Package 'goeveg'

June 13, 2025

Type Package

Title Functions for Community Data and Ordinations

Version 0.7.6

Date 2025-06-13

Maintainer Friedemann von Lampe <fvonlampe@uni-goettingen.de>

Description A collection of functions useful in (vegetation) community analyses and ordinations. Includes automatic species selection for ordination diagrams, NMDS stress/scree plots, species response curves, merging of taxa as well as calculation and sorting of synoptic tables.

License GPL (>= 2)

LazyData TRUE

Depends R (>= 3.6.0)

Imports vegan, fields, mgcv, Hmisc

Suggests vegdata, BiodiversityR, cluster

URL https://github.com/fvlampe/goeveg/

BugReports https://github.com/fvlampe/goeveg/issues

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Friedemann von Lampe [aut, cre] (ORCID:
<https://orcid.org/0009-0004-8015-9202>),
Schellenberg Jenny [aut]

Repository CRAN

Date/Publication 2025-06-13 15:30:02 UTC

Contents

`clean_matrix` *Clean species matrix*

Description

The function cleans a species matrix by removing species without occurrence (frequency = 0) and samples without any species (species number = 0) in one step.

It will also subset the corresponding observations of environmental data (samples in rows) or species trait data (species in rows), if passed to the function.

Usage

```
clean_matrix(matrix, env = NULL, traits = NULL)
```

Arguments

`matrix`	Community data, a matrix-like object with samples in rows and species in columns. Missing values (NA) or empty character cell values will be transformed to 0.
`env`	Optionally, a data frame of environmental variables, with samples in rows and variables in columns
`traits`	Optionally, a data frame of species traits, with species in rows and trait variables in columns

Value

If only a species matrix is provided, the return will be a cleaned species matrix. If environmental and/or trait data are also provided, the result will be a list of the cleaned and subsetted matrices/data frames.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

Examples

```
# Clean species matrix
schedenveg.clean <- clean_matrix(schedenveg)

# Clean species matrix and subset environmental data
scheden.clean <- clean_matrix(schedenveg, schedenenv)
schedenveg.clean <- scheden.clean$matrix
schedenenv.clean <- scheden.clean$env
```

`cov2per` *Conversion between cover-abundance codes and percentage cover*

Description

These functions perform a conversion between cover-abundance codes from different survey scales and percentage cover. They can be applied on a matrix-like object or a single vector.

`cov2per` performs conversion from cover-abundance to percentage values

`per2cov` performs conversion from percentage to cover-abundance values

You may chose between a set of scales with pre-defined conversion values, in `scale_tabs` or define your own conversion table following the same format.

Usage

```
cov2per(matrix, scale = "braun.blanquet", multiscale = FALSE)

per2cov(matrix, scale = "braun.blanquet", multiscale = FALSE)
```

Arguments

`matrix`	Community data, a vector or matrix-like object with cover-abundance values
`scale`	Cover-abundance scale (from `scale_tabs`) or dataframe with custom conversion table following the same format.
`multiscale`	A logical evaluation to `FALSE` *(default)* or `TRUE` indicating whether individual scales per sample should be used. For individual scales, you must provide a vector of the same length as the number of samples to the `scale` argument, defining the scale (from `scale_tabs`) for each sample (with samples as rows in matrix).

Value

A dataframe or vector containing the transformed data

Details

If scales are not only cover-based but also abundance-based (e.g. Braun-Blanquet, Kohler) there are often no unique definitions about their conversion into percentage cover. Therefore it is necessary to define and give reference to the applied conversion table.

Cover-abundance codes are transformed into the mean percentage cover of their class. For the conversion of percentage cover to cover-abundance codes, all values between the lower and upper border of the class are transformed into the corresponding code.

The included cover-abundance scales and the associated conversion tables with references are explained in `scale_tabs`. On this site you also find definitions for defining a custom table.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

See Also

`scale_tabs` for explanation and references of included conversion tables

Examples

```
## Conversion of species matrix with percentage cover to Braun-Blanquet values
schedenveg.bb <- per2cov(schedenveg)

## Conversion of only 10 samples to Londo values
schedenveg.londo <- per2cov(schedenveg[, 1:10], scale = "londo")

## Conversion of species matrix with Braun-Blanquet values to percentage cover
schedenveg.per <- cov2per(schedenveg.bb)
```

---

cv *Coefficient of variation (CV)*

Description

Compute the coefficient of variation (CV). The CV, also known as relative standard deviation (RSD), is a standardized measure of dispersion of a probability distribution or frequency distribution. It is defined as the ratio of the standard deviation to the mean and is often expressed as a percentage. In contrast to the standard deviation, it enables comparison between datasets as the CV is independent of the unit in which the measurement has been taken. If `na.rm` is `TRUE` then missing values are removed before computation proceeds.

Usage

```
cv(x, na.rm = FALSE)
```

Arguments

x	a numeric vector
na.rm	logical. Should missing values be removed?

Value

A numeric scalar – the sample coefficient of variation.

Details

The coefficient of variation (CV) should be computed only for data measured on a ratio scale (i.e. data with an absolute zero). The CV may not have any meaning for data on an interval scale.

According to *Dormann 2017* CV-values below 0.05 (5%) indicate very high precision of the data, values above 0.2 (20%) low precision. However, this is considered as a rule of thumb. In studies of highly variable systems (e.g. some ecological studies) CV values above 1 may occur.

The CV of a zero-length vector (after removal of NAs if na.rm = TRUE) is not defined and gives an error. If there is only a single value, sd is NA and cv returns NA.

References

Dormann, C. (2017). Parametrische Statistik. Verteilungen, maximum likelihood und GLM in R. *Springer*. doi:10.1007/9783662546840

Frost, J. (2023). Coefficient of variation in statistics. Statistics by Jim. https://statisticsbyjim.com/basics/coefficient-variation/.

