.
`comp_out`	Proportion of clades that are identical between the NJ tree calculated for the window and the tree calculated for the full dataset.
`comp_depth_out`	Proportion of shallow clades that are identical.
`pos_tr_out`	Index of window position for tree-based analyses.
`noncon_out`	Proportion of zero non-conspecific distances.
`nd_out`	The sum of diagnostic nucleotides for each species.
`zero_out`	The number of zero-length distances.
`dist_mean_out`	Overall mean K2P distance of each window.
`pos_out`	Index of window position.
`dat_zero_out`	Number of zero inter-specific distances in the full dataset.
`boxplot_out`	Always FALSE. Required for `plot.slidWin`.
`distMeasures`	Value of argument. Required for `plot.slidWin`.
`treeMeasures`	Value of argument. Required for `plot.slidWin`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`dist.dna`, `plot.slidWin`, `rankSlidWin`, `slideNucDiag`.

Examples

```
## Not run:
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

slideAnalyses(dolomedes, doloSpp, 200, interval=10, treeMeasures=TRUE)

## End(Not run)
```

`slideBoxplots` *Boxplots across windows*

Description

Calculates boxplots of genetic distances using sliding windows.

Usage

```
slideBoxplots(DNAbin, sppVector, width, interval = 1,
  method = "nonCon")
```

Arguments

`DNAbin`	A DNA alignment of class 'DNAbin'.
`sppVector`	A species vector (see `sppVector`).
`width`	Width of windows.
`interval`	Distance between each window in number of base pairs. Default of 1. Giving the option of `"codons"` sets the size to 3.
`method`	Options of `"overall"`, `"interAll"` or `"nonCon"` (the default).

Details

Giving `method="overall"` calculates the boxplot for the distance matrix of each window.

Giving `method="interAll"` calculates boxplots for the inter- and intra-specific distances of each window, showing the result for ALL inter-specific distances.

Giving `method="nonCon"` calculates boxplots for the inter- and intra-specific distances of each window, showing the result for only the nearest-conspecific distances for each individual.

Value

A list with

`treeMeasures`	Logical. Tree measures calculated? Always FALSE.
`distMeasures`	Logical. Distance measures calculated? Always FALSE.
`bp_out`	If `method="overall"`, contains the boxplot objects of each window.
`bp_InterSpp_out`	If `method!="overall"`, contains the boxplot objects of the interspecific distances of each window.
`bp_IntraSpp_out`	If `method!="overall"`, contains the boxplot objects of the intraspecific distances of each window.
`bp_range_out`	range of y-axis values.
`pos_out`	x-axis values.
`boxplot_out`	Logical. Boxplots calculated? Always TRUE.
`method`	The method used for calculating boxplots. `"overall"`, `"interAll"` or `"nonCon"`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`boxplot`, `plot.slidWin`, `slideAnalyses`, `slidingWindow`.

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloNonCon <- slideBoxplots(dolomedes, doloSpp, 200, interval=10)
graphics::plot(doloNonCon)

doloOverall <- slideBoxplots(dolomedes, doloSpp, 200, interval=10, method="overall")
graphics::plot(doloOverall)

doloInterall <- slideBoxplots(dolomedes, doloSpp, 200, interval=10, method="interAll")
graphics::plot(doloInterall)
```

slideNucDiag *Sliding nucleotide diagnostics*

Description

Calculates the number of diagnostic nucleotides in sliding windows.

Usage

```
slideNucDiag(DNAbin, sppVector, width, interval = 1)
```

Arguments

`DNAbin`	A DNA alignment of class 'DNAbin'.
`sppVector`	Species vector (see `sppVector`).
`width`	Desired width of windows in number of base pairs.
`interval`	Distance between each window in number of base pairs. Default of 1. Giving the option of `"codons"` sets the size to 3.

Details

Determines the number of diagnostic nucleotides for each species in each window.

Value

A matrix giving the number of diagnostic nucleotides for each species (rows) in each window (columns).

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`slideAnalyses`, `slideBoxplots`, `slidingWindow`.

Examples

```
data(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

slideNucDiag(dolomedes, doloSpp, 200, interval = 3)

slidND <- slideNucDiag(dolomedes, doloSpp, 200, interval = 3)

#Number of basepairs for each species
graphics::matplot(t(slidND), type = "l")
```

```
#Number of basepairs for a single species
graphics::plot(slidND[4, ], type = "l")

