Package: analogue (via r-universe)

September 2, 2024

Type Package

Title Analogue and Weighted Averaging Methods for Palaeoecology

Version 0.17-7

Date 2024-09-02

Depends R (>= 3.5.0), vegan (>= 2.2-0)

Imports mgcv, MASS, stats, graphics, grid, brglm, princurve (>= 2.0.2), lattice

Suggests testthat

Maintainer Gavin L. Simpson <ucfagls@gmail.com>

BugReports https://github.com/gavinsimpson/analogue/issues

NeedsCompilation yes

Description Fits Modern Analogue Technique and Weighted Averaging transfer function models for prediction of environmental data from species data, and related methods used in palaeoecology.

License GPL-2

ByteCompile true

URL https://github.com/gavinsimpson/analogue

Repository https://gavinsimpson.r-universe.dev

RemoteUrl https://github.com/gavinsimpson/analogue

RemoteRef HEAD

RemoteSha 0b3ee38b4c637e6622db169253c2e0075147867a

Contents

analogue-package *Analogue and weighted averaging methods for palaeoecology*

Description

analogue is a package for quantitative palaeoecology with a focus on analogue methods, transfer functions, and data handling and display.

Analogue methods

analogue provides functions for analogue matching and the modern analogue technique (MAT) via `analog` and `mat`. A wide range of dissimilarity coefficients are available via the `distance` function.

Additional analysis of modern and no-analogue problems is facilitated via a range of functions implementing many methods from the literature. In particular, the receiver operating characteric (ROC) curves method of Gavin et al (2003) is available in `roc` and a related method employing direct logistic regression modelling (Simpson & Birks, 2012) is available in `logitreg`.

Transfer function methods

Several approaches to fitting transfer function models are provided by **analogue**:

`wa`**:** Simple and tolerance-downweighted weighted averaging with classical, inverse, and monotonic spline deshrinking.

`mat`**:** The modern analogue technique (MAT).

`pcr`**:** Principal components regression with ecologically meaningful transformations

A range of functions for working with and exploring training sets and palaeoenvironmental reconstructions is also included in **analogue**. These include

`crossval` leave-one-out, repeated k-fold, and bootstrap cross-validation methods.

`compare` compare properties of taxa or other proxies across modern and fossil data sets.

`evenSample` are training set samples evenly distributed along the gradient of interest?

`splitSample` splits a gradient into a set of bins or chunks and samples evenly from within each chunk to create a representative test set for cross-validation.

`timetrack` overlays a fossil or secondary data set on to an (constrained) ordination of a modern or reference data set.

`weightedCor` implements the weighted correlation test of a Weighted Averaging reconstruction as proposed by Telford & Birks (2011).

Utilities

analogue provides a range of utilities for working with palaeo data.

`tran` a range of transformations applicable to or commonly used with palaeo data.

`Stratiplot` draws stratigraphic diagrams using the Lattice package.

`join` merging of modern/training set and fossil data sets.

`chooseTaxa` selects taxa that meet certain abundance and occurrence criteria.

Documentation

A full tutorial and worked example for the main features of analogue matching and MAT is avilable in the vignette

`analogue_methods` Analogue Methods in Palaeoecology

Author(s)

Gavin L. Simpson, Jari Oksanen

Maintainer: Gavin L. Simpson <ucfagls@gmail.com>

References

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Simpson, G.L. & Birks H.J.B. (2012) Statistical Learning in Palaeolimnology. In Birks, H.J.B, Lotter, A.F. Juggins S., and Smol, J.P. (Eds) *Tracking Environmental Change Using Lake Sediments, Volume 5: Data Handling and Numerical Techniques*. Springer, Dordrecht.

Telford R.J. and Birks, H.J.B. (2011) A novel method for assessing the statistical significance of quantitative reconstructions inferred from biotic assemblages. *Quanternary Science Reviews* **30**:1272-1278.

`abernethy` *Abernethy Forest Pollen Sequence*

Description

The classic pollen data set from Abernethy Forest in the Scottish highlands, UK. The data originate from the work of Hilary Birks and Rolf Mathewes (1978) and have been analysed in several texts on quantitative numerical palaeoecology.

The data set consists of 36 pollen taxa from 49 levels, with two additional variables; `Age`, the age of each sample, and `Depth` the depth (in cm) below the surface of the peat sequence from which the core was taken.

Usage

```
data(abernethy)
```

Format

`abernethy` is a data frame with 49 samples on 36 species plus sample Age and Depth (in cm).

Source

These data were provided in electronic format by Prof. H. John B. Birks. The original source is Birks and Mathewes (1978).

References

Birks, H.H. and Mathewes, R.W. (1978) Studies in the vegetational history of Scotland. *New Phytologist* **80**, 455-484.

Examples

```
data(abernethy)
head(abernethy)

(plt <- Stratiplot(Age ~ . - Depth,
                   data = chooseTaxa(abernethy, n.occ = 5, max.abun = 10),
                   type = "poly"))
```

analog *Analogue matching*

Description

Analogue matching is a more general implementation of the modern analogue methodology than MAT, where we are only interested in identifying sufficiently similar samples from a modern training as being suitable modern analogues for one or more fossil samples.

Usage

```
analog(x, ...)

## Default S3 method:
analog(x, y,
       method = c("euclidean", "SQeuclidean", "chord", "SQchord",
                  "bray", "chi.square", "SQchi.square",
                  "information", "chi.distance", "manhattan",
                  "kendall", "gower", "alt.gower", "mixed"),
       keep.train = TRUE, ...)

## S3 method for class 'distance'
analog(x, train = NULL, keep.train = TRUE, ...)
```

Arguments

`x, y`	data frames with same columns. `x` is training data and `y`, the test data.
`method`	character string naming the dissimilarity methods to be used. See Details below.
`keep.train`	logical; should the dissimilarity matrix for the training set be stored?
`train`	a pre-computed dissimilarity matrix for the training set samples. Objects of classes `"dist"`, `"vegdist"`, and `"distance"` are currently accepted.
`...`	arguments passed to or from other methods.

Details

`analog` implements analogue matching *sensu* Flower et al (1997) and Simpson et al (2005), where the aim is to identify suitable close analogues of fossil samples from a modern training set. These results are generally used within ecological restoration, but the identification of close modern analogues for fossil samples is also used as a technique for assessing transfer function reconstructions.

`analog` is a simple and very general function that generates a pairwise dissimilarity matrix for the modern training set, and a second matrix containing the pairwise dissimilarities between each fossil sample and each sample in the training set. These results can then be assessed using other functions and to extract the close modern analogues using function `cma`. See the See Also section below.

Analysis of the pairwise dissimilarity matrix for the modern training set can be used to decide on a suitable dissimilarity threshold for defining close modern analogues. By default this matrix is returned as part of the output from the `analog` function.

Value

A list of class "analog" with the following components:

`analogs`	matrix of pairwise dissimilarities between each fossil sample (`y`) and each sample in the modern training set (`x`).
`train`	if argument `keep.train` is `TRUE` then a pairwise dissimilarity matrix for the modern training set.
`call`	the matched function call.
`method`	character; the dissimilarity coefficient used.

Author(s)

Gavin L. Simpson

References

Flower, R.J., Juggins, S. and Battarbee, R.W. (1997) Matching diatom assemblages in lake sediment cores and modern surface sediment samples: the implications for lake conservation and restoration with special reference to acidified systems. *Hydrobiologia* **344**; 27–40.

Simpson, G.L., Shilland, E.M., Winterbottom, J. M. and Keay, J. (2005) Defining reference conditions for acidified waters using a modern analogue approach. *Environmental Pollution* **137**; 119–133.

See Also

`distance` for the function that calculates the dissimilarity matrices. `cma` for extraction of close modern analogues. `dissimilarities` and `plot.dissimilarities` for analysis of distribution of pairwise dissimilarity matrix for modern training set.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## Imbrie and Kipp foraminfera sea-surface temperature

## analog matching between SWAP and RLGH core
ik.analog <- analog(ImbrieKipp, V12.122, method = "chord")
ik.analog
summary(ik.analog)
```

```
## Can take pre-computed dissimilarity objects
d1 <- distance(ImbrieKipp, V12.122)
d2 <- distance(ImbrieKipp)
ik <- analog(d1, d2, keep.train = TRUE)
ik
```

`bayesF` *Bayes factors*

Description

Calculates Bayes factors or likelihood ratios of analogue and no-analogue results.

Usage

```
bayesF(x, prior = rep(0.5, 2))

## S3 method for class 'bayesF'
plot(x, group = "all", xlab = NULL, ylab = "Pr (A+ | d)",
        col = "red", abline.col = "lightgrey", abline.lty = "dashed", ...)
```

Arguments

`x`	for `bayesF` an object of class `roc`. For the plot method, an object of class `bayesF`, usually the result of a call to `bayesF`.
`prior`	numeric; the prior probabilities of analogue and no-analogue, provided as a vector of length 2 whose elements sum to 1. If not provided, the function will use the relative occurences of analogue and no analogue situations used to evaluate the ROC curve.
`group`	character vector of length 1 giving the name of the group to plot, or `"all"` to plot all groups in `x`.
`xlab, ylab`	the x- and y-axis labels for the plot.
`col`	colour of the line used to draw the posterior probability.
`abline.col`	colour of the vertical line drawn to indicate the optimal dissimilarity determined from the ROC curve.
`abline.lty`	Line type for indicator of optimal ROC dissimilarity threshold. See `par` for the allowed line types.
`...`	other plot arguments passed to plotting functions. Currently ignored.

Details

LR(+), is the likelihood ratio of a positive test result, that the value of d assigns the sample to the group it belongs to. LR(-) is the likelihood ratio of a negative test result, that the value of d assigns the sample to the wrong group.

LR(+) is defined as $LR(+) = TPF/FPF$ (or sensitivity / (1 - specificity)), and LR(-) is defined as $LR(-) = FPF/TNF$ (or (1 - sensitivity) / specificity), in Henderson (1993).

The posterior probability of analogue given a dissimilarity is the LR(+) likelihood ratio values multiplied by the prior odds of analogue, for given values of the dissimilarity, and is then converted to a probability.

The plotting function currently only draws the posterior probability of analogue based on the Bayes factor or likelihood ratio of a positive event (analogue).

Value

For `plot.bayesF` a plot on the currently active device.

For `bayesF`, a list containing the results of computing Bayes factors for each group in `x`. Each component of this list is itself a list with the following components:

`bayesF`, `posterior.odds`, `posterior.probs`, `prior.prob`	Bayes factors, posterior odds and probabilities and prior probabilities of true analogue and true non-analogue events. Each components is a list with two components; `pos` (for true analogue events) and `neg` (for true non-analogue events). The components of `prior.prob` are vectors of length 1, whilst components of the other lists are numeric vectors.
`roc.points`	numeric; the points at which the ROC curve was evaluated.
`optimal`	numeric; the optimal dissimilarity as assessed by the ROC curve.
`max.roc`	numeric; the position along the ROC curve at which the slope of the ROC curve is maximal. This is the index of this point on the curve, and can be used to extract the element of `bayesF`, `posterior.odds` and `posterior.probs` for the optimal dissimilarity.

Author(s)

Gavin L. Simpson

References

Brown, C.D., and Davis, H.T. (2006) Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems* **80**, 24–38.

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Henderson, A.R. (1993) Assessing test accuracy and its clinical consequences: a primer for receiver operating characteristic curve analysis. *Annals of Clinical Biochemistry* **30**, 834–846.

See Also

`roc` and `plot.bayesF`.

Examples

```
## load the example data
data(swapdiat, swappH, rlgh)

## merge training and test set on columns
dat <- join(swapdiat, rlgh, verbose = TRUE)

## extract the merged data sets and convert to proportions
swapdiat <- dat[[1]] / 100
rlgh <- dat[[2]] / 100

## fit an analogue matching (AM) model using the squared chord distance
## measure - need to keep the training set dissimilarities
swap.ana <- analog(swapdiat, rlgh, method = "SQchord",
                   keep.train = TRUE)

## fit the ROC curve to the SWAP diatom data using the AM results
## Generate a grouping for the SWAP lakes
METHOD <- if (getRversion() < "3.1.0") {"ward"} else {"ward.D"}
clust <- hclust(as.dist(swap.ana$train), method = METHOD)
grps <- cutree(clust, 12)

## fit the ROC curve
swap.roc <- roc(swap.ana, groups = grps)
swap.roc

## calculate the Bayes factors of analogue and no-analogue
## (uses observed probabilities of analogue/no-analogue
swap.bayes <- bayesF(swap.roc)
swap.bayes

## plot the probability of analogue
plot(swap.bayes)

## Not run:
## calculate the Bayes factors of analogue and no-analogue
## with prior probabilities c(0.5, 0.05)
swap.bayes2 <- bayesF(swap.roc, prior = c(0.5, 0.05))
swap.bayes

## plot the probability of analogue
plot(swap.bayes2)

## End(Not run)
```

bootstrap *Bootstrap estimation and errors*

Description

Function to calculate bootstrap statistics for transfer function models such as bootstrap estimates, model RMSEP, sample specific errors for predictions and summary statistics such as bias and R^2 between oberved and estimated environment.

`residuals` method for objects of class `"bootstrap.mat"`.

Usage

```
bootstrap(object, ...)

## Default S3 method:
bootstrap(object, ...)

## S3 method for class 'mat'
bootstrap(object, newdata, newenv, k,
          weighted = FALSE, n.boot = 1000, ...)

## S3 method for class 'bootstrap.mat'
fitted(object, k, ...)

## S3 method for class 'bootstrap.mat'
residuals(object, which = c("model", "bootstrap"), ...)
```

Arguments

`object`	an R object of class `"mat"` for which bootstrap statistics are to be generated, or an object of class `"bootstrap.mat"` from which fitted values or residuals are extracted.
`newdata`	a data frame containing samples for which bootstrap predictions and sample specific errors are to be generated. May be missing — See Details. `"newdata"` must have the same number of columns as the training set data.
`newenv`	a vector containing environmental data for samples in `"newdata"`. Used to calculate full suite of errors for new data such as a test set with known environmental values. May be missing — See Details. `"newenv"` must have the same number of rows as `"newdata"`.
`k`	numeric; how many modern analogues to use to generate the bootstrap statistics (and, if requested, the predictions), fitted values or residuals.
`weighted`	logical; should the weighted mean of the environment for the `"k"` modern analogues be used instead of the mean?
`n.boot`	Number of bootstrap samples to take.
`which`	character; which set of residuals to return, the model residuals or the residuals of the bootstrap-derived estimates?
`...`	arguments passed to other methods.

Details

`bootstrap` is a fairly flexible function, and can be called with or without arguments `newdata` and `newenv`.

If called with only `object` specified, then bootstrap estimates for the training set data are returned. In this case, the returned object will not include component `predictions`.

If called with both `object` and `newdata`, then in addition to the above, bootstrap estimates for the new samples are also calculated and returned. In this case, component `predictions` will contain the apparent and bootstrap derived predictions and sample-specific errors for the new samples.

If called with `object`, `newdata` and `newenv`, then the full `bootstrap` object is returned (as described in the Value section below). With environmental data now available for the new samples, residuals, RMSE(P) and R^2 and bias statistics can be calculated.

The individual components of `predictions` are the same as those described in the components relating to the training set data. For example, `returned.object$predictions$bootstrap` contains the components as `returned.object$bootstrap`.

It is not usual for environmental data to be available for the new samples for which predictions are required. In normal palaeolimnological studies, it is more likely that `newenv` will not be available as we are dealing with sediment core samples from the past for which environmental data are not available. However, if sufficient training set samples are available to justify producing a training and a test set, then `newenv` will be available, and `bootstrap` can accomodate this extra information and calculate apparent and bootstrap estimates for the test set, allowing an independent assessment of the RMSEP of the model to be performed.

Typical usage of `residuals` is

```
resid(object, which = c("model", "bootstrap"), \dots)
```

Value

For `bootstrap.mat` an object of class `"bootstrap.mat"` is returned. This is a complex object with many components and is described in `bootstrapObject`.

For `residuals`, a list containg the requested residuals and metadata, with the following components:

`model`	Leave one out residuals for the MAT-estimated model.
`bootstrap`	residuals for the bootstrapped MAT model.
`k`	numeric; indicating the size of model used in estimates and predictions.
`n.boot`	numeric; the number of bootstrap samples taken.
`auto`	logical; whether `"k"` was choosen automatically or user-selected.
`weighted`	logical; whether the weighted mean was used instead of the mean of the environment for *k*-closest analogues.

Author(s)

Gavin L. Simpson

References

Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C. and ter Braak, C.J.F. (1990). Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London; Series B*, **327**; 263–278.

See Also

`mat`, `plot.mat`, `summary.bootstrap.mat`, `residuals`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## Imbrie and Kipp foraminfera sea-surface temperature
## fit the MAT model using the squared chord distance measure
ik.mat <- mat(ImbrieKipp, SumSST, method = "SQchord")

## bootstrap training set
## IGNORE_RDIFF_BEGIN
ik.boot <- bootstrap(ik.mat, n.boot = 100)
ik.boot
summary(ik.boot)
## IGNORE_RDIFF_END

## Bootstrap fitted values for training set
## IGNORE_RDIFF_BEGIN
fitted(ik.boot)
## IGNORE_RDIFF_END

## residuals
resid(ik.boot) # uses abbreviated form
```

`bootstrap.wa` *Bootstrap estimation and errors for WA models*

Description

Function to calculate bootstrap statistics for transfer function models such as bootstrap estimates, model RMSEP, sample specific errors for predictions and summary statistics such as bias and R^2 between oberved and estimated environment.

Usage

```
## S3 method for class 'wa'
bootstrap(object, n.boot = 1000, verbose = TRUE, ...)
```

Arguments

`object`	an R object of class `"wa"` for which bootstrap statistics are to be generated.
`n.boot`	numeric; the number of bootstrap samples to draw.
`verbose`	logical; should bootstrap progress be printed to the console?
`...`	arguments passed to other methods.

Details

See `bootstrap.mat` for further details. This method is not as feature packed as `bootstrap.mat` but can be used to evaluate the model performance of WA transfer function models.

Value

An object with the same components as `predict.wa`.

Author(s)

Gavin L. Simpson

References

Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C. and ter Braak, C.J.F. (1990). Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London; Series B*, **327**; 263–278.

See Also

`wa`, `plot.wa`.

Examples

```
## Imbrie and Kipp
data(ImbrieKipp)
data(SumSST)
ik.wa <- wa(SumSST ~ ., data = ImbrieKipp, tol.dw = TRUE,
            min.tol = 2, small.tol = "min")
ik.wa

## compare actual tolerances to working values
with(ik.wa, rbind(tolerances, model.tol))

## bootstrap the WA model
ik.boot <- bootstrap(ik.wa, n.boot = 100)

## performance statistics
performance(ik.boot)
```

`bootstrapObject` *Bootstrap object description*

Description

Objects of class `bootstrap.mat` are a complex containing many sub-components. This object is described here in more detail.

Details

A large object is returned with some or all of the following depending on whether `newdata` and `newenv` are supplied or not.

- **`observed`:** vector of observed environmental values.
- **`model`:** a list containing the apparent or non-bootstrapped estimates for the training set. With the following components:
  - **`estimated`:** estimated values for the response
  - **`residuals`:** model residuals
  - **`r.squared`:** Apparent R^2 between observed and estimated values of response
  - **`avg.bias`:** Average bias of the model residuals
  - **`max.bias`:** Maximum bias of the model residuals
  - **`rmse`:** Apparent error (RMSE) for the model.
  - **`k`:** numeric; indicating the size of model used in estimates and predictions
- **`bootstrap`:** a list containing the bootstrap estimates for the training set. With the following components:
  - **`estimated`:** Bootstrap estimates for the response
  - **`residuals`:** Bootstrap residuals for the response
  - **`r.squared`:** Bootstrap derived R^2 between observed and estimated values of the response
  - **`avg.bias`:** Average bias of the bootstrap derived model residuals
  - **`max.bias`:** Maximum bias of the bootstrap derived model residuals
  - **`rmsep`:** Bootstrap derived RMSEP for the model
  - **`s1`:** Bootstrap derived S1 error component for the model
  - **`s2`:** Bootstrap derived S2 error component for the model
  - **`k`:** numeric; indicating the size of model used in estimates and predictions
- **`sample.errors`:** a list containing the bootstrap-derived sample specific errors for the training set. With the following components:
  - **`rmsep`:** Bootstrap derived RMSEP for the training set samples
  - **`s1`:** Bootstrap derived S1 error component for training set samples
  - **`s2`:** Bootstrap derived S2 error component for training set samples
- **`weighted`:** logical; whether the weighted mean was used instead of the mean of the environment for *k*-closest analogues
- **`auto`:** logical; whether `"k"` was choosen automatically or user-selected

`n.boot`**:** numeric; the number of bootstrap samples taken

`call`**:** the matched call

`type`**:** model type

`predictions`**:** a list containing the apparent and bootstrap-derived estimates for the new data, with the following components:

- `observed`**:** the observed values for the new samples — only if `newenv` is provided
- `model`**:** a list containing the apparent or non-bootstrapped estimates for the new samples. A list with the same components as `model`, above
- `bootstrap`**:** a list containing the bootstrap estimates for the new samples, with some or all of the same components as `bootstrap`, above
- `sample.errors`**:** a list containing the bootstrap-derived sample specific errors for the new samples, with some or all of the same components as `sample.errors`, above

Author(s)

Gavin L. Simpson

See Also

`mat`, `plot.mat`, `summary.bootstrap.mat`, `residuals`

`caterpillarPlot` *Caterpillar plot of species' WA optima and tolerance range.*

Description

Draws a caterpillar plot of the weighted average optima and tolerance range for each of the species in a data set.

Usage

```
## Default S3 method:
caterpillarPlot(x, tol, mult = 1, decreasing = TRUE,
                labels, xlab = NULL, pch = 21, bg = "white",
                col = "black", lcol = col, lwd = 2, frame.plot = FALSE, ...)

## S3 method for class 'data.frame'
caterpillarPlot(x, env, useN2 = TRUE, xlab, ...)

## S3 method for class 'wa'
caterpillarPlot(x, type = c("observed","model"), ...)
```

Arguments

`x`	For the `default` method, a numeric vector of species optima. For the `data.frame` method a species data matrix or data frame. For the `wa` method an object of class `"wa"`.
`tol`	numeric; vector of species tolerances.
`env`	numeric; variable for which optima and tolerances are required.
`useN2`	logical; should Hill's N2 values be used to produce un-biased tolerances?
`decreasing`	logical; should the sort order of the species be increasing or decreasing?
`mult`	numeric; multiplication factor for species' tolerances.
`labels`	character; vector of labels for the species names with which to annotate the y-axis. If missing, `names(x)` is used.
`xlab`	character; the x-axis label. If `NULL`, the default, a description of `env` is used.
`pch, bg, col`	The plotting character to use and its background and foreground colour. See `par`.
`lcol, lwd`	The colour and line width to use for the tolerance range.
`type`	character; `"observed"` uses the actual tolerances observed from the data. `"model"` uses the tolerances used in the WA model where very small tolerances have been reset for some definition of small.
`frame.plot`	logical; should a box be drawn round the plot?
`...`	Additional graphical arguments to be passed on to plotting functions.

Details

The function may also be called using the short form name `caterpillar`:

```
caterpillar(x, ...)
```

Value

The function results in a plot on the currently active device. A data frame with components `Optima` and `Tolerance` is returned invisibly.

Author(s)

Gavin L. Simpson

See Also

For the underlying computations `optima` and `tolerance`.

Examples

```
data(ImbrieKipp)
data(SumSST)

## default plot
caterpillar(ImbrieKipp, SumSST)

## customisation
opttol <-
    caterpillar(ImbrieKipp, SumSST, col = "red2",
                bg = "yellow", lcol = "blue",
                xlab = expression(Summer ~ Sea ~ Surface ~
                                  Temperature~(degree*C)))

## invisibly returns the optima and tolerances
head(opttol)
```

`chooseTaxa` *Select taxa (variables) on basis of maximum abundance attained and number of occurrences.*

Description

Select taxa (variables) from an object on the basis of one or both of maximum abundance and number of occurrences greater than user-specified values. This is a simple utility function to encapsulate this common task in filtering palaeoecological data sets.

Usage

```
chooseTaxa(object, ...)

## Default S3 method:
chooseTaxa(object, n.occ = 1, max.abun = 0,
           type = c("AND","OR"), value = TRUE, na.rm = FALSE,
           ...)
```

Arguments

`object`	an R object for which a suitable method exists. The default method assumes a matrix-like object such as a data frame or a numeric matrix.
`n.occ`	numeric; number of occurrences representing the lower limit for selection. A taxon is included in the returned subset if it is present a total of `n.occ` times or more. See argument `type` for a modifier which might exclude the taxon even if it would be included on the basis of `n.occ`.
`max.abun`	numeric; maximum abundance representing the lower limit for selection. A taxon is included in the returned subset if it attains abundance equal to or greater than `max.abun` in one or more sample. See argument `type` for a modifier which might exclude the taxon even if it would be included on the basis of `max.abun`.

`type`	character; one of `"AND"` or `"OR"`, controlling how the criteria `n.occ` and `max.abun` are combined to generate a subset of the variables in `object`.
`value`	logical; should the data for the selected taxa be returned? If `TRUE`, the default, the data for the chosen taxa are returned. If `FALSE`, a logical vector is returned, indicating which taxa met the selection criteria.
`na.rm`	logical; should missing values `NA`s be excluded from the calculation of abundances and occurrence?
`...`	arguments passed on to subsequent methods.

Value

If `value = TRUE`, returns the supplied data frame or matrix with a subset of columns (taxa) that meet the criteria chosen. If `value = FALSE`, a logical vector is returned.

Author(s)

Gavin L. Simpson

Examples

```
data(ImbrieKipp)
IK2 <- chooseTaxa(ImbrieKipp, n.occ = 5)
dim(ImbrieKipp)
dim(IK2)

## return a logical vector to select species/columns
chooseTaxa(ImbrieKipp, n.occ = 5, value = FALSE)
```

`cma` *Close modern analogues*

Description

Extracts and formats close modern analogue samples from a modern reference set that are closer than a defined cut off threshold.

Usage

```
cma(object, ...)

## Default S3 method:
cma(object, ...)

## S3 method for class 'analog'
cma(object, cutoff, prob = c(0.01, 0.025, 0.05), ...)

## S3 method for class 'mat'
```

```
cma(object, k, cutoff, prob = c(0.01, 0.025, 0.05), ...)

## S3 method for class 'predict.mat'
cma(object, k, cutoff, prob = c(0.01, 0.025,
0.05), ...)

## S3 method for class 'cma'
plot(x, method = c("overplot", "jitter", "stack"),
   jitter = 0.1, vertical = FALSE,
   draw.quant = TRUE, xlab = NULL, ylab = "",
   main = "", cex.axis = NULL, ...,
   col.quant = "red", lty.quant= "dashed")
```

Arguments

`object`	an object for which close modern analogues are to be returned. Currently only for objects of class `analog`.
`k`	numeric; the number of analogues to return.
`cutoff`	numeric; critical value determining level below which samples from the modern reference set are defined as close modern analogues. May be missing, in which case the 2.5% quantile of the training set dissimilarities is used unless `object$train` is `NULL`, in which case `"cutoff"` must be supplied.
`prob`	numeric vector of probabilities with values in [0,1], for which quantiles of the distribution of training set dissimilarities will be calculated. See `quantile`.
`...`	arguments to be passed to other `cma` methods or additional arguments passed to `stripchart`.
`x`	an object of class `"cma"`.
`method`	the method to be used to separate coincident points. The default method `"overplot"` causes such points to be overplotted, but it is also possible to specify `"jitter"` to jitter the points, or `"stack"` have coincident points stacked. The last method only makes sense for very granular data.
`jitter`	when `method="jitter"` is used, `jitter` gives the amount of jittering applied.
`vertical`	when vertical is `TRUE` the plots are drawn vertically rather than the default horizontal.
`draw.quant`	logical; should the quantiles be drawn on the stripchart?
`xlab, ylab, main`	Graphical parameters
`cex.axis`	The magnification to be used for axis annotation relative to the current setting of `cex`. See `par`.
`col.quant, lty.quant`	colour and line type in which to drawn the quantile lines.