See Also

sd

Examples

```
## Calculate CV for variable soil depth
cv(schedenenv$soil_depth)
```

---

deg2rad *Conversion between degrees and radians*

Description

deg2rad performs conversion from degrees to radians.

rad2deg performs conversion from radians to degrees.

Usage

```
deg2rad(x)

rad2deg(x)
```

Arguments

x	a numeric vector

Value

a numeric vector the same length as x

Details

Radians and degrees are both units used for measuring angles.

A degree is a measure of angle equal to 1/360th of a revolution, or circle. A radian is the measurement of angle equal to the length of an arc divided by the radius of the circle or arc. A circle is comprised of 2*pi radians, which is the equivalent of 360 degrees.

A common application in ecological studies is the conversion of measured exposition (in degrees) of plots into statistically meaningful measures, such as the north value or the east value. For this, the cosine (for northness) or sine (for eastness) is applied to the radian of the exposition.

References

BIPM (2019): The International System of Units (SI). Bureau international des poids et mesures, ninth edition. https://www.bipm.org/en/publications/si-brochure, ISBN 978-92-822-2272-0

Examples

```
## Covert the value pi to degrees
rad2deg(pi)

# Calculate north and east values based on exposition measured in degrees
north <- cos(deg2rad(schedenenv$exp))
east <- sin(deg2rad(schedenenv$exp))
```

---

merge_taxa *Merge taxa with identical names*

Description

The function offers a simple way to merge taxa with identical names in a vegetation table, e.g. due to necessary harmonization of the taxon level, to combine taxa of different layers or to remove duplicates. The original cover-abundance scales are maintained.

Usage

```
merge_taxa(
  vegtable,
  scale = "percentage",
  layers = "FALSE",
  method = "independent",
  clean_matrix = FALSE,
  backtransform = TRUE
)
```

Arguments

`vegtable`	Data frame with samples in columns and taxa in rows. Taxon names must be in the first column.
`scale`	Cover-abundance scale(s) of data. Default is percentage cover (values between 0 and 100) (`"percentage"`). Alternatively it can be one of the included scales in `scale_tabs` (e.g. `"braun.blanquet"`) or a custom scale defined as data frame following the same format. You can also provide a vector of the same length as the number of samples, to define individual scales (from `scale_tabs`) for each sample.
`layers`	A logical evaluation to `TRUE` or `FALSE` *(default)* indicating whether vegetation layers are to be included with layer information stored in the second column.
`method`	Choice of method to combine cover. `"independent"` *(default)* or `"exclusive"`. See details for methods.
`clean_matrix`	A logical evaluation to `TRUE` or `FALSE` *(default)* indicating whether taxa that do not occur (i.e. sum of abundances is zero) or samples that are empty (i.e. do not have any taxa present) are to be removed.
`backtransform`	A logical evaluation to `TRUE` *(default)* or `FALSE` indicating whether cover-abundance values should be back-transformed into their original cover-abundance scale or left as percentage cover.

Value

A data frame based on `vegtable` with merged taxa.

Details

The format required for this function is a data frame with samples in columns and taxa in rows, which corresponds to the export format of vegetation tables from Turboveg (Hennekens & Schaminee 2001). Taxon names must be in the first column of the table (not row names as these do not allow duplicates). If vegetation layers are to be included, layer information must be stored in the second column of the table. Taxa will then be merged only within the defined layers.

If a cover-abundance scale from `scale_tabs` is defined, all cover-abundance values will be transformed into percentage cover for the merging process, and then back-transformed into the original cover-abundance scale.

When combining cover values there are two possibilities following Fischer (2014) and Tichý & Holt (2011).

- `method = "independent"` *(default)* assumes that covers can overlap and that they do so independently of each other (e.g. individuals of the lower layer are growing beneath individuals of the upper layer). In case of two layers the sum of the cover of layer 1 and layer 2 is the sum of the covers minus the overlap. ($p1 + p2- p1 * p2) Usually the choice when merging the same taxon from different (sub-)layers.
- `method = "exclusive"` assumes the covers are mutually exclusive, cover values will be simply summed up (e.g. individuals grow side by side). Usually the chouce when merging within layers, e.g. aggregating distinct taxa.

Percentage cover values will eventually be truncated to 100

You may use the function `trans_matrix` to easily transpose the resulting vegetation table into a statistically analyzable species matrix.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

References

Fischer, H. S. (2015): On the combination of species cover values from different vegetation layers. *Applied Vegetation Science*, **18**: 169–170. doi:10.1111/avsc.12130

Tichý, L. & Holt, J. (2011): JUICE. Program for management, analysis and classification of ecological data. Vegetation Science Group, Masaryk University Brno, CZ.

Examples

```
## Merge taxa with identical names without any layer information
# Transpose table to required format
schedenveg.t <- data.frame(species = names(schedenveg), t(schedenveg))
# Add two duplicated taxa
schedenveg.t <- rbind(schedenveg.t, schedenveg.t[c(55, 61), ])

# Merge duplicated taxa using default 'independent' method
schedenveg.merged <- merge_taxa(schedenveg.t, scale = "percentage")
```

`ordiselect` *Species selection for ordination plots*

Description

This function simplifies the selection of relevant species in ordination diagrams. It works with result objects from the `vegan` package. The selection can be based upon cover abundances, frequency values and/or species fit to multivariate analysis (see Details). The result is a vector of names of the selected species and can be used for the `select` argument in ordination plots.

