Package 'mFD'

July 22, 2025

Title Compute and Illustrate the Multiple Facets of Functional Diversity

Version 1.0.7

Description Computing functional traits-based distances between pairs of species for species gathered in assemblages allowing to build several functional spaces. The package allows to compute functional diversity indices assessing the distribution of species (and of their dominance) in a given functional space for each assemblage and the overlap between assemblages in a given functional space, see: Chao et al. (2018) <doi:10.1002/ecm.1343>, Maire et al. (2015) <doi:10.1111/geb.12299>, Mouillot et al. (2013) <doi:10.1016/j.tree.2012.10.004>, Mouillot et al. (2014) <doi:10.1073/pnas.1317625111>, Ricotta and Szeidl (2009) <doi:10.1016/j.tpb.2009.10.001>. Graphical outputs are included. Visit the 'mFD' website for more information, documentation and examples.

URL https://cmlmagneville.github.io/mFD/, https://github.com/CmlMagneville/mFD

BugReports https://github.com/CmlMagneville/mFD/issues

License GPL-2

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports ade4, ape, betapart (>= 1.5.4), cluster, dendextend, FactoMineR, gawdis, geometry, ggplot2, ggrepel, grid, Hmisc, patchwork, reshape2, rstatix, stats, utils, vegan

Suggests dplyr (>= 1.0.6), knitr, magrittr, rmarkdown (>= 2.6), stringr, tibble, tidyr

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Camille Magneville [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-0489-3822>),
Nicolas Loiseau [aut] (ORCID: <https://orcid.org/0000-0002-2469-1980>),
Camille Albouy [aut] (ORCID: <https://orcid.org/0000-0003-1629-2389>),
Nicolas Casajus [aut] (ORCID: <https://orcid.org/0000-0002-5537-5294>),
Thomas Claverie [aut] (ORCID: <https://orcid.org/0000-0002-6258-4991>),
Arthur Escalas [aut] (ORCID: <https://orcid.org/0000-0002-6450-5716>),
Fabien Leprieur [aut] (ORCID: <https://orcid.org/0000-0001-6869-342X>),
Eva Maire [aut] (ORCID: <https://orcid.org/0000-0002-1032-3394>),
David Mouillot [aut] (ORCID: <https://orcid.org/0000-0003-0402-2605>),
Sebastien Villeger [aut] (ORCID:
<https://orcid.org/0000-0002-2362-7178>)

Maintainer Camille Magneville <camille.magneville@gmail.com>

Repository CRAN

Date/Publication 2024-02-26 14:00:06 UTC

Contents

alpha.fd.fe . . . 3
alpha.fd.fe.plot . . . 5
alpha.fd.hill . . . 7
alpha.fd.multidim . . . 9
alpha.multidim.plot . . . 11
asb.sp.summary . . . 16
background.plot . . . 17
baskets_fruits_weights . . . 18
beta.fd.hill . . . 18
beta.fd.multidim . . . 20
beta.multidim.plot . . . 23
dist.nearneighb . . . 27
dist.point . . . 28
dist.to.df . . . 30
fdis.plot . . . 31
fdiv.plot . . . 34
feve.plot . . . 38
fide.plot . . . 41
fnnd.plot . . . 45
fori.plot . . . 48
fric.plot . . . 51
fruits_traits . . . 55
fruits_traits_cat . . . 56
fspe.plot . . . 57
funct.dist . . . 61
funct.space.plot . . . 63
fuse . . . 67
mst.computation . . . 69
panels.to.patchwork . . . 70

pool.plot 74
quality.fspaces 76
quality.fspaces.plot 79
sp.filter 81
sp.to.fe 83
sp.tr.summary 85
tr.cont.fspace 86
tr.cont.scale 88
traits.faxes.cor 89
vertices 91

Index **93**

`alpha.fd.fe` *Compute the set of indices based on number of species in Functional Entities*

Description

This function computes the set of indices based on number of species in Functional Entities (FEs) following Mouillot *et al.* (2014).

Usage

```
alpha.fd.fe(
  asb_sp_occ,
  sp_to_fe,
  ind_nm = c("fred", "fored", "fvuln"),
  check_input = TRUE,
  details_returned = TRUE
)
```

Arguments

`asb_sp_occ`	a matrix linking occurrences (coded as 0/1) of species (columns) in a set of assemblages (rows). Warning: **An assemblage must contain at least one species**.
`sp_to_fe`	a list with details of species clustering into FE from `sp.to.fe`.
`ind_nm`	a vector of character strings with the names of functional diversity indices to compute among 'fred', 'fored' and 'fvuln'. **Indices names must be written in lower case letters**. Default: all the indices are computed.
`check_input`	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: `check.input = TRUE`.
`details_returned`	a logical value indicating whether details about indices computation should be returned. These details are required by `alpha.fd.fe.plot` to plot FEs indices.

Value

A list with:

- *asb_fdfe* a matrix containing for each assemblage (rows), values of functional diversity indices (same names than in 'ind_nm') as well as the number of species ('nb_sp') and the number of FE (nb_fe);
- if *details_returned* is TRUE,
- *details_fdfe* a list with *asb_fe_nbsp* a matrix with number of species per FE in each assemblage.

Author(s)

Camille Magneville

References

Mouillot *et al.* (2014) Functional over-redundancy and high functional vulnerability in global fish faunas on tropical reefs. *PNAS*, **38**, 13757-13762.

Examples

```
# Load Species*Traits dataframe:
data('fruits_traits', package = 'mFD')

# Load Traits categories dataframe:
data('fruits_traits_cat', package = 'mFD')

# Load Assemblages*Species matrix:
data('baskets_fruits_weights', package = 'mFD')

# Remove continuous trait:
fruits_traits <- fruits_traits[, -5]
fruits_traits_cat <- fruits_traits_cat[-5, ]

# Compute gathering species into FEs:
sp_to_fe_fruits <- mFD::sp.to.fe(sp_tr = fruits_traits,
 tr_cat = fruits_traits_cat,
 fe_nm_type = 'fe_rank', check_input = TRUE)

# Get the occurrence dataframe:
asb_sp_fruits_summ <- mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)
asb_sp_fruits_occ <- asb_sp_fruits_summ$'asb_sp_occ'

# Compute alpha fd indices:
alpha.fd.fe(
   asb_sp_occ       = asb_sp_fruits_occ,
   sp_to_fe         = sp_to_fe_fruits,
   ind_nm           = c('fred', 'fored', 'fvuln'),
   check_input      = TRUE,
   details_returned = TRUE)
```

`alpha.fd.fe.plot` *Illustrate Functional Diversity indices based on Functional Entities*

Description

Graphical representation of distribution of species in Functional Entities (FE) and of indices from Mouillot *et al.* (2014). **To plot functional indices, functional indices values must have been computed first through the use of the** `alpha.fd.fe` function.

Usage

```
alpha.fd.fe.plot(
  alpha_fd_fe,
  plot_asb_nm,
  plot_ind_nm = c("fred", "fored", "fvuln"),
  name_file = NULL,
  color_fill_fored = "darkolivegreen2",
  color_line_fred = "darkolivegreen4",
  color_fill_bar = "grey80",
  color_fill_fvuln = "lightcoral",
  color_arrow_fvuln = "indianred4",
  size_line_fred = 1.5,
  size_arrow_fvuln = 1,
  check_input = TRUE
)
```

Arguments

`alpha_fd_fe`	output from the function `alpha.fd.fe` applied on assemblage of interest with `details_returned = TRUE`.
`plot_asb_nm`	a vector containing the name of the assemblage to plot.
`plot_ind_nm`	a vector containing the names of the indices to plot. It can be `'fred'` to plot functional redundancy (FRed), `'fored'` to plot functional over-redundancy (FOred) and/or `'fvuln'` to plot functional vulnerability (FVuln). Default is all 3 indices.
`name_file`	a character string with name of file to save the figure (without extension). Default: `name_file = NULL` which means plot is displayed.
`color_fill_fored`	a R color name or an hexadecimal code referring to the color used to fill the part of barplots that contain species in excess in species-rich FEs. It refers to the FORed value. Default: `color_fill_fored = "darkolivegreen2"`.
`color_line_fred`	a R color name or an hexadecimal code referring to the color used to draw the horizontal line referring to the FRed value. Default: `color_line_fred = "darkolivegreen4"`.
`color_fill_bar`	a R color name or an hexadecimal code referring to the color used to draw barplots. Default: `color_fill_bar = "grey80"`.

`color_fill_fvuln`
: a R color name or an hexadecimal code referring to the color used to fill barplot containing only one species for illustrating FVuln. Default: `color_fill_fvuln = "lightcoral"`.

`color_arrow_fvuln`
: a R color name or an hexadecimal code referring to the color used to draw the horizontal arrow showing the proportion of FEs containing only one species for illustrating FVuln. If there is only one FE containing one species, the arrow will be a point. Default: `color_arrow_fvuln = "indianred4"`.

`size_line_fred`
: a numeric value referring to the size of the horizontal line illustrating FRed. Default: `size_line_fred = 1.5`.

`size_arrow_fvuln`
: a numeric value referring to the size of the arrow showing the proportion of FEs containing only one species. Default: `size_arrow_fvuln = 1`.

`check_input`
: a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: `check_input = TRUE`.

Value

A `patchwork` object with a barplot of number of species per FE. Indices names provided in 'plot_ind_nm' are illustrated. Functional Redundancy (average number of species per FE) is illustrated with a horizontal line. Functional Over-redundancy (proportion of species in excess in FE richer than average) is illustrated with top part of these bars filled with 'color_fill_fored'. Functional Vulnerability (proportion of FE with a single species) is illustrated with bars of these vulnerable FE filled with 'color_fill_fvuln' and the double-head arrow highlighting their number. FE-based indices values on top of the plot. if `name_file` is provided, plot saved as a 300dpi png file in the working directory.

Author(s)

Camille Magneville and Sebastien Villeger

References

Mouillot *et al.* (2014) Functional over-redundancy and high functional vulnerability in global fish faunas on tropical reefs. *PNAS*, **111**, 13757-13762.

Examples

```
# Load Species*Traits dataframe
data("fruits_traits", package = "mFD")

# Load Traits categories dataframe
data("fruits_traits_cat", package = "mFD")

# Load Assemblages*Species matrix
data("baskets_fruits_weights", package = "mFD")

# Remove continuous trait
fruits_traits  <- fruits_traits[ , -5]
```

```
fruits_traits_cat <- fruits_traits_cat[-5, ]

# Compute gathering species into FEs
sp_to_fe_fruits <- mFD::sp.to.fe(
  sp_tr       = fruits_traits,
  tr_cat      = fruits_traits_cat,
  fe_nm_type  = "fe_rank",
  check_input = TRUE)

# Get the occurrence matrix
asb_sp_fruits_summ <- mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)
asb_sp_fruits_occ <- asb_sp_fruits_summ$"asb_sp_occ"

# Compute alpha fd indices
alpha_fd_fe_fruits <- mFD::alpha.fd.fe(
  asb_sp_occ       = asb_sp_fruits_occ,
  sp_to_fe         = sp_to_fe_fruits,
  ind_nm           = c("fred", "fored", "fvuln"),
  check_input      = TRUE,
  details_returned = TRUE)

# Plot fd fe indices
mFD::alpha.fd.fe.plot(
  alpha_fd_fe       = alpha_fd_fe_fruits,
  plot_asb_nm       = c("basket_1"),
  plot_ind_nm       = c("fred", "fored", "fvuln"),
  name_file         = NULL,
  color_fill_fored  = "darkolivegreen2",
  color_line_fred   = "darkolivegreen4",
  color_fill_bar    = "grey80",
  color_fill_fvuln  = "lightcoral",
  color_arrow_fvuln = "indianred4",
  size_line_fred    = 1.5,
  size_arrow_fvuln  = 1,
  check_input       = TRUE)
```

`alpha.fd.hill` *Compute Functional alpha-Diversity indices based on Hill Numbers*

Description

Compute functional alpha diversity applied to distance between species following the framework from Chao *et al.*(2019).

Usage

```
alpha.fd.hill(
  asb_sp_w,
  sp_dist,
  q = c(0, 1, 2),
```

```
  tau = "mean",
  check_input = TRUE,
  details_returned = TRUE
)
```

Arguments

asb_sp_w	a matrix with weight of species (columns) in a set of assemblages (rows). Rows and columns should have names. NA are not allowed.
sp_dist	a matrix or dist object with distance between species. Species names should be provided and match those in 'asb_sp_w'. NA are not allowed.
q	a vector containing values referring to the order of diversity to consider, could be 0, 1 and/or 2.
tau	a character string with name of function to apply to distance matrix (i.e. among all pairs of species) to get the threshold used to define 'functionally indistinct set of species'. Could be 'mean' (default), 'min' or 'max'. If tau = 'min' and there are null values in sp_dist, the threshold is the lowest strictly positive value and a warning message is displayed.
check_input	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check.input = TRUE.
details_returned	a logical value indicating whether the user want to store values used for computing indices (see list below)

Value

A list with:

- *asb_FD_Hill* a matrix containing indices values for each level of q (columns, named as 'FD_qx') for each assemblage (rows, named as in **asb_sp_w**)
- *tau_dist* the threshold value applied to distance between species to compute diversity according to function provided in **tau**
- if **details_returned** turned to TRUE a list *details* with
  - *asb_totw* a vector with total weight of each assemblage
  - *asb_sp_relw* a matrix with relative weight of species in assemblages

Note

FD is computed applying the special case where function 'f' in equation 3c is linear:f(dij(tau)) = dij(tau)/tau, hence f(0) = 0 and f(tau) = 1. FD computed with q=2 and tau = 'max' is equivalent to the Rao's quadratic entropy from Ricotta & Szeidl (2009, J Theor Biol). FD computed with tau = 'min' is equivalent to Hill number taxonomic diversity, thus with q=0 it is species richness (S), with q = 1 it is exponential of Shannon entropy (H) and with q = 2 it is 1/(1-D) where D is Simpson diversity. Note that even when q=0, weights of species are accounted for in FD. Hence to compute FD based only on distance between species present in an assemblage (i.e. a richness-like index) , asb_sp_w has to contain only species presence/absence coded as 0/1 with q=0 and tau = 'mean'. If asb_sp_w contains only 0/1 and q>0, it means that all species have the same contribution to FD.

Author(s)

Sebastien Villeger and Camille Magneville

References

Chao *et al.* (2019) An attribute diversity approach to functional diversity, functional beta diversity, and related (dis)similarity measures. *Ecological Monographs*, **89**, e01343.

Examples

```
# Load Species*Traits dataframe:
data('fruits_traits', package = 'mFD')

# Load Assemblages*Species dataframe:
data('baskets_fruits_weights', package = 'mFD')

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute alpha fd hill indices:
alpha.fd.hill(
   asb_sp_w         = baskets_fruits_weights,
   sp_dist          = sp_dist_fruits,
   q                = c(0, 1, 2),
   tau              = 'mean',
   check_input      = TRUE,
   details_returned = TRUE)
```

---

alpha.fd.multidim *Compute a set of alpha functional indices for a set of assemblages*

Description

This function computes a set of multidimensional space based indices of alpha functional diversity. The user can choose which functional indices to compute.

Usage

```
alpha.fd.multidim(
  sp_faxes_coord,
  asb_sp_w,
 ind_vect = c("fide", "fdis", "fmpd", "fnnd", "feve", "fric", "fdiv", "fori", "fspe"),
  scaling = TRUE,
```

```
  check_input = TRUE,
  details_returned = TRUE,
  verbose = TRUE
)
```

Arguments

sp_faxes_coord	a matrix of species coordinates in a chosen functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
asb_sp_w	a matrix linking weight of species (columns) and a set of assemblages (rows).
ind_vect	a vector of character string of the name of functional indices to compute. **Indices names must be written in lower case letters**. Possible indices to compute are: 'fide', fdis', 'fmpd', 'fnnd', 'feve', 'fric', 'fdiv', 'fori' and 'fspe'. Default: all the indices are computed.
scaling	a logical value indicating if scaling is to be done (TRUE) or not (FALSE) on functional indices. Scaling is used to be able to compare indices values between assemblages. Default: scaling = TRUE.
check_input	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check.input = TRUE.
details_returned	a logical value indicating whether the user want to store details. Details are used in graphical functions and thus must be kept if the user want to have graphical outputs for the computed indices.
verbose	a logical value indicating whether progress details should be printed in the console. If FALSE does not provide percent progress when computing diversity indices.

