Package ‘windex’

February 6, 2025

Type Package

Title Analysing Convergent Evolution using the Wheatsheaf Index

Version 2.1.0

Date 2025-02-05

Author Kevin Arbuckle [aut, cre],
Amanda Minter [aut]

Maintainer Kevin Arbuckle <kevin.arbuckle@swansea.ac.uk>

Description Analysing convergent evolution using the Wheatsheaf index, described in Arbuckle et al. (2014) <doi:10.1111/2041-210X.12195>, and some other unrelated but perhaps useful functions.

License GPL-2

Depends phytools, geiger (>= 2.0), ape (>= 4.0), phangorn,
scatterplot3d, utils, methods, R (>= 3.0.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2025-02-06 11:00:02 UTC

Contents

backLog *Backtransforms coefficient estimates from logistic regression*

Description

Takes raw coefficient values (e.g. from a logistic regression) and returns back-transformed estimates.

Usage

```
backLog(x)
```

Arguments

`x`	A numerical value to be back-transformed.

Value

Returns the back-transformed estimate.

Author(s)

Kevin Arbuckle

`error.bars` *Plots error bars on a barplot*

Description

Adds error bars to a barplot.

Usage

```
error.bars(x,y,upper,lower=upper,length=0.1,...)
```

Arguments

`x` Command to produce a barplot (saved as an object).

`y` Vector of heights of bars (same as vector given as height argument to barplot).

`upper` Vector of upper confidence interval to plot as error bars.

`lower` Vector of lower confidence interval to plot as error bars (defaults to symmetrical error bars but can be specified separately for lower and upper confidence intervals to enable asymmetrical error bars to be plotted).

`length` Width of floor and ceiling of error bars in inches (not the length of the confidence interval, just an aesthetic choice), passed to arrows().

`...` Arguments to be passed to arrows() to customise appearance of error bars.

Author(s)

Kevin Arbuckle

Examples

```
means<-c(4,5,7,11)
ci<-c(0.5,1.2,0.7,1)
x<-barplot(means,main="",ylim=c(0,15),ylab="Mean number of things",xlab="Colours",
names.arg=c("Blue","Red","Yellow","Black"),cex.lab=1.5,col=c("blue","red","yellow","black"))
error.bars(x,y=means,upper=ci,col="grey30")
```

`LRbarplot` *Plots a barplot from logistic regression parameter estimates*

Description

Takes a fitted logistic regression model (e.g. with glm or phyloglm functions) and plots a barplot for a categorical predictor of the probability of the response variable based on estimated coefficients and standard errors.

Usage

```
LRbarplot(mod,parIndex=2,yName="response",
xLevels=paste(c(1:length(c(1,parIndex)))),main="",...)
```

Arguments

`mod` Fitted logistic regression model.

`parIndex` Index giving the parameter(s) for the variable to be plotted, in order of the model output with the intercept being 1 (but not included in this argument). Default of 2 will work for a model with a binary predictor as the first (or only) explanatory variable in the model, as this is the second parameter listed. Note that for predictors with more than two levels this will be a vector of length k-1, where k is the number of levels, to specify each parameter estimated in addition to the intercept.

`yName` Name of response variable in the model to be used as part of y-axis label (doesn't have to be identical to the model input).

`xLevels` Vector of names of levels of the categorical variable to be used as x-axis labels. Levels must be given in the same order they appear in the model, but they don't have to be identical to the model input. These should be changed as the default is simply to use sequential numbers and so is fairly uninformative.

`main` Main title for plot (defaults to no title).

`...` Arguments to be passed to barplot() to customise appearance of plot.

Note

Because this is plotting directly from estimated coefficients, the probabilities will be conditional on the intercept (which gives the reference level as the first one plotted). For models with a single explanatory variable this will always be unproblematic, and should often be good for a small number of covariates, but as the model gets more complex this approach may become a poorer representation of the data. The function was originally intended for plotting models for which the raw data often do not clearly show effects (e.g. when raw data doesn't consider phylogenetic effects or covariates included in the model).