Details

The plot method is simply a wrapper to `stripchart`.

The methods for `mat` and `predict.mat` objects allow the user to select the k-closest analogues (argument `k`) or those samples as close or closer than a stated threshold of dissimilarity (argument cutoff). Only one of k and cutoff may be specified. If neither is specified, getK is used to extract the value for k stored within object. As such, the default is to return the automatically selected set of k closest samples, behaviour that is consistent with other functions in the package.

Value

For the plot method, a plot on the current device. Invisibly the plotted data are returned; see Note for further details.

A list of class "cma" with the following components:

close	a named list of named vectors of close modern analogues and their dissimilarities. The names of the list components are the names of the fossil samples. The named vector in each component of close is the distances for the close modern analogues from the training set that are as close as cutoff, or closer, to the fossil sample.
call	the matched call.
cutoff	the cutoff threshold used to define close modern analogues.
quant	numeric vector of the requested quantiles. Note returned by the predict.mat method.
probs	the probabilities of the requested quantiles.
method	character; the dissimilarity coefficient used
n.analogs	numeric vector of the number of analogues per fossil sample.

Note

Only objects of classes analog, mat, and predict.mat are supported.

The plot method invisibly returns a list with the following components:

distances a vector of stacked distances extracted from object.

groups a factor listing the fossil sample for which the distances are the distances to the close modern analogues for the training set.

Author(s)

Gavin L. Simpson

References

Flower, R.J., Juggins, S. and Battarbee, R.W. (1997) Matching diatom assemblages in lake sediment cores and modern surface sediment samples: the implications for lake conservation and restoration with special reference to acidified systems. *Hydrobiologia* **344**; 27–40.

Simpson, G.L., Shilland, E.M., Winterbottom, J. M. and Keay, J. (2005) Defining reference conditions for acidified waters using a modern analogue approach. *Environmental Pollution* **137**; 119–133.

See Also

analog, stripchart, or boxplot for an alternative representation.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## analog matching between SWAP and RLGH reference samples
(ik.ana <- analog(ImbrieKipp, V12.122, method = "chord"))

## close modern analogues
(ik.cma <- cma(ik.ana, cutoff = 0.4))
summary(ik.cma)

## plot the results
plot(ik.cma)
```

---

compare *Compare proxies across two data sets*

Description

`compare()` compares a proxy dataset with a training set or other data set that is considered to be the master. A range of metrics is returned, currently for samples only.

Usage

```
compare(x, ...)

## Default S3 method:
compare(x, y, env,
        by = c("sites", "species"),
        ordination = "rda",
        method = "chord",
        transform = NULL,
        n2limit = 5L,
        ...)
```

Arguments

`x`	data frame; training set samples to compare against
`y`	data frame; passive or core samples
`env`	numeric vector of environmental or contraint data for residual length ordination. Ignored if `by = "species"`.
`by`	character; compare data sets by sites or species (proxies).
`ordination`	character; which constrained ordination method to use
`method`	character; which dissimilarity method to use. See `distance`.
`transform`	character: should a transformation be applied to the data. Ignored.
`n2limit`	integer; the criterion for indicating species with potentially poorly estimated optima. The default value of `5L` is one suggested by R. Telford.
`...`	arguments passed to other methods.

Details

ToDo

Value

If `by = "species"` a data frame of diagnostics for each species (proxy) in `y` relative to `x`. If `by = "sites"`, the diagnostics are for each sample (row) in `y`. Depending on the value of `by` some of the following columns will be returned

`sumMissing`	numeric; abundance sum for species missing from the training set `x`.
`sumPoorOpt`	numeric; abundance sum for species with potentially poorly estimated optima.
`closestSamp`	numeric; minimum dissimilarity to a sample in the training data `x`.
`residLen`	numeric; the squared residual length for each sample in `y`. A measure of how well the sample fits into the species-environment relationship of a constrained ordination. See `residLen` for details. Not given if `env` is not provided.
`inTrain`	logical; simple indicator of whether a species in `y` is present in the training data `x`.
`n2`	numeric; Hill's N2 for each species in `y`.
`n2Train`	numeric; as for `n2` but computed from the training data `x`.
`max`	numeric; the maximum abundance of each species computed using `y`.
`maxTrain`	numeric; as for `max` but computed using the training data `x`.

Author(s)

Gavin L. Simpson

Examples

```
data(ImbrieKipp, V12.122, SumSST)
compare(ImbrieKipp, V12.122, env = SumSST, ordination = "rda",
        method = "chord")
```

`crossval` *Cross-validation of palaeoecological transfer function models*

Description

Performs leave-one-out, *k*-fold, *n* *k*-fold and bootstrap cross-validation of palaeoecological transfer function models.

Usage

```
crossval(obj, ...)

## S3 method for class 'wa'
crossval(obj, method = c("LOO","kfold","bootstrap"),
         nboot = 100, nfold = 10, folds = 5,
         verbose = getOption("verbose"), ...)

## S3 method for class 'pcr'
crossval(obj, method = c("LOO","kfold","bootstrap"),
         ncomp, nboot = 100, nfold = 10, folds = 5,
         verbose = getOption("verbose"), ...)
```

Arguments

`obj`	A fitted transfer function model. Currently, only objects of class `wa` and `pcr` are supported.
`method`	character; type of cross-validation.
`ncomp`	numeric; number of components to fit, as in models with `1:ncomp` components.
`nboot`	numeric; number of bootstrap samples.
`nfold`	numeric; number of chunks into which the training data are split. The *k* in *k*-fold.
`folds`	numeric; the number of times *k*-fold CV is performed.
`verbose`	logical; should progress of the CV be displayed?
`...`	Arguments passed to other methods.

Value

Returns an object of class `"crossval"`, a list with the following components:

`fitted.values`	numeric vector; the cross-validated estimates of the response.
`residuals`	numeric vector; residuals computed from the cross-validated estimates of the response.
`performance`	data frame; cross-validation performance statistics for the model.
`CVparams`	list; parameters holding details of the cross-validation process.
`call`	the matched call.

Author(s)

Gavin L. Simpson

See Also

`wa`

Examples

```
## Load the Imbrie & Kipp data and
## summer sea-surface temperatures
data(ImbrieKipp)
data(SumSST)

## fit the WA model
mod <- wa(SumSST ~., data = ImbrieKipp)
mod

## Leave one out CV
cv.loo <- crossval(mod)
cv.loo

## k-fold CV (k == 10)
cv.kfold <- crossval(mod, method = "kfold", kfold = 10, folds = 1)
cv.kfold

## n k-fold CV (k == 10, n = 10)
cv.nkfold <- crossval(mod, method = "kfold", kfold = 10, folds = 10)
cv.nkfold

## bootstrap with 100 bootstrap samples
cv.boot <- crossval(mod, method = "bootstrap", nboot = 100)
cv.boot

## extract fitted values and residuals
fitted(cv.boot)
resid(cv.boot)

## Principal Components Regression
mpcr <- pcr(SumSST ~., data = ImbrieKipp, ncomp = 10)
crossval(mpcr, method = "kfold", kfold = 10, folds = 2, ncomp = 10)

crossval(mpcr, method = "bootstrap", nboot = 100, ncomp = 10)
```

`densityplot.residLen` *Lattice density plot for residual lengths*

Description

Lattice `densityplot` method for `residLen` objects.

Usage

```
## S3 method for class 'residLen'
densityplot(x, ..., xlab = NULL, ylab = NULL)
```

Arguments

`x`	Object of class `"residLen"`, the result of a call to `residLen`.
`xlab, ylab`	Axis labels. If not supplied, suitable defaults are generated, depending on whether RDA or CCA was used as the underlying ordination model.
`...`	Additional arguments passed to `densityplot`.

Value

Returns an object of class `"trellis"`. See `densityplot` for details.

Author(s)

Gavin L. Simpson

See Also

`residLen`, `plot.residLen`, `hist.residLen`, `histogram.residLen`.

Examples

```
## load the Imbrie and Kipp example data
data(ImbrieKipp, SumSST, V12.122)

## squared residual lengths for Core V12.122
rlens <- residLen(ImbrieKipp, SumSST, V12.122)
rlens

## plot the density functions of the residual distances
densityplot(rlens)
```

`deshrink` *Deshrinking techniques for WA transfer functions*

Description

In Weighted Averaging models averages are taken twice and thus WA estimates shrink towards the training set mean and need to be deshrunk.`deshrink` performs this deshrinking using several techniques, whilst `deshrinkPred` will deshrink WA estimates for new samples given a set of deshrinking coefficients.

Usage

```
deshrink(env, wa.env,
         type = c("inverse", "classical", "expanded", "none",
                  "monotonic"))

deshrinkPred(x, coef,
         type = c("inverse", "classical", "expanded", "none",
                  "monotonic"))
```

Arguments

`env`	numeric; original environmental values.
`wa.env`	numeric; initial weighted average estimates.
`type`	character; the type of deshrinking. One of `"inverse"`, `"classical"`, `"expand"`, `"none"`.
`x`	numeric; estimates to be deshrunk.
`coef`	numeric; deshrinking coefficients to use. Currently needs to be a vector of length 2. These should be supplied in the order β_0, β_1.

Value

For `deshrinkPred` a numeric vector of deshrunk estimates.

For an object of class `"deshrink"`, inheriting from class `"list"`, with two components. The type of deshrinking performed is stroed within attribute `"type"`. The componets of the returned object are:

`coefficients`	The deshrinking coefficients used.
`env`	The deshrunk WA estimates.

Warning

`deshrinkPred`, does not currently check that the correct coefficients have been supplied in the correct order.

Author(s)

Gavin L. Simpson & Jari Oksanen

References

Birks, H.J.B. (1995) Quantitative environmental reconstructions. In *Statistical modelling of Quaternary science data* (eds.~D.Maddy & J.S. Brew). Quaternary Research Association technical guide 5. Quaternary Research Association, Cambridge.

See Also

`wa`

`dissimilarities` *Extract dissimilarity coefficients from models*

Description

Extracts a vector of dissimilarity coefficients from an object for further analysis.

Usage

```
dissimilarities(object, ...)
dissim(object, ...)

## S3 method for class 'analog'
dissimilarities(object, which = c("train", "analogs"),
                 ...)

## S3 method for class 'mat'
dissimilarities(object, ...)
```

Arguments

`object`	an R object from which the dissimilarity values are to be extracted. Currently only for objects of class `"analog"`.
`which`	character; which set of dissimilarities should be extracted. One of `"train"` or `"analogs"`.
`...`	arguments passed to other methods.

Details

The function can be called using the much shorter name `"dissim"`.

Value

A vector of dissimilarities.

Author(s)

Gavin L. Simpson

See Also

`analog`, `plot.dissimilarities`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## analog matching between SWAPImbrie & Kipp and V12.122 core
ik.analog <- analog(ImbrieKipp, V12.122, method = "chord")
ik.analog
summary(ik.analog)

## compare training set dissimilarities with normals
## and derive cut-offs
ik.dij <- dissim(ik.analog)
plot(ik.dij)
```

`distance` *Flexibly calculate dissimilarity or distance measures*

Description

Flexibly calculates distance or dissimilarity measures between a training set `x` and a fossil or test set `y`. If `y` is not supplied then the pairwise dissimilarities between samples in the training set, `x`, are calculated.

Usage

```
distance(x, ...)

## Default S3 method:
distance(x, y, method = "euclidean", weights = NULL,
         R = NULL, dist = FALSE, double.zero = FALSE, ...)

## S3 method for class 'join'
distance(x, ...)

oldDistance(x, ...)
## Default S3 method:
oldDistance(x, y, method = c("euclidean", "SQeuclidean",
            "chord", "SQchord", "bray", "chi.square",
            "SQchi.square", "information", "chi.distance",
            "manhattan", "kendall", "gower", "alt.gower",
            "mixed"),
            fast = TRUE,
            weights = NULL, R = NULL, ...)
## S3 method for class 'join'
oldDistance(x, ...)
```

Arguments

`x`	data frame or matrix containing the training set samples, or and object of class `join`.
`y`	data frame or matrix containing the fossil or test set samples.
`method`	character; which choice of dissimilarity coefficient to use. One of the listed options. See Details below.
`weights`	numeric; vector of weights for each descriptor.
`R`	numeric; vector of ranges for each descriptor.
`dist`	logical; should the dissimilarity matrix be returned as an object of class `"dist"`? Ignored if `y` is supplied.
`double.zero`	logical; if `FALSE`, the default, double zeroes are not counted in the distance calculation. If `TRUE`, absences of a variable in both samples counts as a similarity between the two samples. Currently this only affects methods `"mixed"` and `"metric.mixed"` forms of Gower's general coefficient.
`fast`	logical; should fast versions of the dissimilarities be calculated? See details below.
`...`	arguments passed to other methods

Details

A range of dissimilarity coefficients can be used to calculate dissimilarity between samples. The following are currently available:

`euclidean`	$d_{jk} = \sqrt{\sum_i (x_{ij} - x_{ik})^2}$
`SQeuclidean`	$d_{jk} = \sum_i (x_{ij} - x_{ik})^2$
`chord`	$d_{jk} = \sqrt{\sum_i (\sqrt{x_{ij}} - \sqrt{x_{ik}})^2}$
`SQchord`	$d_{jk} = \sum_i (\sqrt{x_{ij}} - \sqrt{x_{ik}})^2$
`bray`	$d_{jk} = \frac{\sum_i \lvert x_{ij} - x_{ik} \rvert}{\sum_i (x_{ij} + x_{ik})}$
`chi.square`	$d_{jk} = \sqrt{\sum_i \frac{(x_{ij} - x_{ik})^2}{x_{ij} + x_{ik}}}$
`SQchi.square`	$d_{jk} = \sum_i \frac{(x_{ij} - x_{ik})^2}{x_{ij} + x_{ik}}$
`information`	$d_{jk} = \sum_i (p_{ij} log(\frac{2p_{ij}}{p_{ij} + p_{ik}}) + p_{ik} log(\frac{2p_{ik}}{p_{ij} + p_{ik}}))$
`chi.distance`	$d_{jk} = \sqrt{\sum_i (x_{ij} - x_{ik})^2 / (x_{i+} / x_{++})}$
`manhattan`	$d_{jk} = \sum_i (\lvert x_{ij} - x_{ik} \rvert)$
`kendall`	$d_{jk} = \sum_i MAX_i - minimum(x_{ij}, x_{ik})$

`gower`	$d_{jk} = \sum_i \frac{\lvert p_{ij} - p_{ik} \rvert}{R_i}$
`alt.gower`	$d_{jk} = \sqrt{2 \sum_i \frac{\lvert p_{ij} - p_{ik} \rvert}{R_i}}$ where R_i is the range of proportions for descriptor (variable) i
`mixed`	$d_{jk} = \frac{\sum_{i=1}^{p} w_i s_{jki}}{\sum_{i=1}^{p} w_i}$ where w_i is the weight for descriptor i and s_{jki} is the similarity between samples j and k for descriptor (variable) i.
`metric.mixed`	as for `mixed` but with ordinal variables converted to ranks and handled as quantitative variables in Gower's m

Argument `fast` determines whether fast C versions of some of the dissimilarity coefficients are used. The fast versions make use of `dist` for methods `"euclidean"`, `"SQeuclidean"`, `"chord"`, `"SQchord"`, and `vegdist` for `method == "bray"`. These fast versions are used only when `x` is supplied, not when `y` is also supplied. Future versions of `distance` will include fast C versions of all the dissimilary coefficients and for cases where `y` is supplied.

Value

A matrix of dissimilarities where columns are the samples in `y` and the rows the samples in `x`. If `y` is not provided then a square, symmetric matrix of pairwise sample dissimilarities for the training set `x` is returned, unless argument `dist` is `TRUE`, in which case an object of class `"dist"` is returned. See `dist`.

The dissimilarity coefficient used (`method`) is returned as attribute `"method"`. Attribute `"type"` indicates whether the object was computed on a single data matrix (`"symmetric"`) or across two matrices (i.e. the dissimilarties between the rows of two matrices; `"asymmetric"`.

Warning

For `method = "mixed"` it is essential that a factor in `x` and `y` have the same levels in the two data frames. Previous versions of analogue would work even if this was not the case, which will have generated incorrect dissimilarities for `method = "mixed"` for cases where factors for a given species had different levels in `x` to `y`.

`distance` now checks for matching levels for each species (column) recorded as a factor. If the factor for any individual species has different levels in `x` and `y`, an error will be issued.

Note

The dissimilarities are calculated in native R code. As such, other implementations (see See Also below) will be quicker. This is done for one main reason - it is hoped to allow a user defined function to be supplied as argument `"method"` to allow for user-extension of the available coefficients.

The other advantage of `distance` over other implementations, is the simplicity of calculating only the required pairwise sample dissimilarities between each fossil sample (`y`) and each training set sample (`x`). To do this in other implementations, you would need to merge the two sets of samples, calculate the full dissimilarity matrix and then subset it to achieve similar results.

Author(s)

Gavin L. Simpson and Jari Oksanen (improvements leading to method `"metric.mixed"` and proper handling of ordinal data via Podani's (1999) modification of Gower's general coefficient in method `"mixed"`).

References

Faith, D.P., Minchin, P.R. and Belbin, L. (1987) Compositional dissimilarity as a robust measure of ecological distance. *Vegetatio* **69**, 57–68.

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Kendall, D.G. (1970) A mathematical approach to seriation. *Philosophical Transactions of the Royal Society of London - Series B* **269**, 125–135.

Legendre, P. and Legendre, L. (1998) *Numerical Ecology*, 2nd English Edition. Elsevier Science BV, The Netherlands.

Overpeck, J.T., Webb III, T. and Prentice I.C. (1985) Quantitative interpretation of fossil pollen spectra: dissimilarity coefficients and the method of modern analogues. *Quaternary Research* **23**, 87–108.

Podani, J. (1999) Extending Gower's General Coefficient of Similarity to Ordinal Characters. *Taxon* **48**, 331–340).

Prentice, I.C. (1980) Multidimensional scaling as a research tool in Quaternary palynology: a review of theory and methods. *Review of Palaeobiology and Palynology* **31**, 71–104.

See Also

`vegdist` in package **vegan**, `daisy` in package **cluster**, and `dist` provide comparable functionality for the case of missing `y`.

Examples

```
## simple example using dummy data
train <- data.frame(matrix(abs(runif(200)), ncol = 10))
rownames(train) <- LETTERS[1:20]
colnames(train) <- as.character(1:10)
fossil <- data.frame(matrix(abs(runif(100)), ncol = 10))
colnames(fossil) <- as.character(1:10)
rownames(fossil) <- letters[1:10]

## calculate distances/dissimilarities between train and fossil
## samples
test <- distance(train, fossil)

## using a different coefficient, chi-square distance
test <- distance(train, fossil, method = "chi.distance")

## calculate pairwise distances/dissimilarities for training
## set samples
test2 <- distance(train)

## Using distance on an object of class join
dists <- distance(join(train, fossil))
str(dists)

## calculate Gower's general coefficient for mixed data
## first, make a couple of variables factors
## fossil[,4] <- factor(sample(rep(1:4, length = 10), 10))
## train[,4] <- factor(sample(rep(1:4, length = 20), 20))
## ## now fit the mixed coefficient
## test3 <- distance(train, fossil, "mixed")

## ## Example from page 260 of Legendre & Legendre (1998)
x1 <- t(c(2,2,NA,2,2,4,2,6))
x2 <- t(c(1,3,3,1,2,2,2,5))
Rj <- c(1,4,2,4,1,3,2,5) # supplied ranges

## 1 - distance(x1, x2, method = "mixed", R = Rj)

## note this gives ~0.66 as Legendre & Legendre describe the
## coefficient as a similarity coefficient. Hence here we do
## 1 - Dij here to get the same answer.

## Tortula example from Podani (1999)
data(tortula)
Dij <- distance(tortula[, -1], method = "mixed") # col 1 includes Taxon ID

## Only one ordered factor
data(mite.env, package = "vegan")
Dij <- distance(mite.env, method = "mixed")

## Some variables are constant
data(BCI.env, package = "vegan")
Dij <- distance(BCI.env, method = "mixed")
```

`evenSample` *Number of samples per gradient segments*

Description

The number of samples in sections along the gradient is a useful diagnostic as to the quality of reconstructions at gradient values within those sections.

Usage

```
evenSample(grad, n = 10)
```

Arguments

`grad`	numeric; vector of gradient values
`n`	number of segments to partition the gradient into

Details

The sampling design of a training set, i.e. the number of samples taken at points along the gradient, can influence the uncertainty in the transfer function predictions at those values of the gradient. Poorly sampled sections of the gradient may have far larger RMSEP than the overall model RMSEP.

Value

Numeric vector of length `n` containing the numbers of samples per gradient segment.

Author(s)

Gavin L. Simpson

Examples

```
data(SumSST)
ev <- evenSample(SumSST) ## not an even sample...
plot(ev)
```

`fitted.logitreg` *Fitted values for the training set from logistic regression models*

Description

Extracts fitted values for training set samples from logistic regression models fitted to each group of samples that describe the probability two samples are analogues (from the same group) as a function of dissimilarity between the paired samples.

Usage

```
## S3 method for class 'logitreg'
fitted(object, combined = FALSE, ...)
```

Arguments

`object`	an object of class `"logitreg"` resulting from a call to `logitreg`.
`combined`	logical; should the fitted values for the overall combined analysis be returned.
`...`	arguments passed to other methods.

Value

If `combined == FALSE` (the default) then a matrix of fitted probabilities, where the rows are the training set samples and the columns the groupings, is returned. If `combined == TRUE`, then a list with components `"group"` and `"combined"`. `"group"` is a matrix of fitted probabilities as above. `"combined"` is a vector of fitted values for the entire set of pairwise comparisons considered.

Author(s)

Gavin L. Simpson

See Also

See `logitreg` for example usage.

fuse *Fused dissimilarities*

Description

Combines dissimilarities from two or more dissimilarity objects into a single dissimilarity object so that both original dissimilarities contribute equally. Weighted combinations of the original objects can also be created.

Usage

```
fuse(..., weights = NULL)

## S3 method for class 'matrix'
fuse(..., weights = NULL)

## S3 method for class 'dist'
fuse(..., weights = NULL)
```

Arguments

`...`	objects to fuse. Methods currently exist for objects of class `"matrix"` and `"dist"` objects. Method dispatch is performed on the first object specified in `...`. A minimum of two objects must be supplied.
`weights`	numeric; vector of weights that sum to 1, one weight for each object supplied in `...`.

Details

Fuses, or combines, dissimilarity objects in a very flexible way to create a single dissimilarity object that incorporates the separate dissimilarities. In analogue matching, we may wish to combine information from two or more proxies, such as diatoms and cladocera, or from biological and chemical or physical data in the case of matching modern samples.

The function can also be used to fuse dissimilarity objects created from a single data set but using different dissimilarity coefficients. In this way one could create a new dissimilarity object combining dissimilarity based on abundance data and presence absence data into a single measure.

`fuse` uses the method of Melssen et al. (2006) to combine dissimilarities. The dissimilarities in each dissimilarity object are scaled so that the maximum dissimilarity in each object is 1. The scaled dissimilarity objects are then weighted according to the supplied weights. If no weights are supplied (the default) the dissimilarity objects are weighted equally; `weights = rep(1/N, N)`, where `N` is the number of dissimilarity objects fused.

$$D_{fused}(j,k) = \sum_{i=1}^{N} w_i D_{ijk}$$

where $D_{fused}(j,k)$ is the fused dissimilarity between samples j and k, w_i is the weight assigned to the ith dissimilarity object and D_{ijk} is the dissimilarity between j and k for the ith dissimilarity object.

Value

`fuse` returns an object of class `"dist"` with the attribute `"method"` set to `"fuse"`.

This is the case even if the supplied objects are full dissimilarity matrices. If you want a full dissimilarity object, use `as.matrix.dist` on the returned object.

The returned object contains an extra attribute `"weights"`, which records the weights used in the fusing.

Author(s)

Gavin L. Simpson

References

Melssen W., Wehrens R. and Buydens L. (2006) Supervised Kohonen networks for classification problems. *Chemometrics and intelligent laboratory systems* **83**, 99–113.

See Also

`dist`, `vegdist`, `distance`.

Examples

```
train1 <- data.frame(matrix(abs(runif(100)), ncol = 10))
train2 <- data.frame(matrix(sample(c(0,1), 100, replace = TRUE),
                     ncol = 10))
rownames(train1) <- rownames(train2) <- LETTERS[1:10]
colnames(train1) <- colnames(train2) <- as.character(1:10)

d1 <- vegdist(train1, method = "bray")
d2 <- vegdist(train2, method = "jaccard")

dd <- fuse(d1, d2, weights = c(0.6, 0.4))
dd
str(dd)
```

getK *Extract and set the number of analogues*

Description

An extractor function to access the number of analogues used in particular models. The stored value of k can be updated using `setK`.

Usage

```
getK(object, ...)

## S3 method for class 'mat'
getK(object, weighted = FALSE, ...)

## S3 method for class 'bootstrap.mat'
getK(object, which = c("bootstrap", "model"),
     prediction = FALSE, ...)

## S3 method for class 'predict.mat'
getK(object, which = c("model", "bootstrap"),
     ...)

setK(object, weighted = FALSE) <- value

## S3 replacement method for class 'mat'
setK(object, weighted = FALSE) <- value
```

Arguments

`object`	an R object; currently only for objects of class `mat` and class `bootstrap.mat`.
`weighted`	logical; extract/set number of analogues for a weighted or un-weighted model?
`which`	character; which k should be extracted, the one from the model or the one from the bootstrap results?
`prediction`	logical; should the extracted k be the one that is minimum for the test set (`newdata`) or the model (`object`).
`...`	further arguments to other methods.
`value`	integer; replacement value for k.

Details

`getK` is a generic accessor function, and `setK<-` is a generic replacement function.

Objects of class `bootstrap.mat` contain several different `k`'s. If no predictions are performed, there will be two `k`'s, one for the model and one from bootstrapping the model. Where predictions are performed **with** `newenv` supplied, in addition to the `k`'s above, there will be two `k`' for the predictions, one for the model-based and one for the bootstrap-based predictions. To select `k` for the predictions, use `prediction = TRUE`. Argument `which` determines whether the model-based or the bootstrap-based `k` is returned.

Value

For `getK`, an integer value that is the number of analogues stored for use. The returned object has attributes "auto" and "weighted". "auto" refers to whether the extracted value of k was set automatically (`TRUE`) or by the user (`FALSE`). "weighted" states if the returned value is for a `weighted` analysis or an un-`weighted` analysis (`FALSE`).

For `setK<-`, the updated object.

Author(s)

Gavin L. Simpson

See Also

`mat`

Examples

```
## Imbrie and Kipp Sea Surface Temperature
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training set and core samples
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
ImbrieKippCore <- dat[[2]] / 100

## fit a MAT model
ik.mat <- mat(ImbrieKipp, SumSST, method = "chord")

## How many analogues gives lowest RMSE?
getK(ik.mat)
## note that this value was chosen automatically

## Now set k to be 10
setK(ik.mat) <- 10

## check
getK(ik.mat)
```

`gradientDist` *Positions of samples along a unit-length ordination gradient.*

Description

Extracts information as to the locations of samples along an ordination gradient. `gradientDist()` standardises the entire gradient to the interval 0, . . . , 1, to allow comparison between methods or data sets.

