Package 'douconca'

May 9, 2025

Type Package

Title Double Constrained Correspondence Analysis for Trait-Environment Analysis in Ecology

Version 1.2.3

Date 2025-05-09

Description Double constrained correspondence analysis (dc-CA) analyzes (multi-)trait (multi-)environment ecological data by using the 'vegan' package and native R code. Throughout the two step algorithm of ter Braak et al. (2018) is used. This algorithm combines and extends community- (sample-) and species-level analyses, i.e. the usual community weighted means (CWM)-based regression analysis and the species-level analysis of species-niche centroids (SNC)-based regression analysis. The two steps use canonical correspondence analysis to regress the abundance data on to the traits and (weighted) redundancy analysis to regress the CWM of the orthonormalized traits on to the environmental predictors. The function dc_CA() has an option to divide the abundance data of a site by the site total, giving equal site weights. This division has the advantage that the multivariate analysis corresponds with an unweighted (multi-trait) community-level analysis, instead of being weighted. The first step of the algorithm uses vegan::cca(). The second step uses wrda() but vegan::rda() if the site weights are equal. This version has a predict() function. For details see ter Braak et al. 2018 <doi:10.1007/s10651-017-0395-x>.
and ter Braak & van Rossum 2025 <doi:10.1016/j.ecoinf.2025.103143>.

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports ggplot2 (>= 3.5.1), ggrepel, gridExtra, permute, rlang, stats, vegan (>= 2.6-8)

RoxygenNote 7.3.2

NeedsCompilation no

Suggests rmarkdown, knitr, tinytest

VignetteBuilder knitr

URL `https://zenodo.org/records/13970152`,
`https://github.com/Biometris/douconca`

BugReports `https://github.com/Biometris/douconca/issues`

Author Cajo J.F ter Braak [aut] (ORCID:
<`https://orcid.org/0000-0002-0414-8745`>),
Bart-Jan van Rossum [aut, cre] (ORCID:
<`https://orcid.org/0000-0002-8673-2514`>)

Maintainer Bart-Jan van Rossum `<bart-jan.vanrossum@wur.nl>`

Repository CRAN

Date/Publication 2025-05-09 12:40:01 UTC

Contents

`anova.cca0` *Permutation Test for canonical correspondence analysis*

Description

`anova.cca0` performs residual predictor permutation for cca0, which is robust against differences in the weights (ter Braak & te Beest, 2022). The arguments of the function are similar to those of `anova.cca`, but more restricted.

Usage

```
## S3 method for class 'cca0'
anova(
  object,
  ...,
  permutations = 999,
  by = c("omnibus", "axis"),
  n_axes = "all"
)
```

Arguments

`object`	an object from `dc_CA`.
`...`	unused.
`permutations`	a list of control values for the permutations as returned by the function `how`, or the number of permutations required (default 999), or a permutation matrix where each row gives the permuted indices.
`by`	character `"axis"` which sets the test statistic to the first eigenvalue of the RDA model. Default: `NULL` which sets the test statistic to the weighted variance fitted by the predictors (=sum of all constrained eigenvalues). The default is quicker computationally as it avoids computation of an svd of permuted data sets.
`n_axes`	number of axes used in the test statistic (default: `"all"`). Example, the test statistic is the sum of the first two eigenvalues, if `n_axes=2`. With a numeric `n_axes` and model `~X + Condition(Z)`, the residuals of `X` with respect to `Z` are permuted with a test statistic equal to the sum of the first `n_axes` eigenvalues of the fitted `Y` in the model `Y ~ X + Z`, with `Y` the response in the model. In the default `"all"`, the test statistic is all eigenvalues of the model `Y ~ X|Z`, *i.e.* the effects of `X` after adjustment for the effects on `Y` of `Z`. If `by = "axis"`, the setting of `n_axes` is ignored.

Details

The algorithm is based on published R-code for residual predictor permutation in canonical correspondence analysis (ter Braak & te Beest, 2022), but using QR-decomposition instead of ad-hoc least-squares functions.

Note that `anova.cca0` is much slower than `anova.cca`.

As anova.cca is implemented in lower level language, it is difficult to see what it implements. In simulations (with and without Condition()) anova.cca gives results that are very similar to residual predictor permutation (RPP), but, with by = "axis", it can be conservative and is then less powerful than RPP (as tested with vegan 2.6-8).

Value

A list with two elements with names table and eigenvalues. The table is as from anova.cca and eigenvalues gives the CCA eigenvalues.

References

ter Braak, C.J.F. & te Beest, D.E. (2022). Testing environmental effects on taxonomic composition with canonical correspondence analysis: alternative permutation tests are not equal. Environmental and Ecological Statistics. 29 (4), 849-868. doi:10.1007/s10651022005454

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"

mod <- cca0(formula = abun ~ A1 + Moist + Mag + Use + Manure,
            data = dune_trait_env$envir)

mod # Proportions equal to those Canoco 5.15

scores(mod, which_cor = c("A1", "X_lot"), display = "cor")

set.seed(123)
anova(mod)
anova(mod, by = "axis")

mod2 <- vegan::cca(abun ~ A1 + Moist + Mag + Use + Manure,
                   data = dune_trait_env$envir)
anova(mod2, by = "axis")

dat <- dune_trait_env$envir
dat$Mag <- "SF"
predict(mod, type = "lc", newdata = dat)
```

---

anova.dcca   *Community- and Species-Level Permutation Test in Double Constrained Correspondence Analysis (dc-CA)*

Description

anova.dcca performs the community- and species-level permutation tests of dc-CA and combines these in the 'max test', which takes the maximum of the *P*-values. The function arguments are similar to (but more restrictive than) those of anova.cca.

Usage

```
## S3 method for class 'dcca'
anova(
  object,
  ...,
  rpp = TRUE,
  permutations = 999,
  by = c("omnibus", "axis"),
  n_axes = "all"
)
```

Arguments

object	an object from dc_CA.
...	unused.
rpp	Logical indicating residual predictor permutation (default TRUE). When FALSE, residual response permutation is used.
permutations	a list of control values for the permutations for species and sites (species first, sites second, for traits and environment) as returned by the function how, or the number of permutations required (default 999, or a two-vector with the number for the species-level test first and that for the sites-level second), or a list of two permutation matrices (again, species first, sites second) where each row gives the permuted indices.
by	character "axis" which performs a series of tests, one for each axis, with the eigenvalue of the axis as test statistic. Default: NULL which sets the test statistic to the inertia (sum of all double constrained eigenvalues; named constraintsTE in the inertia element of dc_CA. The interpretation of this inertia is, at the species-level, the environmentally constrained inertia explained by the traits (without trait covariates) and, at the community-level, the trait-constrained inertia explained by the environmental predictors (without covariates). The default (NULL) is computationally quicker as it avoids computation of an svd of permuted data sets.
n_axes	number of axes used in the test statistic (default: "all"). Example, the test statistic is the sum of the first two eigenvalues, if n_axes=2. With a numeric n_axes and model ~X + Condition(Z), the residuals of X with respect to Z are permuted with a test statistic equal to the sum of the first n_axes eigenvalues of the fitted Y in the model Y ~ X + Z, with Y the response in the model. In the default "all", the test statistic is all eigenvalues of the model Y ~ X|Z, *i.e.* the effects of X after adjustment for the effects on Y of Z. If by = "axis", the setting of n_axes is ignored.

Details

In the general case of varying site abundance totals (`divideBySiteTotals = FALSE`) both the community-level test and the species-level test use residualized predictor permutation (ter Braak 2022), so as to ensure that `anova.dcca` is robust against differences in species and site total abundance in the `response` (ter Braak & te Beest, 2022). The community-level test uses `anova_sites`. For the species-level test, `anova_species` is used.

With equal site weights, obtained with `divide.by.site.total = TRUE`, the community-level test is obtained by applying `anova` to `object$RDAonEnv` using `anova.cca`. This performs residualized response permutation which performs about equal to residualized predictor permutation in the equi-weight case. The function `anova.cca` is implemented in C and therefore a factor of 20 or so quicker than the native R-code used in `anova_sites`.

The `n_axes` argument is new, and can be considered experimental.

Value

A list of 3 of structures as from `anova.cca`. The elements are `c("species", "sites", "maxP")`

References

ter Braak, C.J.F. & te Beest, D.E. 2022. Testing environmental effects on taxonomic composition with canonical correspondence analysis: alternative permutation tests are not equal. Environmental and Ecological Statistics. 29 (4), 849-868. doi:10.1007/s10651022005454

ter Braak, C.J.F. (2022) Predictor versus response permutation for significance testing in weighted regression and redundancy analysis. Journal of statistical computation and simulation, 92, 2041-2059. doi:10.1080/00949655.2021.2019256

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)
anova(mod)

a_species <- anova_species(mod)
a_species
# anova_species can be used for manual forward selection of
# trait variables, as done for environmental variables in the demo
# dune_FS_dcCA.r, based on the first eigenvalue and its significance
# and adding the first axis of the constrained species scores from mod to
# the Condition of a new mod.
(eig1_and_pval <- c(eig = a_species$eig[1], pval = a_species$table$`Pr(>F)`[1]))
```

```
a_species$eig
anova_sites(mod)
```

anova.wrda — *Permutation Test for weighted redundancy analysis*

Description

anova.wrda performs residual predictor permutation for weighted redundancy analysis (wRDA), which is robust against differences in the weights (ter Braak, 2022). The arguments of the function are similar to those of anova.cca, but more restricted.

Usage

```
## S3 method for class 'wrda'
anova(
  object,
  ...,
  permutations = 999,
  by = c("omnibus", "axis"),
  n_axes = "all"
)
```

Arguments

object	an object from dc_CA.
...	unused.
permutations	a list of control values for the permutations as returned by the function how, or the number of permutations required (default 999), or a permutation matrix where each row gives the permuted indices.
by	character "axis" which sets the test statistic to the first eigenvalue of the RDA model. Default: NULL which sets the test statistic to the weighted variance fitted by the predictors (=sum of all constrained eigenvalues). The default is quicker computationally as it avoids computation of an svd of permuted data sets.
n_axes	number of axes used in the test statistic (default: "all"). Example, the test statistic is the sum of the first two eigenvalues, if n_axes=2. With a numeric n_axes and model ~X + Condition(Z), the residuals of X with respect to Z are permuted with a test statistic equal to the sum of the first n_axes eigenvalues of the fitted Y in the model Y ~ X + Z, with Y the response in the model. In the default "all", the test statistic is all eigenvalues of the model Y ~ X|Z, *i.e.* the effects of X after adjustment for the effects on Y of Z. If by = "axis", the setting of n_axes is ignored.

Details

The algorithm is based on published R-code for residual predictor permutation in weighted redundancy analysis (ter Braak, 2022), but using QR-decomposition instead of ad-hoc least-squares functions.

Value

A list with two elements with names table and eigenvalues. The table is as from anova.cca and eigenvalues gives the wrda eigenvalues.