Value

The following list is returned:

- *functional_diversity_indices* matrix containing indices values (columns) for each assemblage (rows)
- *details* list: a **asb_sp_occ** data.frame of species occurrences in each assemblage ; a **asb_sp_relatw** matrix of relative weight of species in each assemblage ; a **sp_coord_all_asb** list of matrices of species coordinates along functional axes for species present in each assemblage ; a **vert_nm_all_asb** list of vectors of species names being vertices of the convex hull for each assemblage ; a **mst_all_asb** list of data.frames summarizing link between species in the minimum spanning tree of each assemblage ; a **grav_center_vert_coord_all_asb** list of vectors of coordinates of the vertices gravity center for each assemblage ; a **mean_dtogravcenter_all_asb** list of vectors containing mean distance to the species gravity center for each assemblage ; a **dist_gravcenter_global_pool** vector containing the distance of each species to the gravity center of all species from the global pool ; a **dist_nn_global_pool** data.frame showing the distances of each species from the global pool to its nearest neighbor ; a **nm_nn_all_asb** data.frame containing the name of each nearest neighbor of each species present in a given assemblage ; a **dist_nn_all_asb** data.frame containing distance of each species present in a given assemblage to its nearest neighbor.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data('fruits_traits', package = 'mFD')

# Load Assemblages*Species dataframe:
data('baskets_fruits_weights', package = 'mFD')

# Load Traits categories dataframe:
data('fruits_traits_cat', package = 'mFD')

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(
  sp_dist             = sp_dist_fruits,
  maxdim_pcoa         = 10,
  deviation_weighting = 'absolute',
  fdist_scaling       = FALSE,
  fdendro             = 'average')

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute alpha diversity indices
alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c('PC1', 'PC2', 'PC3', 'PC4')],
  asb_sp_w         = baskets_fruits_weights,
  ind_vect         = c('fdis', 'fmpd', 'fnnd', 'feve', 'fric', 'fdiv',
                       'fori', 'fspe'),
  scaling          = TRUE,
  check_input      = TRUE,
  details_returned = TRUE)

# Retrieve alpha diversity indices table
fd_ind_values_fruits <- alpha_fd_indices_fruits$functional_diversity_indices
fd_ind_values_fruits
```

`alpha.multidim.plot` *Plot functional space and chosen functional indices*

Description

Compute a graphical representation of functional indices. **To plot functional indices, functional indices values must have been retrieve through the use of the** `alpha.fd.multidim` **function**.

Usage

```
alpha.multidim.plot(
  output_alpha_fd_multidim,
  plot_asb_nm,
  ind_nm = c("fide", "fric", "fdiv", "fdis", "feve", "fori", "fspe", "fnnd"),
  faxes = NULL,
  faxes_nm = NULL,
  range_faxes = c(NA, NA),
  color_bg = "grey95",
  shape_sp = c(pool = 3, asb1 = 21, asb2 = 21),
  size_sp = c(pool = 0.7, asb1 = 1, asb2 = 1),
  color_sp = c(pool = "grey50", asb1 = "#0072B2", asb2 = "#D55E00"),
  color_vert = c(pool = "grey50", asb1 = "#0072B2", asb2 = "#D55E00"),
  fill_sp = c(pool = NA, asb1 = "#FFFFFF30", asb2 = "#FFFFFF30"),
  fill_vert = c(pool = NA, asb1 = "#0072B2", asb2 = "#D55E00"),
  color_ch = c(pool = NA, asb1 = "#0072B2", asb2 = "#D55E00"),
  fill_ch = c(pool = "white", asb1 = "#0072B2", asb2 = "#D55E00"),
  alpha_ch = c(pool = 1, asb1 = 0.3, asb2 = 0.3),
  shape_centroid_fdis = c(asb1 = 22, asb2 = 22),
  shape_centroid_fdiv = c(asb1 = 24, asb2 = 25),
  shape_centroid_fspe = 23,
  color_centroid_fspe = "black",
  size_sp_nm = 3,
  color_sp_nm = "black",
  plot_sp_nm = NULL,
  fontface_sp_nm = "plain",
  save_file = FALSE,
  check_input = TRUE
)
```

Arguments

`output_alpha_fd_multidim`
: a list of objects retrieved through the `alpha.fd.multidim` function.

`plot_asb_nm`
: a vector containing name(s) of assemblage(s) to plot.

`ind_nm`
: a vector of character string of the name of functional indices to plot. **Indices names must be written in lower case letters**. Possible indices to compute are: "fdis", "feve", "fric", "fdiv", "fori" and "fspe". Default: all the indices are computed.

`faxes`
: a vector with names of axes to plot. **You can only plot from 2 to 4 axes for graphical reasons: vector length should be between 2 and 4**. Default: faxes = NULL (the four first axes will be plotted).

`faxes_nm` a vector with axes labels if the user want different axes labels than `faxes` ones. Default: faxes_nm = faxes (labels will the the same that `faxes` ones).

`range_faxes` a vector with minimum and maximum for values for axes. Note that to have a fair representation of position of species in all plots, all axes must have the same range. Default: faxes_lim = c(NA, NA) (the range is computed according to the range of values among all axes, all axes having the same range).

`color_bg` a R color name or an hexadecimal code used to fill plot background. Default: `color_bg = "grey95"`.

`shape_sp` a vector gathering numeric values referring to the symbol used to draw species from the global pool and the plotted assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...). (0 = high transparency, 1 = no transparency).

`size_sp` a vector gathering numeric values referring to the size of species belonging to the global pool and the plotted assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`color_sp` a vector gathering R color names or hexadecimal codes referring to the color of species from the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`color_vert` a vector gathering R color names or hexadecimal codes referring to the color of vertices from the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`fill_sp` a vector gathering R color names or hexadecimal codes referring to the filled color of species from the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`fill_vert` a vector gathering R color names or hexadecimal codes referring to the filled color of vertices from the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`color_ch` a vector gathering R color names or hexadecimal codes referring to the color of the convex pool of the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`fill_ch` a vector gathering R color names or hexadecimal codes referring to the color to fill the convex pool of the global pool and studied assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...).

`alpha_ch` a vector gathering numeric values referring to the opacity of convex hulls of the global pool and the plotted assemblage(s). It should be written as c(pool = "...", asb1 = "...", ...). (0 = high transparency, 1 = no transparency).

`shape_centroid_fdis`
a vector gathering numeric value(s) used to draw FDis centroid size.

`shape_centroid_fdiv`
a vector gathering numeric value(s) used to draw FDiv centroid size.

`shape_centroid_fspe`
a vector gathering numeric value used to draw FSpe centroid (i.e. center of the functional space) size.

`color_centroid_fspe`
a vector gathering R color name or hexadecimal code used to draw FSpe centroid (i.e. center of the functional space) color.

size_sp_nm	a numeric value referring to the size of species names if plotted.
color_sp_nm	a R color name or hexadecimal code referring to the color of names of species if plotted.
plot_sp_nm	a vector containing species names that are to be plotted. Default: plot_nm_sp = NULL (no name plotted).
fontface_sp_nm	a character string for font of species labels (e.g. "italic", "bold"). Default: fontface_sp_nm = 'plain'.
save_file	a logical value telling if plots should be locally saved or not.
check_input	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check.input = TRUE.

Value

If name_file is NULL, it returns a list of one ggplot2 plots per functional index containing plots for combinations of up to four axes, a patchwork figure gathering all combinations of axes and a ggplot2 figure showing the plot caption. If name_file is not NULL, then those plots are saved locally.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")
```

```
# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute alpha diversity indices:
alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  asb_sp_w         = baskets_fruits_weights,
  ind_vect         = c("fdis", "fmpd", "fnnd", "feve", "fric", "fdiv",
                       "fori", "fspe"),
  scaling          = TRUE,
  check_input      = TRUE,
  details_returned = TRUE)

# Plot all fd alpha indices:
plots_alpha <- mFD::alpha.multidim.plot(
output_alpha_fd_multidim = alpha_fd_indices_fruits,
plot_asb_nm              = c("basket_1", "basket_5"),
ind_nm                   = c("fdis", "fide", "fnnd", "feve",
                             "fric", "fdiv", "fori",
                             "fspe"),
faxes                    = NULL,
faxes_nm                 = NULL,
range_faxes              = c(NA, NA),
color_bg                 = "grey95",
shape_sp                 = c(pool = 3, asb1 = 21,
                             asb2 = 21),
size_sp                  = c(pool = 0.7, asb1 = 1,
                             asb2 = 1),
color_sp                 = c(pool = "grey50",
                             asb1 = "#1F968BFF",
                             asb2 = "#DCE319FF"),
color_vert               = c(pool = "grey50",
                             asb1 = "#1F968BFF",
                             asb2 = "#DCE319FF"),
fill_sp                 = c(pool =  NA,
                            asb1 = "#1F968BFF",
                            asb2 = "#DCE319FF"),
fill_vert               = c(pool = NA,
                            asb1 = "#1F968BFF",
                            asb2 = "#DCE319FF"),
color_ch                = c(pool = NA,
                            asb1 = "#1F968BFF",
                            asb2 = "#DCE319FF"),
fill_ch                 = c(pool = "white",
                            asb1 = "#1F968BFF",
                            asb2 = "#DCE319FF"),
alpha_ch                = c(pool = 1, asb1 = 0.3,
                            asb2 = 0.3),
shape_centroid_fdis     = c(asb1 = 22,  asb2 = 24),
shape_centroid_fdiv     = c(asb1 = 22,  asb2 = 24),
shape_centroid_fspe     = 23,
color_centroid_fspe     = "black",
```

```
size_sp_nm              = 3,
color_sp_nm             = "black",
plot_sp_nm              = NULL,
fontface_sp_nm          = "plain",
save_file               = FALSE,
check_input             = TRUE)

# Check FRic plot:
plots_alpha$fric$patchwork
```

`asb.sp.summary` *Summarize Assemblage x Species data frame*

Description

This function computes a summary helping you to picture assemblages. For this function to work, there must be no NA in your `asb_sp_w` data frame.

Usage

```
asb.sp.summary(asb_sp_w)
```

Arguments

`asb_sp_w`	a matrix showing assemblages (rows) composition in species (columns). Note that species names **must be** the names of rows.

Value

A list with:

`asb_sp_w_occ`	a matrix with species occurrences in each assemblage.
`sp_tot_w`	a vector gathering species biomass/abundance per species.
`asb_tot_w`	a vector gathering total abundance/biomass per assemblage.
`asb_sp_richn`	a vector gathering species richness per assemblage.
`asb_sp_nm`	a list gathering the names of species of each assemblage.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Assemblages x Species Matrix
data('baskets_fruits_weights', package = 'mFD')

# Summarize Assemblages Data
mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)
```

`background.plot` *Plot background of multidimensional plots*

Description

This function creates a ggplot object with customized axes range (same for both), names and background

Usage

```
background.plot(range_faxes, faxes_nm, color_bg)
```

Arguments

`range_faxes`	a vector with minimum and maximum values of axes. Note that to have a fair representation of position of species in all plots, they should have the same range. Default: `range_faxes = c(NA, NA)` (the range is computed according to the range of values among all axes).
`faxes_nm`	a vector with axes labels for figure.
`color_bg`	a R color name or an hexadecimal code used to fill plot background. Default: `color_bg = "grey95"`.

Value

A ggplot object plotting background of multidimensional graphs.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
background <- background.plot(range_faxes = c(-1, 2),
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")
background
```

baskets_fruits_weights *Dataset: Baskets Composition in Fruits Species*

Description

This dataset represents the abundance of 25 fruits species in 10 baskets.

Usage

```
baskets_fruits_weights
```

Format

A matrix of integers with 10 rows (baskets) and 25 columns (species).

See Also

fruits_traits, fruits_traits_cat

Examples

```
## Not run:
# Load Assemblages x Species Matrix
data("baskets_fruits_weights", package = "mFD")
baskets_fruits_weights[1:5, 1:5]

# Summarize Assemblages Data
mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)

## End(Not run)
```

beta.fd.hill *Compute Functional beta-Diversity indices based on Hill Numbers*

Description

Compute functional beta-diversity indices based on Hill numbers applied to distance between species following the framework from Chao *et al.* (2019).

Usage

```
beta.fd.hill(
  asb_sp_w,
  sp_dist,
  q = c(0, 1, 2),
  tau = "mean",
  beta_type = "Jaccard",
  check_input = TRUE,
  details_returned = TRUE
)
```

Arguments

asb_sp_w	a matrix with weight of species (columns) in a set of assemblages (rows). Rows and columns should have names. NA are not allowed.
sp_dist	a matrix or dist object with distance between species. Species names should be provided and match those in 'asb_sp_w'. NA are not allowed.
q	a vector containing values referring to the order of diversity to use
tau	a character string with name of function to apply to distance matrix (i.e. among all pairs of species) to get the threshold used to define 'functionally indistinct set of species'. Could be qet to 'mean' (default), 'min' or 'max'.
beta_type	a character string with name of framework used for computing beta-diversity, either 'Jaccard' (default) or 'Sorensen'.
check_input	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check_input = TRUE.
details_returned	a logical value indicating whether the user want to store values used for computing indices (see list below)

Value

A list with:

- *asb_FDbeta* a list with for each value of q a *dist* object with beta functional diversity indices for all pairs of assemblages item if **store.details** turned to TRUE a list *details* with
  - *malpha_fd_q* a list with for each value of q a *dist* object with mean alpha functional diversity indices for all pairs of assemblages
  - *gamma_fd_q* a list with for each value of q a *dist* object with gamma functional diversity indices for all pairs of assemblages

Note

When q=1 Jaccard-like and Sorensen-like beta-diversity are identical. FD computed with tau='min' is equivalent to Hill number taxonomic beta diversity. If tau='min' and there are species with null distance, tau is set to the minimum non-null value and a warning message is displayed. Indices values are stored as *dist* objects to optimize memory. See below example of how merging distance values in a *dataframe* with dist.to.df

Author(s)

Sebastien Villeger and Camille Magneville

References

Chao *et al.* (2019) An attribute diversity approach to functional diversity, functional beta diversity, and related (dis)similarity measures. *Ecological Monographs*, **89**, e01343.

Examples

```
# Load Species*Traits dataframe:
data('fruits_traits', package = 'mFD')

# Load Traits types dataframe:
data('fruits_traits_cat', package = 'mFD')

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute beta functional hill indices:
baskets_beta <- beta.fd.hill(
      asb_sp_w         = baskets_fruits_weights,
      sp_dist          = sp_dist_fruits,
      q                = c(0,1,2),
      tau              = 'mean',
      beta_type        = 'Jaccard',
      check_input      = TRUE,
      details_returned = TRUE)

# Then use the mFD::dist.to.df function to ease visualizing result:
## for q = 0:
mFD::dist.to.df(list_dist = list(FDq2 = baskets_beta$beta_fd_q$q0))
## for q = 1:
mFD::dist.to.df(list_dist = list(FDq2 = baskets_beta$beta_fd_q$q1))
## for q = 2:
mFD::dist.to.df(list_dist = list(FDq2 = baskets_beta$beta_fd_q$q2))
```

`beta.fd.multidim` *Compute Functional beta-Diversity indices for pairs of assemblages in a multidimensional space*

Description

Computes a set of indices of pairwise functional beta-diversity (dissimilarity and its turnover and nestedness-resultant components) based on overlap between convex hulls in a multidimensional space. For details about indices formulas see Villeger *et al.* (2013). This functions stands on `functional.betapart.core.pairwise`.

Usage

```
beta.fd.multidim(
  sp_faxes_coord,
  asb_sp_occ,
  check_input = TRUE,
  beta_family = "Jaccard",
  details_returned = TRUE,
  betapart_step = TRUE,
  betapart_chullopt = list(conv1 = "Qt", conv2 = "QJ"),
  betapart_para = FALSE,
  betapart_para_opt = betapart::beta.para.control()
)
```

Arguments

`sp_faxes_coord`	a matrix with coordinates of species (rows) on functional axes (columns). Species coordinates have been retrieved thanks to `tr.cont.fspace` or `quality.fspaces`.
`asb_sp_occ`	a matrix with presence/absence (coded as 0/1) of species (columns) in a set of assemblages (rows).
`check_input`	a logical value defining whether inputs are checked before computation of indices. Possible error messages will thus may be more understandable for the user than R error messages. Default: `check_input = TRUE`.
`beta_family`	a character string for the type of beta-diversity index to use, 'Jaccard' (default) and/or 'Sorensen'.
`details_returned`	a logical value indicating whether the user wants to details_returned. Details are used in the graphical function `beta.multidim.plot` and thus must be kept if the user want to have graphical outputs for the computed indices.
`betapart_step`	a logical value indicating whether the computation progress should be displayed in the R console. Default: `betapart_step = TRUE`.
`betapart_chullopt`	a A list of two named vectors of character conv1 and conv2 defining the options that will be used to compute the convex hulls (through the options of geometry::convhulln function). For further details see help of `functional.betapart.core.pairwise`. Default: `betapart_chullopt = c(conv1 = 'Qt', conv2 = 'QJ')`.
`betapart_para`	a logical value indicating whether internal parallelization should be used to compute pairwise dissimilarities. Default: `betapart_para = FALSE`.
`betapart_para_opt`	a list with details about parallelization. Default value means those parameters are set according to computer specifications. `nc` is the number of cores (default

= 4), `type` is a character string with code of method used (default PSOCK), `LB` is a boolean specifying whether load-balancing is applied (default is TRUE) and `size` is a numeric value for number of tasks performed at each time (default is 1). See help of `functional.betapart.core.pairwise` for more details.

