Package ‘funspace’

June 17, 2024

Type Package

Title Creating and Representing Functional Trait Spaces

Version 0.2.2

Description Estimation of functional spaces based on traits of organisms.
The package includes functions to impute missing trait values (with or without considering phylogenetic information), and to create, represent and analyse two dimensional functional spaces based on principal components analysis, other ordination methods, or raw traits. It also allows for mapping a third variable onto the functional space. See 'Carmona et al. (2021)' <doi:10.1038/s41586-021-03871-y>, 'Puglielli et al. (2021)' <doi:10.1111/nph.16952>, 'Carmona et al. (2021)' <doi:10.1126/sciadv.abf2675>, 'Carmona et al. (2019)' <doi:10.1002/ecy.2876> for more information.

License GPL-3

Depends R (>= 2.10)

Imports ade4, ape, ks, mgcv, missForest, MASS, paran, vegan, phytools, viridis

Suggests testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Carlos P. Carmona [cre] (<`https://orcid.org/0000-0001-6935-4913`>), Nicola Pavanetto [aut] (<`https://orcid.org/0000-0002-9441-863X`>), Giacomo Puglielli [aut] (<`https://orcid.org/0000-0003-0085-4535`>)

Maintainer Carlos P. Carmona <`perezcarmonacarlos@gmail.com`>

Repository CRAN

Date/Publication 2024-06-17 05:00:02 UTC

Contents

`funspace` *Functional space*

Description

Defines the functional structure of a set of species

Usage

```
funspace(
  x,
  PCs = c(1, 2),
  group.vec = NULL,
  fixed.bw = TRUE,
  n_divisions = 100,
  trait_ranges = NULL,
  threshold = 0.999
)
```

Arguments

`x`	Data to create the functional space. It can be either a PCA object obtained using the `princomp` function, a PCoA obtained using the `capscale` function from the `vegan` package, an NMDS generated with the `metaMDS` or `monoMDS` functions from `vegan`, a TPDs object generated with the `TPD` package or a matrix or data frame with at least two columns (representing two dimensions which can be either traits or ordination scores obtained with other methods).
`PCs`	A vector specifying the Principal Components to be considered (e.g. choosing `PCs = c(1,2)` would lead to to consider the first and the second principal components). Only applies if `x` contains a PCA. Defaults to `c(1, 2)`, which selects the first two principal components.

`group.vec`	An object of class factor specifying the levels of the grouping variable.
`fixed.bw`	Logical indicating whether the same bandwidth that is used in the kde estimation for the whole dataset should also be used for the kde estimation of individual groups of observations (`fixed.bw = T`), or if a different bandwidth has to be estimated for each group ((`fixed.bw = F`)). Defaults to TRUE, which makes the most extreme quantiles of the individual groups to coincide with those of the global distribution, and allows for more meaningful comparisons of the amount of functional space occupied by groups (functional richness).
`n_divisions`	The number of equal-length parts in which each principal component should be divided to calculate the grid in which calculations are based. Higher values of n_divisions will result in larger computation times, but also more smooth graphics. Defaults to 100.
`trait_ranges`	A list indicating the range of values that will be considered in the calculations for each of the considered PCA components. The list should contain the range (minimum and maximum) of values that will be considered. Each element of the list corresponds with one PCA component. The order of the components must be the same as the order provided in `PCs`. Defaults to NULL, in which case ranges are automatically calculated to ensure the functional space considered is sufficiently large to encompass the whole TPD function.
`threshold`	The probability threshold to consider to estimate the TPD function. TPD functions are positive across the whole trait space; `threshold` defines boundaries beyond which the TPD function is set to 0 (see Carmona et al. 2016; 2019 for more information). Defaults to 0.999.

Details

The functional structure of a set of organisms refers to how these organisms are distributed within a functional space (a space defined by traits). Functional structure can be expressed in probabilistic terms using trait probability density functions (TPD). TPD functions reflect how densely the organisms occupy the different parts of the functional space, and are implemented in the package `TPD` (Carmona et al. 2019).

`funspace` allows the user to define functional structure in a two-dimensional functional space created using a PCA, other ordination methods, or raw traits. The function automatically estimates the probability of occurrence of trait combinations within the space using kernel density estimation with unconstrained bandwidth using the functions from the `ks` R package (Duong, 2007). Contour lines can be drawn at any quantile of the probability distribution. Colored areas, corresponding to the target quantiles, visually summarize the probability of occurrence of certain trait combinations within the trait space.