Author(s)

Kevin Arbuckle

Examples

```
bitten<-sample(c(0,1),100,replace=TRUE,prob=c(0.4,0.6))
sex<-sample(c("Male","Female"),100,replace=TRUE,prob=c(0.5,0.5))
hab<-sample(c("Forest","Savannah"),100,replace=TRUE,prob=c(0.3,0.7))
sp<-sample(c("Cobra","Viper","Mamba","Boomslang"),100,replace=TRUE,
prob=c(0.4,0.25,0.2,0.15))
testdat<-data.frame(bitten,sex,hab,sp)

# Single binary variable in model
testmod1<-glm(bitten~sex,data=testdat,family=binomial)
```

```
LRbarplot(testmod1,parIndex=2,yName="receiving bite",
xLevels=levels(as.factor(testdat$sex)),cex.lab=1.5,cex.axis=1.5,
cex.names=1.5,col=c("coral1","cornflowerblue"))
mtext("Sex",line=3,adj=0.5,font=2,cex=1.5, side=1)

# Single 4-level variable in model
testmod2<-glm(bitten~sp,data=testdat,family=binomial)
LRbarplot(testmod2,parIndex=2:4,yName="receiving bite",
xLevels=levels(as.factor(testdat$sp)),cex.lab=1.5,cex.axis=1.5,
cex.names=1.5,col=c("green","tan2","ivory4","sienna"))
mtext("Species",line=3,adj=0.5,font=2,cex=1.5, side=1)

# Plotting the second of two binary variables in the model
testmod3<-glm(bitten~sex+hab,data=testdat,family=binomial)
LRbarplot(testmod3,parIndex=3,yName="receiving bite",
xLevels=levels(as.factor(testdat$hab)),cex.lab=1.5,cex.axis=1.5,
cex.names=1.5,col=c("dark green","light yellow"))
mtext("Habitat",line=3,adj=0.5,font=2,cex=1.5, side=1)
```

lrTest *Calculates likelihood ratio test*

Description

Takes raw log-likelihood values plus degrees of freedom and performs a likelihood ratio test.

Usage

```
lrTest(small,big,df)
```

Arguments

small	Numerical value giving the smaller of the two log-likelihoods being compared.
big	Numerical value giving the larger of the two log-likelihoods being compared.
df	A numerical value giving the degrees of freedom for the test.

Value

LR	Likelihood ratio statistic
P	P-value from likelihood ratio test

Author(s)

Kevin Arbuckle

mark.dist *Summarising distributions of (exam/coursework/etc.) marks for (UK) university teaching*

Description

Provides a breakdown of marks including summary statistics, plotted histogram with test of Normality, and distribution of grades (for UK system, i.e. 1st, 2.1, 2.2, 3rd, fail). Note that grading system is assumed to be out of 100 and grade boundaries are assumed to be 40 (from fail to 3rd), 50 (from 3rd to 2.2), 60 (from 2.2 to 2.1), and 70 (from 2.1 to 1st).

Usage

```
mark.dist(marks,plot=TRUE,col="light grey",main=NULL,xlab="Marks",xlim=c(0,100),
showBounds=FALSE,y=20,digits=2,...)
```

Arguments

`marks`	Numerical vector containing the marks being summarised.
`plot`	A logical indicating whether to plot a histogram of mark distribution.
`col`	Fill colour for histogram, passed to hist().
`main`	Title for histogram, passed to hist().
`xlab`	Text for x-axis label of histogram, passed to hist().
`xlim`	Numerical vector of length 2 giving start and end points of x-axis of histogram, passed to hist().
`showBounds`	A logical which, if TRUE, adds dashed lines and text to the histogram illustrating grade boundaries.
`y`	If showBounds=TRUE, y gives the height to plot the text on the histogram (this will likely need tweaked for each case).
`digits`	Number of digits to round values in grade breakdown table, default gives proportions to a precision of 1 percent (0.01).
`...`	Additional arguments passed to hist() to customise the histogram.

Value

`Summary`	Summary statistics of the mark distribution
`NormalityTest`	Results of Shapiro-Wilk normality test
`GradeBreakdown`	Proportion of marks falling into each grade

Author(s)

Kevin Arbuckle

Examples

```
testscores<-rnorm(n=85,mean=60,sd=15)
mark.dist(testscores)
```

`modSel.fitMk` *Model selection table for categorical trait evolution models fit in the fitMk function in the package phytools, based on information theoretical measures.*

Description

Creates a model selection table based on either AIC or AICc for categorical trait evolution models fit using the fitMk() function in the phytools package.