References

ter Braak, C.J.F. (2022) Predictor versus response permutation for significance testing in weighted regression and redundancy analysis. Journal of statistical computation and simulation, 92, 2041-2059. doi:10.1080/00949655.2021.2019256

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
response <- dune_trait_env$comm[, -1]  # must delete "Sites"

w <- rep(1, 20)
w[1:10] <- 8
w[17:20] <- 0.5

object <- wrda(formula = response ~ A1 + Moist + Mag + Use + Condition(Manure),
               data = dune_trait_env$envir,
               weights = w)
object # Proportions equal to those Canoco 5.15

mod_scores <- scores(object, display = "all")
scores(object, which_cor = c("A1", "X_lot"), display = "cor")
anova(object)
```

anova_sites *Utility function: community-level permutation test in Double Constrained Correspondence Analysis (dc-CA)*

Description

anova_sites performs the community-level permutation test of dc-CA when site weights vary. The test uses residual predictor permutation (ter Braak 2022), which is robust against differences in site total abundance in the response in dc_CA (ter Braak & te Beest, 2022). The arguments of the function are similar to those of anova.cca, but more restricted. With equal site-totals as in dc_CA, anova(object$RDAonEnv) is much faster.

Usage

```
anova_sites(object, permutations = 999, rpp = TRUE, n_axes = "all", by = NULL)
```

Arguments

`object`	an object from `dc_CA`.
`permutations`	a list of control values for the permutations as returned by the function `how`, or the number of permutations required (default 999), or a permutation matrix where each row gives the permuted indices.
`rpp`	Logical indicating residual predictor permutation (default `TRUE`). When `FALSE`, residual response permutation is used.
`n_axes`	number of axes used in the test statistic (default: `"all"`). Example, the test statistic is the sum of the first two eigenvalues, if `n_axes=2`. With a numeric `n_axes` and model `~X + Condition(Z)`, the residuals of `X` with respect to `Z` are permuted with a test statistic equal to the sum of the first `n_axes` eigenvalues of the fitted `Y` in the model `Y ~ X + Z`, with `Y` the response in the model. In the default `"all"`, the test statistic is all eigenvalues of the model `Y ~ X|Z`, *i.e.* the effects of `X` after adjustment for the effects on `Y` of `Z`. If `by = "axis"`, the setting of `n_axes` is ignored.
`by`	character `"axis"` which sets the test statistic to the first eigenvalue of the dc-CA model. Default: `NULL` which sets the test statistic to the inertia (sum of all double constrained eigenvalues; named `constraintsTE` in the inertia element of `dc_CA`). This is the trait constrained inertia explained by the environmental predictors (without covariates), which is equal to the environmentally-constrained inertia explained by the traits (without trait covariates). The default is quicker computationally as it avoids computation of an svd of permuted data sets.

Details

The algorithm is analogous to that of `anova.wrda`. The function is used in `anova.dcca`.

Value

A list with two elements with names `table` and `eigenvalues`. The `table` is as from `anova.cca` and `eigenvalues` gives the dc-CA eigenvalues.

References

ter Braak, C.J.F. & te Beest, D.E. 2022. Testing environmental effects on taxonomic composition with canonical correspondence analysis: alternative permutation tests are not equal. Environmental and Ecological Statistics. 29 (4), 849-868. doi:10.1007/s10651022005454

ter Braak, C.J.F. (2022) Predictor versus response permutation for significance testing in weighted regression and redundancy analysis. Journal of statistical computation and simulation, 92, 2041-2059. doi:10.1080/00949655.2021.2019256

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
```

```
mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)
anova(mod)

a_species <- anova_species(mod)
a_species
# anova_species can be used for manual forward selection of
# trait variables, as done for environmental variables in the demo
# dune_FS_dcCA.r, based on the first eigenvalue and its significance
# and adding the first axis of the constrained species scores from mod to
# the Condition of a new mod.
(eig1_and_pval <- c(eig = a_species$eig[1], pval = a_species$table$`Pr(>F)`[1]))
a_species$eig
anova_sites(mod)
```

---

anova_species — *Utility function: Species-level Permutation Test in Double Constrained Correspondence Analysis (dc-CA)*

Description

`anova_species` performs the species-level permutation test of dc-CA which is part of `anova.dcca`. The test uses residual predictor permutation (ter Braak 2022), which is robust against differences in species total abundance in the `response` in `dc_CA` (ter Braak & te Beest, 2022). The arguments of the function are similar to those of `anova.cca`, but more restrictive.

Usage

```
anova_species(
  object,
  permutations = 999,
  rpp = TRUE,
  n_axes = "all",
  by = NULL
)
```

Arguments

`object`	an object from `dc_CA`.
`permutations`	a list of control values for the permutations as returned by the function `how`, or the number of permutations required (default 999) or a permutation matrix where each row gives the permuted indices.
`rpp`	Logical indicating residual predictor permutation (default `TRUE`). When `FALSE`, residual response permutation is used.

`n_axes`	number of axes used in the test statistic (default: `"all"`). Example, the test statistic is the sum of the first two eigenvalues, if `n_axes=2`. With a numeric `n_axes` and model `~X + Condition(Z)`, the residuals of `X` with respect to `Z` are permuted with a test statistic equal to the sum of the first `n_axes` eigenvalues of the fitted `Y` in the model `Y ~ X + Z`, with `Y` the response in the model. In the default `"all"`, the test statistic is all eigenvalues of the model `Y ~ X|Z`, *i.e.* the effects of `X` after adjustment for the effects on `Y` of `Z`. If `by = "axis"`, the setting of `n_axes` is ignored.
`by`	character `"axis"` which sets the test statistic to the first eigenvalue of the dc-CA model. Default: `NULL` which set the test statistic to the inertia (sum of all double constrained eigenvalues; named `constraintsTE` in the inertia element of `dc_CA`). This is the environmentally constrained inertia explained by the traits (without trait covariates), which is equal to the trait-constrained inertia explained by the environmental predictors (without covariates). The default is quicker computationally as it avoids computation of an svd of permuted data sets.

Details

In `anova_species`, the first step extracts the species-niche centroids (SNC) with respect to all dc-CA ordination axes from an existing dc-CA analysis. The second step, applies a weighted redundancy analysis of these SNCs with the traits as predictors. The second step is thus a species-level analysis, but the final results (eigenvalues/ordination axes) are identical to those of the analyses steps in `dc_CA`.The second step uses R-code that is analogous to that of `anova.wrda`.

Value

A list with two elements with names `table` and `eigenvalues`. The `table` is as from `anova.cca` and `eigenvalues` gives the dc-CA eigenvalues. This output can be used for scripting forward selection of traits, similar to the forward selection of environmental variables in the demo `dune_select.r`.

References

ter Braak, C.J.F. & te Beest, D.E. 2022. Testing environmental effects on taxonomic composition with canonical correspondence analysis: alternative permutation tests are not equal. Environmental and Ecological Statistics. 29 (4), 849-868. doi:10.1007/s10651022005454

ter Braak, C.J.F. (2022) Predictor versus response permutation for significance testing in weighted regression and redundancy analysis. Journal of statistical computation and simulation, 92, 2041-2059. doi:10.1080/00949655.2021.2019256

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
```

```
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)
anova(mod)

a_species <- anova_species(mod)
a_species
# anova_species can be used for manual forward selection of
# trait variables, as done for environmental variables in the demo
# dune_FS_dcCA.r, based on the first eigenvalue and its significance
# and adding the first axis of the constrained species scores from mod to
# the Condition of a new mod.
(eig1_and_pval <- c(eig = a_species$eig[1], pval = a_species$table$`Pr(>F)`[1]))
a_species$eig
anova_sites(mod)
```

`cca0` *Performs a canonical correspondence analysis*

Description

`cca0` is formula-based implementation of canonical correspondence analysis.

Usage

```
cca0(
  formula,
  response = NULL,
  data,
  traceonly = FALSE,
  cca_object = NULL,
  object4QR = NULL
)
```

Arguments

`formula`	one or two-sided formula for the rows (samples) with row predictors in `data`. The left hand side of the formula is ignored if it is specified in the next argument (`response`). Specify row covariates (if any) by adding + `Condition(covariate-formula)` to `formula` as in `rda`. The `covariate-formula` should not contain a ~ (tilde).
`response`	matrix or data frame of the abundance data (dimension *n* x *m*). Rownames of `response`, if any, are carried through. BEWARE: all rows and columns should have positive sums! Can be `NULL` if `cca_object` is supplied or if the response is `formula` is two-sided.
`data`	matrix or data frame of the row predictors, with rows corresponding to those in `response` (dimension *n* x *p*).

`traceonly`	logical, default `FALSE`. If `TRUE`, only the explained variance of the predictors and the `Condition()` are returned, *i.e* without performing a singular value decom-postion.
`cca_object`	a vegan-type cca-object of *transposed* `response`, from which chisq_residuals and row and column weights can be obtained.
`object4QR`	a vegan-type cca-object with weighted QR's for `formula`, i.e. `qr(Z)` and `qr(XZ)` obtainable via `get_QR(object4QR, model = "pCCA")` and `get_QR(object4QR, model = "CCA")`, respectively.

Details

The algorithm is a wrda on the abundance data after transformation to chi-square residuals.

It is much slower than `cca`. The only reason to use it, is that `anova.cca0` does residualized predictor permutation. It is unknown to the authors of `douconca` which method `anova.cca` implements. See `anova.cca0`.

Compared to `cca`, `cca0` does not have residual axes, *i.e.* no CA of the residuals is performed.

Value

All scores in the `cca0` object are in scaling `"sites"` (1): the scaling with *Focus on Case distances*.

The returned object has class `c("cca0" "wrda")` so that the methods `print`, `predict` and `scores` can use the `wrda` variant.

References

ter Braak C.J.F. and P. Šmilauer (2018). Canoco reference manual and user's guide: software for ordination (version 5.1x). Microcomputer Power, Ithaca, USA, 536 pp.

Oksanen, J., et al. (2022) vegan: Community Ecology Package. R package version 2.6-8. `https://CRAN.R-project.org/package=vegan`.

See Also

`scores.wrda`, `anova.cca0`, `print.wrda` and `predict.wrda`

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"

mod <- cca0(formula = abun ~ A1 + Moist + Mag + Use + Manure,
            data = dune_trait_env$envir)

mod # Proportions equal to those Canoco 5.15

scores(mod, which_cor = c("A1", "X_lot"), display = "cor")
```

```
set.seed(123)
anova(mod)
anova(mod, by = "axis")

mod2 <- vegan::cca(abun ~ A1 + Moist + Mag + Use + Manure,
                  data = dune_trait_env$envir)
anova(mod2, by = "axis")

dat <- dune_trait_env$envir
dat$Mag <- "SF"
predict(mod, type = "lc", newdata = dat)
```

`coef.dcca` *Coefficients of double-constrained correspondence analysis (dc-CA)*

Description

Fourth-corner coefficients and regression coefficients (of full or reduced rank) to predict traits from environment, environment from traits and response from trait and environment data.

Usage

```
## S3 method for class 'dcca'
coef(
  object,
  ...,
  type = c("fourth_corner", "all_reg", "env2traits_reg", "traits2env_reg"),
  rank = "full",
  normed = TRUE
)
```

Arguments

`object`	return value of `dc_CA`.
`...`	Other arguments passed to the function (currently ignored).
`type`	type of coefficients, `c("fourth_corner", "all_reg", "env2traits_reg", "traits2env_reg")` for fourth-corner coefficients and regression coefficients for all trait x environmental predictors, environmental predictors only and trait predictors only for prediction of the (transformed) response, traits and environmental values, respectively.
`rank`	rank (number of axes to use). Default "full" for all axes (no rank-reduction).
`normed`	logical (default `TRUE`) giving standardized regression coefficients and biplot scores. When `FALSE`, (regular) regression coefficients and (unstandardized) biplot scores.

Details

Regression coefficients are for standardized traits and environmental variables.

With covariates, `coef()` gives partialfourth-corner correlations. With `rank = 2`, `coef()` gives the two-dimensional approximation of the full-rank fourth-corner correlations in the biplot that displays the traits and environmental variables at arrow heads or points at `scores(mod, display = c("bp", "bp_traits"))`.

Value

a matrix with coefficients. The exact content of the matrix depends on the `type` of coefficient that is asked for.

Regression coefficients for a response variable are usually column-vectors. With **X** the matrix of units-by-predictors and **B** the matrix of predictors-by-response-variables, predictions or fits are of the form **Y = XB**. Analogously, `type = "trait2env"` gives a trait-by-environment matrix and `type = "env2traits"` gives an environment-by-trait matrix.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Condition(Manure),
             formulaTraits = ~ SLA + Height + LDMC + Condition(Seedmass) + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

# regression coefficients
coef(mod, type = "env2traits")
coef(mod, type = "traits2env")
coef(mod, type = "fourth")
coef(mod, type = "all_reg")
```

`dc_CA` *Performs (weighted) double constrained correspondence analysis (dc-CA)*

Description

Double constrained correspondence analysis (dc-CA) for analyzing (multi-)trait (multi-)environment ecological data using library `vegan` and native R code. It has a `formula` interface which allows to assess, for example, the importance of trait interactions in shaping ecological communities. The function `dc_CA` has an option to divide the abundance data of a site by the site total, giving equal site weights. This division has the advantage that the multivariate analysis corresponds with an unweighted (multi-trait) community-level analysis, instead of being weighted (Kleyer et al. 2012, ter Braak and van Rossum, 2025).

Usage