Value

`funspace` The function returns an object of class `funspace` containing characteristics of the functional space and the trait probability distributions. The object includes estimations of functional richness and functional divergence for all observations taken together (global) and for each individual group (if groups are provided). The `funspace` class has specific methods exists for the generic functions `plot` and `summary`.

References

CP Carmona, F de Bello, NWH Mason, J Leps (2019). Trait Probability Density (TPD): measuring functional diversity across scales based on trait probability density with R. Ecology e02876. T Duong, T., (2007). ks: Kernel Density Estimation and Kernel Discriminant Analysis for Multivariate Data in R. J. Stat. Softw. 21(7), 1-16.

Examples

```
# 1. Plotting a space based on a PCA
x <- princomp(GSPFF)
funtest <- funspace(x = x, PCs = c(1, 2), threshold = 0.95)
summary(funtest)
plot(funtest, type = "global")

#2. To include groups, let's consider two major families.
# We will use two raw traits, ph and sla:
selFam <- c("Pinaceae", "Fabaceae")
selRows <- which(GSPFF_tax$family %in% selFam)
GSPFF_subset <- GSPFF[selRows, c("ph", "sla")]
tax_subset <- droplevels(GSPFF_tax[selRows, ])
funtest <- funspace(x = GSPFF_subset, threshold = 0.95, group.vec = tax_subset$family)
summary(funtest)
plot(funtest, type = "global")
plot(funtest, type = "groups", axis.title.x = "Plant height",
   axis.title.y = "Specific leaf area",
   quant.plot = TRUE, pnt = TRUE, pnt.cex = 0.5,
   pnt.col = rgb(0, 1, 1, alpha = 0.2))
```

`funspaceDim` *Dimensionality of a trait space*

Description

Calculating the dimensionality of a functional space based on PCA

Usage

```
funspaceDim(data)
```

Arguments

`data`	A `data.frame` or `matrix` containing trait data

Details

`funspaceDim` allows the user to identify the number of dimensions that are needed to build a trait space. The identified dimensions are those that minimize redundancy while maximizing the information contained in the trait data. The number of significant PCA axes to be retained is determined by using the `paran()` function of the R package `paran` (Dinno, 2018). `paran()` is based on the method proposed by Horn (1965), which involves contrasting the eigenvalues produced through PCAs run on (30 * (number of variables)) random datasets with the same number of variables and observations as the input dataset. Eigenvalues > 1 are retained in the adjustment.

Value

`funspaceDim` returns the number of dimensions to be retained. The output is stored and printed out in the R console as well.

References

Horn, J.L. (1965). A rationale and test for the number of factors in factor analysis. Psychometrika 30: 179-185.

Dinno, A. (2018). paran: Horn's test of principal components/factors. R package version 1.5.2.

Examples

```
# Dimensionality of the GSPFF
funspaceDim(GSPFF)
```

`funspaceGAM` *Functional space GAM*

Description

Mapping response variables in a functional space

Usage

```
funspaceGAM(y, funspace, family = "gaussian", minObs = 30)
```

Arguments

`y`	vector including the variable to be mapped inside the functional space. There must be a correspondence between the elements of `y` and the observations used to make the PCA (contained in 'pca.object'), both in the number of elements and in their order.
`funspace`	An object of class `funspace` providing the functional space to be considered. See function `funspace`
`family`	A family object specifying the distribution and link to use in the gam model. Defaults to "gaussian". See package `mgcv` for more details.

`minObs`	minimum number of observations needed in a group to make a model (defaults to 30).

Details

Different response variables can be mapped onto a functional space. In `funspace`, we follow the approach by Carmona et al. (2021), in which a generalized additive model is estimated across the bidimensional functional space. The resulting models show the predicted values of the response variable at each position of the portion of the functional space that is defined in the TPD of the global set of observations or of individual groups.

Value

The function returns an object of class `funspace` containing the functional space, trait probability distributions, and the fitted gam models. The `funspace` class has specific methods exists for the generic functions `plot` and `summary`.

