

# Package ‘rioja’

January 5, 2012

**Type** Package

**Title** Analysis of Quaternary Science Data

**Version** 0.7-3

**Date** 2011-12-24

**Author** Steve Juggins

**Maintainer** Steve Juggins <Stephen.Juggins@ncl.ac.uk>

**Depends** vegan

**Suggests** lattice, RODBC, gdata, mgcv

**Description** Functions for the analysis of Quaternary science data, including constrained clustering, WA, WAPLS, IKFA, MLRC and MAT transfer functions, and stratigraphic diagrams.

**License** GPL-3

**URL** <http://www.staff.ncl.ac.uk/staff/stephen.juggins/>

**Repository** CRAN

**Date/Publication** 2012-01-05 15:49:22

## R topics documented:

rioja-package . . . . .	2
aber . . . . .	3
chclust . . . . .	4
compare.datasets . . . . .	6
gutils . . . . .	8
IK . . . . .	9
IKFA . . . . .	9
inkspot . . . . .	14
interp.dataset . . . . .	16

LWR	17
MAT	19
Merge	23
MLRC	25
Ponds	28
PTF	29
read.C2Model	31
read.CEP	32
read.Tilia	34
RLGH	35
sp.plot	36
strat.plot	38
strat.plot.simple	42
SWAP	43
utils	44
WA	45
WAPLS	50

<b>Index</b>	<b>55</b>
--------------	-----------

---

rioja-package

*Analysis of Quaternary Science Data*

---

## Description

rioja: An R package for the analysis of Quaternary science data. Contains functions for constrained clustering, transfer functions, and plotting stratigraphic data.

## Details

Package: rioja  
 Type: Package  
 Version: 0.7-3  
 Date: 2012-01-05  
 License: GPL-3

### Main functions:

WA	Weighted averaging transfer functions
WAPLS	Partial least squares (PLS) and Weighted Averaging PLS transfer functions
MAT	Modern analogue technique transfer functions
IKFA	Imbrie and Kipp Factor Analysis
MLRC	Maximum likelihood response curves
LWR	Locally weighted transfer functions
crossval	Cross validate a transfer function model

predict	Predict using a transfer function model, with sample-specific errors
read.CEP	Read and write data in Cornell Ecology Program format
read.Tilia	Read Tilia files
read.C2Model	Import results form a C2 model
sp.plot	Plot species distributions along one or two environmental gradients
strat.plot	Plot a stratigraphic diagram
strat.plot.simple	Plot a simple stratigraphic diagram
chclust	Constrained heirarchical clustering (zonation)
bstick	Broken stick model to determine number of significant groups in a constrained cluster analysis
inkspot	Plot a two-way ordered diagram of species abundances
Merge	Merge two or more datasets by column names
interp.dataset	Interpolate one dataset to the depth or ages of another
compare.datasets	Assess conformity of species occurrence and abundance in two datasets

#### Datasets:

aber	Pollen stratigraphic data from the Abernethy forest
IK	Imbrie and Kipp foraminifera data
Ponds	Southeast England ponds and pools diatom dataset
SWAP	Surface Waters Acidification Programme lake diatom and pH dataset
RLGH	Diatom stratigraphic data from the Round Loch of Glenhead

#### Author(s)

Steve Juggins

Maintainer: Steve Juggins <Stephen.Juggins@ncl.ac.uk>

---

aber

*Abernethy Forest pollen data*

---

#### Description

Pollen stratigraphic data from Abernethy Forest, Scotland, spanning approximately 5500 - 12100 BP (from Birks & Mathews 1978). Data are percentages of 36 dryland pollen taxa in 49 samples.

**Usage**

```
data(aber)
```

**Source**

Birks, HH & Mathews, RW (1978). Studies in the vegetational history of Scotland V. Late Devonian and early Flandrian macrofossil stratigraphy at Abernethy Forest, Invernessshire. *New Phytologist* **80**, 455-84.

**Examples**

```
data(aber)
strat.plot(aber, scale.percent=TRUE, y.rev=TRUE)
```

---

chclust

*Constrained hierarchical clustering*


---

**Description**

Constrained hierarchical clustering.

**Usage**

```
chclust(d, method = "coniss")

## S3 method for class 'chclust'
plot(x, labels = NULL, hang = 0.1, axes = TRUE,
      xvar=1:(length(x$height)+1), xlim=NULL, ylim=NULL,
      x.rev = FALSE, y.rev=FALSE, horiz=FALSE, ...)

## S3 method for class 'chclust'
bstick(n, ng=10, plot=TRUE, ...)
```

**Arguments**

d	a dissimilarity structure as produced, for example, by <code>dist</code> or <code>vegdist</code> .
method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) either "coniss" or "conslink".
x, n	a constrained cluster object of class <code>chclust</code> produced by <code>chclust</code> .
xvar	numeric vector containing x-coordinates for the leaves of the dendrogram (see <i>details</i> below).
x.rev,y.rev	logical flags to reverse the x- or y-axis (and dendrogram labels). Defaults to FALSE.
horiz	logical indicating if the dendrogram should be drawn horizontally or not. Note that y-axis still refers to the dendrogram height even after rotating.

xlim, ylim	optional x- and y-limits of the plot, passed to the underlying plot function. The defaults for these show the full dendrogram.
labels, hang, axes	further arguments as in hclust.
ng	number of groups to display.
plot	logical to plot a broken stick model. Defaults to TRUE.
...	further graphical arguments. Use cex to change the text size of the x-axis labels, and cex.axis to change size of the y-axis values.

### Details

chclust performs a constrained hierarchical clustering of a distance matrix, with clusters constrained by sample order. Returns an object of class chclust which can be plotted and interrogated. See Grimm (1987), Gordon & Birks (1972) and Birks & Gordon (1985) for discussion of the coniss and conslink algorithms. The resulting dendrogram can be plotted with plot. This is an extension of plclust that allows the dendrogram to be plotted horizontally or vertically (default). plot also accepts a numeric vector coordinates for x-axis positions of the leaves of the dendrogram. These could, for example, be the stratigraphic depths of core samples or geographic distances along a line transect.

bstick.chclust compares the dispersion of a hierarchical classification to that obtained from a broken stick model and displays the results graphically. See Bennett (1996) for details. bstick is a generic function and the default method is defined in package vegan. If this package is loaded the function may be called using bstick, otherwise use bstick.chclust.

### Value

Function chclust returns an object of class chclust, derived from hclust.

### Author(s)

Steve Juggins

### References

- Bennett, K. (1996) Determination of the number of zones in a biostratigraphic sequence. *New Phytologist*, **132**, 155-170.
- Birks, H.J.B. & Gordon, A.D. (1985) *Numerical Methods in Quaternary Pollen Analysis* Academic Press, London.
- Gordon, A.D. & Birks, H.J.B. (1972) Numerical methods in Quaternary palaeoecology I. Zonation of pollen diagrams. *New Phytologist*, **71**, 961-979.
- Grimm, E.C. (1987) CONISS: A FORTRAN 77 program for stratigraphically constrained cluster analysis by the method of incremental sum of squares. *Computers & Geosciences*, **13**, 13-35.

### See Also

[hclust](#), [plclust](#), [cutree](#), [dendrogram](#), [bstick](#).

**Examples**

```

data(RLGH)
diss <- dist(sqrt(RLGH$spec/100))
clust <- chclust(diss)
bstick(clust, 10)
# Basic diagram
plot(clust, hang=-1)
# Rotated through 90 degrees
plot(clust, hang=-1, horiz=TRUE)
# Rotated and observations plotted according to sample depth.
plot(clust, xvar=RLGH$depth$Depth, hang=-1, horiz=TRUE, x.rev=TRUE)

# Conslink for comparison
clust <- chclust(diss, method = "conslink")
plot(clust, hang=-1)

```

---

compare.datasets

*Compare datasets for matching variables (species)*


---

**Description**

Compare two datasets and summarise species occurrence and abundance of species recorded in dataset one across dataset two. Useful for examining the conformity between sediment core and training set species data.

**Usage**

```

compare.datasets(y1, y2, n.cut=c(5, 10, 20, 50),
                 max.cut=c(2, 5, 10, 20, 50))

## S3 method for class 'compare.datasets'
plot(x, y, subset=1:nrow(x$obs), ...)

```

**Arguments**

y1, y2	two data frames or matrices, usually of biological species abundance data, to compare.
x	an object of class <code>compare.datasets</code> produce by function <code>compare.datasets</code> .
y	original dataset (ie. y1 above) used in comparison.
n.cut	vector of abundances to be used for species occurrence calculations (see details).
max.cut	vector of occurences to be used for species maximum abundance calculations (see details).
subset	a vector giving row indices to plot. Used to limit the number of plots with larges datasets.
...	additional arguments to <code>xyplot</code> .