Value

A list with:

- *pairasb_fbd_indices* a list with for each index a *dist* object with values for all pairs of assemblages. Indices names start with the abbreviation of the type of dissimilarity index ('jac' for Jaccard-like and 'sor' for Sorensen-like dissimilarity) and end with abbreviation of component ('diss': dissimilarity, 'turn' its turnover component, and 'nest' its nestedness-resultant component).
- if *store_details* is TRUE,
- *details_beta* list: **inputs** a list with *sp_faxes_coord* and *asb_sp_occ* on which indices were computed (required for drawing graphics), **pool_vertices** a list of vectors (1 per assemblage) with names of species being vertices of the convex hull shaping all species; **asb_FRic** a vector with volume of the convex hull shaping each assemblage (relative to volume filled by all species) ; **asb_vertices** a list of vectors (1 per assemblage) with names of species being vertices of the convex hull

Note

All assemblages should have a number of species strictly higher than the number of functional axes. Computing intersection of convex hulls in space of >5 dimensions is yet impossible with most computers. This function uses R libraries 'betapart' (> =1.5.4) for indices computation. Indices values are stored as *dist* objects to optimize memory. See below example of how merging distance values in a *dataframe* with `dist.to.df`.

Author(s)

Sebastien Villeger and Camille Magneville

References

Villeger *et al.* (2013) Decomposing functional beta-diversity reveals that low functional beta-diversity is driven by low functional turnover in European fish assemblages. *Global Ecology and Biogeography*, **22**, 671-681.

Examples

```
## Not run:
 # Load Species*Traits dataframe:
data('fruits_traits', package = 'mFD')

# Load Assemblages*Species dataframe:
data('baskets_fruits_weights', package = 'mFD')

# Load Traits categories dataframe:
```

```
data('fruits_traits_cat', package = 'mFD')

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(
 sp_dist             = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = 'absolute',
 fdist_scaling       = FALSE,
 fdendro             = 'average')

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Get the occurrence dataframe:
asb_sp_fruits_summ <- mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)
asb_sp_fruits_occ <- asb_sp_fruits_summ$'asb_sp_occ'

# Compute beta diversity indices:
beta_fd_fruits <- mFD::beta.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c('PC1', 'PC2', 'PC3', 'PC4')],
  asb_sp_occ       = asb_sp_fruits_occ,
  check_input      = TRUE,
  beta_family      = c('Jaccard'),
  details_returned = TRUE)

# merging pairwise beta-diversity indices in a data.frame
dist.to.df(beta_fd_fruits$pairasb_fbd_indices)

## End(Not run)
```

`beta.multidim.plot` *Illustrate Functional beta-Diversity indices for pairs of assemblages in a multidimensional space*

Description

Illustrate overlap between convex hulls shaping species assemblages in a multidimensional functional space.**Before plotting beta functional diversity indices should have been computed using the** `beta.fd.multidim` **function**.

Usage

```
beta.multidim.plot(
  output_beta_fd_multidim,
  plot_asb_nm,
  beta_family,
  plot_sp_nm = NULL,
  faxes = NULL,
  name_file = NULL,
  faxes_nm = NULL,
  range_faxes = c(NA, NA),
  color_bg = "grey95",
  shape_sp = c(pool = 3, asb1 = 22, asb2 = 21),
  size_sp = c(pool = 0.7, asb1 = 1.2, asb2 = 1),
  color_sp = c(pool = "grey50", asb1 = "blue", asb2 = "red"),
  fill_sp = c(pool = NA, asb1 = "white", asb2 = "white"),
  fill_vert = c(pool = NA, asb1 = "blue", asb2 = "red"),
  color_ch = c(pool = NA, asb1 = "blue", asb2 = "red"),
  fill_ch = c(pool = "white", asb1 = "blue", asb2 = "red"),
  alpha_ch = c(pool = 1, asb1 = 0.3, asb2 = 0.3),
  nm_size = 3,
  nm_color = "black",
  nm_fontface = "plain",
  check_input = TRUE
)
```

Arguments

`output_beta_fd_multidim`
: the list returned by `beta.fd.multidim` when `details_returned = TRUE`. Thus, even if this function will illustrate functional beta-diversity for a single pair of assemblages, plots will be scaled according to all assemblages for which indices were computed.

`plot_asb_nm`
: a vector with names of the 2 assemblages for which functional beta-diversity will be illustrated.

`beta_family`
: a character string for the type of beta-diversity index for which values will be printed, `'Jaccard'` (default) and/or `'Sorensen'`.

`plot_sp_nm`
: a vector containing species names that are to be plotted. Default: `plot_nm_sp = NULL` (no name plotted).

`faxes`
: a vector with names of axes to plot (as columns names in `output_beta_fd_multidim$details$input$s`). **You can only plot from 2 to 4 axes for graphical reasons**. Default: `faxes = NULL` (the four first axes will be plotted).

`name_file`
: a character string with name of file to save the figure (without extension). Default: `name_file = NULL` which means plot is displayed.

`faxes_nm`
: a vector with axes labels for figure. Default: as `faxes`).

`range_faxes`
: a vector with minimum and maximum values of axes. Note that to have a fair representation of position of species in all plots, they should have the same

range. Default: faxes_lim = c(NA, NA) (the range is computed according to the range of values among all axes).

color_bg — a R color name or an hexadecimal code used to fill plot background. Default: color_bg = "grey95".

shape_sp — a vector with 3 numeric values referring to the shape of symbol used for species from the 'pool' absent from the 2 assemblages, and for species present in the 2 assemblages ('asb1', and 'asb2'), respectively. Default: shape_sp = c(pool = 3, asb1 = 22, asb2 = 21) so cross, square and circle.

size_sp — a numeric value referring to the size of symbols for species. Default: is size_sp = c(pool = 0.8, asb1

color_sp — a vector with 3 names or hexadecimal codes referring to the colour of symbol for species. Default is: color_sp = c(pool = "grey50", asb1 = "blue", asb2 = "red").

fill_sp — a vector with 3 names or hexadecimal codes referring to the color to fill symbol (if shape_sp > 20) for species of the pool and of the 2 assemblages. Default is: fill_sp = c(pool = NA, asb1 = "white", asb2 = "white").

fill_vert — a vector with 3 names or hexadecimal codes referring to the colour to fill symbol (if shape_sp > 20) for species being vertices of the convex hulls of the pool of species and of the 2 assemblages. Default is: fill_vert = c(pool = NA, asb1 = "blue", asb2 = "red").

color_ch — a vector with 3 names or hexadecimal codes referring to the border of the convex hulls of the pool of species and by the 2 assemblages. Default is: color_ch = c(pool = NA, asb1 = "blue", asb2 = "red").

fill_ch — a vector with 3 names or hexadecimal codes referring to the filling of the convex hull of the pool of species and of the 2 assemblages. Default is fill_ch = c(pool = "white", asb1 = "blue", asb2 = "red").

alpha_ch — a vector with 3 numeric value for transparency of the filling of the convex hulls (0 = high transparency, 1 = no transparency). Default is: alpha_ch = c(pool = 1, asb1 = 0.3, asb2 = 0.3).

nm_size — a numeric value for size of species label. Default is 3 (in points).

nm_color — a R color name or an hexadecimal code referring to the color of species label. Default is black.

nm_fontface — a character string for font of species labels (e.g. "italic", "bold"). Default is 'plain'.

check_input — a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check.input = TRUE.

Value

If name_file is NULL, it returns a patchwork figure with overlap between convex hulls projected in 2-dimensional spaces for the given pair of assemblages. Values of functional beta-diversity indices are shown on top-right corner of the figure. If name_file is not NULL, the plot is saved locally.

Author(s)

Sebastien Villeger and Camille Magneville

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
                                   ordinal_var   = "classic",
                                   weight_type   = "equal",
                                   stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
                                  sp_dist             = sp_dist_fruits,
                                  maxdim_pcoa         = 10,
                                  deviation_weighting = "absolute",
                                  fdist_scaling       = FALSE,
                                  fdendro             = "average")

# Retrieve species coordinates matrix:
 sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Get the occurrence dataframe:
 asb_sp_fruits_summ <- mFD::asb.sp.summary(asb_sp_w = baskets_fruits_weights)
 asb_sp_fruits_occ <- asb_sp_fruits_summ$"asb_sp_occ"

# Compute beta diversity indices:
 beta_fd_fruits <- mFD::beta.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  asb_sp_occ       = asb_sp_fruits_occ,
  check_input      = TRUE,
  beta_family      = c("Jaccard"),
  details_returned = TRUE)

# Compute beta fd plots:
 beta.multidim.plot(
   output_beta_fd_multidim = beta_fd_fruits,
   plot_asb_nm             = c("basket_1", "basket_6"),
   beta_family             = c("Jaccard"),
   plot_sp_nm              = c("apple", "cherry", "lemon"),
   faxes                   = paste0("PC", 1:4),
   name_file               = NULL,
   faxes_nm                = NULL,
   range_faxes             = c(NA, NA),
```

```
color_bg              = "grey95",
shape_sp              = c(pool = 3, asb1 = 22, asb2 = 21),
size_sp               = c(pool = 0.8, asb1 = 1, asb2 = 1),
color_sp              = c(pool = "grey50", asb1 = "blue",
                          asb2 = "red"),
fill_sp               = c(pool = NA, asb1 = "white", asb2 = "white"),
fill_vert             = c(pool = NA, asb1 = "blue", asb2 = "red"),
color_ch              = c(pool = NA, asb1 = "blue", asb2 = "red"),
fill_ch               = c(pool = "white", asb1 = "blue",
                          asb2 = "red"),
alpha_ch              = c(pool = 1, asb1 = 0.3, asb2 = 0.3),
nm_size               = 3,
nm_color              = "black",
nm_fontface           = "plain",
check_input           = TRUE)
```

`dist.nearneighb` *Compute distance of a given point to its nearest neighbor in the functional space and the identity of the nearest neighbor*

Description

This function is used in functional indices computation.

Usage

```
dist.nearneighb(sp_faxes_coord, ref_sp)
```

Arguments

`sp_faxes_coord`	a matrix of species coordinates in a chosen functional space. Species coordinates have been retrieved thanks to `tr.cont.fspace` or `quality.fspaces`.
`ref_sp`	a character string referring to the name of the reference species.

Value

A list containing the nearest neighbor identity `nn_id` and a list of the distance of the reference point to its nearest neighbor `nn_ref_sp_dist`.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
                                   ordinal_var   = "classic",
                                   weight_type   = "equal",
                                   stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
                                  sp_dist             = sp_dist_fruits,
                                  maxdim_pcoa         = 10,
                                  deviation_weighting = "absolute",
                                  fdist_scaling       = FALSE,
                                  fdendro             = "average")

# Retrieve species coordinates matrix:
 sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute the distance of "pear" to its nearest neighbor(s):
 dist_nn_pear <- dist.nearneighb(sp_faxes_coord_fruits, ref_sp = "pear")
 dist_nn_pear
```

`dist.point` *Compute distances of all points to a given point in the functional space*

Description

This function computes the distances of all species to a reference species. It is used in FSpe, FOri and FNND computation.

Usage

```
dist.point(sp_faxes_coord, ref_sp)
```

Arguments

sp_faxes_coord	a matrix of species coordinates in a chosen functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
ref_sp	a character string referring to the name of the reference species.

Value

A vector of species distances to the reference species.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
                                   ordinal_var   = "classic",
                                   weight_type   = "equal",
                                   stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
                                  sp_dist             = sp_dist_fruits,
                                  maxdim_pcoa         = 10,
                                  deviation_weighting = "absolute",
                                  fdist_scaling       = FALSE,
                                  fdendro             = "average")

# Retrieve species coordinates matrix:
 sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Retrieve the distances of all species to "pear":
 dist_pear <- dist.point(sp_faxes_coord_fruits, ref_sp = "pear")
 dist_pear
```

`dist.to.df` *Merge distance object(s) into a single data frame*

Description

This function merges distance object(s) into a single data frame which rows are pairs of elements and column(s) distance metric(s). It stands on the `dist_long` function.

Usage

```
dist.to.df(list_dist)
```

Arguments

`list_dist`	a list of dist object(s). All dist objects should have a name (e.g. name of distance metric) and the same labels (i.e. names of sets between which distance was computed).

Value

A data frame which first and second columns (names `x1` and `x2`) contain names of the 2 sets involved in each pair, and with one column for each dist object (named after its name in `list_dist`.

Author(s)

Sebastien Villeger

Examples

```
# Create dist objects:
dist_A <- round(dist(matrix(runif(10, 0, 1), 5, 2,
                      dimnames = list(letters[1:5], NULL))), 2)
dist_B <- round(dist(matrix(runif(10, 0, 1), 5, 2,
                      dimnames = list(letters[1:5], NULL))), 2)
dist_C <- round(dist(matrix(runif(10, 0, 1), 5, 2,
                      dimnames = list(letters[1:5], NULL))), 2)

# First example with only 1 distance:
dist.to.df(list(dA = dist_A))

# Second example with 3 distances:
dist.to.df(list(d1 = dist_A, d2 = dist_B, d3 = dist_C))
```

`fdis.plot` *Plot FDis index*

Description

This function plots FDis index for a given pair of functional axes and for one or several assemblages. It adds segments between species relative weights and assemblage cenntroid on the background plot.

Usage

```
fdis.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_fide_coord2D,
  plot_sp = TRUE,
  shape_sp,
  color_sp,
  fill_sp,
  shape_fide,
  size_fide,
  color_fide,
  fill_fide,
  color_segment,
  width_segment,
  linetype_segment
)
```

Arguments

`ggplot_bg`	a ggplot object of the plot background retrieved through the `background.plot` function.
`asb_sp_coord2D`	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
`asb_sp_relatw`	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the `alpha.fd.multidim`.
`asb_fide_coord2D`	a list (with names as in `asb_sp_coord2D`) of vectors with coordinates of the fide centroid of species for each assemblage for a given pair of axes.
`plot_sp`	a logical value indicating whether species of each assemblage should be plotted or not. Default: `plot_sp = TRUE`
`shape_sp`	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`color_sp` a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_sp` a R color name or an hexadecimal code referring to the color of species symbol filling (if `shape_sp` > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`shape_fide` a numeric value referring to the shape of the symbol used for fide centroid if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`size_fide` a numeric value referring to the size of the symbol used for fide centroid plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`color_fide` a R color name or an hexadecimal code referring to the color of fide centroid if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_fide` a R color name or an hexadecimal code referring to the color to fill assemblage centroid symbol (if `shape_sp` > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`color_segment` a R color name or an hexadecimal code referring to the color of of the segment linking axes and centroid from the studied assemblage if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`width_segment` a numeric value referring to the width of the segment linking fide centroid and functional axes if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`linetype_segment` a numeric value referring to the linetype of the segment linking fide centroid and functional axes if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object showing FDis index for one or several assemblage(s) and a given pair of functional axes.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                               range_faxes = range_faxes_lim,
                               faxes_nm    = c("PC 1", "PC 2"),
                               color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm         = "basket_1",
                             sp_faxes_coord = sp_faxes_coord_fruits,
                             asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]
```

```
# Use alpha.fd.multidim() function to get inputs to plot FIde:
alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
sp_faxes_coord    = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
 asb_sp_w         = baskets_fruits_weights,
 ind_vect         = c("fdis"),
 scaling          = TRUE,
 check_input      = TRUE,
 details_returned = TRUE)

# Retrieve fide values through alpha.fd.multidim outputs:
fruits_asb_fide_coord2D <-
 alpha_fd_indices_fruits$functional_diversity_indices[c("fide_PC1",
                                                        "fide_PC2")]
fruits_asb_fide_coord2D_b1 <- fruits_asb_fide_coord2D[c("basket_1"), ]
fruits_asb_sp_relatw_b1 <-
        alpha_fd_indices_fruits$details$asb_sp_relatw["basket_1", ]

# Retrieve FDis plot:
fdis_plot <- fdis.plot(ggplot_bg = ggplot_bg_fruits,
          asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
          asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
          asb_fide_coord2D = list(basket_1 = fruits_asb_fide_coord2D_b1),
          plot_sp = TRUE,
          shape_sp = 16,
          color_sp = "red",
          fill_sp = "red",
          shape_fide = list(basket_1 = 18),
          size_fide = list(basket_1 = 5),
          color_fide = list(basket_1 = "blue"),
          fill_fide = list(basket_1 = "blue"),
          color_segment = list(basket_1 = "red"),
          width_segment = list(basket_1 = 1),
          linetype_segment = list(basket_1 = "dashed"))

fdis_plot
```