Usage

```
gradientDist(object, ...)

## Default S3 method:
gradientDist(object, na.rm = TRUE, ...)

## S3 method for class 'cca'
gradientDist(object, na.rm = TRUE, axis = 1L,
             scaling = 0, ...)

## S3 method for class 'prcurve'
gradientDist(object, na.rm = TRUE, ...)
```

Arguments

`object`	an R object of an appropriate type. For the default method, any R object that can be coerced to a vector.
`na.rm`	logical; should missing values be removed?
`axis`	numeric, length 1; the ordination axis to take as the gradient.
`scaling`	Scaling to apply to the site scores. Default is to do no scaling. See `scores.cca` for details.
`...`	additional arguments passed to other methods. In the `"cca"` method, these are also passed to `scores.cca`.

Value

A numeric vector of positions along the gradient, scaled to the range 0, . . . , 1.

Author(s)

Gavin L. Simpson

See Also

See `cca` and `prcurve` for functions that produce objects that `gradientDist()` can work with.

Examples

```
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit PCA
aber.pca <- rda(abernethy2)

## Distance along the first PCA axis
gradientDist(aber.pca)
```

`hist.residLen` *Histogram plot for residual lengths*

Description

Base graphics histogram plot method for `residLen` objects.

Usage

```
## S3 method for class 'residLen'
hist(x, breaks = "Sturges", freq = TRUE,
     probs = c(0.9, 0.95, 0.99), ncol = 1, lcol = "red",
     llty = "dashed", xlab = NULL, ylab = NULL,
     main = "Residual distances", rug = TRUE, ...)
```

Arguments

`x`	Object of class `"residLen"`, the result of a call to `residLen`.
`breaks`	How breakpoints for the histogram are determined. See `hist` for more details.
`freq`	logical; if `TRUE`, the histogram graphic is a representation of frequencies, the `counts` component of the result; if `FALSE`, probability densities, component `density`, are plotted (so that the histogram has a total area of one). Defaults to `TRUE`
`probs`	numeric; vector of probability quantiles to compute from the sets of residual distances.
`ncol`	numeric; number of columns for the plot layout. Choices are `1` or `2`. Determines whether the histograms are plotted above or beside each other.
`lcol, llty`	colour and line-type for the quantiles.
`xlab, ylab`	Axis labels. If not supplied, suitable defaults are generated, depending on whether RDA or CCA was used as the underlying ordination model.
`main`	character; title for the plot.
`rug`	logical; should rug plots of the actual distances be drawn?
`...`	additional arguments passed to `hist`.

Value

A plot on the current device.

Returns a list with two components (`train` and `passive`), each of which is an object returned by `hist`.

Author(s)

Gavin L. Simpson

See Also

`residLen`, `plot.residLen`, `histogram.residLen`, `densityplot.residLen`.

Examples

```
## load the Imbrie and Kipp example data
data(ImbrieKipp, SumSST, V12.122)

## squared residual lengths for Core V12.122
rlens <- residLen(ImbrieKipp, SumSST, V12.122)
rlens

## plot a histogram of the residual distances
hist(rlens)
```

`histogram.residLen` *Lattice histogram plot for residual lengths*

Description

Lattice `histogram` method for `residLen` objects.

Usage

```
## S3 method for class 'residLen'
histogram(x, ..., xlab = NULL, ylab = NULL,
          type = c("percent", "count", "density"))
```

Arguments

`x`	Object of class `"residLen"`, the result of a call to `residLen`.
`xlab, ylab`	Axis labels. If not supplied, suitable defaults are generated, depending on whether RDA or CCA was used as the underlying ordination model.
`type`	Character string indicating type of histogram to be drawn. `"percent"` and `"count"` give relative frequency and frequency histograms, and can be misleading when breakpoints are not equally spaced. `"density"` produces a density scale histogram. See `histogram` for further details.
`...`	Additional arguments passed to `histogram`.

Value

Returns an object of class `"trellis"`. See `histogram` for details.

Author(s)

Gavin L. Simpson

See Also

residLen, plot.residLen, hist.residLen, densityplot.residLen.

Examples

```
## load the Imbrie and Kipp example data
data(ImbrieKipp, SumSST, V12.122)

## squared residual lengths for Core V12.122
rlens <- residLen(ImbrieKipp, SumSST, V12.122)
rlens

## plot a histogram of the residual distances
histogram(rlens)
```

ImbrieKipp *Imbrie and Kipp foraminifera training set*

Description

The classic Imbrie and Kipp (1971) training set of counts on 27 species of foraminifera from 61 ocean sediment core surface samples and associated measures of summer and winter sea-surface temperatures and salinity at each location.

110 sediment cores samples from core V12-122 are also supplied in `V12.122`.

Usage

```
data(SumSST)
data(WinSST)
data(Salinity)
data(V12.122)
```

Format

`ImbrieKipp` is a data frame with 61 observations on the following 27 species:

`O.univ` a numeric vector

`G.cglob` a numeric vector

`G.ruber` a numeric vector

`G.tenel` a numeric vector

`G.saccu` a numeric vector

`G.rubes` a numeric vector

`G.pacL` a numeric vector

`G.pacR` a numeric vector

`G.bullo` a numeric vector

`G.falco` a numeric vector

`G.calid` a numeric vector

`G.aequi` a numeric vector

`G.gluti` a numeric vector

`G.duter` a numeric vector

`G.infla` a numeric vector

`G.trnL` a numeric vector

`G.trnR` a numeric vector

`G.crasf` a numeric vector

`G.scitu` a numeric vector

`G.mentu` a numeric vector

`P.obliq` a numeric vector

`C.nitid` a numeric vector

`S.dehis` a numeric vector

`G.digit` a numeric vector

`Other` a numeric vector

`G.quin` a numeric vector

`G.hirsu` a numeric vector

Summer and Winter sea-surface temperatures, and salinity values for the 61 sites in the Imbrie and Kipp training set (`ImbrieKipp`):

`SumSST` a numeric vector of summer sea-surface water temperatures. Values are in degrees C.

`WinSST` a numeric vector of winter sea-surface water temperatures. Values are in degrees C.

`Salinity` a numeric vector of sea water salinity values.

`V12.122` is a data frame with 110 observations from core V12-122 on the following 28 species:

`O.univ` a numeric vector

`G.cglob` a numeric vector

`G.ruber` a numeric vector

`G.tenel` a numeric vector

`G.saccu` a numeric vector

`G.rubes` a numeric vector

`G.pacL` a numeric vector

`G.pacR` a numeric vector

`G.bullo` a numeric vector

`G.falco` a numeric vector

`G.calid` a numeric vector

`G.aequi` a numeric vector

`G.gluti` a numeric vector

`G.duter` a numeric vector

`G.infla` a numeric vector

`G.trnL` a numeric vector

`G.trnR` a numeric vector

`G.crasf` a numeric vector

`G.scitu` a numeric vector

`G.mentu` a numeric vector

`P.obliq` a numeric vector

`C.nitid` a numeric vector

`S.dehis` a numeric vector

`G.digit` a numeric vector

`G.hexag` a numeric vector

`G.cglom` a numeric vector

`cfH.pel` a numeric vector

`Other` a numeric vector

Source

Imbrie and Kipp (1971) TODO: Get the full reference.

These data were provided in electronic format by Prof. H. John B. Birks.

Examples

```
data(ImbrieKipp)
head(ImbrieKipp)

data(SumSST)
data(WinSST)
data(Salinity)

plot(cbind(SumSST, WinSST, Salinity))

data(V12.122)
head(V12.122)
```

join	*Merge species data sets on common columns (species)*

Description

Merges any number of species matrices on their common columns to create a new data set with number of columns equal to the number of unqiue columns across all data frames. Needed for analysis of fossil data sets with respect to training set samples.

Usage

```
join(..., verbose = FALSE, na.replace = TRUE, split = TRUE, value = 0,
     type = c("outer", "left", "inner"))

## S3 method for class 'join'
head(x, ...)

## S3 method for class 'join'
tail(x, ...)
```

Arguments

`...`	for `join`, data frames containing the data sets to be merged. For the `head` and `tail` methods, additional arguments to `head` and `tail`, in particular `"n"` to control the number of rows of each `join`ed data set to display.
`verbose`	logical; if `TRUE`, the function prints out the dimensions of the data frames in `"\dots"`, as well as those of the returned, merged data frame.
`na.replace`	logical; samples where a column in one data frame that have no matching column in the other will contain missing values (`NA`). If `na.replace` is `TRUE`, these missing values are replaced with zeros. This is standard practice in ecology and palaeoecology. If you want to replace with another value, then set `na.replace` to `FALSE` and do the replacement later.
`split`	logical; should the merged data sets samples be split back into individual data frames, but now with common columns (i.e. species)?
`value`	numeric; value to replace `NA` with if `na.replace` is `TRUE`.
`type`	logical; type of join to perform. `"outer"` returns the *union* of the variables in data frames to be merged, such that the resulting objects have columns for all variables found across all the data frames to be merged. `"left"` returns the left outer (or the left) join, such that the merged data frames contain the set of variables found in the first supplied data frame. `"inner"` returns the inner join, such that the merged data frame contain the intersection of the variables in the supplied data frames. See Details.
`x`	an object of class `"join"`, usually the result of a call to `join`.

Details

When merging multiple data frames the set of variables in the merged data can be determined via a number of routes. `join` provides for two (currently) join types; the *outer* join and the *left outer* (or simply the *left*) join. Which type of join is performed is determined by the argument `type`.

The *outer* join returns the union of the set of variables found in the data frames to be merged. This means that the resulting data frame(s) contain columns for all the variable observed across all the data frames supplied for merging.

With the *left outer* join the resulting data frame(s) contain only the set of variables found in the first data frame provided.

The *inner* join returns the intersection of the set of variables found in the supplied data frames. The resulting data frame(s) contains the variables common to all supplied data frames.

Value

If `split = TRUE`, an object of class `"join"`, a list of data frames, with as many components as the number of data frames originally merged.

Otherwise, an object of class `c("join", "data.frame")`, a data frame containing the merged data sets.

`head.join` and `tail.join` return a list, each component of which is the result of a call to `head` or `tail` on each data set compont of the joined object.

Author(s)

Gavin L. Simpson

See Also

`merge`

Examples

```
## load the example data
data(swapdiat, swappH, rlgh)

## merge training and test set on columns
dat <- join(swapdiat, rlgh, verbose = TRUE)

## extract the merged data sets and convert to proportions
swapdiat <- dat[[1]] / 100
rlgh <- dat[[2]] / 100

## merge training and test set using left join
head(join(swapdiat, rlgh, verbose = TRUE, type = "left"))

## load the example data
data(ImbrieKipp, SumSST, V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)
```

```
## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## show just the first few lines of each data set
head(dat, n = 4)

## show just the last few lines of each data set
tail(dat, n = 4)

## merge training and test set using inner join
head(join(ImbrieKipp, V12.122, verbose = TRUE, type = "inner"))

## merge training and test set using outer join and replace
## NA with -99.9
head(join(ImbrieKipp, V12.122, verbose = TRUE, value = -99.9))
```

logitreg *Logistic regression models for assessing analogues/non-analogues*

Description

Fits logistic regression models to each level of `group` to model the probability of two samples being analogues conditional upon the dissimilarity between the two samples.

Usage

```
logitreg(object, groups, k = 1, ...)

## Default S3 method:
logitreg(object, groups, k = 1,
         biasReduced = FALSE, ...)

## S3 method for class 'analog'
logitreg(object, groups, k = 1, ...)

## S3 method for class 'logitreg'
summary(object, p = 0.9, ...)
```

Arguments

`object`	for `logitreg`; a full dissimilarity matrix. For `summary.logitreg` an object of class `"logitreg"`, the result of a call to `logitreg`.
`groups`	factor (or object that can be coerced to one) containing the group membership for each sample in `object`.
`k`	numeric; the `k` closest analogues to use in the model fitting.

`biasReduced`	logical; should Firth's method for bias reduced logistic regression be used to fit the models? If `TRUE`, model fits are performed via `brglm`. The default, `FALSE`, indicates that models will be fitted via the standard `glm` function.
`p`	probability at which to predict the dose needed.
`...`	arguments passed to other methods. These arguments are passed on to `glm` or `brglm`. See their respective helps pages for details. Note that `logitreg` sets internally the `formula`, `data`, and `family` arguments and hence can not be specified by the user.

Details

Fits logistic regression models to each level of `group` to model the probability of two samples being analogues (i.e. in the same group) conditional upon the dissimilarity between the two samples.

This function can be seen as a way of directly modelling the probability that two sites are analogues, conditional upon dissimilarity, that can also be done less directly using `roc` and `bayesF`.

Often, the number of true analogues in the training set is small, both in absolute terms and as a proportion of comparisons. Logistic regression is known to suffer from a small-sample bias. Firth's method of bias reduction is a general solution to this problem and is implemented in `logitreg` through the **brglm** package of Ioannis Kosmidis.

Value

`logitreg` returns an object of class `"logitreg"`; a list whose components are objects returned by `glm`. See `glm` for further details on the returned objects.

The components of this list take their names from `group`.

For `summary.logitreg` an object of class `"summary.logitreg"`, a data frame with summary statistics of the model fits. The components of this data frame are:

`In`, `Out`	The number of analogue and non-analogue dissimilarities analysed in each group,
`Est.(Dij)`, `Std.Err`	Coefficient and its standard error for dissimilarity from the logit model,
`Z-value`, `p-value`	Wald statistic and associated p-value for each logit model.
`Dij(p=?)`, `Std.Err(Dij)`	The dissimilarity at which the posterior probability of two samples being analogues is equal to p, and its standard error. These are computed using `dose.p`.

Note

The function may generate warnings from function `glm.fit`. These should be investigated and not simply ignored.

If the message is concerns fitted probabilities being numerically 0 or 1, then check the fitted values of each of the models. These may well be numerically 0 or 1. Heed the warning in `glm` and read the reference cited therein which **may** indicate problems with the fitted models, such as (quasi-)complete separation.

Author(s)

Gavin L. Simpson

References

Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika* **80**, 27-38.

See Also

roc, bayesF, glm, and brglm.

Examples

```
## load the example data
data(swapdiat, swappH, rlgh)

## merge training and test set on columns
dat <- join(swapdiat, rlgh, verbose = TRUE)

## extract the merged data sets and convert to proportions
swapdiat <- dat[[1]] / 100
rlgh <- dat[[2]] / 100

## fit an analogue matching (AM) model using the squared chord distance
## measure - need to keep the training set dissimilarities
swap.ana <- analog(swapdiat, rlgh, method = "SQchord",
                   keep.train = TRUE)

## fit the ROC curve to the SWAP diatom data using the AM results
## Generate a grouping for the SWAP lakes
METHOD <- if (getRversion() < "3.1.0") {"ward"} else {"ward.D"}
clust <- hclust(as.dist(swap.ana$train), method = METHOD)
grps <- cutree(clust, 6)

## fit the logit models to the analog object
swap.lrm <- logitreg(swap.ana, grps)
swap.lrm

## summary statistics
summary(swap.lrm)

## plot the fitted logit curves
plot(swap.lrm, conf.type = "polygon")

## extract fitted posterior probabilities for training samples
## for the individual groups
fit <- fitted(swap.lrm)
head(fit)

## compute posterior probabilities of analogue-ness for the rlgh
## samples. Here we take the dissimilarities between fossil and
## training samples from the `swap.ana` object rather than re-
## compute them
pred <- predict(swap.lrm, newdata = swap.ana$analogs)
head(pred)

## Bias reduction
## fit the logit models to the analog object
swap.brlrm <- logitreg(swap.ana, grps, biasReduced = TRUE)
summary(swap.brlrm)
```

`mat` *Modern Analogue Technique transfer function models*

Description

Modern Analogue Technique (MAT) transfer function models for palaeoecology. The fitted values are the, possibly weighted, averages of the environment for the *k*-closest modern analogues. MAT is a *k*-NN method.

Usage

```
mat(x, ...)

## Default S3 method:
mat(x, y,
    method = c("euclidean", "SQeuclidean", "chord", "SQchord",
               "bray", "chi.square", "SQchi.square",
               "information", "chi.distance", "manhattan",
               "kendall", "gower", "alt.gower", "mixed"),
    kmax, ...)

## S3 method for class 'formula'
mat(formula, data, subset, na.action,
    method = c("euclidean", "SQeuclidean", "chord", "SQchord",
               "bray", "chi.square", "SQchi.square",
               "information", "chi.distance", "manhattan",
               "kendall", "gower", "alt.gower", "mixed"),
    model = FALSE, ...)

## S3 method for class 'mat'
fitted(object, k, weighted = FALSE, ...)

## S3 method for class 'mat'
residuals(object, k, weighted = FALSE, ...)
```

Arguments

`x`	a data frame containing the training set data, usually species data.

`y`	a vector containing the response variable, usually environmental data to be predicted from `x`.
`formula`	a symbolic description of the model to be fit. The details of model specification are given below.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `wa` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The "factory-fresh" default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`method`	a character string indicating the dissimilarity (distance) coefficient to be used to define modern analogues. See Details, below.
`model`	logical; If `TRUE` the model frame of the fit is returned.
`kmax`	numeric; limit the maximum number of analogues considered during fitting. By default, `kmax` is equal to $n - 1$, where n is the number of sites. For large data sets this is just wasteful as we wouldn't expect to be averaging over the entire training set. `kmax` can be used to restrict the upper limit on the number of analogues considered.
`object`	an object of class `mat`.
`k`	numeric; the *k*-closest analogue models' for which fitted values and residuals are returned. Overides the default stored in the object.
`weighted`	logical; should weighted averages be used instead of simple averages?
`...`	arguments can be passed to `distance` to provide additional optios required for some dissimilarities.

Details

The Modern Analogue Technique (MAT) is perhaps the simplest of the transfer function models used in palaeoecology. An estimate of the environment, x, for the response for a fossil sample, y, is the, possibly weighted, mean of that variable across the *k*-closest modern analogues selected from a modern training set of samples. If used, weights are the reciprocal of the dissimilarity between the fossil sample and each modern analogue.

A typical model has the form `response ~ terms` where `response` is the (numeric) response data frame and `terms` is a series of terms which specifies a linear predictor for `response`. A typical form for `terms` is `.`, which is shorthand for "all variables" in `data`. If `.` is used, `data` must also be provided. If specific species (variables) are required then `terms` should take the form `spp1 + spp2 + spp3`.

Pairwise sample dissimilarity is defined by dissimilarity or distance coefficients. A variety of coefficients are supported — see `distance` for details of the supported coefficients.

k is chosen by the user. The simplest choice for *k* is to evaluate the RMSE of the difference between the predicted and observed values of the environmental variable of interest for the training set samples for a sequence of models with increasing *k*. The number of analogues chosen is the value of *k* that has lowest RMSE. However, it should be noted that this value is biased as the data used to build the model are also used to test the predictive power.

An alternative approach is to employ an optimisation data set on which to evaluate the size of k that provides the lowest RMSEP. This may be impractical with smaller sample sizes.

A third option is to bootstrap re-sample the training set many times. At each bootstrap sample, predictions for samples in the bootstrap test set can be made for $k = 1, ..., n$, where n is the number of samples in the training set. k can be chosen from the model with the lowest RMSEP. See function `bootstrap.mat` for further details on choosing k.

The output from `summary.mat` can be used to choose k in the first case above. For predictions on an optimsation or test set see `predict.mat`. For bootstrap resampling of `mat` models, see `bootstrap.mat`.

The fitted values are for the training set and are taken as the, possibly weighted, mean of the environmental variable in question across the *k*-closest analogues. The fitted value for each sample does **not** include a contribution from itself — it is the closest analogue, having zero dissimilarity. This spurious distance is ignored and analogues are ordered in terms of the non-zero distances to other samples in the training set, with the *k*-closest contributing to the fitted value.

Typical usages for `residuals.mat` are:

```
    resid(object, k, weighted = FALSE, \dots)
```

Value

`mat` returns an object of class `mat` with the following components:

`standard`	list; the model statistics based on simple averages of *k*-closest analogues. See below.
`weighted`	list; the model statistics based on weighted of *k*-closest analogues. See below.
`Dij`	matrix of pairwise sample dissimilarities for the training set `x`.
`orig.x`	the original training set data.
`orig.y`	the original environmental data or response, `y`.
`call`	the matched function call.
`method`	the dissimilarity coefficient used.

If `model = TRUE` then additional components `"terms"` and `"model"` are returned containing the `terms` object and model frame used.

`fitted.mat` returns a list with the following components:

`estimated`	numeric; a vector of fitted values.
`k`	numeric; this is the *k*-closest analogue model with lowest apparent RMSE.
`weighted`	logical; are the fitted values the weighted averages of the environment for the *k*-closest analogues. If `FALSE`, the fitted values are the average of the environment for the *k*-closest analogues.

Note

The object returned by `mat` contains lists `"standard"` and `"weighted"` both with the following elements:

`est` a matrix of estimated values for the training set samples for models using k analogues, where $k = 1, ..., n$. n is the number of smaples in the training set. Rows contain the values for each model of size k, with colums containing the estimates for each training set sample.

`resid` matrix; as for `"est"`, but containing the model residuals.

`rmsep` vector; containing the leave-one-out root mean square error or prediction.

`avg.bias` vector; contains the average bias (mean of residuals) for models using k analogues, where $k = 1, ..., n$. n is the number of smaples in the training set.

`max.bias` vector; as for `"avg.bias"`, but containing the maximum bias statistics.

`r.squared` vector; as for `"avg.bias"`, but containing the R^2 statistics.

Author(s)

Gavin L. Simpson

References

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Overpeck, J.T., Webb III, T. and Prentice I.C. (1985) Quantitative interpretation of fossil pollen spectra: dissimilarity coefficients and the method of modern analogues. *Quaternary Research* **23**, 87–108.

Prell, W.L. (1985) The stability of low-latitude sea-surface temperatures: an evaluation of the CLIMAP reconstruction with emphasis on the positive SST anomalies, Report TR 025. U.S. Department of Energy, Washington, D.C.

Sawada, M., Viau, A.E., Vettoretti, G., Peltier, W.R. and Gajewski, K. (2004) Comparison of North-American pollen-based temperature and global lake-status with CCCma AGCM2 output at 6 ka. *Quaternary Science Reviews* **23**, 87–108.

See Also

`summary.mat`, `bootstrap.mat` for boostrap resampling of MAT models, `predict.mat` for making predictions from MAT models, `fitted.mat` and `resid.mat` for extraction of fitted values and residuals from MAT models respectively. `plot.mat` provides a `plot.lm`-like plotting tool for MAT models.

Examples

```
## Imbrie and Kipp Sea Surface Temperature
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training set and core samples
```

```
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
ImbrieKippCore <- dat[[2]] / 100

## fit the MAT model using the squared chord distance measure
ik.mat <- mat(ImbrieKipp, SumSST, method = "chord")
ik.mat

## model summary
summary(ik.mat)

## fitted values
fitted(ik.mat)

## model residuals
resid(ik.mat)

## draw summary plots of the model
par(mfrow = c(2,2))
plot(ik.mat)
par(mfrow = c(1,1))

## reconstruct for the V12.122 core data
coreV12.mat <- predict(ik.mat, V12.122, k = 3)
coreV12.mat
summary(coreV12.mat)

## draw the reconstruction
reconPlot(coreV12.mat, use.labels = TRUE, display.error = "bars",
          xlab = "Depth", ylab = "SumSST")

## fit the MAT model using the squared chord distance measure
## and restrict the number of analogues we fit models for to 1:20
ik.mat2 <- mat(ImbrieKipp, SumSST, method = "chord", kmax = 20)
ik.mat2
```

mcarlo *Monte Carlo simulation of dissimilarities*

Description

Permutations and Monte Carlo simulations to define critical values for dissimilarity coefficients for use in MAT reconstructions.

Usage

```
mcarlo(object, ...)
```

```
## Default S3 method:
mcarlo(object, nsamp = 10000,
       type = c("paired", "complete", "bootstrap", "permuted"),
       replace = FALSE,
       method = c("euclidean", "SQeuclidean", "chord", "SQchord",
                  "bray", "chi.square", "SQchi.square",
                  "information", "chi.distance", "manhattan",
                  "kendall", "gower", "alt.gower", "mixed"),
       is.dcmat = FALSE, diag = FALSE, ...)

## S3 method for class 'mat'
mcarlo(object, nsamp = 10000,
       type = c("paired", "complete", "bootstrap", "permuted"),
       replace = FALSE, diag = FALSE, ...)

## S3 method for class 'analog'
mcarlo(object, nsamp = 10000,
       type = c("paired", "complete", "bootstrap", "permuted"),
       replace = FALSE, diag = FALSE, ...)
```

Arguments

`object`	an R object. Currently only object's of class `"mat"`, `"analog"` or matrix-like object of species data allowed.
`nsamp`	numeric; number of permutations or simulations to draw.
`type`	character; the type of permutation or simulation to perform. See Details, below.
`replace`	logical; should sampling be done with replacement?
`method`	character; for raw species matrices, the dissimilarity coefficient to use. This is predefined when fitting a MAT model with `mat` or analogue matching via `analogue` and is ignored in the `"mcarlo"` methods for classes `"mat"` and `"analog"`.
`is.dcmat`	logical; is `"object"` a dissimilarity matrix. Not meant for general use; used internally by `"mat"` and `"analogue"` methods to instruct the `"default"` method that `"object"` is already a dissimilarity matrix, so there is no need to recalculate.
`diag`	logical; should the dissimilarities include the diagonal (zero) values of the dissimilarity matrix. See Details.
`...`	arguments passed to or from other methods.

Details

Only `"type"` `"paired"` and `"bootstrap"` are currently implemented.

`distance` produces square, symmetric dissimilarity matrices for training sets. The upper triangle of these matrices is a duplicate of the lower triangle, and as such is redundant. `mcarlo` works on the lower triangle of these dissimilarity matrices, representing all pairwise dissimilarity values for training set samples. The default is **not** to include the diagonal (zero) values of the dissimilarity matrix. If you feel that these diagonal (zero) values are part of the population of dissimilarities then use `"diag = TRUE"` to include them in the permutations.

Value

A vector of simulated dissimilarities of length "nsamp". The "method" used is stored in attribute "method".

Note

The performance of these permutation and simulation techniques still needs to be studied. This function is provided for pedagogic reasons. Although recommended by Sawada et al (2004), sampling with replacement ("replace = TRUE") and including diagonal (zero) values ("diag = TRUE") simulates too many zero distances. This is because the same training set sample can, on occasion be drawn twice leading to a zero distance. It is impossible to find in nature two samples that will be perfectly similar, and as such sampling **with** replacement **and** "diag = TRUE" seems undesirable at best.

Author(s)

Gavin L. Simpson

References

Sawada, M., Viau, A.E., Vettoretti, G., Peltier, W.R. and Gajewski, K. (2004) Comparison of North-American pollen-based temperature and global lake-status with CCCma AGCM2 output at 6 ka. *Quaternary Science Reviews* **23**, 87–108.

See Also

mat for fitting MAT models and analog for analogue matching. roc as an alternative method for determining critical values for dissimilarity measures when one has grouped data.

plot.mcarlo provides a plotting method to visualise the distribution of simulated dissimilarities.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## perform the modified method of Sawada (2004) - paired sampling,
## with replacement
ik.mcarlo <- mcarlo(ImbrieKipp, method = "chord", nsamp = 1000,
                    type = "paired", replace = FALSE)
ik.mcarlo
```

```
## plot the simulated distribution
layout(matrix(1:2, ncol = 1))
plot(ik.mcarlo)
layout(1)
```

`minDC` *Extract minimum dissimilarities*

Description

Minimum dissimilarity is a useful indicator of reliability of reconstructions performed via MAT and other methods, and for analogue matching. Minimum dissimilarity for a sample is the smallest dissimilarity between it and the training set samples.

Usage

```
minDC(x, ...)

