

Package ‘cati’

July 30, 2014

Type Package

Title Community Assembly by Traits: Individuals and beyond

Version 0.9

Date 2014-04-14

Author Adrien Taudiere, Cyrille Violle with contribution of Francois Munoz

Maintainer Adrien Taudiere <adrien.taudiere@cefe.cnrs.fr>

Depends R (>= 3.0.0), nlme, ade4, ape

Imports e1071, mice, rasterVis, hypervolume

Suggests lattice, vegan, FD, spacodiR, entropart, fBasics, picante

Description A package to detect and quantify community assembly processes using trait values of individuals or populations, the T-statistics and other metrics, and dedicated null models.

License GPL (>= 2)

URL <https://sourceforge.net/p/cati-r/code>

R topics documented:

cati-package	2
AbToInd	2
as.listofindex	3
ComIndex	5
ComIndexMulti	10
CVNND	13
decompWithin	14
finch.ind	15
partvar	16
plot.listofindex	17
plotCorTstats	22
plotDistri	23
plotRandtest	25
plotSESvar	26
plotSpPop	27

plotSpVar	29
RaoRel	30
ses	34
ses.listofindex	35
traitflex.anova	37
Tstats	39

Index	44
--------------	-----------

cati-package	<i>Community Assembly by Traits: Individuals and beyond</i>
--------------	---

Description

A package to detect and quantify community assembly processes using trait values of individuals or populations, the T-statistics and other metrics, and dedicated null models.

Details

Package: cati
 Type: Package
 Version: 0.9
 Date: 2014-04-14
 License: GPL (>= 2)
 Depends: R (>= 3.0.2), ape, nlme, e1071, moments, lattice, rasterVis, hypervolume, mice, ade4

This package provides functions to calculate T-statistics (Tstats function) and other uni-traits metrics (ComIndex function) to test community assembly traits measured on individuals and beyond (e.g. populations, functional groups). Variance partitioning (partvar function) and density plot (plot-Distri function) are also available. Finally, this package includes functions to summarize community assembly metrics and functions to plot standardized effect size of index.

Author(s)

Adrien Taudiere; adrien.taudiere@cefe.cnrs.fr
 Cyrille Violle

AbToInd	<i>Internal function. Transform abundance data matrix into individual like matrix.</i>
---------	--

Description

Transform abundance data matrix into individual like matrix to allows the use of ComIndex and ComIndexMulti on populationnal or specific traits values.

Usage

```
AbToInd(traits, com, type = "count")
```

Arguments

traits	Individual Matrix of traits with traits in columns. "traits" matrix must have row names (e.g. species or populationnal names).
com	Community data matrix with species in rows and sites in column.
type	Either "count" or "abundance". Use abundance when all values in the com matrix are not superior to one. Using abundance type is EXPERIMENTAL. This function round abundance to fit count data.

Details

Internal function

Value

A list of objects:

\$traits	Individual traits matrix
\$sp	Vector of species attributes
\$ind.plot	Vector of sites attributes

Author(s)

Adrien Taudiere

as.listofindex	<i>Transform index results in a list of index</i>
----------------	---

Description

Transform various results from functions Tstast, ComIndex or ComIndexMulti in a list of index. Useful to use the functions plot.listofindex (S3 method) and ses.listofindex.

Usage

```
as.listofindex(x, namesindex = NULL)
```

Arguments

x	A list of objects of class Tstast, ComIndex or ComIndexMulti
namesindex	Optionnal, the names of index in the same order as in x.

Value

A list of observed values and corresponding "null" values (i.e. produced by null models) in the form "list(index1, null model index1, index2, null model index2 ...)"

Author(s)

Adrien Taudiere

See Also

[ses.listofindex](#); [plot.listofindex](#)

Examples

```
data(finch.ind)
oldpar <- par()

####
#The function ComIndex allow to choose your own function
#(like mean, range, variance...) to calculate customize index.

require(e1071)

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

res.finch.sp_mn2 <- ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

res.finch.sp_mn3 <- ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

funct <- c("mean(x, na.rm=TRUE)", "kurtosis(x, na.rm=TRUE)",
"max(x, na.rm=TRUE) - min(x, na.rm=TRUE)", "CVNND(x)" )

res.finch.sp_mn2 <- ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
nullmodels = c("2","2","2","2"), ind.plot=ind.plot.finch, nperm=9, print=FALSE)

res.finch.sp_mn3 <- ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot=ind.plot.finch, nperm=9, print=FALSE)

####
#We can represent Standardized Effect Size (ses) using the
#function plot(as.listofindex(list1, list2, list3))

#The function ComIndex allow to choose your own function
#(like mean, range, variance...) to calculate customize index.

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

res.finch.sp_mn2 <- ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

res.finch.sp_mn3 <- ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

funct <- c("mean(x, na.rm=TRUE)", "kurtosis(x, na.rm=TRUE)",
"max(x, na.rm=TRUE) - min(x, na.rm=TRUE)", "CVNND(x)" )
```

```

funct <- c("mean(x, na.rm=TRUE)", "kurtosis(x, na.rm=TRUE)",
"max(x, na.rm=TRUE) - min(x, na.rm=TRUE)", "CVNND(x)" )

res.finch.sp_mn2 <- ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
nullmodels = c("2", "2", "2", "2"), ind.plot=ind.plot.finch, nperm=9, print=FALSE)

res.finch.sp_mn3 <- ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
nullmodels = c("2sp", "2sp", "2sp", "2sp"), ind.plot=ind.plot.finch, nperm=9, print=FALSE)

####
#We can represent Standardized Effect Size (ses)
#using the function plot(as.listofindex(list1, list2, list3))

list.ind2 <- list(res.finch.sp_mn2, res.finch.sp_mn3)
index.list2 <- as.listofindex(list.ind2)

plot(index.list2, type = "bytraits")

plot(index.list2)

```

ComIndex

Computing the moments of the trait distribution and other metrics to test and quantify the non-random assembly of communities

Description

Computing the moments of the trait distribution and other metrics to test and quantify the non-random assembly of communities.

Usage

```

ComIndex(traits = NULL, index = NULL, namesindex = NULL,
nullmodels = NULL, ind.plot = NULL, sp = NULL, com = NULL,
reg.pool = NULL, nperm = 99, printprogress = TRUE, ind.value = TRUE,
type = "count")

```

Arguments

traits	Individual Matrix of traits with traits in column.
index	A vector of function to apply to traits vectors in the form "mean(x, na.rm = TRUE)" or "range(x)", see examples for more complexe functions.
namesindex	A vector of names for index.
nullmodels	A vector of values corresponding to null models tu use for each index. A value of 1 corresponds to a randomization of individual values within a given community. A value of 2 corresponds to randomization of individual values within region, ie within all the dataset. A value of 2sp corresponds to randomization of population values (each individual value are replaced by the mean value of it population) within region. Finally a value of 2sp.prab mirror null model 2 but without taking indo account species abundance. For example, if nullmodels = c(1,2), the first

	index will be calculated on the null model 1 and the second index on the null model 2.
	If only one value is given, all the the null model will be determined by this value.
ind.plot	Factor defining the name of the plot (site or community) in which the individual is.
sp	Factor defining the species which the individual belong to.
com	Community data matrix with species (or populations) in rows and sites in column. Use only if ind.value = FALSE. "traits" matrix and "com" matrix must have the same number of rows.
reg.pool	Regional pool data for traits. If not informed, traits is considere as the regional pool. This matrix need to be larger (more rows) than the matrix "traits". Use only for null model 2.
nperm	Number of permutations. If NULL, only observed values are returned;
printprogress	Logical value; print progress during the calcul or not.
ind.value	Do the data are from individual value. If not, an internal function transform the abundance data matrix of com into a individual like matrix to compute the function.
type	Only if ind.value = FALSE. Either "count" or "abundance". Use abundance when one value or more in the com matrix are inferior to one. Using abundance type is EXPERIMENTAL. This function round abundance to fit count data.

Details

Compute statistics (e.g. mean, range, CVNND and kurtosis) to test community assembly using null models. For each statistic this function returns observed values and the related null distribution. This function implement three null models which keep unchanged the number of individual per community. Model 1 corresponds to randomization of individual values within community. Model 2 corresponds to randomization of individual values within region. Model 3 corresponds to randomization of population values within region.

In most cases, models 1 and 2 correspond to index at the individual level and the model 3 to index at the species level (or any other aggregate variable like genus, family or fonctionnal group).

Value

An object of class "ComIndex" corresponding to a list of lists:

\$obs	List of observed values for each trait in each community. Each component of the list corresponds to a matrix containing the result for each custom function.
\$Null	List of null values for each trait in each community. Each component of the list corresponds to an array containing the result of the permutations for each custom function.
\$list.index	List of index values and related null models. Internal use in other function. Traits in columns.
\$list.index.t	List of index values and related null models. Internal use in other function. Traits in rows.
\$sites_richness	Number of species per site.
\$namestraits	Names of traits.