Usage

```
modSel.fitMk(...,tree=NULL,type="AICc")
```

Arguments

`...`	A set of categorical trait evolution models fit with the fitMk function in the package phytools, which you want to compare.
`tree`	Either an object of class phylo used to fit the models being compared, or a numerical value giving the number of species in the tree used to fit the models being compared. Only necessary when type="AICc".
`type`	Type of information theoretical measure you want to use (AICc or AIC are allowed), defaults to AICc.

Value

Model selection table with rownames corresponding to input model names and columns for K (number of parameters), logLik (log-likelihood), AICc (or AIC), deltaAICc (or deltaAIC, the difference between each model and the best model), Weight (Akaike weights, aka model probabilities), and Evidence ratio (the amount of evidence for the best model relative to each model such that, for instance, 4 would mean that model has 4x less evidence supporting it than the best model).

Author(s)

Kevin Arbuckle

Examples

```
## Not run:
# Three models initially run in phytools using fitMk (see help file for that package to fit
# these models) and saved as objects named 'er', 'sym', and 'ard'.

# Model selection table using AICc
```

```
modSel.fitMk(er,sym,ard,tree=phy)

# Model selection table using AIC
modSel.fitMk(er,sym,ard,tree=phy,type="AIC")
## End(Not run)
```

`modSel.geiger` *Model selection table for phenotypic evolution models fit in the package geiger, based on information theoretical measures.*

Description

Creates a model selection table based on either AIC or AICc for phenotypic evolution models fit using the fitContinuous() and fitDiscrete() functions in the geiger package.

Usage

```
modSel.geiger(...,type="AICc")
```

Arguments

`...`	A set of phenotypic evolution models fitted with either fitContinuous() or fitDiscrete() in the package geiger, which you want to compare.
`type`	Type of information theoretical measure you want to use (AICc or AIC are allowed), defaults to AICc.

Value

Model selection table with rownames corresponding to input model names and columns for K (number of parameters), logLik (log-likelihood), AICc (or AIC), deltaAICc (or deltaAIC, the difference between each model and the best model), Weight (Akaike weights, aka model probabilities), and Evidence ratio (the amount of evidence for the best model relative to each model such that, for instance, 4 would mean that model has 4x less evidence supporting it than the best model).

Author(s)

Kevin Arbuckle

Examples

```
## Not run:
# Two models initially run in geiger using fitContinuous (see help file for that package to fit
# these models) and saved as objects named 'bm' and 'ou'.

# Model selection table using AICc
modSel.geiger(bm,ou)

# Model selection table using AIC
modSel.geiger(bm,ou,type="AIC")
## End(Not run)
```

`modSel.phylolm` *Model selection table for phylogenetic (logistic) regressions fit in the phylolm and phyloglm functions in the package phylolm, based on information theoretical measures.*

Description

Creates a model selection table based on either AIC or AICc for phylogenetic (logistic) regressions fit using the phylolm() or phyloglm() functions in the phylolm package.

Usage

```
modSel.phylolm(...,tree=NULL,type="AICc",method=c("phylolm","logistic"))
```

Arguments

`...`	A set of phylogenetic (logistic) regressions fit in the phylolm and phyloglm functions in the package phylolm, which you want to compare.
`tree`	Either an object of class phylo used to fit the models being compared, or a numerical value giving the number of species in the tree used to fit the models being compared. Only necessary when type="AICc".
`type`	Type of information theoretical measure you want to use (AICc or AIC are allowed), defaults to AICc.
`method`	Whether the models being compared are standard phylogenetic regression (fit with phylolm function) or phylogenetic logistic regression (fit with phyloglm function). Defaults to phylolm but issues a warning if you haven't specified the method.

Value

Model selection table with rownames corresponding to input model names and columns for K (number of parameters), logLik (log-likelihood), AICc (or AIC), deltaAICc (or deltaAIC, the difference between each model and the best model), Weight (Akaike weights, aka model probabilities), and Evidence ratio (the amount of evidence for the best model relative to each model such that, for instance, 4 would mean that model has 4x less evidence supporting it than the best model).