## Default S3 method:
minDC(x, ...)

## S3 method for class 'predict.mat'
minDC(x, ...)

## S3 method for class 'analog'
minDC(x, probs = c(0.01, 0.02, 0.05, 0.1), ...)

## S3 method for class 'wa'
minDC(x, y,
      method = c("euclidean", "SQeuclidean", "chord", "SQchord",
                 "bray", "chi.square", "SQchi.square", "information",
                 "chi.distance", "manhattan", "kendall", "gower",
                 "alt.gower", "mixed"),
      percent = FALSE, probs = c(0.01, 0.025, 0.05, 0.1), ...)
```

Arguments

`x`	an object of class `"predict.mat"`, `"analog"` or a object with a component named `"minDC"`.
`probs`	numeric; vector of probabilities with values in [0,1].
`y`	an optional matrix-like object containing fossil samples for which the minimum dissimilarities to training samples are to be calculated.
`method`	character; which choice of dissimilarity coefficient to use. One of the listed options. See `distance`.
`percent`	logical; Are the data percentages? If `TRUE`, the data (`x` and `y`) will be divided by 100 to convert them to the proportions expected by `distance`.
`...`	other arguments to be passed to other methods. Currently ignored.

Value

minDC returns an object of class "minDC".

An object of class minDC is a list with some or all of the following components:

minDC	numeric; vector of minimum dissimilarities.
method	character; the dissimilarity coefficient used.
quantiles	numeric; named vector of probability quantiles for distribution of dissimilarities of modern training set.

Note

The "default" method of minDC will attempt to extract the relevant component of the object in question. This may be useful until a specific minDC method is written for a given class.

Author(s)

Gavin L. Simpson

See Also

predict.mat, and plot.minDC for a plotting method.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## fit the MAT model using the squared chord distance measure
ik.mat <- mat(ImbrieKipp, SumSST, method = "SQchord")
ik.mat

## reconstruct for the V12-122 core data
v12.mat <- predict(ik.mat, V12.122)

## extract the minimum DC values
v12.mdc <- minDC(v12.mat)
v12.mdc

## draw a plot of minimum DC by time
plot(v12.mdc, use.labels = TRUE, xlab = "Depth (cm.)")
```

n2 *Calculate Hill's N2 diversity measure*

Description

Hills N2 is a measure of species diversity, commonly referred to as "effective" diversity. If computed on the rows (samples) then the "effective" number of species in each row is returned, whereas, if computed on the columns (species) then the "effective" number of occurences of each species in the data set is returned.

Usage

```
n2(x, ...)

## Default S3 method:
n2(x, which = c("species", "sites"), ...)
```

Arguments

`x`	matrix or data frame of species data
`which`	character; compute N2 on the rows (`"sites"`) or the columns (`"species"`) of `x`
`...`	arguments passed to other methods

Value

A numeric vector of N2 values.

Author(s)

Gavin L. Simpson

Examples

```
data(swapdiat)
sppN2 <- n2(swapdiat, "species")
head(sppN2)
sampN2 <- n2(swapdiat, "sites")
head(sampN2)
```

optima	*Weighted averaging optima and tolerance ranges*

Description

Computes weighted average optima and tolerance ranges from species abundances and values of the environment.

Usage

```
optima(x, ...)

## Default S3 method:
optima(x, env, boot = FALSE, nboot = 1000,
       alpha = 0.05, ...)

## Default S3 method:
tolerance(x, env, useN2 = TRUE, ...)
```

Arguments

`x`	Species data matrix or data frame.
`env`	Numeric; variable for which optima or tolerances are required.
`boot, nboot`	logical (`boot`), numeric (`nboot`); should bootstrap resampling be employed to estimate the optima, and if so how many bootstrap samples to draw?
`alpha`	numeric; 1 - `alpha` gives the coverage for the percentile bootstrap confidence interval.
`useN2`	logical; should Hill's N2 values be used to produce un-biased tolerances?
`...`	Arguments passed to other methods.

Value

Both functions return a named vector containing the WA optima or tolerances for the environmental gradient specified by `env`.

Note

Objects of class `"optima"` or `"tolerance"` can be coerced to data frames using methods for `as.data.frame`.

Author(s)

Gavin L. Simpson

See Also

`wa`

Examples

```
## Load the Imbrie & Kipp data and
## summer sea-surface temperatures
data(ImbrieKipp)
data(SumSST)

## WA optima
(opt <- optima(ImbrieKipp, SumSST))

## WA tolerances
(tol <- tolerance(ImbrieKipp, SumSST, useN2 = TRUE))

## caterpillar plot
caterpillarPlot(opt, tol)

## convert to data frame
as.data.frame(opt)
as.data.frame(tol)

## bootstrap WA optima - 100 resamples too low for SD & pCI
bopt <- optima(ImbrieKipp, SumSST, boot = TRUE, nboot = 100)
head(bopt)
```

---

panel.Loess *Loess smooths to stratigraphic diagrams*

Description

A modified version of `panel.loess`, for drawing Loess smooths on stratigraphic diagrams.

Usage

```
panel.Loess(x, y,
            span = 1/3, degree = 1,
            family = c("symmetric","gaussian"),
            evaluation = 50,
            lwd = plot.line$lwd,
            lty = plot.line$lty,
            col,
            col.line = plot.line$col,
            type,
            ...)
```

Arguments

`x, y`	variables defining the contents of the panel.
`span, degree, family, evaluation`	arguments passed to `loess.smooth`, for which `panel.Loess` is a wrapper.

`lwd, lty, col, col.line`	graphical parameters.
`type`	for compatibility with `panel.loess`, but is ignored within the function.
`...`	graphical parameters can be supplied. Color can usually be specified by `col`, `col.line`, the latter overriding the first for lines.

Details

The standard panel function `panel.loess` treats the data as the x-axis acting as the time component. For stratigraphic plots where time flows along the y-axis, we want the smoother to be fitted with the x-axis data as the response and the time component (y-axis) as the predictor.

This modified version of `panel.loess` flips the two axes to produce the desired effect. Note also that it does not have argument `horizontal` as this is not required or supported by `Stratiplot`. In other respects, `panel.Loess` is equivalent to the lattice panel function `panel.loess`.

User should note that warnings can be generated by the fitting function if span is set too small for species with few observations. In such cases, the user is directed to the help page for `loess.smooth`, but increasing span slightly can often stop the warnings.

Author(s)

Gavin L. Simpson, slightly modified from the Lattice function `panel.loess` by Deepayan Sarkar.

See Also

`loess.smooth`, `panel.loess`.

`panel.Stratiplot` *Panel function for stratigraphic diagrams*

Description

A Lattice panel function for drawing individuals panels on stratigraphic diagrams using a range of plot types commonly used within palaeoecology.

Usage

```
panel.Stratiplot(x, y,
                 type = "l",
                 col,
                 pch = plot.symbol$pch,
                 cex = plot.symbol$cex,
                 col.line = plot.line$col,
                 col.symbol = plot.symbol$col,
                 col.refline = ref.line$col,
                 col.smooth = "red",
                 col.poly = plot.line$col,
                 lty = plot.line$lty,
                 lwd = plot.line$lwd,
                 lty.smooth = plot.line$lty,
                 lwd.smooth = 2,
                 lwd.h = 3,
                 fill = plot.symbol$fill,
                 zones = NULL,
                 col.zones = plot.line$col,
                 lty.zones = plot.line$lty,
                 lwd.zones = plot.line$lwd,
                 gridh = -1, gridv = -1,
                 ...)
```

Arguments

`x, y`	variables defining the contents of the panel.
`type`	character vector consisting of one or more of the following: `"l"`, `"p"`, `"o"`, `"b"`, `"h"`, `"g"`, `"smooth"`, and `"poly"`. It `type` has more than one element, the effects of each component are combined, though note that some elements will over-plot, and hence obscure, earlier elements. For types `"l"`, `"p"`, `"o"`, `"b"` and `"g"` the standard Lattice interpretation is observed. See `panel.xyplot` for further details. Note that `type "b"` is the same as `type "o"`. `"g"` adds a reference grid using `panel.grid` in the background. For `"h"`, histogram-like bars are plotted from the **y-axis**, not from the x-axis with `plot` and `panel.loess`. For `"smooth"` a loess fit is added to each panel using `panel.Loess`. For `"poly"`, a shaded polygon, or silhouette, is drawn for each panel.
`col, col.line, col.symbol, col.poly, col.refline, col.smooth, col.zones`	colour parameters. For all but `col.smooth`, default colours are obtained from `plot.symbol` and `plot.line` using `trellis.par.get`. `col.refline` controls the colour of the reference line drawn at value 0 on the x-axis, as well as the colour of the grid lines if drawn.
`pch, cex, lty, lwd, fill`	other graphical parameters, defaults for which are obtained from `plot.symbol` and `plot.line` using `trellis.par.get`.
`lty.smooth`	line type for the loess smoother. The default is obtained from `plot.line` using `trellis.par.get`.
`lwd.smooth, lwd.h`	The line width for the loess smoother and histogram-like bars respectively.
`zones`	numeric; vector of zone boundary positions on scale of the depth/time (y-)axis.
`lty.zones, lwd.zones`	line type and width for the zone markers. The defaults are obtained from `plot.line`.
`gridh, gridv`	numeric arguments corresponding to `h` and `v` of `panel.grid`, which control the number of grid lines drawn.
`...`	extra arguments passed on to the underlying panel functions; `panel.points`, `panel.lines`, `panel.segments`, `panel.polygon`, `panel.Loess` and `panel.refline`.

Details

Creates stratigraphic scatter plots of x and y, with various modifications possible via the type argument.

Zones can be drawn on the panels by supplying the numeric vector of zone boundaries as argument zones. The panel function will then draw horizontal lines across the panels at the desired y-axis locations. Note that the panel function does **not** attempt to identify the zone boundaries automatically; these must be determined via a chronological (constrained) cluster analysis function or similar.

Note that all the arguments controlling the display can be supplied directly to a high-level call of the function Stratiplot.

Note

Histogram-like bars (type = "h") are drawn with lineend = "butt" to improve their appearance. This can not be changed by the user and you can't include that grid parameter in any call to panel.Stratiplot that uses type = "h".

Author(s)

Gavin L. Simpson

See Also

Stratiplot, panel.Loess, panel.xyplot.

---

pcr *Prinicpal component regression transfer function models*

Description

Fits a palaeoecological transfer function model using principal component regression, using an optional transformation of the matrix of predictor variables when these are species abundance data.

Usage

```
## Default S3 method:
pcr(x, y, ncomp, tranFun, ...)

## S3 method for class 'formula'
pcr(formula, data, subset, na.action, ..., model = FALSE)

Hellinger(x, ...)

ChiSquare(x, apply = FALSE, parms)

## S3 method for class 'pcr'
performance(object, ...)
```

```
## S3 method for class 'pcr'
residuals(object, comps = NULL, ...)

## S3 method for class 'pcr'
fitted(object, comps = NULL, ...)

## S3 method for class 'pcr'
coef(object, comps = NULL, ...)

## S3 method for class 'pcr'
screeplot(x, restrict = NULL,
          display = c("RMSE","avgBias","maxBias","R2"),
          xlab = NULL, ylab = NULL, main = NULL, sub = NULL, ...)

## S3 method for class 'pcr'
eigenvals(x, ...)
```

Arguments

`x`	Matrix or data frame of predictor variables. Usually species composition or abundance data for transfer function models. For `screeplot` and `eigenvals`, an object of class `"pcr"`.
`y`	Numeric vector; the response variable to be modelled.
`ncomp`	numeric; number of principal components to build models for. If not supplied the largest possible number of components is determined.
`tranFun`	function; a function or name of a function that performs a transformation of the predictor variables `x`. The function must be self-contained as no arguments are passed to the function when it is applied. See Details for more information.
`formula`	a model formula.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables specified on the RHS of the model formula. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `pcr` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`model`	logical; If `TRUE` the model frame is returned?
`apply`	logical; should an existing tranformation, using pre-computed meta-parameters, be applied?
`parms`	list; a named list of parameters computed during model fitting that can be used to apply the transformation during prediction.
`object`	an object of class `"pcr"`.

`comps`	numeric; which components to return.
`restrict`	numeric; limit the number of components on the screeplot.
`display`	character; which model performance statistic should be drawn on the screeplot?
`xlab`, `ylab`, `main`, `sub`	character; labels for the plot.
`...`	Arguments passed to other methods.

Details

When applying cross-validation (CV) to transfer function models, any transformation of the predictors must be applied separately during each iteration of the CV procedure to the part of the data used in fitting the model. In the same way, any samples to be predicted from the model must use any meta-parameters derived from the training data only. For examle, centring is appled to the training data only and the variables means used to centre the training data are used to centre the test samples. The variable means should not be computed on a combination of the training and test samples.

When using PCR, we might wish to apply a transformation to the species data predictor variables such that the PCA of those data preserves a dissimilarity coefficient other than the Euclidean distance. This transformation is applied to allow PCA to better describe patterns in the species data (Legendre & Gallagher 2001).

How this is handled in `pcr` is to take a user-supplied function that takes a single argument, the matrix of predictor variables. The function should return a matrix of the same dimension as the input. If any meta-parameters are required for subsequent use in prediction, these should be returned as attribute `"parms"`, attached to the matrix.

Two example transformation functions are provided implementing the Hellinger and Chi Square transformations of Legendre & Gallagher (2001). Users can base their transformation functions on these. `ChiSquare()` illustrates how meta-parameters should be returned as the attribute `"parms"`.

Value

Returns an object of class `"pcr"`, a list with the following components:

`fitted.values`	matrix; the PCR estimates of the response. The columns contain fitted values using C components, where C is the Cth column of the matrix.
`coefficients`	matrix; regression coefficients for the PCR. Columns as per `fitted` above.
`residuals`	matrix; residuals, where the Cth column represents a PCR model using C components.
`scores`	
`loadings`	
`Yloadings`	
`xMeans`	numeric; means of the predictor variables in the training data.
`yMean`	numeric; mean of the response variable in the training data.
`varExpl`	numeric; variance explained by the PCR model. These are the squares of the singular values.
`totvar`	numeric; total variance in the training data

`call`	the matched call.
`tranFun`	transformation function used. `NA` if none supplied/used.
`tranParms`	list; meta parameters used to computed the transformed training data.
`performance`	data frame; cross-validation performance statistics for the model.
`ncomp`	numeric; number of principal components computed

Author(s)

Gavin L. Simpson

See Also

`wa`

Examples

```
## Load the Imbrie & Kipp data and
## summer sea-surface temperatures
data(ImbrieKipp)
data(SumSST)

## normal interface and apply Hellinger transformation
mod <- pcr(ImbrieKipp, SumSST, tranFun = Hellinger)
mod

## formula interface, but as above
mod2 <- pcr(SumSST ~ ., data = ImbrieKipp, tranFun = Hellinger)
mod2

## Several standard methods are available
fitted(mod, comps = 1:4)
resid(mod, comps = 1:4)
coef(mod, comps = 1:4)

## Eigenvalues can be extracted
eigenvals(mod)

## screeplot method
screeplot(mod)
```

`performance` *Transfer function model performance statistics*

Description

A simple extractor function to access the model performance statistics of transfer function models.

Usage

```
performance(object, ...)
```

Arguments

`object`	A transfer function object.
`...`	Arguments passed to other methods. Currently ignored.

Details

`performance` is a generic function for use with a number of fitted models objects in **analogue**. The available methods are:

`wa` Weighted Averaging Models.

`predict.wa` Predictions from a Weighted Average Model.

`pcr` Principal Component Regression models.

`bootstrap.wa` Bootstrapped Weighted Averaging Models.

`crossval` Cross-validated models fitted via `crossval`.

Value

A named vector containing the extracted model performance statistics.

Author(s)

Gavin L. Simpson

See Also

`wa`, `predict.wa`, `bootstrap.wa`.

Examples

```
data(ImbrieKipp)
data(SumSST)

## fit the WA model
mod <- wa(SumSST ~., data = ImbrieKipp)
mod

## the model performance statistics
performance(mod)
```

plot.dissimilarities *Plots the distribution of extracted dissimilarities*

Description

Produces a plot of the distribution of the extracted dissimilarities and a reference normal distribution with comparable mean and sd.

Usage

```
## S3 method for class 'dissimilarities'
plot(x, prob = 0.05,
     legend = TRUE, n.rnorm = 1e+05, col = "black",
     col.ref = "red", lty = "solid", lty.quant = "dotted",
     xlab = NULL, ylab = NULL, main = NULL, sub = NULL, ...)
```

Arguments

`x`	an object of class `"dissimilarities"`.
`prob`	numeric; density probability defining the threshold for close modern analogues.
`legend`	logical; draw a legend on the plotted figure?
`n.rnorm`	numeric; number of random normal deviates for reference line.
`col`, `col.ref`	colours for the dissimilarity and reference density functions drawn.
`lty`, `lty.quant`	line types for the dissimilarity and reference density functions drawn.
`xlab`, `ylab`	character; x- and y-axis labels.
`main`, `sub`	character; main and subtitle for the plot.
`...`	graphical arguments passed to other graphics functions.

Value

A plot on the currently active device.

Author(s)

Gavin L. Simpson

See Also

`dissimilarities`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## analog matching between SWAPImbrie & Kipp and V12.122 core
ik.analog <- analog(ImbrieKipp, V12.122, method = "chord")
ik.analog
summary(ik.analog)

## compare training set dissimilarities with normals
## and derive cut-offs
ik.dij <- dissim(ik.analog)
plot(ik.dij)
```

`plot.evenSample` *Plot distribution of samples along gradient*

Description

`plot` method for objects produced by `evenSample`. Draws a histogram of the number of samples per gradient segment.

Usage

```
## S3 method for class 'evenSample'
plot(x, add = FALSE, xlim = NULL, ylim = NULL, col = "grey",
     border = "grey", lty = NULL, ylab, xlab, main = NULL, sub = NULL,
     ann = TRUE, axes = TRUE, ...)
```

Arguments

`x`	an object of class `"evenSample"`, resulting from a call to `evenSample`
`add`	logical; should the histogram of counts be added to an existing plot?
`xlim, ylim`	numeric; user-specified axis limits. If not suplied suitable defaults are used
`col, border`	colours for the fill and border of the histogram bars respectively.
`lty`	the line type with which to draw the histogram bars

`ylab, xlab, main, sub`	character strings used to label the plot
`ann`	logical; should the default annotations be added to the plot. This relates to the plot main and sub titles and the x and y axis labels.
`axes`	logical; should plot axes be drawn?
`...`	additional arguments passed to/from other methods.

Value

A plot is draw is drawn on the currently active device.

Author(s)

Gavin L. Simpson

See Also

`evenSample` for the function used to create objects used by this plot method.

`plot.logitreg` *Produces plots of analogue logistic regression models*

Description

Draws the fitted logistic regression function describing the posterior probability that two sites are analogues conditional upon the dissimilarity between the two samples. Confidence intervals are also computed and displayed if requested.

Usage

```
## S3 method for class 'logitreg'
plot(x, group = "all", npred = 100,
     conf.int = 0.9, conf.type = c("none", "polygon", "lines"),
     xlab = expression(D[ij]), ylab = "Pr (A+ | d)",
     rug = TRUE, ticksize = 0.02,
     col = "red", ref.col = "lightgrey",
     lwd = 2, conf.lwd = 1, conf.lty = "dashed",
     shade = "lightgrey", ...)
```

Arguments

`x`	object to plot; an object of class `"logitreg"`, usually the result of a call to `logitreg`.
`group`	The group to plot the logit model for. Can be one of the group labels or `"Combined"` to draw the individual logit models. Alternatively, and the default, is to use `"all"`, which divides the plotting region into the required number of plotting regions and draws all the fitted curves.

`npred`	number of points at which the fitted curves are evaluated for plotting purposes.
`conf.int`	numeric; the confidence interval required.
`conf.type`	character; how should the confidence interval be drawn. Default is not to draw the confidence interval.
`xlab`, `ylab`	character; the x and y axis labels.
`rug`	logical; should rug plots be drawn?
`ticksize`	The size of the tick marks used in rug plots.
`col`	The colour in which to draw the line representing the fitted values.
`ref.col`	The colour of the reference lines drawn at 0 and 1.
`lwd`	The line width in which to draw the line representing the fitted values.
`conf.lwd`, `conf.lty`	Line width and line type for the confidence interval. Only used if `conf.type = "lines"` is specified.
`shade`	The colour for the fill and border of the confidence interval if `conf.type = "polygon"` is specified.
`...`	arguments passed on to `plot`.

Value

A plot on the current device.

Author(s)

Gavin L. Simpson

See Also

`logitreg` for an example, `roc`

`plot.mat` *Plot diagnostics for a mat object*

Description

Five plots (selectable by `which`) are currently available: a plot of estimated against observed values, a plot of residuals against estimated values, and screeplots of the apparent RMSE, average bias and maximum bias for MAT models of size k, where $k = 1, \ldots, n$. By default, the first three and '5' are provided.

Usage

```
## S3 method for class 'mat'
plot(x,
     which = c(1:3, 5),
     weighted = FALSE,
     k,
     caption = c("Inferred vs Observed", "Residuals vs Fitted",
                 "Leave-one-out errors", "Average bias",
                 "Maximum bias"),
     max.bias = TRUE,
     n.bias = 10,
     restrict = 20,
     sub.caption = NULL,
     main = "",
     ask = prod(par("mfcol")) < length(which) &&
                                 dev.interactive(),
     ...,
     panel = if (add.smooth) panel.smooth else points,
     add.smooth = getOption("add.smooth"))
```

Arguments

`x`	an object of class `"mat"`.
`which`	which aspects of the `"mat"` object to plot if a subset of the plots is required, specify a subset of the numbers `1:5`.
`weighted`	logical; should the analysis use weighted mean of env data of analogues as fitted/estimated values?
`k`	numeric; the number of analogues to use. If missing `k` is chosen automatically as the `k` that achieves lowest RMSE.
`caption`	captions to appear above the plots.
`max.bias`	logical, should max bias lines be added to residuals.
`n.bias`	numeric, number of sections to calculate maximum bias for.
`restrict`	logical; restrict comparison of k-closest model to $k \leq$ `restrict`.
`sub.caption`	common title-above figures if there are multiple; used as 'sub' (s.'title') otherwise. If `NULL`, as by default, a possibly shortened version of `deparse(x$call)` is used.
`main`	title to each plot-in addition to the above `caption`.
`ask`	logical; if `TRUE`, the user is *ask*ed before each plot, see `par(ask=.)`.
`...`	graphical arguments passed to other graphics functions.
`panel`	panel function. The useful alternative to `points`, `panel.smooth`, can be chosen by `add.smooth = TRUE`.
`add.smooth`	logical indicating if a smoother should be added to fitted & residuals plots; see also `panel` above.

Details

This plotting function is modelled closely on plot.lm and many of the conventions and defaults for that function are replicated here.

sub.caption - by default the function call - is shown as a subtitle (under the x-axis title) on each plot when plots are on separate pages, or as a subtitle in the outer margin (if any) when there are multiple plots per page.

Value

One or more plots, drawn on the current device.

Author(s)

Gavin L. Simpson. Code borrows heavily from plot.lm.

See Also

mat

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## MAT
ik.mat <- mat(ImbrieKipp, SumSST, method = "chord")

## summary plot of MAT model
layout(matrix(1:4, ncol = 2, byrow = TRUE))
plot(ik.mat)
layout(1)
```

plot.mcarlo *Plot Monte Carlo simulated dissimilarity distributions*

Description

A plot.lm-like plotting function for objects of class "mcarlo" to visualise the simulated distribution of dissimilarities.

Usage

```
## S3 method for class 'mcarlo'
plot(x,
     which = c(1:2),
     alpha = 0.05,
     caption = c("Distribution of dissimilarities",
       expression(paste("Simulated probability Pr (Dissim < ",
           alpha, ")"))),
     col.poly = "lightgrey",
     border.poly = "lightgrey",
     ask = prod(par("mfcol")) < length(which) &&
                                  dev.interactive(),
     ...)
```

Arguments

x	an object of class "mcarlo", usually the result of a call to mcarlo.
which	numeric; which of the plots should be produced?
alpha	numeric; the Monte Carlo significance level to be marked on the cumulative frequency plot.
caption	character, length 2; captions to appear above the plots.
col.poly, border.poly	character; the colour to draw the region and border of the polygon enclosing the Monte Carlo significance on the cummulative frequency plot.
ask	logical; should the function wait for user confirmation to draw each plot? If missing, the function makes a reasonable attempt to guess the current situation and act accordingly.
...	additional graphical parameters to be passed to the plotting functions. Currently ignored.

Details

The "Distribution of dissimilarities" plot produces a histogram and kernel density estimate of the distribution of simulated dissimilarity values.

The "Simulated probability" plot shows a cumulative probability function of the simulated dissimlarity values, and highlights the proportion of the curve that is less than alpha.

Value

One or more plots on the current device.

Author(s)

Gavin L. Simpson

References

Sawada, M., Viau, A.E., Vettoretti, G., Peltier, W.R. and Gajewski, K. (2004) Comparison of North-American pollen-based temperature and global lake-status with CCCma AGCM2 output at 6 ka. *Quaternary Science Reviews* **23**, 87–108.

See Also

`mcarlo`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## perform the modified method of Sawada (2004) - paired sampling,
## with replacement
ik.mcarlo <- mcarlo(ImbrieKipp, method = "chord", nsamp = 1000,
                    type = "paired", replace = FALSE)
ik.mcarlo

## plot the simulated distribution
layout(matrix(1:2, ncol = 1))
plot(ik.mcarlo)
layout(1)
```

`plot.minDC` *Plot of minimum dissimilarity per sample*

Description

Minimum dissimilarity is a useful indicator of reliability of reconstructions performed via MAT and other methods, and for analogue matching. Minimum dissimilarity for a sample is the smallest dissimilarity between it and the training set samples.

Usage

```
## S3 method for class 'minDC'
plot(x, depths, use.labels = FALSE,
          quantiles = TRUE, rev.x = TRUE, type = "l",
          xlim, ylim, xlab = "", ylab = "Dissimilarity",
          main = "", sub = NULL,
          col.quantile = "red", lty.quantile = "dotted",
          ...)
```

Arguments

`x`	an object of class `"minDC"`.
`depths`	numeric; a vector of depths for which predicted values exist or will be generated. Can be missing, in which case, **if** `use.labels = TRUE`, the function will attempt to derive suitable values for you. See Details below.
`use.labels`	logical; should `reconPlot` attempt to derive values for argument `depths` from the names of the predicted values? Only use if `depths` is missing. See Details below.
`quantiles`	logical; should the probability quantiles be drawn on the plot?
`rev.x`	logical; should the depth/age axis be reversed (drawn from high to low)?
`type`	type of line drawn. See `par` and argument `"type"`.
`xlab, ylab`	character; the x- and y-axis labels respectively.
`main, sub`	character; main title and subtitle for the plot.
`xlim, ylim`	numeric, length 2; the x- and y-limits for the plotted axes. If not provided, the function will calculate appropriate values to cover the range of plotted values and any quantile lines (if requested via `"quantiles"`.
`col.quantile`	colour in which to draw the quantile lines.
`lty.quantile`	line type in which to draw the quantile lines.
`...`	arguments to be passed to methods, such as graphical parameters (see `par`). Currently ignored.

Details

Conventionally, these plots are drawn on a depth or an age scale. Argument `depths` is used to provide the depth or age axis, against which the predicted values are plotted.

If `depths` is not provided, then the function will try to derive the appropriate values from the labels of the predictions if `use.labels = TRUE`. You must provide `depths` or set `use.labels = TRUE` otherwise an error will result. The derived labels will be coerced to numerics. If your labels are coercible, then you'll either get nonsense on the plot or an error from R. If so, provide suitable values for `depths`.

Value

A plot on the currently active device.

Author(s)

Gavin L. Simpson

See Also

minDC

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## fit the MAT model using the chord distance measure
(ik.mat <- mat(ImbrieKipp, SumSST, method = "chord"))

## reconstruct for the RLGH core data
v12.mat <- predict(ik.mat, V12.122)

## extract the minimum DC values
v12.mdc <- minDC(v12.mat)
v12.mdc

## draw a plot of minimum DC by time
plot(v12.mdc, use.labels = TRUE, xlab = "Depth (cm.)")
```

---

plot.prcurve *Plot a fitted principal curve in PCA space*

Description

Projects the principal curve into PCA space and draws it and the underlying data in a biplot.