Author(s)

Adrien Taudiere

See Also

[ComIndexMulti](#); [plot.listofindex](#); [ses](#)

Examples

```
data(finch.ind)
oldpar <- par()

####
#The function ComIndex allow to choose your own function
#(like mean, range, variance...) to calculate customize index.

require(e1071)

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

## Not run:
res.finch.sp_mn2 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.sp_mn3 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

####
#We can represent Standardized Effect Size (ses)
#using the function plot(as.listofindex(list1, list2, list3))

list.ind2 <- list(res.finch.sp_mn2, res.finch.sp_mn3)
index.list2 <- as.listofindex(list.ind2)

plot(index.list2, type = "bytraits")

plot(index.list2)

####
#This allows to calcul index per site
#for example using "tapply(x, sites, mean)".

funct <- c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) max(x, na.rm = TRUE) -
min(x, na.rm = TRUE) )", "tapply(x, ind.plot.finch, function(x)
CVNND(x))" )

##Null model 1 is trivial for this function
#because randomisation is within community only
```

```

res.finch.ind_mn1 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c(1,1,1,1), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.ind_mn2 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

####
#We can calcul metrics with or without intraspecific variance.
#Calculation of trait averages per population
#(name_sp_site is a name of a population)
#like in the function ComIndex
#and determine the site for each population (sites_bypop)

name_sp_sites = paste(sp.finch, ind.plot.finch, sep = "_")
traits.by.pop <- apply(traits.finch, 2 , function (x)
tapply(x, name_sp_sites, mean , na.rm = TRUE))

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep = "_"),
split = "_"), function(x) x[3])

funct.withoutIV <- c("tapply(x, unlist(sites_bypop), function(x)
mean(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop), function(x)
kurtosis(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop), function(x)
max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x)
mean(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)
kurtosis(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)
max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

res.finch.withIV <- ComIndex(traits = traits.finch, index = funct.withIV,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.withoutIV <- ComIndex(traits = traits.finch, index = funct.withoutIV,
sp = sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

####
#We can also represent T-statistics and custom index thanks to
#the plot.listofindex function.

res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch, sp = sp.finch,
nperm = 9, print = FALSE)

list.ind <- list(res.finch.withIV, res.finch.withoutIV ,res.finch)

index.list1 <- as.listofindex(list.ind, namesindex = c("mean", "kurtosis",
"range", "CVNND", "mean.pop", "kurtosis.pop", "range.pop", "CVNND.pop",

```



```

"T_IP.IC", "T_IC.IR", "T_PC.PR"))

class(index.list1)

par(mfrow = c(2,3))
plot(index.list1,type = "bytraits", bysite = TRUE)

par(mfrow = c(2,2))
plot(index.list1,type = "bytraits")
par(mfrow = c(1,1))

plot(index.list1,type = "simple")
plot(index.list1,type = "simple_range")
plot(index.list1,type = "normal")
plot(index.list1,type = "barplot")

## End(Not run)

#####
###Using ind.value = FALSE and community data matrix if there is no data
#available at the individual level.
## Not run:

#create traits data at the species level
traits_by_sp <- apply(traits.finch,2,function(x) tapply(x,sp.finch,
function(x) mean(x, na.rm = T)))

#create traits data at the populational level
names_sp_ind_plot <- as.factor(paste(sp.finch, ind.plot.finch, sep = "@"))
traits_by_pop <- apply(traits.finch,2,function(x) tapply(x,names_sp_ind_plot,
function(x) mean(x, na.rm = T) ))

#create community data matrix at the species or populational level
w1 <- table(sp.finch,ind.plot.finch)
dim(w1)
dim(traits_by_sp)

w2 <- table(names_sp_ind_plot,ind.plot.finch)
dim(w2)
dim(traits_by_pop)

#Choose indices
require(e1071)

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

#####
#with species value

res <- AbToInd(traits_by_sp, w1)

ComIndex(traits_by_sp, nullmodels = 2, ind.value = FALSE, index = funct,
sp = rownames(traits_by_sp), com = w1, nperm = 9)

```

```
#####
#with population value
res <- AbToInd(traits_by_pop, w2)
sp.sp <- unlist(strsplit(rownames(traits_by_pop), "@"))[seq(1,39*2,2)]

ComIndex(traits_by_pop, nullmodels = 2, ind.value = FALSE, index = funct,
sp = sp.sp, com = w2)

## End(Not run)

#####
####Simple example using null model 2sp.prab (species level without taking
# into account for species abundance, prab for presence/absence)

## Not run:
traits_by_sp <- apply(traits.finch, 2, function(x) tapply(x, name_sp_sites, mean, na.rm=T))
sites_bysp<-unlist(strsplit(rownames(traits_by_sp), split="_"))[seq(3,3*dim(traits_by_sp)[1], by=3) ]

funct.withoutIV.prab <- c("tapply(x, unlist(sites_bysp), function(x) mean(x, na.rm = TRUE))",
"tapply(x, unlist(sites_bysp), function(x) kurtosis(x, na.rm = TRUE))",
"tapply(x, unlist(sites_bysp), function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, unlist(sites_bysp), function(x) CVNND(x))")

res.finch.withoutIV.prab <- ComIndex(traits = traits.finch, index = funct.withoutIV.prab,
sp = sp.finch, nullmodels = rep("2sp.prab", times=4), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

list.ind2 <- list(res.finch.withoutIV, res.finch.withoutIV.prab)
index.list2 <- as.listofindex(list.ind2, namesindex =
c("mean.pop", "kurtosis.pop", "range.pop", "CVNND.pop",
"mean.prab", "kurtosis.prab", "range.prab", "CVNND.prab"))

plot(index.list2)

## End(Not run)
```

ComIndexMulti

Computing multitraits metrics to test and quantify the non-random assembly of communities

Description

Computing multitraits metrics to test and quantify the non-random assembly of communities

Usage

```
ComIndexMulti(traits = NULL, index = NULL, by.factor = NULL,
namesindex = NULL, nullmodels = NULL, ind.plot = NULL, sp = NULL,
com = NULL, reg.pool = NULL, nperm = 99, printprogress = TRUE,
ind.value = TRUE, type = "count")
```

Arguments

traits	Individual Matrix of traits with traits in column.
index	A vector of functions to apply to traits vectors in the form "mean(x, na.rm = TRUE)" or "range(x)".
by.factor	A factor to split the Matrix of traits and compute index for each subset eg for each site.
namesindex	A vector of names for metrics.
nullmodels	A vector of values corresponding to null models to use for each index. A value of 1 corresponds to a randomization of individual values within a given community. A value of 2 corresponds to randomization of individual values within region, ie within all the dataset. A value of 2sp corresponds to randomization of population values (each individual value are replaced by the mean value of its population) within region. Finally a value of 2sp.prab mirror null model 2 but without taking into account species abundance. For example, if nullmodels = c(1,2), the first index will be calculated on the null model 1 and the second index on the null model 2. If only one value is given, all the null model will be determined by this value.
ind.plot	Factor defining the name of the plot (site or community) in which the individual is.
sp	Factor defining the species which the individual belong to.
com	Community data matrix with species (or populations) in rows and sites in column. Use only if ind.value = FALSE. "traits" matrix and "com" matrix must have the same number of rows.
reg.pool	Regional pool data for traits. If not informed, traits is considered as the regional pool. This matrix needs to be larger (more rows) than the matrix "traits". Use only for null model 2.
nperm	Number of permutations. If NULL, only observed values are returned;
printprogress	Logical value; print progress during the calculation or not.
ind.value	Do the data are from individual value. If not, an internal function transforms the abundance data matrix of com into an individual-like matrix to compute the function.
type	Only if ind.value = FALSE. Either "count" or "abundance". Use abundance when all values in the com matrix are not superior to one.

Value

A list of lists:

\$obs	List of observed values for each trait in each community. Each component of the list corresponds to a matrix containing the result for each custom function.
\$Null	List of null values for each trait in each community. Each component of the list corresponds to an array containing the result of the permutations for each custom function.
\$list.index	List of index values and associated null models. Internal use in other function. Traits in columns.
\$list.index.t	List of index values and associated null models. Internal use in other function. Traits in rows.
\$sites_richness	Number of species per site.
\$namestraits	Names of traits.