#Total number of basepairs per window
graphics::plot(colSums(slidND), type = "l")
```

`slidingWindow` *Create windows along an alignment*

Description

Creates windows of a specified width along a DNA alignment.

Usage

```
slidingWindow(DNAbin, width, interval = 1)
```

Arguments

`DNAbin`	A DNA alignment of class ‘DNAbin’.
`width`	Width of each window.
`interval`	Numeric or option of `"codons"`. This sets interval between windows. Default of 1. Setting the option to "codons" gives an interval of 3.

Details

Sliding window analyses are often used to determine the variability along sequences. This can be useful for investigating whether there is evidence for recombination, developing shorter genetic markers, or for determining variation within a gene.

Analyses can be conducted on each window using `lapply`.

Value

A list of ‘DNAbin’ objects, with each alignment being `width` bases in length. The list has length of the DNA alignment minus the width. The positions covered by each window can be retreived with `attr(x,"window")`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`lapply`, `slideAnalyses`, `slideBoxplots`.

Examples

```
data(woodmouse)
woodmouse <- woodmouse[,1:20]
win1 <- slidingWindow(woodmouse, width = 10)
length(win1)

win2 <- slidingWindow(woodmouse, width = 10, interval = 2)
length(win2)

win3 <- slidingWindow(woodmouse, width = 10, interval = "codons")
length(win3)

win4 <- slidingWindow(woodmouse, width = 15)
length(win4)
attr(win4[[1]], "window")
attr(win4[[2]], "window")
```

`sppDist` *Intra and inter-specific distances*

Description

Separates a distance matrix into its inter- and intra-specific components.

Usage

```
sppDist(distobj, sppVector)
```

Arguments

`distobj`	A distance matrix.
`sppVector`	The species vector (see `sppVector`).

Details

This function can be used to produce histograms and other charts exploring the ‘barcode gap’, such as in the examples below.

Value

A list with two elements:

`inter`	A numeric vector containing ALL inter-specific pairwise distances.
`intra`	A numeric vector containing ALL intra-specific pairwise distances.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

sppDistMatrix.

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

doloSpDist <- sppDist(doloDist, doloSpp)

doloSpDist

#Histogram of the barcode gap
transGreen <- rgb(0, 1, 0, 0.5) #Make a slightly transparent colour to see some overlap
graphics::hist(doloSpDist$inter, col="grey")
graphics::hist(doloSpDist$intra, col=transGreen, add=TRUE)

#Boxplot of the same
graphics::boxplot(doloSpDist)
```

`sppDistMatrix` *Mean intra- and inter-specific distance matrix*

Description

Creates a matrix giving the mean distances within and between species.

Usage

```
sppDistMatrix(distobj, sppVector)
```

Arguments

`distobj`	A distance matrix.
`sppVector`	The species vector (see sppVector).

Value

A square matrix with dimensions `length(sppVector)`. It contains the mean intra specific distances down the diagonal, and the mean pairwise distance between the species in the triangles. The two triangles are identical.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

sppDistMatrix(doloDist, doloSpp)
```

---

sppVector *Species Vectors*

Description

A grouping variable that gives an identity to the individuals in various analyses.

Details

Species vectors are the key concept behind a lot of `spider`'s functionality. They are the method used to group data from individuals into species. It is important to note that "species" in this context can mean any cluster (real or otherwise) that is of interest. Populations, demes, subspecies and genera could be the taxa segregated by "species vectors".

The two characteristics of a species vector are UNIQUENESS between species and CONSISTENCY within them. R recognises differences of a single character between elements, leading to `spider` considering these elements to represent different species.

There is an easy way and a hard way to create species vectors. The hard way is to type out each element in the vector, making sure no typos or alignment errors are made.

The easy way is to add species designations into your data matrix from the beginning in such a way that it is easy to use R's data manipulation tools to create a species vector from the names of your data. See the examples for a few ways to do this.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

Functions for creating species vectors: `strsplit`, `substr`, `sapply`.

Functions that use species vectors: `nearNeighbour`, `monophyly`, `nonConDist`, `nucDiag`, `rmSingletons`, `slideAnalyses`, `slideBoxplots`, `sppDist`, `sppDistMatrix`, `threshOpt`.

Examples

```
data(dolomedes)
#Dolomedes species vector
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

data(anoteropsis)
#Anoteropsis species vector
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))
```

`stats.BOLD` *Downloads DNA sequences from the Barcode of Life Database (BOLD)*

Description

These functions allow DNA sequences to be downloaded from the Barcode of Life Database (BOLD).

Usage