Usage

```
ordiselect(
  matrix,
  ord,
  ablim = 1,
  fitlim = 1,
  choices = c(1, 2),
  freq = FALSE,
  na.rm = FALSE,
  method = "axes",
  env,
  p.max = 0.05
)
```

Arguments

matrix	Community data, a matrix-like object with samples in rows and species in columns.
ord	vegan ordination result object (e.g. from decorana, cca or metaMDS).
ablim	Proportion of species with highest abundances to be displayed. Value between 0 and 1. Use negative sign for selection of lowest abundances, i.e. rarest species.
fitlim	Proportion of species with best fit to be displayed. Value between 0 and 1.
choices	Axes shown.
freq	Whether to use cover abundances (= default) or frequencies of matrix. If TRUE, frequencies of species are used.
na.rm	Set to TRUE if your ordination object contains NA (e.g. due to selection)
method	The species fit method: "axes" (= default) or "factors". See details for methods.
env	Fitted environmental variables (result object of envfit) for method = "factors". Only factor variables are used.
p.max	Significance limit for variables used in method = "factors".

Value

A vector of variable length containing the names of selected species from matrix.

Details

Two methods for species fit are implemented.

- In method = "axes" (default) species scores are used for selecting best fitting species. The basic assumption is that species that show high correlations to ordination axes provide a good fit to the assumed gradients, Hence high scores along ordination axes mean high correlation. All species with highest axis scores, defined by the threshold given in argument fitlim, will be filtered from the total ordination result.

- In `method = "factors"`, Euclidean distances between species and environmental variable centroids are calculated. Only factor variables are used from `envfit` output. The species with smallest distances, defined by `fitlim` argument as a threshold, will be filtered from the ordination result. The `p.max` argument allows selection of only significant variables, default is `p.max = 0.05`.

The species fit methods work well both in eigenvalue-based and in distance-based ordinations and provide good option of objective reduction of visible species in ordination plot for better interpretation issues. If axes fit should be applied on distance-based ordination, species scores need to be calculated during the analysis, e.g. by selecting `wascores = TRUE` in `metaMDS`. It is mostly recommendable to combine the species fit limit with an abundance limit to avoid overinterpretation of rare species.

For the abundance limit, note that the final proportion of the selected species may be higher than the indicated proportion if there are identical values in the abundances. For selection of least abundant (rarest) species you can use a negative sign, e.g. `ablim = -0.3` for the 30 percent least abundant species.

If both limits are defined only species meeting both conditions are selected. If no limit is defined for one of the arguments `ablim, fitlim`, all species are displayed.

The default for `matrix` input is a cover-abundance-matrix. This matrix should also be used for ordination.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>) and Jenny Schellenberg

Examples

```
## Calculate DCA
library(vegan)
scheden.dca <- decorana(schedenveg)

## Select the 30% most abundant species and call the result
limited <- ordiselect(schedenveg, scheden.dca, ablim = 0.3)
limited

# Use the result in plotting
plot(scheden.dca, display="n")
points(scheden.dca, display="sites")
points(scheden.dca, display="species",
   select = limited, pch = 3, col = "red", cex = 0.7)
ordipointlabel(scheden.dca, display="species",
    select = limited, col="red", cex = 0.7, add = TRUE)


## Select the 70% of the species with the best fit to the axes (highest species scores)
## AND belonging to the 30% most frequent species
limited <- ordiselect(schedenveg, scheden.dca, ablim = 0.3,
   fitlim = 0.7, freq = TRUE)

## Select the 30% least frequent species and call the result
```

```
limited <- ordiselect(schedenveg, scheden.dca, ablim = -0.3, freq = TRUE)
limited

## Select the 20% of species with the best fit to community assignment
## AND belonging to the 50% most abundant
## in NDMS for axes 1 & 3
nmds <- metaMDS(schedenveg, k = 3)   # run NMDS
env13 <- envfit(nmds, schedenenv, choices = c(1, 3))
limited13 <- ordiselect(schedenveg, nmds, method = "factors",
                        fitlim = 0.1, ablim = 1,
                        choices = c(1,3), env = env13)

# Use the result in plotting
plot(nmds, display="sites", choices = c(1, 3))
plot(env13, p.max = 0.05)
points(nmds, display="species", choices = c(1,3),
    select = limited13, pch = 3, col="red", cex=0.7)
ordipointlabel(nmds, display="species", choices = c(1,3),
    select = limited13, col="red", cex=0.7, add = TRUE)
```

---

racurve *Rank-abundance curves*

Description

This function draws a rank-abundance curve for community data. You can optionally add labels for a selected number of species. If you wish to draw multiple rank-abundance curves for selected samples use racurves.

Usage

```
racurve(
  matrix,
  main = "Rank-abundance diagram",
  nlab = 0,
  ylog = FALSE,
  frequency = FALSE,
  ylim = NULL,
  xlim = NULL
)
```

Arguments

matrix	Community data, a matrix-like object with samples in rows.
main	The main title (optional).
nlab	Number of labeled species (default = 0). Species are labeled in decreasing order beginning from the highest relative abundance.

ylog	If set on TRUE the y-axis is displayed on a log-scale.
frequency	If set on TRUE frequencies of species are calculated instead of relative abundances.
xlim, ylim	Define axis limits

Value

Returns an (invisible) list composed of:

abund	abundances of each species (in decreasing order)
rel.abund	relative abundances of each species (in decreasing order)
freq	frequency of each species (in decreasing order)

Details

Rank abundance curves or Whittaker plots (see *Whittaker 1965*) are used to display relative species abundance as biodiversity component. They are a means to visualize species richness and species evenness.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

References

Whittaker, R. H. (1965). Dominance and Diversity in Land Plant Communities: Numerical relations of species express the importance of competition in community function and evolution. *Science* **147 :** 250-260. doi:10.1126/science.147.3655.250

See Also

racurves for multiple curves and rankabundance from package BiodiversityR for a more sophisticated function

Examples

```
## Draw rank-abundance curve
racurve(schedenveg)

## Draw rank-abundance curve and label first 5 species
racurve(schedenveg, nlab = 5)

## Draw rank-abundance curve with log-scaled axis
racurve(schedenveg, ylog = TRUE)

## Draw rank-abundance curve with frequencies and no main title
racurve(schedenveg, frequency = TRUE, nlab = 1, main = "")
```

racurves *Multiple rank-abundance curves*

Description

This function draws multiple rank-abundance curves for selected samples into one diagram. If you wish to draw a simple rank-abundance curve see racurve.