## Details

Function `compare.datasets` compares two datasets. It summarises the species profile (number of occurrences etc.) and sample profile (number of species in each sample etc.) of dataset 1. For those species recorded in dataset 1 it also provides summaries of their occurrence and abundance in dataset 2. It is useful diagnostic for checking the conformity between core and training set data, specifically for identifying core taxa absent from the training set, and core samples with portions of their assemblage missing from the training set. Function `write.list.Excel` saves the output of `compare.datasets` in Excel format for more convenient browsing.

`plot.compare.datasets` provides a simple visualisation of the comparisons. It produces a matrix of plots, one for each sample in dataset 1, showing the abundance of each taxon in dataset 1 (x-axis) against the N2 value of that taxon in dataset 2 (y-axis, with symbols scaled according to abundance in dataset 2). The plots should aid identification of samples with high abundance of taxa that are rare (low N2) or have low abundance in the training set. Taxa that are absent from the training set are indicated with a red "+".

## Value

Function `compare.datasets` returns a list with two named elements:

<code>vars</code>	data frame listing for each variable in the first dataset: <code>N.occure</code> = number of occurrences in dataset 1, <code>N2</code> , Hill's N2 for species in dataset 1, <code>Max</code> = maximum value in dataset 1, <code>N.2</code> = number of occurrences in dataset 2, <code>N2.2</code> = Hill's N2 for species in dataset 2, <code>Max.2</code> = maximum value in dataset 2, <code>N.005</code> , number of occurrences where the species is greater than 5 etc.
<code>objs</code>	data frame listing for each observation in the first dataset: <code>N.taxa</code> = number of species greater than zero abundance, <code>N2</code> , Hill's N2 for samples, <code>Max</code> = maximum value, <code>total</code> = sample total, <code>M.002</code> = number of taxa with a maximum abundance greater than 2 etc., <code>N2.005</code> = number of taxa in dataset 1 with more than 5 occurrences in dataset 2 etc., <code>Sum.N2.005</code> = sample total including only those taxa with at least 5 occurrences in dataset 2 etc., <code>M2.005</code> = number of taxa in dataset 1 with maximum abundance greater than 2 in dataset 2 etc., and <code>Sum.M2.005</code> = sample total including only those taxa with a maximum abundance greater than 2 in dataset 2 etc.

Function `plot.compare.datasets` returns an object of class `trellis` which may be plotted.

## Author(s)

Steve Juggins

## See Also

`write.list.Excel` to save the output of `compare.datasets` in Excel format.

## Examples

```
# compare diatom data from core from Round Loch of Glenhead
# with SWAP surface sample dataset
data(RLGH)
```

```
data(SWAP)
result <- compare.datasets(RLGH$spec, SWAP$spec)
result

## Not run:
#save comparison to Excel for more convenient browsing
write.list.Excel(result, "Comparison.xls")

#visualise the comparison
plot.compare.datasets(result, RLGH$spec)

## End(Not run)
```

---

gutils

*Graphic utilities.*

---

## Description

Functions to perform simple graphics or enhance existing plots.

## Usage

```
hulls(x, y, gr, col.gr=NULL)
```

```
figCnvt(fig1, fig2)
```

## Arguments

<code>x, y</code>	vectors of x, y coordinates.
<code>gr</code>	factor to group observations.
<code>col.gr</code>	a single colour or a vector of colours of length <code>nG</code> , where <code>nG</code> is the number of groups.
<code>fig1, fig2</code>	original fig dimensions ( <code>fig1</code> ) and new fig2 dimensions ( <code>fig2</code> ). See details.

## Details

Function `hulls` is a wrapper for `chull` to add convex hulls to a scatterplot, optionally specifying a different colour for each hull.

Function `figCnvt` projects a set of fig dimensions `fig2` with respect to an original set `fig1`. Useful for laying out plots where the plotting region has already been partitioned using `fig`.

## Value

Function `figCnvt` returns a vector of 4 values specifying the new new figure dimensions.

## Author(s)

Steve Juggins

**Examples**

```
data(iris)
with(iris, plot(Sepal.Width, Sepal.Length, col=as.integer(Species)))
with(iris, hulls(Sepal.Width, Sepal.Length, gr=(Species)))
```

---

IK

*Imbrie and Kipp foraminifera data*

---

**Description**

Core-top foraminifera data from the Atlantic and Indian Oceans and core V12.122 from the Caribbean published by Imbrie and Kipp (1971). Dataset is a list with the following names components: spec relative abundances (percentages) of 22 foraminifera taxa in 61 core-top samples, (env) sea surface temperature and salinity measurements for the core-top samples, and (core) relative abundances of 28 foraminifer taxa in 110 samples from core V12.122.

**Usage**

```
data(IK)
```

**References**

Imbrie, J. & Kipp, N.G. (1971). A new micropaleontological method for quantitative paleoclimatology: application to a Late Pleistocene Caribbean core. In *The Late Cenozoic Glacial Ages* (ed K.K. Turekian), pp. 77-181. Yale University Press, New Haven.

**Examples**

```
data(IK)
names(IK$spec)
pairs(IK$env)
```

---

IKFA

*Imbrie & Kipp Factor Analysis*

---

**Description**

Functions for reconstructing (predicting) environmental values from biological assemblages using Imbrie & Kipp Factor Analysis (IKFA), as used in palaeoceanography.

**Usage**

```
IKFA(y, x, nFact = 5, IsPoly = FALSE, IsRot = TRUE,
      ccoef = 1:nFact, check.data=TRUE, lean=FALSE, ...)

IKFA.fit(y, x, nFact = 5, IsPoly = FALSE, IsRot = TRUE,
         ccoef = 1:nFact, lean=FALSE)

## S3 method for class 'IKFA'
predict(object, newdata=NULL, sse=FALSE, nboot=100,
        match.data=TRUE, verbose=TRUE, ...)

communality(object, y)

## S3 method for class 'IKFA'
crossval(object, cv.method="loo", verbose=TRUE, ngroups=10,
         nboot=100, ...)

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

## S3 method for class 'IKFA'
rand.t.test(object, n.perm=999, ...)

## S3 method for class 'IKFA'
screeplot(x, rand.test=TRUE, ...)

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

## S3 method for class 'IKFA'
summary(object, full=FALSE, ...)

## S3 method for class 'IKFA'
plot(x, resid=FALSE, xval=FALSE, nFact=max(x$ccoef),
     xlab="", ylab="", ylim=NULL, xlim=NULL, add.ref=TRUE,
     add.smooth=FALSE, ...)

## S3 method for class 'IKFA'
residuals(object, cv=FALSE, ...)

## S3 method for class 'IKFA'
coef(object, ...)

## S3 method for class 'IKFA'
fitted(object, ...)
```

**Arguments**

<code>y</code>	a data frame or matrix of biological abundance data.
<code>x, object</code>	a vector of environmental values to be modelled or an object of class <code>wa</code> .
<code>newdata</code>	new biological data to be predicted.
<code>nFact</code>	number of factor to extract.
<code>IsRot</code>	logical to rotate factors.
<code>ccoef</code>	vector of factor numbers to include in the predictions.
<code>IsPoly</code>	logical to include quadratic of the factors as predictors in the regression.
<code>check.data</code>	logical to perform simple checks on the input data.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to FALSE if you are sure the column names match exactly.
<code>lean</code>	logical to exclude some output from the resulting models (used when cross-validating to speed calculations).
<code>full</code>	logical to show head and tail of output in summaries.
<code>resid</code>	logical to plot residuals instead of fitted values.
<code>xval</code>	logical to plot cross-validation estimates.
<code>xlab, ylab, xlim, ylim</code>	additional graphical arguments to plot <code>wa</code> .
<code>add.ref</code>	add 1:1 line on plot.
<code>add.smooth</code>	add loess smooth to plot.
<code>cv.method</code>	cross-validation method, either "loo", "lgo" or "bootstrap".
<code>verbose</code>	logical or integer to show feedback during cross-validation. If TRUE print feedback every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation, or a vector contain leave-out group membership.
<code>sse</code>	logical indicating that sample specific errors should be calculated.
<code>rand.test</code>	logical to perform a randomisation t-test to test significance of cross validated factors.
<code>n.perm</code>	number of permutations for randomisation t-test.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

**Details**

Function `IKFA` performs Imbrie and Kipp Factor Analysis, a form of Principal Components Regression (Imbrie & Kipp 1971).

Function `predict` predicts values of the environmental variable for `newdata` or returns the fitted (predicted) values from the original modern dataset if `newdata` is NULL. Variables are matched

between training and newdata by column name (if `match.data` is TRUE). Use `compare.datasets` to assess conformity of two species datasets and identify possible no-analogue samples.