Author(s)

Adrien Taudiere

See Also

[ComIndex](#); [plot.listofindex](#); [ses](#)

Examples

```
data(finch.ind)

####
#For most multivariate functions we need to replace (or exclude)
#NA values.

#For this example, we use the package mice to complete the data.

## Not run:
names.sp_ind_plot <- as.factor(paste(sp.finch, ind.plot.finch, sep = "_"))

comm <- t(table(ind.plot.finch,1:length(ind.plot.finch)))

library(mice)
traits = traits.finch
mice <- mice(traits.finch)
traits.finch.mice <- complete(mice)

####
#A simple example to illustrate the concept of the function
#ComIndexMulti

res.sum.1 <- ComIndexMulti(traits.finch,
index = c("sum(scale(x), na.rm = TRUE)", "sum(x, na.rm = TRUE)"),
by.factor = names.sp_ind_plot, nullmodels = c(2,2),
ind.plot = ind.plot.finch, nperm = 50, sp = sp.finch)

attributes(ses.listofindex(as.listofindex(res.sum.1)))

####
#A more interesting example using the function hypervolume
#from the package hypervolume.
#We show here several results which differe in there factor
#that delimit the group to calculate different hypervolume
#(argument by_factor).

require(hypervolume)

res.hv.1 <- ComIndexMulti(traits.finch.mice, index =
c("as.numeric (try(hypervolume(na.omit(x), warnings = FALSE, bandwidth=0.2, verbose=FALSE)@Volume))"),
by.factor = rep(1,length(names.sp_ind_plot)), nullmodels = c(2,2),
ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)

res.hv.2 <- ComIndexMulti(traits.finch.mice, index =
c("as.numeric(try(hypervolume(na.omit(x), warnings = FALSE, bandwidth=0.2, verbose=FALSE)@Volume))"),
by.factor = names.sp_ind_plot, nullmodels = c(2,2),
ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)
```

```

res.hv.3 <- ComIndexMulti(traits.finch.mice, index =
  c("as.numeric(try(hypervolume(na.omit(x), warnings = FALSE, bandwidth=0.2, verbose=FALSE)@Volume))"),
  by.factor = ind.plot.finch, nullmodels = c(2,2),
  ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)

res.hv.4 <- ComIndexMulti(traits.finch.mice, index =
  c("as.numeric(try(hypervolume(na.omit(x), warnings = FALSE, bandwidth=0.2, verbose=FALSE)@Volume))"),
  by.factor = sp.finch, nullmodels = c(2,2), ind.plot = ind.plot.finch,
  nperm = 9, sp = sp.finch)

list.ind.multi <- as.listofindex(list(res.hv.2, res.hv.3, res.hv.4))

ses.listofindex(list.ind.multi)

plot(list.ind.multi)
plot(list.ind.multi, xlim = c(-200,20))

## End(Not run)

```

CVNND	<i>Coefficient of variation of the nearest neighbourhood distance (CVNND)</i>
-------	---

Description

Coefficient of variation of the nearest neighbourhood distance (CVNND). Internal use for other function.

Usage

```
CVNND(trait, div_range = FALSE)
```

Arguments

trait	Trait vector.
div_range	Does CVNND need to be divide by the range.

Value

One value corresponding to the coefficient of variation of the nearest neighbourhood for the trait.

Author(s)

Adrien Taudiere

References

Jung, Vincent, Cyrille Violle, Cedric Mondy, Lucien Hoffmann, et Serge Muller. 2010. Intraspecific variability and trait-based community assembly: Intraspecific variability and community assembly. *Journal of Ecology* 98 (5): 1134-1140. doi:10.1111/j.1365-2745.2010.01687.x.

Examples

```
data(finch.ind)
CVNND(traits.finch[,1])
```

decompWithin	<i>Variance partitioning for multiple traits</i>
--------------	--

Description

This function decomposes the variation in community trait composition into three sources: (i) the intraspecific trait variability, (ii) the variability due to species turnover and (iii) their covariation is also separated. This decomposition is computed for the whole variation in the trait values and, The formula specified, across the contribution of various explanatory variables considered in the model. Barplot.decompWithin allow to plot the result of the decomposition.

Usage

```
decompWithin(traits = NULL, formula = ~1, ind.plot = NULL, sp = NULL,
printprogress = TRUE, ...)

## S3 method for class 'decompWithin'
barplot(height, resume = TRUE, ...)
```

Arguments

traits	Matrix of traits with traits in column
height	An object of class decompWithin obtain by the function decompWithin.
formula	The formula parameter must be a one-sided formula, i.e. starting with a tilde (~) character. The response variable is specified by the next two arguments, specif.avg and const.avg. By default set to ~1.
ind.plot	Factor defining the name of the plot (site or community) in which the individual is.
sp	Factor defining the species which the individual belong to.
printprogress	Logical value; print progress during the calcul or not.
resume	Logical. If resume = FALSE, plot one graphic by traits.
...	Optional additional arguments

Value

An object of class "decompWithin".

Author(s)

Adrien Taudiere Jan Leps

References

Leps, Jan, Francesco de Bello, Petr Smilauer and Jiri Dolezal. 2011. Community trait response to environment: disentangling species turnover vs intraspecific trait variability effects. *Ecography* 34 (5): 856-863.

See Also

[barplot.decompWithin](#); [traitflex.anova](#)

Examples

```
data(finch.ind)

res.decomp <- decompWithin(traits = traits.finch, sp = sp.finch,
  ind.plot = ind.plot.finch, print = FALSE)

barplot.decompWithin(res.decomp)

par(mfrow = c(2,2))
barplot.decompWithin(res.decomp, resume = FALSE)
par(mfrow = c(1,1))
```

finch.ind

Finch morphological data

Description

Individual morphological data for Darwin's finches. `finch` is the all data.frame. `ind.plot.finch` and `sp.finch` respectively correspond to the Island and the species attribute of each individual. `traits.finch` is the matrix of traits with four traits in rows and 2677 individuals in columns.

Usage

```
data(finch.ind)
```

Format

A data.frame of 2677 individuals in rows and 14 columns.

Details

See <http://bioquest.org/birdd/morph.php> for more information on database.

Source

<http://bioquest.org/birdd/morph.php>

Examples

```
data(finch.ind)
```

partvar	<i>Variance partitioning accross nested scales</i>
---------	--

Description

Variance partitioning accross nested scales using a decomposition (varcomp function) of variance on restricted maximum likelihood (REML) method (lme function). See Messier et al. 2010 for more information. barPartvar and piePartvar are associated plotting functions.

Usage

```
partvar(traits, factors, printprogress = TRUE)
barPartvar(partvar, col.bar = NA, ...)
piePartvar(partvar, col.pie = NA, ...)
```

Arguments

traits	Matrix of traits with traits in column
factors	A matrix of factors with the first column corresponds to the higher level factor, the second row the second higher level factor and so on.
printprogress	Logical value; print progress during the calcul or not.
partvar	The result of the partvar function.
col.bar	Vector of colors of bars
...	Any additional arguments are passed to the pie function.
col.pie	Vector of color for pie.

Value

An object of class "partvar" corresponding to a matrix of variance values with traits in rows and nested factors in column.

Author(s)

Adrien Taudiere Julie Messier

References

Messier, Julie, Brian J. McGill, et Martin J. Lechowicz. 2010. How do traits vary across ecological scales? A case for trait-based ecology: How do traits vary across ecological scales? Ecology Letters 13(7): 838-848. doi:10.1111/j.1461-0248.2010.01476.x.

See Also

[piePartvar](#); [barPartvar](#)

Examples

```

data(finch.ind)

genus <- as.vector(unlist(strsplit(as.vector(sp.finch), "_"))[seq(1, length(sp.finch)*2, by = 2)])

res.partvar.finch <- partvar(traits = traits.finch,
  factors = cbind(sites = as.factor(as.vector(ind.plot.finch)),
  species = as.factor(as.vector(sp.finch)), genus = as.factor(genus)))

res.partvar.finch <- partvar(traits=traits.finch,
  factors=cbind(sites=as.factor(as.vector(ind.plot.finch)),
  species=as.factor(as.vector(sp.finch)), genus=as.factor(genus)))

res.partvar.finch

oldpar <- par()

par(mfrow = c(2,2), mai = c(0.2,0.2,0.2,0.2))
piePartvar(res.partvar.finch, col = c("red", "green", "blue", "purple"))
par(oldpar)

barPartvar(res.partvar.finch, col = c("red", "green", "blue", "purple"))

par(mfrow=c(2,2), mai=c(0.2,0.2,0.2,0.2))
piePartvar(res.partvar.finch, col=c("red", "green", "blue", "purple"))
par(oldpar)

barPartvar(res.partvar.finch, col=c("red", "green", "blue", "purple"))

```

plot.listofindex

*Plot community assembly index***Description**

Plot community assembly index and confidence intervals using a list of index. S3 method for class listofindex.

Usage

```

## S3 method for class 'listofindex'
plot(x, type = "normal",
  col.index = c("red", "purple", "green"), add.conf = TRUE,
  color.cond = TRUE, val.quant = c(0.025, 0.975),
  grid.v = TRUE, grid.h = TRUE, xlim = NULL, ylim = NULL,
  cex.text = 0.8, plot.ask = FALSE, srt.text = 90,
  bysite = FALSE, ...)