Usage

```
racurves(matrix, main = "Rank-abundance diagram", bw = TRUE)
```

Arguments

matrix	Community data, a matrix-like object with samples in rows and species in columns. Rank-abundance curves are drawn for all selected rows (samples).
main	The main title (optional).
bw	If set on FALSE the lines will be drawn in colors instead of black/white lines with different line types.

Value

No return value, only diagram.

Details

Rank abundance curves or Whittaker plots (see *Whittaker 1965*) are used to display relative species abundance as biodiversity component. They are a means to visualize species richness and species evenness.

The axes of the diagram will be scaled according automatically. As the line type is used to differentiate between samples, a maximum of 6 curves per diagram is feasible in black/white mode.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

References

Whittaker, R. H. (1965). Dominance and Diversity in Land Plant Communities: Numerical relations of species express the importance of competition in community function and evolution. *Science* **147 :** 250-260. doi:10.1126/science.147.3655.250

See Also

racurve for a simple curve and rankabundance from package BiodiversityR for a more sophisticated function

Examples

```
## Draw multiple rank-abundance curves for selected samples
racurves(schedenveg[c(1,7,20,25), ])

## Draw multiple rank-abundance curves for selected samples with coloured lines
racurves(schedenveg[c(1,7,20,25), ], bw = FALSE)
```

`scale_tabs` *Conversion tables for cover-abundance scales*

Description

Dataset containing the conversion tables for cover-abundance scales used by the functions of this package, e.g. `cov2per` and `merge_taxa`.

Usage

```
data(scale_tabs)
```

Format

A list of 6 dataframes, each containing one conversion table

- `"braun.blanquet"` - Braun-Blanquet scale (Braun-Blanquet 1929, 1964). Conversion based on default values of the Turboveg program (Hennekens & Schaminee 2001).
- `"braun.blanquet2"` - Extended Braun-Blanquet scale (Reichelt & Wilmanns 1973). Conversion based on default values of the Turboveg program (Hennekens & Schaminee 2001).
- `"kohler"` - Kohler scale (Kohler 1978). Own conversion adapted from Lüderitz et al. (2009) and Janauer & Heindl (1998).
- `"kohler.zeltner"` - Simplified 3-level Kohler scale (Kohler & Zeltner 1974). Own conversion.
- `"londo"` - Londo scale (Londo 1976).
- `"niwap"` - Scale used by Lower Saxony species survey programs (Schacherer 2001). Own conversion as approximation based on a/b-values, which are based on individual counts. Needs to be adapted to sample sizes.
- `"pa"` - Presence/absence data (1/0).

Each dataframe has three columns:

- `"code"` - Cover-abundance code of scale
- `"cov_mean"` - Mean percentage cover of class for transformation into percentage values. All code values will be transformed to the corresponding value by `cov2per`
- `"cov_max"` - Maximum percentage cover of class. All values greater then the next lower class value up to this value will be transformed to the corresponding code by `per2cov`

References

Braun-Blanquet, J. (1964): Pflanzensoziologie: Grundzüge der Vegetationskunde (3. Aufl.). Wien-New York: Springer. doi:10.1007/9783709181102

Hennekens, S. M. & Schaminée, J. H. (2001): TURBOVEG, a comprehensive data base management system for vegetation data. *Journal of Vegetation Science*, **12**: 589–591. doi:10.2307/3237010

Janauer, G. A. & Heindl, E. (1998): Die Schätzskala nach Kohler: Zur Gültigkeit der Funktion f(y) = ax3 als Maß für die Pflanzenmenge von Makrophyten. *Verhandlungen der zoologisch-botanischen Gesellschaft in Wien*, **135**: 117–128.

Londo, G. (1976): The decimal scale for releves of permanent quadrats. *Vegetatio*, **33**: 61–64. doi:10.1007/BF00055300

Lüderitz, V., Langheinrich, U., & Kunz, C. (Eds.) (2009): Flussaltwässer: Ökologie und Sanierung (1. Aufl.). Wiesbaden: Vieweg + Teubner.

Kohler, A. (1978): Methoden der Kartierung von Flora und Vegetation von Süßwasserbiotopen. *Landschaft + Stadt*, **10**: 73–85.

Kohler, A. & Zeltner, G. (1974): Verbreitung und Ökologie von Makrophyten in Weichwasserflüssen des Oberpfälzer Waldes. *Hoppea*, **33**: 171–232.

Reichelt, G. & Wilmanns, O. (1973): Vegetationsgeographie. Braunschweig: Westermann.

Tichý, L., Hennekens, S. M., Novák, P., Rodwell, J. S., Schaminée, J. H. J. & Chytrý, M. (2020): Optimal transformation of species cover for vegetation classification. *Applied Vegetation Science*, **23**: 710–717. doi:10.1111/avsc.12510

Schacherer, A. (2001): Das Niedersächsische Pflanzenartenerfassungsprogramm. Herausgegeben vom Niedersächsischen Landesamt für Ökologie - Fachbehörde für Naturschutz. *Informationsdienst Naturschutz Niedersachsen*, **21(5 - Supplement Pflanzen)**: 1–20.