```
dc_CA(
  formulaEnv = NULL,
  formulaTraits = NULL,
  response = NULL,
  dataEnv = NULL,
  dataTraits = NULL,
  divideBySiteTotals = NULL,
  dc_CA_object = NULL,
  env_explain = TRUE,
  use_vegan_cca = FALSE,
  verbose = TRUE
)
```

Arguments

`formulaEnv`	two-sided or one-sided formula for the rows (samples) with row predictors in `dataEnv`. The left hand side of the formula is ignored if it is specified in the `response` argument. Specify row covariates (if any) by adding + `Condition(covariate-formula)` to `formulaEnv` as in `rda`. The `covariate-formula` should not contain a ~ (tilde). Default: `NULL` for `~.`, i.e. all variables in `dataEnv` are predictor variables.
`formulaTraits`	formula or one-sided formula for the columns (species) with column predictors in `dataTraits`. When two-sided, the left hand side of the formula is not used. Specify column covariates (if any) by adding + `Condition(covariate-formula)` to `formulaTraits` as in `cca`. The `covariate-formula` should not contain a ~ (tilde). Default: `NULL` for `~.`, i.e. all variables in `dataTraits` are predictor traits.
`response`	matrix, data frame of the abundance data (dimension *n* x *m*) or list with community weighted means (CWMs) from `fCWM_SNC`, NULL. If `NULL`, the response should be at the left-hand side of `formulaEnv`. See Details for analyses starting from community weighted means. Rownames of `response`, if any, are carried through.
`dataEnv`	matrix or data frame of the row predictors, with rows corresponding to those in `response`. (dimension *n* x *p*).
`dataTraits`	matrix or data frame of the column predictors, with rows corresponding to the columns in `response`. (dimension *m* x *q*).
`divideBySiteTotals`	logical; default `TRUE` for closing the data by dividing the rows in the `response` by their total. However, the default is `FALSE`, when the species totals are proportional to `N2*(N-N2)` with `N2` the Hill numbers of order 2 of the species and `N` the number of sites, as indicator that the response data have been pre-processed to `N2`-based marginals using `ipf2N2`.
`dc_CA_object`	optional object from an earlier run of this function. Useful if the same formula for the columns (`formulaTraits`), `dataTraits` and `response` are used with a new formula for the rows. If set, the data of the previous run is used and the result of its first step is taken for the new analysis and `env_explain` is set to `FALSE`.

`env_explain`	logical (default `TRUE`) for calculation of the inertia explained by the environmental variable (based on a CCA of abundance (with `divideBySiteTotals`, if true) on the environmental formula).
`use_vegan_cca`	default `TRUE`.
`verbose`	logical for printing a simple summary (default: TRUE)

Details

Empty (all zero) rows and columns in `response` are removed from the `response` and the corresponding rows from `dataEnv` and `dataTraits`. Subsequently, any columns with missing values are removed from `dataEnv` and `dataTraits`. It gives an error ('name_of_variable' not found), if variables with missing entries are specified in `formulaEnv` and `formulaTraits`.

Computationally, dc-CA can be carried out by a single singular value decomposition (ter Braak et al. 2018), but it is here computed in two steps. In the first step, the transpose of the `response` is regressed on to the traits (the column predictors) using `cca` with `formulaTraits`. The column scores of this analysis (in scaling 1) are community weighted means (CWM) of the orthonormalized traits. These are then regressed on the environmental (row) predictors using `wrda` with `formulaEnv` or using `rda`, if site weights are equal.

A dc-CA can be carried out on, what statisticians call, the sufficient statistics of the method. This is useful, when the abundance data are not available or could not be made public in a paper attempting reproducible research. In this case, `response` should be a list with as first element community weighted means (e.g. `list(CWM = CWMs)`) with respect to the traits, and the trait data, and, optionally, further list elements, for functions related to `dc_CA`. The minimum is a `list(CWM = CWMs, weight = list(columns = species_weights))` with CWM a matrix or data.frame, but then `formulaEnv`, `formulaTraits`, `dataEnv`, `dataTraits` must be specified in the call to `dc_CA`. The function `fCWM_SNC` and its example show how to set the `response` for this and helps to create the `response` from abundance data in these non-standard applications of dc-CA. Species and site weights, if not set in `response$weights` can be set by a variable `weight` in the data frames `dataTraits` and `dataEnv`, respectively, but formulas should then not be `~.`.

The statistics and scores in the example `dune_dcCA.r`, have been checked against the results in Canoco 5.15 (ter Braak & Šmilauer, 2018).

Value

A list of `class dcca`; that is a list with elements

CCAonTraits a `cca.object` from the `cca` analysis of the transpose of the closed `response` using formula `formulaTraits`.

formulaTraits the argument `formulaTraits`. If the formula was `~.`, it was changed to explicit trait names.

data a list of `Y`, `dataEnv` and `dataTraits`, after removing empty rows and columns in `response` and after closure if `divideBySiteTotals = TRUE` and with the corresponding rows in `dataEnv` and `dataTraits` removed.

weights a list of unit-sum weights of columns and rows. The names of the list are `c("columns", "rows")`, in that order.

Nobs number of sites (rows).

CWMs_orthonormal_traits Community weighted means w.r.t. orthonormalized traits.

RDAonEnv a `wrda` object or `cca.object` from the `wrda` or, if with equal row weights, `rda` analysis, respectively of the column scores of the `cca`, which are the CWMs of orthonormalized traits, using formula `formulaEnv`.

formulaEnv the argument `formulaEnv`. If the formula was `~.`, it was changed to explicit environmental variable names.

eigenvalues the dc-CA eigenvalues (same as those of the `rda` analysis).

c_traits_normed0 mean, sd, VIF and (regression) coefficients of the traits that define the dc-CA axes in terms of the traits with t-ratios missing indicated by `NA`s for 'tval1'.

inertia a one-column matrix with, at most, six inertias (weighted variances):

- total: the total inertia.
- conditionT: the inertia explained by the condition in `formulaTraits` if present (neglecting row constraints).
- traits_explain: the trait-structured variation, *i.e.* the inertia explained by the traits (without constaints on the rows and conditional on the Condition in `formulaTraits`). This is the maximum that the row predictors could explain in dc-CA (the sum of the last two items is thus less than this value).
- env_explain: the environmentally structured variation, *i.e.* the inertia explained by the environment (without constraints on the columns but conditional on the Condition `formulaEnv`). This is the maximum that the column predictors could explain in dc-CA (the item `constraintsTE` is thus less than this value). The value is `NA`, if there is collinearity in the environmental data.
- conditionTE: the trait-constrained variation explained by the condition in `formulaEnv`.
- constraintsTE: the trait-constrained variation explained by the predictors (without the row covariates).

If `verbose` is `TRUE` (or after `out <- print(out)` is invoked) there are three more items.

- `c_traits_normed`: mean, sd, VIF and (regression) coefficients of the traits that define the dc-CA trait axes (composite traits), and their optimistic t-ratio.
- `c_env_normed`: mean, sd, VIF and (regression) coefficients of the environmental variables that define the dc-CA axes in terms of the environmental variables (composite gradients), and their optimistic t-ratio.
- `species_axes`: a list with four items
  - `species_scores`: a list with names `c("species_scores_unconstrained", "lc_traits_scores")` with the matrix with species niche centroids along the dc-CA axes (composite gradients) and the matrix with linear combinations of traits.
  - `correlation`: a matrix with inter-set correlations of the traits with their SNCs.
  - `b_se`: a matrix with (unstandardized) regression coefficients for traits and their optimistic standard errors.
  - `R2_traits`: a vector with coefficient of determination (R2) of the SNCs on to the traits. The square-root thereof could be called the species-trait correlation in analogy with the species-environment correlation in CCA.
- `sites_axes`: a list with four items

- `site_scores`: a list with names `c("site_scores_unconstrained", "lc_env_scores")` with the matrix with community weighted means (CWMs) along the dc-CA axes (composite gradients) and the matrix with linear combinations of environmental variables.
- `correlation`: a matrix with inter-set correlations of the environmental variables with their CWMs.
- `b_se`: a matrix with (unstandardized) regression coefficients for environmental variables and their optimistic standard errors.
- `R2_env`: a vector with coefficient of determination (R2) of the CWMs on to the environmental variables. The square-root thereof has been called the species-environmental correlation in CCA.

All scores in the `dcca` object are in scaling `"sites"` (1): the scaling with *Focus on Case distances* .

References

Kleyer, M., Dray, S., Bello, F., Lepš, J., Pakeman, R.J., Strauss, B., Thuiller, W. & Lavorel, S. (2012) Assessing species and community functional responses to environmental gradients: which multivariate methods? Journal of Vegetation Science, 23, 805-821. doi:10.1111/j.16541103.2012.01402.x

ter Braak, CJF, Šmilauer P, and Dray S. (2018). Algorithms and biplots for double constrained correspondence analysis. Environmental and Ecological Statistics, 25(2), 171-197. doi:10.1007/s106510170395x

ter Braak C.J.F. and P. Šmilauer (2018). Canoco reference manual and user's guide: software for ordination (version 5.1x). Microcomputer Power, Ithaca, USA, 536 pp.

ter Braak, C.J.F. and van Rossum, B. (2025). Linking Multivariate Trait Variation to the Environment: Advantages of Double Constrained Correspondence Analysis with the R Package Douconca. Ecological Informatics, 88. doi:10.1016/j.ecoinf.2025.103143

Oksanen, J., et al. (2024). vegan: Community Ecology Package. R package version 2.6-6.1. `https://CRAN.R-project.org/package=vegan`.

See Also

`plot.dcca`, `scores.dcca`, `print.dcca` and `anova.dcca`

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"
mod <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
 verbose = FALSE)

print(mod) # same output as with verbose = TRUE (the default of verbose).
anova(mod, by = "axis")
# For more demo on testing, see demo dune_test.r
```

```
mod_scores <- scores(mod)
# correlation of axes with a variable that is not in the model
scores(mod, display = "cor", scaling = "sym", which_cor = list(NULL, "X_lot"))

cat("head of unconstrained site scores, with meaning\n")
print(head(mod_scores$sites))

mod_scores_tidy <- scores(mod, tidy = TRUE)
print("names of the tidy scores")
print(names(mod_scores_tidy))
cat("\nThe levels of the tidy scores\n")
print(levels(mod_scores_tidy$score))

cat("\nFor illustration: a dc-CA model with a trait covariate\n")
mod2 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits)

cat("\nFor illustration: a dc-CA model with both environmental and trait covariates\n")
mod3 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
  verbose = FALSE)

cat("\nFor illustration: same model but using dc_CA_object = mod2 for speed, ",
    "as the trait model and data did not change\n")
mod3B <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
               dataEnv = dune_trait_env$envir,
               dc_CA_object = mod2,
   verbose= FALSE)
cat("\ncheck on equality of mod3 (from data) and mod3B (from a dc_CA_object)\n",
    "the expected difference is in the component 'call'\n ")

print(all.equal(mod3[-c(5,12)], mod3B[-c(5,12)])) #  only the component call differs
print(mod3$inertia[-c(3,5),]/mod3B$inertia) #       and mod3 has two more inertia items
```

---

dune_trait_env *Dune meadow data with plant species traits and environmental variables*

Description

The data dune_trait_env contains three data frames with abundance data of 28 plant species in 20 samples (relevés), trait data (9 traits: of which 5 morphological and 4 ecological (Ellenberg indicator values) and environmental data (9 environmental variables, four of which are geographic coordinates). Compared to the data in Jongman et al. (1987, 1995), the two moss species are lacking, and the traits of plant species and the geographic coordinates of the samples are added.

The data and the following description are an edited version of the DataKey in the Jamil2013_AJ data set in the CESTES database (Jeliazkov et al. 2020).

The Dune Meadow Data originate from a MSc thesis report of Batterink & Wijffels (1983). It consisted of 80 relevés in about 68 dune meadows (lots) on the island Terschelling in the Netherlands. A subset of their data was selected by Caspar Looman as an example data set for the edited book Jongman, ter Braak and van Tongeren (1987, 1995). The subset consists of 20 relevés, 28 species (and 2 mosses, excluded here) and 5 environmental variables.

The trait data were taken from the LEDA database by Jamil et al 2013. The spatial coordinates were retrieved by geo-referencing in GIS of the maps in Batterink & Wijffels (1993) by Ruut Wegman and Cajo ter Braak. The X, Y coordinates are by geo-referencing the relevé locations on Kaart 3a, 3b and 3c; the X_lot, Y_lot coordinates are from Kaart 2a, 2b and 2c.

For the Ellenberg indicator values see Ellenberg (1992).

The data `dune_trait_env` is a list with elements that are data frames each

- `comm`: community data; vegetation data.
- `traits`: trait data, taken from the LEDA database.
- `envir`: environmental data, taken from Jongman et al. (1987,1995).

The community data collection was done by the Braun-Blanquet method; the data are recorded according to the ordinal scale of van der Maarel (1979, Vegetatio, 39, 97-114); see pages XVII-XVIII and 18 in Jongman, ter Braak & van Tongeren 1995. Nomenclature follows Heukels-Van der Meijden (1983) Flora van Nederland, 20th ed.