```

Arguments

x A list of index and related null models obtained from to the as.listofindex function.

type	Type of plot. Possible type = "simple", "simple_range", "normal", "barplot" and "bytraits".
col.index	Vector of colors for index.
add.conf	Logical value; Add confidence intervals or not.
color.cond	Logical value; If color.cond = TRUE, color points indicate T-statistics values significantly different from the null model and grey points are not different from null model.
val.quant	Numeric vectors of length 2, giving the quantile to calculate confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.
grid.v	Logical value; print vertical grid or not
grid.h	Logical value; print horizontal grid or not
xlim	Numeric vectors of length 2, giving the x coordinates range
ylim	Numeric vectors of length 2, giving the y coordinates range
cex.text	Numeric value; the magnification to be used for text relative to the current setting of cex
plot.ask	Logical value; ask for plotting the next plot or not.
srt.text	Degree of rotation for text.
bysite	Logical value; plot per site or by traits.
...	Any additional arguments are passed to the plot function creating the core of the plot and can be used to adjust the look of resulting graph.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[as.listofindex](#); [plot.Tstats](#); [ses.listofindex](#)

Examples

```
data(finch.ind)
oldpar <- par()

####
#The function ComIndex allow to choose your own function
#(like mean, range, variance...) to calculate customize index.

require(e1071)

funct <- c("mean(x, na.rm = TRUE)", "kurtosis(x, na.rm = TRUE)",
"max(x, na.rm = TRUE) - min(x, na.rm = TRUE)", "CVNND(x)" )

## Not run:
res.finch.sp_mn2 <- ComIndex(traits = traits.finch, index = funct,
```

```

sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.sp_mn3 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

funct <- c("mean(x, na.rm=TRUE)", "kurtosis(x, na.rm=TRUE)",
"max(x, na.rm=TRUE) - min(x, na.rm=TRUE)", "CVNND(x)" )

## End(Not run)

## Not run:
res.finch.sp_mn2 <- ComIndex(traits=traits.finch, index=funct,
sp=sp.finch, nullmodels = c("2","2","2","2"), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

res.finch.sp_mn3 <- ComIndex(traits=traits.finch, index=funct,
sp=sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

####
#We can represent Standardized Effect Size (ses)
#using the function plot(as.listofindex(list1, list2, list3))

list.ind2 <- list(res.finch.sp_mn2, res.finch.sp_mn3)
index.list2 <- as.listofindex(list.ind2)

plot(index.list2, type = "bytraits")

plot(index.list2)

## End(Not run)

####
#This allows to calcul index per site
#for example using "tapply(x, sites, mean)".

funct <- c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) max(x, na.rm = TRUE) -
min(x, na.rm = TRUE) )", "tapply(x, ind.plot.finch, function(x) CVNND(x))")

funct <- c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm=TRUE))",
"tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm=TRUE))",
"tapply(x, ind.plot.finch, function(x) max(x, na.rm=TRUE) -
min(x, na.rm=TRUE) )", "tapply(x, ind.plot.finch, function(x) CVNND(x))")

##Null model 1 is trivial for this function
#because randomisation is within community only

## Not run:

```

```

res.finch.ind_mn1 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c(1,1,1,1), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.ind_mn2 <- ComIndex(traits = traits.finch, index = funct,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.ind_mn1 <- ComIndex(traits=traits.finch, index=funct,
sp=sp.finch, nullmodels = c(1,1,1,1), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

res.finch.ind_mn2 <- ComIndex(traits=traits.finch, index=funct,
sp=sp.finch, nullmodels = c("2","2","2","2"), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

####
#We can calcul metrics with or without intraspecific variance.
#Calculation of trait averages per population
#(name_sp_site is a name of a population)
#like in the function ComIndex
#and determine the site for each population (sites_bypop)

name_sp_sites = paste(sp.finch, ind.plot.finch, sep = "_")

traits.by.pop <- apply(traits.finch, 2 , function (x)

tapply(x, name_sp_sites, mean , na.rm = TRUE))

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep = "_"),
split = "_"), function(x) x[3])

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep = "_"),
split = "_"), function(x) x[3])

funct.withoutIV <- c("tapply(x, unlist(sites_bypop),
function(x) mean(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop),
function(x) kurtosis(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop),
function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withoutIV <- c("tapply(x, unlist(sites_bypop),
function(x) mean(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop),
function(x) kurtosis(x, na.rm = TRUE))", "tapply(x, unlist(sites_bypop),
function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x)
mean(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)
kurtosis(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)
max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x)
mean(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)
kurtosis(x, na.rm = TRUE))", "tapply(x, ind.plot.finch, function(x)

```

```

max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

res.finch.withIV <- ComIndex(traits = traits.finch, index = funct.withIV,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.withIV <- ComIndex(traits = traits.finch, index = funct.withIV,
sp = sp.finch, nullmodels = c("2","2","2","2"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

res.finch.withoutIV <- ComIndex(traits = traits.finch, index = funct.withoutIV,
sp = sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot = ind.plot.finch,
nperm = 9, print = FALSE)

tapply(x, name_sp_sites, mean , na.rm=TRUE))

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep="_"),
split="_"), function(x) x[3])

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep="_"),
split="_"), function(x) x[3])

funct.withoutIV <- c("tapply(x, unlist(sites_bypop),
function(x) mean(x, na.rm=TRUE))", "tapply(x, unlist(sites_bypop),
function(x) kurtosis(x, na.rm=TRUE))", "tapply(x, unlist(sites_bypop),
function(x) max(x, na.rm=TRUE) - min(x, na.rm=TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withoutIV <- c("tapply(x, unlist(sites_bypop),
function(x) mean(x, na.rm=TRUE))", "tapply(x, unlist(sites_bypop),
function(x) kurtosis(x, na.rm=TRUE))", "tapply(x, unlist(sites_bypop),
function(x) max(x, na.rm=TRUE) - min(x, na.rm=TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x)
mean(x, na.rm=TRUE))", "tapply(x, ind.plot.finch, function(x)
kurtosis(x, na.rm=TRUE))", "tapply(x, ind.plot.finch, function(x)
max(x, na.rm=TRUE) - min(x, na.rm=TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x)
mean(x, na.rm=TRUE))", "tapply(x, ind.plot.finch, function(x)
kurtosis(x, na.rm=TRUE))", "tapply(x, ind.plot.finch, function(x)
max(x, na.rm=TRUE) - min(x, na.rm=TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

res.finch.withIV <- ComIndex(traits=traits.finch, index=funct.withIV,
sp=sp.finch, nullmodels = c("2","2","2","2"), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

res.finch.withIV <- ComIndex(traits=traits.finch, index=funct.withIV,
sp=sp.finch, nullmodels = c("2","2","2","2"), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

res.finch.withoutIV <- ComIndex(traits=traits.finch, index=funct.withoutIV,
sp=sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"), ind.plot=ind.plot.finch,

```

```

nperm=9, print=FALSE)

## End(Not run)

####
#We can also represent T-statistics and custom index thanks to
#the plot.listofindex function.

## Not run:

res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch, sp = sp.finch,
nperm = 9, print = FALSE)

list.ind <- list(res.finch.withIV, res.finch.withoutIV ,res.finch)

index.list1 <- as.listofindex(list.ind, namesindex = c("mean", "kurtosis",

res.finch <- Tstats(traits.finch, ind_plot=ind.plot.finch, sp=sp.finch,
nperm=9, print=FALSE)

list.ind <- list(res.finch.withIV, res.finch.withoutIV ,res.finch)

index.list1 <- as.listofindex(list.ind, namesindex=c("mean", "kurtosis",

"range", "CVNND", "mean.pop", "kurtosis.pop", "range.pop", "CVNND.pop",
"T_IP.IC", "T_IC.IR", "T_PC.PR"))

class(index.list1)

par(mfrow = c(2,3))
plot(index.list1,type = "bytraits", bysite = TRUE)

par(mfrow = c(2,2))
plot(index.list1,type = "bytraits")
par(mfrow = c(1,1))

plot(index.list1,type = "simple")
plot(index.list1,type = "simple_range")
plot(index.list1,type = "normal")
plot(index.list1,type = "barplot")

## End(Not run)

```

plotCorTstats

Plot the bivariate relationships between T-statistics

Description

Plot the bivariate relationships between the three T-statistics namely T_IP.IC, T_IC.IR and T_PC.PR.