`schedenenv` *Header data for Vegetation releves from Scheden*

Description

An example vegetation dataset containing 28 grassland releves from Scheden, Niedersachsen, Germany. The releves were done May 2016 during a students field course at the University of Goettingen. Locations at the study site are based on the diploma thesis from *Eichholz (1997)*

Usage

```
data(schedenenv)
```

Format

A data frame with 28 rows (samples) and 10 variables

- **comm:** Plant community as defined in 1997: *Arrhenatheretum* (A) or *Gentiano-Koelerietum* (GK)
- **altit:** Altitude (m)

- **exp:** Exposition of plot (degrees)
- **north:** North value as cosine of aspect
- **slope:** Slope (degrees)
- **cov_herb:** Cover of herb layer (%)
- **cov_litt:** Cover of litter (%)
- **cov_moss:** Cover of mosses (%)
- **cov_opensoil:** Cover of open soil (%)
- **height_herb:** Average height of herb layer (cm)
- **soil_depth:** Soil depth (cm)

References

Eichholz, A. (1997): Wiesen und Magerrasen am Suedhang des Hohen Hagen. Diplomarbeit Biologie, University of Goettingen.

`schedenveg` *Vegetation releves from Scheden*

Description

An example vegetation dataset containing 28 grassland releves from Scheden, Niedersachsen, Germany. The releves were done May 2016 during a students field course at the University of Goettingen. Locations at the study site are based on the diploma thesis from *Eichholz (1997)*

Usage

```
data(schedenveg)
```

Format

A data frame with 28 rows (samples) and 155 variables (species)

References

Eichholz, A. (1997): Wiesen und Magerrasen am Suedhang des Hohen Hagen. Diplomarbeit Biologie, University of Goettingen.

`screeplot_NMDS` *Scree plot/Stress plot for NMDS*

Description

This function provides a plot of stress values against a given number of tested dimensions (default `k = 6`) in NMDS. This scree plot (or stress plot) shows the decrease in ordination stress with an increase in the number of ordination dimensions. It is based on function `metaMDS` (`vegan` package) and uses the `monoMDS` engine.

Usage

```
screeplot_NMDS(
  matrix,
  distance = "bray",
  k = 6,
  trymax = 20,
  autotransform = TRUE
)
```

Arguments

`matrix`	Community data, a matrix-like object with samples in rows and species in columns.
`distance`	Dissimilarity index used in vegdist.
`k`	Number of dimensions (default `k = 6`).
`trymax`	Maximum number of random configuration for iterative search search of stable solution.
`autotransform`	Whether to use transformation (see `metaMDS`) or not. Default is `autotransform = TRUE`.

Value

A numeric vector of length *k* containing stress values for *k* dimensions.

Details

The simplest indicator for the goodness of non-metric multidimensional scaling (NMDS) is the stress value. Stress is a value between 0 and 1 and expresses a proportion between the distance in the original dissimilarity matrix and the fitted distance in ordination space. The lower the stress value, the better is the fit. Details and exact formula are found under function `monoMDS`. Stress value depends on dimensionality: it is decreasing with increasing dimensionality. On the other hand, stress-reduction does not mean to maximize interpretation capability. Low-dimensional projections are often better to interpret and are so preferable for interpretation issues.

A scree plot (or sometimes also called stress plot) is a diagnostic plot to explore both, dimensionality and interpretative value. Often the 'elbow' of the plot is an indicator to determine the optimal number of dimensions to capture most information contained in the data. However, for ecological

data this is rarely seen, so that the following rule of thumb under consideration of the interpretability can be used.

Clarke 1993 suggests the following guidelines for stress values: <0.05 = excellent, <0.10 = good, <0.20 = usable without putting too much reliance on details, >0.20 = could be dangerous to interpret, > 0.35 = samples effectively randomly placed. The plot shows the border of the 0.20 stress value limit. Solutions with higher stress values should be interpreted with caution and those with stress above 0.35 are highly suspect.

It should be taken into account that the stress value naturally increases with the number of samples and/or variables: "The greater the number of samples, the harder it will be to reflect the complexity of their inter-relationship in a two-dimensional plot" (Clarke 1993, p. 125). Furthermore high stress values can be attributed to all points or only to few or even single points, that represent samples different to the others.

The scree plot is not an exclusive method to determine the optimal number of dimensions. The effect of individual points on the stress value can be explored with the `goodness`-function. Large values indicate poor fit and can be easily visualized in an ordination diagram.

Another diagnostic plot is the the Shepard diagram (`stressplot`), which is a scatterplot of the (Euclidean) distances in ordination space against the original dissimilarities. However, as the number of pairwise elements increases rapidly with the number of samples, the Shepard diagram will mostly lead to a high 'correlation-like' fit and lacks therefore a reliable and precise interpretability.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>) and Jenny Schellenberg

References

Clarke, K. R. (1993). Non-parametric multivariate analysis of changes in community structure. *Austral J Ecol* **18:** 117-143. doi:10.1111/j.14429993.1993.tb00438.x

See Also

`metaMDS` `stressplot`

Examples

```
## Use of function with default values
screeplot_NMDS(schedenveg)

## Use of function for testing 10 dimensions
screeplot_NMDS(schedenveg, k = 10)

## Alternative diagnostic plots
library(vegan)
nmds <- metaMDS(schedenveg, k = 2)

# Draw Shepard plot
stressplot(nmds)

# Calculate goodness of fit
```

```
gof <- goodness(object = nmds)

# Draw NMDS ordination diagram with sites
plot(nmds, display = "sites", type = "n", cex = 0.7)
# Add the points with size reflecting goodness of fit (bigger = worse fit)
points(nmds, display = "sites", cex = 2*gof/mean(gof))
```

---

sem *Standard error of the mean (SEM)*

Description

Compute the standard error of the mean (SEM). The SEM is the standard deviation of the sample-mean's estimate of a population mean. It therefore describes the accuracy of the calculation of a sample's mean. If `na.rm` is `TRUE` then missing values are removed before computation proceeds.