`fdiv.plot` *Plot FDiv indice*

Description

This plot fDiv indice for a given pair of functional axes and one or several assemblages. This function adds mean distance to center of gravity of vertices, points and vertices of 1:N assemblages on the background plot

Usage

```
fdiv.plot(
  ggplot_bg,
```

```
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_vertices_nD,
  asb_vertG_coord2D,
  plot_sp = TRUE,
  shape_sp,
  color_sp,
  fill_sp,
  shape_vert,
  color_vert,
  fill_vert,
  shape_vertG,
  size_vertG,
  color_vertG,
  fill_vertG
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
asb_sp_relatw	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the alpha.fd.multidim.
asb_vertices_nD	a list (with names as in asb_sp_coord2D) of vectors with names of species being vertices in n dimensions.
asb_vertG_coord2D	a list (with names as in asb_sp_coord2D) containing for each assemblage the coordinates of center of gravity of vertices for a given pair of axes
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE.
shape_sp	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
color_sp	a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_sp	a R color name or an hexadecimal code referring to the color of species symbol filling (if shape_sp > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`shape_vert` a numeric value referring to the shape of the symbol used for vertices plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`color_vert` a R color name or an hexadecimal code referring to the color of vertices if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...). If color_vert = NA, vertices are not plotted (for shapes only defined by color, ie shape inferior to 20. Otherwise `fill` must also be set to NA).

`fill_vert` a R color name or an hexadecimal code referring to the color of vertices symbol filling (if `shape_vert` >20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...). If `fill` = `NA` and `color` = `NA`, vertices are not plotted (if `shape_vert` superior to 20

`shape_vertG` a numeric value referring to the shape to use to plot the center of gravity of vertices if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`size_vertG` a numeric value referring to the size of the symbol used for the center of gravity of vertices if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`color_vertG` a R color name or an hexadecimal code referring to the color of the center of gravity of vertices. if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_vertG` a R color name or an hexadecimal code referring to the color to fill the center of gravity of vertices (if `shape_vert` >20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

Value

A ggplot object plotting background of multidimensional graphs and FDiv indice.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")
```

```
# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                               range_faxes = range_faxes_lim,
                               faxes_nm    = c("PC 1", "PC 2"),
                               color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                             sp_faxes_coord = sp_faxes_coord_fruits,
                             asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]

 # Use alpha.fd.multidim() function to get inputs to plot FDiv:
 alpha_fd_ind <- mFD::alpha.fd.multidim(
sp_faxes_coord   = sp_faxes_coord_fruits[ , c("PC1", "PC2", "PC3", "PC4")],
asb_sp_w         = baskets_fruits_weights,
ind_vect         = c("fdiv"),
scaling          = TRUE,
check_input      = TRUE,
details_returned = TRUE)
```

```
# Retrieve inputs of the fdiv.plot() function for "basket_1" and PC1, PC2
# ... through alpha.fd.multidim outputs:
fruits_asb_sp_relatw_b1 <-
        alpha_fd_ind$details$asb_sp_relatw["basket_1", ]
fruits_asb_vertices_nD_b1_2D <-
                    alpha_fd_ind$details$asb_vert_nm["basket_1"]
fruits_asb_vertG_coord_b1 <-
                    alpha_fd_ind$details$asb_G_coord["basket_1"]
fruits_asb_vertG_coord_b1_2D <-
            fruits_asb_vertG_coord_b1[[1]][c("PC1", "PC2")]

# Retrieve FDiv plot:
fdiv_plot <- fdiv.plot(
          ggplot_bg         = ggplot_bg_fruits,
          asb_sp_coord2D    = list(basket_1 = fruits_asb_sp_coord2D_b1),
          asb_sp_relatw     = list(basket_1 = fruits_asb_sp_relatw_b1),
          asb_vertices_nD   = fruits_asb_vertices_nD_b1_2D,
          asb_vertG_coord2D = list(basket_1 = fruits_asb_vertG_coord_b1_2D),
          plot_sp           = TRUE,
          shape_sp          = 16,
          color_sp          = c(basket_1 = "red"),
          fill_sp           = "red",
          color_vert        = "red",
          fill_vert         = "red",
          shape_vert        = 16,
          shape_vertG       = list(basket_1 = 18),
          size_vertG        = list(basket_1 = 2),
          color_vertG       = list(basket_1 = "blue"),
          fill_vertG        = list(basket_1 = "blue"))
fdiv_plot
```

`feve.plot` *Plot FEve index*

Description

This function plots FEve index for a given pair of functional axes and one or several assemblages. It adds Minimum Spanning Tree (MST) of a given assemblage on the background plot.

Usage

```
feve.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_mst,
  plot_sp = TRUE,
  shape_sp,
  color_sp,
```

```
  fill_sp,
  color_mst,
  width_mst,
  linetype_mst
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
asb_sp_relatw	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the alpha.fd.multidim.
asb_mst	a list (with names as in asb_sp_coord2D) of vectors with names of species linked in the MST of the studied assemblage.
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE
shape_sp	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
color_sp	a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_sp	a R color name or an hexadecimal code referring to the color of species symbol filling (if shape_sp > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
color_mst	a R color name or an hexadecimal code referring to the color the MST if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
width_mst	a numeric value referring to the width of segments of the MST if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).
linetype_mst	a numeric value referring to the linetype of the segments representing the MST if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object showing FEve index on the background plot.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                               range_faxes = range_faxes_lim,
                               faxes_nm    = c("PC 1", "PC 2"),
                               color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                             sp_faxes_coord = sp_faxes_coord_fruits,
                             asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]
```

```
# Use alpha.fd.multidim() function to get inputs to plot FIde:
alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
 sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
 asb_sp_w         = baskets_fruits_weights,
 ind_vect         = c("feve"),
 scaling          = TRUE,
 check_input      = TRUE,
 details_returned = TRUE)

# Retrieve fide values through alpha.fd.multidim outputs:
fruits_asb_mst_b1 <-
        alpha_fd_indices_fruits$details$asb_mst["basket_1"]
fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]
fruits_asb_sp_relatw_b1 <-
        alpha_fd_indices_fruits$details$asb_sp_relatw["basket_1", ]

# Retrieve FEve plot:
feve_plot <- feve.plot(ggplot_bg = ggplot_bg_fruits,
          asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
          asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
          asb_mst = fruits_asb_mst_b1,
          plot_sp = TRUE,
          shape_sp = 16,
          color_sp = "red",
          fill_sp = "red",
          color_mst = list(basket_1 = "blue"),
          width_mst = list(basket_1 = 1),
          linetype_mst = list(basket_1 = "solid"))

feve_plot
```

`fide.plot` *Plot FIde index*

Description

Plot FIde index given a pair of functional axes for one or several assemblages. It adds the centroid of species for each assemblage and segments showing centroid coordinates on functional axes on the background plot.

Usage

```
fide.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_fide_coord2D,
  plot_sp = TRUE,
```

```
  shape_sp,
  color_sp,
  fill_sp,
  shape_fide,
  size_fide,
  color_fide,
  fill_fide,
  color_segment,
  width_segment,
  linetype_segment
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
asb_sp_relatw	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the alpha.fd.multidim.
asb_fide_coord2D	a list (with names as in asb_sp_coord2D) of vectors with coordinates of the centroid of species for each assemblage for a given pair of axes.
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE
shape_sp	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
color_sp	a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_sp	a R color name or an hexadecimal code referring to the color of species symbol filling (if shape_sp > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
shape_fide	a numeric value referring to the shape of the symbol used for fide centroid plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
size_fide	a numeric value referring to the size of the symbol used for fide centroid plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`color_fide` a R color name or an hexadecimal code referring to the color of species centroid for agiven assemblage if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_fide` a R color name or an hexadecimal code referring to the color to fill assemblage centroid symbol (if `shape_sp` > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`color_segment` a R color name or an hexadecimal code referring to the color of of the segment linking axes and centroid from the studied assemblage if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`width_segment` a numeric value referring to the width of the segment linking fide centroid and functional axes if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`linetype_segment`
a numeric value referring to the linetype of the segment linking fide centroid and functional axes if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object with FIde index, species and background.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
```

```
                                ordinal_var   = "classic",
                                weight_type   = "equal",
                                stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                             range_faxes = range_faxes_lim,
                             faxes_nm    = c("PC 1", "PC 2"),
                             color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                           sp_faxes_coord = sp_faxes_coord_fruits,
                           asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]

 # Use alpha.fd.multidim() function to get inputs to plot FIde:
 alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
 sp_faxes_coord    = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  asb_sp_w         = baskets_fruits_weights,
  ind_vect         = c("fide"),
  scaling          = TRUE,
  check_input      = TRUE,
  details_returned = TRUE)

 # Retrieve fide values through alpha.fd.multidim outputs:
 fruits_asb_fide_coord2D <-
  alpha_fd_indices_fruits$functional_diversity_indices[c("fide_PC1",
                                                         "fide_PC2")]
 fruits_asb_fide_coord2D_b1 <- fruits_asb_fide_coord2D[c("basket_1"), ]
 fruits_asb_sp_relatw_b1 <-
         alpha_fd_indices_fruits$details$asb_sp_relatw["basket_1", ]

 # Retrieve FIde plot:
 fide_plot <- fide.plot(ggplot_bg = ggplot_bg_fruits,
```

```
            asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
            asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
            asb_fide_coord2D = list(basket_1 = fruits_asb_fide_coord2D_b1),
            plot_sp = TRUE,
            shape_sp = 16,
            color_sp = "red",
            fill_sp = "red",
            shape_fide = list(basket_1 = 18),
            size_fide = list(basket_1 = 5),
            color_fide = list(basket_1 = "blue"),
            fill_fide = list(basket_1 = "blue"),
            color_segment = list(basket_1 = "red"),
            width_segment = list(basket_1 = 1),
            linetype_segment = list(basket_1 = "dashed"))

fide_plot
```

`fnnd.plot` *Plot FNND index*

Description

This function plots FNND index for a given pair of functional axes and for one or several assemblages

Usage

```
fnnd.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_nn_asb,
  plot_sp = TRUE,
  shape_sp,
  color_sp,
  fill_sp,
  color_segment,
  width_segment,
  linetype_segment
)
```

Arguments

`ggplot_bg`	a ggplot object of the plot background retrieved through the `background.plot` function.
`asb_sp_coord2D`	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes

`asb_sp_relatw` a list of vector gathering species relative weight in each assemblage. It can be retrieved through the `alpha.fd.multidim`.

`asb_nn_asb` a list gathering for each species of a studied assemblage its nearest neighbour(s) in the assemblage.

`plot_sp` a logical value indicating whether species of each assemblage should be plotted or not. Default: `plot_sp = TRUE`

`shape_sp` a numeric value referring to the shape used to plot species belonging to the studied assemblage.

`color_sp` a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_sp` a R color name or an hexadecimal code referring to the color of species symbol filling (if `shape_sp` > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`color_segment` a R color name or an hexadecimal code referring to the color of of the segment linking nearest neighbors in a given assemblage or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`width_segment` a numeric value referring to the size of the segment linking nearest neighbors in a given assemblage or a vector of numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRwidth", asb2 = "secondRwidth", ...).

`linetype_segment`
a character string referring to the linetype used to link nearest neighbors in the studied assemblages or a vector of character strings if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object with FNND index.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")
```

```
# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                               range_faxes = range_faxes_lim,
                               faxes_nm    = c("PC 1", "PC 2"),
                               color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                             sp_faxes_coord = sp_faxes_coord_fruits,
                             asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]

 # Use alpha.fd.multidim() function to get inputs to plot FIde:
 alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  asb_sp_w         = baskets_fruits_weights,
  ind_vect         = c("fnnd"),
  scaling          = TRUE,
  check_input      = TRUE,
  details_returned = TRUE)

 # Retrieve nearest neighbor(s) names and relative weights ...
```

```
# ... through alpha.fd.multidim outputs:
fruits_asb_nn_asb_b1 <-
        alpha_fd_indices_fruits$details$asb_nm_nn_asb["basket_1"]
fruits_asb_sp_relatw_b1 <-
        alpha_fd_indices_fruits$details$asb_sp_relatw["basket_1", ]

# Retrieve FNND plot:
fnnd_plot <- fnnd.plot(ggplot_bg = ggplot_bg_fruits,
          asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
          asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
          asb_nn_asb = fruits_asb_nn_asb_b1 ,
          plot_sp = TRUE,
          shape_sp = 16,
          color_sp = "red",
          fill_sp = "red",
          color_segment = list(basket_1 = "blue"),
          width_segment = list(basket_1 = 1),
          linetype_segment = list(basket_1 = "solid"))

fnnd_plot
```

`fori.plot` *Plot FOri*

Description

This function plots FOri index for a given pair of functional axes and for one or several assemblages. It adds the distance of species from the studied to their nearest neighbour(s) from the global pool.

Usage

```
fori.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  asb_nn_pool,
  pool_coord2D,
  plot_pool = TRUE,
  plot_sp = TRUE,
  shape_pool,
  size_pool,
  color_pool,
  fill_pool,
  shape_sp,
  color_sp,
  fill_sp,
  color_segment,
  width_segment,
  linetype_segment
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
asb_sp_relatw	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the alpha.fd.multidim.
asb_nn_pool	a list gathering for each species of a studied assemblage its nearest neighbour(s) in the global pool.
pool_coord2D	a matrix (ncol = 2) with coordinates of species present in the global pool for a given pair of functional axes.
plot_pool	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_pool = TRUE.
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE
shape_pool	a numeric value referring to the shape used to plot species pool.
size_pool	a numeric value referring to the size of species belonging to the global pool.
color_pool	a R color name or an hexadecimal code referring to the color of the pool. This color is also used for FRic convex hull color.
fill_pool	a R color name or an hexadecimal code referring to the colour to fill species symbol (if shape_sp > 20) and the assemblage convex hull.
shape_sp	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
color_sp	a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_sp	a R color name or an hexadecimal code referring to the color of species symbol filling (if shape_sp > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
color_segment	color_segment a R color name or an hexadecimal code referring to the color of of the segment linking nearest neighbors in the global pool or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
width_segment	a numeric value referring to the size of the segment linking nearest neighbors in the global pool or a vector of numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRwidth", asb2 = "secondRwidth", ...).

linetype_segment
: a character string referring to the linetype used to link nearest neighbors in the global pool or a vector of character strings if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object with FOri index.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
## Not run:
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05
```

```
# Retrieve the background plot:
ggplot_bg_fruits <- mFD::background.plot(
                              range_faxes = range_faxes_lim,
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")

# Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                            sp_faxes_coord = sp_faxes_coord_fruits,
                            asb_sp_w       = baskets_fruits_weights)
fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]

# Use alpha.fd.multidim() function to get inputs to plot FIde:
alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
 sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
 asb_sp_w         = baskets_fruits_weights,
 ind_vect         = c("fori"),
 scaling          = TRUE,
 check_input      = TRUE,
 details_returned = TRUE)

# Retrieve nearest neighbor(s) names through alpha.fd.multidim outputs:
fruits_asb_nn_pool_b1 <-
        alpha_fd_indices_fruits$details$asb_nm_nn_pool["basket_1"]

# Retrieve FNND plot:
fori_plot <- fori.plot(ggplot_bg = ggplot_bg_fruits,
          asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
          asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
          asb_nn_pool = fruits_asb_nn_pool_b1,
          pool_coord2D = sp_faxes_coord_fruits[, c("PC1", "PC2")],
          plot_pool = TRUE,
          plot_sp = TRUE,
          shape_sp = 16,
          color_sp = "red",
          fill_sp = "red",
          shape_pool = 4,
          size_pool = 2,
          color_pool = "grey",
          fill_pool = "grey",
          color_segment = list(basket_1 = "red"),
          width_segment = list(basket_1 = 1),
          linetype_segment = list(basket_1 = "solid"))

fori_plot

## End(Not run)
```

`fric.plot` *Plot FRic index*

Description

This function plots FRic index for a given pair of functional axes and one or several assemblages. It adds convex hull(s), points and vertices of 1:N assemblages on the background plot

Usage

```
fric.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_vertices_nD,
  plot_sp = TRUE,
  color_ch,
  fill_ch,
  alpha_ch,
  shape_sp,
  size_sp,
  color_sp,
  fill_sp,
  shape_vert,
  size_vert,
  color_vert,
  fill_vert
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of axes for a given pair of functional axes.
asb_vertices_nD	a list (with names as in asb_sp_coord2D) of vectors with names of species being vertices in n dimensions.
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE.
color_ch	a R color name or an hexadecimal code referring to the color of the border of the convex hull filled by the pool of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_ch	a R color name or an hexadecimal code referring to the color of convex hull filling if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondR-colorname", ...).
alpha_ch	a numeric value referring to the value of transparency of the convex hull filling (0 = high transparency, 1 = no transparency) if one assemblage to plot or a vector

numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRtransparency", asb2 = "secondRtransparency", ...).