Usage

```
plotCorTstats(tstats = NULL, val.quant = c(0.025, 0.975),
  add.text = FALSE, bysite = FALSE, col.obj = NULL, plot.ask = TRUE,
  multipanel = TRUE, ...)
```

Arguments

tstats	The list resulting from the function Tstats.
val.quant	Numeric vector of length 2, giving the quantile to calculate confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.
add.text	Logical value; Add text or not.
bysite	Logical value; plot per site or by traits.
col.obj	Vector of colors for object (either traits or sites).
plot.ask	Logical value; Ask for new plot or not.
multipanel	Logical value. If TRUE divides the device to shown several traits graphics in the same device.
...	Any additional arguments are passed to the plot function creating the core of the plot and can be used to adjust the look of resulting graph.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[Tstats](#); [plot.Tstats](#); [plotSESvar](#)

Examples

```
data(finch.ind)
res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch,
  sp = sp.finch, nperm = 9)

plotCorTstats(res.finch, bysite = FALSE)
plotCorTstats(res.finch, bysite = TRUE)
```

plotDistri

Plot function to represent density of trait values

Description

Plot function to represent density of trait values

Usage

```
plotDistri(traits = NULL, var.1 = NULL, var.2 = NULL, col.dens = NULL,
plot.ask = TRUE, ylim.cex = 1, cex.leg = 0.8, polyg = TRUE,
multipanel = TRUE, leg = TRUE)
```

Arguments

traits	Matrix of traits with traits in column.
var.1	The first variable defines the division of each plots, in most case either a vector of species or name of sites.
var.2	The second variable define the division by color, in most case either a vector of species or name of sites.
col.dens	A vector of colors for the second variable.
plot.ask	Logical value; ask for plotting the next plot or not.
ylim.cex	Numeric value; the magnification to be used for range of y axe
cex.leg	Numeric value; the magnification to be used for legend relative to the current setting of cex
polyg	Logical value; do the mean distribution is full or empty
multipanel	Logical value. If TRUE divides the device to shown several traits graphics in the same device.
leg	Logical value; if TRUE print the legend.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[plotSpPop](#)

Examples

```
data(finch.ind)

#Plot the distribution of trait values for populations,
#species, sites and regional scales.

#First, let try the distribution for all populations
#of Darwin finches.

par(mfrow = c(4,4), cex = 0.5)

plotDistri(traits.finch, sp.finch, ind.plot.finch, ylim.cex = 3,
plot.ask = FALSE, multipanel = FALSE, leg = FALSE)

par(mfrow = c(1,1), cex = 1)
```



```

#Then we can inverse the second and the third arguments
#to plot the distribution for all finches species.

par(mfrow = c(4,4), cex = 0.5)

plotDistri(traits.finch, ind.plot.finch, sp.finch, ylim.cex = 8,
plot.ask = FALSE, multipanel = FALSE, leg = FALSE)

par(mfrow = c(1,1), cex = 1)

#You can also plot trait distribution for all species in the region

plotDistri(traits.finch, rep("region", times = dim(traits.finch)[1]),
sp.finch, ylim.cex = 6, plot.ask = FALSE, leg = FALSE)

#You can also plot trait distribution for all sites
#without taking into account species identity

plotDistri(traits.finch, rep("toutes_sp", times = dim(traits.finch)[1]),
ind.plot.finch, ylim.cex = 3, plot.ask = FALSE)

par(mfrow = c(4,4), cex = 0.5)

```

plotRandtest

Plot result of observed indices values against null distribution

Description

Function to plot result of observed indices values against null distribution.

Usage

```
plotRandtest(x, alternative = c("greater", "less", "two-sided"), ...)
```

Arguments

<code>x</code>	An object of class listofindex, ComIndex, ComIndexMulti or Tstats.
<code>alternative</code>	Indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less". You can specify just the initial letter. "greater" corresponds to positive association, "less" to negative association.
<code>...</code>	Any additional arguments are passed to the plot function creating the core of the plot and can be used to adjust the look of resulting graph.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[ComIndex](#); [ComIndexMulti](#); [Tstats](#); [as.listofindex](#); [plot.listofindex](#)

Examples

```
data(finch.ind)
res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch,
sp = sp.finch, nperm = 9, print = FALSE)

par(mfrow = c(3,4))

plotRandtest(res.finch)
plotRandtest(res.finch, alter = "two-sided")
```

plotSESvar

Plot SES values against a variable

Description

Plot standardized effect size values against a variable

Usage

```
plotSESvar(index.list, variable = NULL, ylab = "variable",
color.traits = NULL, val.quant = c(0.025, 0.975), resume = FALSE,
multipanel = TRUE)
```

Arguments

index.list	A list of index and the associate null models in the forme: list(index_1 = index_1_observed, index_1_nm = null.model.index_1, index_2 = index_2_observed, index_2_nm = null.model.index_2, ...).
variable	The variable against standardized effect sizes are plotted.
ylab	Label for the variable.
color.traits	A vector of colors corresponding to traits.
val.quant	Numeric vectors of length 2, giving the quantile to calcul confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.
resume	Logical value; resume = FALSE by default; Simplify the plot by plotting the mean and standard error for index value of multiple traits
multipanel	Logical value. If TRUE divides the device to shown several traits graphics in the same device.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[plot.listofindex](#); [ses](#)

Examples

```
data(finch.ind)
res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch, sp = sp.finch,
nperm = 9)

par(mfrow = c(2,2))
species.richness <- table(ind.plot.finch)
plotSESvar(as.listofindex(list(res.finch)), species.richness,
multipanel = FALSE)

#Same plot with resume = TRUE.

par(mfrow = c(2,2))
plotSESvar(as.listofindex(list(res.finch)), species.richness,
resume = TRUE, multipanel = FALSE)
par(mfrow = c(1,1))
```

plotSpPop

Plot populations values against species values

Description

Plot populations values against species values. The objectif is to see the contribution of intra-specific vs inter-specific variation to trait gradient.

Usage

```
plotSpPop(traits = NULL, ind.plot = NULL, sp = NULL,
col.ind = rgb(0.5, 0.5, 0.5, 0.5), col.pop = NULL, col.sp = NULL,
col.site = NULL, resume = FALSE, p.val = 0.05, min.ind.signif = 10,
multipanel = TRUE, col.nonsignif.lm = rgb(0, 0, 0, 0.5),
col.signif.lm = rgb(1, 0.1, 0.1, 0.8), silent = FALSE)
```

Arguments

traits	Individual Matrix of traits with traits in columns.
ind.plot	Factor defining the name of the plot in which the individual is.
sp	Factor defining the species which the individual belong to.
col.ind	Color for individual values.
col.pop	Color for populational mean values.
col.sp	Color for species mean values.
col.site	Color for sites mean values.
resume	Logical, if TRUE plot a simple form of the plot.
p.val	Chosen p.value to print significant linear relationship using linear model. Argument past to the lm funtion internally.

`min.ind.signif` Minimum individual to print significant linear relationship.
`multipanel` Logical value. If TRUE divides the device to shown several traits graphics in the same device.
`col.nonsignif.lm` Color for non significant linear relationship.
`col.signif.lm` Color for significant linear relationship.
`silent` Logical value, if `resume = FALSE` do not print warnings argument.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[plotDistri](#)

Examples

```

data(finch.ind)
plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE)

#If we change the value of the threshold
#(alpha = 10% instead of 5%
#and the minimum individual to represent significativity
#fixed to 3 instead of 10 by default)
#we can see some significant relationships.

plotSpPop(traits.finch, ind.plot.finch, sp.finch, p.val = 0.1,
min.ind.signif = 3, silent = TRUE)

#For a more simple figure, add the option resume = TRUE.
#Again if we change the value of the threshold
#(alpha = 10% instead of 5%
#and the minimum individual to represent significativity
# fixed to 3 instead of 10 by default)
#we can see some significant relationships.

plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE,
resume = TRUE, col.pop = "grey")

plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE,
resume = TRUE, col.pop = "grey", col.sp = "black")

plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE,
resume = TRUE, col.pop = "grey", col.sp = "black",
p.val = 0.1, min.ind.signif = 3)

```

plotSpVar

*Plot populations values against species values***Description**

Plot populations values against species values. The objectif is to see the contribution of intra-specific vs inter-specific variation to trait gradient.

Usage

```
plotSpVar(traits = NULL, ind.plot = NULL, sp = NULL, variable = NULL,
col.ind = rgb(0.5, 0.5, 0.5, 0.5), col.pop = NULL, col.sp = NULL,
col.site = NULL, resume = FALSE, p.val = 0.05, min.ind.signif = 10,
multipanel = TRUE, col.nonsignif.lm = rgb(0, 0, 0, 0.5),
col.signif.lm = rgb(1, 0.1, 0.1, 0.8), silent = FALSE)
```

Arguments

traits	Individual Matrix of traits with traits in columns.
ind.plot	Factor defining the name of the plot in which the individual is.
sp	Factor defining the species which the individual belong to.
variable	A matrix of variables corresponding to each site (in rows) and each trait (in columns). If you want to plot all traits against one variable, variable can be a vector of numerical values.
col.ind	Color for individual values.
col.pop	Color for populational mean values.
col.sp	Color for species mean values.
col.site	Color for sites mean values.
resume	Logical, if TRUE plot a simple form of the plot.
p.val	Choosen p.value to print significant linear relationship using linear model. Argument past to the lm funtion internally.
min.ind.signif	Minimum individual to print significant linear relationship.
multipanel	Logical value. If TRUE divides the device to shown several traits graphics in the same device.
col.nonsignif.lm	Color for non significant linear relationship.
col.signif.lm	Color for significant linear relationship.
silent	Logical value, if resume = FALSE do not print warnings argument.