Usage

```
## S3 method for class 'prcurve'
plot(x, axes = 1:2, scaling = 0, segments = TRUE,
     col = "red", col.seg = "forestgreen", lwd = 2,
     lwd.seg = 1, ...)

## S3 method for class 'prcurve'
lines(x, axes = 1:2, scaling = 0, segments = TRUE,
      col = "red", col.seg = "forestgreen", lwd = 2,
      lwd.seg = 1, ...)
```

Arguments

`x`	an object of class `"prcurve"`.
`axes`	numeric vector of length 2; this is passed to the `choices` argument of the `scores` function.
`scaling`	numeric; the scaling to use. See `scores.rda` for the available options. The default is not to scale the scores, but `scaling` = 1 might be a useful alternative.
`segments`	logical; should segments be drawn between the observed points to the location on the principal curve on to which they project.
`col`	The colour to draw the principal curve in.
`col.seg`	The colour to draw the segments in.
`lwd`, `lwd.seg`	The line thickness used to draw the principal curve and segments respectively.
`...`	additional arguments passed on to `points` when drawing the observations in PCA space.

Value

A plot on the currently active device. The function does not return anything.

Author(s)

Gavin L. Simpson

See Also

`prcurve`; `rda` for the code used to perform the PCA.

Examples

```
## Load the Abernethy Forest data
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit the principal curve using varying complexity of smoothers
## for each species
aber.pc <- prcurve(abernethy2, method = "ca", trace = TRUE,
                   vary = TRUE, penalty = 1.4)

## Plot the curve
plot(aber.pc)

## The lines() method can be used to add the principal curve to an
## existing plot
ord <- rda(abernethy2)
plot(ord, scaling = 1)
lines(aber.pc, scaling = 1)
```

`plot.residLen` *Plot method for residual lengths*

Description

Base graphics plot method for `residLen` objects.

Usage

```
## S3 method for class 'residLen'
plot(x, probs = c(0.9, 0.95, 0.99), ncol = 1,
     lcol = "red", llty = "dashed", xlab = NULL, ylab = NULL,
     main = "Residual distances", rug = TRUE, ...)
```

Arguments

`x`	Object of class `"residLen"`, the result of a call to `residLen`.
`probs`	numeric; vector of probability quantiles to compute from the sets of residual distances.
`ncol`	numeric; number of columns for the plot layout. Choices are `1` or `2`. Determines whether the histograms are plotted above or beside each other.
`lcol, llty`	colour and line-type for the quantiles.
`xlab, ylab`	Axis labels. If not supplied, suitable defaults are generated, depending on whether RDA or CCA was used as the underlying ordination model.
`main`	character; title for the plot.
`rug`	logical; should rug plots of the actual distances be drawn?
`...`	additional arguments passed to `plot`.

Value

A plot on the current device.

Returns, invisibly, a list with two components (`train` and `passive`), each and object of the type returned by `density`.

Author(s)

Gavin L. Simpson

See Also

`residLen`, `plot.residLen`, `histogram.residLen`, `densityplot.residLen`.

Examples

```
## load the Imbrie and Kipp example data
data(ImbrieKipp, SumSST, V12.122)

## squared residual lengths for Core V12.122
rlens <- residLen(ImbrieKipp, SumSST, V12.122)
rlens

## plot a histogram of the residual distances
plot(rlens)
```

`plot.roc` *Plot ROC curves and associated diagnostics*

Description

Produces up to four plots (selectable by `"which"`) from the results of a call to `roc`, including the ROC curve itself.

Usage

```
## S3 method for class 'roc'
plot(x,
     which = c(1:3,5),
     group = "Combined",
     prior = NULL,
     show.stats = TRUE,
     abline.col = "grey",
     abline.lty = "dashed",
     inGroup.col = "red",
     outGroup.col = "blue",
     lty = "solid",
     caption = c("ROC curve", "Dissimilarity profiles",
                 "TPF - FPF vs Dissimilarity",
                 "Likelihood ratios"),
     legend = "topright",
     ask = prod(par("mfcol")) < length(which) && dev.interactive(),
     ...)
```

Arguments

`x`	an object of class `"roc"`.
`which`	numeric vector; which aspects of `"roc"` object to plot if a subset of the plots is required, specify a subset of the numbers `1:5`.
`group`	character vector of length 1 giving the name of the group to plot.
`prior`	numeric vector of length 2 (e.g. `c(0.5, 0.5)`) specifiying the prior probabilities of analogue and no-analogue. Used to generate posterior probability of analogue using Bayes factors in plot 5 (`which = 5`).
`show.stats`	logical; should concise summary statistics of the ROC analysis be displayed on the plot?
`abline.col`	character string or numeric value; the colour used to draw vertical lines at the optimal dissimilarity on the plots.
`abline.lty`	Line type for indicator of optimal ROC dissimilarity threshold. See `par` for the allowed line types.
`inGroup.col`	character string or numeric value; the colour used to draw the density curve for the dissimilarities between sites in the same group.
`outGroup.col`	character string or numeric value; the colour used to draw the density curve for the dissimilarities between sites not in the same group.
`lty`	vector of at most length 2 (elements past the second in longer vectors are ignored) line types. The first element of `lty` will be used where a single line is drawn on a plot. Where two lines are drawn (for analogue and non-analogue cases), the first element pertains to the analogue group and the second element to the non-analogue group. See `par` for the allowed line types.
`caption`	vector of character strings, containing the captions to appear above each plot.
`legend`	character; position of legends drawn on plots. See Details section in `legend` for keywords that can be specified.
`ask`	logical; if `TRUE`, the user is *ask*ed before each plot, see `par(ask=.)`.
`...`	graphical arguments passed to other graphics functions.

Details

This plotting function is modelled closely on `plot.lm` and many of the conventions and defaults for that function are replicated here.

First, some definitions:

TPF True Positive Fraction, also known as *sensitivity*.

TNF True Negative Fraction, also known as *specificity*.

FPF False Positive Fraction; the complement of TNF, calculated as 1 - TNF. This is often referred to a 1 - specificity. A false positive is also known as a type I error.

FNF False Negative Fraction; the complement of TPF, calculated as 1 - TPF. A false negative is also known as a type II error.

AUC The Area Under the ROC Curve.

The "ROC curve" plot (`which = 1`,) draws the ROC curve itself as a plot of the False Positive Fraction against the True Positive Fraction. A diagonal 1:1 line represents no ability for the dissimilarity coefficient to differentiate between groups. The AUC statistic may also be displayed (see argument `"show.stats"` above).

The "Dissimilarity profile" plot (`which = 2`), draws the density functions of the dissimilarity values (*d*) for the correctly assigned samples and the incorrectly assigned samples. A dissimilarity coefficient that is able to well distinguish the sample groupings will have density functions for the correctly and incorrectly assigned samples that have little overlap. Conversely, a poorly discriminating dissimilarity coefficient will have density profiles for the two assignments that overlap considerably. The point where the two curves cross is the optimal dissimilarity or critical value, *d'*. This represents the point where the difference between TPF and FPF is maximal. The value of *d* at the point where the difference between TPF and FPF is maximal will not neccesarily be the same as the value of *d'* where the density profiles cross. This is because the ROC curve has been estimated at discrete points *d*, which may not include excatly the optimal *d'*, but which should be close to this value if the ROC curve is not sampled on too coarse an interval.

The "TPF - FPF vs Dissimilarity" plot, draws the difference between the ROC curve and the 1:1 line. The point where the ROC curve is farthest from the 1:1 line is the point at which the ROC curve has maximal slope. This is the optimal value for *d*, as discussed above.

The "Likelihood ratios" plot, draws two definitions of the slope of the ROC curve as the likelihood functions LR(+), and LR(-). LR(+), is the likelihood ratio of a positive test result, that the value of *d* assigns the sample to the group it belongs to. LR(-) is the likelihood ratio of a negative test result, that the value of *d* assigns the sample to the wrong group.

LR(+) is defined as $LR(+) = TPF/FPF$ (or sensitivity / (1 - specificity)), and LR(-) is defined as $LR(-) = FPF/TNF$ (or (1 - sensitivity) / specificity), in Henderson (1993).

The “probability of analogue” plot, draws the posterior probability of analogue given a dissimilarity. This is the LR(+) likelihood ratio values multiplied by the prior odds of analogue, for given values of the dissimilarity, and is then converted to a probability.

Value

One or more plots, drawn on the current device.

Author(s)

Gavin L. Simpson. Code borrows heavily from `plot.lm`.

References

Brown, C.D., and Davis, H.T. (2006) Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems* **80**, 24–38.

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Henderson, A.R. (1993) Assessing test accuracy and its clinical consequences: a primer for receiver operating characteristic curve analysis. *Annals of Clinical Biochemistry* **30**, 834–846.

See Also

roc for a complete example

---

plot.sppResponse *Plot species responses along gradients or latent variables*

Description

Observed abundances and fitted response curves along

Usage

```
## S3 method for class 'sppResponse'
plot(x,
     which = seq_along(x),
     display = c("observed", "fitted"),
     xlab = "Gradient", ylab = "Abundance",
     main = NULL,
     lcol = "red",
     lwd = 2,
     ...)
```

Arguments

`x`	an object of class `"sppResponse"`.
`which`	numeric or logical vector; which species (components of `x`) to plot.
`display`	character; plot the observed data or the fitted species response curve or both?
`xlab, ylab, main`	titles for the x and y axis label and the main title. If `main` is `NULL`, suitable main titles are taken from `x`.
`lwd`	numeric; width of the line used to draw the fitted species response function if drawn.
`lcol`	the colour to use to draw the fitted species response.
`...`	graphical arguments passed `plot` and `lines` used internally.

Details

Currently, no attempt is made to split the device into an appropriate number of panels. This is up to the user to decide. See the Examples section of `sppResponse` for one way to handle this.

Value

One or more plots, drawn on the current device.

Author(s)

Gavin L. Simpson

See Also

sppResponse for a complete example using fitted responses along principal curves.

`plot.wa` *Plot diagnostics for a weighted averaging model*

Description

Two plots (selectable by `which`) are currently available: a plot of estimated against observed values, a plot of residuals against estimated values.

Usage

```
## S3 method for class 'wa'
plot(x,
     which = 1:2,
     caption = c("Inferred vs Observed", "Residuals vs Fitted"),
     max.bias = TRUE,
     n.bias = 10,
     sub.caption = NULL,
     main = "",
     ask = prod(par("mfcol")) < length(which) &&
                                 dev.interactive(),
     ...,
     panel = if (add.smooth) panel.smooth else points,
     add.smooth = getOption("add.smooth"))
```

Arguments

`x`	an object of class `"wa"`.
`which`	which aspects of the `"wa"` object to plot if a subset of the plots is required, specify a subset of the numbers `1:2`.
`caption`	captions to appear above the plots.
`max.bias`	logical, should max bias lines be added to residuals.
`n.bias`	numeric, number of sections to calculate maximum bias for.
`sub.caption`	common title-above figures if there are multiple; used as 'sub' (s.'title') otherwise. If `NULL`, as by default, a possibly shortened version of `deparse(x$call)` is used.
`main`	title to each plot-in addition to the above `caption`.
`ask`	logical; if `TRUE`, the user is *ask*ed before each plot, see `par(ask=.)`.
`...`	graphical arguments passed to other graphics functions.
`panel`	panel function. The useful alternative to `points`, `panel.smooth`, can be chosen by `add.smooth = TRUE`.
`add.smooth`	logical indicating if a smoother should be added to fitted & residuals plots; see also `panel` above.

Details

This plotting function is modelled closely on `plot.lm` and many of the conventions and defaults for that function are replicated here.

`sub.caption` - by default the function call - is shown as a subtitle (under the x-axis title) on each plot when plots are on separate pages, or as a subtitle in the outer margin (if any) when there are multiple plots per page.

Value

One or more plots, drawn on the current device.

Author(s)

Gavin L. Simpson. Code borrows heavily from `plot.lm`.

See Also

`mat`

Examples

```
## see full example in ?wa
```

`Pollen` *North American Modern Pollen Database*

Description

A database of modern pollen samples from a network of sites from North America and Greenland, compiled by Whitmore et al. (2005). Associated climatic and vegetation data are also record. The version of the NAMPD included here is latest version 1-7.3 (February 2013), as of January 2016.

Usage

```
data(Pollen)

data(Climate)

data(Biome)

data(Location)
```

Format

For `Pollen`, a data frame of 4833 samples and 135 columns (the unique identifier and 134 pollen taxa).

For `Biome`, a data frame of 4833 samples on, currently, a single vegetation variable (plus the unique identifier):

`ID2` Unique, sequential number assigned by NAMPD.

`Fedorova` Factor; Vegetation type (Biome) See Whitmore et al. (2005), Figs 3 & 4. Reclassified biomes from Fedorova et al (1994).

For `Location`, a data frame of the latitude and longitude locational data for 4833 samples.

`ID2` Unique, sequential number assigned by NAMPD.

`Latitude` Latitude of the sampling location in decimal degrees.

`Longitude` Longitude of the sampling location in decimal degrees.

For `Climate`, a data frame with 4833 observations on the following 32 variables.

`ID2` Unique, sequential number assigned by NAMPD.

`t[jan:dec]` numeric vectors; Mean monthly temperatures for the indicated month. Degrees C.

`p[jan:dec]` numeric vectors; Mean total monthly precipitation (mm) for the indicated month.

`tave` numeric; annual average temperature (Degrees C)

`tmax` numeric; maximum temperature (in Degrees C) observed over the period of record.

`tmin` numeric; minimum temperature (in Degrees C) observed over the period of record.

`gdd0` numeric; Growing degree days computed using a base of 0 degrees C.

`gdd5` numeric; Growing degree days computed using a base of 5 degrees C.

`mtco` numeric; mean temperature of the coldest month.

`mtwa` numeric; mean temperature of the warmest month.

`annp` numeric; mean annual total precipitation (mm).

Details

These datasets were extracted from Version 1.7 of the North American Modern Pollen Database.

All pollen species were included, however, only the Vegetation type (Biome) field of the AVHRR data and selected Climatic variables were extracted. Requests for additional variables to be included in the versions of the data included in the package should me sent to the package maintainer.

Warning

Note that the data for the pollen species are a mixture of types. The `DataForm` variable contains information on the type of data included for each site. The codes are:

RC raw counts

RP raw counts expressed as percentages

DC digitised counts

DP digitised counts expressed as percentages

PM counts in permille

This value is not known for all samples.

Source

The database is archived electronically at two sites:

http://www.lpc.uottawa.ca/data/modern/index.html

https://williamspaleolab.github.io/datavis/

References

Whitmore, J., Gajewski, K., Sawada, M., Williams, J.W., Shuman, B., Bartlein, P.J., Minckley, T., Viau, A.E., Webb III, T., Anderson, P.M., and Brubaker L.B., 2005. North American and Greenland modern pollen data for multi-scale paleoecological and paleoclimatic applications. *Quaternary Science Reviews* **24**:1828–1848.

Williams, J.W., Shuman, B., Bartlein, P.J., Whitmore, J., Gajewski, K., Sawada, M., Minckley, T., Shafer, S., Viau, A.E., Webb, III, T., Anderson, P.M., Brubaker, L.B., Whitlock, C. and Davis, O.K., 2006. *An Atlas of Pollen-Vegetation-Climate Relationships for the United States and Canada*. American Association of Stratigraphic Palynologists Foundation, Dallas, TX, 293p.

Williams, J.W. and Shuman, B., 2008. Obtaining accurate and precise environmental reconstructions from the modern analog technique and North American surface pollen dataset. *Quaternary Science Reviews*, **27**: 669–687.

Fedorova, I.T., Volkova, Y.A., Varlyguin, E., 1994. *World vegetation cover. Digital raster data on a 30-minute cartesian orthonormal geodetic (lat/long) 1080x2160 grid.* In: Global Ecosystems Database Version 2.0. USDOC/NOAA National Geophysical Data Center, Bould, CO.

Examples

```
data(Pollen)

data(Climate)

data(Biome)

data(Location)
```

---

prcurve *Fits a principal curve to m-dimensional data*

Description

A principal curve is a non-parametric generalisation of the principal component and is a curve that passes through the *middle* of a cloud of data points for a certain definition of ‘middle’.

Usage

```
prcurve(X, method = c("ca", "pca", "random", "user"), start = NULL,
        smoother = smoothSpline, complexity, vary = FALSE,
        maxComp, finalCV = FALSE, axis = 1, rank = FALSE,
        stretch = 2, maxit = 10, trace = FALSE, thresh = 0.001,
        plotit = FALSE, ...)

initCurve(X, method = c("ca", "pca", "random", "user"), rank = FALSE,
          axis = 1, start)
```

Arguments

`X`	a matrix-like object containing the variables to which the principal curve is to be fitted.
`method`	character; method to use when initialising the principal curve. `"ca"` fits a correspondence analysis to `X` and uses the `axis`-th axis scores as the initial curve. `"pca"` does the same but fits a principal components analysis to `X`. `"random"` produces a random ordering as the initial curve.
`start`	numeric vector specifying the initial curve when `method = "user"`. Must be of length `nrow(X)`.
`smoother`	function; the choice of smoother used to fit the principal curve. Currently, the only options are `smoothSpline`, which is a wrapper to `smooth.spline`, and `smoothGAM`, which is a wrapper to `gam`.
`complexity`	numeric; the complexity of the fitted smooth functions. The function passed as argument `smoother` should arrange for this argument to be passed on to relevant aspect of the underlying smoother. In the case of `smoothSpline`, complexity is the `df` argument of `smooth.spline`.
`vary`	logical; should the complexity of the smoother fitted to each variable in `X` be allowed to vary (i.e. to allow a more or less smooth function for a particular variable. If `FALSE` the median complexity over all *m* variables is chosen as the fixed complexity for all *m* smooths.
`maxComp`	numeric; the upper limt on the allowed complexity.
`finalCV`	logial; should a final fit of the smooth function be performed using cross validation?
`axis`	numeric; the ordinaion axis to use as the initial curve.
`rank`	logical; should rank position on the gradient be used? Not yet implemented.
`stretch`	numeric; a factor by which the curve can be extrapolated when points are projected. Default is 2 (times the last segment length).
`maxit`	numeric; the maximum number of iterations.
`trace`	logical; print progress on the iterations be printed to the console?
`thresh`	numeric; convergence threshold on shortest distances to the curve. The algorithm is considered to have converged when the latest iteration produces a total residual distance to the curve that is within `thresh` of the value obtained during the previous iteration.
`plotit`	logical; should the fitting process be plotted? If `TRUE`, then the fitted principal curve and observations in `X` are plotted in principal component space.
`...`	additional arguments are passed solely on to the function `smoother`.

Value

An object of class "prcurve" with the following components:

s	a matrix corresponding to X, giving their projections onto the curve.
tag	an index, such that s[tag,] is smooth.
lambda	for each point, its arc-length from the beginning of the curve.
dist	the sum-of-squared distances from the points to their projections.
converged	logical; did the algorithm converge?
iter	numeric; the number of iterations performed.
totalDist	numeric; total sum-of-squared distances.
complexity	numeric vector; the complexity of the smoother fitted to each variable in X.
call	the matched call.
ordination	an object of class "rda", the result of a call to rda. This is a principal components analysis of the input data X.
data	a copy of the data used to fit the principal curve.

Note

The fitting function uses function project_to_curve in package **princurve** to find the projection of the data on to the fitted curve.

Author(s)

Gavin L. Simpson

See Also

smoothGAM and smoothSpline for the wrappers fitting smooth functions to each variable.

Examples

```
## Load Abernethy Forest data set
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit the principal curve using the median complexity over
## all species
aber.pc <- prcurve(abernethy2, method = "ca", trace = TRUE,
                   vary = FALSE, penalty = 1.4)

## Extract fitted values
fit <- fitted(aber.pc) ## locations on curve
abun <- fitted(aber.pc, type = "smooths") ## fitted response

## Fit the principal curve using varying complexity of smoothers
## for each species
aber.pc2 <- prcurve(abernethy2, method = "ca", trace = TRUE,
                    vary = TRUE, penalty = 1.4)

## Predict new locations
take <- abernethy2[1:10, ]
pred <- predict(aber.pc2, take)

## Not run:
## Fit principal curve using a GAM - currently slow ~10secs
aber.pc3 <- prcurve(abernethy2 / 100, method = "ca", trace = TRUE,
                   vary = TRUE, smoother = smoothGAM, bs = "cr", family = mgcv::betar())

## End(Not run)
```

`predict.logitreg` *Posterior probability of analogue-ness for fossil samples*

Description

Predict the posterior probability of analogue-ness between fossil and training set samples based on logistic regression fits.

Usage

```
## S3 method for class 'logitreg'
predict(object, newdata, group = "all", k = 1, ...)
```

Arguments

`object`	an object of class `"logitreg"`, resulting from a call to `logitreg`.
`newdata`	an object of class `"distance"` representing the dissimilarity between fossil samples and those from the training set.
`group`	character; for which group(s) are predictions sought. `"all"`, the default, indicates that predictions for all groups and the combined analysis are produced. Currently ignored.
`k`	numeric; the number of close modern analogues to each fossil sample to consider. Not currently used and may be removed from the method as varying `k` should require refitting the logistic regression model with that number of close modern analogues considered.
`...`	additional arguments passed to other methods.

Value

A matrix of posterior probabilities of analogue-ness is returned. The rows represent the fossil samples and the columns the groupings. There are $g+1$ columns where g is the number of groups. The $g+1$th column represents the Combined analysis which is the overall posterior probability that a fossil sample is an analogue to a training set sample (i.e. to any one of the groups).

Author(s)

Gavin L. Simpson

See Also

See `logitreg` for example usage. `cma` for extraction of close modern analogue information and `analog` for the underlying computations.

`predict.mat` *Predict method for Modern Analogue Technique models*

Description

Predicted values based on a MAT model object.

Usage

```
## S3 method for class 'mat'
predict(object, newdata, k, weighted = FALSE,
        bootstrap = FALSE, n.boot = 1000,
        probs = c(0.01, 0.025, 0.05, 0.1), ...)
```

Arguments

`object`	an object of `mat`.
`newdata`	data frame; required only if predictions for some new data are required. Mst have the same number of columns, in same order, as `x` in `mat`. See example below or `join` for how to do this. If `newdata` not provided, the fitted values are returned.
`k`	number of analogues to use. If missing, `k` is chosen automatically as the `k` that achieves lowest RMSE.
`weighted`	logical; should the analysis use the weighted mean of environmental data of analogues as predicted values?
`bootstrap`	logical; should bootstrap derived estimates and sample specific errors be calculated-ignored if `newdata` is missing.
`n.boot`	numeric; the number of bootstrap samples to take.
`probs`	numeric; vector of probabilities with values in [0,1].
`...`	arguments passed to of from other methods.

Details

This function returns one or more of three sets of results depending on the supplied arguments:

Fitted values: the fitted values of the `mat` model are returned if `newdata` is missing.

Predicted values: the predicted values for some new samples are returned if `newdata` is supplied. Summary model statistics and estimated values for the training set are also returned.

Bootstrap predictions and standard errors: if `newdata` is supplied and `bootstrap = TRUE`, the predicted values for `newdata` plus bootstrap estimates and standard errors for the new samples and the training set are returned.

The latter is simply a wrapper for `bootstrap(model, newdata, ...)` - see `bootstrap.mat`.

Value

A object of class `predict.mat` is returned if `newdata` is supplied, otherwise an object of `fitted.mat` is returned. If `bootstrap = FALSE` then not all components will be returned.

`observed`	vector of observed environmental values.
`model`	a list containing the model or non-bootstrapped estimates for the training set. With the following components: `estimated` estimated values for `"y"`, the environment. `residuals` model residuals. `r.squared` Model R^2 between observed and estimated values of `"y"`. `avg.bias` Average bias of the model residuals. `max.bias` Maximum bias of the model residuals. `rmsep` Model error (RMSEP). `k` numeric; indicating the size of model used in estimates and predictions.
`bootstrap`	a list containing the bootstrap estimates for the training set. With the following components: `estimated` Bootstrap estimates for `"y"`. `residuals` Bootstrap residuals for `"y"`. `r.squared` Bootstrap derived R^2 between observed and estimated values of `"y"`. `avg.bias` Average bias of the bootstrap derived model residuals. `max.bias` Maximum bias of the bootstrap derived model residuals. `rmsep` Bootstrap derived RMSEP for the model. `s1` Bootstrap derived S1 error component for the model. `s2` Bootstrap derived S2 error component for the model. `k` numeric; indicating the size of model used in estimates and predictions.
`sample.errors`	a list containing the bootstrap-derived sample specific errors for the training set. With the following components: `rmsep` Bootstrap derived RMSEP for the training set samples. `s1` Bootstrap derived S1 error component for training set samples. `s2` Bootstrap derived S2 error component for training set samples.

`weighted`	logical; whether the weighted mean was used instead of the mean of the environment for *k*-closest analogues.
`auto`	logical; whether `"k"` was choosen automatically or user-selected.
`n.boot`	numeric; the number of bootstrap samples taken.
`predictions`	a list containing the model and bootstrap-derived estimates for the new data, with the following components:
	`observed` the observed values for the new samples — only if `newenv` is provided.
	`model` a list containing the model or non-bootstrapped estimates for the new samples. A list with the same components as `model`, above.
	`bootstrap` a list containing the bootstrap estimates for the new samples, with some or all of the same components as `bootstrap`, above.
	`sample.errors` a list containing the bootstrap-derived sample specific errors for the new samples, with some or all of the same components as `sample.errors`, above.
`method`	the dissimilarity measure used to fit the underlying MAT models.
`quantiles`	probability quantiles of the pairwise dissimilarities computed from the training set.
`minDC`	smallest distances between each sample in `newdata` and the training set samples.
`Dij`	dissimilarities between `newdata` and training set samples.

Author(s)

Gavin L. Simpson

References

Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C. and ter Braak, C.J.F. (1990). Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London; Series B*, **327**; 263–278.

See Also

`mat`, `bootstrap.mat`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
```

```
V12.122 <- dat[[2]] / 100

## fit the MAT model using the chord distance measure
(ik.mat <- mat(ImbrieKipp, SumSST, method = "chord"))

## predict for V12.122 data
predict(ik.mat, V12.122)
```

---

predict.pcr — *Predicted values from a principal components regression*

Description

Calculates predicted values from a fitted principal components regression model. Leave-one-out, bootstrap or n k-fold crossvalidated predictions are also implemented.

Usage

```
## S3 method for class 'pcr'
predict(object, newdata, ncomp = object$ncomp,
        CV = c("none", "LOO", "bootstrap", "kfold"),
        verbose = FALSE, nboot = 100, kfold = 10, folds = 5,
        ...)
```

Arguments

`object`	a fitted model of class `"pcr"`, the result of a call to `pcr`.
`newdata`	data frame of new observations for which predictions are sought.
`ncomp`	numeric; the PCR components for which predictions are sought. If `ncomp = c`, predictions for components `1:c` are produced.
`CV`	character; the type of crossvalidation required. Currently, no crossvalidation methods are implemented.
`verbose`	logical; should progress on crossvalidation be printed to the console?
`nboot`	numeric; the number of bootstrap samples to draw.
`kfold`	numeric; the number of folds to split data into.
`folds`	numeric; the number of repetitions of k-fold CV.
`...`	arguments passed to other methods.

Details

`predict.pcr` arranges for any transformation applied to the training data to be applied to the `newdata` prior to prediction.

Value

A matrix of predicted values with rows representing samples in `newdata` and columns, the PCR components requested via `ncomp`.

Author(s)

Gavin L. Simpson

See Also

`pcr`

Examples

```
## Load the Imbrie & Kipp data and
## summer sea-surface temperatures
data(ImbrieKipp)
data(SumSST)

## choose 10 samples to act as a test set, for illustration
take <- c(5,58,31,51,42,28,30,57,8,50)

## normal interface and apply Hellinger transformation
mod <- pcr(ImbrieKipp[-take, ], SumSST[-take], tranFun = Hellinger)

## predictions
predict(mod, ImbrieKipp[take, ], ncomp = 4)

## predictions
set.seed(123)
predict(mod, ImbrieKipp[take, ], ncomp = 4, CV = "bootstrap",
        nboot = 100)
```

`predict.prcurve` *Predict new locations & fitted values on a principal curve*

Description

Locations on a fitted principal curve are predicted by projecting the new observations in m dimensions on to the corresponding closest point on the curve. Fitted values for data used to fit the curve are available with respect to the principal curve or to the individual smooth functions.

Usage

```
## S3 method for class 'prcurve'
predict(object, newdata, ...)