Author(s)

Kevin Arbuckle

Examples

```
## Not run:
# Three models initially run in phylolm using phylolm or phyloglm
# (see help file for that package to fit these models) and saved
# as objects named 'mod1', 'mod2', and 'mod3'.
# Model selection table for phylogenetic regressions using AICc
modSel.phylolm(mod1,mod2,mod3,tree=phy,method="phylolm")

# Model selection table for phylogenetic regressions using AIC
modSel.phylolm(mod1,mod2,mod3,type="AIC",tree=phy,method="phylolm")

# Model selection table for phylogenetic logistic regressions using AICc
modSel.phylolm(mod1,mod2,mod3,tree=phy,method="logistic")
## End(Not run)
```

`modSel.rayDISC` *Model selection table for evolutionary pathway models fit in the package corHMM, based on information theoretical measures.*

Description

Creates a model selection table based on either AIC or AICc for evolutionary pathway models fit using the rayDISC() function in the corHMM package.

Usage

```
modSel.rayDISC(...,type="AICc")
```

Arguments

`...` A set of evolutionary pathway models fitted with rayDISC() in the package corHMM, which you want to compare.

`type` Type of information theoretical measure you want to use (AICc or AIC are allowed), defaults to AICc.

Value

Model selection table with rownames corresponding to input model names and columns for K (number of parameters), logLik (log-likelihood), AICc (or AIC), deltaAICc (or deltaAIC, the difference between each model and the best model), Weight (Akaike weights, aka model probabilities), and Evidence ratio (the amount of evidence for the best model relative to each model such that, for instance, 4 would mean that model has 4x less evidence supporting it than the best model).

Author(s)

Kevin Arbuckle

Examples

```
## Not run:
# Two models initially run in corHMM using rayDISC (see help file for that package to fit
# these models) and saved as objects named 'rev' (for reversible) and 'non' (for non-reversible).

# Model selection table using AICc
modSel.rayDISC(non,rev)

# Model selection table using AIC
modSel.rayDISC(non,rev,type="AIC")
## End(Not run)
```

`modSelTab` *Model selection table based on information theoretical measures.*

Description

Creates a model selection table based on either AICc, AIC or BIC for a range of model types. The function is written for GLM style models (e.g. using glm, lm, aov, lmer, or glmer functions) but should work for any model to which the base R functions logLik(), model.frame() and AIC() or BIC() can be applied.

Usage

```
modSelTab(...,type="AICc")
```

Arguments

`...` A set of fitted models you want to compare.

`type` Type of information theoretical measure you want to use (AICc, AIC and BIC are allowed), defaults to AICc.

Value

Model selection table with rownames corresponding to input model names and columns for K (number of parameters), logLik (log-likelihood), AICc (or AIC or BIC), deltaAICc (or deltaAIC or deltaBIC, the difference between each model and the best model), Weight (Akaike weights, aka model probabilities), and Evidence ratio (the amount of evidence for the best model relative to each model such that, for instance, 4 would mean that model has 4x less evidence supporting it than the best model).

Author(s)

Kevin Arbuckle

Examples

```
# Simulating some variables
y<-rnorm(mean=100,sd=30,500)
x1<-0.5*y+10+rnorm(mean=20,sd=10,500)
x2<-3*y-45+rnorm(mean=40,sd=150,500)

# Fitting GLMs to those variables to give three models for comparison
m1<-glm(y~x1)
m2<-glm(y~x2)
m3<-glm(y~x1+x2)

# Model selection table using AICc
modSelTab(m1,m2,m3)

# Model selection table using AIC
modSelTab(m1,m2,m3,type="AIC")

# Model selection table using BIC
modSelTab(m1,m2,m3,type="BIC")
```

`nodeDist` *Plotting and/or retrieving distribution of divergence times of a pair of species from a set of phylogenetic trees.*

Description

Plots and/or retrieves the distribution of age estimates of the most recent common ancestor of a specified pair of species across trees (for instance a posterior distribution).

Usage

```
nodeDist(trees,sp1,sp2,relTime=F,fillcol="blue",xlabel="Age (mya)",main="",
return.ages=F,plot=T,add=F,...)
```

Arguments

`trees` An object of class multiPhylo containing a set of time-calibrated trees.