Value

None; used for the side-effect of producing a plot.

Author(s)

Adrien Taudiere

See Also

[plotDistri](#)

Examples

```
data(finch.ind)

#Random variable for this example
variable <- c(1,5,15,6,3,25)

plotSpVar(traits.finch, ind.plot.finch, sp.finch, variable,
silent = TRUE)

#If we change the value of the threshold
#(alpha = 10% instead of 5%
#and the minimum individual to represent significativity
#fixed to 3 instead of 10 by default)
#we can see some significant relationships.

plotSpVar(traits.finch, ind.plot.finch, sp.finch, variable,
p.val = 0.1, min.ind.signif = 3, silent = TRUE)

#For a more simple figure, add the option resume = TRUE.
#Again if we change the value of the threshold
#(alpha = 10% instead of 5%
#and the minimum individual to represent significativity
# fixed to 3 instead of 10 by default)
#we can see some significant relationships.

plotSpVar(traits.finch, ind.plot.finch, sp.finch, variable,
silent = TRUE, resume = TRUE, col.pop = "grey")

plotSpVar(traits.finch, ind.plot.finch, sp.finch, variable,
silent = TRUE, resume = TRUE, col.pop = "grey", col.sp = "black")

plotSpVar(traits.finch, ind.plot.finch, sp.finch, variable,
silent = TRUE, resume = TRUE, col.pop = "grey", col.sp = "black",
p.val = 0.1, min.ind.signif = 3)
```

RaoRel

Alpha, gamma and beta-components for taxonomic, functional and phylogenetic diversity

Description

The Rao function computes alpha, gamma and beta-components for taxonomic, functional and phylogenetic diversity with the Rao index. The script integrates two functions: "Qdecomp", by Villeger et Mouillot (J Ecol, 2008) modified by Wilfried Thuiller, and "disc", by S. Pavoine, in the package ade4. For a regional assemblage of C local communities $\gamma = \text{mean}(\alpha) + \beta$, where: gamma is the diversity of the regional pool, alpha is the diversity of the local community and beta is the turn over between local communities diversity is estimated with the Rao quadratic entropy index (Rao 1982)

Usage

```
RaoRel(sample, dfunc, dphyl, weight = FALSE, Jost = FALSE,
        structure = NULL)
```

Arguments

sample	Community matrix of abundance (c x s) of the s species for the c local communities.
dfunc	matrix (s x s) or dist object with pairwise functional trait distances between the s species
dphyl	As dfunc but for phylogenetic distances
weight	Defining if the correction by Vileger & Mouillot (J Ecol, 2008) is applied or not
Jost	Defining if the Jost correction is applied (Jost 2007)
structure	A data frame containing the name of the group to which samples belong see de Bello et al, 2011 for more details.

Details

NA are not allowed in "locabrel <- abundances/ablocist". NA are automatically replaced by 0 in "sample". This function use the function "Qdecomp" by Sebastien Vileger & David Mouillot (J Ecol, 2008) modified by Wilfried Thuiller and the function disc originally proposed by Sandrine Pavoine.

Value

The results are organized for Taxonomic diversity (\$TD), Functional diversity (\$FD) and phylogenetical diversity (\$PD). Beta and gamma diversities are calculated for the whole data set and for each pair of samples ("Pairwise_samples"):

\$Richness_per_plot(number of species per sample)

\$Relative_abundance (species relative abundances per plot)

\$Pi (species regional relative abundance)

\$Wc (weighing factor),

\$Mean_Alpha (mean alpha diversity; for taxonomic diversity the Simpson index is calculated)

\$Alpha (alpha diversity for each sample; for taxonomic diversity the Simpson index is calculated)

\$Gamma (gamma diversity; for taxonomic diversity the Simpson index is calculated)

\$Beta_add (Gamma-Mean_Alpha)

\$Beta_prop (Beta_add*100/Gamma)

\$Pairwise_samples\$Alpha (mean alpha for each pair of samples)

\$Pairwise_samples\$Gamma (gamma for each pair of samples)

\$Pairwise_samples\$Beta_add (beta for each pair of samples as Gamma-Mean_Alpha)

\$Pairwise_samples\$Beta_prop (beta for each pair of samples as Beta_add*100/Gamma)

Author(s)

Francesco De Bello et al., 2011

References

De Bello, Francesco, Sandra Lavorel, Cecile H. Albert, Wilfried Thuiller, Karl Grigulis, Jiri Dolezal, stepan Janecek, et Jan Leps. 2011. Quantifying the relevance of intraspecific trait variability for functional diversity: Intraspecific variability in functional diversity. *Methods in Ecology and Evolution* 2: 163-174.

Examples

```
data(finch.ind)

## Not run:
comm <- t(table(ind.plot.finch,1:length(ind.plot.finch)))
comm.sp <- table(sp.finch, ind.plot.finch)
class(comm.sp) <- "matrix"

traits.finch.sp <- apply( apply(traits.finch, 2, scale ), 2,

function(x) tapply(x, sp.finch, mean, na.rm = TRUE))

mat.dist <- as.matrix(dist(traits.finch.sp))^2

res.rao <- RaoRel(sample = as.matrix(comm.sp), dfunc = mat.dist, dphyl = NULL,
weight = FALSE, Jost = FALSE, structure = NULL)

function(x) tapply(x, sp.finch, mean, na.rm=TRUE))

mat.dist <- as.matrix(dist(traits.finch.sp))^2

res.rao <- RaoRel(sample=as.matrix(comm.sp), dfunc=mat.dist, dphyl=NULL,
weight=FALSE, Jost=FALSE, structure=NULL)

witRao <- res.rao$FD$Mean_Alpha #overall within species variance
betRao <- res.rao$FD$Beta_add   #between species variance
totRao <- res.rao$FD$Gamma      #the total variance

witRao+betRao
totRao

#Now let's take the abundance to calculate Rao diversity.

res.rao.w <- RaoRel(sample = as.matrix(comm.sp), dfunc = mat.dist, dphyl = NULL,
weight = TRUE, Jost = FALSE, structure = NULL)

res.rao.w <- RaoRel(sample=as.matrix(comm.sp), dfunc=mat.dist, dphyl=NULL,
weight=TRUE, Jost=FALSE, structure=NULL)

witRao.w <- res.rao.w$FD$Mean_Alpha #overall within species variance
betRao.w <- res.rao.w$FD$Beta_add   #between species variance
totRao.w <- res.rao.w$FD$Gamma      #the total variance

witRao.w
betRao.w
```



```

#Plot the results

barplot(cbind(c(witRao.w, betRao.w), c(witRao, betRao)),
names.arg = c("abundance" ,"presence"),
legend.text = c("within species", "between species"),
ylab = "Rao", ylim = c(0,10))

#We can do this analysis for each trait separately.
#First we need to replace (or exclude) NA values.
#For this example, we use the package mice to complete the data.

comm <- t(table(ind.plot.finch,1:length(ind.plot.finch)))

library(mice)

traits = traits.finch

traits=traits.finch

mice <- mice(traits.finch)
traits.finch.mice <- complete(mice)

traits.finch.mice.sp <- apply(apply(traits.finch.mice, 2, scale ), 2,
function(x) tapply(x, sp.finch, mean, na.rm = TRUE))

function(x) tapply(x, sp.finch, mean, na.rm=TRUE))

trait.rao.w <- list()
witRao.w.bytrait <- c()
betRao.w.bytrait <- c()

for (t in 1 : 4){
  trait.rao.w[[t]] <- RaoRel(sample = as.matrix(comm.sp),
    dfunc = dist(traits.finch.mice.sp[,t]), dphyl = NULL, weight = TRUE,
    Jost = FALSE, structure = NULL)
}

for(t in 1 : 4){
  trait.rao.w[[t]] <- RaoRel(sample=as.matrix(comm.sp),
    dfunc=dist(traits.finch.mice.sp[,t]), dphyl=NULL, weight=TRUE,
    Jost=FALSE, structure=NULL)

  witRao.w.bytrait <- c(witRao.w.bytrait, trait.rao.w[[t]]$FD$Mean_Alpha)
  betRao.w.bytrait <- c(betRao.w.bytrait, trait.rao.w[[t]]$FD$Beta_add)
}

#Plot the results by traits.

barplot(t(cbind( witRao.w.bytrait, betRao.w.bytrait)),
names.arg = colnames(traits.finch),

```

```

legend.text = c("within species", "between species"),
ylab = "Rao", ylim = c(0,1.5))

## End(Not run)

```

ses

Standardized effect size and confidence interval for a matrix of statistics

Description

Calcul standardized effect size and confidence interval for a matrix of statistics and the related null model expressed as a list or as an array. Internal function use by other functions of the package. You can transpose the observed matrix to represent either the SES by traits or by plots. Warnings, to detect automatically the correspondence between dimension of observed matrix and null model list or array, observed matrix needs to have different numbers of rows and columns.