IKFA has methods `fitted` and `rediduals` that return the fitted values (estimates) and residuals for the training set, `performance`, which returns summary performance statistics (see below), `coef` which returns the species coefficients, and `print` and `summary` to summarise the output. IKFA also has a `plot` method that produces scatter plots of predicted vs observed measurements for the training set.

Function `rand.t.test` performs a randomisation t-test to test the significance of the cross-validated components after van der Voet (1994).

Function `screepplot` displays the RMSE of prediction for the training set as a function of the number of factors and is useful for estimating the optimal number for use in prediction. By default `screepplot` will also carry out a randomisation t-test and add a line to scree plot indicating percentage change in RMSE with each component annotate with the p-value from the randomisation test.

## Value

Function IKFA returns an object of class IKFA with the following named elements:

<code>coefficients</code>	species coefficients (the updated "optima").
<code>meanY</code>	weighted mean of the environmental variable.
<code>iswapls</code>	logical indicating whether analysis was IKFA (TRUE) or PLS (FALSE).
<code>T</code>	sample scores.
<code>P</code>	variable (species) scores.
<code>npls</code>	number of pls components extracted.
<code>fitted.values</code>	fitted values for the training set.
<code>call</code>	original function call.
<code>x</code>	environmental variable used in the model.
<code>standx, meanT, sdX</code>	additional information returned for a PLSif model.

Function `crossval` also returns an object of class IKFA and adds the following named elements:

<code>predicted</code>	predicted values of each training set sample under cross-validation.
<code>residuals.cv</code>	prediction residuals.

If function `predict` is called with `newdata=NULL` it returns the fitted values of the original model, otherwise it returns a list with the following named elements:

<code>fit</code>	predicted values for newdata.
------------------	-------------------------------

if sample specific errors were generated the list will also include:

<code>fit.boot</code>	mean of the bootstrap estimates of newdata.
<code>v1</code>	squared standard error of the bootstrap estimates for each new sample.
<code>v2</code>	mean squared error for the training set samples, across all bootstrap samples.

SEP                    standard error of prediction, calculated as the square root of  $v1 + v2$ .

Function `performance` returns a matrix of performance statistics for the IKFA model. See [performance](#), for a description of the summary.

Function `rand.t.test` returns a matrix of performance statistics together with columns indicating the p-value and percentage change in RMSE with each higher component (see van der Veot (1994) for details).

### Author(s)

Steve Juggins

### References

Imbrie, J. & Kipp, N.G. (1971). A new micropaleontological method for quantitative paleoclimatology: application to a Late Pleistocene Caribbean core. In *The Late Cenozoic Glacial Ages* (ed K.K. Turekian), pp. 77-181. Yale University Press, New Haven.

van der Voet, H. (1994) Comparing the predictive accuracy of models using a simple randomization test. *Chemometrics and Intelligent Laboratory Systems*, **25**, 313-323.

### See Also

[WA](#), [MAT](#), [performance](#), and [compare.datasets](#) for diagnostics.

### Examples

```
data(IK)
spec <- IK$spec
SumSST <- IK$env$SumSST
core <- IK$core

fit <- IKFA(spec, SumSST)
fit
# cross-validate model
fit.cv <- crossval(fit, cv.method="lgo")
# How many components to use?
screplot(fit.cv)

#predict the core
pred <- predict(fit, core, npls=2)

#plot predictions - depths are in rownames
depth <- as.numeric(rownames(core))
plot(depth, pred$fit[, 2], type="b")

# fit using only factors 1, 2, 4, & 5
# and using polynomial terms
# as Imbrie & Kipp (1971)
fit2 <- IKFA(spec, SumSST, ccoef=c(1, 2, 4, 5), IsPoly=TRUE)
fit2.cv <- crossval(fit2, cv.method="lgo")
screplot(fit2.cv)
```

```
## Not run:
# predictions with sample specific errors
# takes approximately 1 minute to run
pred <- predict(fit, core, sse=TRUE, nboot=1000)
pred

## End(Not run)
```

---

inkspot

*Two-way ordered bubble plot of a species by sites data table*


---

## Description

Plots a two-way ordered bubble plot of a species by sites data table, where species are rows and sites are columns. The sites can be ordered and the functions will sort species to cluster abundances on the diagonal.

## Usage

```
inkspot(data, gradient=1:nrow(data), use.rank=FALSE,
        reorder.species = TRUE, x.axis=c("sites", "gradient",
        "none"), x.axis.top=FALSE, site.names=NULL, spec.names=NULL,
        pch=21, cex.max=3, col="black", bg="darkgrey",
        legend.values=c(2, 5, 10, 20, 50), ...)
```

## Arguments

<code>data</code>	data frame to be plotted.
<code>gradient</code>	a vector for ordering sites along the x-axis.
<code>use.rank</code>	logical to indicate that the rank rather than absolute values of gradient should be used to plot site positions. Defaults to FALSE.
<code>reorder.species</code>	should species be reordered to reflect pattern in site ordering? Defaults to TRUE.
<code>x.axis</code>	controls labelling of x.axis. <code>sites</code> uses site names, <code>gradient</code> plots an axis reflecting values of the supplied gradient, and <code>none</code> omits labels and draws ticks at the site positions.
<code>x.axis.top</code>	logical to include an x-axis on the top of the figure labelled with values of the gradient.
<code>site.names, spec.names</code>	character vectors of site or species names to annotate the axes. Defaults to row and column names.
<code>cex.max</code>	maximum size of plotting symbol. Symbols are scaled so maximum species abundance has a symbol of this size.
<code>pch, col, bg</code>	plotting symbol and line / fill colour.
<code>...</code>	additional arguments to plot.
<code>legend.values</code>	if not null, places a legend in the top-left corner displaying the listed values.

**Details**

Function `inkspot` plots a two-way table of species by sites as a bubble plot, with sites ordered along the x-axis, species on the y-axis, and abundance indicated by scaled symbols ("bubbles"). It is a useful way to visualise species distribution along an environmental, spatial or temporal gradient. If `gradient` is not given sites are plotting in the order they appear in the input data. Otherwise sites are plotting according to the values in `gradient`. If site labels overlap (multiple sites at similar values of `gradient`), labels can be suppressed `x.axis="none"`, or replaced with the gradient axis `x.axis="gradient"`. A gradient axis can also be added to the top x-axis (`x.axis.top=TRUE`). Symbols are scaled so that the maximum abundance has a symbol size of `cex.max`. If sites are spaced unevenly along the gradient, or if many labels overlap, sites may be plotted evenly spaced using `use.rank=TRUE`. In this case the function will place top axis labels (if requested) at the appropriate positions along the gradient.

**Value**

Function `inkspot` returns a list with two named elements:

<code>spec</code>	index of the species order.
<code>site</code>	index of the site order.

**Author(s)**

Steve Juggins

**See Also**

[vegemite](#) in package `vegan` for a tabular alternative.

**Examples**

```
data(SWAP)
mx <- apply(SWAP$spec, 2, max)
spec <- SWAP$spec[, mx > 10]
#basic plot of data with legend
inkspot(spec, cex.axis=0.6)

#order sites by pH
pH <- SWAP$pH
inkspot(spec, pH, cex.axis=0.6)

# add a top axis
inkspot(spec, pH, x.axis.top=TRUE, cex.axis=0.6)

# order by pH but plot sites at regular intervals to avoid label overlap
inkspot(spec, pH, use.rank=TRUE, x.axis.top=TRUE, cex.axis=0.6)

# or add long taxon names
oldmar <- par("mar")
par(mar=c(3,12,2,1))
nms <- SWAP$names[mx > 10, 2]
inkspot(spec, pH, spec.names=as.character(nms), use.rank=TRUE,
```

```
x.axis.top=TRUE, cex.axis=0.6)
par(mar=oldmar)
```

---

interp.dataset	<i>Interpolate a dataset</i>
----------------	------------------------------

---

### Description

Given a data frame of variables measured along a temporal or spatial gradient, interpolate each variable to new values of the gradient. Useful for interpolating sediment core data to the depths of ages of another sequences, or to evenly spaced intervals.

### Usage

```
interp.dataset(y, x, xout, method=c("linear", "loess", "sspline"),
  rep.negt=TRUE, span=0.25, df=min(20, nrow(y)*.7), ...)
```

### Arguments

y	data frame to be interpolated.
x	numeric vector giving ages, depths (ie. x-values( for data frame to be interpolated.
xout	numeric vector of values to interpolate to.
method	interpolation method, should be an unambiguous abbreviation of either linear, loess or sspline. See details.
rep.negt	logical to indicate whether or not to replace negative values with zero in the interpolated data.
span	span for loess, default=0.25.
df	degrees of freedom for smoothing spline, default is the lower of 20 or 0.7 * number of samples.
...	additional arguments to loess, smooth.spline.