`shape_sp` a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`size_sp` a numeric value referring to the size of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`color_sp` a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`fill_sp` a R color name or an hexadecimal code referring to the color of species symbol filling (if `shape_sp` > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

`shape_vert` a numeric value referring to the shape of the symbol used for vertices plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).

`size_vert` a numeric value referring to the size of the symbol used for vertices plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRsize", asb2 = "secondRsize", ...).

`color_vert` a R color name or an hexadecimal code referring to the color of vertices if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...). If color_vert = NA, vertices are not plotted (for shapes only defined by color, ie shape inferior to 20. Otherwise `fill` must also be set to NA).

`fill_vert` a R color name or an hexadecimal code referring to the color of vertices symbol filling (if `shape_vert` >20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...). If `fill` = `NA` and `color` = `NA`, vertices are not plotted (if `shape_vert` superior to 20

Value

A ggplot object plotting background of multidimensional graphs and FRic convex hulls.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
  sp_faxes_coord_fruits <-
    fspaces_quality_fruits$details_fspaces$sp_pc_coord

 # Retrieve species coordinates matrix for the assemblage "basket_1":
  sp_filter <- mFD::sp.filter(asb_nm          = c("basket_1"),
                              sp_faxes_coord = sp_faxes_coord_fruits,
                              asb_sp_w       = baskets_fruits_weights)
  sp_faxes_coord_fruits_b1 <- sp_filter$`species coordinates`

 # Reduce it to the two studid axes: PC1 and PC2:
 sp_faxes_coord_fruits_b1_2D <- sp_faxes_coord_fruits_b1[, c("PC1", "PC2")]

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits + c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                              range_faxes = range_faxes_lim,
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")

 # Retrieve vertices names:
```

```
vert_nm_fruits <- vertices(sp_faxes_coord_fruits_b1_2D,
 order_2D = TRUE, check_input = TRUE)

# Plot in white the convex hull of all fruits species:
ggplot_fric <- mFD::fric.plot(
          ggplot_bg       = ggplot_bg_fruits,
          asb_sp_coord2D  = list(basket_1 = sp_faxes_coord_fruits_b1_2D),
          asb_vertices_nD = list(basket_1 = vert_nm_fruits),
          plot_sp         = TRUE,
          color_ch        = c("basket_1" = "black"),
          fill_ch         = c("basket_1" = "white"),
          alpha_ch        = c("basket_1" = 0.3),
          size_sp = c("basket_1" = 1),
          shape_sp = c("basket_1" = 16),
          color_sp = c("basket_1" = "red"),
          fill_sp = c("basket_1" = "red"),
          size_vert = c("basket_1" = 1),
          color_vert = c("basket_1" = "red"),
          fill_vert = c("basket_1" = "red"),
          shape_vert = c("basket_1" = 16))
ggplot_fric
```

`fruits_traits` *Dataset: Traits Values of Fruits Species*

Description

This dataset represents the value of 6 traits for 15 fruits species. **Important:** Species names must be specified as the data frame row names (not in an additional column).

Usage

```
fruits_traits
```

Format

A data frame with 25 rows (species) and the following columns (traits):

Size an ordered factor describing the size of fruits species

Plant an unordered factor describing the type of plant

Climate an ordered factor describing the climate regions

Seed an ordered factor describing the type of seed

Sugar a numeric describing the quantity of sugar

Use.raw an integer (percentage) describing the proportion of a raw use (fuzzy trait) of the fruit

Use.pastry an integer (percentage) describing the proportion of a pastry use (fuzzy trait) of the fruit

Use.jam an integer (percentage) describing the proportion of a jam use (fuzzy trait) of the fruit

See Also

fruits_traits_cat, baskets_fruits_weights

Examples

```
## Not run:
# Load Species x Traits Data Frame
data("fruits_traits", package = "mFD")
fruits_traits

# Load Traits Information
data("fruits_traits_cat", package = "mFD")
fruits_traits_cat

# Summarize Species x Traits Data
mFD::sp.tr.summary(tr_cat = fruits_traits_cat, sp_tr = fruits_traits)

## End(Not run)
```

---

fruits_traits_cat *Dataset: Fruits Traits Informations*

Description

This dataset summarizes information about the 6 traits used in the fruits_traits dataset.

Usage

```
fruits_traits_cat
```

Format

A data frame with 8 rows (traits) and the following three columns:

trait_name a character giving the trait name

trait_type a character giving the trait type (**O** for Ordinal trait, **N** for Nominal trait, **Q** for Quantitative trait, and **F** for Fuzzy-coded trait)

fuzzy_name a character giving the name of fuzzy-coded traits (i.e. Use) to which 'sub-traits' (i.e. raw, pastry, and jam) belongs

Note

If your dataset does not contain fuzzy trait, the column fuzzy_name can be ignored but the first two columns are mandatory.

Traits in this dataset correspond to columns (traits) of the fruits_traits dataset.

See Also

fruits_traits, baskets_fruits_weights

Examples

```
## Not run:
# Load Traits Information
data("fruits_traits_cat", package = "mFD")
fruits_traits_cat

# Load Species x Traits Data Frame
data("fruits_traits", package = "mFD")

# Summarize Species x Traits Data
mFD::sp.tr.summary(tr_cat = fruits_traits_cat, sp_tr = fruits_traits)

## End(Not run)
```

---

fspe.plot *Plot FSpe*

Description

This function plots FSpe index for a given pair of functional axes and for one or several assemblages. It adds the mean position of species from the global pool and the distance of each species from the studied assemblage(s) on the background plot

Usage

```
fspe.plot(
  ggplot_bg,
  asb_sp_coord2D,
  asb_sp_relatw,
  center_coord2D,
  pool_coord2D,
  plot_pool = TRUE,
  plot_sp = TRUE,
  shape_pool,
  size_pool,
  color_pool,
  fill_pool,
  shape_sp,
  color_sp,
  fill_sp,
  color_center,
  fill_center,
  shape_center,
  size_center,
```

```
  color_segment,
  width_segment,
  linetype_segment
)
```

Arguments

ggplot_bg	a ggplot object of the plot background retrieved through the background.plot function.
asb_sp_coord2D	a list of matrix (ncol = 2) with coordinates of species present in each assemblage for a given pair of functional axes.
asb_sp_relatw	a list of vector gathering species relative weight in each assemblage. It can be retrieved through the alpha.fd.multidim.
center_coord2D	a list containing the coordinates of the center of the global pool for two given functional axes
pool_coord2D	a list of matrix (ncol = 2) with coordinates of species present in the global pool for a given pair of functional axes.
plot_pool	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_pool = TRUE.
plot_sp	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_sp = TRUE
shape_pool	a numeric value referring to the shape used to plot species pool.
size_pool	a numeric value referring to the size of species belonging to the global pool.
color_pool	a R color name or an hexadecimal code referring to the color of the pool. This color is also used for FRic convex hull color. Default: color_pool = "#0072B2".
fill_pool	a R color name or an hexadecimal code referring to the colour to fill species symbol (if shape_sp > 20) and the assemblage convex hull.
shape_sp	a numeric value referring to the shape of the symbol used for species plotting if one assemblage to plot or a vector numeric values if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRshape", asb2 = "secondRshape", ...).
color_sp	a R color name or an hexadecimal code referring to the color of species if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
fill_sp	a R color name or an hexadecimal code referring to the color of species symbol filling (if shape_sp > 20) if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).
color_center	a R color name or an hexadecimal code referring to the color of the center of the global pool.
fill_center	a R color name or an hexadecimal code referring to the colour to fill the center of the global pool (if shape_sp > 20).

shape_center — a numeric value referring to the shape used to plot the center of the global pool.

size_center — a numeric value referring to the size of the center of the global pool.

color_segment — a R color name or an hexadecimal code referring to the color of the segments linking each species of a given assemblage to the center of functional space if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRcolorname", asb2 = "secondRcolorname", ...).

width_segment — a numeric value referring to the width of the segments linking each species of a given assemblage to the center of functional space if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRwidth", asb2 = "secondRwidth", ...).

linetype_segment — a character string referring to the linetype of the segment linking each species of a given assemblage to the center of functional space if one assemblage to plot or a vector of R color names or hexadecimal codes if several assemblages to plot. If more than one assemblage to plot, the vector should be formatted as: c(asb1 = "firstRlinetype", asb2 = "secondRlinetype", ...).

Value

A ggplot object with FSpe index on the background plot.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
## Not run:
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
```

```
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits +
 c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                              range_faxes = range_faxes_lim,
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")

 # Retrieve the matrix of species coordinates for "basket_1" and PC1 and PC2
 sp_filter <- mFD::sp.filter(asb_nm          = "basket_1",
                             sp_faxes_coord = sp_faxes_coord_fruits,
                             asb_sp_w       = baskets_fruits_weights)
 fruits_asb_sp_coord_b1 <- sp_filter$`species coordinates`
 fruits_asb_sp_coord2D_b1 <- fruits_asb_sp_coord_b1[, c("PC1", "PC2")]

 # Use alpha.fd.multidim() function to get inputs to plot FIde:
 alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
  sp_faxes_coord   = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  asb_sp_w         = baskets_fruits_weights,
  ind_vect         = c("fspe"),
  scaling          = TRUE,
  check_input      = TRUE,
  details_returned = TRUE)

 # Retrieve nearest neighbor(s) names through alpha.fd.multidim outputs:
 fruits_center_coord2D <-
        alpha_fd_indices_fruits$details$pool_O_coord[c("PC1", "PC2")]


 # Retrieve FNND plot:
 fspe_plot <- fspe.plot(ggplot_bg = ggplot_bg_fruits,
           asb_sp_coord2D = list(basket_1 = fruits_asb_sp_coord2D_b1),
           asb_sp_relatw = list(basket_1 = fruits_asb_sp_relatw_b1),
           center_coord2D = fruits_center_coord2D,
           pool_coord2D = sp_faxes_coord_fruits[, c("PC1", "PC2")],
           plot_pool = TRUE,
           plot_sp = TRUE,
           shape_sp = 16,
           color_sp = "red",
           fill_sp = "red",
```

```
            shape_pool = 4,
            size_pool = 2,
            color_pool = "grey",
            fill_pool = "grey",
            color_center = "blue",
            fill_center = "blue",
            shape_center = 18,
            size_center = 3,
            color_segment = list(basket_1 = "red"),
            width_segment = list(basket_1 = 1),
            linetype_segment = list(basket_1 = "solid"))

 fspe_plot

## End(Not run)
```

`funct.dist` *Compute functional distance between species*

Description

For a given combination of traits, this function returns the functional distance matrix between species.

Usage

```
funct.dist(
  sp_tr,
  tr_cat,
  metric,
  scale_euclid = "scale_center",
  ordinal_var = "classic",
  weight_type = "equal",
  stop_if_NA = TRUE
)
```

Arguments

`sp_tr`	a data frame of traits values (columns) for each species (rows).
`tr_cat`	a data frame containing three columns for each trait (rows): • **trait_name**: the name of all traits as in `sp_tr` data frame; • **trait_type**: the category code for each trait as followed: `N` for Nominal traits (factor variable), `O` for Ordinal traits (ordered variable), `C` for Circular traits (integer values), `Q` for quantitative traits (numeric values) that is allowed **only** if there are at least 2 species with the same value, and `F` for fuzzy traits (i.e. described with several values defined with several column); • **fuzzy_name**: name of fuzzy-coded trait to which 'sub-trait' belongs (if trait is not fuzzy, ignored so could be trait name or NA).

- **trait_weight**: Optional, a numeric vector of length n (traits number) to specify a weight for each trait.

`metric` the distance to be computed: `euclidean`, the Euclidean distance, `gower`, the Classical Gower distance as defined by Gower (1971), extent by de Bello *et al.* (2021) and based on the `gawdis` function.

`scale_euclid` only when computing euclidean distance a string value to compute (or not) scaling of quantitative traits using the `tr.cont.scale` function. Possible options are: `range` (standardize by the range: $(x' = x - min(x))/(max(x) - min(x))$) `center` (use the center transformation: $x' = x - mean(x)$), `scale` (use the scale transformation: $x' = \frac{x}{sd(x)}$), `scale_center` (use the scale-center transformation: $x' = \frac{x - mean(x)}{sd(x)}$), or `noscale` traits are not scaled Default is `scale_center`.

`ordinal_var` a character string specifying the method to be used for ordinal variables (i.e. ordered). `classic` simply treats ordinal variables as continuous variables; `metric` refers to Eq. 3 of Podani (1999); `podani` refers to Eqs. 2a-b of Podani (1999), Both options convert ordinal variables to ranks. Default is `classic`.

`weight_type` the type of used method to weight traits. `user` user defined weights in tr_cat, `equal` all traits having the same weight. More methods are available using `gawdis` from `gawdis` package. To compute gower distance with fuzzy trait and weight please refer to `gawdis`. Default is `equal`.

`stop_if_NA` a logical value to stop or not the process if the `sp_tr` data frame contains NA. Functional measures are sensitive to missing traits. For further explanations, see the Note section. Default is `TRUE`.

Value

a `dist` object containing distance between each pair of species.

Note

If the `sp_tr` data frame contains `NA` you can either chose to compute anyway functional distances (but keep in mind that **Functional measures are sensitive to missing traits!**) or you can delete species with missing or extrapolate missing traits (see Johnson *et al.* (2020)).

Author(s)

Nicolas Loiseau and Sebastien Villeger

References

de Bello *et al.* (2021) Towards a more balanced combination of multiple traits when computing functional differences between species. *Method in Ecology and Evolution*, **12**, 443-448.
Gower (1971) A general coefficient of similarity and some of its properties. *Biometrics*, **27**, 857-871.
Johnson *et al.* (2020) Handling missing values in trait data. *Global Ecology and Biogeography*, **30**, 51-62.
Podani (1999) Extending Gower's general coefficient of similarity to ordinal characters, *Taxon*, **48**, 331-340.