`sp1` Name of one of the two species for which the divergence time is of interest.

`sp2` Name of the other of the two species for which the divergence time is of interest.

`relTime` Logical whether to plot relative (to age of root) divergence times or absolute times (defaults to absolute times).

`fillcol` Colour to plot distribution.

`xlabel` Label for the x-axis.

`main` Title for plot (if desired, defaults to no title).

`return.ages` Logical whether to return a vector of the relative (to age of root) divergence times or absolute times (defaults to FALSE, i.e. only plotting the distribution).

`plot`	Logical whether to plot distribution of divergence times, either absolute or relative as controlled by relTime argument (defaults to TRUE).
`add`	Logical whether to overlay distribution on an existing plot (make sure xlim and ylim on original call are set to accomodate both distributions).
`...`	Additional arguments to customise output, passed to plot.

Value

Density plot of the distribution of relative or absolute divergence times across trees for the specified pair of species, and/or a vector of those divergence times.

Author(s)

Kevin Arbuckle

Examples

```
## Not run:
# Density plot of absolute divergence times
nodeDist(trees,"Naja_haje","Naja_nivea")

# Density plot of relative divergence times (root age for each tree set to 1)
nodeDist(trees,"Naja_haje","Naja_nivea",relTime=T,xlabel="Relative time")
## End(Not run)
```

`pir` *Calculates phylogenetic imbalance ratio (PIR)*

Description

Calculates the PIR to assess suitability of categorical traits for modelling approaches, following Gardner and Organ (2021).

Usage

```
pir(tree,trait1,trait2=NULL)
```

Arguments

`tree`	Phylogenetic tree of class 'phylo'.
`trait1`	Named vector containing states of a categorical trait. Must be a character or a factor and names must match tip labels of the tree.
`trait2`	An optional second trait when the intention is to test suitability of modelling a correlation between two categorical traits. Argument requirements are the same as trait1.

Value

CI	Consistency index
NIR	Normalised imbalance ratio (a measure of class imbalance across states or, if there are two traits, state combinations)
PIR	Phylogenetic imbalance ratio

Note

This function implements the phylogenetic imbalance ratio recommended in concert with its component parts (consistency index and normalised imbalance ratio) by Gardner and Organ (2021) to assess the suitability of categorical trait data for modelling in phylogenetic comparative methods. Each of these three indices ranges from 0 to 1. Low values of CI indicate high levels of homoplasy, which is linked to higher evolutionary sample sizes, whereas low values of NIR indicate a balanced distribution of traits (similar proportion of species in each state) and this often enables better and more data-driven parameter estimation from models. PIR is the product of CI and NIR, with lower values again preferred for phylogenetic comparative models. Gardner and Organ (2021) recommended a rule of thumb of PIR<0.1 as indicative that the categorical trait data are suitable for model-based analysis, but see that paper for more detailed discussion.

Author(s)

Kevin Arbuckle

References

Gardner, J.D. and Organ, C.L. 2021. Evolutionary sample size and consilience in phylogenetic comparative analysis. Systematic Biology 70:1061 - 1075.

Examples

```
data(sample.tree)

# Single trait (perhaps intended for estimating transition rates)
t1<-sample(c("brown","blue","green"),length(sample.tree$tip.label),replace=TRUE)
names(t1)<-sample.tree$tip.label
pir(sample.tree,trait1=t1)

# Two traits (perhaps intended for testing correlations)
t2<-sample(c("0","1"),length(sample.tree$tip.label),replace=TRUE)
names(t2)<-sample.tree$tip.label
pir(sample.tree,trait1=t1,trait2=t2)
```

plotTrait *Provides a visualisation of up to three-dimensional trait space (not accounting for phylogeny) for data exploration*

Description

Plots the trait space occupied by up to 3 traits with focals highlighted in red.

Usage

```
plotTrait(dat, traits, focal = dat[, 2], ...)
```

Arguments

`dat`	A dataframe containing a column of 0s and 1s to denote non-focal and focal taxa respectively, and columns of trait data which you which to plot.
`traits`	Column numbers (or names) for 1-3 traits which you want to plot.
`focal`	Column in the dataframe containing the focal designations.
`...`	Arguments to be passed to plot (or scatterplot3d for 3 traits) to customise output.