### Details

Function `interp.dataset` interpolates the columns of data frame with rows measured at intervals given by `x`, to new intervals given by `xout`. This function is useful to interpolate one set of sediment core data to the depth or ages of another, or to a regular set of intervals. Interpolation can be done using linear interpolation between data points in the original series (default), using a fitted `link[stats]{loess}` locally weighted regression, or by `link[stats]{smooth.spline}`. The latter two methods will also smooth the data and additional arguments may be passed to these functions to control the amount of smoothing.

The effects of the interpolation can be visualised against the original sequence using `strat.plot.simple`.

**Value**

Function `interp.datasets` returns a data frame of the input data interpolated to the values given in `xout`. Values of `xout` outside the range of the original data are replaced by NA.

**Author(s)**

Steve Juggins

**See Also**

[approx](#), [loess](#), [smooth.spline](#) for details of interpolation methods, and [strat.plot.simple](#) to visualise the effect of interpolation and / or smoothing.

**Examples**

```
data(RLGH)
spec <- RLGH$spec
depth <- RLGH$depth$Depth

# interpolate new dataset to every 0.5 cm
# using default method (linear)
x.new <- seq(0, 20, by=0.5)
sp.interp <- interp.dataset(y=spec, x=depth, xout=x.new)
## Not run:
# examine the results and compare to original data
strat.plot.simple(spec, depth, sp.interp, x.new)

## End(Not run)
```

---

LWR

*Weighted averaging (LWR) regression and calibration*


---

**Description**

Functions for reconstructing (predicting) environmental values from biological assemblages using weighted averaging (LWR) regression and calibration.

**Usage**

```
LWR(y, x, FUN=WA, dist.method="sq.chord", k=30, lean=TRUE,
    fit.model=TRUE, check.data=TRUE, verbose=TRUE, ...)

## S3 method for class 'LWR'
predict(object, newdata=NULL, k = object$k, sse=FALSE,
        nboot=100, match.data=TRUE, verbose=TRUE, lean=TRUE, ...)

## S3 method for class 'LWR'
crossval(object, k=object$k, cv.method="loo", verbose=TRUE,
```

```

        ngroups=10, nboot=100, ...)

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

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

## S3 method for class 'LWR'
summary(object, full=FALSE, ...)

## S3 method for class 'LWR'
residuals(object, cv=FALSE, ...)

## S3 method for class 'LWR'
fitted(object, ...)

```

### Arguments

<code>y</code>	a data frame or matrix of biological abundance data.
<code>x, object</code>	a vector of environmental values to be modelled or an object of class LWR.
<code>dist.method</code>	distance measure used to derfine closest analogues.
<code>k</code>	number of close analogues to use in calibration function.
<code>FUN</code>	calibration function (e.g. WA, WAPLS etc).
<code>newdata</code>	new biological data to be predicted.
<code>fit.model</code>	TRUE fits model to training set. FALSE omist this step and builds a LWR object than can be used for prediction.
<code>check.data</code>	logical to perform simple checks on the input data.
<code>full</code>	logical to show head and tail of output in summaries.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to FALSE if you are sure the column names match exactly.
<code>lean</code>	logical to exclude some output from the resulting models (used when cross-validating to speed calculations).
<code>cv.method</code>	cross-validation method, either "loo", "lgo" or "bootstrap".
<code>verbose</code>	logical or integer to show feedback during cross-validaton. If TRUE print feed-back every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation.
<code>sse</code>	logical indicating that sample specific errors should be calculated.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

**Details**

Function LWR performs ... To do.

**Value**

Function LWR returns an object of class LWR with the following named elements:

**Author(s)**

Steve Juggins

**See Also**

[WAPLS](#), [MAT](#), and [compare.datasets](#) for diagnostics.

---

MAT	<i>Palaeoenvironmental reconstruction using the Modern Analogue Technique (MAT)</i>
-----	---

---

**Description**

Functions for reconstructing (predicting) environmental values from biological assemblages using the Modern Analogue Technique (MAT), also know as k nearest neighbours (k-NN).

**Usage**

```
MAT(y, x, dist.method="sq.chord", k=5, lean=TRUE)

## S3 method for class 'MAT'
predict(object, newdata=NULL, k=object$k, sse=FALSE,
        nboot=100, match.data=TRUE, verbose=TRUE, lean=TRUE,
        ...)

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

## S3 method for class 'MAT'
crossval(object, k=object$k, cv.method="lgo",
         verbose=TRUE, ngroups=10, nboot=100, ...)

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

## S3 method for class 'MAT'
summary(object, full=FALSE, ...)

## S3 method for class 'MAT'
```

```

plot(x, resid=FALSE, xval=FALSE, k=5, wMean=FALSE, xlab="",
     ylab="", ylim=NULL, xlim=NULL, add.ref=TRUE,
     add.smooth=FALSE, ...)

## S3 method for class 'MAT'
residuals(object, cv=FALSE, ...)

## S3 method for class 'MAT'
fitted(object, ...)

## S3 method for class 'MAT'
screepplot(x, ...)

paldist(y, dist.method="sq.chord")

paldist2(y1, y2, dist.method="sq.chord")

```

### Arguments

<code>y, y1, y2</code>	data frame containing biological data.
<code>newdata</code>	data frame containing biological data to predict from.
<code>x</code>	a vector of environmental values to be modelled, matched to <code>y</code> .
<code>dist.method</code>	dissimilarity coefficient. See details for options.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to <code>FALSE</code> if you are sure the column names match exactly.
<code>k</code>	number of analogues to use.
<code>lean</code>	logical to remove items form the output.
<code>object</code>	an object of class <code>MAT</code> .
<code>resid</code>	logical to plot residuals instead of fitted values.
<code>xval</code>	logical to plot cross-validation estimates.
<code>wMean</code>	logical to plot weighted-mean estimates.
<code>xlab, ylab, xlim, ylim</code>	additional graphical arguments to <code>plot.wa</code> .
<code>add.ref</code>	add 1:1 line on plot.
<code>add.smooth</code>	add loess smooth to plot.
<code>cv.method</code>	cross-validation method, either <code>"lgo"</code> or <code>"bootstrap"</code> .
<code>verbose</code>	logical or integer to show feedback during cross-validaton. If <code>TRUE</code> print feedback every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation, or a vector contain leave-out group membership.
<code>sse</code>	logical indicating that sample specific errors should be calculated.

<code>full</code>	logical to indicate a full or abbreviated summary.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

## Details

MAT performs an environmental reconstruction using the modern analogue technique. Function MAT takes a training dataset of biological data (species abundances) `y` and a single associated environmental variable `x`, and generates a model of closest analogues, or matches, for the modern data using one of a number of dissimilarity coefficients. Options for the latter are: "euclidean", "sq.euclidean", "chord", "sq.chord", "chord.t", "sq.chord.t", "chi.squared", "sq.chi.squared", "bray". "chord.t" are true chord distances, "chord" refers to the the variant of chord distance using in palaeoecology (e.g. Overpeck et al. 1985), which is actually Hellinger's distance (Legendre & Gallagher 2001). There are various help functions to plot and extract information from the results of a MAT transfer function. The function `predict` takes MAT object and uses it to predict environmental values for a new set of species data, or returns the fitted (predicted) values from the original modern dataset if `newdata` is NULL. Variables are matched between training and `newdata` by column name (if `match.data` is TRUE). Use [compare.datasets](#) to assess conformity of two species datasets and identify possible no-analogue samples.

MAT has methods `fitted` and `residuals` that return the fitted values (estimates) and residuals for the training set, `performance`, which returns summary performance statistics (see below), and `print` and `summary` to summarise the output. MAT also has a `plot` method that produces scatter plots of predicted vs observed measurements for the training set.

Function `screepplot` displays the RMSE of prediction for the training set as a function of the number of analogues (`k`) and is useful for estimating the optimal value of `k` for use in prediction.

`paldist` and `paldist1` are helper functions though they may be called directly. `paldist` takes a single data frame or matrix returns a distance matrix of the row-wise dissimilarities. `paldist2` takes two data frames of matrices and returns a matrix of all row-wise dissimilarities between the two datasets.