References

CP Carmona, et al. (2021). Erosion of global functional diversity across the tree of life. Science Advances eabf2675

Examples

```
# 1. GAM on a space based on a PCA
x <- princomp(GSPFF)
funtest <- funspace(x = x, PCs = c(1, 2), threshold = 0.95)
y <- abs(x$scores[, 1] * x$scores[, 2]) + rnorm(nrow(GSPFF), mean = 0, sd = 1)
funtestGAM <- funspaceGAM(y = y, funspace = funtest)
plot(funtestGAM, quant.plot = TRUE, quant.col = "grey90")
summary(funtestGAM)
```

`funspaceNull` *Null models in functional space*

Description

Comparing the amount of occupied functional space against null models

Usage

```
funspaceNull(
  funspace,
  nrep = 100,
  alter = "greater",
  null.distribution = "multnorm",
  verbose = TRUE
)
```

Arguments

`funspace`	An object of class `funspace`
`nrep`	`numeric`The number of generated null surfaces
`alter`	`character`. The hypothesis to be tested when comparing the observed trait space against the null model. Options are 'greater', 'less', and 'two-sided'. See specification of the `as.randtest` function in the `ade4` R package
`null.distribution`	`character`. Data distribution for null model building. Available options are 'multnorm' and 'uniform' to generate data with a multivariate normal or uniform distribution, respectively.
`verbose`	`logical`. Do you want to information about the progress of the null model to be written to the console?

Details

`funspaceNull` The function tests for the statistical difference between the size (functional richness) of the considered TPD, obtained using the `funspace` function, against a vector of functional richness values generated using null models (see below) across a user-defined number of iterations. Two null models are currently available for testing. One generates data with a multivariate normal distribution, creating a dataset with normally distributed variables having the same mean and covariance than the observations used to build the functional space (see Carmona et al. 2021). This null model returns a theoretical TPD where some trait combinations (those around the mean of the trait space axes, thus towards the center of the null trait space) are more likely than others (i.e., this null model resembles an ellipse). The other null model generates a dataset with variables following a uniform distribution (see null model 1 in Diaz et al. 2016), creating a distribution where all trait combinations within the range of the original observations are equally possible (i.e., the approximate shape of this null model is a rectangle). Note that the function does not work for funspace objects that are based on a TPDs object created using the package `TPD`

Value

`funspaceNull` The function returns the list containing all the simulated datasets, the area of the observed trait space, the mean value of the area for the null model (calculated across iterations), the p-value of the difference between observed and simulated trait space, as well as a standardized effect size of the difference between observed trait space and mean null model areas. This output is reported together with the output of `funspace`.

References

CP Carmona, et al. (2021). Fine-root traits in the global spectrum of plant form and function. Nature 597, 683–687 S Diaz, et al. (2016). The global spectrum of plant form and function. Nature 529, 167–171

Examples

```
# 1. PCA space, multivariate model (see Carmona et al. 2021, Nature)
x <- princomp(GSPFF)
funtest <- funspace(x = x, PCs = c(1, 2), threshold = 0.95)
```

```
funtestNull <- funspaceNull(funtest, null.distribution = 'multnorm', nrep = 1000)
summary(funtestNull)

#'# 2. Two raw traits and uniform distribution (see Diaz et al. 2016, Nature)
x <- GSPFF[, c("ph", "sla")]
funtest <- funspace(x = x, threshold = 0.95)
funtestNull <- funspaceNull(funtest, null.distribution = 'uniform', nrep = 1000)
summary(funtestNull)
```

GSPFF *Aboveground traits from the global spectrum of plant form and function (complete data)*

Description

Data on six aboveground traits for 2,630 species with complete trait information. Data was processed from the TRY database (https://www.try-db.org/TryWeb/Home.php) and used in the paper "Fine-root traits in the global spectrum of plant form and function (Carmona et al. 2021, Nature). The data is available in https://doi.org/10.6084/m9.figshare.13140146. All traits are log10 transformed and scaled.

Usage

```
GSPFF
```

Format

## ‘GSPFF‘ A data frame with 2,630 rows and 6 columns:

la leaf area

ln leaf nitrogen content

ph plant height

sla specific leaf area

ssd specific stem density

sm seed mass ...