## S3 method for class 'prcurve'
fitted(object, type = c("curve","smooths"), ...)
```

Arguments

`object`	an object of class `prcurve`.
`newdata`	a matrix or data frame of new observations within the space of the orginal data. Variables are matched against those of the original data via their `names` or `colnames`. If a data frame is supplied, it is converted to a matrix via `data.matrix`.
`type`	character; the type of fitted values to return.
`...`	other arguments passed to other methods. Not currently used.

Details

Fitting a principal curve involves two procedures. In one, the current curve is bent towards the data via the fitting of spline functions with distance along the curve as the predictor variable and each variable in turn as the response. The second procedure, a projection step, involves projecting the observed points in m dimensions on to locations along the current curve to which they are closest in the hyperspace.

Given a fitted curve, the projection step can be used to find new points on the fitted curve by projecting the new points located in the hyperspace on to points on the curve to which they are closest.

Fitted values are available for the data used to the fit the principal curve. There are two types of fitted value available. For `type = "curve"`, the fitted locations on the principal curve. For `type = "smooths"`, the fitted values of the variables from the individual smooth functions with respect to distance along the principal curve.

Value

A matrix of points in the space of the original data. Rows correspond to the new samples and columns to the variables (ordered as per the original data used to fit the curve).

How these points are ordered along the fitted curve is contained in attributed `tag`.

Author(s)

Gavin L. Simpson

See Also

See `prcurve` for details on fitting principal curves and an example.

`predict.wa` *Predict from a weighted average model*

Description

Model predictions and cross-validation predictions for weighted averaging transfer function models.

Usage

```
## S3 method for class 'wa'
predict(object, newdata,
        CV = c("none", "LOO", "bootstrap", "nfold"),
        verbose = FALSE, n.boot = 100, nfold = 5, ...)
```

Arguments

`object`	an object of class `"wa"`, usually the result of a call to `wa`
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`CV`	Should cross-validation be performed? Leave-one-out (`"LOO"`), bootstrap (`"bootstrap"`) and k-fold (`"nfold"`) CV are currently available.
`verbose`	Should CV progress be printed to the console?
`n.boot`	The number of bootstrap samples or k-fold steps.
`nfold`	Number of subsets in k-fold CV.
`...`	further arguments passed to or from other methods.

Details

Not all CV methods produce the same output. `CV = "bootstrap"` and `CV = "nfold"` produce sample specific errors.

Value

An object of class `"predict.wa"`, a list with the following components:

`pred`	A list with components `pred` and `rmsep` containing the predicted values and the sample specific errors if available.
`performance`	A list with model performance statistics.
`model.pred`	A list with components `pred` and `rmsep` containing the predicted values for the training set samples and the sample specific errors if available.
`call`	the matched function call.
`CV.method`	The CV method used.

Author(s)

Gavin L. Simpson and Jari Oksanen (k-fold CV)

References

Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C. and ter Braak, C.J.F. (1990). Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London; Series B*, **327**; 263–278.

See Also

`wa`, `predict.mat`, `performance`, `reconPlot`.

Examples

```
## Imbrie and Kipp
data(ImbrieKipp)
ImbrieKipp <- ImbrieKipp / 100
data(SumSST)
ik.wa <- wa(SumSST ~ ., data = ImbrieKipp, tol.dw = TRUE,
            min.tol = 2, small.tol = "min")
ik.wa

## load V12.122 core data
data(V12.122)
V12.122 <- V12.122 / 100

## predict summer sea-surface temperature for V12.122 core
set.seed(2)
v12.pred <- predict(ik.wa, V12.122, CV = "bootstrap", n.boot = 100)

## draw the fitted reconstruction
reconPlot(v12.pred, use.labels = TRUE, display = "bars")

## extract the model performance stats
performance(v12.pred)
```

rankDC *Rank correlation between environmental and species dissimilarities.*

Description

Computes the rank correlation between distances between sites in terms of gradient variables, such as environmental ones, and species composition.

Usage

```
rankDC(g, y, dc = c("chord", "bray", "euclidean", "chi.square", "gower"),
       method = "spearman")

## S3 method for class 'rankDC'
plot(x, sort = TRUE, decreasing = FALSE,
     xlab = "Rank correlation", color = "blue",
     pch = 21, bg = "blue", lcolor = "grey",
     lty = "solid", ...)

## S3 method for class 'rankDC'
dotplot(x, data = NULL, sort = TRUE, decreasing = FALSE,
     xlab = "Rank correlation", ...)
```

Arguments

`g`	the gradient values; usually a data frame of environmental data.
`y`	the species data; usually a data frame.
`dc`	character; the set of dissimilarity coefficients for which rank correlations with gradient distance are to be computed.
`method`	character; the correlation method to use. See the `method` argument of `cor`.
`x`	an object of class `"rankDC"`.
`data`	NULL; ignored, only present for purposes of conformance to generic definition.
`sort, decreasing`	logical; should observations be sorted prior to plotting? If so, should they be sorted in order of decreasing size?
`xlab`	The x-axis label for the plot.
`color, pch, bg, lcolor, lty`	arguments passed to `dotchart`.
`...`	arguments passed to other methods, including `dotchart` and `dotplot`.

Value

A named vector of rank correlations is returned.

Author(s)

Gavin L. Simpson, based on `rankindex` from the vegan package.

See Also

`rankindex` from package vegan. For the `dotplot` method, see `dotplot`.

Examples

```
data(swappH)
data(swapdiat)

rc <- rankDC(swappH, swapdiat, dc = c("chord","euclidean","gower"))

## base plot uses dotchart()
plot(rc)

## lattice version of the base plot
dotplot(rc)
```

reconPlot *Stratigraphic plots of palaeoenvironmental reconstructions*

Description

Draws a palaeoenvironmental reconstruction of predicted environmental values for sub-fossil assemblages.

Usage

```
reconPlot(x, ...)

## Default S3 method:
reconPlot(x, depths, errors,
          display.error = c("none", "bars", "lines"),
          rev.x = TRUE,
          col.error = "grey", lty.error = "dashed",
          type = "l",
          xlim, ylim,
          xlab = "", ylab = "", main = "",
          ...)

## S3 method for class 'predict.mat'
reconPlot(x, depths, use.labels = FALSE,
          predictions = c("model", "bootstrap"),
          display.error = c("none", "bars", "lines"),
          sample.specific = TRUE, ...)

## S3 method for class 'predict.wa'
reconPlot(x, depths, use.labels = FALSE,
          display.error = c("none", "bars", "lines"),
          sample.specific = TRUE, ...)
```

Arguments

`x`	An R object.
`depths`	numeric; a vector of depths for which predicted values exist or will be generated. Can be missing, in which case, **if** `use.labels = TRUE`, the function will attempt to derive suitable values for you. See Details below.
`errors`	numeric; a vector of errors for plotting error bars or lines.
`display.error`	character; hown should error bars be drawn on the plot? One of `"none"`, `"bars"`, or `"lines"`. If `"bars"`, error bars are drawn for each sample. If `"lines"`, lines are drawn enclosing the region prediction +/- RMSEP.
`rev.x`	logical; should the depth/age axis be reversed (drawn from high to low)?
`col.error, lty.error`	the colour and type of line drawn. See `par` and arguments `"col"` and `"lty"`.

`type`	type of line drawn. See `par` and argument `"type"`.
`xlab, ylab`	character; the x- and y-axis labels respectively.
`main`	character; main title for the plot.
`xlim, ylim`	numeric, length 2; the x- and y-limits for the plotted axes. If not provided, the function will calculate appropriate values to cover the range of plotted values and any error bars (if requested via `"error.bars"`.
`use.labels`	logical; should `reconPlot` attempt to derive values for argument `depths` from the names of the predicted values? Only use if `depths` is missing. See Details below.
`predictions`	character; one of `"model"` or `"bootstrap"`. Which type of predicted values should be plotted? The model predictions (`"model"`) or the bootstrap-derived predictions (`"bootstrap"`).
`sample.specific`	logical; should sample specific errors be used? Only for `predictions = "bootstrap"`.
`...`	arguments passed to other graphics functions.

Details

Conventionally, these plots are drawn on a depth or an age scale. Argument `depths` is used to provide the depth or age axis, against which the predicted values are plotted.

If `depths` is not provided, then the function will try to derive the appropriate values from the labels of the predictions if `use.labels = TRUE`. You must provide `depths` or set `use.labels = TRUE` otherwise an error will result. The derived labels will be coerced to numerics. If your labels are not coercible, then you'll either get nonsense on the plot or an error from R. If so, provide suitable values for `depths`.

Value

A plot on the currently active device.

Author(s)

Gavin L. Simpson

See Also

`mat`, and `predict.mat` for MAT transfer functions and `wa` and `predict.wa` for WA models.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)
```

```
## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## Fit a MAT model
(ik.mat <- mat(ImbrieKipp, SumSST, method = "chord"))

## Reconstruct pH for the RLGH core
v12.pH <- predict(ik.mat, V12.122)

## draw the reconstruction
reconPlot(v12.pH, use.labels = TRUE, display.error = "bars",
          xlab = "Depth", ylab = "Summer Seas-surface Temperature")
```

`residLen` *Squared residual length diagnostics*

Description

The squared residual length between the fitted values of a constrained ordination and the original species data is one diagnostic for transfer function models.

Usage

```
residLen(X, env, passive, method = c("cca","rda"))

fittedY(ord, newdata, colsum)

sqrlLinear(Y, fitted)

sqrlUnimodal(Y, colsum, fitted)
```

Arguments

`X`	data frame; the training set species data.
`env`	vector; the training set environmental data.
`passive`	data frame; the passive samples species data.
`method`	the ordination technique to use. One of `"rda"` or `"cca"`, with the latter the default.
`ord`	an ordination object, the result of a call to `cca` or `rda`.
`newdata`	Species data matrix for passive samples. Must have same columns as data used to fit `ord`.
`colsum`	column (species) sums for training set data used to fit `ord`.
`Y`	Original species data matrix, the response for which squared residual lengths are to be computed.
`fitted`	The fitted values of the response derived from the constrained ordination model.

Details

The squared residual lengths are computed for the training set samples and the passive samples separately. Passive samples that are poorly fitted in the transfer function model will have large squared residual distances between the observed species data and the fitted values from the constrained ordination.

`residLen` is the main user-interface function and can be called with either the training data and passive samples.

`fittedY` returns the fitted approximation of the passive sample response data (i.e. species data). `sqrlLinear` and `sqrlUnimodal` return the squared residual distances between the observed species data and the fitted values from the constrained ordination model.

Value

`fittedY` returns a matrix of fitted species abundances for passive samples.

`sqrlLinear` and `sqrlUnimodal` return a vector of residual distances.

`residLen` returns an object of class `"residLen"` with the attribute `"method"` set to `"method"`. This object is a list with the following components:

`train, passive`	The squared residual lengths for the training set and the passive samples.
`ordination`	The fitted ordination.
`call`	The matched call.

Author(s)

Gavin L. Simpson

References

Ter Braak C.J.F. and Smilauer P. (2002) CANOCO Reference manual and CanoDraw for Windows User's guide: Software for Canonical Ordination (version 4.5). Microcomputer Power (Ithaca, NY, USA), 500pp.

See Also

`cca` and `predict.cca` for some of the underlying computations.

Examples

```
## load the Imbrie and Kipp example data
data(ImbrieKipp, SumSST, V12.122)

## squared residual lengths for Core V12.122
rlens <- residLen(ImbrieKipp, SumSST, V12.122)
rlens

## as before but using linear RDA
residLen(ImbrieKipp, SumSST, V12.122, method = "rda")
```

`residuals.prcurve` *Residuals of a principal curve fit.*

Description

Returns various representations of the residuals of a principal curve fit.

Usage

```
## S3 method for class 'prcurve'
residuals(object, which = c("distance", "raw", "smooths", "pca"),
          ...)
```

Arguments

`object`	an object of class `"prcurve"`, the result of a call to `prcurve`.
`which`	character; the type of residuals to return. See Details.
`...`	arguments passed to other methods. See Details.

Details

Various types of residual are available for the principal curve. In a departure from the usual convention, which residuals are returned is controlled via the `which` argument. This is to allow users to pass a `type` argument to the `residuals` method for the function used to fit the individual smooth functions when `which = "smooths"`.

The types of residuals available are

`"distance"` the default residual for a principal curve. This residual is taken as the Euclidean distance between each observations and the point on the principal curve to which it projects, in full multivariate space.

`"raw"` raw residuals are the basis for `"distance"` residuals, and are the difference between the observed and fitted values (position on the curve) for each observation in terms of each variable in the data set. These residuals are in the form of a matrix with number of observation *rows* and number of variables *cols*.

`"smooths"` these residuals are the result of calling `residuals()` on each of the smooth models fitted to the individual variables. See below for further details. A matrix of the same dimensions as for `which = "raw"` is returned.

`"pca"` similar to the raw residuals, but expressed in terms of the principal components of the input data. Hence these residuals are the difference between each observation's location in PCA space and their corresponding location on the curve.

For `"smooths"` residuals, what is returned is governed by the `residuals` method available for the smooth model fitted to the individual variables. For principal curves fitted using the `smoothSpline` plugin, see `smooth.spline`. For principal curves fitted via the `smoothGAM` plugin, see `residuals.gam`.

`...` can be used to pass additional arguments to these `residuals` methods. In particular, the `type` argument is commonly used to choose which type of residual is returned by the specific methods.

In the case of principal curves fitted using the plugin `smoothSpline`, residuals for `which = "smooths"` are only available if the the additional argument `keep.data` was specified during fitting via `prcurve`. See the examples for an illustration of this usage.

Value

A vector of residual distances (`which = "distance"`) or a matrix of residuals (for the other options).

Author(s)

Gavin L. Simpson

See Also

`prcurve` for fitting a principal curve.

Examples

```
## Load Abernethy Forest data set
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit the principal curve, preserving the data in the smooth.spline
## smooth functions fitted via keep.data = TRUE
aber.pc <- prcurve(abernethy2, method = "ca", keep.data = TRUE)

## default "distance" residuals
res <- resid(aber.pc)
head(res)

## residuals from the underlying smooth models, also illustrates
## how to select specific types of residual from the individual
## method using argument 'type'
res <- resid(aber.pc, which = "smooths", type = "deviance")
dim(res)
head(res[, 1:5])  # just show a few species residuals
```

`rlgh` *Round Loch of Glenhead Diatoms*

Description

Diatom sub-fossil assemblage counts from the Round Loch of Glenhead.

Usage

```
data(rlgh)
```

Format

A data frame with 101 observations on 139 diatom species.

Details

The samples are taken from various depths down a sediment core retrieved from the Round Loch of Glenhead, Galloway, Scotland.

References

Jones, V.J., Stevenson, A.C. and Battarbee, R.W. (1989) Acidification of lakes in Galloway, south west Scotland — A diatom and pollen study of the post-glacial history of the Round Loch of Glenhead. *Journal of Ecology* **77(1)**, 1–23.

Examples

```
data(rlgh)
```

`RMSEP` *Root mean square error of prediction*

Description

Calculates or extracts the RMSEP from transfer function models.

Usage

```
RMSEP(object, ...)

## S3 method for class 'mat'
RMSEP(object, k, weighted = FALSE,
          ...)

## S3 method for class 'bootstrap.mat'
RMSEP(object, type = c("birks1990", "standard"),
          ...)

## S3 method for class 'bootstrap.wa'
RMSEP(object, type = c("birks1990", "standard"),
          ...)
```

Arguments

`object`	An R object.
`k`	numeric; the number of analogues to use in calculating the RMSEP. May be missing. If missing, `k` is extracted from the model using `getK`.

`weighted`	logical; Return the RMSEP for the weighted or unweighted model? The default is for an unweighted model.
`type`	The type of RMSEP to return/calculate. See Details, below.
`...`	Arguments passed to other methods.

Details

There are two forms of RMSEP in common usage. Within palaeoecology, the RMSEP of Birks et al. (1990) is most familiar:

$$\mathrm{RMSEP} = \sqrt{s_1^2 + s_2^2}$$

where where s_1 is the standard deviation of the out-of-bag (OOB) residuals and s_2 is the mean bias or the mean of the OOB residuals.

In the wider statistical literature, the following form of RMSEP is more commonly used:

$$\mathrm{RMSEP} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

where y_i are the observed values and $\hat{y}_i$ the transfer function predictions/fitted values.

The first form of RMSEP is returned by default or if `type = "birks1990"` is supplied. The latter form is returned if `type = "standard"` is supplied.

The RMSEP for objects of class `"mat"` is a leave-one-out cross-validated RMSEP, and is calculated as for `type = "standard"`.

Value

A numeric vector of length 1 that is the RMSEP of `object`.

Author(s)

Gavin L. Simpson

References

Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C. and ter Braak, C.J.F. (1990). Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London; Series B*, **327**; 263–278.

See Also

`mat`, `bootstrap`, `wa`, `bootstrap.wa`.

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)
```

```
## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## fit the MAT model using the squared chord distance measure
(ik.mat <- mat(ImbrieKipp, SumSST, method = "chord"))

## Leave-one-out RMSEP for the MAT model
RMSEP(ik.mat)

## bootstrap training set
(ik.boot <- bootstrap(ik.mat, n.boot = 100))

## extract the Birks et al (1990) RMSEP
RMSEP(ik.boot)

## Calculate the alternative formulation
RMSEP(ik.boot, type = "standard")
```

roc *ROC curve analysis*

Description

Fits Receiver Operator Characteristic (ROC) curves to training set data. Used to determine the critical value of a dissimilarity coefficient that best descriminate between assemblage-types in palaeoecological data sets, whilst minimising the false positive error rate (FPF).

Usage

```
roc(object, groups, k = 1, ...)

## Default S3 method:
roc(object, groups, k = 1, thin = FALSE,
    max.len = 10000, ...)

## S3 method for class 'mat'
roc(object, groups, k = 1, ...)

## S3 method for class 'analog'
roc(object, groups, k = 1, ...)
```

Arguments

`object`	an R object.
`groups`	a vector of group memberships, one entry per sample in the training set data. Can be a factor, and will be coerced to one if supplied vecvtor is not a factor.
`k`	numeric; the `k` closest analogues to use to calculate ROC curves.
`thin`	logical; should the points on the ROC curve be thinned? See Details, below.
`max.len`	numeric; length of analolgue and non-analogue vectors. Used as limit to thin points on ROC curve to.
`...`	arguments passed to/from other methods.

Details

A ROC curve is generated from the within-group and between-group dissimilarities.

For each level of the grouping vector (`groups`) the dissimilarity between each group member and it's `k` closest analogues within that group are compared with the `k` closest dissimilarities between the non-group member and group member samples.

If one is able to discriminate between members of different group on the basis of assemblage dissimilarity, then the dissimilarities between samples within a group will be small compared to the dissimilarities between group members and non group members.

`thin` is useful for large problems, where the number of analogue and non-analogue distances can conceivably be large and thus overflow the largest number R can work with. This option is also useful to speed up computations for large problems. If `thin == TRUE`, then the larger of the analogue or non-analogue distances is thinned to a maximum length of `max.len`. The smaller set of distances is scaled proportionally. In thinning, we approximate the distribution of distances by taking `max.len` (or a fraction of `max.len` for the smaller set of distances) equally-spaced probability quantiles of the distribution as a new set of distances.

Value

A list with two components; i, `statistics`, a summary of ROC statistics for each level of `groups` and a combined ROC analysis, and ii, `roc`, a list of ROC objects, one per level of `groups`. For the latter, each ROC object is a list, with the following components:

`TPF`	The true positive fraction.
`FPE`	The false positive error
`optimal`	The optimal dissimilarity value, asessed where the slope of the ROC curve is maximal.
`AUC`	The area under the ROC curve.
`se.fit`	Standard error of the AUC estimate.
`n.in`	numeric; the number of samples within the current group.
`n.out`	numeric; the number of samples not in the current group.
`p.value`	The p-value of a Wilcoxon rank sum test on the two sets of dissimilarities. This is also known as a Mann-Whitney test.
`roc.points`	The unique dissimilarities at which the ROC curve was evaluated

`max.roc`	numeric; the position along the ROC curve at which the slope of the ROC curve is maximal. This is the index of this point on the curve.
`prior`	numeric, length 2. Vector of observed prior probabilities of true analogue and true non-analogues in the group.
`analogue`	a list with components `yes` and `no` containing the dissimilarities for true analogue and true non-analogues in the group.

Author(s)

Gavin L. Simpson, based on code from Thomas Lumley to optimise the calculation of the ROC curve.

References

Brown, C.D., and Davis, H.T. (2006) Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems* **80**, 24–38.

Gavin, D.G., Oswald, W.W., Wahl, E.R. and Williams, J.W. (2003) A statistical approach to evaluating distance metrics and analog assignments for pollen records. *Quaternary Research* **60**, 356–367.

Henderson, A.R. (1993) Assessing test accuracy and its clinical consequences: a primer for receiver operating characteristic curve analysis. *Annals of Clinical Biochemistry* **30**, 834–846.

See Also

`mat` for fitting of MAT models. `bootstrap.mat` and `mcarlo` for alternative methods for selecting critical values of analogue-ness for dissimilarity coefficients.

Examples

```
## load the example data
data(swapdiat, swappH, rlgh)

## merge training and test set on columns
dat <- join(swapdiat, rlgh, verbose = TRUE)

## extract the merged data sets and convert to proportions
swapdiat <- dat[[1]] / 100
rlgh <- dat[[2]] / 100

## fit an analogue matching (AM) model using the squared chord distance
## measure - need to keep the training set dissimilarities
swap.ana <- analog(swapdiat, rlgh, method = "SQchord",
                   keep.train = TRUE)

## fit the ROC curve to the SWAP diatom data using the AM results
## Generate a grouping for the SWAP lakes
METHOD <- if (getRversion() < "3.1.0") {"ward"} else {"ward.D"}
clust <- hclust(as.dist(swap.ana$train), method = METHOD)
grps <- cutree(clust, 12)

## fit the ROC curve
```

```
swap.roc <- roc(swap.ana, groups = grps)
swap.roc

## draw the ROC curve
plot(swap.roc, 1)

## draw the four default diagnostic plots
layout(matrix(1:4, ncol = 2))
plot(swap.roc)
layout(1)
```

scores.prcurve — scores *method for principal curve objects of class* "prcurve".

Description

A scores method to extract the position on the curve to which each observation projects (`display = "curve"`) or the coordinates of the curve in the dimensions of the input data (`display = "dimensions"`).

Usage

```
## S3 method for class 'prcurve'
scores(x, display = c("curve", "dimensions"), ...)
```

Arguments

`x`	an object of class "prcurve", usually from a call to prcurve.
`display`	character; which type of scores to extract. `display = "curve"` returns the position along the curve onto which each observation projects; this can be used like a PCA axis score. `display = "dimensions"` returns the coordinates of the curve in the dimensions of the original data.
`...`	Arguments passed to other methods. Not used.

Value

If `display = "curve"` a 1-column matrix is returned with a row for each observation in the input data. If `display = "dimensions"`, a matrix of coordinates for the principal curve is returned. The dimensions of this matrix relate to the dimensions of the input data; if there were n samples (rows) and m variables (columns) then the matrix returned by `scores.prcurve` will have n rows and m columns.

Author(s)

Gavin L. Simpson

See Also

prcurve for fitting principal curves to data.

Examples

```
## Load the Abernethy Forest data set
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit the principal curve using varying complexity of smoothers
## for each species
aber.pc <- prcurve(abernethy2, method = "ca", trace = TRUE,
                   vary = TRUE, penalty = 1.4)

## Extract position on the curve
pos <- scores(aber.pc, display = "curve")
head(pos)

## Extract the coordinates of the curve
coord <- scores(aber.pc, display = "dimensions")
dim(coord)
all.equal(dim(coord), dim(abernethy2))
```

---

screeplot *Screeplots of model results*

Description

Draws screeplots of performance statistics for models of varying complexity.

Usage

```
## S3 method for class 'mat'
screeplot(x, k, restrict = 20,
          display = c("rmsep", "avg.bias",
                      "max.bias", "r.squared"),
          weighted = FALSE,  col = "red", xlab = NULL,
          ylab = NULL, main = NULL, sub = NULL, ...)