## Value

Function MAT returns an object of class MAT which contains the following items:

<code>call</code>	original function call to MAT.
<code>fitted.vales</code>	fitted (predicted) values for the training set, as the mean and weighted mean (weighed by dissimilarity) of the <code>k</code> closest analogues.
<code>diagnostics</code>	standard deviation of the <code>k</code> analogues and dissimilarity of the closest analogue.
<code>dist.n</code>	dissimilarities of the <code>k</code> closest analogues.
<code>x.n</code>	environmental values of the <code>k</code> closest analogues.
<code>match.name</code>	column names of the <code>k</code> closest analogues.
<code>x</code>	environmental variable used in the model.
<code>dist.method</code>	dissimilarity coefficient.
<code>k</code>	number of closest analogues to use.
<code>y</code>	original species data.

`cv.summary` summary of the cross-validation (not yet implemented).  
`dist` dissimilarity matrix (returned if `lean=FALSE`).

If function `predict` is called with `newdata=NULL` it returns a matrix of fitted values from the original training set analysis. If `newdata` is not `NULL` it returns list with the following named elements:

`fit` predictions for `newdata`.  
`diagnostics` standard deviations of the `k` closest analogues and distance of closest analogue.  
`dist.n` dissimilarities of the `k` closest analogues.  
`x.n` environmental values of the `k` closest analogues.  
`match.name` column names of the `k` closest analogues.  
`dist` dissimilarity matrix (returned if `lean=FALSE`).

If sample specific errors were requested the list will also include:

`fit.boot` mean of the bootstrap estimates of `newdata`.  
`v1` squared standard error of the bootstrap estimates for each new sample.  
`v2` mean squared error for the training set samples, across all bootstram samples.  
`SEP` standard error of prediction, calculated as the square root of `v1 + v2`.

Functions `paldist` and `paldist2` return dissimilarity matrices. `performance` returns a matrix of performance statistics for the MAT model, with columns for RMSE, R2, mean and max bias for each number of analogues up to `k`. See [performance](#) for a description of the output.

### Author(s)

Steve Juggins

### References

Legendre, P. & Gallagher, E. (2001) Ecologically meaningful transformations for ordination of species. *Oecologia*, **129**, 271-280.  
 Overpeck, J.T., Webb, T., III, & Prentice, I.C. (1985) Quantitative interpretation of fossil pollen spectra: dissimilarity coefficients and the method of modern analogs. *Quaternary Research*, **23**, 87-108.

### See Also

[WAPLS](#), [WA](#), [performance](#), and [compare.datasets](#) for diagnostics.

### Examples

```
# pH reconstruction of the RLGH, Scotland, using SWAP training set
# shows recent acidification history
data(SWAP)
data(RLGH)
fit <- MAT(SWAP$spec, SWAP$pH, k=20) # generate results for k 1-20
#examine performance
performance(fit)
```

```

print(fit)
# How many analogues?
screepplot(fit)
# do the reconstruction
pred.mat <- predict(fit, RLGH$spec, k=10)
# plot the reconstruction
plot(RLGH$depths$Age, pred.mat$fit[, 1], type="b", ylab="pH", xlab="Age")

#compare to a weighted average model
fit <- WA(SWAP$spec, SWAP$pH)
pred.wa <- predict(fit, RLGH$spec)
points(RLGH$depths$Age, pred.wa$fit[, 1], col="red", type="b")
legend("topleft", c("MAT", "WA"), lty=1, col=c("black", "red"))

```

---

Merge	<i>Merges two or more data frames on the basis of common column names.</i>
-------	--

---

### Description

Merges two or more data frames on the basis of common column names.

### Usage

```
Merge(..., join="outer", fill=0, split=FALSE)
```

### Arguments

...	two or more data frames to merge.
join	type of join to perform. Should be an unambiguous abbreviation of either "outer", "inner", or "leftouter". An outer join produces a data frame that contains all the unique column names of the input data, ie, the union of all input column names. An inner join produces a data frame containing only column names that are common across the input data, ie. the intersection of the input column names. A left outer join produces a data frame containing all column names of the first data frame only: column names that occur in subsequent data frames are omitted.
fill	value to use to fill non-matched columns. Defaults to zero which is appropriate for species abundance data.
split	logical to return a single data frame (TRUE) or a named list containing separate (original) data frames with a common set of merged columns (FALSE). Defaults to TRUE (a single data frame).

## Details

Merge is a utility function for combining separate datasets of biological count data that have only a subset of taxa (column names) in common. The outer join is appropriate for merging prior to a joint ordination or for merging a training set and core data prior to environmental reconstruction using the modern analogue technique (MAT). A left outer join should be used to prepare data for an ordination of a training set and subsequent projection of a second onto the ordination axes. The function is capitalised to distinguish it from merge in the base R.

## Value

If split is set to FALSE the function returns a single data frame with the number of rows equal to the combined rows of the input data and columns sorted alphabetically according to the join type. Otherwise returns a named list of the merged data frames.

## Author(s)

Steve Juggins

## See Also

[merge](#).

## Examples

```
data(RLGH)
data(SWAP)
# Merge RLGH core data with SWAP training set
# Extract species data from datasets
SWAPsp <- SWAP$spec
RLGHsp <- RLGH$spec
# full outer join for joint ordination of both datasets
comb <- Merge(SWAPsp, RLGHsp)

## Not run:
# superimpose core trajectory on ordination plot
library(vegan) # decorana
ord <- decorana(comb, iweigh=1)
par(mfrow=c(1,2))
plot(ord, display="sites")
sc <- scores(ord, display="sites")
sc <- sc[(nrow(SWAPsp)+1):nrow(comb), ]
lines(sc, col="red")
title("Joint DCA ordination of surface and core")

# Do the same but this time project core passively
# Note we cannot use data from the outer join since decorana
# will delete taxa only present in the core - the resulting
# ordination model will then not match the taxa in the core
comb2 <- Merge(SWAPsp, RLGHsp, join="leftouter", split=TRUE)
ord2 <- decorana(comb2$SWAPsp, iweigh=1)
sc2 <- predict(ord2, comb2$RLGHsp, type="sites")
```

```

plot(ord2, display="sites")
lines(sc2, col="red")
title("DCA with core added \"passively\"")

## End(Not run)

```

---

MLRC	<i>Palaeoenvironmental reconstruction using Maximum Likelihood Response Surfaces</i>
------	--

---

## Description

Functions for reconstructing (predicting) environmental values from biological assemblages using Maximum Likelihood response Surfaces.

## Usage

```

MLRC(y, x, check.data=TRUE, lean=FALSE, ...)

MLRC.fit(y, x, n.cut=2, use.glm=FALSE, max.iter=50, lean=FALSE, ...)

## S3 method for class 'MLRC'
predict(object, newdata=NULL, sse=FALSE, nboot=100,
        match.data=TRUE, verbose=TRUE, ...)

## S3 method for class 'MLRC'
crossval(object, cv.method="loo", verbose=TRUE, ngroups=10,
        nboot=100, ...)

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

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

## S3 method for class 'MLRC'
summary(object, full=FALSE, ...)

## S3 method for class 'MLRC'
plot(x, resid=FALSE, xval=FALSE, xlab="", ylab="",
     ylim=NULL, xlim=NULL, add.ref=TRUE, add.smooth=FALSE, ...)

## S3 method for class 'MLRC'
residuals(object, cv=FALSE, ...)

## S3 method for class 'MLRC'
coef(object, ...)

```

```
## S3 method for class 'MLRC'
fitted(object, ...)
```

### Arguments

<code>y</code>	a data frame or matrix of biological abundance data.
<code>x, object</code>	a vector of environmental values to be modelled or an object of class <code>wa</code> .
<code>n.cut</code>	cutoff value for number of occurrences. Species with fewer than <code>n.cut</code> occurrences will be excluded for the analysis.
<code>use.glm</code>	logical to use <code>glm</code> to fit responses rather than internal code. Defaults to <code>FALSE</code> .
<code>newdata</code>	new biological data to be predicted.
<code>max.iter</code>	maximum iterations of the logit regression algorithm.
<code>check.data</code>	logical to perform simple checks on the input data.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to <code>FALSE</code> if you are sure the column names match exactly.
<code>lean</code>	logical to exclude some output from the resulting models (used when cross-validating to speed calculations).
<code>full</code>	logical to show head and tail of output in summaries.
<code>resid</code>	logical to plot residuals instead of fitted values.
<code>xval</code>	logical to plot cross-validation estimates.
<code>xlab, ylab, xlim, ylim</code>	additional graphical arguments to <code>plot.wa</code> .
<code>add.ref</code>	add 1:1 line on plot.
<code>add.smooth</code>	add loess smooth to plot.
<code>cv.method</code>	cross-validation method, either "loo", "lgo" or "bootstrap".
<code>verbose</code>	logical or integer to show feedback during cross-validation. If <code>TRUE</code> print feedback every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation, or a vector contain leave-out group membership.
<code>sse</code>	logical indicating that sample specific errors should be calculated.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

### Details

Function `MLRC` Maximim likelihood reconstruction using response curves.