Source

<https://doi.org/10.6084/m9.figshare.13140146>

`GSPFF_missing` *Aboveground traits from the global spectrum of plant form and function (incomplete data)*

Description

Data on six aboveground traits for 10,746 species with incomplete trait information. Data was processed from the TRY database (https://www.try-db.org/TryWeb/Home.php) and used in the paper "Fine-root traits in the global spectrum of plant form and function (Carmona et al. 2021, Nature). The data is available in https://doi.org/10.6084/m9.figshare.13140146. Only species with information for at least three traits are included. All traits are log10 transformed and scaled.

Usage

```
GSPFF_missing
```

Format

## 'GSPFF_missing' A data frame with 10,746 rows and 6 columns:

la leaf area

ln leaf nitrogen content

ph plant height

sla specific leaf area

ssd specific stem density

sm seed mass ...

Source

<https://doi.org/10.6084/m9.figshare.13140146>

`GSPFF_missing_tax` *Taxonomic information for plants from the global spectrum of plant form and function (incomplete data)*

Description

Taxonomic data for 10,746 species with incomplete trait information (species with at least three traits).

Usage

```
GSPFF_missing_tax
```

Format

## ‘GSPFF_missing_tax‘ A data frame with 10,746 rows and 3 columns:

genus genus to which the species belongs

family family to which the species belongs

order order to which the species belongs ...

Source

<https://doi.org/10.6084/m9.figshare.13140146>

`GSPFF_tax` *Taxonomic information for plants from the global spectrum of plant form and function (complete data)*

Description

Taxonomic data for 2,630 species with complete trait information.

Usage

```
GSPFF_tax
```

Format

## ‘GSPFF_tax‘ A data frame with 2,630 rows and 3 columns:

genus genus to which the species belongs

family family to which the species belongs

order order to which the species belongs ...

Source

<https://doi.org/10.6084/m9.figshare.13140146>

impute *Imputing Trait Information*

Description

Imputing incomplete trait information, with the possibility of using phylogenetic information

Usage

```
impute(
  traits,
  phylo = NULL,
  addingSpecies = FALSE,
  nEigen = 10,
  messages = TRUE
)
```

Arguments

`traits`	A matrix or data.frame containing trait information with missing values. The rows correspond to observations (generally species) and the columns to the variables (generally traits). Traits can be continuous and/or categorical. Row names of the `traits` object must contain the names of the species. We recommend writing species name in the format "Genus_species" or "Genus species".
`phylo`	(optional) A phylogenetic tree (an object of class "phylo") containing the evolutionary relationships between species. `phylo` is used to estimate phylogenetic eigenvectors that are added to the `traits` matrix. Not all species in `traits` need to be necessarily included in `phylo`, despite this is highly recommended. Note that in order to assign phylogenetic information to species reliably, the names in `phylo$tip.label` must be exactly the same as `row.names(traits)`, although not necessarily in the same order. Note that computing cophenetic distances for very large trees (ca. 30,000 species) can result in memory allocation problems.
`addingSpecies`	Logical, defaults to FALSE. Should species present in the trait matrix but not in the phylogeny be added to the phylogeny? If TRUE, the `phytools::add.species.to.genus` function is used to add species to the root of the genus (in case there are any other congeneric species in the tree). Note that `phytools::add.species.to.genus` has other arguments that provide more flexibility, but those are not considered here for simplicity; users who want to make use of those options can instead modify their phylogenetic tree beforehand.
`nEigen`	The number of phylogenetic eigenvectors to be considered. Defaults to 10.
`messages`	Logical, defaults to TRUE. Should the function return messages?

Details

`impute` imputes trait values in trait matrices with incomplete trait information. It uses the Random Forest approach implemented in the `missForest` package. Phylogenetic information can be incorporated in the imputation in the form of a phylogenetic tree, from which a number of phylogenetic eigenvectors are added to the trait matrix.

Value

The function returns a list containing both the original trait data (incomplete) and the imputed trait data.

Examples

```
# GSPFF_missing dataset includes >10,000 species.
# Preparing and imputing this data takes very long time.
# Let's select a small random subset:
selectSPS <- 200
set.seed(2)
subset_traits <- GSPFF_missing[sample(1:nrow(GSPFF_missing), selectSPS), ]
deleteTips <- setdiff(phylo$tip.label, rownames(subset_traits))
subset_phylo <- ape::drop.tip(phylo, tip = deleteTips)
GSPFF_subset <- impute(traits = subset_traits, phylo = subset_phylo, addingSpecies = TRUE)
pca <- princomp(GSPFF_subset$imputed)
funtest <- funspace(pca)
plot(funtest, pnt = TRUE, pnt.cex = 0.2, arrows = TRUE)
summary(funtest)
```

`phylo` *Phylogeny for species from the global spectrum of plant form and function (incomplete data)*

Description

Phylogenetic tree including information for 10,746 species with incomplete trait information (species with at least three traits), contained in `GSPFF_missing`.