```
stats.BOLD(taxon)
```

Arguments

`taxon`	A character vector of the names of the taxa of interest.

Details

`search.BOLD` retrieves BOLD process identification numbers for any given taxon using the API for BOLD version 3.0. By default, it only returns the first 500 process IDs for the given taxon. By selecting the option `exhaustive = TRUE`, the function can be made to search for more than 500 process IDs, but is much slower.

`stats.BOLD` retrieves the total number of records for the given taxon.

`read.BOLD` downloads the sequences associated with the process identification numbers using a brute force method of downloading the specimen record, then searching and splitting the HTML code to remove the relevant information. This process is likely to make the function fairly unstable if BOLD make any changes to their website.

Previous versions of `read.BOLD` used the eFetch web service offered by BOLD to enable batch retrieval of records, however from October 2012 BOLD deprecated eFetch without providing a replacement service.

Value

search.BOLD returns a character vector giving the process identification numbers of the specimens found by the search.

read.BOLD returns an object of class ‘DNAbin’. This object has the attributes "species", "accession_num", and "gene".

Warning

On 26 Oct 2011, attempts to access records using the eFetch system through a web browser resulted in an error, saying that eFetch and eSearch are offline for maintainance.

As of 7 March 2012, both functions have been modified to interface with the new BOLD architecture, and work as expected.

29 Oct 2012: It appears that BOLD has taken eFetch offline permanently, rendering read.BOLD as it currently stands useless. While we may be able to work out something, this will require a complete rewrite of the function. search.BOLD continues to work as intended.

17 Dec 2012: A new version of read.BOLD has been released that appears to work (for the time being). 15 Feb 2018: ’stats.BOLD’ is deprecated. Please use the rOpenSci ’bold’ package for better functionality.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

BOLD web services: http://www.boldsystems.org/index.php/resources/api?type=webservices.

BOLD version 3.0 http://v3.boldsystems.org/.

See Also

stats.BOLD, search.BOLD, read.GB. help, ~~~

Examples

```
## Not run:
stats.BOLD("Pisauridae")

search.BOLD(c("Danio kyathit", "Dolomedes", "Sitona discoideus"))

nn <- search.BOLD("Pisauridae")
pisaurid <- read.BOLD(nn)

ape::write.dna(pisaurid, "filename.fas", format="fasta")
## End(Not run)
```

tajima.K *Calculate Tajima's K index of divergence*

Description

Calculates Tajima's K index of divergence.

Usage

```
tajima.K(DNAbin, prop = TRUE)
```

Arguments

`DNAbin`	An object of class 'DNAbin'.
`prop`	Logical. Should the function report the number of substitutions per nucleotide? Default of TRUE.

Value

A vector of length 1. If `prop = FALSE`, the mean number of substitutions between any two sequences is returned. If `prop = TRUE` (the default), this number is returned as the mean number of substitutions per nucleotide (i.e. the above divided by the length of the sequences).

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Tajima, F. (1983). Evolutionary relationship of DNA sequences in finite populations. _Genetics_ *105*, 437-460.

See Also

`dist.dna`.

Examples

```
data(anoteropsis)
tajima.K(anoteropsis)
tajima.K(anoteropsis, prop = FALSE)
```

tclust *Clustering by a threshold*

Description

Identifies clusters, excluding individuals greater than the threshold from any member.

Usage

```
tclust(distobj, threshold = 0.01)
```

Arguments

`distobj`	A distance object (usually from `dist.dna`).
`threshold`	Distance cutoff for clustering. Default of 0.01 (1%).

Details

If two individuals are more distant than `threshold` from each other, but both within `threshold` of a third, all three are contained in a single cluster.

Value

A list with each element giving the index of the individuals contained in each cluster.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

See Also

`dist.dna`, `localMinima`. See Also as `help`, ~~~

Examples

```

data(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
 function(x) paste(x[1], x[2], sep="_"))
anoDist <- ape::dist.dna(anoteropsis)

tclust(anoDist)

#Names of individuals
anoClust <- tclust(anoDist)
lapply(anoClust, function(x) anoSpp[x])
```

threshID *Measures of identification accuracy*

Description

Tests of barcoding efficacy using distance-based methods.

Usage