Function `predict` predicts values of the environmental variable for `newdata` or returns the fitted (predicted) values from the original modern dataset if `newdata` is `NULL`. Variables are matched

between training and newdata by column name (if `match.data` is TRUE). Use [compare.datasets](#) to assess conformity of two species datasets and identify possible no-analogue samples.

MLRC has methods `fitted` and `rediduals` that return the fitted values (estimates) and residuals for the training set, `performance`, which returns summary performance statistics (see below), `coef` which returns the species coefficients, and `print` and `summary` to summarise the output. MLRC also has a `plot` method that produces scatter plots of predicted vs observed measurements for the training set.

## Value

Function MLRC returns an object of class MLRC with the following named elements:

To do

Function `crossval` also returns an object of class MLRC and adds the following named elements:

`predicted` predicted values of each training set sample under cross-validation.

`residuals.cv` prediction residuals.

If function `predict` is called with `newdata=NULL` it returns the fitted values of the original model, otherwise it returns a list with the following named elements:

`fit` predicted values for newdata.

if sample specific errors were generated the list will also include:

`fit.boot` mean of the bootstrap estimates of newdata.

`v1` squared standard error of the bootstrap estimates for each new sample.

`v2` mean squared error for the training set samples, across all bootstrap samples.

`SEP` standard error of prediction, calculated as the square root of  $v1 + v2$ .

Function `performance` returns a matrix of performance statistics for the MLRC model. See [performance](#), for a description of the summary.

## Author(s)

Steve Juggins

## References

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

Juggins, S. (1992) Diatoms in the Thames Estuary, England: Ecology, Palaeoecology, and Salinity Transfer Function. *Bibliotheca Diatomologica*, **Band 25**, 216pp.

Oksanen, J., Laara, E., Huttunen, P., & Merilainen, J. (1990) Maximum likelihood prediction of lake acidity based on sedimented diatoms. *Journal of Vegetation Science*, **1**, 49-56.

ter Braak, C.J.F. & van Dam, H. (1989) Inferring pH from diatoms: a comparison of old and new calibration methods. *Hydrobiologia*, **178**, 209-223.

**See Also**

[WA](#), [MAT](#), [performance](#), and [compare.datasets](#) for diagnostics.

**Examples**

```
data(IK)
spec <- IK$spec / 100
SumSST <- IK$env$SumSST
core <- IK$core / 100

fit <- MLRC(spec, SumSST)
fit

#predict the core
pred <- predict(fit, core)

#plot predictions - depths are in rownames
depth <- as.numeric(rownames(core))
plot(depth, pred$fit[, 1], type="b")

## Not run:
# this is slow!
# cross-validate model
fit.cv <- crossval(fit, cv.method="loo", verbose=5)

# predictions with sample specific errors
pred <- predict(fit, core, sse=TRUE, nboot=1000, verbose=5)

## End(Not run)
```

---

Ponds

*Southeast England ponds and pools diatom and water chemistry dataset.*

---

**Description**

Diatom and associated water chemistry data for 30 small ponds & pools from SE England collected by, and described in Bennion (1994). Dataset is a list with the following named elements: (spec) diatom relative abundances for 48 selected common taxa, (env) lake names, UK GB grid references, lake depth (m) and mean lake-water chemistry. Units are ueq/l except pH, conductivity (uS/cm), alkalinity (meq/l), total phosphorus and chlorophyll-a (ug/l), and nitrate (mg/l). Column names in spec are short, 6-character alphanumeric codes for each diatom taxon. Ponds\$names contains the full names for each taxon, in the correct order).

**Usage**

```
data(Ponds)
```

**Source**

Bennion, H. (1994) A diatom-phosphorus transfer function for shallow, eutrophic ponds in southeast England. *Hydrobiologia*, **275/276**, 391-410.

**Examples**

```
data(Ponds)
names(Ponds$spec)
hist(Ponds$env$TP)
```

PTF

*Palaeoecological transfer functions***Description**

Functions for diagnosing and interpreting palaeoecological transfer functions.

**Usage**

```
## Default S3 method:
performance(object, ...)

## Default S3 method:
crossval(object, ...)
```

**Arguments**

`object` a transfer function model from `wa`, `wapls` etc.  
`...` additional arguments.

**Details**

Package `rioja` implements a number of numerical methods for inferring the value of an environmental variable from a set of species abundances, given a modern training set of species data and associated environmental values. In palaeoecology these are known as "transfer functions" or "inference models" and are used to hindcast or "reconstruct" past environmental conditions from sub-fossil species assemblages preserved in sediment cores. The techniques included are weighted averaging (*WA*), partial least squares (PLS) and weighted average partial least squared (*WAPLS*), Imbrie and Kipp Factor Analysis (*IKFA*) a form of principal components regression, Maximum Likelihood response Curves (*MLRC*), and the Modern Analogue Technique (*MAT*, a form of k-NN non-parametric regression (see Juggins & Birks (2010) for a review).

The techniques are implemented in a consistent way and include functions for fitting a model to a training set of species and environmental data, with the function name named after the technique: that is, *WA* fits a weighted averaging model. Any model can be cross-validated using the `crossval` function, which allows internal cross-validation using leave-one-out, leave-n-out, or bootstrapping.

There are a number of generic functions that can be used to summarise and diagnose the models: (`print`, `summary`, `performance` and `plot`). Some techniques have additional diagnostic functions such as `screeplot` and `rand.t.test` to help estimate the appropriate number of components (WAPLS), factors (IKFA) or number of analogues (IKFA).

Predictions for new species data can be made using `predict`, with an option to calculate sample-specific errors using bootstrapping, after the method described in Birks et al. (1990).

## Value

Function `performance` returns a list with a named matrix object which contains the following columns:

RMSE	root mean squared error, defined as the square root of the average squared error between the observed and predicted values for the training set.
R2	squared correlation between observed and predicted values.
Avg.Bias	mean bias (mean of the residuals between measured and predicted values).
Max.Bias	maximum bias, calculated by dividing the environmental gradient into a number of equal spaced segments (10 by default) and calculating the average bias for each segment. The maximum bias is maximum of these 10 values and quantifies the tendency for the model to over- or under-estimate at particular part of the gradient (ter Braak & Juggins 1993).

If the transfer function object has been cross-validated, (ie. is the output of `crossval`), the list returned by `performance` also contains a matrix named `crossval`, which contains the above statistics calculated for the cross-validation predictions.

Function `crossval` returns an object of the original class and adds the following named elements:

<code>predicted</code>	predicted values of each training set sample under cross-validation.
<code>residuals.cv</code>	prediction residuals.

Function `rand.t.test` is a generic function that performs a randomisation t-test to test the significance of a cross-validated model, after van der Voet (1994). Methods exist for [WA](#), [WAPLS](#) and [IKFA](#).

## Author(s)

Steve Juggins

## References

- Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C., & ter Braak, C.J.F. (1990) Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London*, **B**, **327**, 263-278.
- Juggins, S., & Birks, HJB. (2010) Environmental Reconstructions. In Birks et al. (eds) *Tracking Environmental Change using Lake Sediments: Data Handling and Statistical Techniques.*, Kluwer Academic Publishers.
- van der Voet, H. (1994) Comparing the predictive accuracy of models using a simple randomization test. *Chemometrics and Intelligent Laboratory Systems*, **25**, 313-323.

---

read.C2Model	<i>Imports results from a C2 model into R</i>
--------------	---

---

### Description

Imports data from a C2 model saved in Excel format into R for further processing.

### Usage

```
read.C2Model(fName)

## S3 method for class 'C2'
print(x, ...)

## S3 method for class 'C2'
summary(object, ...)
```

### Arguments

fName	filename to read.
x, object	object of class C2 produced by read.C2Model.
...	additional arguments to plot and print.

### Details

Function to import data from a C2 model into R for further processing. C2 is a Windows program for the analysis and visualisation of palaeoecological data (Juggins (2007)). On Windows read.C2Model uses the package RODBC to import data from the Excel spreadsheet whereas on Linux and MacOS X the function uses read.xls in the package gdata.

### Value

Returns a list with elements corresponding to the worksheets in the saved C2 model: these are named after the original worksheets under Windows but "Sheet1", "Sheet2", etc. on Linux and MacOS X.

### Author(s)

Steve Juggins

### Source

Juggins. S. (2007) *C2 Version 1.5 User guide. Software for ecological and palaeoecological data analysis and visualisation*. Newcastle University, Newcastle upon Tyne, UK. 73pp.

### See Also

[read.CEP](#).