The morphological `traits` are

- `SLA`: Specific Leaf Area
- `Height`: Canopy height of a shoot
- `LDMC`: Leaf dry matter content
- `Seedmass`: Seed mass
- `Lifespan`: Life span. Nominal; annual vs. perennial

The ecological `traits` (habitat requirements) are the Ellenberg values

- `F`: Moisture (ranging [1 to 12] (low to high))
- `R`: Soil acidity, ranging [1 to 9] (acidic to alkaline)
- `N`: Nitrogen requirement, ranging [1 to 9] (low to high)
- `L`: Light requirement, ranging [1 to 9] (low to high)

The data frame `envir` contains the environmental variables

- `A1`: horizon thickness
- `Moist`: Moisture content of the soil (a five point scale)
- `Mag`: Grassland management type
- `Use`: type of use (Agricultural grassland use (1) hay production (2) intermediate (3) grazing)
- `Manure`: Quantity of manure applied based on N and P manuring (N/P class in B&W 1983)
- `X`: longitude geographical coordinates (m) of the 2x2 m2 sample (relevé)

- `Y`: latitude geographical coordinates (m) of the 2x2 m2 sample (relevé)
- `X_lot`: longitude geographical coordinates (m) of the lot center
- `Y_lot`: latitude geographical coordinates (m) of the lot center

The management types are standard farming (SF), biological farming (BF), hobby farming (HF), nature conservation management (NM). The coordinates are Rijksdriehoekscoordinaten in meters. `https://nl.wikipedia.org/wiki/Rijksdriehoekscoordinaten`

References

Batterink, M. & Wijffels, G. (1983) Een vergelijkend vegetatiekundig onderzoek naar de typologie en invloeden van het beheer van 1973 tot 1982 in de duinweilanden op Terschelling. Landbouwhogeschool. ISN 215909.01. WUR library stacks 704B58.

Ellenberg, H. (1992) Indicator values of plants in Central Europe. Scripta Geobotanica, 18, 258 pp.

Jamil, T., Ozinga, W.A., Kleyer, M. & ter Braak, C.J.F. (2013) Selecting traits that explain species–environment relationships: a generalized linear mixed model approach. Journal of Vegetation Science, 24, 988-1000. doi:10.1111/j.16541103.2012.12036.x.

Jeliazkov, A., Mijatovic, D., and 78 others. (2020) A global database for metacommunity ecology, integrating species, traits, environment and space. Scientific Data, 7. doi:10.1038/s4159701903447.

Jongman, R.H.G., ter Braak, C.J.F. & van Tongeren, O.F.R. (1987) Data analysis in community and landscape ecology. Pudoc, Wageningen. ISBN 90-220-0908-4.

Jongman, R.H.G., ter Braak, C.J.F. & van Tongeren, O.F.R. (1995) Data analysis in community and landscape ecology. Cambridge University Press, Cambridge. ISBN 0-521-47574-0. `https://edepot.wur.nl/248017`.

Kleyer, M., and 33 others (2008) The LEDA Traitbase: a database of life-history traits of the Northwest European flora. Journal of Ecology, 96, 1266-1274. doi:10.1111/j.13652745.2008.01430.x.

`fCWM_SNC` *Calculate community weighted means and species niche centroids for double constrained correspondence analysis*

Description

Double constrained correspondence analysis (dc-CA) can be calculated directly from community weighted means (CWMs), with the trait data from which the CWMs are calculated, and the environmental data and weights for species and sites (the abundance totals for species and sites). Statistical testing at the species level requires also the species niche centroids (SNCs). The function `fCWM_SNC` calculates the CWMs and SNCs from the trait and environmental data, respectively, using a formula interface, so as to allow categorical traits and environmental variables. The resulting object can be set as the `response` argument in `dc_CA` so as to give the same output as a call to `dc_CA` with the abundance data as `response`, at least up to sign changes of the axes.

Usage

```
fCWM_SNC(
  response = NULL,
  dataEnv = NULL,
  dataTraits = NULL,
  formulaEnv = NULL,
  formulaTraits = NULL,
  divideBySiteTotals = NULL
)
```

Arguments

`response`	matrix, data frame of the abundance data (dimension *n* x *m*) or list with community weighted means (CWMs) from `fCWM_SNC`, NULL. If `NULL`, the response should be at the left-hand side of `formulaEnv`. See Details for analyses starting from community weighted means. Rownames of `response`, if any, are carried through.
`dataEnv`	matrix or data frame of the row predictors, with rows corresponding to those in `response`. (dimension *n* x *p*).
`dataTraits`	matrix or data frame of the column predictors, with rows corresponding to the columns in `response`. (dimension *m* x *q*).
`formulaEnv`	two-sided or one-sided formula for the rows (samples) with row predictors in `dataEnv`. The left hand side of the formula is ignored if it is specified in the `response` argument. Specify row covariates (if any) by adding + `Condition(covariate-formula)` to `formulaEnv` as in `rda`. The `covariate-formula` should not contain a `~` (tilde). Default: `NULL` for `~.`, i.e. all variables in `dataEnv` are predictor variables.
`formulaTraits`	formula or one-sided formula for the columns (species) with column predictors in `dataTraits`. When two-sided, the left hand side of the formula is not used. Specify column covariates (if any) by adding + `Condition(covariate-formula)` to `formulaTraits` as in `cca`. The `covariate-formula` should not contain a `~` (tilde). Default: `NULL` for `~.`, i.e. all variables in `dataTraits` are predictor traits.
`divideBySiteTotals`	logical; default `TRUE` for closing the data by dividing the rows in the `response` by their total. However, the default is `FALSE`, when the species totals are proportional to `N2*(N-N2)` with `N2` the Hill numbers of order 2 of the species and `N` the number of sites, as indicator that the response data have been pre-processed to `N2`-based marginals using `ipf2N2`.

Details

The argument `formulaTraits` determines which CWMs are calculated. The CWMs are calculated from the trait data (non-centered, non-standardized). With trait covariates, the other predictor traits are adjusted for the trait covariates by weighted regression, after which the overall weighted mean trait is added. This has the advantage that each CWM has the scale of the original trait.

The SNCs are calculated analogously from environmental data.

Empty (all zero) rows and columns in response are removed from the response and the corresponding rows from dataEnv and dataTraits. Subsequently, any columns with missing values are removed from dataEnv and dataTraits. It gives an error (object 'name_of_variable' not found), if variables with missing entries are specified in formulaEnv and formulaTraits.

In the current implementation, formulaEnv and formulaTraits should contain variable names as is, *i.e.* transformations of variables in the formulas gives an error ('undefined columns selected') when the scores function is applied.

Value

The default returns a list of CWM, SNC, weights, formulaTraits, inertia (weighted variance explained by the traits and by the environmental variables) and a list of data with elements dataEnv and dataTraits.

References

Kleyer, M., Dray, S., Bello, F., Lepš, J., Pakeman, R.J., Strauss, B., Thuiller, W. & Lavorel, S. (2012) Assessing species and community functional responses to environmental gradients: which multivariate methods? Journal of Vegetation Science, 23, 805-821. doi:10.1111/j.16541103.2012.01402.x

ter Braak, CJF, Šmilauer P, and Dray S. 2018. Algorithms and biplots for double constrained correspondence analysis. Environmental and Ecological Statistics, 25(2), 171-197. doi:10.1007/s10651-0170395x

ter Braak C.J.F. and P. Šmilauer (2018). Canoco reference manual and user's guide: software for ordination (version 5.1x). Microcomputer Power, Ithaca, USA, 536 pp.

Oksanen, J., et al. (2022) vegan: Community Ecology Package. R package version 2.6-4. https://CRAN.R-project.org/package=vegan.

See Also

dc_CA, plot.dcca, scores.dcca, print.dcca and anova.dcca

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

CWMSNC <- fCWM_SNC(formulaEnv = ~ A1 + Moist + Manure + Use + Condition(Mag),
                  formulaTraits = ~ SLA + Height + LDMC + Condition(Seedmass) + Lifespan,
                   response = dune_trait_env$comm[, -1],  # must delete "Sites"
                   dataEnv = dune_trait_env$envir,
                   dataTraits = dune_trait_env$traits)
names(CWMSNC)
#CWMSNC$SNC <- NULL # would give correct dc-CA but no species-level t-values or test
mod <- dc_CA(response = CWMSNC) # formulas and data are in CWMSNC!
# note that output also gives the environment-constrained inertia,
# which is a bonus compare to the usual way to carry out a dcCA.
anova(mod)
```

`fitted.dcca` *Fitted values of double-constrained correspondence analysis (dc-CA)*

Description

Community weighted means (CWM) and species-niche centroids (SNC), as fitted (in full or reduced rank) from the environmental data and trait data, respectively, and the fitted response from trait and environment data.

Usage

```
## S3 method for class 'dcca'
fitted(object, ..., type = c("CWM", "SNC", "response"), rank = "full")
```

Arguments

`object`	return value of `dc_CA`.
`...`	Other arguments passed to the function (currently ignored).
`type`	type of prediction, `c( "CWM","SNC", "response")` for environmental values, values of traits, response (expected abundance).
`rank`	rank (number of axes to use). Default "full" for all axes (no rank-reduction).

Details

If `type="response"` the rowsums of `object$data$Y` are used to scale the fit to these sums. Many of the predicted response values may be negative, indicating expected absences (0) or small expected response values.

Value

a matrix with fitted value. The exact content of the matrix depends on the `type` of fits that are asked for.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Condition(Manure),
             formulaTraits = ~ SLA + Height + LDMC + Condition(Seedmass) + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)
# fit the mean traits at each site (20x6),
# that is CWM at each site
```

```
CWM.traits <- fitted(mod, type = "CWM")
head(CWM.traits)

# fit the mean environment for each species (28x8)
# that is SNC of each species
SNC.env <- fitted(mod, type = "SNC")
head(SNC.env)

fit.resp <- fitted(mod, type = "response")
# fitted often gives negative values and dc_CA cannot have negatives in the
# response so, modify fit.resp,
# which gives about similar eigenvalues as the original data
fit.resp[fit.resp < 0] <- 0
mod3 <- dc_CA(formulaEnv = mod$formulaEnv,
              formulaTraits = mod$formulaTraits,
              response = fit.resp,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)
mod3$eigenvalues / mod$eigenvalues
```

fN2 *Hill number of order 2: N2*

Description

Calculate Hill number N2.

Usage

```
fN2(x)
```

Arguments

x	a numeric vector.

Value

scalar: Hill number N2.

References

Hill,M.O. (1973). Diversity and evenness: a unifying notation and its consequences. Ecology, 54, 427-432. doi:10.2307/1934352.

ter Braak, C.J.F. (2019). New robust weighted averaging- and model-based methods for assessing trait-environment relationships. Methods in Ecology and Evolution, 10 (11), 1962-1971. doi:10.1111/2041210X.13278.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
Y <- dune_trait_env$comm[, -1] # must delete "Sites"
Y_N2 <- ipf2N2(Y, updateN2 = FALSE, N2N_N2_species = FALSE)
attr(Y_N2, "iter") # 4

# show that column margins of the transform matrix are
# equal to the Hill N2 values
diff(range(colSums(Y_N2) / apply(X = Y, MARGIN = 2, FUN = fN2))) #  8.881784e-16
diff(range(rowSums(Y_N2) / apply(X = Y, MARGIN = 1, FUN = fN2))) #  0.07077207

Y_N2i <- ipf2N2(Y, updateN2 = TRUE, N2N_N2_species = FALSE)
attr(Y_N2i, "iter") # 5
diff(range(colSums(Y_N2i) / apply(X = Y_N2i, MARGIN = 2, FUN = fN2))) # 2.220446e-15
diff(range(rowSums(Y_N2i) / apply(X = Y_N2i, MARGIN = 1, FUN = fN2))) # 0.105742

# the default version:
Y_N2N_N2i <- ipf2N2(Y)
# ie.
# Y_N2N_N2i <- ipf2N2(Y, updateN2 = TRUE, N2N_N2_species = TRUE)
attr(Y_N2N_N2i, "iter") # 16
N2 <- apply(X = Y_N2N_N2i, MARGIN = 2, FUN = fN2)
N <- nrow(Y)
diff(range(colSums(Y_N2N_N2i) / (N2 * (N - N2)))) # 2.220446e-16

N2_sites <- apply(X = Y_N2N_N2i, MARGIN = 1, FUN = fN2)
R <- rowSums(Y_N2N_N2i)
N * max(N2_sites / sum(N2_sites) - R / sum(R)) # 0.009579165

sum(Y_N2N_N2i) - sum(Y)

mod0 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              divide = FALSE,
              verbose = FALSE)

mod1 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_N2N_N2i,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)

mod1$eigenvalues / mod0$eigenvalues
# ratios of eigenvalues greater than 1,
# indicate axes with higher (squared) fourth-corner correlation
# ipf2N2 for a presence-absence data matrix
Y_PA <- 1 * (Y > 0)
Y_PA_N2 <- ipf2N2(Y_PA, N2N_N2_species = FALSE)
attr(Y_PA_N2, "iter") # 1
diff(range(Y_PA - Y_PA_N2)) # 4.440892e-16, i.e no change

Y_PA_N2i <- ipf2N2(Y_PA, N2N_N2_species = TRUE)
attr(Y_PA_N2i, "iter") # 9
N_occ <- colSums(Y_PA) # number of occurrences of species
N <- nrow(Y_PA)
plot(N_occ, colSums(Y_PA_N2i))
cor(colSums(Y_PA_N2i), N_occ * (N - N_occ)) # 0.9916
mod2 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_PA,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              divideBySiteTotals = FALSE,
              verbose = FALSE)

mod3 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_PA_N2i,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)

mod3$eigenvalues / mod2$eigenvalues
# ratios of eigenvalues greater than 1,
# indicate axes with higher (squared) fourth-corner correlation
```

FS *Default forward selection function.*

Description

Default forward selection function.

Usage