Author(s)

Kevin Arbuckle and Amanda Minter

Examples

```
data(sample.data)
plotTrait(sample.data,c(”ou1”,”ou2”),focal=sample.data[,2])
```

prune2data *Prunes a phylogenetic tree to match a vector (e.g. of species names in a dataset)*

Description

Takes a phylo object and vector of names to be matched to tip labels and returns a pruned phylogeny containing only tip labels that match those in the vector.

Usage

```
prune2data(tree, species)
```

Arguments

`tree`	Phylogenetic tree of class ’phylo’.
`species`	Vector of names to be matched against tip labels of the tree.

Value

Returns a phylogenetic tree of the class 'phylo' containing only tips whose labels match the input vector (species)

Author(s)

Kevin Arbuckle

Examples

```
data(sample.data)
data(sample.tree)
tree<-prune2data(sample.tree,sample.data$species[1:10])
plot(tree)
```

---

richYuleInputs *Generates the inputs necessary to run richness.yule.test() in ape*

Description

The function richness.yule.test() in the package ape requires two inputs - a dataframe with species richness of pairs of sister lineages which differ in the presence of a binary trait of interest, and a vector of divergence times of each of those sister group pairs. The richYuleInputs function generates these in a format which can be entered as the two required arguments.

Usage

```
richYuleInputs(tree, x, rich=NULL)
```

Arguments

`tree`	Phylogenetic tree of class 'phylo' with branch lengths in units of time.
`x`	Named vector representing the binary trait (labelled as 0 and 1 for absence and presence respectively).
`rich`	Optional named vector of species richness for each tip (for instance to account for incomplete sampling or when tips represent more than one species).

Value

`sisRich`	Dataframe containing two columns (species richness in sister lineages with and without the trait of interest) and rows representing different sister pairs
`divTimes`	Vector of divergence times of sister pairs (corresponding to rows of sisRich) differing in possession of a trait.

Author(s)

Kevin Arbuckle

See Also

richness.yule.test

Examples

```
data(sample.data)
data(sample.tree)
trait<-sample.data$focals
names(trait)<-sample.data$species

ryi<-richYuleInputs(sample.tree,trait)
richness.yule.test(ryi$sisRich,ryi$divTimes)
```

sample.data *Sample data for windex*

Description

Simulated data in a format suitable for use with the Wheatsheaf index functions in the windex package.

Usage

```
data(sample.data)
```

Format

A data frame with 100 observations on the following 9 variables.

`species` a factor

`focals` a numeric vector

`bm1` a numeric vector

`bm2` a numeric vector

`bm3` a numeric vector

`ou1` a numeric vector

`ou2` a numeric vector

`ou3` a numeric vector

`bin` a numeric vector

Details

The three 'bm' columns are values for three trait that have not evolved convergently. The three 'ou' columns are values for three traits that have evolved convergently with respect to the focal designation. bin is a column that was only for utility when creating the dataset.

Examples

```
dat<-data(sample.data)
summary(dat)
```

sample.tree *Phylogenetic tree to accompany sample.data*

Description

A simulated phylogeny from which the sample.data dataset was simulated, for use with the Wheatsheaf index functions in the windex package.

Usage

```
data(sample.tree)
```

Format

Phylogenetic tree of the class 'phylo' with 100 tips and (ultrametric) branch lengths.

Examples

```
data(sample.tree)
summary(sample.tree)
plot(sample.tree)
```

se *Calculates standard error*

Description

Calculates standard error of a numerical vector.

Usage

```
se(x)
```

Arguments

`x`	Numerical vector.

Value

Returns the standard error of the values in the vector.

Author(s)

Kevin Arbuckle

`test.windex` *Tests for particularly strong convergent evolution given topological constraints*

Description

The P-value returned is for the null hypothesis that the calculated Wheatsheaf index is no higher than expected by chance given the topology of the phylogenetic tree. Note that this is not a test for convergence per se, but of whether the convergence is unexpectedly strong.