Usage

```
phylo
```

Format

## ‘phylo‘ An object of class `"phylo"`

plot.funspace *Functional space plotting*

Description

Takes a `funspace` object produced by `funspace()` or `funspaceGAM()` and plots the trait probability distribution (TPD) or the map of the response variable (depending of which kind of funspace object is provided) in a functional space.

Usage

```
## S3 method for class 'funspace'
plot(
  x = NULL,
  type = "global",
  which.group = NULL,
  quant.plot = FALSE,
  quant = NULL,
  quant.lty = 1,
  quant.col = "grey30",
  quant.lwd = 1,
  quant.labels = TRUE,
  colors = NULL,
  ncolors = 100,
  pnt = FALSE,
  pnt.pch = 19,
  pnt.cex = 0.5,
  pnt.col = "grey80",
  arrows = FALSE,
  arrows.length = 1,
  arrows.head = 0.08,
  arrows.col = "black",
  arrows.label.col = "black",
  arrows.label.pos = 1.1,
  arrows.label.cex = 1,
  axis.title = TRUE,
  axis.title.x = NULL,
  axis.title.y = NULL,
  axis.title.cex = 1,
  axis.title.line = 2,
  axis.cex = 1,
  globalContour = TRUE,
  globalContour.quant = NULL,
  globalContour.lwd = 3,
  globalContour.lty = 1,
  globalContour.col = "grey50",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

x	A funspace object produced by funspace() or funspaceGAM().
type	character indicating whether the plots should represent the global distribution of observations (type = "global"), or be separated by the groups (type = "groups") provided when the funspace object was created. Defaults to "global". In the case of funspace objects based on a TPD function created with the TPD package, only groups are plotted (there is no "global" distribution).
which.group	when plotting groups, either a character or a number indicating the name (character) or position (number) of a single group to be plotted individually.
quant.plot	Logical, Default is TRUE. Should contour lines representing quantiles (specified in quant) be plotted?
quant	A vector specifying the quantiles to be plotted (in case quant.plot is set to TRUE. In case a TPD function is plotted, the quantiles represent the quantiles of the trait probability density function (lower quantiles indicate areas with higher probability density). In case a GAM object is plotted, the quantiles of the fitted response variable are plotted. In case a TPD function is plotted, default quantiles are 0.99 (or the selected threshold if it is lower), 0.5 and 0.25. In the GAM alternative, default quantiles are 0.99, 0.5 and 0.25.
quant.lty	type of line to be used to represent quantiles. See lty argument in graphics::par().
quant.col	Color to be used in the quantile lines. Defaults to "grey30".
quant.lwd	Line width to be used in the quantile lines. Defaults to 1.
quant.labels	Logical, Default is TRUE. Should labels be added to quantile lines?
colors	A vector defining the colors of plotted quantiles in the TPD case. Only two colors need to be specified. The first color is automatically assigned to the highest quantile in quantiles (e.g. 0.99), the second color is assigned to the lowest quantile. These colors are then used to automatically generate a gradient from the greatest to the lowest quantile. Any color is admitted. Default is NULL, in which case c("yellow", "red") is used in case a trait probability density function is plotted and to viridis::viridis(5) in the GAM case.
ncolors	number of colors to include in the color gradients set by colors. Defaults to 100.
pnt	Logical, defaults to FALSE. Should data points be added to the functional space?
pnt.pch	Numerical. Graphical parameter to select the type of point to be drawn. Default is set to 19. See pch argument in graphics::par()..
pnt.cex	Numerical. Graphical parameter to set the size of the points. Default is 0.5. See cex argument in graphics::par().
pnt.col	Graphical parameter to set the points color. Default is "grey80".
arrows	Logical, defaults to FALSE. In case the functional space is based on a PCA, should the loadings of the original traits be represented by arrows in the functional space?

`arrows.length`	Numerical. Graphical parameter to set the length of the arrow (see `arrows`). Lower values lead to shorter arrows, which can help to make arrows fit within the represented functional space. Defaults to 1.
`arrows.head`	Numerical. Graphical parameter to set the length of the arrow head (see `arrows`). Defaults to 0.08.
`arrows.col`	Graphical parameter to set the arrows color (see `arrows`). Default is `"black"`.
`arrows.label.col`	Graphical parameter to set the color of the arrows labels color. Default is `"black"`.
`arrows.label.pos`	Numerical. Graphical parameter to set the position of the arrow labels with respect to the arrow heads. Default is 1.1, which draws arrow labels slightly beyond the arrow heads. A value of 1 means drawing labels on top of arrow heads.
`arrows.label.cex`	Numerical. Graphical parameter to set the size of arrow labels. Defaults to 1.
`axis.title`	Logical. Default is `TRUE`. Should axes titles be plotted?
`axis.title.x`	Character. The title to be plotted in the x axis if `axis.title` is set to `TRUE`. If not specified, a default axis title is plotted.
`axis.title.y`	Character. The title to be plotted in the y axis if `axis.title` is set to `TRUE`. If not specified, a default axis title is plotted.
`axis.title.cex`	Numerical. Graphical parameter to set the size of the axes titles. Default is 1.
`axis.title.line`	Numerical. Graphical parameter to set the on which margin line to plot axes titles. Default is 2.
`axis.cex`	Numerical. Graphical parameter to set the size of the axes annotation. Default is 1.
`globalContour`	Logical, Default is `TRUE`. Should a contour line representing the global distribution be plotted when `type` is set to `"groups"`. Adding a global contour lines provides a common reference for all groups and makes comparisons easier.
`globalContour.quant`	A vector specifying the quantiles to be plotted (in case `globalContour` is set to `TRUE`. Defaults to the threshold selected when the provided `funspace` object was originally created.
`globalContour.lwd`	Line width to be used in the global contour lines. Defaults to 3.
`globalContour.lty`	type of line to be used to represent the global contour lines. See `lty` argument in `graphics::par()`. Defaults to 1 (a continuous line).
`globalContour.col`	Graphical parameter to set the color of the global contour lines. Default is `"grey50"`.
`xlim`	the x limits (x1, x2) of the plot.
`ylim`	the y limits (y1, y2) of the plot.
`...`	Other arguments

Details

Produces default plots. If the input object was generated with `funspace()`, the plot shows a bivariate functional trait space displaying trait probability densities (for single or multiple groups). If the input object was generated with `funspaceGAM`, the plot shows a heatmap depicting how a target variable is distributed within the functional trait space (for single or multiple groups).

Value

No return value. This function is called for its side effect: generating plots.

Examples