Usage

```
sem(x, na.rm = FALSE)
```

Arguments

`x`	a numeric vector
`na.rm`	logical. Should missing values be removed?

Value

A numeric scalar – the standard error of the mean.

Details

The SEM of a zero-length vector (after removal of `NA`s if `na.rm = TRUE`) is not defined and gives an error. The SEM of a length-one vector is `NA`.

See Also

`sd`

Examples

```
## Calculate mean and SEM for variable soil depth
mean(schedenenv$soil_depth)
sem(schedenenv$soil_depth)
```

specresponse *Species response curves*

Description

This function fits species response curves to visualize species responses to environmental gradients or ordination axes. It is based on Logistic Regression (binomial family) using Generalized Linear Models (GLMs) or Generalized Additive Models (GAMs) with integrated smoothness estimation. The function can draw response curves for single or multiple species.

Usage

```
specresponse(
  species,
  var,
  main,
  xlab,
  model = "auto",
  method = "env",
  axis = 1,
  points = FALSE,
  bw = FALSE,
  lwd = NULL,
  na.action = na.omit
)
```

Arguments

species	Species data (either a community matrix object with samples in rows and species in columns - response curves are drawn for all (selected) columns; or a single vector containing species abundances per plot).
var	Vector containing environmental variable (per plot) **OR** vegan ordination result object if method = "ord".
main	Optional: Main title.
xlab	Optional: Label of x-axis.
model	Defining the assumed species response: Default model = "auto" selects the model automatically based on AIC. Other methods are model = "linear" (linear response), model = "unimodal" (unimodal response), model = "bimodal" (bimodal response) and model = "gam" (using GAM with regression smoother).
method	Method defining the type of variable. Default method = "env" fits a response curve to environmental variables. Alternatively method = "ord" fits a response along ordination axes.
axis	Ordination axis (only if method = "ord").
points	If set on TRUE the species occurrences are shown as transparent points (the darker the point the more samples at this x-value). To avoid overlapping they are shown with vertical offset when multiple species are displayed.

bw	If set on TRUE the lines will be drawn in black/white with different line types instead of colors.
lwd	Optional: Graphical parameter defining the line width.
na.action	Optional: a function which indicates what should happen when the data contain NAs. The default is 'na.omit' (removes incomplete cases).

Value

Returns an (invisible) list with results for all calculated models. This list can be stored by assigning the result. For each model short information on type, parameters, explained deviance and corresponding p-value (based on chi-squared test) are printed.

Details

For response curves based on environmental gradients the argument var takes a single vector containing the variable corresponding to the species abundances.

For a response to ordination axis (method = "ord") the argument var requires a vegan ordination result object (e.g. from decorana, cca or metaMDS). First axis is used as default.

By default the response curves are drawn with automatic GLM model selection based on AIC out of GLMs with 1 - 3 polynomial degrees (thus excluding bimodal responses which must be manually defined). The GAM model is more flexible and chooses automatically between an upper limit of 3 - 6 degrees of freedom for the regression smoother.

Available information about species is reduced to presence-absence as species abundances can contain much noise (being affected by complex factors) and the results of Logistic Regression are easier to interpret showing the "probabilities of occurrence". Be aware that response curves are only a simplification of reality (model) and their shape is strongly dependent on the available dataset.

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

Examples

```
## Draw species response curve for one species on environmental variable
## with points of occurrences
specresponse(schedenveg$ArrElat, schedenenv$soil_depth, points = TRUE)

## Draw species response curve on environmental variable with custom labels
specresponse(schedenveg$ArrElat, schedenenv$soil_depth, points = TRUE,
       main = "Arrhenatherum elatius", xlab = "Soil depth")

## Draw species response curve on ordination axes
## First calculate DCA
library(vegan)
scheden.dca <- decorana(schedenveg)

# Using a linear model on first axis
specresponse(schedenveg$ArrElat, scheden.dca, method = "ord", model = "linear")
# Using an unimodal model on second axis
```

```
specresponse(schedenveg$ArrElat, scheden.dca, method = "ord", axis = 2, model = "unimodal")

## Community data: species (columns) need to be selected; call names() to get column numbers
names(schedenveg)
## Draw multiple species response curves on variable in black/white and store the results
res <- specresponse(schedenveg[ ,c(9,18,14,19)], schedenenv$height_herb, bw = TRUE)
# Call the results for Anthoxanthum odoratum
summary(res$AntOdor)

## Draw the same curves based on GAM
specresponse(schedenveg[ ,c(9,18,14,19)], schedenenv$height_herb, bw = TRUE, model = "gam")

## Draw multiple species response curves on variable with
## custom x-axis label and points of occurrences
specresponse(schedenveg[ ,c(9,18,14,19)], schedenenv$height_herb,
    xlab = "Height of herb layer (cm)", points = TRUE)

## Draw multiple species response curves on ordination axes
specresponse(schedenveg[ ,c(9,18,14,19)], scheden.dca, method = "ord")
specresponse(schedenveg[ ,c(9,18,14,19)], scheden.dca, method = "ord", axis = 2)
```

---

synsort *Sorting functions for synoptic tables*

Description

This function sorts synoptic tables from syntable function output. Sorting criteria can be either numerical values in synoptic tables, such as cluster-wise frequencies or fidelity measures, as well as combined criteria that also take into account differential character (according to the criteria defined by Tsiripidis et al., 2009).

The algorithm aims to sort species to blocked structure considering the defined criteria and input tables, with the best characterizing species on the top of the block, followed by species with descending importance for plant community description.