Examples

```
# Load Species x Traits data
data("fruits_traits", package = "mFD")

# Load Traits x Categories data
data("fruits_traits_cat", package = "mFD")

# Remove fuzzy traits for this example and thus remove lat column:
fruits_traits     <- fruits_traits[ , -c(6:8)]
fruits_traits_cat <- fruits_traits_cat[-c(6:8), ]
fruits_traits_cat <- fruits_traits_cat[ , -3]

# Compute Functional Distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)
sp_dist_fruits
```

`funct.space.plot` *Plot species position in a functional space*

Description

This function illustrates the position of species along pairs of axes of a functional space

Usage

```
funct.space.plot(
  sp_faxes_coord,
  faxes = NULL,
  name_file = NULL,
  faxes_nm = NULL,
  range_faxes = c(NA, NA),
  color_bg = "grey95",
  color_pool = "darkturquoise",
  fill_pool = "white",
  shape_pool = 21,
  size_pool = 1,
  plot_ch = TRUE,
  color_ch = "darkblue",
  fill_ch = "white",
  alpha_ch = 1,
  plot_vertices = TRUE,
  color_vert = "darkturquoise",
```

```
  fill_vert = "darkturquoise",
  shape_vert = 22,
  size_vert = 1,
  plot_sp_nm = NULL,
  nm_size = 3,
  nm_color = "black",
  nm_fontface = "plain",
  check_input = TRUE
)
```

Arguments

sp_faxes_coord	a matrix of species coordinates in a multidimensional functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
faxes	a vector with names of axes to plot (as columns names in sp_faxes_coord). **You can only plot from 2 to 4 axes for graphical reasons**. Default: faxes = NULL (the four first axes will be plotted).
name_file	a character string with name of file to save the figure (without extension). Default: name_file = NULL which means plot is displayed.
faxes_nm	a vector with axes labels for figure. Default: as faxes).
range_faxes	a vector with minimum and maximum values of axes used for all plots to have a fair representation of position of species. Default: range_faxes = c(NA, NA) (the range is computed according to the range of values among all axes). If at least one of the value provided is within the range of coordinates, then convex hull could not be plotted so plot_ch should be FALSE.
color_bg	a R color name or an hexadecimal code used to fill plot background. Default: color_bg = "grey95".
color_pool	a R color name or an hexadecimal code referring to the color of symbol for species. Default: color_pool = 'darkgreen'.
fill_pool	a R color name or an hexadecimal code referring to the color to fill species symbol (if shape_pool >20). Default: fill_pool = 'white'.
shape_pool	a numeric value referring to the shape of symbol used for species. Default: shape_pool = 21 (filled circle).
size_pool	a numeric value referring to the size of symbol for species. Default: size_pool = 1.
plot_ch	a logical value indicating whether the convex hull shaping the pool of species should be illustrated. If plot_ch = TRUE, convex hull of all species in the multidimensional space described in sp_faxes_coord is computed and its projection in 2D spaces are drawn as polygons. Default: plot_ch = TRUE.
color_ch	a R color name or an hexadecimal code referring to the border of the convex hull filled by the pool of species. Default: color_ch = "darkblue".
fill_ch	a R color name or an hexadecimal code referring to the filling of the convex hull filled by the pool of species. Default: fill_ch = "white".
alpha_ch	a numeric value for transparency of the filling of the convex hull (0 = high transparency, 1 = no transparency). Default: alpha_ch = 1.

plot_vertices	a logical value defining whether vertices of the convex hull shaping the pool of species should be illustrated. If plot_vertices = TRUE, vertices of convex hull computed in the multidimensional space from sp_faxes_coord and are plotted with aesthetics listed below '..._vert' (species not being vertices are plotted with aesthetics described above for '.._sp'. Default: plot_vertices = TRUE.
color_vert	a character value referring to the color of symbol for vertices if plot_vertices = TRUE. Default: color_vert = 'darkturquoise'.
fill_vert	a character value referring to the color for filling symbol for vertices (if shape_vert >20). Default: fill_vert = 'darkturquoise'.
shape_vert	a numeric value referring to the symbol used to show vertices position if plot_vertices = TRUE. Default: shape_vert = 23 (filled diamond).
size_vert	a numeric value referring to the size of symbol for vertices Default: size_vert = 1.
plot_sp_nm	a vector containing species names that are to be printed near their position. Default: plot_nm_sp = NULL (no name plotted).
nm_size	a numeric value for size of species label. Default is 3 (in points).
nm_color	a R color name or an hexadecimal code referring to the color of species label. Default: nm_color = 'black'.
nm_fontface	a character string for font of species labels (e.g. "italic", "bold"). Default: nm_fontface = 'plain'.
check_input	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: check_input = TRUE.

Value

If name_file is NULL, a list containing ggplot2 objects is returned: plots of functional space along all pairs of axes (named according to axes names, e.g. "PC1_PC2"), figure 'caption', and the full figure 'patchwork' built using the library patchwork. If name_file is not NULL a 300dpi png file is saved in the working directory. Ranges of axes are the same for all panels and if required projection of the convex hull computed in the multidimensional space provided as input sp_faxes_coord is illustrated with a polygon. Species being vertices of this convex hull are shown with aesthetics provided as inputs ..._vert. Labels for species listed in plot_sp_nm are added with if required arrows using ggrepel. Summary about species and dimensionality are printed on top-right corner of the figure.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")
```

```
# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
                                   ordinal_var   = "classic",
                                   weight_type   = "equal",
                                   stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
  sp_dist             = sp_dist_fruits,
  maxdim_pcoa         = 10,
  deviation_weighting = "absolute",
  fdist_scaling       = FALSE,
  fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Plot functional spaces:
mFD::funct.space.plot(
 sp_faxes_coord    = sp_faxes_coord_fruits[, c("PC1", "PC2", "PC3", "PC4")],
  faxes             = NULL,
  name_file         = NULL,
  faxes_nm          = NULL,
  range_faxes       = c(NA, NA),
  color_bg          = "grey95",
  color_pool          = "darkturquoise",
  fill_pool           = "white",
  shape_pool          = 21,
  size_pool           = 1,
  plot_ch           = TRUE,
  color_ch          = "darkblue",
  fill_ch           = "white",
  alpha_ch          = 1,
  plot_vertices     = TRUE,
  color_vert        = "darkturquoise",
  fill_vert         = "darkturquoise",
  shape_vert        = 22,
  size_vert         = 1,
 plot_sp_nm         = NULL,
 nm_size            = 3,
 nm_color           = "black",
 nm_fontface        = "plain",
 check_input        = TRUE)
```

`fuse` *Compute FUSE (Functionally Unique, Specialized and Endangered)*

Description

This index takes into account species functional uniqueness (also called Functional Originality), species specialisation and species IUCN status.

Usage

```
fuse(sp_dist, sp_faxes_coord, nb_NN = 5, GE, standGE = FALSE)
```

Arguments

`sp_dist`	a dist object provided by `funct.dist`, `daisy` or `dist.ktab`.
`sp_faxes_coord`	a data frame with the coordinates of the species on a multidimensional space based on a selected number of axes derived from a Principal Coordinate Analysis (PCoA). The species are in rows and the PCOA axes are in column.
`nb_NN`	a numerical value giving the number of nearest neighbor to consider. Default: `nb_NN = 5`.
`GE`	a numerical vector giving the IUCN status rank (DD = NA, LC = 0, NT = 1, VU = 2, EN = 3, CR = 4) or the IUCN extinction probability associated with each status. See Mooers *et al.* (2008) for further information. For example, DD = NA, LC = 0, NT = 0.1, VU = 0.4, EN = 0.666, and CR = 0.999).
`standGE`	a logical value to standardize the GE values.

Value

A data frame with species in rows and the following metrics in columns:

- FUSE: functionally unique, specialized and endangered (see Pimiento *et al.* (2020);
- FUn_std: functional uniqueness standardized between 0 and 1 (see Mouillot *et al.* (2013);
- FSp_std: functional specialization standardized between 0 and 1 (see Mouillot *et al.* (2013);

Author(s)

Fabien Leprieur and Camille Albouy

References

Mouillot *et al.* (2013) Rare species support vulnerable functions in high-diversity ecosystems. *PLoS Biology*, **11**, e1001569.
Pimiento *et al.* (2020) Functional diversity of marine megafauna in the Anthropocene. *Science Advances*, **6**, eaay7650.
Violle *et al.* (2007) Let the concept of trait be functional! *Oikos*, **116**, 882-892.

Examples

```
# Load species traits data:
sp_tr <- read.csv(system.file('extdata', 'data_traits_MMA_ursus.csv',
  package = 'mFD'), dec = ',', sep = ';', header = TRUE, row.names = 1,
  na.strings='NA')

# Trait compilation and ordination:
dimorphism      <- ordered(sp_tr$dimorphism)
breeding_site   <- ordered(sp_tr$breeding_site)
social_behavior <- ordered(sp_tr$social_behavior)
weight_max      <- log(sp_tr$adult_weight_max)
social_group    <- log(sp_tr$social_group_mean)

# Trait Matrix construction:
sp_tr_end <- data.frame(
  main_diet = sp_tr$main_diet,
  foraging_water_depth = sp_tr$foraging_water_depth,
  foraging_location = sp_tr$foraging_location,
  fasting_strategy = sp_tr$fasting_strategy,
  female_sexual_maturity = sp_tr$female_sexual_maturity,
  weaning = sp_tr$weaning,
  gestation = sp_tr$gestation, inter_litter = sp_tr$inter_litter,
  breeding_site = sp_tr$breeding_site,
  social_group = sp_tr$social_group_mean,
  social_behavior = sp_tr$social_behavior,
  weight_max = sp_tr$adult_weight_max,
  dimorphism = sp_tr$dimorphism)

rownames(sp_tr_end) <- rownames(sp_tr)

# Function weigthing vector:
v <- c(0.25, 0.25, 0.25, 0.25, 0.20, 0.20, 0.20, 0.20, 0.20, 0.5, 0.5, 0.5,
  0.5)

# Gower distance calculation:
sp_tr_end$main_diet <- as.factor(sp_tr_end$main_diet)
sp_tr_end$foraging_water_depth <- as.factor(sp_tr_end$foraging_water_depth)
sp_tr_end$foraging_location <- as.factor(sp_tr_end$foraging_location)
sp_tr_end$breeding_site <- as.factor(sp_tr_end$breeding_site)
sp_tr_end$social_behavior <- as.factor(sp_tr_end$social_behavior)

sp_dist_tr <- cluster::daisy(sp_tr_end, metric = c('gower'),
  type = list(symm = c(4)), weights = v)

# Principal coordinate analyses
Pcoa <- ade4::dudi.pco(ade4::quasieuclid(sp_dist_tr), scann = FALSE,
                       nf = 40)

sp_faxes_coord <- Pcoa$li[1:40]

# FUSE calculation:
 FUSE_res <- mFD::fuse(
```

```
   sp_dist        = sp_dist_tr,
   sp_faxes_coord = as.matrix(sp_faxes_coord),
   nb_NN          = 5,
   GE             = sp_tr$IUCN_num,
   standGE        = TRUE)
FUSE_res

FUSE_res2 <- mFD::fuse(
   sp_dist        = sp_dist_tr,
   sp_faxes_coord = as.matrix(sp_faxes_coord),
   nb_NN          = 5,
   GE             = sp_tr$IUCN_50,
   standGE        = TRUE)
FUSE_res2

FUSE_res3 <- mFD::fuse(
   sp_dist        = sp_dist_tr,
   sp_faxes_coord = as.matrix(sp_faxes_coord),
   nb_NN          = 5,
   GE             = sp_tr$IUCN_100,
   standGE        = TRUE)
FUSE_res3
```

`mst.computation` *Compute the Minimum Spanning Tree (MST) linking species of a given assemblage*

Description

This function computes the MST linking species of a given assemblage and is used to compute FEve index.

Usage

```
mst.computation(sp_faxes_coord_k)
```

Arguments

`sp_faxes_coord_k`
: a matrix relating species coordinates for species present in a given assemblage.

Value

A dist object summarizing the MST for all species of a given assemblage `mst_asb_k`.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
                                   ordinal_var   = "classic",
                                   weight_type   = "equal",
                                   stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
                                  sp_dist             = sp_dist_fruits,
                                  maxdim_pcoa         = 10,
                                  deviation_weighting = "absolute",
                                  fdist_scaling       = FALSE,
                                  fdendro             = "average")

# Retrieve species coordinates matrix:
 sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute the distance of "pear" to its nearest neighbor(s):
 mst_fruits <- mst.computation(sp_faxes_coord_fruits)
 mst_fruits
```

`panels.to.patchwork` *Plot individual plots along a pair of functional axes into a unique graph*

Description

This function gathers panels into a unique `patchwork` graph with caption.

Usage

```
panels.to.patchwork(panels, plot_caption)
```

Arguments

panels	a list of ggplot objects illustrating an index on two given functional axes. There must be either one, three or six ggplot objects given the number of studied functional axes.
plot_caption	a ggplot object illustrating the caption of the final patchwork plot.

Value

A unique ggplot object gathering functional panels and caption.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
## Retrieve FRic plot:
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
  sp_faxes_coord_fruits <-
    fspaces_quality_fruits$details_fspaces$sp_pc_coord

 # Retrieve species coordinates matrix for the assemblage "basket_1":
  sp_filter <- mFD::sp.filter(asb_nm          = c("basket_1"),
                              sp_faxes_coord = sp_faxes_coord_fruits,
                              asb_sp_w       = baskets_fruits_weights)
  sp_faxes_coord_fruits_b1 <- sp_filter$`species coordinates`
```

```
 # Reduce it to the two studed axes: PC1 and PC2:
 sp_faxes_coord_fruits_b1_2D <- sp_faxes_coord_fruits_b1[, c("PC1", "PC2")]

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits)
 range_faxes_lim <- range_sp_coord_fruits + c(-1, 1)*(range_sp_coord_fruits[2] -
 range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                              range_faxes = range_faxes_lim,
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")

 # Retrieve vertices names:
 vert_nm_fruits <- vertices(sp_faxes_coord_fruits_b1_2D,
  order_2D = TRUE, check_input = TRUE)

 # Plot in white the convex hull of all fruits species:
 ggplot_fric <- mFD::fric.plot(
           ggplot_bg       = ggplot_bg_fruits,
           asb_sp_coord2D  = list(basket_1 = sp_faxes_coord_fruits_b1_2D),
           asb_vertices_nD = list(basket_1 = vert_nm_fruits),
           plot_sp         = TRUE,
           color_ch        = c("basket_1" = "black"),
           fill_ch         = c("basket_1" = "white"),
           alpha_ch        = c("basket_1" = 0.3),
           size_sp = c("basket_1" = 1),
           shape_sp = c("basket_1" = 16),
           color_sp = c("basket_1" = "red"),
           fill_sp = c("basket_1" = "red"),
           size_vert = c("basket_1" = 1),
           color_vert = c("basket_1" = "red"),
           fill_vert = c("basket_1" = "red"),
           shape_vert = c("basket_1" = 16))
 ggplot_fric

## Create a caption summing up FRic values
# retrieve values to plot:
top_fric <- c("Functional richness", "basket_1", "")

asb_fd_ind <- alpha_fd_indices_fruits <- mFD::alpha.fd.multidim(
sp_faxes_coord   = sp_faxes_coord_fruits[ , c("PC1", "PC2", "PC3", "PC4")],
asb_sp_w         = baskets_fruits_weights,
ind_vect         = c("fric"),
scaling          = TRUE,
check_input      = TRUE,
details_returned = TRUE)

values_fric <- c(round(asb_fd_ind$functional_diversity_indices["basket_1",
"fric"], 3), "")
```

```
# customize position of texts in the plot:
spread_faxes <- (range_sp_coord_fruits[2] - range_sp_coord_fruits[1])
hh <- c(1, 2.5, 4, 5.5)
vv <- 0.3

# plot window:
x <- NULL
y <- NULL
plot_caption <- ggplot2::ggplot(data.frame(x = range_sp_coord_fruits,
                                           y = range_sp_coord_fruits),
                                ggplot2::aes(x = x, y = y)) +
 ggplot2::scale_x_continuous(limits = range_sp_coord_fruits,
 expand = c(0, 0)) +
 ggplot2::scale_y_continuous(limits = range_sp_coord_fruits,
 expand = c(0, 0)) +
 ggplot2::theme_void() + ggplot2::theme(legend.position = "none") +
 ggplot2::geom_rect(xmin = range_sp_coord_fruits[1],
                    xmax = range_sp_coord_fruits[2],
                    ymin = range_sp_coord_fruits[1],
                    ymax = range_sp_coord_fruits[2],
                    fill = "white", colour ="black")

# plot names of index and of assemblages:
h   <- NULL
v   <- NULL
top <- NULL
x <- NULL
y <- NULL
plot_caption <- plot_caption +
 ggplot2::geom_text(data = data.frame(
   h = range_sp_coord_fruits[1] + spread_faxes * 0.15 * hh[c(1,3:4)],
   v = range_sp_coord_fruits[2] - spread_faxes * rep(0.2, 3),
   top = top_fric),
   ggplot2::aes(x = h, y = v, label = top),
   size = 3, hjust = 0.5, fontface = "bold")

# plot FRic values:
values_lab <- NULL
data_caption <- data.frame(
  h = range_sp_coord_fruits[1] + spread_faxes * 0.15 * hh[2:4],
  v = range_sp_coord_fruits[2] - spread_faxes*rep(vv, 3),
  values_lab = c("FRic", values_fric))
plot_caption <- plot_caption +
  ggplot2::geom_text(data = data_caption,
                    ggplot2::aes(x = h, y = v, label = values_lab),
                    size = 3, hjust = 0.5, fontface = "plain")