**Examples**

```
## Not run:
pth <- system.file("example.datasets/C2_IK_Model.xls", package="rioja")
C2.IK <- read.C2Model(pth)
summary(C2.IK)

## End(Not run)
```

read.CEP

*Reads and writes data in Cornell Ecology Program (CEP) format***Description**

Reads and writes data in Cornell Ecology Program (CEP) format used by CANOCO and other programs.

**Usage**

```
read.CEP(fName, mValue=-99.9, impZero=0)

write.CEP(x, fName, fTitle=fName, type = "condensed",
          nSig=5, nCouplets=4, mValue=-99.9)
```

**Arguments**

fName	filename to read.
mValue	missing value code in the CEP file, converted to NA on import.
impZero	value to assign to "missing" taxa in a condensed CEP file. These are usually implicitly zero and the default is to read them as such.
x	data frame to save in CEP format.
fTitle	title for header in CEP file.
type	format of CEP file. Should be an unambiguous abbreviation of either condensed or full. condensed is usually used for species data and full for environmental data
nSig	number of significant digits to include in output.
nCouplets	number of couplets per line to include in condensed file.

**Details**

read.CEP and write.CEP are functions to read and write data in the Cornell Ecology Program (CEP) format. CEP *condensed* format was originally designed by Mark Hill as an efficient way of storing species abundance data for use with his DECORANA and TWINSPAN programs (Hill 1979a, b), and subsequently adopted by CANOCO (ter Braak & Smilauer, 1998) where *full* and *free* format variations were added. read.CEP and write.CEP can read/write data in condensed and full

formats. The CEP in `read.CEP` is capitalised to distinguish it from a similar function (`read.cep`) in the `vegan` package, which can also import data in CEP *free*

`write.CEP` takes the site and species names from the `rownames` and `colnames` of the input data frame. Names in CEP format are limited to a maximum of 8 characters - if they exceed this limit they will be abbreviated using the function `make.cepnames` in the package `vegan` and a warning issued. The function invisibly returns a list of original and saved names.

### Value

`read.CEP` returns a data frame with species as columns and sites as rows. Columns with zero totals (ie. species with no occurrences) are deleted. Column and row names are taken from the species and site names in the CEP file: invalid characters in the names (including spaces) are replaced by "." and names are made unique using `make.names`.

`write.CEP` returns a list with two named elements, `site.names` and `sp.names`, each character matrix of two columns giving the *original* and *saved* site or species names.

### Note

The code uses a combination of C and C++, with several functions converted to C from FORTRAN using `f2c`. It works on 32 bit systems but currently generates an error on some 64-bit systems. Use `read.cep` in the package `vegan` as an alternative.

### Author(s)

Steve Juggins

### References

Hill, M.O. (1979a). DECORANA - A FORTRAN program for detrended correspondence analysis and reciprocal averaging. Cornell University, Ithaca, New York. Program manual..

Hill, M.O. (1979b). TWINSpan - A FORTRAN program for arranging multivariate data in an ordered two-way table by classification of the individuals and attributes. Cornell University, Ithaca, New York. Program manual. 90pp.

ter Braak, C.J.F. & Smilauer, P. (1998) CANOCO Reference Manual and User's Guide to CANOCO for Windows: Software for Canonical Community Ordination (version 4) Microcomputer Power, Ithaca, NY, USA.

### See Also

[read.cep](#).

### Examples

```
## Not run:
pth <- system.file("example.datasets/RLGHLongCore.cep", package="rioja")
rlgh <- read.CEP(pth)
depth <- as.numeric(gsub("r_", "", rownames(rlgh)))
mx <- apply(rlgh, 2, max)
rlgh.sub <- rlgh[, mx > 5]
```

```
strat.plot(rlgh.sub, scale.percent=TRUE, yvar=depth, y.rev=TRUE,
wa.order="bottomleft")

## End(Not run)
```

---

read.Tilia	<i>Read data in Tilia format</i>
------------	----------------------------------

---

### Description

Reads data in Tilia format used by the Tilia program for plotting stratigraphic diagrams.

### Usage

```
read.Tilia(fName)
```

### Arguments

fName	filename to read.
-------	-------------------

### Details

read.Tilia reads data in the Tilia format. Tilia is a program written by Eric Grimm to plot stratigraphic diagrams and is popular with palynologists. Note that any spaces or other illegal characters in the species codes will be converted to decimal places on import.

### Value

Returns a list with three names elements:

data	data frame with species as columns and sites as rows. Column and row names are taken from the Tilia file.
vars	names and types of each variable in the dataset.
levels	names depths, and optionally ages of each level in the core.

### Note

The code uses a combination of C and C++ and needs 1-byte structure alignment to read the Tilia binary file. It works on 32 bit systems but currently generates an error on some 64-bit Linux systems.

### Author(s)

Steve Juggins

### See Also

[read.CEP](#).

## Examples

```
## Not run:
pth <- system.file("example.datasets/WOLSFELD.TIL", package="rioja")
WOLS <- read.Tilia(pth)
sel <- WOLS$vars$Sums == "A" | WOLS$vars$Sums == "B"
spec <- WOLS$data[, sel]
totals <- apply(spec, 1, sum)
spec.pc <- spec / totals * 100
mx <- apply(spec.pc, 2, max)
spec.sub <- spec.pc[, mx > 5]
age <- WOLS$levels$Chron2
strat.plot(spec.sub, scale.percent=TRUE, yvar=age, y.rev=TRUE,
           wa.order="bottomleft")

## End(Not run)
```

---

 RLGH

*Diatom stratigraphic data from the Round Loch of Glenhead, Galloway, Southwest Scotland*

---

## Description

Diatom stratigraphic data from the Round Loch of Glenhead, Galloway, Southwest Scotland from core K05, first published in Allott et al. (1992) and re-analysed in Juggins et al. (1996) and Battarbee et al. (2005). Data are relative abundances (percentages) of a subset of 41 diatom taxa in 20 samples, and includes all taxa with a maximum abundance of 1 percent in any core sample. Dataset is a list with the following named elements: spec diatom relative abundances, depth associated sediment core depths and 210Pb ages. Column names in RLGH\$spec are short, 6-character alphanumeric codes for each diatom taxon. RLGH\$names contains the full names for each taxon, in the correct order). Note that some rare and low abundance taxa have been removed so the percentages do not sum to 100.

## Usage

```
data(RLGH)
```

## References

- Battarbee, R.W., Monteith, D.T., Juggins, S. Evans, C.D., Jenkins, A. & Simpson, G.L. (2005) Reconstructing pre-acidification pH for an acidified Scottish loch: A comparison of palaeolimnological and modelling approaches. *Environmental Pollution*, **137**, 135-149.
- Allott, T.E.H., Harriman, R., & Battarbee, R.W. (1992) Reversibility of acidification at the Round Loch of Glenhead, Galloway, Scotland. *Environmental Pollution*, **77**, 219-225.
- Juggins, S., Flower, R., & Battarbee, R. (1996) Palaeolimnological evidence for recent chemical and biological changes in UK Acid Waters Monitoring Network sites. *Freshwater Biology*, **36**, 203-219.

**Examples**

```
data(RLGH)
names(RLGH$spec)
names(RLGH$depth)
```

---

sp.plot

*Plots species distributions along an environmental gradient*


---

**Description**

Plots a matrix of species distributions along a single environmental gradient using lattice.

**Usage**

```
## Default S3 method:
sp.plot(y, x, n.cut=5, sort.vars=c("original", "wa",
  "alphabetical"), subset=NULL, sp.scale=c("fixed", "free",
  "exaggerated"), xlab=NULL, ylab=NULL, sp.max=NULL,
  cex.max=NULL, as.table=TRUE, ...)

## S3 method for class 'formula'
sp.plot(formula, data=NULL, n.cut=5,
  sort.vars=c("original", "wa", "alphabetical"), subset=NULL,
  sp.scale=c("fixed", "free", "exaggerated"), xlab=NULL,
  ylab=NULL, sp.max=NULL, cex.max=10, as.table=TRUE, ...)
```

**Arguments**

y	data frame or matrix of species abundances.
x	vector of environmental values, or a matrix or data.frame with one or two columns (see details).
formula	model formula, where the left hand side gives a data frame or matrix of species' abundances and the right hand side gives one or two environmental variables.
data	data frame containing the variables on the right-hand side of the formula.
n.cut	cut.off for species occurrences. Species with fewer than n.cut occurrences will be omitted from plot.
sort.vars	how to order variables in the plot).
subset	a logical vector or list of column indices to include in plot.
sp.scale	how to scale species abundances in the plot (see details).
xlab, ylab	labels for the x- and y-axes.
sp.max	maximum value for scaling abundances (see details).
cex.max	maximum symbol size for plotting abundances (see details).
as.table	order of plots on page. Default TRUE strat top-left. Set to FALSE to start bottom left.
...	additional arguments to xyplot.