Usage

```
synsort(
  syn1,
  syn2 = syn1,
  matrix,
  cluster,
  method = "allspec",
  min1 = 0,
  min2 = 0
)
```

Arguments

syn1	Input synoptic table 1, a data frame with numerical data format, usually from syntable function output. See Details for input table format. The values of this table will be displayed in the final output table.
syn2	Optional second input table with additional numeric or differential character sorting criteria.
matrix	Species-sample matrix, already used for syntable function input
cluster	Integer or character vector/factor with classification cluster identity. Ensure matching order of cluster identity and samples in matrix for correct allocation of cluster numbers to samples.
method	Sorting algorithm and synoptic table output options (method = c("allspec", "alldiff")). See Details.
min1	Cluster-wise threshold minimum value for species shown in the final sorted synoptic table. Species below that minimum will be listed in the output ($others section).
min2	Threshold minimum value for considering species values of a numerical second input table syn2. Species below that minimum will not be displayed in final synoptic table, but will be listed in the output ($others section).

Value

Returns an (invisible) list composed of:

- $output Sorting method description
- $species Information to species included in the output table
- $samplesize Sample sizes in clusters
- $syntable Sorted synoptic table, with the numeric values of syn1 in the left-side columns and differential character of species on the right-side of the output table. See Tsiripidis et al. (2009) for details and criteria for the assignment of a differential species as p = positive, n = negative, pn = positive/negative.
- $others Species that are omitted in Synoptic table due to their failing reaching the given threshold values min1 and min2. Sorted alphabetically.
- $samples Sorted original species-sample matrix, with original Plot-IDs (as column names) and the cluster identity (Cluster_No as first row of output samples table)

Details

Two types of sorted synoptic tables can be created with this function:

- method = "allspec" (*default*) creates a sorted synoptic table basing on one or two numeric input tables, e.g. percentage or absolute frequencies, or phi fidelity values. Sorting criteria can be either given by only one input table by using only syn1 argument, as well as by two input tables with specifying syn2, too. Thereby, only values of syn1 will be shown in the final sorted table.

- `method = "alldiff"`: With including differential species character as sorting criteria, `syn1` must be numeric (e.g. percentage frequency) and `syn2` must contain information on differential character (output from `syntable` function with defined `type = "diffspec"`). The result table shows ALL diagnostic and non-diagnostic species, as long as they match the `min1` and `min2` thresholds. The algorithm detects highest cluster values of species calculated from `syn1` as base for sorting, but will consider differential character criterion from `syn2` as well. Species with high values in `syn1` AND positive differential character will then be listed on the top of a species block. Within such a block, the differentiating and high-abundant species are sorted in a way favoring species that are positive in only one or at least few clusters.

Author(s)

Jenny Schellenberg (<jschell@gwdg.de>)

References

Bruelheide, H. (2000): A new measure of fidelity and its application to defining species groups. *Journal of Vegetation Science* **11**: 167-178. doi:10.2307/3236796

Chytry, M., Tichy, L., Holt, J., Botta-Dukat, Z. (2002): Determination of diagnostic species with statistical fidelity measures. *Journal of Vegetation Science* **13**: 79-90. doi:10.1111/j.16541103.2002.tb02025.x

Sokal, R.R. & Rohlf, F.J. (1995): Biometry. 3rd edition Freemann, New York.

Tsiripidis, I., Bergmeier, E., Fotiadis, G. & Dimopoulos, P. (2009): A new algorithm for the determination of differential taxa. *Journal of Vegetation Science* **20**: 233-240. doi:10.1111/j.1654-1103.2009.05273.x

See Also

`syntable`

Examples

```
### Synoptic table of Scheden vegetation data using syntable()-function:
# classification to create a vector of cluster identity
library(cluster)
pam1 <- pam(schedenveg, 4)


### One input table for sorting:
## Synoptic table with percentage frequency of species in clusters, all species
unordered <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                      type = "percfreq")   # Unordered synoptic percentage frequency table
sorted <- synsort(syn1 = unordered$syntable, matrix = schedenveg,
                  cluster = pam1$clustering, method = "allspec", min1 = 0)
sorted             # view results
## Not run:
# Export sorted synoptic table
write.csv(sorted$syntab, "syntab.csv")
# Export sorted species-sample matrix with original releve data for postprocessing
write.csv(sorted$samples, "output_species_sample.csv")
## End(Not run)
```

```
## Synoptic table with only phi values
phi <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                    type = "phi")        # calculates cluster-wise phi for each species
phi_table <- synsort(syn1 = phi$syntable, matrix = schedenveg, cluster = pam1$clustering,
                     method = "allspec", min1 = 0.3)
phi_table     # view results


### Two numerical tables for sorting:
## Synoptic table showing percentage frequencies, but only for species with minimum phi-value
## of 0.3 AND exclude species with less than 25% percentage frequency

unordered <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                    type = "percfreq")   # Unordered synoptic percentage frequency table
phitable <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                    type = "phi")        # calculates cluster-wise phi for each species
# now sorting and arranging
phi_complete <- synsort(syn1 = unordered$syntable, syn2 = phitable$syntable,
                     matrix = schedenveg, cluster = pam1$clustering, method = "allspec",
                       min1 = 25, min2 = 0.3)
phi_complete      # view results

### Differential species analysis
differential <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                         type = "diffspec")

## Synoptic table with percentage frequency (only species >25%) and
## differential character.
complete <- synsort(syn1 = unordered$syntable, syn2 = differential$syntable,
                    matrix = schedenveg, cluster = pam1$clustering,
                    method = "alldiff", min1 = 25)
complete            # view result table
differential$differentials  # list differential species for clusters
```

`syntable` *Synoptic tables and calculation of cluster-wise frequencies, fidelity and differential species character*

Description

Synoptic tables are a tool for the visualization and interpretation of previously defined plant species groups (clusters), e.g. from cluster analysis, classification methods or pre-defined categories, e.g. spatial distribution units. They help to determine characteristic patterning of species occurrences in plant communities by calculating cluster-wise percentage or absolute frequencies, mean/median cover values, fidelity (phi) or differential species character.