## Create patchwork:
patchwork_fric <- mFD::panels.to.patchwork(list(ggplot_fric), plot_caption)
patchwork_fric
```

`pool.plot` *Plot species from the pool*

Description

Plot all species from the study case and associated convex hull

Usage

```
pool.plot(
  ggplot_bg,
  sp_coord2D,
  vertices_nD,
  plot_pool = TRUE,
  shape_pool = 3,
  size_pool = 0.8,
  color_pool = "grey95",
  fill_pool = NA,
  color_ch = NA,
  fill_ch = "white",
  alpha_ch = 1,
  shape_vert = 3,
  size_vert = 1,
  color_vert = "black",
  fill_vert = NA
)
```

Arguments

`ggplot_bg`	a ggplot object of the plot background retrieved through the `background.plot` function.
`sp_coord2D`	a list of matrix (ncol = 2) with coordinates of species present in the pool for a given pair of axes
`vertices_nD`	a list (with names as in sp_coord2D) of vectors with names of species being vertices in n dimensions.
`plot_pool`	a logical value indicating whether species of each assemblage should be plotted or not. Default: plot_pool = TRUE.
`shape_pool`	a numeric value referring to the shape used to plot species pool. Default: `shape_pool = 16`(filled circle).
`size_pool`	a numeric value referring to the size of species belonging to the global pool. Default: `size_pool = 1`.
`color_pool`	a R color name or an hexadecimal code referring to the color of the pool. This color is also used for FRic convex hull color. Default: `color_pool = "#0072B2"`.

fill_pool	a R color name or an hexadecimal code referring to the colour to fill species symbol (if shape_sp > 20) and the assemblage convex hull. Default: fill_pool = '#0072B2'.
color_ch	a R color name or an hexadecimal code referring to the border of the convex hull filled by the pool of species. Default: color_ch = "black".
fill_ch	a R color name or an hexadecimal code referring to the filling of the convex hull filled by the pool of species. Default is: fill_ch = "white".
alpha_ch	a numeric value for transparency of the filling of the convex hull (0 = high transparency, 1 = no transparency). Default: alpha_ch = 0.3.
shape_vert	a numeric value referring to the shape used to plot vertices if vertices should be plotted in a different way than other species. If shape_vert = NA, no vertices plotted. Default: shape_vert = NA.
size_vert	a numeric value referring to the size of symbol for vertices. Default: size_vert = 1.
color_vert	a R color name or an hexadecimal code referring to the color of vertices if plotted. If color_vert = NA, vertices are not plotted (for shapes only defined by color, ie shape inferior to 20. Otherwise fill must also be set to NA). Default: color_vert = NA.
fill_vert	a character value referring to the color for filling symbol for vertices (if shape_vert >20). If fill = NA and color = NA, vertices are not plotted (if shape_vert superior to 20. Otherwise color_vert = NULL is enough). Default is NA.

Value

A ggplot object plotting background of multidimensional graphs and species from the global pool (associated convex hull if asked).

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
```

```
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(sp_dist = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits_2D <-
  fspaces_quality_fruits$details_fspaces$sp_pc_coord[ , c("PC1", "PC2")]

# Set faxes limits:
# set range of axes if c(NA, NA):
 range_sp_coord_fruits  <- range(sp_faxes_coord_fruits_2D)
 range_faxes_lim <- range_sp_coord_fruits +
   c(-1, 1)*(range_sp_coord_fruits[2] -
   range_sp_coord_fruits[1]) * 0.05

 # Retrieve the background plot:
 ggplot_bg_fruits <- mFD::background.plot(
                              range_faxes = range_faxes_lim,
                              faxes_nm    = c("PC 1", "PC 2"),
                              color_bg    = "grey90")

 # Retrieve vertices names:
 vert_nm_fruits <- vertices(sp_faxes_coord_fruits_2D,
  order_2D = TRUE, check_input = TRUE)

 # Plot the pool:
 plot_pool_fruits <- pool.plot(ggplot_bg   = ggplot_bg_fruits,
           sp_coord2D    = sp_faxes_coord_fruits_2D,
           vertices_nD   = vert_nm_fruits,
           plot_pool     = TRUE,
           shape_pool    = 3,
           size_pool     = 0.8,
           color_pool    = "grey95",
           fill_pool     = NA,
           color_ch      = NA,
           fill_ch       = "white",
           alpha_ch      = 1,
           shape_vert    = 3,
           size_vert     = 1,
           color_vert    = "black",
           fill_vert     = NA)
 plot_pool_fruits
```

`quality.fspaces` *Compute functional spaces and their quality*

Description

Compute a Principal Coordinates Analysis (PCoA) using functional distance between species. Then the function evaluates the quality of spaces built using an increasing number of principal components. Quality is evaluated as the (absolute or squared) deviation between trait-based distance (input) and distance in the PCoA-based space (raw Euclidean distance or scaled distance according to its maximum value and maximum of trait-based distance). Option to compute a functional dendrogram and its quality. This function is based on the framework presented in Maire *et al.* (2015).

Usage

```
quality.fspaces(
  sp_dist,
  fdendro = NULL,
  maxdim_pcoa = 10,
  deviation_weighting = "absolute",
  fdist_scaling = FALSE
)
```

Arguments

`sp_dist`	a dist object with pairwise distance among all species (at least 3 species needed). Functional distance matrix from trait values can be computed using `funct.dist` function.
`fdendro`	a character string indicating the clustering algorithm to use to compute dendrogram. Should be one of the method recognized by `hclust` (e.g. 'average' for UPGMA). Default: `fdendro = NULL` (so no dendrogram computed).
`maxdim_pcoa`	a single numeric value with maximum number of PCoA axes to consider to build multidimensional functional spaces. Default: `maxdim_pcoa = 10`. See below about number of axes actually considered.
`deviation_weighting`	a character string referring to the method(s) used to weight the differences between species pairwise distance in the functional space and trait-based distance. `'absolute'` (default) means absolute differences are used to compute mean absolute deviation *mad* index; `'squared'` means squared differences are used to compute root of mean squared deviation *rmsd* index. Both values could be provided to compare quality metrics.
`fdist_scaling`	a vector with logical value(s) specifying whether distances in the functional space should be scaled before computing differences with trait-based distances. Scaling ensures that trait-based distances and distances in the functional space have the same maximum. Default: `FALSE`. Both values could be provided to compare quality metrics.

Value

A list with:

- `$quality_fspaces`: a data frame with quality metric(s) for each functional space. Functional spaces are named as 'pcoa_.d' and if required 'tree_clustering method'. Quality metrics are

named after deviation_weighting ('mad' for 'absolute' and and 'rmsd' for 'squared') and if fdist_scaling is TRUE with suffix '_scaled'.

- `$details_trdist` a list with 2 elements: `$trdist_summary` a vector with minimum (min), maximum (max), mean (mean) and standard deviation (sd) of `sp_dist`; `$trdist_euclidean` a logical value indicating whether `sp_dist` checks Euclidean properties.
- `$details_fspaces` a list with 4 elements: `$sp_pc_coord` a matrix with coordinates of species (rows) along Principal Components (columns) with positive eigenvalues ; `$pc_eigenvalues` a matrix with eigenvalues of axes from PCoA ; `$dendro` a hclust object with the dendrogram details (null if no dendrogram computed) ; `$pairsp_fspaces_dist` a dataframe containing for each pair of species (rows), their names in the 2 first columns ('sp.x' and 'sp.y'), their distance based on trait-values ('tr'), and their Euclidean (for PCoA) or cophenetic (for dendrogram if computed) distance in each of the functional space computed ('pcoa_1d', 'pcoa_2d', ... , 'tree_clust'); if `fdist_scaling = TRUE`, `$pairsp_fspaces_dist_scaled` a data frame with scaled values of distances in functional spaces.
- `$details_deviation` a list of data frames: `$dev_distsp` a dataframe containing for each space (columns) the difference for all species pairs (rows) of the distance in the functional space and trait-based distance (i.e. positive deviation indicates overestimation of actual distance) ; `$abs_dev_distsp` and/or `$sqr_dev_distsp`, dataframes with for each space (columns) and all species pairs (rows) the absolute or squared deviation of distance ; if `fdist_scaling = TRUE` `$dev_distsp_scaled`, and `$abs_dev_distsp_scaled` and/or `$sqr_dev_distsp_scaled`, data frames with deviation computed on scaled distance in functional spaces.

Note

The maximum number of dimensions considered for assessing quality of functional spaces depends on number of PC axes with positive eigenvalues (i.e. axes with negative eigenvalues are not considered); so it could be lower than `$maxdim_pcoa`. The quality metric obtained with deviation_weighting = 'squared' and `fdist_scaling = TRUE` is equivalent to the square-root of the 'mSD' originally suggested in Maire *et al.* (2015).

Author(s)

Sebastien Villeger, Eva Maire, and Camille Magneville

References

Maire *et al.* (2015) How many dimensions are needed to accurately assess functional diversity? A pragmatic approach for assessing the quality of functional spaces *Global Ecology and Biogeography*, **24**, 728-740.

Examples

```
# Load Species x Traits Data
data("fruits_traits", package = "mFD")

# Load Traits x Categories Data
data("fruits_traits_cat", package = "mFD")

# Compute Functional Distance
```

```
sp_dist_fruits <- mFD::funct.dist(
  sp_tr         = fruits_traits,
  tr_cat        = fruits_traits_cat,
  metric        = "gower",
  scale_euclid  = "scale_center",
  ordinal_var   = "classic",
  weight_type   = "equal",
  stop_if_NA    = TRUE)

# Compute Functional Spaces Quality (to retrieve species coordinates)
fspaces_quality_fruits <- mFD::quality.fspaces(
  sp_dist             = sp_dist_fruits,
  maxdim_pcoa         = 10,
  deviation_weighting = "absolute",
  fdist_scaling       = FALSE,
  fdendro             = "average")
 fspaces_quality_fruits

# Retrieve Species Coordinates
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord
sp_faxes_coord_fruits
```

`quality.fspaces.plot` *Plot functional space quality with a chosen quality metric*

Description

Plot functional space quality with a chosen quality metric

Usage

```
quality.fspaces.plot(
  fspaces_quality,
  quality_metric,
  fspaces_plot,
  name_file = NULL,
  range_dist = NULL,
  range_dev = NULL,
  range_qdev = NULL,
  gradient_deviation = c(neg = "darkblue", nul = "grey80", pos = "darkred"),
  gradient_deviation_quality = c(low = "yellow", high = "red"),
  x_lab = "Trait-based distance"
)
```

Arguments

`fspaces_quality`
: output from the `quality.fspaces` function, that is a list with all data needed to illustrate quality of functional spaces based on deviation between species trait-

based distance and distance in functional spaces built using PCoA (and dendrogram).

`quality_metric` a character string with the name of the quality metric to illustrate. Should be one of the column names of `fspaces_quality$quality_fspaces`. See help of `quality.fspaces` for the meaning of these names regarding type of deviation and scaling of distance in functional space. Default: `'mad'` (Mean absolute deviation).

`fspaces_plot` a vector with names of functional spaces to consider. Should be a subset of the row names of `quality_fspaces$quality_fspaces`. Maximum of 10 spaces allowed to keep decent plot size.

`name_file` a character string with name of file to save the figure (without extension). Default: `NULL` which means plot is displayed.

`range_dist` a vector with minimum and maximum values to display for species pairwise distances (x-axis for all panels and y-axes of top panel). Default: NULL, which means range is 0 to maximum distance among all the functional spaces to plot.

`range_dev` a vector with minimum and maximum values to display for deviation to trait-based distance (y-axis of middle panel). Default: NULL, which means range is set to range of deviation among all the functional spaces to plot.

`range_qdev` a vector with minimum and maximum values to display for deviation to trait-based distance (y-axis of bottom panel). Default:NULL, which means range is from 0 to the maximum of (transformed) deviation among all the functional spaces to plot.

`gradient_deviation`
a vector of 3 colors for illustrating raw deviation with `scale_colour_gradient2`. The first value ('neg') is for the lowest negative deviation, the second value ('nul')is for null deviation and the third value ('pos') is for the highest positive deviation. Default gradient is from darkblue to grey to red.

`gradient_deviation_quality`
2 colors (named 'low' and 'high') for illustrating transformed deviation used to compute quality metric with `scale_colour_gradient2` (default gradient is from yellow to red).

`x_lab` a character string with title to display below X axis. Default is 'Trait-based distance'.

Value

A png file (resolution 300dpi) saved in the current working directory. Quality of each functional space is illustrated with three panels : - top row shows trait-based distance between species vs. space-based distance. - middle row shows trait-based distance vs. deviation between space-based and trait-based distances - bottom row shows trait-based distance between species vs. transformed deviation used to compute the quality metric All plots have the same X axis. All plots on a given row have the same Y axis and color palette. Type of distance in functional space (Euclidean in PCoA, Cophenetic on tree) are abbreviated, as well as type of transformation of distance (scaling) and of deviation (Absolute or Squared)

Author(s)

Sebastien Villeger and Camille Magneville

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                  tr_cat        = fruits_traits_cat,
                                  metric        = "gower",
                                  scale_euclid  = "scale_center",
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(
 sp_dist             = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

 # Illustrate the quality of functional spaces:
 mFD::quality.fspaces.plot(
  fspaces_quality            = fspaces_quality_fruits,
  quality_metric             = "mad",
  fspaces_plot               = c("tree_average", "pcoa_2d", "pcoa_3d",
                                 "pcoa_4d", "pcoa_5d"),
  name_file                  = NULL,
  range_dist                 = NULL,
  range_dev                  = NULL,
  range_qdev                 = NULL,
  gradient_deviation         = c(neg = "darkblue", nul = "grey80",
                                    pos = "darkred"),
  gradient_deviation_quality = c(low ="yellow", high = "red"),
  x_lab                      = "Trait-based distance")
```

`sp.filter` *Retrieve information about species in a given assemblage*

Description

This function computes names of species present in an given assemblage, their coordinates in the functional space and their weights. It is used in the `alpha_FD_multidim` function to filter species and compute each functional indices for each community.

Usage

```
sp.filter(asb_nm, sp_faxes_coord, asb_sp_w)
```

Arguments

asb_nm	a string object referring to the name of a given community.
sp_faxes_coord	a matrix of species coordinates in a chosen functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
asb_sp_w	a matrix linking weight of species (columns) and a set of assemblages (rows).

Value

A vector containing names of species present in a given assemblage sp_name_asb_k, a matrix containing coordinates of species present in a given assemblage sp_faxes_coord_k, a matrix containing weight of species present in a given assemblage asb_sp_w_k, a matrix containing relative weight of species present in a given assemblage asb_sp_relatw_k.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
data("fruits_traits_cat", package = "mFD")

# Compute functional distance
sp_dist_fruits <- mFD::funct.dist(
 sp_tr         = fruits_traits,
 tr_cat        = fruits_traits_cat,
 metric        = "gower",
 scale_euclid  = "scale_center",
 ordinal_var   = "classic",
 weight_type   = "equal",
 stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
fspaces_quality_fruits <- mFD::quality.fspaces(
 sp_dist             = sp_dist_fruits,
 maxdim_pcoa         = 10,
 deviation_weighting = "absolute",
 fdist_scaling       = FALSE,
 fdendro             = "average")

# Retrieve species coordinates matrix:
```

```
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Filter species of basket_1 assemblage:
sp.filter(asb_nm         = "basket_1",
          sp_faxes_coord = sp_faxes_coord_fruits,
          asb_sp_w       = baskets_fruits_weights)
```

`sp.to.fe` *Compute Functional Entities composition based on a Species x Traits matrix*

Description

Compute Functional Entities composition based on a Species x Traits matrix

Usage

```
sp.to.fe(sp_tr, tr_cat, fe_nm_type = "fe_rank", check_input = TRUE)
```

Arguments

`sp_tr`	a data frame containing species as rows and traits as columns.
`tr_cat`	a data frame containing three columns for each trait (rows): • **trait_name**: names of all traits as in `sp_tr` data frame; • **trait_type**: category codes for each trait as followed: *N* for Nominal traits (factor variable), *O* for Ordinal traits (ordered variable), *C* for Circular traits (integer values), *Q* for Quantitative traits (numeric values) that is allowed **only** if there are at least 2 species with the same value, and *F* for fuzzy-coded traits (i.e. described with several 'sub-traits'); • **fuzzy_name** name of fuzzy-coded trait to which 'sub-trait' belongs (if trait is not fuzzy, ignored so could be trait name or NA).
`fe_nm_type`	a character string referring to the type of naming functional entities. Two possible values: *"fe_rank"* (FE are named after their decreasing rank in term of number of species *i.e.* fe_1 is the one gathering most species) and *"tr_val"* (FE are named after names of traits and of trait values for each FE, if possible, *see details below*). Default: `fe_nm_type = "fe_rank"`.
`check_input`	a logical value indicating whether key features the inputs are checked (e.g. class and/or mode of objects, names of rows and/or columns, missing values). If an error is detected, a detailed message is returned. Default: `check.input = TRUE`.