Usage

```
ses(obs = NULL, nullmodel = NULL, val.quant = c(0.025, 0.975))
```

Arguments

obs	Observed matrix or vector of values.
nullmodel	Either a list or an array of three (two for a vector of observed values) dimensions corresponding to the null model permutations.
val.quant	Numeric vectors of length 2, giving the quantile to calcul confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.

Value

A list of three components:

\$ses	Observed value of standardized effect size.
\$ses.inf	Lower limit of the confidence interval.
\$ses.sup	Upper limit of the confidence interval.

Author(s)

Adrien Taudiere

See Also

[plot.listofindex](#); [plotSESvar](#); [ses.listofindex](#)

Examples

```
data(finch.ind)

res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch,
sp = sp.finch, nperm = 9)

res.finch <- Tstats(traits.finch, ind_plot=ind.plot.finch,
sp=sp.finch, nperm=9)

ses(res.finch$T_IP.IC, res.finch$T_IP.IC_nm)

ses(t(res.finch$T_IP.IC), res.finch$T_IP.IC_nm)
```

ses.listofindex	<i>Standardized effect size for a list of index.</i>
-----------------	--

Description

Standardized effect size and confidence interval for a list of index.

Usage

```
ses.listofindex(index.list = NULL, val.quant = c(0.025, 0.975))
```

Arguments

index.list	A list of index obtain using the function as.listofindex.
val.quant	Numeric vectors of length 2, giving the quantile to calcul confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.

Value

A list which each component correspond to the result of the ses function for an index. Further, each component is a list of three components:

\$ses	Observed value of standardized effect size.
\$ses.inf	Lower limit of the confidence interval.
\$ses.sup	Upper limit of the confidence interval.

Author(s)

Adrien Taudiere

See Also

[as.listofindex](#); [ses](#)

Examples

```

data(finch.ind)

res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch, sp = sp.finch,
nperm = 9, print = FALSE)

#Calcul of means by population (name_sp_site is a name of a population)
#like in the function ComIndex and determine the site
#for each population (sites_bypop)

name_sp_sites = paste(sp.finch, ind.plot.finch, sep = "_")
traits.by.pop <- apply(traits.finch, 2 , function (x)
tapply(x, name_sp_sites, mean , na.rm = TRUE))

require(e1071)

sites_bypop <- lapply(strsplit(paste(rownames(traits.by.pop), sep = "_")
, split = "_"), function(x) x[3])

funct.withoutIV <- c("tapply(x, unlist(sites_bypop), function(x) mean(x, na.rm=TRUE))",
"tapply(x, unlist(sites_bypop), function(x) kurtosis(x, na.rm=TRUE))",
"tapply(x, unlist(sites_bypop), function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, unlist(sites_bypop), function(x) CVNND(x))" )

funct.withIV <- c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = TRUE))",
"tapply(x, ind.plot.finch, function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE) )",
"tapply(x, ind.plot.finch, function(x) CVNND(x))" )

res.finch.withIV <- ComIndex(traits = traits.finch, index = funct.withIV,
sp = sp.finch, nullmodels = rep("2", times=4), ind.plot = ind.plot.finch, nperm = 9
, print = FALSE)

res.finch.withoutIV <- ComIndex(traits = traits.finch,
index = funct.withoutIV, sp = sp.finch, nullmodels = rep("2sp", times=4),
ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

res.finch.withIV <- ComIndex(traits=traits.finch, index=funct.withIV,
sp=sp.finch, nullmodels = rep("2", times=4), ind.plot=ind.plot.finch,
nperm=9, print=FALSE)

res.finch.withoutIV <- ComIndex(traits=traits.finch,
index=funct.withoutIV, sp=sp.finch, nullmodels = c("2sp","2sp","2sp","2sp"),
ind.plot=ind.plot.finch, nperm=9, print=FALSE)

##Plot T-statistics and custom metrics thanks to
#the plot.listofindex function.

list.ind <- list(res.finch.withIV, res.finch.withoutIV, res.finch)
index.list <- as.listofindex(list.ind,
namesindex=c("mean", "kurtosis", "range", "CVNND",
"mean.pop", "kurtosis.pop", "range.pop", "CVNND.pop",

```

```

      "T_IP.IC", "T_IC.IR", "T_PC.PR"))

class(index.list)

plot(index.list, plot.ask = FALSE)

plot(index.list, plot.ask = FALSE, bysite = FALSE)

ses.list <- ses.listofindex(index.list)
ses.list
attributes(ses.list)

#### An other way to see "ses values"

# Custom theme (from rasterVis package)
require(rasterVis)

my.theme <- BuRdTheme()
# Customize the colorkey
my.ckey <- list(col = my.theme$regions$col)

levelplot(t(rbind(ses.list[[1]]$ses, ses.list[[2]]$ses,
ses.list[[3]]$ses, ses.list[[4]]$ses)), colorkey = my.ckey,
par.settings = my.theme, border = "black")

levelplot(t(rbind(ses.list[[1]]$ses>ses.list[[1]]$ses.sup,
ses.list[[2]]$ses>ses.list[[2]]$ses.sup,
ses.list[[3]]$ses>ses.list[[3]]$ses.sup,
ses.list[[4]]$ses>ses.list[[4]]$ses.sup)),
colorkey = my.ckey, par.settings = my.theme, border = "black")

#For all metrics of the list of index
ses.list.levelplot <- c()

for(i in 1: length(ses.list)){

ses.list.levelplot <- rbind(ses.list.levelplot, ses.list[[i]]$ses)
}

levelplot(t(ses.list.levelplot), colorkey = my.ckey,
par.settings = my.theme, border = "black")

```

traitflex.anova

Variance decomposition for a given trait

Description

This function decomposes variation of trait values within a community into three sources: (i) the intraspecific trait variability, (ii) the variability due to species turnover and (iii) their covariation is also separated. This decomposition is computed for the whole variation in the trait values and, The formula specified, across the contribution of various explanatory variables considered in the model. S3 method plot summarizes graphically the decomposition of trait variation, obtained with the traitflex.anova function. Print is an other S3 method for object of class traitflex.

Usage

```
traitflex.anova(formula, specif.avg, const.avg, ...)
## S3 method for class 'traitflex'
plot(x, plot.total = FALSE, use.percentage = TRUE,
     plot.covar = FALSE, cumul = FALSE,
     legend.pos = if (plot.total) "topleft" else "topright",
     plot.res = TRUE, ...)

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

Arguments

formula	The formula parameter must be a one-sided formula, i.e. starting with a tilde character. The response variable is specified by the next two arguments, specif.avg and const.avg.
specif.avg	Vector with community trait composition values for a single trait. It is calculated from trait values specific to each community (i.e. trait values for individual species are 'specific' to each plot, or habitat, where the species is found)
const.avg	Vector with community trait composition values for a single trait. It is calculated from average (fixed) trait values of individual species (i.e. fixed trait value for individual species used for all habitats where the species is found)
x	An object of class traitflex.
plot.total	Logical value; if TRUE plot not only the individual components of variation, but also the total variation. This is useful particularly when the decomposition was done with non-trivial formula (i.e. with explanatory variables)
use.percentage	Logical value; if TRUE the individual plotted sources of trait variation are shown as percentages of the total variation, on 0-100 scale.
plot.covar	Logical value; if TRUE the covariance between within-species trait variability and the variability due to species composition turnover is plotted as yet another category within the stacked bars. The plot.covar argument is entirely ignored when plotting traitflex object fitted with a formula without any predictor variables.
cumul	Logical value; if TRUE values are shown in a cumulative way.
legend.pos	This argument allows you to specify the position of graph legend. Thus argument is entirely ignored when plotting traitflex object created with a formula without predictors
plot.res	Logical value; if resume = FALSE plot is not shown but the table of values used to print the plot is return.
...	Optional additional arguments.

Details

The formula parameter must be a one-sided formula, i.e. starting with a tilde character. The response variable is specified by the next two arguments, specif.avg and const.avg.

Value

An object of class `traitflex`. There are `print` and `plot` methods available for it. The object contains decomposition of sum of squares into intraspecific variation component, compositional variation component, their covariation and total in a `SumSq` element. This is a data frame with multiple rows if predictors were specified in formula argument. The `RelSumSq` element contains the same table relativized to unit row totals. Finally, the `anova.turnover`, `anova.total`, and `anova.diff` elements contain the three aov objects used to decompose the variation.

Author(s)

Jan Leps et al., 2011

References

Leps, Jan, Francesco de Bello, Petr Smilauer and Jiri Dolezal. 2011. Community trait response to environment: disentangling species turnover vs intraspecific trait variability effects. *Ecography* 34 (5): 856-863.

See Also

`print.traitflex`; `plot.traitflex`

Tstats

Computing observed T-statistics (T for Traits) and null expectations.

Description

Computing observed T-statistics (T for Traits) as three ratios of variance, namely `T_IP.IC`, `T_IC.IR` and `T_PC.PR`. This function can also return the distribution of this three statistics under null models.

Usage