```
x <- princomp(GSPFF)
funtest <- funspace(x = x, PCs = c(1, 2), threshold = 0.95)
plot(funtest, type = "global", quant.plot = TRUE, quant.lwd = 2, pnt = TRUE, pnt.cex = 0.1,
   pnt.col = rgb(0.1, 0.8, 0.2, alpha = 0.2), arrows = TRUE, arrows.length = 0.7)
```

`summary.funspace` *Summarizing Functional Spaces*

Description

`summary` method for class `funspace`"

Usage

```
## S3 method for class 'funspace'
summary(object, ...)
```

Arguments

`object`	A `funspace` object produced by `funspace()`, `funspaceGAM()`, or `funspaceNull()`.
`...`	Other arguments

Details

Produces default summary. If the input object was generated with `funspace()`, the summary includes information about the characteristics of the functional space (particularly if it derives from a PCA), along with functional diversity indicators (functional richness and functional divergence) for the whole set of observations and for each group (in case groups are specified). If the input object was generated with `funspaceGAM()`, the function returns the summary for the GAM models for the whole set of observations and individual groups. In the case of funspace objects based on a TPD object created with the `TPD` package, only information about groups is provided (since there is no global distribution). If the input was generated with `funspaceNull()`, the function returns tests exploring the difference between the observed functional richness and the null model functional richness.

Value

No return value. This function is called for its side effect: summarizing objects of class `"funspace"`.

Examples

```
x <- princomp(GSPFF)
funtest <- funspace(x = x, PCs = c(1, 2), threshold = 0.95)
summary(funtest)
```

Index