```
FS(mod, ...)
```

Arguments

`mod`	A fitted model.
`...`	Further arguments passed to other methods.

Value

The results from applying forward selection on the fitted model.

`FS.dcca` *Forward selection of traits or environmental variables using dc-CA.*

Description

Forward selection of traits or environmental variables using dc-CA.

Usage

```
## S3 method for class 'dcca'
FS(
  mod,
  ...,
  select = c("traits", "env"),
  consider,
  permutations = 999,
  n_axes = "all",
  initial_model = "1",
  factor2categories = TRUE,
  test = TRUE,
  threshold_P = 0.1,
  PvalAdjustMethod = "holm",
  max_step = 10,
  verbose = FALSE
)
```

Arguments

`mod`	dc-CA model.
`...`	unused.
`select`	Character, the default `"traits"` selects trait variables based on the environmental model of `mod`. Set `select = "e"` to select environmental variables based on the trait model of `mod`.
`consider`	character vector of names considered for addition, either trait names in `mod$data$dataTraits` or environmental variable names in in `mod$data$dataEnv`. The names may include transformations of predictor variables, such as `log(.)`, if `consider` does not include factors or if `factor2categories=FALSE`. If `consider` includes factors, such transformations give in a error in the default setting (`factor2categories=TRUE`).
`permutations`	a list of control values for the permutations as returned by the function how, or the number of permutations required (default 999), or a permutation matrix where each row gives the permuted indices.
`n_axes`	number of eigenvalues to select upon. The sum of `n_axes` eigenvalues is taken as criterion. Default `"full"` for selection without dimension reduction to `n_axes`. If `n_axes =1`, selection is on the first eigenvalue for selection of variables that form an optimal one-dimensional model.

`initial_model`	character specifying what should be inside `Condition()`. Default: `"1"` (nothing, the intercept only). Examples: With selection of environmental varialbles, `"region"` for a within-region analysis or `"A*B"` for a within analysis specified by the interaction of factors `A` and `B`, with `region`, `A`, `B` in the data. Analogously, a trait or trait factor can be the initial model.
`factor2categories`	logical, default `TRUE`, to convert factors to their categories (set(s) of indicator values). If `FALSE`, the selection uses, the fit of a factor divided by its number of categories minus 1.
`test`	logical; default: `TRUE`.
`threshold_P`	significance level, after adjustment for testing multiplicity, for addition of a variable to the model.
`PvalAdjustMethod`	method for correction for multiple testing in `p.adjust`, default `"holm"`, which is an improved version Bonferroni.
`max_step`	maximal number of variables selected.
`verbose`	show progress, default: `TRUE`.

Details

The selection is on the basis of the additional fit (inertia) of a variable given the variables already in the model.

The function `FS` does not implement the max test and may thus be liberal. It is recommended to test the final model (element `model_ final` of the returned object) by `anova`.

Value

list with three elements: `final...` with selected variables, `model_final`, and `process` with account of the selection process

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1] # must delete "Sites"

mod <- dc_CA(formulaEnv = abun ~ Moist + Mag,
             formulaTraits = ~ F + R + N + L,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

# selection of traits with environmental model of mod (~ Moist+Mag)
out1 <- FS(mod, consider = c("F", "R", "N", "L"),
           select = "traits", verbose = FALSE)

names(out1)
```

```
out1$finalWithOneExtra
out1$model_final

# selection of environmental variables with trait model of mod (~ F + R + N + L)
out2 <- FS(mod, consider =  c("A1", "Moist", "Mag", "Use", "Manure"),
           select= "env", verbose = FALSE)

names(out2)
out2$finalWithOneExtra
out2$model_final

# selection of environmental variables without a trait model
# i.e. with a single constraint
mod3 <- cca0(mod$data$Y ~ Moist, data = mod$data$dataEnv)
out3 <- FS(mod3, consider = c("A1", "Moist", "Mag", "Use", "Manure"),
           threshold_P = 0.05)

out3$finalWithOneExtra
out3$model_final

# selection of traits without an environmental model
#                         i.e. with a single constraint
tY <- t(mod$data$Y)

mod4 <- cca0(tY ~ L, data = mod$data$dataTraits)

names(mod$data$dataTraits)
out4 <- FS(mod4,
           consider =  c("SLA", "Height", "LDMC", "Seedmass", "Lifespan",
                         "F", "R", "N", "L"))

out4$finalWithOneExtra
out4$model_final
```

`FS.wrda` *Forward selection of predictor variables using wrda or cca0*

Description

Forward selection of predictor variables using wrda or cca0

Usage

```
## S3 method for class 'wrda'
FS(
  mod,
  ...,
  consider = NULL,
  permutations = 999,
  n_axes = "all",
  initial_model = "1",
  factor2categories = TRUE,
  test = TRUE,
  threshold_P = 0.1,
  PvalAdjustMethod = "holm",
  max_step = 10,
  verbose = FALSE
)
```

Arguments

`mod`	initial wrda or cca0 model with at least on predictor variable,
`...`	unused.
`consider`	character vector of names in `mod$data` to consider for addition.
`permutations`	a list of control values for the permutations as returned by the function `how`, or the number of permutations required (default 999), or a permutation matrix where each row gives the permuted indices.
`n_axes`	number of eigenvalues to select upon. The sum of `n_axes` eigenvalues is taken as criterion. Default `"full"` for selection without dimension reduction to `n_axes`. If `n_axes =1`, selection is on the first eigenvalue for selection of variables that form an optimal one-dimensional model.
`initial_model`	character specifying what should be inside `Condition()`. Default: `"1"` (nothing, the intercept only). Examples: `"region"` for a within-region analysis or `"A*B"` for a within analysis specified by the interaction of factors `A` and `B`, with `region`, `A`, `B` in the data.
`factor2categories`	logical, default `TRUE`, to convert factors to their categories (set(s) of indicator values). If `FALSE`, the selection uses, the fit of a factor divided by its number of categories minus 1.
`test`	logical; default: `TRUE`.
`threshold_P`	significance level, after adjustment for testing multiplicity, for addition of a variable to the model.
`PvalAdjustMethod`	method for correction for multiple testing in `p.adjust`, default `"holm"`, which is an improved version Bonferroni.
`max_step`	maximal number of variables selected.
`verbose`	show progress, default: `TRUE`.

Details

The selection is on the basis of the additional fit (inertia) of a variable given the variables already in the model.

The names in `consider` may include transformations of predictor variables, such as `log(.)`, if `consider` does not include factors or if `factor2categories=FALSE`. If `consider` does include factors, such transformations give in a error in the default setting (`factor2categories=TRUE`).

Value

list with three elements: `final...` with selected variables and `model_final` and `process` with account of the selection process If `is.numeric(n_axes)`, then the variance in the returned table is the sum of the n_axes eigenvalues of the current model (all variables so far included).

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1] # must delete "Sites"

mod <- dc_CA(formulaEnv = abun ~ Moist + Mag,
             formulaTraits = ~ F + R + N + L,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

# selection of traits with environmental model of mod (~ Moist+Mag)
out1 <- FS(mod, consider = c("F", "R", "N", "L"),
           select = "traits", verbose = FALSE)

names(out1)
out1$finalWithOneExtra
out1$model_final

# selection of environmental variables with trait model of mod (~ F + R + N + L)
out2 <- FS(mod, consider =  c("A1", "Moist", "Mag", "Use", "Manure"),
           select= "env", verbose = FALSE)

names(out2)
out2$finalWithOneExtra
out2$model_final

# selection of environmental variables without a trait model
# i.e. with a single constraint
mod3 <- cca0(mod$data$Y ~ Moist, data = mod$data$dataEnv)
out3 <- FS(mod3, consider = c("A1", "Moist", "Mag", "Use", "Manure"),
           threshold_P = 0.05)

out3$finalWithOneExtra
out3$model_final

# selection of traits without an environmental model
#                          i.e. with a single constraint
tY <- t(mod$data$Y)

mod4 <- cca0(tY ~ L, data = mod$data$dataTraits)

names(mod$data$dataTraits)
out4 <- FS(mod4,
          consider =  c("SLA", "Height", "LDMC", "Seedmass", "Lifespan",
                        "F", "R", "N", "L"))

out4$finalWithOneExtra
out4$model_final
```

---

getPlotdata — *Utility function: extracting data from a* dc_CA *object for plotting a single axis by your own code or* plot.dcca*.*

Description

getPlotdata extracts data from a dc_CA object for plotting the CWMs and SNCs of a single axis.

Usage

```
getPlotdata(
  x,
  axis = 1,
  envfactor = NULL,
  traitfactor = NULL,
  newnames = NULL,
  facet = TRUE,
  remove_centroids = FALSE
)
```

Arguments

x	results from dc_CA of class dcca.
axis	the axis number to get (default 1).
envfactor	name of row factor to display as color and lines in the CWM plot (default NULL). The default extracts the factor from the environmental model. If set to NA, no additional coloring and lines are displayed in plot.dcca. The parameter sets the groups variable in the CWM_SNC data frame of the return value/in the plot.
traitfactor	name of column factor to display as color and lines in the SNC plot (default NULL). The default extracts the factor from the trait model. If set to NA, no additional coloring and lines are displayed in plot.dcca. The parameter sets the groups variable in the CWM_SNC data frame of the return value/in the plot.
newnames	a list with two elements: names for traits and for environmental variables, default NULL for names derived from the result of scores.dcca with tidy = TRUE.
facet	logical. Default TRUE for CWMs and SNCs plots in separate panels. If FALSE, this parameter changes the position of the environmental centroid names (from left to right).
remove_centroids	logical to remove any centroids from the plot data (default FALSE). Can be a two-vector, *e.g.* c(TRUE, FALSE) to remove only the trait centroids.

Details

The current implementation sets the `traitfactor` to `NA` if the trait model contains more than a single trait factor and the `envfactor` to `NA` if the environmental model contains more than a single environmental factor.

Value

A list with three components

CWM_SNC a data.frame containing plot data

trait_env_scores a vector of scores per trait/environment

newNameList a vector of new names to be used in the plot

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

# must delete "Sites" from response matrix or data frame
Y <- dune_trait_env$comm[, -1] # must delete "Sites"

out <- dc_CA(formulaEnv = ~ A1 + Moist + Use + Manure + Mag,
                   formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
                   response = Y,
                   dataEnv = dune_trait_env$envir,
                   dataTraits = dune_trait_env$traits,
                   verbose = FALSE)
dat <- getPlotdata(out)
names(dat)
names(dat$CWM_SNC)
levels(dat$CWM_SNC$groups)

plot(out)
```

`ipf2N2` *Iterative proportional fitting of an abundance table to Hill-N2 marginals*

Description

Function for preprocessing/transforming an abundance table by iterative proportional fitting, so that the transformed table has marginals `N2` or `N2(N-N2)` with `N` the number of elements in the margin. Hill-N2 is the effective number of species. It is of intrinsic interest in weighted averaging (CWM and SNC) as their variance is approximately inversely proportional to N2 (ter Braak 2019), and therefore of interest in `dc_CA`.

Usage

```
ipf2N2(
  Y,
  max_iter = 100,
  updateN2 = TRUE,
  N2N_N2_species = TRUE,
  N2N_N2_sites = FALSE,
  crit = 0.01
)
```

Arguments

`Y`	abundance table (matrix or dataframe-like), ideally, with names for rows and columns. BEWARE: all rows and columns should have positive sums!
`max_iter`	maximum number of iterative proportional fitting (ipf) iterations.
`updateN2`	logical, default `TRUE`. If `FALSE` the marginal sums are proportional to the N2-marginals of the initial table, but the N2-marginals of the returned matrix may not be equal to their marginal sum. If `updateN2 = TRUE` and `N2N_N2_species=TRUE` (the default), the column marginals are `N2(N-N2)/2` with `N` the number of sites. The row sums are then proportional to, what we term, the effective number of informative species. If `N2N_N2_species = FALSE`, the returned transformed table has N2 columns marginals, *i.e.* `colSums(Y2) = const*N2species(Y2)` with `Y2` the return value of `ipf2N2` and `const` a constant. If converged, N2 row marginals are equal to the row sums, *i.e.* `rowSums(Y2) = approx. N2sites(Y2)`.
`N2N_N2_species`	Set marginals proportional to `N2(N-N2)` Default `TRUE`.
`N2N_N2_sites`	Default `FALSE`. Do not change.
`crit`	stopping criterion.