Usage

```
test.windex(dat, tree, traits, focal = dat[, 2], SE = TRUE, reps,
plot = TRUE, fossil = FALSE, species.col="species", main = "",
line = 2.5, ...)
```

Arguments

`dat` A dataframe containing a column of 0s and 1s to denote non-focal and focal taxa respectively, and columns of trait data which you which to plot.

`tree` Phylogenetic tree of class 'phylo' containing branch lengths. The tree should also be ultrametric.

`traits` Column numbers (or names) of the traits for which you want to calculate a Wheatsheaf index.

`focal` Column in the dataframe containing the focal designations.

`SE` A logical specifying whether to standardise the traits by their standard error across species, default is `SE=TRUE`.

`reps` Number of bootstrap replicates on which to base the P-value.

`plot` A logical indicating whether to plot the bootstrap distribution. If TRUE, a histogram is plotted with the calculated Wheatsheaf index and its 95% confidence interval overlayed on the histogram as a solid and dashed lines (respectively).

`fossil` A logical specifying whether the tree contains fossil tips (i.e. is not ultrametric), in which case a different phylogenetic distance penalty based on shared branch length rather than shared time before divergence will be used instead. Note that this alternative penalty for trees containing fossils hasn't yet been rigorously tested so use cautiously, but it seems to behave as expected.

`species.col` Name of column in the dataset containing species names corresponding to tip labels in the tree, default assumes the column is named "species".

`main` Main title for plot (defaults to no title).

`line` Adjusts position of main title (if one is given), with lower values moving it down and higher values moving it up.

`...` Additional arguments passed to hist() to customise the histogram (when plot=TRUE).

Value

`w`	Calculated Wheatsheaf index
`low95`	Lower bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving
`up95`	Upper bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving
`P`	P-value from bootstrapping the tips of the phylogenetic tree
`boot.dist`	Bootstrap sample of Wheatsheaf index used to calculate P-value

Author(s)

Kevin Arbuckle and Amanda Minter

References

Arbuckle, K., Bennett, C.M. and Speed, M.P. 2014. A simple measure of the strength of convergent evolution. Methods in Ecology and Evolution 5:685 - 693.

See Also

`windex` `windex.sim.test`

Examples

```
## Not run:
data(sample.data)
data(sample.tree)
test.windex(sample.data,sample.tree,traits=c("bm1","bm2"),focal=sample.data[,2],
reps=1000,plot=TRUE,col="light grey")

## End(Not run)
```

`treecheck` *Checks a set of phylogenetic trees to confirm they are ultrametric and binary.*

Description

Takes a set of phylogenetic trees as a multiPhylo object (or a single tree as a phylo object) and reports which (if any) are not binary or ultrametric.

Usage

```
treecheck(trees)
```

Arguments

`trees`	Set of phylogenetic trees of class 'multiPhylo' or 'phylo' containing branch lengths.

Value

Either confirms that all trees are binary and ultrametric or prints warnings stating which trees do not meet those criteria.

Author(s)

Kevin Arbuckle

`treedatacheck` *Checks a set of phylogenetic trees to confirm they are ultrametric and binary and that tip labels match a list of names.*

Description

Takes a set of phylogenetic trees as a multiPhylo object (or a single tree as a phylo object) and a vector (e.g. of species names) and reports which trees (if any) are not binary, not ultrametric, or have tip labels that don't match the vector of names.

Usage

```
treedatacheck(trees,species)
```

Arguments

`trees`	Set of phylogenetic trees of class 'multiPhylo' or 'phylo' containing branch lengths.
`species`	Vector of names to be matched against tip labels of the tree(s).

Value

Either confirms that all trees are binary, ultrametric, and have tip labels matching the list of names, or prints warnings stating which trees do not meet those criteria (with troubleshooting options for mismatches between data and single trees).

Author(s)

Kevin Arbuckle

windex *Calculates Wheatsheaf index with 95% confidence intervals*

Description

Takes a phylo object and trait data and returns the Wheatsheaf index for the traits on the tree along with 95% confidence intervals obtained from jackkniving.

Usage

```
windex(dat, tree, traits, focal = dat[, 2], SE = TRUE, fossil = FALSE,
species.col=”species”)
```

Arguments

`dat` A dataframe containing a column of 0s and 1s to denote non-focal and focal taxa respectively, and columns of trait data which you which to plot.