## S3 method for class 'bootstrap.mat'
screeplot(x, k, restrict = 20,
          display = c("rmsep","avg.bias","max.bias",
                      "r.squared"),
          legend = TRUE, loc.legend = "topright",
          col = c("red", "blue"),
          xlab = NULL, ylab = NULL,
          main = NULL, sub = NULL,
          ...,
          lty = c("solid","dashed"))
```

Arguments

`x`	object of class `mat` and `bootstrap.mat`.
`k`	number of analogues to use. If missing 'k' is chosen automatically as the 'k' that achieves lowest RMSE.
`restrict`	logical; restrict comparison of k-closest model to k <= `restrict`.
`display`	which aspect of `x` to plot? Partial match.
`weighted`	logical; should the analysis use weighted mean of env data of analogues as fitted/estimated values?
`xlab`, `ylab`	x- and y-axis labels respectively.
`main`, `sub`	main and subtitle for the plot.
`legend`	logical; should a legend be displayed on the figure?
`loc.legend`	character; a keyword for the location of the legend. See `legend` for details of allowed keywords.
`col`	Colours for lines drawn on the screeplot. Method for class `"bootstrap.mat"` takes a vector of two colours.
`lty`	vector detailing the line type to use in drawing the screeplot of the apparent and bootstrap statistics, respectively. Code currently assumes that `length(lty)` is 2.
`...`	arguments passed to other graphics functions.

Details

Screeplots are often used to graphically show the results of cross-validation or other estimate of model performance across a range of model complexity.

Four measures of model performance are currently available: i) root mean square error of prediction (RMSEP); ii) average bias — the mean of the model residuals; iii) maximum bias — the maximum average bias calculated for each of *n* sections of the gradient of the environmental variable; and v) model R^2.

For the maximum bias statistic, the response (environmental) gradient is split into *n* = 10 sections.

For the `bootstrap` method, apparent and bootstrap versions of these statistics are available and plotted.

Note

Currently only models of class `mat` and `bootstrap.mat` are supported.

Author(s)

Gavin Simpson

See Also

`screeplot`

Examples

```
## Imbrie and Kipp example
## load the example data
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training and test set on columns
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
V12.122 <- dat[[2]] / 100

## fit the MAT model using the chord distance measure
(ik.mat <- mat(ImbrieKipp, SumSST, method = "chord"))

screeplot(ik.mat)
```

`smoothers` *Smoother plugin function for use in fitting a principal curve*

Description

Functions to be used as plugins to `prcurve` that fit smooth functions to each variable that, when combined, give the principal curve. The functions act as wrappers to the main fitting functions, which currently include `smooth.spline` and `gam`.

Usage

```
smoothSpline(lambda, x, choose = TRUE, complexity, ...,
             penalty = 1, cv = FALSE, keep.data = FALSE,
             control.spar = list(low = 0))

smoothGAM(lambda, x, choose = TRUE, complexity, bs = "tp", ...,
          family = gaussian(), method = "REML", select = FALSE,
          control = gam.control())
```

Arguments

`lambda`	the current projection function; the position that each sample projects to on the current principal curve. This is the predictor variable or covariate in the smooth function.
`x`	numeric vector; used as the response variable in the smooth function. The principal curve algorithm fits a separate scatterplot smoother (or similar smoother) to each variable in `X` in turn as the response.
`choose`	logical; should the underlying smoother function be allowed to choose the degree of smooth complexity for each variable?

`complexity`	numeric; the complexity of the fitted smooth functions.
`penalty`, `cv`, `keep.data`, `control.spar`	arguments to `smooth.spline`.
`bs`, `family`	arguments to `s`.
`method`, `select`, `control`	arguments to `gam`.
`...`	arguments passed on the the underlying function `smooth.spline` and users should read that function's help page for further details.

Value

An object of class `"prcurveSmoother"` with the following components:

`lambda`	for each observations, its arc-length from the beginning of the curve.
`x`	numeric vector of response values.
`fitted.values`	numeric vector of fitted values for the observations generated from the fitted smooth function.
`complexity`	numeric; the degrees of freedom used for the smooth function. The exact details of what these pertain to are in the help for the respective fitting functions `smooth.spline` and `gam`.
`model`	the object fitted by the wrapped fitting function.

Author(s)

Gavin L. Simpson

See Also

`prcurve` for how these functions are used.

`splitSample` *Select samples from along an environmental gradient*

Description

Select samples from along an environmental gradient by splitting the gradient into discrete chunks and sample within each chunk. This allows a test set to be selected which covers the environmental gradient of the training set, for example.

Usage

```
splitSample(env, chunk = 10, take, nchunk,
            fill = c("head", "tail", "random"),
            maxit = 1000)
```

Arguments

`env`	numeric; vector of samples representing the gradient values.
`chunk`	numeric; number of chunks to split the gradient into.
`take`	numeric; how many samples to take from the gradient. Can not be missing.
`nchunk`	numeric; number of samples per chunk. Must be a vector of length `chunk` and `sum(chunk)` must equal `take`. Can be missing (the default), in which case some simple heuristics are used to determine the number of samples chosen per chunk. See Details.
`fill`	character; the type of filling of chunks to perform. See Details.
`maxit`	numeric; maximum number of iterations in which to try to sample `take` observations. Basically here to stop the loop going on forever.

Details

The gradient is split into `chunk` sections and samples are selected from each chunk to result in a sample of length `take`. If `take` is divisible by `chunk` without remainder then there will an equal number of samples selected from each chunk. Where `chunk` is not a multiple of `take` and `nchunk` is not specified then extra samples have to be allocated to some of the chunks to reach the required number of samples selected.

An additional complication is that some chunks of the gradient may have fewer than `nchunk` samples and therefore more samples need to be selected from the remaining chunks until `take` samples are chosen.

If `nchunk` is supplied, it must be a vector stating exactly how many samples to select from each chunk. If `chunk` is not supplied, then the number of samples per chunk is determined as follows:

1. An intial allocation of `floor(take / chunk)` is assigned to each chunk
2. If any chunks have fewer samples than this initial allocation, these elements of `nchunk` are reset to the number of samples in those chunks
3. Sequentially an extra sample is allocated to each chunk with sufficient available samples until `take` samples are selected.

Argument `fill` controls the order in which the chunks are filled. `fill = "head"` fills from the low to the high end of the gradient, whilst `fill = "tail"` fills in the opposite direction. Chunks are filled in random order if `fill = "random"`. In all cases no chunk is filled by more than one extra sample until all chunks that can supply one extra sample are filled. In the case of `fill = "head"` or `fill = "tail"` this entails moving along the gradient from one end to the other allocating an extra sample to available chunks before starting along the gradient again. For `fill = "random"`, a random order of chunks to fill is determined, if an extra sample is allocated to each chunk in the random order and `take` samples are still not selected, filling begins again using the same random ordering. In other words, the random order of chunks to fill is chosen only once.

Value

A numeric vector of indices of selected samples. This vector has attribute `lengths` which indicates how many samples were actually chosen from each chunk.

Author(s)

Gavin L. Simpson

Examples

```
data(swappH)

## take a test set of 20 samples along the pH gradient
test1 <- splitSample(swappH, chunk = 10, take = 20)
test1
swappH[test1]

## take a larger sample where some chunks don't have many samples
## do random filling
set.seed(3)
test2 <- splitSample(swappH, chunk = 10, take = 70, fill = "random")
test2
swappH[test2]
```

---

sppResponse *Species responses along gradients.*

Description

The fitted responses of species along gradients are estimated or extracted from appropriate objects.

Usage

```
sppResponse(x, ...)

## S3 method for class 'prcurve'
sppResponse(x, n = 100, ...)
```

Arguments

`x`	an R object.
`n`	numeric; the number of locations on the gradient to evaluate the response curve.
`...`	additional arguments passed to other methods.

Details

`sppResponse` estimates species responses along indicated gradients.

There is currently no `"default"` method and the only specified method supplied is for objects fitted by `prcurve`. This method extracts the fitted responses of species along the principal curve and is a useful diagnostic for identifying overly-complex curves.

Value

A list is returned with components `observed` and `fitted.values` containing the observed and fitted values of the species response and gradient respectively. Each is a list with two components, `gradient` and `response`, containing the gradient and response values.

Author(s)

Gavin L. Simpson

See Also

`prcurve` for one function that can be used with `sppResponse`. A `plot` method is available; see `plot.sppResponse` for details.

Examples

```
## Load the Abernethy Forest data set
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit the principal curve using varying complexity of smoothers
## for each species
aber.pc <- prcurve(abernethy2, method = "ca", trace = TRUE,
                   vary = TRUE, penalty = 1.4)

## Extract the fitted species response curves
resp <- sppResponse(aber.pc)

## Look at only the most abundant/frequently occurring taxa
take <- chooseTaxa(abernethy2, max.abun = 25, n.occ = 10, value = FALSE)
layout(matrix(1:12, ncol = 3))   # split device into panels
plot(resp, which = take)
layout(1)    # reset device
```

`stdError` *Standard error of MAT fitted and predicted values*

Description

Computes the (weighted) standard deviation of the environment for the *k*-closest analogues for each sample. This was proposed as one measure of reconstruction uncertainty for MAT models (ter Braak, 1995).

Usage

```
stdError(object, ...)

## S3 method for class 'mat'
stdError(object, k, weighted = FALSE, ...)

## S3 method for class 'predict.mat'
stdError(object, k, weighted = FALSE, ...)
```

Arguments

`object`	Object for which the uncertainty measure is to be computed. Currently methods for `mat` and `predict.mat`.
`k`	numeric; how many analogues to take? If missing, the default, `k` is chosen using `getK`.
`weighted`	logical; use a weighted computation?
`...`	Additional arguments passed to other methods. Currently not used.

Details

Two types of standard error can be produced depending upon whether the mean or weighted mean of y for the k closest analogues is used for the MAT predictions. If `weighted = FALSE` then the usual standard deviation of the response for the k closest analogues is returned, whereas for `weighted = TRUE` a weighted standard deviation is used. The weights are the inverse of the dissimilarity between the target observation and each of the k closest analogues.

Value

A named numeric vector of weighted standard deviations of the environment for the *k* closest analogues used to compute the MAT predicted values.

The returned vector has attributes `"k"` and `"auto"`, indicating the number of analogues used and whether this was determined from `object` or supplied by the user.

Author(s)

Gavin L. Simpson

References

Simpson, G.L. (2012) Analogue methods in palaeolimnology. In Birks, H.J.B, Lotter, A.F. Juggins S., and Smol, J.P. (Eds) *Tracking Environmental Change Using Lake Sediments, Volume 5: Data Handling and Numerical Techniques*. Springer, Dordrecht.

ter Braak, C.J.F. (1995) Non-linear methods for multivariate statistical calibration and their use in palaeoecology: a comparison of inverse (*k*-nearest neighbours, partial least squares, and weighted averaging partial least squares) and classical approaches. *Chemometrics and Intelligent Laboratory Systems* **28**:165–180.

See Also

minDC, mat, predict.mat.

Examples

```
## Imbrie and Kipp Sea Surface Temperature
data(ImbrieKipp)
data(SumSST)
data(V12.122)

## merge training set and core samples
dat <- join(ImbrieKipp, V12.122, verbose = TRUE)

## extract the merged data sets and convert to proportions
ImbrieKipp <- dat[[1]] / 100
ImbrieKippCore <- dat[[2]] / 100

## fit the MAT model using the squared chord distance measure
ik.mat <- mat(ImbrieKipp, SumSST, method = "SQchord")

## standard errors - unweighted
stdError(ik.mat)
## standard errors - weighted version for above
stdError(ik.mat, k = getK(ik.mat), weighted = TRUE)

## standard errors - weighted; note this uses more (7) analogues
## than the above as this model had lowest LOO error
stdError(ik.mat, weighted = TRUE)

## reconstruct for the V12-122 core data
coreV12.mat <- predict(ik.mat, V12.122, k = 3)
## standard errors
stdError(coreV12.mat)
```

`Stratiplot` *Palaeoecological stratigraphic diagrams*

Description

Draws palaeoecological stratigraphic diagrams of one or more variables as a function of depth/age, with the time dimension flowing from the bottom to the top of the y-axis, using the **Lattice** graphics package.

Usage

```
Stratiplot(x, ...)

## Default S3 method:
Stratiplot(x, y, type = "l", ylab = NULL, xlab = "",
           pages = 1, rev = TRUE, ylim, sort = c("none", "wa", "var"),
           svar = NULL, rev.sort = FALSE, strip = FALSE, topPad =6,
           varTypes = "relative", absoluteSize = 0.5,
           zoneNames = NULL, drawLegend = TRUE, na.action = "na.omit",
           labelValues = NULL, labelAt = NULL, labelRot = 60, yticks,
           ...)

## S3 method for class 'formula'
Stratiplot(formula, data, subset,
           na.action = "na.pass", type = "l",
           ylab = NULL, xlab = "", pages = 1, ...)
```

Arguments

`x`	matrix-like object; the variables to be plotted.
`y`	numeric vector of depths/ages corresponding to rows in `x`. Length of `y` must be the same as `nrow(x)` or exactly equal to `nrow(x) / ncol(x)`. See Details.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of plot specification are given under 'Details'.
`type`	character; The type of plotting. Can be a vector. Note that not all Lattice 'type's are supported and some new types are allowed. See `panel.Stratiplot` for further details.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables to plot. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `Stratiplot` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is `"na.omit"` for the default method, which strips `NA`s from the stacked data, whereas the default for the formula method is `"na.pass"` which results in `NA` being passed on to the plotting function. See Details for further information.
`ylab, xlab`	the x- and y-axis labels.
`pages`	numeric; the number of pages to draw the plot over. May be useful for data sets with many species.
`rev`	logical; should the y-axis limits be reversed
`ylim`	user supplied limits for the y-axis (time/depth). If not supplied, suitable limits will be determined from the data. As such, in general use `ylim` need not be supplied. If you choose to supply your own `ylim` values, note the default for argument `rev`; the default will reverse the values you supply to `ylim`.
`sort`	character; how should the variables (columns) of `x` be sorted on the plot. `"wa"` sorts by weighted averages of variable `svar` if not `NULL` or of `y` otherwise. The default when `"wa"` is specified is to order by wiehgted average of the depth/time axis – `y`. If `"var"`, then ordering is done as per the **order** of `svar`.

`svar`	vector; optional variable to sort columns of `x` by.
`rev.sort`	logical; should the sorting order be reversed.
`strip`	logical; Should panels have strips containing variable labels drawn on them? Default is `FALSE`, which labels each panel with a label resulting in a more familiar plot style.
`topPad`	numeric; additional padding for the top axis to accomodate long variable names. This is a temporary fudge until the actual space required can be automagically calculated from the variable names themselves. The currently gets most of the way there, but `topPad` is used to add some extra space if required.
`varTypes`	a character vector of length 1 or equal in length to the number of variables plotted. If length 1, the vector is expanded to the required length. Two values are allowed; i. `"relative"`, and ii. `"absolute"`. `"relative"` treats the indicated variable as a percentage type variable and the panel width is scaled relative to the percentage type variables plotted. `"absolute"` treats the indicated variable as an absolute variable whose panel width should be independent of the other panels. Use `"absolute"` for variables that are not species compositional data or other percentage type data.
`absoluteSize`	numeric, length 1. This controls the width of panels for variables marked as `"absolute"`, and is the proportion of the largest non-`"absolute"` panel.
`zoneNames`	character vector of labels, one per zone, with which to label the zone legend, if drawn (see argument `drawLegend`). See Details.
`drawLegend`	logical; should a legend for the zones
`labelValues`	a vector of labels for the variables plotted. Should be equal in length to the number or variables in the resulting plot. The main use for `labelValues` is to provide non-standard labels for the variables, including a vector of expressions. See Examples for an illustration of this.
`labelAt`, `labelRot`	these control the placement and rotation, respectively, of the variable labels. `labelAt` is the coordinate at which the label is drawn; currently only one value is used so you can't place labels in different locations depending on which panel is drawn. This will be fixed in a future version. The default location for the label is the panel mid-point. `labelAt` controls the rotation of the label; it is a numeric value in degree.
`yticks`	This is passed to the `scales` argument of `xyplot` as component `at`. This should be a numeric vector of tick locations for the y (depth/age) axis. Setting this to `NULL` or `FALSE` suppresses ticks on the y axis. The default uses `TRUE`, which uses the default choices for `at` used by `xyplot`.
`...`	additional arguments passed to `panel.Stratiplot` and the underlying `xyplot` function.

Details

The function now includes preliminary code to handle both relative (proportional or percentage data) and absolute data types, and mixtures thereof. Mixtures can be specified by supplying a vector of types to `varTypes`, in the same order as the variables are drawn on the plot.

Plots can be specified symbolically using a formula. A typical model has the form `Y ~ variables`, where `Y` is either the core depths or sample ages/dates (to be plotted on the y-axis) and `variables` is a series of terms which specifies the variables to plot against `Y`. Terms should be specified with the form `var1 + var2 + var3` to plot only those variables. Other, standard, notation for formulae apply, such as model formulae used in `lm`.

For the formula method the default for argument `na.action` is `"na.pass"`, which results in any `NA` values being passed on to the plotting code. This allows for plotting of proxies that been measured on different levels of the stratigraphy. Should you wish to have `NA` removed from the data before plotting, use `na.action = "na.omit"`, though do note this will remove all rows where any column/variable takes the value `NA`. The default `Stratiplot` method, which is used by the formula method for plotting, will strip any `NA` values from the data provided to it. This allows the function to correctly handle the situation where proxies are measured on different levels of the core *and* you are displaying the data using lines of polygons. If the `NA` were not dropped by `Stratiplot.default`, any `NA` values would show up as breaks in the line or polygon drawn for each panel.

In summary, the two methods have different defaults for `na.action` to allow them to handle proxies measured on different levels of the same core. This does mean that you can not use the formula interface **and** strip NA's at the `Stratiplot.default` level. If you need that level of control use `Stratiplot.default` directly by not providing a formula as argument `x` and by supplying data for the y-axis via argument `y`. See Examples for an illustration of these features.

Note that `formula` is **not** passed on to `xyplot`. Instead, the formula is parsed and evaluated within `Stratiplot` and an appropriate data structure formed to facilitate plotting via `xyplot`. As such, the special features of **Lattice** formulae cannot be used.

If zones are drawn on the stratigraphic plot, the `zoneNames` argument can be used to supply a set of names with which to label the zones using a legend. This legend is drawn on the right-hand side of the the straigraphic diagram if `drawLegend = TRUE` is supplied. The `zoneNames` must be supplied in stratigraphic order, as that is the order in which they are drawn in the legend. Whether this ordering is reversed or not will depend on the value of argument `rev`. It is up to the user to provide the labels in the correct order. Zones are specified by the zone boundaries (excluding the core sequence top and bottom), and as a result 1 more label is required than the number of zone boundaries supplied. If no `zoneNames` is not supplied, but a legend is requested, suitable names will be produced. If you do not wish to have any labelling at all, then set `zoneNames = ""` as this will get recycled to the correct length. See the Example section for an illustration of how this drawing zones works.

Value

A side effect of calling `Stratiplot` is that a plot is drawn on the currently active device. A Lattice plot object of class `"trellis"` is returned invisibly. This is a change from pre 0.17-0 version of the package.

Note

The function currently doesn't know about ages/dates and will interpret these as 'depths' instead. This will be fixed in a future version.

Author(s)

Gavin L. Simpson.

See Also

xyplot, panel.Stratiplot, panel.Loess.

Examples

```
data(V12.122)
Depths <- as.numeric(rownames(V12.122))

(plt <- Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
                   data = V12.122,  type = c("h","l","g","smooth")))

## Order taxa by WA in depth --- ephasises change over time
(plt <- Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
                   data = V12.122, type = c("h"), sort = "wa"))

## Using the default interface
spp.want <- c("O.univ","G.ruber","G.tenel","G.pacR")
(plt <- Stratiplot(V12.122[, spp.want], y = Depths,
                   type = c("poly", "g")))

## Adding zones to a Stratigraphic plot
## Default labelling and draw zone legend
## Here we choose 4 arbitrary Depths as the zone boundaries
set.seed(123)
Zones <-sample(Depths, 4)
Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
           data = V12.122, type = c("poly","g"),
           zones = Zones)

## As before, but supplying your own zone labels
zone.labs <- c("A","B","C","D","E")
Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
           data = V12.122, type = c("poly","g"),
           zones = Zones, zoneNames = zone.labs)

## Suppress the drawing of the zone legend
Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
           data = V12.122, type = c("poly","g"),
           zones = Zones, drawLegend = FALSE)

## Add zones and draw a legend, but do not label the zones
Stratiplot(Depths ~ O.univ + G.ruber + G.tenel + G.pacR,
           data = V12.122, type = c("poly","g"),
           zones = Zones, zoneNames = "")

## Show illustration of NA handling
set.seed(42)
dat <- data.frame(Depth = 1:20, LOI = runif(20), TC = NA)
dat <- within(dat, TC[sample(20, 10)] <- runif(10))
## default is 'na.action = "na.pass"'
Stratiplot(Depth ~ LOI + TC, data = dat, type = c("l","p"))
## to remove rows with NA, use 'na.action = "na.omit"'
```

```
Stratiplot(Depth ~ LOI + TC, data = dat, type = c("l","p"),
           na.action = "na.omit")

## Example of two proxies measured on different levels of core
## (Here measurements on alternate levels)
set.seed(5)
dat2a <- data.frame(Depth = seq(1, by = 2, length = 20), LOI = runif(20))
dat2b <- data.frame(Depth = seq(0, by = 2, length = 20), TC = runif(20))
dat2 <- join(dat2a, dat2b, na.replace = FALSE, split = FALSE)
dat2 <- dat2[order(dat2$Depth), ]
head(dat2)

## Default is to allow NA through formula, but drop them when plotting
Stratiplot(Depth ~ LOI + TC, data = dat2, type = c("l","p"))

## compare with this if we didn't suppress NA in default Stratiplot
## method (can't use formula interface for this yet
Stratiplot(dat2[,-1], dat2[,1], type = c("l","p"),
           na.action = "na.pass")
## Notice no lines are draw as there a no "sections" ithout missing
## levels. If you want/desire this behaviour then you can't use formula
## interface yet as there is no way to specify the na.action separately

## Works with matrices
M <- as.matrix(V12.122)
Stratiplot(M, Depths, type = c("h"))

## Custom variable labels using expressions
df <- data.frame(Age = 1:10, Var1 = rnorm(10), Var2 = rnorm(10),
                 Var3 = rnorm(10))
## Use a vector of expressions to label variables on plot
## See ?plotmath for syntax of expressions
exprs <- expression(delta^{15}*N,       # label for Var1
                    delta^{18}*O,       # label for Var2
                    delta^{13}*C)       # label for Var3
Stratiplot(Age ~ ., data = df, labelValues = exprs, varTypes = "absolute")
```

summary.analog *Summarise analogue matching results*

Description

summary method for class "analog".

Usage

```
## S3 method for class 'analog'
summary(object, display = c("dist", "names", "quantiles"),
        k = 10, probs = c(0.01, 0.02, 0.05, 0.1, 0.2), ...)
```

Arguments

`object`	an object of class `"analog"`, usually as a result of a call to `analog`.
`display`	character; one or more of the listed choices. Determines which aspects of the `analog` results are summarised.
`k`	number of analogues to use. If missing, `k` is chosen automatically as the `k` that achieves lowest RMSE.
`probs`	numeric; giving the probabilities of the distribution to return quantiles for. See `quantile`.
`...`	arguments passed to or from other methods.

Value

A list with one or more of the components listed below. Attributes `"method"`, `"train"`, `"call"` and `"k"` contain the dissimilarity coefficient used, whether the training set dissimilarities were saved, the matched function call and the number of close analogues to return respectively.

`dists`	a matrix of dissimilarities between training set samples and fossil samples. The number of rows is given by argument `k`. There is a column for each fossil sample.
`names`	a matrix of names of samples from the training set that are analogues for each fossil sample. The number of rows is given by argument `k`. There is a column for each fossil sample.
`quantiles`	numeric; the quantiles of the distribution of the pairwise dissimilarities for the training set for probabilities `prob`.

Author(s)

Gavin L. Simpson

See Also

`analog`.

Examples

```
## Not run:
## continue the RLGH example from ?join
example(join)

## analog matching between SWAP and RLGH core
swap.analog <- analog(swapdiat, rlgh, method = "chord")
swap.analog
summary(swap.analog)

## End(Not run)
```

summary.bootstrap.mat *Summarise bootstrap resampling for MAT models*

Description

summary method for class "bootstrap.mat".

Usage

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

Arguments

object	an object of class "bootstrap.mat", usually the result of a call to bootstrap.mat.
...	arguments passed to or from other methods.

Value

A data frame with the following components:

observed — vector of observed environmental values.

model — a list containing the apparent or non-bootstrapped estimates for the training set. With the following components:

- **estimated:** estimated values for the response
- **residuals:** model residuals
- **r.squared:** Apparent R^2 between observed and estimated values of y
- **avg.bias:** Average bias of the model residuals
- **max.bias:** Maximum bias of the model residuals
- **rmse:** Apparent error (RMSE) for the model
- **k:** numeric; indicating the size of model used in estimates and predictions

bootstrap — a list containing the bootstrap estimates for the training set. With the following components:

- **estimated:** Bootstrap estimates for the response
- **residuals:** Bootstrap residuals for the response
- **r.squared:** Bootstrap derived R^2 between observed and estimated values of the response
- **avg.bias:** Average bias of the bootstrap derived model residuals
- **max.bias:** Maximum bias of the bootstrap derived model residuals
- **rmsep:** Bootstrap derived RMSEP for the model
- **s1:** Bootstrap derived S1 error component for the model
- **s2:** Bootstrap derived S2 error component for the model
- **k:** numeric; indicating the size of model used in estimates and predictions

`sample.errors`	a list containing the bootstrap-derived sample specific errors for the training set. With the following components: `rmsep`**:** Bootstrap derived RMSEP for the training set samples `s1`**:** Bootstrap derived S1 error component for training set samples `s2`**:** Bootstrap derived S2 error component for training set samples
`weighted`	logical; whether the weighted mean was used instead of the mean of the environment for *k*-closest analogues
`auto`	logical; whether `k` was choosen automatically or user-selected
`n.boot`	numeric; the number of bootstrap samples taken
`call`	the matched call
`call`	model type
`predictions`	a list containing the apparent and bootstrap-derived estimates for the new data, with the following components: `observed`**:** the observed values for the new samples — only if `newenv` is provided `model`**:** a list containing the apparent or non-bootstrapped estimates for the new samples. A list with the same components as `apparent`, above `bootstrap`**:** a list containing the bootstrap estimates for the new samples, with some or all of the same components as `bootstrap`, above `sample.errors`**:** a list containing the bootstrap-derived sample specific errors for the new samples, with some or all of the same components as `sample.errors`, above

Author(s)

Gavin L. Simpson

See Also

`bootstrap.mat`, `mat`, `summary`.

Examples

```
## Not run:
## continue the RLGH example from ?join
example(join)

## fit the MAT model using the squared chord distance measure
swap.mat <- mat(swapdiat, swappH, method = "SQchord")

## bootstrap training set
swap.boot <- bootstrap(swap.mat, k = 10, n.boot = 100)
swap.boot
summary(swap.boot)

## End(Not run)
```

`summary.cma` *Summarise the extraction of close modern analogues*

Description

`summary` method for class `"cma"`.

Usage

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

Arguments

`object`	an object of class `"cma"`, usually the result of a call to `cma`.
`...`	arguments passed to or from other methods.

Value

An object of class `"summary.cma"` with the components of an object of class `cma`, plus:

`distances`	a matrix of distances/dissimilarities. Individual columns contain the ordered close modern analogues for individual fossil samples. Rows of this matrix refer to the k^{th} closest analogue for each fossil sample. See notes below.
`samples`	a matrix of sample names from the reference set that are close modern analogues for a fossil sample. Individual columns contain the ordered close modern analogues for individual fossil samples. Rows of this matrix refer to the k^{th} closest analogue for each fossil sample. See notes below.

Note

Currently, only objects of class `analog` are supported. The number of rows in the returned matrices is equal to the maximum number of close modern analogues identified for an individual fossil sample. If no close modern analogues exist for an individual fossil sample are identified, then the relevant column in `"distances"` will contain all missing values and in `"samples"` the string `"none"`. Rows of individual columns will be padded with missing values if the number of close modern analogues for that sample is less than the maximum number of close modern analogues identified for a single sample.

Author(s)

Gavin L. Simpson

See Also

`cma`

Examples

```
## Not run:
## continue the RLGH example from ?join
example(join)

## analog matching between SWAP and RLGH core
swap.analog <- analog(swapdiat, rlgh, method = "chord")
swap.analog
summary(swap.analog)

## close modern analogues
swap.cma <- cma(swap.analog, cutoff = 0.6)
swap.cma
summary(swap.cma)

## End(Not run)
```

`summary.mat` *Summarise Modern Analogue Technique models*

Description

`summary` method for class `"mat"`.

Usage

```
## S3 method for class 'mat'
summary(object, k = 10,
        digits = max(2, getOption("digits") - 4), ...)
```

Arguments

`object`	an object of class `"cma"`, usually the result of a call to `cma`.
`k`	numeric; maximum modern analogues to use to summarise model fits.
`digits`	numeric; the number of significant digits with which to format results.
`...`	arguments passed to or from other methods.

Value

A list with the components below. The number of analogues used, *k* is returned as attribute `"k"`.

`summ`	a data.frame containing the model fits for training set samples. See notes below.
`tbl`	matrix of summary statistics for an un-weighted model.
`tbl.W`	matrix of summary statistics for a weighted model.
`call`	the matched function call
`quantiles`	the quantiles of the distribution of pairwise dissimilarities for the training set, for `"probs = c(0.01, 0.02, 0.05, 0.1, 0.2)"`.

Note

The returned component "summ" contains the following:

Obs: the observed responses for the training set samples.

Est: the fitted values of the response for training set samples based on the average of *k*-closest analogues.

Resi: the residuals of the fitted model based on the average of *k*-closest analogues.

W.Est: the fitted values of the response for training set samples based on the weighted average of *k*-closest analogues.

W.Resi: the residuals of the fitted model based on the weighted average of *k*-closest analogues.

minDC: dissimilarity of closest analogue in training set for each training set sample.

minResi: smallest residual for an un-weighted model of size "k".

k: size of model leading to minimal residual, "minResi".

minW.Resi: smallest residual for a weighted model of size "k.W".

k.W: size of model leading to minimal residual, "minW.Resi".

Author(s)

Gavin L. Simpson

See Also

mat, summary.

Examples

```
## Not run:
## continue the RLGH example from ?join
example(join)

## fit the MAT model using the squared chord distance measure
swap.mat <- mat(swapdiat, swappH, method = "SQchord")
swap.mat

## model summary
summary(swap.mat)

## model summary - evaluating models using k = 1, ..., 20
## analogues instead of the default, 10.
summary(swap.mat, k = 20)

## End(Not run)
```

summary.predict.mat *Summarise MAT model predictions*

Description

summary method for objects of class "predict.mat".

Usage

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

Arguments

object	an object of class "predict.mat", usually the result of a call to predict.mat.
...	arguments passed to or from other methods.

Value

An object of class "summary.predict.mat", see predict.mat for more details.

Author(s)

Gavin L. Simpson

See Also

predict.mat, mat, bootstrap.mat and summary.

Examples

```
## Not run:
## continue the RLGH example from ?join
example(join)

## fit the MAT model using the squared chord distance measure
swap.mat <- mat(swapdiat, swappH, method = "SQchord")

## predict for RLGH data
swap.pred <- predict(swap.mat, rlgh, bootstrap = FALSE)
summary(swap.pred)

## End(Not run)
```

swapdiat — *SWAP sub-fossil diatom and pH training set*

Description

The Surface Waters Acidifcation Project (SWAP) Palaeolimnology Programme diatom surface sample training set.

Usage

```
data(swapdiat)
```

Format

A data frame of observations on 277 diatom taxa for the 167-lake SWAP diatom-pH training set.

Details

This data set contains the original 167-lake SWAP diatom-pH training set.

The variable names (`colnames`) are DIATCODE codes for individual taxa.

References

Stevenson, A.C., Juggins, S., Birks, H.J.B., Anderson, D.S., Anderson, N.J., Battarbee, R.W., Berge, F., Davis, R.B., Flower, R.J., Haworth, E.Y., Jones, V.J., Kingston, J.C., Kreiser, A.M., Line, J.M., Munro, M.A.R., and Renberg, I. (1995). *The Surface Waters Acidification Project Palaeolimnology programme: modern diatom/lake-water chemistry data-set*. ENSIS Publishing, London.

See Also

`swappH`

Examples

```
data(swapdiat)
```

swappH *SWAP sub-fossil diatom and pH training set*

Description

The Surface Waters Acidifcation Project (SWAP) Palaeolimnology Programme diatom-pH surface sample training set.

Usage

```
data(swappH)
```

Format

Numeric vector containing the pH of the 167 lakes from the SWAP diatom-pH training set. The values are the average of 4 quarterly samples.

References

Stevenson, A.C., Juggins, S., Birks, H.J.B., Anderson, D.S., Anderson, N.J., Battarbee, R.W., Berge, F., Davis, R.B., Flower, R.J., Haworth, E.Y., Jones, V.J., Kingston, J.C., Kreiser, A.M., Line, J.M., Munro, M.A.R., and Renberg, I. (1995). *The Surface Waters Acidification Project Palaeolimnology programme: modern diatom/lake-water chemistry data-set*. ENSIS Publishing, London.

See Also

`swapdiat`

Examples

```
data(swappH)
str(swappH)
```

timetrack *Timetracks of change in species composition*

Description

Project passive (e.g. sediment core) samples into an ordination of a set of training samples.

Usage

```
timetrack(X, passive, env, method = c("cca", "rda"),
          transform = "none", formula, scaling = 3,
          rank = "full", join = "left", correlation = FALSE,
          hill = FALSE, ...)