```
threshID(distobj, sppVector, threshold = 0.01, names = FALSE)
```

Arguments

`distobj`	A distance object (usually from `dist.dna`).
`sppVector`	Vector of species names. See `sppVector`.
`threshold`	Distance cutoff for identifications. Default of 0.01 (1%).
`names`	Logical. Should the names of the nearest match be shown? Default of FALSE.

Details

These functions test barcoding efficacy. All sequences must be identified prior to testing. Each sequence is considered an unknown while the remaining sequences in the dataset constitute the DNA barcoding database that is used for identification. If the identification from the test is the same as the pre-considered identification, a correct result is returned.

`bestCloseMatch` conducts the "best close match" analysis of Meier et al. (2006), considering the closest individual unless it is further than the given threshold, which results in no identification. More than one species tied for closest match results in an assignment of "ambiguous". When the threshold is large, this analysis will return essentially the same result as `nearNeighbour`. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

`nearNeighbour` finds the closest individual and returns if their names are the same (TRUE) or different (FALSE). If `names = TRUE`, the name of the closest individual is returned. Ties are decided by majority rule.

`threshID` conducts a threshold-based analysis, similar to that conducted by the "Identify Specimen" tool provided by the Barcode of Life Database (`http://www.boldsystems.org/index.php/IDS_OpenIdEngine`). It is more inclusive than `bestCloseMatch`, considering ALL sequences within the given threshold. If `names = TRUE`, a list is returned containing the names of all species represented by specimens within the threshold.

These functions are not recommended as identification tools, though they can be used as such when `names = TRUE`.

Value

bestCloseMatch and threshID return a character vector giving the identification status of each individual.

"correct"	The name of the closest match is the same
"incorrect"	The name of the closest match is different
"ambiguous"	More than one species is the closest match (bestCloseMatch), or is within the given threshold (threshID)
"no id"	No species are within the threshold distance

nearNeighbour returns a logical vector or (if names = TRUE) the name for the nearest individual.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Meier, R., Shiyang, K., Vaidya, G., & Ng, P. (2006). DNA barcoding and taxonomy in Diptera: a tale of high intraspecific variability and low identification success. _Systematic Biology_ *55* (5) 715-728.

See Also

nearNeighbour, threshID, dist.dna, sppVector Also as help, ~~~

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split = "_"),
    function(x) paste(x[1], x[2], sep = "_"))

bestCloseMatch(anoDist, anoSpp)
bestCloseMatch(anoDist, anoSpp, threshold = 0.005)
nearNeighbour(anoDist, anoSpp)
nearNeighbour(anoDist, anoSpp, names = TRUE)
threshID(anoDist, anoSpp)
threshID(anoDist, anoSpp, threshold = 0.003)

data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)

bestCloseMatch(doloDist, doloSpp)
bestCloseMatch(doloDist, doloSpp, threshold = 0.005)
nearNeighbour(doloDist, doloSpp)
nearNeighbour(doloDist, doloSpp, names=TRUE)
threshID(doloDist, doloSpp)
threshID(doloDist, doloSpp, threshold = 0.003)
```

threshOpt *Threshold optimisation*

Description

Determines the positive, negative, false positive and false negative rates of identification accuracy for a given threshold.

Usage

```
threshOpt(distobj, sppVector, threshold = 0.01)
```

Arguments

`distobj`	Distance matrix.
`sppVector`	Species vector (see `sppVector`).
`threshold`	Threshold distance for delimiting intra- and inter-specific variation. Default of 0.01.

Details

When run over a range of thresholds, this function allows the optimisation of threshold values based on minimising the identification error rates. See the example below for more details.

Value

A table giving the threshold and number of negative and positive identifications, number of false negative and false positive identifications, and the cumulative error.

Author(s)

Rupert Collins <rupertcollins@gmail.com>

References

Meyer, C. P., and Paulay, G. (2005). DNA barcoding: error rates based on comprehensive sampling. _PLoS Biology_ *3* (12), 2229-2238.

See Also

`localMinima`.

Examples

```
data(anoteropsis)
anoDist <- ape::dist.dna(anoteropsis)
anoSpp <- sapply(strsplit(dimnames(anoteropsis)[[1]], split="_"),
    function(x) paste(x[1], x[2], sep="_"))
threshOpt(anoDist, anoSpp)

data(dolomedes)
doloDist <- ape::dist.dna(dolomedes)
doloSpp <- substr(dimnames(dolomedes)[[1]], 1, 5)
threshOpt(doloDist, doloSpp)

#Conduct the analysis over a range of values to determine the optimum threshold
threshVal <- seq(0.001,0.02, by = 0.001)
opt <- lapply(threshVal, function(x) threshOpt(doloDist, doloSpp, thresh = x))
optMat <- do.call(rbind, opt)
graphics::barplot(t(optMat)[4:5,], names.arg=optMat[,1], xlab="Threshold values",
    ylab="Cumulative error")
graphics::legend(x = 2.5, y = 29, legend = c("False positives", "False negatives"),
    fill = c("grey75", "grey25"))
```

---

tiporder *Orders tip labels by their position on the tree.*

Description

Provides an ordered vector of tip labels, corresponding to their position on the tree.