Details

Applying `ipf2N2` with `N2N_N2_species=FALSE` to an presence-absence data table returns the same table. However, a species that occurs everywhere (or in most of the sites) is not very informative. This is acknowledged with the default option `N2N_N2_species=TRUE`. Then, with `N2N_N2_species=TRUE`, species that occur in more than halve the number of sites are down-weighted, so that the row sum is no longer equal to the richness of the site (the number of species), but proportional to the number of informative species.

Value

a matrix of the same order as the input `Y`, obtained after ipf to N2-marginals.

References

ter Braak, C.J.F. (2019). New robust weighted averaging- and model-based methods for assessing trait-environment relationships. Methods in Ecology and Evolution, 10 (11), 1962-1971. doi:10.1111/2041210X.13278

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
Y <- dune_trait_env$comm[, -1] # must delete "Sites"
Y_N2 <- ipf2N2(Y, updateN2 = FALSE, N2N_N2_species = FALSE)
attr(Y_N2, "iter") # 4

# show that column margins of the transform matrix are
# equal to the Hill N2 values
diff(range(colSums(Y_N2) / apply(X = Y, MARGIN = 2, FUN = fN2))) #  8.881784e-16
diff(range(rowSums(Y_N2) / apply(X = Y, MARGIN = 1, FUN = fN2))) #  0.07077207

Y_N2i <- ipf2N2(Y, updateN2 = TRUE, N2N_N2_species = FALSE)
attr(Y_N2i, "iter") # 5
diff(range(colSums(Y_N2i) / apply(X = Y_N2i, MARGIN = 2, FUN = fN2))) # 2.220446e-15
diff(range(rowSums(Y_N2i) / apply(X = Y_N2i, MARGIN = 1, FUN = fN2))) # 0.105742

# the default version:
Y_N2N_N2i <- ipf2N2(Y)
# ie.
# Y_N2N_N2i <- ipf2N2(Y, updateN2 = TRUE, N2N_N2_species = TRUE)
attr(Y_N2N_N2i, "iter") # 16
N2 <- apply(X = Y_N2N_N2i, MARGIN = 2, FUN = fN2)
N <- nrow(Y)
diff(range(colSums(Y_N2N_N2i) / (N2 * (N - N2)))) # 2.220446e-16

N2_sites <- apply(X = Y_N2N_N2i, MARGIN = 1, FUN = fN2)
R <- rowSums(Y_N2N_N2i)
N * max(N2_sites / sum(N2_sites) - R / sum(R)) # 0.009579165

sum(Y_N2N_N2i) - sum(Y)

mod0 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              divide = FALSE,
              verbose = FALSE)

mod1 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_N2N_N2i,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)

mod1$eigenvalues / mod0$eigenvalues
# ratios of eigenvalues greater than 1,
# indicate axes with higher (squared) fourth-corner correlation
# ipf2N2 for a presence-absence data matrix
Y_PA <- 1 * (Y > 0)
Y_PA_N2 <- ipf2N2(Y_PA, N2N_N2_species = FALSE)
attr(Y_PA_N2, "iter") # 1
diff(range(Y_PA - Y_PA_N2)) # 4.440892e-16, i.e no change

Y_PA_N2i <- ipf2N2(Y_PA, N2N_N2_species = TRUE)
attr(Y_PA_N2i, "iter") # 9
N_occ <- colSums(Y_PA) # number of occurrences of species
N <- nrow(Y_PA)
plot(N_occ, colSums(Y_PA_N2i))
cor(colSums(Y_PA_N2i), N_occ * (N - N_occ)) # 0.9916
mod2 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_PA,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              divideBySiteTotals = FALSE,
              verbose = FALSE)

mod3 <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
              response = Y_PA_N2i,
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
              verbose = FALSE)

mod3$eigenvalues / mod2$eigenvalues
# ratios of eigenvalues greater than 1,
# indicate axes with higher (squared) fourth-corner correlation
```

---

plot.dcca *Plot a single dc-CA axis with CWMs, SNCs, trait and environment scores.*

Description

plot.dcca plots the CWMs and SNCs of a dc-CA axis against this axis, with optional centroids and colors for groups of sites and/or species if available in the data.

Usage

```
## S3 method for class 'dcca'
plot(
  x,
  ...,
  axis = 1,
  gradient_description = "correlation",
  envfactor = NULL,
  traitfactor = NULL,
  nspecies = 20,
  species_groups = NULL,
  widths = c(5, 1, 1),
  newnames = NULL,
  facet = TRUE,
  remove_centroids = FALSE,
  with_lines = 2,
  flip_axis = FALSE,
  expand = 0.2,
  formula = y ~ x,
  verbose = TRUE
)
```

Arguments

`x`	results from `dc_CA` of class `dcca`.
`...`	unused.
`axis`	the axis number to get (default 1).
`gradient_description`	character or 2-character vector for the trait and environmental gradient, respectively specifying what to plot in the vertical line plots to describe the dc-CA axis (trait and environmental gradients). Default: `correlation` for intra-set correlations of both sets of variables with their dc-CA axis. Other values are: `c("weights", "tvalues", "inter_set_correlation")` for regression weights, t-values and inter-set correlation, being the correlation of the SNCs and CWMs with the traits and environmental variables, respectively. `"regression"` is an alias for `"weights"`.
`envfactor`	name of row factor to display as color and lines in the CWM plot (default `NULL`). The default extracts the factor from the environmental model. If set to `NA`, no additional coloring and lines are displayed in `plot.dcca`. The parameter sets the `groups` variable in the `CWM_SNC` data frame of the return value/in the plot.
`traitfactor`	name of column factor to display as color and lines in the SNC plot (default `NULL`). The default extracts the factor from the trait model. If set to `NA`, no additional coloring and lines are displayed in `plot.dcca`. The parameter sets the `groups` variable in the `CWM_SNC` data frame of the return value/in the plot.
`nspecies`	integer. Default `20` for including a vertical species plot with at most `nspecies` that have the highest contribution.
`species_groups`	name of a variable in `dataTraits` of `dc_CA`. Default `NULL` for no grouping. Species groups are colored differentially.
`widths`	relative widths of the CWM-SNC plot, the correlation/weight plot and the species plot. (see `grid.arrange`). Default `c(5, 1, 1)`.
`newnames`	a list with two elements: names for traits and for environmental variables, default `NULL` for names derived from the result of `scores.dcca` with `tidy = TRUE`.
`facet`	logical. Default `TRUE` for CWMs and SNCs plots in separate panels. This parameter changes the position of the centroid names (from left to right for the

	environmental centroids). If `facet = FALSE` and `with_lines = TRUE`, the line fits ignore groups of species and of sites.
`remove_centroids`	logical to remove any centroids from the plot data (default `FALSE`). Can be a two-vector, *e.g.* `c(TRUE, FALSE)` to remove only the trait centroids.
`with_lines`	integer values (0,1,2). Default `2` for straight lines through groups of points, with confidence intervals around the lines. `with_lines=1` drops the confidence intervals and `with_lines=0` suppresses the lines.
`flip_axis`	flip the direction of the axis? (default FALSE).
`expand`	amount of extension of the line plot (default 0.2).
`formula`	formula to use by ggplot geom_smooth (default y~x).
`verbose`	logical. Default `TRUE` for plotting the result.

Details

The current implementation does not distinguish groups of points, if there are two or more factors specified in the model. If you want to label one trait factor, specify `traitfactor="yourfactor"` and similarly specify `envfactor="yourfactor"` for your environmental factor.

No lines are plotted if a single factor defines a model.

If you want to set new names, look at the names with all arguments default, i.e. `myplot <- plot(x)`, and then consult `myplot$nameList$newnames` for the order of the names of traits and environmental variables. Note that covariates should not be in the list of names. Contribution (in the definition of species selection in `nspecies`) is defined (as in CA) as the total species abundance in the (possibly, closed) data multiplied by the square of the score on the axis.

If the `plot.dcca` returns the error `"Error in grid.Call"`, enlarge the plotting area or use `verbose = FALSE` and assign the result.

Value

a ggplot object

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

# must delete "Sites" from response matrix or data frame
Y <- dune_trait_env$comm[, -1] # must delete "Sites"

out <- dc_CA(formulaEnv = ~ A1 + Moist + Use + Manure + Mag,
                   formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
                   response = Y,
                   dataEnv = dune_trait_env$envir,
                   dataTraits = dune_trait_env$traits,
                   verbose = FALSE)
dat <- getPlotdata(out)
```

```
names(dat)
names(dat$CWM_SNC)
levels(dat$CWM_SNC$groups)

plot(out)
```

`plot_dcCA_CWM_SNC` *Plot the CWMs and SNCs of a single dc-CA axis.*

Description

`plot_dcCA_CWM_SNC` plots the CWMs and SNCs of a dc-CA axis against this axis, with optional centroids and colors for groups of sites and/or species if available in the data.

Usage

```
plot_dcCA_CWM_SNC(
  x,
  axis = 1,
  envfactor = NULL,
  traitfactor = NULL,
  facet = TRUE,
  newnames = NULL,
  remove_centroids = FALSE,
  with_lines = 2,
  formula = y ~ x,
  getPlotdata2plotdCCA = NULL
)
```

Arguments

`x`	results from `dc_CA` of class `dcca`.
`axis`	the axis number to get (default 1).
`envfactor`	name of row factor to display as color and lines in the CWM plot (default `NULL`). The default extracts the factor from the environmental model. If set to `NA`, no additional coloring and lines are displayed in `plot.dcca`. The parameter sets the `groups` variable in the `CWM_SNC` data frame of the return value/in the plot.
`traitfactor`	name of column factor to display as color and lines in the SNC plot (default `NULL`). The default extracts the factor from the trait model. If set to `NA`, no additional coloring and lines are displayed in `plot.dcca`. The parameter sets the `groups` variable in the `CWM_SNC` data frame of the return value/in the plot.
`facet`	logical. Default `TRUE` for CWMs and SNCs plots in separate panels. This parameter changes the position of the centroid names (from left to right for the environmental centroids). If `facet = TRUE` and `with_lines = TRUE`, the line fits ignore groups of species and of sites.

newnames	a list with two elements: names for traits and for environmental variables, default NULL for names derived from the result of scores.dcca with tidy = TRUE.
remove_centroids	logical to remove any centroids from the plot data (default FALSE). Can be a two-vector, *e.g.* c(TRUE, FALSE) to remove only the trait centroids.
with_lines	integer values (0,1,2). Default 2 for straight lines through groups of points, with confidence intervals around the lines. with_lines=1 drops the confidence intervals and with_lines=0 suppresses the lines.
formula	formula to use by ggplot geom_smooth (default y~x).
getPlotdata2plotdCCA	the results of an getPlotdata. Default NULL.

Details

The argument getPlotdata2plotdCCA is to allow some modifications of the data frame resulting from getPlotdata. The variable names and score levels should remain untouched. plot_dcCA_CWM_SNC uses the variables: dcCA*k* with axis number *k* and "CWM-SNC", "groups", "points", "sizeweight" for the y-axis, coloring, shape and size of items, respectively.

The current implementation does not distinguish groups of points, if there are two or more factors specified in the model. No lines are plotted if a single factor defines a model.

The function is used in plot.dcca.

Value

a ggplot object

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

# must delete "Sites" from response matrix or data frame
Y <- dune_trait_env$comm[, -1] # must delete "Sites"

out <- dc_CA(formulaEnv = ~ A1 + Moist + Use + Manure + Condition(Mag),
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             response = Y,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

plot_dcCA_CWM_SNC(out, facet = FALSE)
```

`plot_species_scores_bk` *Vertical ggplot2 line plot of ordination scores*

Description

`plot_species_scores_bk` creates a vertical line plot of ordination scores with selection criterion for which scores to plot with names.

Usage

```
plot_species_scores_bk(
  species_scores,
  ylab = "scores",
  threshold = 7,
  y_lab_interval = 0.5,
  speciesname = NULL,
  scoresname = "RDA1",
  selectname = "Fratio1",
  speciesgroup = NULL,
  expand = 0.2,
  verbose = TRUE
)
```

Arguments

`species_scores`	a species-by-scores matrix, a data frame with row names (species names) or a tibble with variable with name `speciesname` containing species names and a column or variable with name `scoresname` containing the scores (default: `"RDA1"`), e.g. species scores from library `vegan`.
`ylab`	y-axis label. Default: $b_k$.
`threshold`	species with criterion (specified by `selectname`) higher than the `threshold` are displayed. Default: `7` (which is the threshold F-ratio for testing a single regression coefficient at `p = 0.01` with `60` df for the error in a multiple regression of each single species onto the condition and the ordination axis). If `selectname` is not in `species_scores`, the `threshold` is divided by `14`, so that the default is 0.5.
`y_lab_interval`	interval of the y-axis ticks. A tick at no effect (0) is always included; default: 0.5.
`speciesname`	name of the variable containing the species names (default `NULL` uses row names).
`scoresname`	name of the column or variable containing the species scores to be plotted (default `"RDA1"`).
`selectname`	name of the column or variable containing the criterion for the selection of species to be displayed. Default: `"Fratio1"`; if `selectname` is not found in `species_scores`, set to `scoresname`.

`speciesgroup`	name of the factor, the levels of which receive different colors in the vertical plot.
`expand`	amount of extension of the line plot (default 0.2).
`verbose`	logical for printing the number of species with names out of the total number (default: `TRUE`).

Details

The absolute value of the criterion values is taken before selection, so that also the species scores of the ordination can serve as a criterion (e.g. `selectname = "RDA1"`). The function has been copied from the `PRC` package at `https://github.com/CajoterBraak/PRC`.

The function is used in `plot.dcca`.

Value

a ggplot object

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

env_scores <- scores(mod, display = "tval")

env_scores <- data.frame(env_scores)
env_scores$group <- c("quantitative", "category")[c(1, 1, 2, 2, 2, 1, 1)]
plot_species_scores_bk(
  species_scores = env_scores,
  ylab = "optimistic t-values",  threshold = 0,  y_lab_interval = 1,
  scoresname = "dcCA1", speciesgroup = "group", verbose = FALSE
)
```

`predict.dcca` *Prediction for double-constrained correspondence analysis (dc-CA)*

Description

Prediction of traits from environment, environment from traits and response from trait and environment data.