## S3 method for class 'timetrack'
fitted(object, which = c("passive", "ordination"),
       model = NULL, choices = 1:2, ...)

## S3 method for class 'timetrack'
predict(object, newdata, ...)

## S3 method for class 'timetrack'
scores(x, which = c("ordination", "passive"),
       scaling = x$scaling, choices = 1:2, display = "sites", ...)

## S3 method for class 'timetrack'
plot(x, choices = 1:2, display = c("wa", "lc"),
     order, type = c("p", "n"), ptype = c("l", "p", "o", "b", "n"),
     pch = c(1,2), col = c("black","red"), lty = "solid", lwd = 1,
     xlim = NULL, ylim = NULL, ...)

## S3 method for class 'timetrack'
points(x, choices = 1:2, which = c("passive", "ordination"),
       display = c("wa","lc"), order, ...)
```

Arguments

`X`	matrix-like object containing the training set or reference samples.
`passive`	matrix-like object containing the samples to be projected into the ordination of `X`. Usually a set of sediment core samples.
`env`	optional data frame of environmental or constraining variables. If provided, a constrained ordination of `X` is performed. If `formula` is supplied variables named in `formula` are looked up with `env`.
`method`	character, resolving to an ordination function available in **vegan**. Currently only `"cca"`, the default, and `"rda"` are supported.
`transform`	character; the name of the transformation to apply to both `X` and `passive`. The transformations are performed using `tran` and valid options are given by that function's `method` argument.
`formula`	a one-sided model formula; if provided, it defines the right hand side of the model formula for the ordination function and is supplied as argument `formula` to the ordination function. E.g.~`formula` = ~ `var1` + `var2`. If supplied then `env` must also be supplied
`scaling`	numeric or character; the ordination scaling to apply. Useful options are likely to be `1` or `3` where the focus is on the samples. For `character`, see options in

	scores.cca: character version of the useful scalings are "sites" and "symmetric". See arguments correlation and hill.
correlation, hill	logical; additional arguments passed to predict.cca and predict.rda. See scores.cca for details.
rank	character; see argument of same name in function predict.cca or predict.rda.
join	character; the tpe of join to perform. See join for details of possible choices, but the default, "left" is most generally applicable.
object, x	an object of class "timetrack".
which	character; which fitted values should be returned?
model	character; which ordination component should be used for the fitted values; the constrained or unconstrained part? See fitted.cca for details, but essentially, one of "CCA" for the constrained part and "CA" for the unconstrained part. If NULL, the default, "CA" is used unless the underlying ordination was constrained, in which case "CCA" is used.
choices	numeric; the length-2 vector of ordination axes to plot.
newdata	a data frame of new observations for which locations in the plot (or a timetrack) are required. This need not have exactly the same set of species as the fitted ordination as internally only those species in newdata that were included in the data used for the ordination will be retained. In addition, if a transformation was applied to the species data used to fit the ordination, the same transformation will be automatically applied to newdata using tran.
display	character; which type of sites scores to display? See scores.cca for details.
order	numeric; vector of indices to use to reorder the passive samples. Useful to get passive samples into temporal order for plotting with a line.
type	character; the type of plotting required for the training set samples. Options are "p" for points or "n" to not draw training set samples.
ptype	character; controls how the time track should be drawn. Default is draw the passive samples connected by a line in the order in which they appear in the data. With ptype = "p" no line is drawn. The other types have their usual meaning from plot.default.
pch	The length-2 vector of plotting characters. The first element is used for the ordination samples, the second for the passive samples.
col	The length-2 vector of plotting colours. The first element is used for the ordination samples, the second for the passive samples.
lty, lwd	graphical parameters for the plotted time track for ptype != "p".
xlim, ylim	user specified axis limits for the plot.
...	arguments passed to other methods. timetrack passes arguments on to tran and the ordination function given in method. fitted passes arguments on to other fitted methods as appropriate. plot passes arguments on to the underlying plotting functions. predict passes arguments on to tran for use in applyign the transformation.

Details

The timetrack is a way to visualise changes in species composition from sediment core samples within an underlying reference ordination or, usually, training set samples. This technique has been most often applied in situations where the underlying ordination is a constrained ordination and thence the timetrack of sediment core samples within the ordination reflects both the change in species composition and the indicative changes in the constraining variables.

The sediment core samples are projected passively into the underlying ordination. By projected passively, the locations of the core samples are predicted on the basis of the ordination species scores. A common set of species (columns) is required to passively place the sediment samples into the ordination. To achieve this, the left outer join of the species compositions of the training set and passive set is determined; the left outer join results in the passive data matrix having the same set of species (variables; columns) as the training set. Any training set species not in the passive set are added to the passive set with abundance 0. Any passive species not in the training set are removed from the passive set.

Value

The `plot` method results in a plot on the currently active device, whilst the `fitted` and `scores` methods return the matrix of fitted locations on the set of ordination axes.

`timetrack` returns an object of class `"timetrack"`, a list with the following components:

`ordination`	the ordination object, the result of the call to the function of the name `method`.
`fitted.values`	the matrix of fitted locations for the passive samples on the ordination axes.
`method`	the ordination function used.
`formula`	if supplied, the model formula used to define the ordination model.
`scaling`	the ordination scaling applied.
`rank`	The rank or the number of axes used in the approximation. The default is to use all axes (full rank) of the `"model"`.
`model`	Show constrained (`"CCA"`) or unconstrained (`"CA"`) results.
`labels`	a list of names for the `X`, `passive`, and `env` arguments.
`call`	The matched function call.
`X`	The training data.
`transform`	The transformation applied, if any.

Author(s)

Gavin L. Simpson

See Also

`cca` and `rda` for the underlying ordination functions.

Examples

```
## load the RLGH and SWAP data sets
data(rlgh, swapdiat)

## Fit the timetrack ordination
mod <- timetrack(swapdiat, rlgh, transform = "hellinger",
                 method = "rda")
mod

## Plot the timetrack
plot(mod, ptype = "b", col = c("forestgreen", "orange"), lwd = 2)

## Other options (reorder the time track)
ord <- rev(seq_len(nrow(rlgh)))
plot(mod, choices = 2:3, order = ord, ptype = "b",
     col = c("forestgreen", "orange"), lwd = 2)

## illustrating use of the formula
data(swappH)
mod2 <- timetrack(swapdiat, rlgh, env = data.frame(pH = swappH),
                  transform = "hellinger", method = "rda",
                  formula = ~ pH)
mod2
plot(mod2)

## scores and fitted methods
## IGNORE_RDIFF_BEGIN
head(fitted(mod, type = "passive"))
head(scores(mod, type = "passive"))
## IGNORE_RDIFF_END

## predict locations in timetrack for new observations
take <- rlgh[1:50, ]
take <- take[ , colSums(take) > 0]
mod3 <- predict(mod, newdata = take)
class(mod3) ## returns a timetrack object
take <- rlgh[-(1:50), ]
take <- take[ , colSums(take) > 0]
mod4 <- predict(mod, newdata = take)

## build a plot up from base parts
plot(mod, type = "n", ptype = "n")
points(mod, which = "ordination", col = "grey", pch = 19, cex = 0.7)
points(mod3, which = "passive", col = "red")
points(mod4, which = "passive", col = "blue")

## Fit the timetrack ordination - passing scaling args
mod <- timetrack(swapdiat, rlgh, transform = "hellinger",
                 method = "rda", scaling = "sites",
                 correlation = TRUE)
mod
plot(mod)
```

tortula *Morphological data for ten taxa of the genus Tortula*

Description

These data are observations on a series of seven morphological variables for individuals in of the *Tortula* sect. *Rurales* De Not. (*Pottiaceae, Musci*.

Usage

```
data(tortula)
```

Format

`tortula` is a data frame of seven morphological measurements on 14 individuals from the genus *Tortula*.

`Taxon`	factor; the species of *Tortula*
`Hydroid`	logical; presence of hydroid cells
`LeafOutline`	ordered; shape of the leaf outline
`Denticulation`	ordered; degree of denticulation
`ApexShape`	ordered; shape of the leaf apex
`Length`	numeric, leaf length
`Diameter`	numeric; leaf diameter
`Papillae`	numeric; number of papillae per cell

The last three variables are the average of ten replicate samples from the same herbarium capsule.

Source

The data were presented in Podani (1999).

References

Podani, J. (1999) Extending Gower's coefficient of similarity to ordinal characters. *Taxon* **48**, 331-340.

Examples

```
data(tortula)
head(tortula)
str(tortula)
```

tran — *Common data transformations and standardizations*

Description

Provides common data transformations and standardizations useful for palaeoecological data. The function acts as a wrapper to function decostand in package vegan for several of the available options.

The formula method allows a convenient method for selecting or excluding subsets of variables before applying the chosen transformation.

Usage

```
## Default S3 method:
tran(x, method, a = 1, b = 0, p = 2, base = exp(1),
     na.rm = FALSE, na.value = 0, ...)

## S3 method for class 'formula'
tran(formula, data = NULL, subset = NULL,
     na.action = na.pass, ...)
```

Arguments

x	A matrix-like object.
method	transformation or standardization method to apply. See Details for available options.
a	Constant to multiply x by. method = "log" only. Can be a vector, in which case the vector of values to multiply each column of x by.
b	Constant to add to x before taking logs. method = "log" only. Can be a vector, in which case the vector of values to add to each column of x.
p	The power to use in the power transformation.
base	the base with respect to which logarithms are computed. See log for further details. The default is to compute natural logarithms.
na.rm	Should missing values be removed before some computations?
na.value	The value with which to replace missing values (NA).
...	Arguments passed to decostand, or other tran methods.
formula	A model formula describing the variables to be transformed. The formula should have only a right hand side, e.g.~~ foo + bar.
data, subset, na.action	See model.frame for details on these arguments. data will generally be the object or environment within which the variables in the forumla are searched for.

Details

The function offers following transformation and standardization methods for community data:

- `sqrt`: take the square roots of the observed values.
- `cubert`: take the cube root of the observed values.
- `rootroot`: take the fourth root of the observed values. This is also known as the root root transformation (Field et al 1982).
- `log`: take the logarithms of the observed values. The tansformation applied can be modified by constants `a` and `b` and the `base` of the logarithms. The transformation applied is $x^* = \log_{\mathrm{base}}(ax + b)$
- `log1p`: computes $log(1 + x)$ accurately also for $|x| << 1$ via `log1p`. Note the arguments `a` and `b` have no effect in this method.
- `expm1`: computes $exp(x) - 1)$ accurately for $|x| << 1$ via `expm1`.
- `reciprocal`: returns the multiplicative inverse or reciprocal, $1/x$, of the observed values.
- `freq`: divide by column (variable, species) maximum and multiply by the number of non-zero items, so that the average of non-zero entries is 1 (Oksanen 1983).
- `center`: centre all variables to zero mean.
- `range`: standardize values into range $0 \dots 1$. If all values are constant, they will be transformed to 0.
- `percent`: convert observed count values to percentages.
- `proportion`: convert observed count values to proportions.
- `standardize`: scale `x` to zero mean and unit variance.
- `pa`: scale `x` to presence/absence scale (0/1).
- `missing`: replace missing values with `na.value`.
- `chi.square`: divide by row sums and square root of column sums, and adjust for square root of matrix total (Legendre & Gallagher 2001). When used with the Euclidean distance, the distances should be similar to the the Chi-square distance used in correspondence analysis. However, the results from `cmdscale` would still differ, since CA is a weighted ordination method.
- `hellinger`: square root of observed values that have first been divided by row (site) sums (Legendre & Gallagher 2001).
- `wisconsin`: applies the Wisconsin double standardization, where columns (species, variables) are first standardized by maxima and then sites (rows) by site totals.
- `pcent2prop`: convert percentages to proportions.
- `prop2pcent`: convert proportions to percentages.
- `logRatio`: applies a log ransformation (see `log` above) to the data, then centres the data by rows (by subtraction of the mean for row *i* from the observations in row *i*). Using this transformation subsequent to PCA results in Aitchison's Log Ratio Analysis (LRA), a means of dealing with closed compositional data such as common in palaeoecology (Aitchison, 1983).
- `power`: applies a power tranformation.
- `rowCentre`, `rowCenter`: Centres `x` by rows through the subtraction of the corresponding row mean from the observations in the row.

- `colCentre` `colCenter`: Centres `x` by columns through the subtraction of the corresponding column mean from the observations in the row.
- `none` `none`: no transformation is applied.

Value

Returns the suitably transformed or standardized `x`. If `x` is a data frame, the returned value is likewise a data frame. The returned object also has an attribute `"tran"` giving the name of applied transformation or standardization `"method"`.

Author(s)

Gavin L. Simpson. Much of the functionality of `tran` is provided by `decostand`, written by Jari Oksanen.

References

Aitchison, J. (1983) Principal components analysis of compositional data. *Biometrika* **70**(1); 57–65.

Field, J.G., Clarke, K.R., & Warwick, R.M. (1982) A practical strategy for analysing multispecies distributions patterns. *Marine Ecology Progress Series* **8**; 37–52.

Legendre, P. & Gallagher, E.D. (2001) Ecologically meaningful transformations for ordination of species data. *Oecologia* **129**; 271-280.

Oksanen, J. (1983) Ordination of boreal heath-like vegetation with principal component analysis, correspondence analysis and multidimensional scaling. *Vegetatio* **52**; 181-189.

See Also

`decostand`

Examples

```
data(swapdiat)
## convert percentages to proportions
sptrans <- tran(swapdiat, "pcent2prop")

## apply Hellinger transformation
spHell <- tran(swapdiat, "hellinger")

## Dummy data to illustrate formula method
d <- data.frame(A = runif(10), B = runif(10), C = runif(10))
## simulate some missings
d[sample(10,3), 1] <- NA
## apply tran using formula
tran(~ . - B, data = d, na.action = na.pass,
     method = "missing", na.value = 0)
```

varExpl *Variance explained by ordination axes*

Description

Extracts information about the variance explained by ordination axes and expresses it in a variety of ways.

Usage

```
varExpl(object, ...)

## S3 method for class 'cca'
varExpl(object, axes = 1L, cumulative = FALSE,
        pcent = FALSE, ...)

## S3 method for class 'prcurve'
varExpl(object, pcent = FALSE, ...)
```

Arguments

`object`	an R object of an appropriate type. Currently only for objects that inherit from classes `"cca"` or `"prcurve"`.
`axes`	numeric vector indicating which axes to compute variance explained for.
`cumulative`	logical; should the variance be explained as a cumulative sum over the axes?
`pcent`	logical; should the variance explained be expressed as a percentage of the total variance.
`...`	additional arguments passed to other methods. Currently not used.

Value

A numeric vector variance explained by each axis.

Author(s)

Gavin L. Simpson

See Also

See `cca` and `prcurve` for functions that produce objects that `varExpl()` can work with.

Examples

```
data(abernethy)

## Remove the Depth and Age variables
abernethy2 <- abernethy[, -(37:38)]

## Fit PCA
aber.pca <- rda(abernethy2)

## Distance along the first PCA axis
varExpl(aber.pca)
```

---

wa *Weighted averaging transfer functions*

Description

Implements the weighted averaging transfer function methodology. Tolerance down-weighting and inverse and classicial deshrinking are supported.

Usage

```
wa(x, ...)

## Default S3 method:
wa(x, env,
   deshrink = c("inverse", "classical", "expanded", "none", "monotonic"),
   tol.dw = FALSE, useN2 = TRUE,
   na.tol = c("min","mean","max"),
   small.tol = c("min","mean","fraction","absolute"),
   min.tol = NULL, f = 0.1, ...)

## S3 method for class 'formula'
wa(formula, data, subset, na.action,
   deshrink = c("inverse", "classical", "expanded", "none", "monotonic"),
   tol.dw = FALSE, useN2 = TRUE, na.tol = c("min","mean","max"),
   small.tol = c("min","mean","fraction","absolute"), min.tol = NULL,
   f = 0.1,..., model = FALSE)

## S3 method for class 'wa'
fitted(object, ...)

## S3 method for class 'wa'
residuals(object, ...)

## S3 method for class 'wa'
coef(object, ...)
```

```
waFit(x, y, tol.dw, useN2, deshrink, na.tol, small.tol,
      min.tol, f)
```

Arguments

`x`	The species training set data
`env, y`	The response vector
`deshrink`	Which deshrinking method to use? One of `"inverse"` or `"classical"`, `"expanded"`, `"none"`, or `"monotonic"`.
`tol.dw`	logical; should species with wider tolerances be given lower weight?
`useN2`	logical; should Hill's N2 values be used to produce un-biased tolerances?
`na.tol`	character; method to use to replace missing (`NA`) tolerances in WA computations. Missing values are replaced with the minimum, average or maximum tolerance observed that is not missing.
`small.tol`	character; method to replace small tolerances. See Details.
`min.tol`	numeric; threshold below which tolerances are treated as being 'small'. Default is not to replace small tolerances.
`f`	numeric, $0 < f < 1$; fraction of environmental gradient `env` to replace small tolerances with if `small.tol = "fraction"` is specified.
`formula`	a model formula
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables specified on the RHS of the model formula. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `wa` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`model`	logical. If `TRUE` the model frame is returned.
`object`	an Object of class `"wa"`, the result of a call to `wa`.
`...`	arguments to other methods.

Details

A typical model has the form `response ~ terms` where `response` is the (numeric) response vector (the variable to be predicted) and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of `.` is shorthand for all terms in `data` not already included in the model.

Species that have very small tolerances can dominate reconstructed values if tolerance down-weighting is used. In `wa`, small tolerances are defined as a tolerance that is $<$ `min.tol`. The default is to not replace small tolerances, and the user needs to specify suitable values of `min.tol`. Function `tolerance` may be of use in computing tolerances before fitting the WA model.

Small tolerances can be adjusted in several ways:

`min` small tolerances are replaced by the smallest observed tolerance that is greater than, or equal to, `min.tol`. With this method, the replaced values will be no smaller than any other observed tolerance. This is the default in **analogue**.

`mean` small tolerances are replaced by the average observed tolerance from the set that are greater than, or equal to, `min.tol`.

`fraction` small tolerances are replaced by the fraction, `f`, of the observed environmental gradient in the training set, `env`.

`absolute` small tolerances are replaced by `min.tol`.

Function `waFit` is the workhorse implementing the actual WA computations. It performs no checks on the input data and returns a simple list containing the optima, tolernances, model tolerances, fitted values, coefficients and the numbers of samples and species. See Value below for details of each component.

Value

An object of class `"wa"`, a list with the following components:

`wa.optima`	The WA optima for each species in the model.
`tolerances`	The actual tolerances calculated (these are weighted standard deviations).
`model.tol`	The tolerances used in the WA model computations. These will be similar to `tol`, but will no contain any `NA`s and any small tolerances will have been replaced with the appropriate value.
`fitted.values`	The fitted values of the response for each of the training set samples.
`residuals`	Model residuals.
`coefficients`	Deshrinking coefficients. Note that in the case of `deshrink = "monotonic"` this is a list with components `sm` (the representation of the smooth term as returned by `smoothCon`) and `p` (solutions to the least squares fit with monotonic constraints, the result of a call to `pcls`).
`rmse`	The RMSE of the model.
`r.squared`	The coefficient of determination of the observed and fitted values of the response.
`avg.bias`, `max.bias`	The average and maximum bias statistics.
`n.samp`, `n.spp`	The number of samples and species in the training set.
`deshrink`	The deshrinking regression method used.
`tol.dw`	logical; was tolerance down-weighting applied?
`call`	The matched function call.
`orig.x`	The training set species data.
`orig.env`	The response data for the training set.

`options.tol`	A list, containing the values of the arguments `useN2`, `na.tol`, `small.tol`, `min.tol`, and `f`.
`terms`, `model`	Model `terms` and `model.frame` components. Only returned by the `formula` method of `wa`.

Author(s)

Gavin L. Simpson and Jari Oksanen

See Also

`mat` for an alternative transfer function method.

Examples

```
data(ImbrieKipp)
data(SumSST)

## fit the WA model
mod <- wa(SumSST ~., data = ImbrieKipp)
mod

## extract the fitted values
fitted(mod)

## residuals for the training set
residuals(mod)

## deshrinking coefficients
coef(mod)

## diagnostics plots
par(mfrow = c(1,2))
plot(mod)
par(mfrow = c(1,1))

## caterpillar plot of optima and tolerances
caterpillarPlot(mod)                 ## observed tolerances
caterpillarPlot(mod, type = "model") ## with tolerances used in WA model

## plot diagnostics for the WA model
par(mfrow = c(1,2))
plot(mod)
par(mfrow = c(1,1))

## tolerance DW
mod2 <- wa(SumSST ~ ., data = ImbrieKipp, tol.dw = TRUE,
           min.tol = 2, small.tol = "min")
mod2

## compare actual tolerances to working values
with(mod2, rbind(tolerances, model.tol))
```

```
## tolerance DW
mod3 <- wa(SumSST ~ ., data = ImbrieKipp, tol.dw = TRUE,
           min.tol = 2, small.tol = "mean")
mod3

## fit a WA model with monotonic deshrinking
mod4 <- wa(SumSST ~., data = ImbrieKipp, deshrink = "monotonic")
mod4

## extract the fitted values
fitted(mod4)

## residuals for the training set
residuals(mod4)
```

`weightedCor` *Weighted correlation test of WA reconstruction*

Description

Weighted correlation between WA optima from training set and axis 1 scores of constrained ordination fitted to fossil data with WA model predictions for fossil samples as constraints.

Usage

```
## Default S3 method:
weightedCor(x, env, fossil, method = c("rda", "cca"),
            test = TRUE, type = c("simulate", "permute"), sim = 999,
            verbose = TRUE, ...)

## S3 method for class 'weightedCor'
plot(x,
     type = c("bubble", "null"),
     weighted = TRUE,
     size = 0.25,
     xlab = paste(x$env, "WA Optima"),
     ylab = "Axis 1 Score",
     xlim,
     main = "",
     sub = NULL,
     border = "gray75",
     col = "gray75",
     obscol = "red",
     fg = "black", ...)
```

Arguments

`x`	training set covariates, a matrix-like object usually of species/proxy data. For the `plot` method, an object of class `"weightedCor"`, the result of a call to `weightedCor`.
`env`	training set response, a vector usually of environmental data.
`fossil`	matrix of fossil/core species/proxy data for which a reconstruction is sought.
`method`	constrained ordination method. One of `"rda"` and `"cca"`. Currently only `"rda"` is supported.
`test`	logical; should the observed correlation be tested?
`type`	the type of test to apply. One of `"simulate"` or `"permute"`. The latter is currently not implemented. For the `plot` method, the type of plot to produce.
`sim`	numeric; number of simulations or permutations to permform as part of the test
`verbose`	logical; should the progress of the test be shown via a progress bar?
`...`	arguments passed to other methods. In the case of the `plot` method, additional graphical parameters can be supplied.
`weighted`	logical; should the null distribution plotted be of the weighted or normal correlation.
`size`	numeric; the size of the largest bubble in inches. See `symbols` and argument `inches` for details.
`xlim`, `xlab`, `ylab`, `main`, `sub`	graphical parameters with their usual meaning.
`border`, `col`	The border and fill colours for the histogram bars.
`fg`	The colour of the bubbles drawn on the bubble plot.
`obscol`	The colour of the indicator for the observed correlation.

Value

The `plot` method produces a plot on the current device. `weightedCor()` returns a list with the following components:

`wtdCorrel`, `Correl`	numeric; the observed weighted and standard correlation.
`data`	data frame; containing the training set WA Optima, axis 1 species scores, and mean abundance for each species.
`ord`	the fitted constrained ordination.
`model`	the fitted WA model.
`method`	the ordination method used.
`ndist`	the null distribution produced. `NULL` if argument `test` was `FALSE`.
`sim`	numeric; the number of simulations or permutations used to test the observed correlations.
`type`	the type of test performed.
`env`	the deparsed version of `env` argument. Used for plotting.
`call`	the matched function call.

Author(s)

Gavin L. Simpson

References

Telford R.J. and Birks, H.J.B. (2011) A novel method for assessing the statistical significance of quantitative reconstructions inferred from biotic assemblages. *Quanternary Science Reviews* **30**:1272-1278.

See Also

`wa` for details on fitting weighted average models.

Examples

```
data(ImbrieKipp, SumSST, V12.122)

Cor <- weightedCor(ImbrieKipp, env = SumSST,
                   fossil = V12.122, type = "simulate", sim = 49)
Cor

plot(Cor)
plot(Cor, type = "null")
```

Index