`syntable` function calculates an unordered synoptic table for plant community analysis, using an input species-sample data frame and a vector of cluster identity input. The unordered output table can be sorted automatically with `synsort` function in this package.

Usage

```
syntable(matrix, cluster, abund = "percentage", type = "percfreq", digits = 0)
```

Arguments

matrix	Species matrix or data frame with species in columns and samples in rows. Missing values (NA) will be transformed to 0. If non-numeric abundance values are present, the matrix will be transformed to presence/absence with all non-zero values defined as 1. Species and sample names must be defined as column- and row names, respectively.
cluster	Integer or character vector/factor with classification cluster identity. Ensure matching order of cluster identity and samples in matrix for correct allocation of cluster numbers to samples.
abund	Type of abundances. Define whether input species matrix or data frame is percentage cover (abund = "percentage", default) or presence/absence data (abund = "pa", with values 0/1). You may use function cov2per to transform cover-abundance values from different scales into percentage cover.
type	Type of synoptic table output type = c("percfreq", "totalfreq", "mean","median", "diffspec", "phi"). See Details.
digits	Integer indicating the number of decimal places to be displayed in result tables (default 0)

Value

The function returns an (invisible) list of result components.

$syntable	unordered synoptic table for given species and clusters
$samplesize	total number of samples per cluster

Additionally for differential species character calculation:

$onlydiff	Synoptic table only with differential species
$others	List of non-differential species
$differentials	Lists differential species for each cluster

Details

For synoptic table calculation, six types are available.

- type = "percfreq" Creates a percentage frequency table *(default)*
- type = "totalfreq" Creates an absolute frequency table
- type = "mean" Calculates mean of species values given in matrix per cluster
- type = "median" Calculates median of species values given in matrix per cluster
- type = "diffspec" Calculates differential character of species according to Tsiripidis et al. 2009, with resulting character p = positive, n = negative, pn = positive- negative or no differential character (-). Consider that differential character is always restricted to some and not necessarily all of the other units, thus considering percentage frequency is essential for correct interpretation of the diagnostic species character. This calculation needs at least three groups.

- `type = "phi"` Calculates fidelity measure phi (algorithm basing on Sokal & Rohlf 1995, Bruelheide 2000). Values are ranging between -1 and 1 with high values near 1 indicating high fidelity.

For sorting the output synoptic table, use `synsort` function, providing several options.

Author(s)

Jenny Schellenberg (<jschell@gwdg.de>) and Friedemann von Lampe

References

Bruelheide, H. (2000): A new measure of fidelity and its application to defining species groups. *Journal of Vegetation Science* **11**: 167-178. doi:10.2307/3236796

Chytry, M., Tichy, L., Holt, J., Botta-Dukat, Z. (2002): Determination of diagnostic species with statistical fidelity measures. *Journal of Vegetation Science* **13**: 79-90. doi:10.1111/j.16541103.2002.tb02025.x

Sokal, R.R. & Rohlf, F.J. (1995): Biometry. 3rd edition Freemann, New York.

Tsiripidis, I., Bergmeier, E., Fotiadis, G. & Dimopoulos, P. (2009): A new algorithm for the determination of differential taxa. *Journal of Vegetation Science* **20**: 233-240. doi:10.1111/j.1654-1103.2009.05273.x

See Also

`synsort`

Examples

```
## Synoptic table of Scheden vegetation data
library(cluster)
pam1 <- pam(schedenveg, 4)  # PAM clustering with 4 clusters output

## 1) Unordered synoptic percentage frequency table
percfreq <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                         type = "percfreq")
                         percfreq                   # view results

## 2) Differential species analysis
differential <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                             type = "diffspec")
# show complete table with differential character of species
differential$syntable
# list differential species for second cluster
differential$differentials[2]

## 3) Synoptic table with phi fidelity
phitable <- syntable(schedenveg, pam1$clustering, abund = "percentage",
                         type = "phi")
phitable

## 4) Synoptic percentage frequency table based on historical classification from 1997
percfreq <- syntable(schedenveg, schedenenv$comm, abund = "percentage",
```

```
                          type = "percfreq")
percfreq
```

`trans_matrix` *Transpose species matrix*

Description

The function transposes a species matrix, while preserving correct species and sample names. The new column names must be stored as row names of the data frame. They may also be stored in the first column, when chosing the argument `row.names = F`.

Usage

```
trans_matrix(matrix, row.names = T, rmchar = FALSE)
```

Arguments

`matrix`	Community data, a data frame.
`row.names`	A logical evaluation indicating whether the new column names are stored as row names of the data frame `TRUE` *(default)* or in the first column `FALSE`.
`rmchar`	A logical evaluation indicating whether the first character of the original column names should be removed (default: `FALSE`).

Value

A transposed data frame.

Details

Sometimes vegetation data is organized as a data frame with samples in columns and taxa in rows, with taxon names stored in the first column, e.g. as result of the function `merge_taxa`. In this case you can use `row.names = F` to directly convert this species matrix into a statistically analyzable format, e.g. with `vegan`.

If your dataframe contains prepended "X" to each header due to numbered samples, you can use `rmchar = TRUE` to remove the first character of the column names during transposing. (You may also avoid this problem at all by using `check.names = FALSE` when loading the data in `read.table`)

Author(s)

Friedemann von Lampe (<fvonlampe@uni-goettingen.de>)

Examples

```
# Transpose species matrix
schedenveg.trans <- trans_matrix(schedenveg)
```

Index