Usage

```
## S3 method for class 'dcca'
predict(
  object,
  ...,
 type = c("envFromTraits", "traitsFromEnv", "response", "SNC", "CWM", "lc", "lc_traits"),
  rank = "full",
  newdata = NULL,
  weights = NULL,
  scaling = "symmetric"
)
```

Arguments

`object`	return value of `dc_CA`.
`...`	Other arguments passed to the function (currently ignored).
`type`	type of prediction, `c("envFromTraits", "traitsFromEnv","response", "lc", "lc_traits")` for environmental values, values of traits, response (expected abundance) and constrained scores for sites and species. `"SNC"` is equivalent with `"envFromTraits"`. `"CWM"` is equivalent with `"traitsFromEnv"`.
`rank`	rank (number of axes to use). Default "full" for all axes (no rank-reduction).
`newdata`	Data in which to look for variables with which to predict. For `type = "envFromTraits"` or `"traitsFromEnv"` or `type = "lc_traits"` or `"lc"`, `newdata` is a data frame of trait and environmental values, respectively, which are used for the prediction or the calculation of scores. If omitted, fitted values are generated (use `fitted.dcca` instead). For `type = "response"`, newdata is a list of two data frames with trait and environmental values in this order, *e.g.* `list(traits = dataTraits, env = dataEnv)`.
`weights`	list of weights of species and of sites in `newdata` when `type = "response"`, else ignored (default NULL yielding equal species and site weights, both summing to 1). Example: weights = list(species = c(100, 1, 1), sites = c(1, 1, 1, 1)), in that order, with traits of three new species in newdata[[1]] and environmental values (and levels of factors) of four new sites in newdata[[2]]. Species weights are scaled to a sum of one.
`scaling`	numeric (1,2 or 3) or character `"sites"`, `"species"` or `"symmetric"`. Default: "symmetric". Either site- (1) or species- (2) related scores are scaled by eigenvalues, and the other set of scores have unit weighted mean square or with 3 both are scaled symmetrically to weighted mean squares equal to the square root of eigenvalues. Negative values are treated as the corresponding positive ones by `abs(scaling)`.

Details

Variables that are in the model but not in `newdata` are set to their weighted means in the training data. Predictions are thus at the (weighted) mean of the quantitative variables not included. Predictions with not-included factors are at the reference level (the first level of the factor).

For `type = "response"`, many of the predicted values may be negative, indicating expected absences (0) or small expected response values.

With `type = "traitsFromEnv"` and `newdata = NULL`, predict gives the fitted mean traits, *i.e.* the fitted community weighted means. With `type = "envFromTraits"` and `newdata = NULL`, predict gives the fitted mean environment, *i.e.* the fitted species niche centroids (see `fitted.dcca`). See `fitted.dcca`.

Value

a matrix with the predictions. The exact content of the matrix depends on the `type` of predictions that are being made.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites

mod <- dc_CA(formulaEnv = ~ A1 + Moist + Mag + Use + Condition(Manure),
             formulaTraits = ~ SLA + Height + LDMC + Condition(Seedmass) + Lifespan,
             response = dune_trait_env$comm[, -1],  # must delete "Sites"
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
             verbose = FALSE)

# Ten 'new' sites with a subset of the variables in mod
# X_lot will be ignored as it is not part of the model
newEnv <- dune_trait_env$envir[1:10, c("A1", "Mag", "Manure", "X_lot")]
newEnv[2,"A1"] <- 3.0
rownames(newEnv) <- paste0("NewSite", 1:10)

pred.traits <- predict(mod, type = "traitsFromEnv", newdata = newEnv)
head(pred.traits)


# Eight 'new' species with a subset of traits that are included in the model
# Variable "L" will be ignored as it is not in the model
newTraits <- dune_trait_env$traits[1:8, c("Species", "SLA", "LDMC", "L")]
newTraits[3,"SLA"]<- 18
rownames(newTraits) <- paste("Species",LETTERS[1:8] )# or another meaningful name.

pred.env <- predict(mod, type = "envFromTraits", newdata = newTraits)
head(pred.env)


pred.resp <- predict(mod, type = "response", newdata = list(newTraits, newEnv),
                     weights = list(species = rep(1:2, 4), sites = rep(1, 10)))
colSums(pred.resp) # about alternating 0.8 and 1.6 (reflecting the new species weights)
rowSums(pred.resp) # about equal rowsums
```

predict.wrda	*Prediction from cca0 and wrda models*

Description

Prediction of response and lc scores from environment data using `cca0` and `wrda` models.

Usage

```
## S3 method for class 'wrda'
predict(
  object,
  ...,
  type = c("response", "lc"),
  rank = "full",
  newdata = NULL,
  weights = NULL,
  scaling = "symmetric"
)
```

Arguments

`object`	return value of `cca0` or `wrda`.
`...`	Other arguments passed to the function (currently ignored).
`type`	type of prediction, `c( "response", "lc")` for response (expected abundance) and constrained scores for sites.
`rank`	rank (number of axes to use). Default "full" for all axes (no rank-reduction).
`newdata`	Data in which to look for variables with which to predict.
`weights`	list of weights of species and of sites in `newdata` when `type = "response"`, else ignored (default NULL yielding equal species and site weights, both summing to 1). Example: weights = list(species = c(100, 1, 1), sites = c(1, 1, 1, 1)), in that order, with traits of three new species in newdata[[1]] and environmental values (and levels of factors) of four new sites in newdata[[2]]. Species weights are scaled to a sum of one.
`scaling`	numeric (1,2 or 3) or character `"sites"`, `"species"` or `"symmetric"`. Default: "symmetric". Either site- (1) or species- (2) related scores are scaled by eigenvalues, and the other set of scores have unit weighted mean square or with 3 both are scaled symmetrically to weighted mean squares equal to the square root of eigenvalues. Negative values are treated as the corresponding positive ones by `abs(scaling)`.

Details

Variables that are in the model but not in `newdata` are set to their weighted means in the training data. Predictions are thus at the (weighted) mean of the quantitative variables not included. Predictions with not-included factors are at the reference level (the first level of the factor).

In a cca0 model with type = "response", many of the predicted values may be negative, indicating expected absences (0) or small expected response values.

Value

a matrix with the predictions. The exact content of the matrix depends on the type of predictions that are being made.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"

mod <- cca0(formula = abun ~ A1 + Moist + Mag + Use + Manure,
            data = dune_trait_env$envir)

mod # Proportions equal to those Canoco 5.15

scores(mod, which_cor = c("A1", "X_lot"), display = "cor")

set.seed(123)
anova(mod)
anova(mod, by = "axis")

mod2 <- vegan::cca(abun ~ A1 + Moist + Mag + Use + Manure,
                   data = dune_trait_env$envir)
anova(mod2, by = "axis")

dat <- dune_trait_env$envir
dat$Mag <- "SF"
predict(mod, type = "lc", newdata = dat)
```

---

print.dcca *Print a summary of a dc-CA object.*

Description

Print a summary of a dc-CA object.

Usage

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

Arguments

x	a dc-CA object from dc_CA.
...	Other arguments passed to the function (currently ignored).

Details

`x <- print(x)` is more efficient for `scores.dcca` than just `print(x)` if `dc_CA` is called with `verbose = FALSE`).

Value

No return value, results are printed to console.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"
mod <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
 verbose = FALSE)

print(mod) # same output as with verbose = TRUE (the default of verbose).
anova(mod, by = "axis")
# For more demo on testing, see demo dune_test.r

mod_scores <- scores(mod)
# correlation of axes with a variable that is not in the model
scores(mod, display = "cor", scaling = "sym", which_cor = list(NULL, "X_lot"))

cat("head of unconstrained site scores, with meaning\n")
print(head(mod_scores$sites))

mod_scores_tidy <- scores(mod, tidy = TRUE)
print("names of the tidy scores")
print(names(mod_scores_tidy))
cat("\nThe levels of the tidy scores\n")
print(levels(mod_scores_tidy$score))

cat("\nFor illustration: a dc-CA model with a trait covariate\n")
mod2 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits)

cat("\nFor illustration: a dc-CA model with both environmental and trait covariates\n")
mod3 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
  verbose = FALSE)

cat("\nFor illustration: same model but using dc_CA_object = mod2 for speed, ",
    "as the trait model and data did not change\n")
mod3B <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
               dataEnv = dune_trait_env$envir,
               dc_CA_object = mod2,
   verbose= FALSE)
cat("\ncheck on equality of mod3 (from data) and mod3B (from a dc_CA_object)\n",
    "the expected difference is in the component 'call'\n ")

print(all.equal(mod3[-c(5,12)], mod3B[-c(5,12)])) #  only the component call differs
print(mod3$inertia[-c(3,5),]/mod3B$inertia) #       and mod3 has two more inertia items
```

`print.wrda` *Print a summary of a wrda or cca0 object*

Description

Print a summary of a wrda or cca0 object

Usage

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

Arguments

x	an object from `wrda` or `cca0`
...	Other arguments passed to the function (currently ignored).

Value

No return value, results are printed to console.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
response <- dune_trait_env$comm[, -1]  # must delete "Sites"

w <- rep(1, 20)
w[1:10] <- 8
w[17:20] <- 0.5

object <- wrda(formula = response ~ A1 + Moist + Mag + Use + Condition(Manure),
               data = dune_trait_env$envir,
               weights = w)
object # Proportions equal to those Canoco 5.15
```

```
mod_scores <- scores(object, display = "all")
scores(object, which_cor = c("A1", "X_lot"), display = "cor")
anova(object)
```

---

scores.dcca *Extract results of a double constrained correspondence analysis (dc-CA)*

Description

This function works very much like the vegan `scores` function, in particular `scores.cca`, with the additional results such as regression coefficients and linear combinations of traits (`'reg_traits'`, `'lc_traits'`). All scores from CA obey the so called transition formulas and so do the scores of CCA and dc-CA. The differences are, for CCA, that the linear combinations of environmental variables (the *constrained* site scores) replace the usual (*unconstrained*) site scores, and for dc-CA, that the linear combinations of traits (the *constrained* species scores) also replace the usual (*unconstrained*) species scores in the transition formulas.

Usage

```
## S3 method for class 'dcca'
scores(
  x,
  ...,
  choices = 1:2,
  display = "all",
  scaling = "sym",
  which_cor = "in model",
  normed = TRUE,
  tidy = FALSE
)
```

Arguments

`x`	object of class `"dcca"`, *i.e.* result of `dc_CA`.
`...`	Other arguments passed to the function (currently ignored).
`choices`	integer vector of which axes to obtain. Default: all dc-CA axes.
`display`	a character vector, one or more of `c("all", "species","sites", "sp", "wa", "lc", "bp", "cor", "ic", "reg", "tval", "cn","lc_traits", "reg_traits", "tval_traits", "cor_traits", "ic_traits","bp_traits", "cn_traits")`. The most items are as in `scores.cca`, except `"cor"` and `"ic"`, for inter-set and intra-set correlations, respectively, and `"tval"` for the (over-optimistic) t-values of the regression coefficients. The remaining scores are analogous scores for species and traits.