```
Tstats(traits, ind_plot, sp, reg.pool = NULL, nperm = NULL, printprogress = TRUE,
p.value = TRUE)
```

```
## S3 method for class 'Tstats'
barplot(height, val.quant = c(0.025, 0.975),
col.Tstats = c("red", "purple", "green", "white"), ylim = NULL, ...)
```

```
## S3 method for class 'Tstats'
plot(x, val.quant = c(0.025, 0.975),
col.Tstats = c("red", "purple", "green"), type = "normal",
add.conf = TRUE, ylim = NULL, xlim = NULL, ...)
```

```
summary_Tstats(x, val.quant = c(0.025, 0.975), type = "all")
```

Arguments

traits	Individual Matrix of traits with traits in columns. For one trait, use as.matrix().
ind_plot	Factor defining the name of the plot in which the individual is.
sp	Factor defining the species which the individual belong to.
reg.pool	Regional pool data for traits. If not informed, traits is considere as the regional pool. This matrix need to be larger (more rows) than the matrix "traits". Use only for null model 2.
nperm	Number of permutations. If NULL, only observed values are returned;
printprogress	Logical value; print progress during the calcul or not.
p.value	Logical value; calcul p-value or not. p-value are here defined as: $(x+1)/y$ with x the number of null values greater or lesser than the observed value and y the total number of null values.
x	An object of class Tstats.
height	An object of class Tstats.
val.quant	Numeric vectors of length 2, giving the quantile to calcul confidence interval. By default val.quant = c(0.025,0.975) for a bilateral test with alpha = 5%.
col.Tstats	A vector of three color correspond to the three T-statistics.
type	For the plot function, type of plot. Either "color_cond", "simple", "simple_sd", "normal" and "barplot". For the summary function, type of summary statistics. Either "binary", "percent", "p.value", "site" or "all".
add.conf	Logical value; Add confidence intervals or not.
ylim	limits for the y axis.
xlim	limits for the x axis.
...	Any additional arguments are passed to the plot function creating the core of the plot and can be used to adjust the look of resulting graph.

Details

S3 method plot: -Normal type plot means, standard deviations, ranges and confidence intervals of T-statistics. -Color_cond type plot T-statistics for each site and traits with color for significant values and grey for non signifivative ones. -Simple_sd type plot means, standard deviations and confidence intervals of T-statistics -Simple type plot T-statistics for each site and traits and the mean confidence intervals by traits -Barplot type plot means, standard deviations and confidence intervals of T-statistics in a barplot fashion

S3 method summary: -Binary type only test if a T-statistics is significantly different from the null expectation for each trait. -Percent type determine the percentage of sites were the T-statistics is significantly different from the null expectation for each trait. Asterix shows global significance of the test. -P-value type determine the p-value (two unilateral tests) of the T-statistics for each trait and sites. -Site type allows to know in which sites T-statistics deviate from the null expectation. -All type do all the precedent type of summary.

Value

A list of statistics:

\$T_IP.IC	Observed ratio between variance of individuals in populations and individuals in communities
-----------	--

\$T_IC.IR	Observed ratio between variance of individuals in communities and individuals in the region
\$T_PC.PR	Observed ratio between variance of populations in communities and populations in the region
\$T_PC.PR	Observed ratio between variance of populations in communities and individuals in the region
\$variances\$var_IP	variance of individuals within populations
\$variances\$var_PC	variance of populations within communities
\$variances\$var_CR	variance of communities within the region
\$variances\$var_IC	variance of individuals within communities
\$variances\$var_PR	variance of populations within the region
\$variances\$var_IR	variance of individuals within the region
\$variances\$var_IP_nm1	variance of individuals within populations in null model 1
\$variances\$var_PC_nm3	variance of populations within communities in null model 3
\$variances\$var_IC_nm1	variance of communities within the region in null model 1
\$variances\$var_IC_nm2	variance of individuals within communities in null model 2
\$variances\$var_PR_nm3	variance of populations within the region in null model 3
\$variances\$var_IR_nm2	variance of individuals within the region in null model 2
\$T_IP.IC_nm	If nperm is numeric; Result of simulation for T_IP.IC
\$T_IC.IR_nm	If nperm is numeric; Result of simulation for T_IC.IR
\$T_PC.PR_nm	If nperm is numeric; Result of simulation for T_PC.PR
\$pval\$T_IP.IC.inf	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_IP.IC observed < T_IP.IC simulated
\$pval\$T_IC.IR.inf	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_IC.IR observed < T_IC.IR simulated
\$pval\$T_PC.PR.inf	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_PC.PR observed < T_PC.PR simulated
\$pval\$T_IP.IC.sup	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_IP.IC observed > T_IP.IC simulated
\$pval\$T_IC.IR.sup	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_IC.IR observed > T_IC.IR simulated
\$pval\$T_PC.PR.sup	If p.value = TRUE; Result of p-value using the alternate hypothesis : T_PC.PR observed > T_PC.PR simulated

Author(s)

Adrien Taudiere Cyrille Violle

References

Violle, Cyrille, Brian J. Enquist, Brian J. McGill, Lin Jiang, Cecile H. Albert, Catherine Hulshof, Vincent Jung, et Julie Messier. 2012. The return of the variance: intraspecific variability in community ecology. *Trends in Ecology & Evolution* 27 (4): 244-252. doi:10.1016/j.tree.2011.11.014.

See Also

[ComIndex](#); [ComIndexMulti](#); [plotCorTstats](#); [plotSESvar](#); [plot.listofindex](#)

Examples

```
data(finch.ind)

res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch,
sp = sp.finch, nperm = 9, print = FALSE)

attributes(res.finch)

#Tstats class is associated to S3 methods plot, barplot and summary

plot(res.finch)

plot(res.finch, type = "color_cond")
plot(res.finch, type = "simple")
plot(res.finch, type = "simple_sd")
plot(res.finch, type = "barplot")

attributes(summary_Tstats(res.finch))
head(summary_Tstats(res.finch)$p.value, 10)

summary_Tstats(res.finch, type = "binary")
summary_Tstats(res.finch, type = "percent")
summary_Tstats(res.finch, type = "site")
summary_Tstats(res.finch, type = "p.value")
summary_Tstats(res.finch, type = "all")

barplot(res.finch)

attributes(summary_Tstats(res.finch))
head(summary_Tstats(res.finch)$p.value, 10)

#### An other way to see "ses values" of T-statistics

# Custom theme (from rasterVis package)
require(rasterVis)

my.theme <- BuRdTheme()
# Customize the colorkey
my.ckey <- list(col = my.theme$regions$col)

levelplot(t(ses(res.finch$T_IP.IC, res.finch$T_IP.IC_nm)$ses),
colorkey = my.ckey, par.settings = my.theme, border = "black")
```

```
#### Use a different regional pool than the binding of studied communities

#create a random regional pool for the example

reg.p <- rbind(traits.finch, traits.finch[sample(1:2000,300), ])

res.finch2 <- Tstats(traits.finch, ind_plot = ind.plot.finch,
  sp = sp.finch, nperm = 9, print = FALSE)

reg.p <- rbind(traits.finch, traits.finch[sample(1:2000,300),])

res.finch2 <- Tstats(traits.finch, ind_plot=ind.plot.finch,
  sp=sp.finch, nperm=9, print=FALSE)
```

Index

.Random.seed (finch.ind), 15

AbToInd, 2

as.listofindex, 3, 18, 26, 35

barPartvar, 16

barPartvar (partvar), 16

barplot.decompWithin, 15

barplot.decompWithin (decompWithin), 14

barplot.Tstats (Tstats), 39

cati (cati-package), 2

cati-package, 2

ComIndex, 5, 12, 26, 42

ComIndexMulti, 7, 10, 26, 42

CVNND, 13

decompWithin, 14

finch.ind, 15

ind.plot.finch (finch.ind), 15

partvar, 16

piePartvar, 16

piePartvar (partvar), 16

plot.listofindex, 4, 7, 12, 17, 26, 27, 34, 42

plot.traitflex, 39

plot.traitflex (traitflex.anova), 37

plot.Tstats, 18, 23

plot.Tstats (Tstats), 39

plotCorTstats, 22, 42

plotDistri, 23, 28, 30

plotRandtest, 25

plotSESvar, 23, 26, 34, 42

plotSpPop, 24, 27

plotSpVar, 29

print.traitflex, 39

print.traitflex (traitflex.anova), 37

RaoRel, 30

ses, 7, 12, 27, 34, 35

ses.listofindex, 4, 18, 34, 35

sp.finch (finch.ind), 15

summary_Tstats (Tstats), 39

traitflex.anova, 15, 37

traits.finch (finch.ind), 15

Tstats, 23, 26, 39