`tree` Phylogenetic tree of class ’phylo’ containing branch lengths. The tree should also be ultrametric.

`traits` Column numbers (or names) for the traits for which you want to calculate a Wheatsheaf index.

`focal` Column in the dataframe containing the focal designations.

`SE` A logical specifying whether to standardise the traits by their standard error across species, default is `SE=TRUE`.

`fossil` A logical specifying whether the tree contains fossil tips (i.e. is not ultrametric), in which case a different phylogenetic distance penalty based on shared branch length rather than shared time before divergence will be used instead. Note that this alternative penalty for trees containing fossils hasn’t yet been rigorously tested so use cautiously, but it seems to behave as expected.

`species.col` Name of column in the dataset containing species names corresponding to tip labels in the tree, default assumes the column is named "species".

Value

`w` Calculated Wheatsheaf index

`low95` Lower bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving

`up95` Upper bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving

Author(s)

Kevin Arbuckle and Amanda Minter

References

Arbuckle, K., Bennett, C.M. and Speed, M.P. 2014. A simple measure of the strength of convergent evolution. Methods in Ecology and Evolution 5:685 - 693.

See Also

test.windex windex.sim.test

Examples

```
data(sample.data)
data(sample.tree)
windex(sample.data,sample.tree,traits=c("ou1","ou2"),focal=sample.data[,2],
SE=TRUE)
```

windex.sim.test *Tests for presence of convergent evolution (over and above that expected from Brownian motion)*

Description

The P-value returned is for the null hypothesis that the calculated Wheatsheaf index is no higher than expected for traits evolving under Brownian motion (parameterised with rates of evolution and trait covariances estimated from the original traits).

Usage

```
windex.sim.test(dat, tree, traits, focal = dat[, 2], SE = TRUE, Nsims,
plot = TRUE, fossil = FALSE, species.col="species", main = "",
line = 2.5, ...)
```

Arguments

dat	A dataframe containing a column of 0s and 1s to denote non-focal and focal taxa respectively, and columns of trait data which you which to plot.
tree	Phylogenetic tree of class 'phylo' containing branch lengths. The tree should also be ultrametric.
traits	Column numbers (or names) for the traits for which you want to calculate a Wheatsheaf index.
focal	Column in the dataframe containing the focal designations.
SE	A logical specifying whether to standardise the traits by their standard error across species, default is SE=TRUE.
Nsims	Number of simulations on which to base the P-value.

`plot` A logical indicating whether to plot the simulated distribution. If TRUE, a histogram is plotted with the calculated Wheatsheaf index and its 95% confidence interval overlayed on the histogram as a solid and dashed lines (respectively).

`fossil` A logical specifying whether the tree contains fossil tips (i.e. is not ultrametric), in which case a different phylogenetic distance penalty based on shared branch length rather than shared time before divergence will be used instead. Note that this alternative penalty for trees containing fossils hasn't yet been rigorously tested so use cautiously, but it seems to behave as expected.

`species.col` Name of column in the dataset containing species names corresponding to tip labels in the tree, default assumes the column is named "species".

`main` Main title for plot (defaults to no title).

`line` Adjusts position of main title (if one is given), with lower values moving it down and higher values moving it up.

`...` Additional arguments passed to hist() to customise the histogram (when plot=TRUE).

Value

`w` Calculated Wheatsheaf index

`low95` Lower bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving

`up95` Upper bound of 95% confidence interval for the Wheatsheaf index obtained by jackkniving

`P` P-value obtained from comparing observed Wheatsheaf index to simulations under Brownian motion on the phylogenetic tree

`sim.dist` Wheatsheaf indices of simulated datasets used to calculate P-value

Author(s)

Kevin Arbuckle

References

Arbuckle, K., Bennett, C.M. and Speed, M.P. 2014. A simple measure of the strength of convergent evolution. Methods in Ecology and Evolution 5:685 - 693.

See Also

`windex` `test.windex`

Examples

```
## Not run:
data(sample.data)
data(sample.tree)
windex.sim.test(sample.data,sample.tree,traits=c("bm1","bm2"),focal=sample.data[,2],Nsims=1000,
plot=TRUE,col="light grey")

## End(Not run)
```

Index