Usage

```
tiporder(phy, labels = TRUE)
```

Arguments

phy	A tree of class ‘phylo’.
labels	Logical. Should labels be printed? If FALSE, the indices are given. Default of TRUE.

Value

A character or numeric vector giving the names of the tip in the order of their position on the tree. The order is that from top to bottom when the tree is plotted with `direction = "rightwards"`.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(anoteropsis)
anoTree <- ape::nj(ape::dist.dna(anoteropsis))
tiporder(anoTree)
tiporder(anoTree, labels = FALSE)


data(woodmouse)
woodTree <- ape::nj(ape::dist.dna(woodmouse))
tiporder(woodTree)
tiporder(ape::ladderize(woodTree))
```

`titv` *Number of pairwise transitions and transversions in an alignment.*

Description

Calculates the number of pairwise transitions and transversions between sequences.

Usage

```
titv(DNAbin)
```

Arguments

`DNAbin`	A DNA alignment of class 'DNAbin'.

Value

A square matrix with dimensions of `length(dat)`. The upper triangle contains the number of transversions. The lower triangle contains the number of transitions.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

Examples

```
data(dolomedes)

subs <- titv(dolomedes)

#Transversions
subs[upper.tri(subs)]
tv <- t(subs)
tv <- tv[lower.tri(tv)]

#Transitions
```

```
ti <- subs[lower.tri(subs)]


#Saturation plot
doloDist <- ape::dist.dna(dolomedes)
graphics::plot(doloDist, ti, type="p", pch=19, col="blue",
    main="Saturation plot of number of transitions and transversions\n
    against K2P distance. Red: transversions. Blue: transitions")
graphics::points(doloDist, tv, pch=19, col="red")
```

tree.comp *Tree comparisons*

Description

Compares the clades between two trees.

Usage

```
tree.comp(phy1, phy2, method = "prop")
```

Arguments

`phy1, phy2`	Trees of class 'phylo' to compare.
`method`	One of the following options: • `"prop"`—returns the proportion of clades that are the same between the two trees • `"shallow"`—returns the proportion of shallow clades (clades where `node.depth` < median `node.depth`) that are the same between the two trees default of `"prop"`. • `"PH85"`—returns the topological distance of Penny and Hendy (1985).

Details

This function is a modification of the `dist.topo` function in ape to give similarity between the two trees as a proportion, and to account for the unreliable resolution of deeper nodes that affect some methods of tree construction (such as NJ).

It is important that the tip labels of the two trees are the same. If the tip labels are different between the two trees, the method will not recognise any similarity between them.

This function does not take into account differences in branch length. The `"score"` method in `dist.topo` does this if desired.

Value

Numeric vector of length 1.

If `method = "prop"`, the number returned is the proportion of nodes in the first tree for which there is a node in the second that contains the same tips. Higher number represents greater similarity. If it is 1, the trees are identical. If 0, the trees have no similarity whatsoever.

When `method = "shallow"`, only those nodes tipwards of the median node depth are taken into account. This will not be useful for small trees, but may be helpful with larger datasets.

`"PH85"` is the Penny and Hendy (1985) distance. This measure is the default of `dist.topo`. In this measure, the smaller the number, the closer the trees are. If the trees are identical, this results in 0.

Author(s)

Samuel Brown <s_d_j_brown@hotmail.com>

References

Penny, D. and Hendy, M. D. (1985) The use of tree comparison metrics. _Systematic Zoology_ *34* 75-82.

See Also

`node.depth`, `dist.topo`.

Examples

```
set.seed(15)
tr <- ape::rtree(15)
set.seed(22)
tr2 <- ape::rtree(15)
tree.comp(tr, tr2)
tree.comp(tr, tr2, method="PH85")
tree.comp(tr, tr2, method="shallow")
```

`woodmouse` *Cytochrome b Gene Sequences of Woodmice*

Description

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

Format

A DNAbin object containing 8 sequences with a length of 18 base pairs stored as a matrix.

Source

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

Index