Details

`fe_nm_type = "tr_val"` is allowed **only** if:

- there are less than 7 traits;
- none of them is fuzzy-coded (so that names are not too long)

- all trait names and all trait values have different 2 first letters

If these 3 conditions are met, names of Functional Entities are made as a character string of up to 2 letters for trait name in upper case font then up to 2 letters for trait value in lower case font, separated by "_" between traits. Trait names are abbreviated to a single letter whenever possible. *Examples:* ("TAc2_TBxx_TCyy", "TAc3_TBff_TCyy") or ("A2_Bx_Cy", "A3_Bf_Cy")

Value

A list of objects containing:

- **fe_nm**: a vector with names of all FE (following fe_nm_type). FE are ordered according to the decreasing number of species they gather.
- **sp_fe**: a vector containing for each species the name of the FE it belongs to. FE order is done according to decreasing number of species.
- **fe_tr**: a data frame containing traits values (variables in columns) for each FE (rows). FE order is done according to decreasing number of species.
- **fe_nb_sp**: a vector with species number per FE. If all FE have only one species, a warning message is returned. FE are ordered according to the decreasing number of species they gather.
- **details_fe**: a list containing: *fe_codes* a vector containing character referring to traits values (like a barcode) with names as in `fe_nm_type` and sorted according to `fe_nb_sp`; *tr_uval* a list containing for each trait a vector of its unique values or a data frame for fuzzy-coded traits; *fuzzy_E* a list with for each fuzzy-coded trait a data frame with names of entities (E) and names of species (sp); *tr_nb_uval* a vector with number of unique values per trait (or combinations for fuzzy-coded traits); *max_nb_fe* the maximum number of FE possible given number of unique values per trait.

Author(s)

Sebastien Villeger, Nicolas Loiseau, and Camille Magneville

Examples

```
# Load species traits data:
 data("fruits_traits", package = "mFD")

# Transform species traits data:
# Only keep the first 4 traits to illustrate FEs:
 fruits_traits <- fruits_traits[ , c(1:4)]

# Load trait types data:
 data("fruits_traits_cat", package = "mFD")

# Transform the trait types data to only keep traits 1 - 4:
 fruits_traits_cat <- fruits_traits_cat[c(1:4), ]

# Gather species into FEs:
## gathering species into FEs (FEs named according to the decreasing...
## ...  number of species they gather):
 sp_FEs <- mFD::sp.to.fe(
```

```
      sp_tr      = fruits_traits,
      tr_cat     = fruits_traits_cat,
      fe_nm_type = "fe_rank")

## display FEs names:
sp_FEs$fe_nm

## display for each species the name of the FE it belongs to:
sp_FEs$sp_fe

## display trait values for each FE:
sp_FEs$fe_tr

## display the number of species per FEs:
sp_FEs$fe_nb_sp
```

`sp.tr.summary` *Summarize Species x Traits data frame*

Description

This function computes a summary data helping to choose the type of analysis you can do with your data. For this function to work, there must be no NA in your `sp_tr` data frame.

Usage

```
sp.tr.summary(tr_cat, sp_tr, stop_if_NA = TRUE)
```

Arguments

`tr_cat`	a data frame containing three columns for each trait (rows): • **trait_name**: the name of all traits as in `sp_tr` data frame; • **trait_type**: the category code for each trait as followed: `N` for Nominal traits (factor variable), `O` for Ordinal traits (ordered variable), `C` for Circular traits (integer values)(circular traits can not be used in mFD function used to compute functional distance but ok for summary function and function to group species into Functional Entities), `Q` for quantitative traits (numeric values), and `F` for fuzzy traits (i.e. described with several values defined with several column); • **fuzzy_name**: name of fuzzy-coded trait to which 'sub-trait' belongs (if trait is not fuzzy, ignored so could be trait name or NA). • **trait_weight**: weights of each traits if the user wants to specify a weight for each trait.
`sp_tr`	a data frame of traits values (columns) for each species (rows). Note that species names **must be** specified in the row names.
`stop_if_NA`	a logical value indicating whether the process should stop if there is some NA in the `sp_tr` dataframe. If you continue with functional analysis, we remind you that functional measures, are sensitive to missing traits

Value

If there is no fuzzy-coded trait, a three-elements list with:

`tr_summary_list`
: a table summarizing for each trait the number of species per modality for non-continuous trait and min, max, mean, median, and quartiles for continuous traits.

`tr_types`
: a list containing traits type.

`mod_list`
: a list containing modalities for all traits.

If there is fuzzy-coded trait, a four-elements list with:

`tr_summary_non_fuzzy_list`
: a table summarizing for each trait the number of species per modality for non-continuous trait and min, max, mean, median, and quartiles for continuous traits.

`tr_summary_fuzzy_list`
: a table summarizing for each subtrait min, max, mean, median and quartiles

`tr_types`
: a list containing traits type.

`mod_list`
: a list containing modalities for non-continuous trait.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species x Traits data
data('fruits_traits', package = 'mFD')

# Load Traits x Categories data
data('fruits_traits_cat', package = 'mFD')

# Summarize Species x Traits data
mFD::sp.tr.summary(tr_cat = fruits_traits_cat, sp_tr = fruits_traits)
```

---

tr.cont.fspace *Build a functional space based on continuous traits only*

Description

This function computes a functional space based on continuous standardized traits or continuous raw traits matrix. User can either choose to compute functional space based on PCA analysis or using one trait for one functional axis. For PCA analysis, center and scale arguments are considered `FALSE`: if you want to center, scale or standardize by any mean your data, please use `tr.cont.scale` function. Option makes it possible to compute correlation between traits.

Usage

```
tr.cont.fspace(
  sp_tr,
  pca = TRUE,
  nb_dim = 7,
  scaling = "scale_center",
  compute_corr = "pearson"
)
```

Arguments

sp_tr	a data frame of traits values (columns) for each species (rows). Note that species names **must be** specified in the row names and traits must be **continuous** (raw or standardized).
pca	a logical value. If TRUE a PCA analysis is computed, elsewhere the functional space is computed with one trait for each dimension. Default is TRUE.
nb_dim	an integer referring to the maximum number of dimensions for multidimensional functional spaces. Final number of dimensions depends on the number of positive eigenvalues obtained with the PCA. High value for nb_dim can increase computation time. Default is nb_dim = 7.
scaling	a string value to compute (or not) scaling of traits using the tr.cont.scale function. Possible options are: range (standardize by the range), center (use the center transformation: $x' = x - mean(x)$), scale (use the scale transformation: $x' = \frac{x}{sd(x)}$), scale_center (use the scale-center transformation: $x' = \frac{x - mean(x)}{sd(x)}$), or no_scale Default is scale_center.
compute_corr	a string value to compute Pearson correlation coefficients between traits (compute_corr = 'pearson'). You can choose not to compute correlation coefficient by setting compute_corr to none.

Value

A list containing a matrix with mAD and mSD values for each functional space to assess the quality of functional spaces), a matrix containing eigenvalues for each axis, the percentage of variance explained by each axis and the cumulative percentage of variance, a data frame containing species coordinates on each functional axis, list of distance matrices in the functional space (Euclidean distances based on trait values and coordinates in the functional spaces), a dist object containing initial euclidean distances based on traits and a matrix of correlation coefficients between traits (if required).

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
load(system.file('extdata', 'sp_tr_cestes_df', package = 'mFD'))
```

```
mFD::tr.cont.fspace(
    sp_tr        = sp_tr,
    pca          = TRUE,
    nb_dim       = 7,
    scaling      = 'scale_center',
    compute_corr = 'pearson')
```

`tr.cont.scale` *Scale continuous traits*

Description

This function standardizes continuous traits. It can be useful before computing functional space. You will have to choose which standardized method to use based on your data. For this function to work, there must be no NA in your `sp_tr` data frame.

Usage

```
tr.cont.scale(sp_tr, std_method = "scale_center")
```

Arguments

`sp_tr`	a data frame of traits values (columns) for each species (rows). Note that species names **must be** specified in the row names and traits must be **continuous**.
`std_method`	a character string referring to the standardization method. Possible values: `range` (standardize by the range), `center` (use the center transformation: $x' = x - mean(x)$), `scale` (use the scale transformation: $x' = \frac{x}{sd(x)}$), or `scale_center` (use the scale-center transformation: $x' = \frac{x - mean(x)}{sd(x)}$). Default is `scale_center`.

Value

A data frame of standardized trait values (columns) for each species (rows).

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
load(system.file('extdata', 'sp_tr_cestes_df', package = 'mFD'))

mFD::tr.cont.scale(sp_tr = sp_tr, std_method = 'scale_center')
```

traits.faxes.cor *Correlation between Traits and Axes*

Description

Compute relationship between all traits and all axes of the functional space. For continuous trait a linear model is computed and r2 and p-value are returned. For other types of traits, a Kruskal-Wallis test is computed and eta2 statistics is returned. Option allows to plot trait-axis relationships with scatterplot and boxplot for continuous and non-continuous traits, respectively.

Usage

```
traits.faxes.cor(
  sp_tr,
  sp_faxes_coord,
  tr_nm = NULL,
  faxes_nm = NULL,
  plot = FALSE,
  name_file = NULL,
  color_signif = "darkblue",
  color_non_signif = "gray80",
  stop_if_NA = TRUE
)
```

Arguments

sp_tr	a data frame containing species as rows and traits as columns.
sp_faxes_coord	a matrix of species coordinates in a multidimensional functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
tr_nm	a vector gathering the names of traits (as in sp_tr) to consider. If NULL all traits are considered.
faxes_nm	a vector gathering the names of PCoA axes (as in sp_faxes_coord) to consider.
plot	a logical value indicating whether plot illustrating relations between trait and axes should be drawn. **You can only plot relationships for up to 10 traits and/or 10 axes**.
name_file	the file name (without extension) to save the plot as a 300 dpi JPEG file. Default is NULL which means plot is only displayed. If plot = FALSE this argument is ignored.
color_signif	an R color name or an hexadecimal code referring to the color of points when relationships between the trait and the axis is significant. Default is darkblue.
color_non_signif	an R color name or an hexadecimal code referring to the color of points when relationships between the trait and the axis are not significant. Default is gray80.
stop_if_NA	a logical value to stop or not the process if the sp_tr data frame contains NA. Functional measures are sensitive to missing traits. For further explanations, see the Note section. Default is TRUE.

Value

1 data frame with for each combination of trait and axis (rows), the name of the test performed, and the corresponding statistics and p-value. If `plot = TRUE` a multi-panel figure with traits as columns and axes as rows is also plotted. When relationships between trait and axis is significant the points are colored, else they remain grayish.

Author(s)

Nicolas Loiseau and Sebastien Villeger

Examples

```
# Load Species x Traits Data
data("fruits_traits", package = "mFD")

# Load Traits categories dataframe
data("fruits_traits_cat", package = "mFD")

# Compute Functional Distance
sp_dist_fruits <- mFD::funct.dist(sp_tr  = fruits_traits,
 tr_cat = fruits_traits_cat,
 metric = "gower",
 scale_euclid  = "scale_center",
 ordinal_var = "classic",
 weight_type = "equal",
 stop_if_NA  = TRUE)

# Compute Functional Spaces Quality (to retrieve species coordinates)
fspaces_quality_fruits <- mFD::quality.fspaces(
  sp_dist             = sp_dist_fruits,
  maxdim_pcoa         = 10,
  deviation_weighting = "absolute",
  fdist_scaling       = FALSE,
  fdendro             = "average")

# Retrieve Species Coordinates
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute Correlation between Traits and Functional Axes
mFD::traits.faxes.cor(
  sp_tr          = fruits_traits,
  sp_faxes_coord = sp_faxes_coord_fruits,
  tr_nm          = NULL,
  faxes_nm       = NULL,
  name_file      = NULL,
  color_signif   = "darkblue",
  color_non_signif = "gray80")
```

vertices *Compute vertices of the Minimal Convex Hull shaping species from a single assemblage in a multidimensional functional space*

Description

This function identifies species that are vertices of the minimal convex hull enclosing a community in a multidimensional functional space. This function is using the convhulln function.

Usage

```
vertices(sp_faxes_coord, order_2D = FALSE, check_input = FALSE)
```

Arguments

sp_faxes_coord	a matrix of species coordinates in a chosen functional space. Species coordinates have been retrieved thanks to tr.cont.fspace or quality.fspaces.
order_2D	a logical value defining whether vertices names are reordered so that they define a convex polygon in 2D which is convenient for plotting. Default is FALSE, vertices ordered as in row names of 'sp_faxes_coord'.
check_input	a logical value defining whether inputs are checked before computation: species names must be put as row.names, there must be no NA and species number must be superior to (axes number + 1). Default: check_input = TRUE.

Value

A vector containing names of species being vertices vert_nm.

Author(s)

Camille Magneville and Sebastien Villeger

Examples

```
# Load Species*Traits dataframe:
 data("fruits_traits", package = "mFD")

# Load Assemblages*Species dataframe:
 data("baskets_fruits_weights", package = "mFD")

# Load Traits categories dataframe:
 data("fruits_traits_cat", package = "mFD")

# Compute functional distance
 sp_dist_fruits <- mFD::funct.dist(sp_tr         = fruits_traits,
                                   tr_cat        = fruits_traits_cat,
                                   metric        = "gower",
                                   scale_euclid  = "scale_center",
```

```
                                  ordinal_var   = "classic",
                                  weight_type   = "equal",
                                  stop_if_NA    = TRUE)

# Compute functional spaces quality to retrieve species coordinates matrix:
 fspaces_quality_fruits <- mFD::quality.fspaces(
                                 sp_dist             = sp_dist_fruits,
                                 maxdim_pcoa         = 10,
                                 deviation_weighting = "absolute",
                                 fdist_scaling       = FALSE,
                                 fdendro             = "average")

# Retrieve species coordinates matrix:
sp_faxes_coord_fruits <- fspaces_quality_fruits$details_fspaces$sp_pc_coord

# Compute vertices and order them clockwise:
 vert_nm <- vertices(sp_faxes_coord_fruits[ , c("PC1", "PC2")],
  order_2D = TRUE, check_input = TRUE)
 vert_nm
```

Index

∗ **datasets**
baskets_fruits_weights, 18
fruits_traits, 55
fruits_traits_cat, 56

alpha.fd.fe, 3, *5*
alpha.fd.fe.plot, *3*, 5
alpha.fd.hill, 7
alpha.fd.multidim, 9, *12*, *31*, *35*, *39*, *42*, *46*, *49*, *58*
alpha.multidim.plot, 11
asb.sp.summary, 16

background.plot, 17, *31*, *35*, *39*, *42*, *45*, *49*, *52*, *58*, *74*
baskets_fruits_weights, 18
beta.fd.hill, 18
beta.fd.multidim, 20, *23, 24*
beta.multidim.plot, 23

convhulln, *91*

daisy, *67*
dist.ktab, *67*
dist.nearneighb, 27
dist.point, 28
dist.to.df, *19*, 30
dist_long, *30*

fdis.plot, 31
fdiv.plot, 34
feve.plot, 38
fide.plot, 41
fnnd.plot, 45
fori.plot, 48
fric.plot, 51
fruits_traits, 55
fruits_traits_cat, 56
fspe.plot, 57
funct.dist, 61, *67*, *77*
funct.space.plot, 63
functional.betapart.core.pairwise, *21*, *22*
fuse, 67

gawdis, *62*

hclust, *77*

mst.computation, 69

panels.to.patchwork, 70
pool.plot, 74

quality.fspaces, *10*, *21*, *27*, *29*, *64*, 76, *79*, *80*, *82*, *89*, *91*
quality.fspaces.plot, 79

scale_colour_gradient2, *80*
sp.filter, 81
sp.to.fe, *3*, 83
sp.tr.summary, 85

tr.cont.fspace, *10*, *21*, *27*, *29*, *64*, *82*, 86, *89*, *91*
tr.cont.scale, *62*, *86, 87*, 88
traits.faxes.cor, 89

vertices, 91