`scaling`	numeric (1,2 or 3) or character `"sites"`, `"species"` or `"symmetric"`. Default: "symmetric". Either site- (1) or species- (2) related scores are scaled by eigenvalues, and the other set of scores have unit weighted mean square or with 3 both are scaled symmetrically to weighted mean squares equal to the square root of eigenvalues. Negative values are treated as the corresponding positive ones by `abs(scaling)`.
`which_cor`	character or list of trait and environmental variables names (in this order) in the data frames for which inter-set correlations must calculated. Default: a character ("in_model") for all traits and variables in the model, including collinear variables and levels.
`normed`	logical (default `TRUE`) giving standardized regression coefficients and biplot scores. When `FALSE`, (regular) regression coefficients and (unstandardized) biplot scores.
`tidy`	Return scores that are compatible with `ggplot2`: all scores are in a single data.frame, score type is identified by factor variable `score`, the names by variable `label`, and species weights (in `dc_CA` are in variable `weight`. See `scores.cca`.

Details

The function is modeled after `scores.cca`.

The t-ratios are taken from a multiple regression of the unconstrained species (or site) scores on to the traits (or environmental variables).

An example of `which_cor` is: `which_cor = list(traits = "SLA",env = c("acidity", "humidity"))`.

Value

A data frame if `tidy = TRUE`. Otherwise, a matrix if a single item is asked for and a named list of matrices if more than one item is asked for. The following names can be included: `c("sites", "constraints_sites", "centroids", "regression", "t_values","correlation", "intra_set_correlation", "biplot", "species","constraints_species", "regression_traits", "t_values_traits","correlation_traits "intra_set_correlation_traits", "biplot_traits","centroids_traits")`. Each matrix has an attribute `"meaning"` explaining its meaning. With `tidy = TRUE`, the resulting data frame has attributes `"scaling"` and `"meaning"`; the latter has two columns: (1) name of score type and (2) its meaning, usage and interpretation.

An example of the meaning of scores in scaling `"symmetric"` with `display ="all"`:

sites CMWs of the trait axes (constraints species) in scaling 'symmetric' optimal for biplots and, almost so, for inter-site distances.

constraints_sites linear combination of the environmental predictors and the covariates (making the ordination axes orthogonal to the covariates) in scaling 'symmetric' optimal for biplots and, almost so, for inter-site distances.

regression mean, sd, VIF, standardized regression coefficients and their optimistic t-ratio in scaling 'symmetric'.

t_values t-values of the coefficients of the regression of the CWMs of the trait composite on to the environmental variables

correlation inter set correlation, correlation between environmental variables and the sites scores (CWMs)

intra_set_correlation intra set correlation, correlation between environmental variables and the dc-ca axis (constrained sites scores)

biplot biplot scores of environmental variables for display with biplot-traits for fourth-corner correlations in scaling 'symmetric'.

centroids environmental category means of the site scores in scaling 'symmetric' optimal for biplots and, almost so, for inter-environmental category distances.

species SNC on the environmental axes (constraints sites) in scaling 'symmetric' optimal for biplots and, almost so, for inter-species distances.

constraints_species linear combination of the traits and the trait covariates (making the ordination axes orthogonal to the covariates) in scaling 'symmetric' optimal for biplots and, almost so, for inter-species distances.

regression_traits mean, sd, VIF, standardized regression coefficients and their optimistic t-ratio in scaling 'symmetric'.

t_values_traits t-values of the coefficients of the regression of the SNCs along a dc-CA axis on to the traits

correlation_traits inter set correlation, correlation between traits and the species scores (SNCs)

intra_set_correlation_traits intra set correlation, correlation between traits and the dc-ca axis (constrained species scores)

biplot_traits biplot scores of traits for display with biplot scores for fourth-corner correlation in scaling 'symmetric'.

centroids_traits trait category means of the species scores in scaling 'symmetric' optimal for biplots and, almost so, for inter-trait category distances.

The statements on optimality for distance interpretations are based on the `scaling` and the relative magnitude of the dc-CA eigenvalues of the chosen axes.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
abun <- dune_trait_env$comm[, -1]  # must delete "Sites"
mod <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
             formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,
             dataEnv = dune_trait_env$envir,
             dataTraits = dune_trait_env$traits,
 verbose = FALSE)

print(mod) # same output as with verbose = TRUE (the default of verbose).
anova(mod, by = "axis")
# For more demo on testing, see demo dune_test.r

mod_scores <- scores(mod)
# correlation of axes with a variable that is not in the model
scores(mod, display = "cor", scaling = "sym", which_cor = list(NULL, "X_lot"))

cat("head of unconstrained site scores, with meaning\n")
```

```
print(head(mod_scores$sites))

mod_scores_tidy <- scores(mod, tidy = TRUE)
print("names of the tidy scores")
print(names(mod_scores_tidy))
cat("\nThe levels of the tidy scores\n")
print(levels(mod_scores_tidy$score))

cat("\nFor illustration: a dc-CA model with a trait covariate\n")
mod2 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Mag + Use + Manure,
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits)

cat("\nFor illustration: a dc-CA model with both environmental and trait covariates\n")
mod3 <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
              formulaTraits = ~ SLA + Height + LDMC + Lifespan + Condition(Seedmass),
              dataEnv = dune_trait_env$envir,
              dataTraits = dune_trait_env$traits,
  verbose = FALSE)

cat("\nFor illustration: same model but using dc_CA_object = mod2 for speed, ",
    "as the trait model and data did not change\n")
mod3B <- dc_CA(formulaEnv = abun ~ A1 + Moist + Use + Manure + Condition(Mag),
               dataEnv = dune_trait_env$envir,
               dc_CA_object = mod2,
   verbose= FALSE)
cat("\ncheck on equality of mod3 (from data) and mod3B (from a dc_CA_object)\n",
    "the expected difference is in the component 'call'\n ")

print(all.equal(mod3[-c(5,12)], mod3B[-c(5,12)])) #  only the component call differs
print(mod3$inertia[-c(3,5),]/mod3B$inertia) #       and mod3 has two more inertia items
```

`scores.wrda` *Extract results of a weighted redundancy analysis (wrda) or a cca0 object.*

Description

This function works very much like the vegan `scores` function, in particular `scores.cca`, but with regression coefficients for predictors.

Usage

```
## S3 method for class 'wrda'
scores(
  x,
  ...,
  choices = 1:2,
  display = "all",
  scaling = "sym",
  which_cor = "in model",
  normed = TRUE,
  tidy = FALSE
)
```

Arguments

`x`	object of class `"wrda"`, *i.e.* result of `wrda` or `cca0`.
`...`	Other arguments passed to the function (currently ignored).
`choices`	integer vector of which axes to obtain. Default: all wrda axes.
`display`	a character vector, one or more of `c("all", "species","sites", "sp", "wa", "lc", "bp", "cor", "ic", "reg", "tval", "cn")`. The most items are as in `scores.cca`, except `"cor"` and `"ic"`, for inter-set and intra-set correlations, respectively, and `"tval"` for the (over-optimistic) t-values of the regression coefficients.
`scaling`	numeric (1,2 or 3) or character `"sites", "species" or "symmetric"`. Default: "symmetric". Either site- (1) or species- (2) related scores are scaled by eigenvalues, and the other set of scores have unit weighted mean square or with 3 both are scaled symmetrically to weighted mean squares equal to the square root of eigenvalues. Negative values are treated as the corresponding positive ones by `abs(scaling)`.
`which_cor`	character vector environmental variables names in the data frames for which inter-set correlations must calculated. Default: a character ("in_model") for all predictors in the model, including collinear variables and levels.
`normed`	logical (default `TRUE`) giving standardized regression coefficients and biplot scores. When `FALSE`, (regular) regression coefficients and (unstandardized) biplot scores.
`tidy`	Return scores that are compatible with `ggplot2`: all variable `score`, the names by variable `label`. See weights (in `dc_CA` are in variable `weight`. See `scores.cca`.

Details

The function is modeled after `scores.cca`.

An example of which_cor is: `which_cor = c("acidity", "humidity")`

Value

A data frame if `tidy = TRUE`. Otherwise, a matrix if a single item is asked for and a named list of matrices if more than one item is asked for. The following names can be included: `c("sites","constraints_sites", "centroids", "regression", "t_values", "correlation","intra_set_correlation", "biplot", "species")`. Each matrix has an attribute `"meaning"` explaining its meaning. With `tidy = TRUE`, the resulting data frame has attributes `"scaling"` and `"meaning"`; the latter has two columns: (1) name of score type and (2) its meaning, usage and interpretation.

An example of the meaning of scores in scaling `"symmetric"` with `display = "all"`:

sites CMWs of the trait axes (constraints species) in scaling 'symmetric' optimal for biplots and, almost so, for inter-site distances.

constraints_sites linear combination of the environmental predictors and the covariates (making the ordination axes orthogonal to the covariates) in scaling 'symmetric' optimal for biplots and, almost so, for inter-site distances.

regression mean, sd, VIF, standardized regression coefficients and their optimistic t-ratio in scaling 'symmetric'.

t_values t-values of the coefficients of the regression of the CWMs of the trait composite on to the environmental variables

correlation inter set correlation, correlation between environmental variables and the sites scores (CWMs)

intra_set_correlation intra set correlation, correlation between environmental variables and the dc-ca axis (constrained sites scores)

biplot biplot scores of environmental variables for display with biplot-traits for fourth-corner correlations in scaling 'symmetric'.

centroids environmental category means of the site scores in scaling 'symmetric' optimal for biplots and, almost so, for inter-environmental category distances.

species SNC on the environmental axes (constraints sites) in scaling 'symmetric' optimal for biplots and, almost so, for inter-species distances.

The statements on optimality for distance interpretations are based on the `scaling` and the relative magnitude of the dc-CA eigenvalues of the chosen axes.

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
response <- dune_trait_env$comm[, -1]  # must delete "Sites"

w <- rep(1, 20)
w[1:10] <- 8
w[17:20] <- 0.5

object <- wrda(formula = response ~ A1 + Moist + Mag + Use + Condition(Manure),
               data = dune_trait_env$envir,
               weights = w)
object # Proportions equal to those Canoco 5.15

mod_scores <- scores(object, display = "all")
scores(object, which_cor = c("A1", "X_lot"), display = "cor")
anova(object)
```

`wrda` *Performs a weighted redundancy analysis*

Description

`wrda` is formula-based implementation of weighted redundancy analysis.

Usage

```
wrda(
  formula,
  response = NULL,
  data,
  weights = rep(1/nrow(data), nrow(data)),
  traceonly = FALSE,
  cca_object = NULL,
  object4QR = NULL
)
```

Arguments

`formula`	one or two-sided formula for the rows (samples) with row predictors in `data`. The left hand side of the formula is ignored if it is specified in the next argument (`response`). Specify row covariates (if any) by adding + `Condition(covariate-formula)` to `formula` as in `rda`. The `covariate-formula` should not contain a ~ (tilde).
`response`	matrix or data frame of the abundance data (dimension *n* x *m*). Rownames of `response`, if any, are carried through. Can be `NULL` if `cca_object` is supplied or if the response is `formula` is two-sided.
`data`	matrix or data frame of the row predictors, with rows corresponding to those in `response` (dimension *n* x *p*).
`weights`	row weights (a vector). If not specified unit weights are used.
`traceonly`	logical, default `FALSE`. If `TRUE`, only the explained variance of the predictors and the `Condition()` are returned, *i.e* without performing a singular value decompostion.
`cca_object`	a vegan-type cca-object of *transposed* `response`, from which centred abundance values and row and column weights can be obtained.
`object4QR`	a vegan-type cca-object with weighted QR's for `formula`, i.e. `qr(Z)` and `qr(XZ)` obtainable via `get_QR(object4QR, model = "pCCA")` and `get_QR(object4QR, model = "CCA")`, respectively.

Details

The algorithm is a modified version of published R-code for weighted redundancy analysis (ter Braak, 2022).

Compared to `rda`, `wrda` does not have residual axes, *i.e.* no SVD or PCA of the residuals is performed.

Value

All scores in the `wrda` object are in scaling `"sites"` (1): the scaling with *Focus on Case distances*.

References

ter Braak C.J.F. and P. Šmilauer (2018). Canoco reference manual and user's guide: software for ordination (version 5.1x). Microcomputer Power, Ithaca, USA, 536 pp.

Oksanen, J., et al. (2022) vegan: Community Ecology Package. R package version 2.6-4. https://CRAN.R-project.org/package=vegan.

See Also

scores.wrda, anova.wrda, print.wrda

Examples

```
data("dune_trait_env")

# rownames are carried forward in results
rownames(dune_trait_env$comm) <- dune_trait_env$comm$Sites
response <- dune_trait_env$comm[, -1]  # must delete "Sites"

w <- rep(1, 20)
w[1:10] <- 8
w[17:20] <- 0.5

object <- wrda(formula = response ~ A1 + Moist + Mag + Use + Condition(Manure),
               data = dune_trait_env$envir,
               weights = w)
object # Proportions equal to those Canoco 5.15

mod_scores <- scores(object, display = "all")
scores(object, which_cor = c("A1", "X_lot"), display = "cor")
anova(object)
```

Index