## Details

Function `sp.plot` plots all or a subset of variables in a data frame (`y`) (usually species abundances, with a column for each species) against one or two environmental gradients given in `x`. If `x` contains a single environmental variable, either as a vector or a matrix / data frame with one column, a matrix of scatterplots is produced, with species abundance on the y-axis and the environmental values on the x-axis. If `x` is a matrix or data frame containing two columns, a matrix of plots is produced with the first variable on the x-axis and the second on the y-axis, and symbols scaled according to the species' abundances.

Individual plots may be scaled in one of three ways: "fixed" uses a common scaling for species abundances for all plots; "exaggerated" uses a common scaling, but any species with a maximum abundance of less than a tenth of the overall maximum is exaggerated by 10, and the text "x10" indicated in the plot; free uses a separate scaling for each species, and displays the maximum value if plotting against two environmental variables.

`cex.max` sets the maximum symbol size for the plots. `sp.max` sets the maximum species value that will be set to `cex.max`. By default this will be the overall species maximum but this can be overridden to provide a uniform scaling: for example, with percentage data, setting `cex.max` to 5 and `sp.max` to 100 will scale points proportionately so 100 percent gets a size of 5.

The order of variables in the plot may be changed using `sort.vars` - `sort.vars="wa"` sorts variables according to their wa optima along the first environmental variable.

The number of rows and columns in the plot may be controlled using the `layout` argument passed to `xyplot`. For example `layout=c(10,1)` will produce a plot with 1 row and 10 columns. Additional arguments may be passed to the underlying `lattice` routines: for example `col`, `pch` and `fill` to control the symbol size and colour, and `between=list(y=0.5, x=0.5)` to add a space between figures.

The function uses `lattice` to produce the plots so if you call it from a loop or using `source` you will need to wrap it in a `print` or `plot` statement.

## Value

Function `sp.plot` returns an object of class `trellis`.

## Author(s)

Steve Juggins

## See Also

[lattice](#), [xyplot](#), and [inkspot](#) for a more compact way to visualise multiple species distributions along a single gradient.

## Examples

```
## Not run:
data(IK)
spec <- IK$spec
env <- IK$env

#basic scatter plot vs SumSST
```

```

sp.plot(spec, env$SumSST)

# exaggerate less abundant taxa
sp.plot(spec, env$SumSST, sp.scale="exag")

# separate y-scale for each
sp.plot(spec, env$SumSST, sp.scale="free")

# order by SumSST preference
sp.plot(spec, env$SumSST, sp.scale="free", sort.vars="wa")

# add a y-space
sp.plot(spec, env$SumSST, sp.scale="free", sort.vars="wa",
between=list(y=0.5))

# two environmental variables
sp.plot(spec, env[, c(1, 3)])

# exaggerate less abundant taxa
sp.plot(spec, env[, c(1, 3)], sp.scale="exag")

# separate y-scale for each
sp.plot(spec, env[, c(1, 3)], sp.scale="free")

# same using the formula interface
sp.plot(spec ~ SumSST + Salinity, data=env, sp.scale="free")

# Exaggerated scaling using transparent colours
sp.plot(spec ~ SumSST + Salinity, data=env, sp.scale="exag",
col="salmon", pch=21, fill=rgb(0, 0, 1, alpha=0.3), sp.max=100)

# Change the layout (3 * 3 matrix)
# and start plots bottom left (as.table=FALSE
sp.plot(spec ~ SumSST + Salinity, data=env, sp.scale="exag",
col="salmon", pch=21, fill=rgb(0, 0, 1, alpha=0.3), sp.max=100,
layout=c(3,3), as.table=FALSE)

## End(Not run)

```

---

strat.plot

*Plot a stratigraphic diagram*


---

### Description

Plots a diagram of multiple biological, physical or chemical parameters against depth or time, as used in geology & palaeoecology.

### Usage

```
strat.plot (d, yvar = NULL, scale.percent = FALSE,
```

```

graph.widths=1, minmax=NULL,
scale.minmax = TRUE, xLeft = 0.07, xRight = 1,
yBottom = 0.07, yTop = 0.8, title = "", cex.title=1.8,
y.axis=TRUE, min.width = 5, ylim = NULL, y.rev = FALSE,
y.tks=NULL, ylabel = "", cex.ylabel=1, cex.yaxis=1,
xSpace = 0.01, wa.order = "none", plot.line = TRUE,
col.line = "black", lwd.line = 1, plot.bar = TRUE,
lwd.bar = 1, col.bar = "grey", sep.bar = FALSE,
plot.poly = FALSE, col.poly = "grey", col.poly.line = "black",
lwd.poly = 1, x.names=NULL, cex.xlabel = 1.1, srt.xlabel=90,
mgp=c(3, .5, 0), cex.axis=.8, clust = NULL, clust.width=0.1,
orig.fig=NULL, add=FALSE, ...)

```

```
addZone (x, upper, lower=NULL, ...)
```

```
addClustZone(x, clust, nZone, ...)
```

### Arguments

d	a matrix or data frame of variables to plot.
yvar	a vector of depths or ages to use for the y-axis (defaults to sample number).
scale.percent	logical to scale x-axes for (biological) percentage data.
graph.widths	a vector of relative widths for each curve, used if scale.percent=FALSE.
minmax	2 * nvar matrix of min and max values to scale each curve if scale.percent=FALSE.
scale.minmax	logical to show only min and max values on x-axes (to avoid label crowding).
xLeft, xRight, yBottom, yTop	x, y position of plot on page, in relative units.
title	main title for plot.
x.names	character vector of names for each graph, of same length as ncol(d).
cex.title	size of label for title.
y.axis	logical to control drawing to left-hand y-axis scale. Defaults to TRUE.
min.width	minimum upper value of x-axis when scaled for percent data.
ylim	numeric vector of 2 values to control limit of y-axis. Defaults to data range.
y.rev	logical to reverse y-axis. Defaults to FALSE.
y.tks	numerical vector listing values of y-axis ticks / labels.
ylabel	label for y-axis.
cex.ylabel, cex.yaxis	text size for y-axis labels and values.
xSpace	space between graphs, in relative units.
wa.order	"none", "topleft" or "bottomleft", to sort variables according to the weighted average with y.
plot.line, plot.poly, plot.bar	logical flags to plot graphs as lines, silhouettes, or bars.

<code>col.line, col.poly.line</code>	colour of lines and silhouette outlines. Can be a single colour or a vector of colours, one for each graph.
<code>col.poly</code>	silhouette fill colour. Can be a single colour or a vector of colours, one for each graph.
<code>lwd.line, lwd.poly, lwd.bar</code>	line widths for line, silhouette or bar graphs.
<code>col.bar</code>	colour of bars in a bar graph. <code>col.bar</code> can be a vector to specify colours of individual bars or graphs.
<code>sep.bar</code>	If true, colours in <code>col.bar</code> are applied to individual bars, otherwise individual graphs.
<code>cex.xlabel</code>	size of label for variable names.
<code>srt.xlabel</code>	rotation angle for variable names.
<code>mgp</code>	value of <code>mgp</code> for x-axes. See <code>par</code> for details.
<code>cex.axis</code>	text size for x-axis labels. See <code>par</code> for details.
<code>clust</code>	an constrained classification object of class <code>chclust</code> to add to plot.
<code>clust.width</code>	width of dendrogram to add to right of plot, in relative units.
<code>orig.fig</code>	<code>fig</code> values to specify area of window in which to place diagram. See <code>par</code> for details. Defaults to whole window.
<code>add</code>	logical to control drawing of new page. See <code>par</code> for details. Defaults to FALSE in which a call to <code>strat.plot</code> will start a new diagram. Set to TRUE to add a diagram to an existing plot.
<code>x</code>	a stratigraphic diagram object produced by <code>strat.plot</code> .
<code>upper, lower</code>	upper and (optional) lower limits of a zone to add to an existing stratigraphic diagram.
<code>nZone</code>	number of zones to draw.
<code>...</code>	further graphical arguments.

### Details

`strat.plot` plots a series of variables in a stratigraphic diagram. Diagrams can be plotted as line graphs and / or bar charts. Samples are plotted on the y-axis by sample number by default but may be plotted against sample age or depth by specifying a variable for `yvar`. Margins of the plotting area can be changed using `xLeft`, `xRight`, `yBottom` and `yTop`. A dendrogram produced by `chclust` can be added to the right of the diagram.

The function `addZone` can be used to add a horizontal line or box to an existing plot, and `addClustZone` will add a specified number of zones from a dendrogram (see examples).

The function uses `fig` to split the screen and may be incompatible with `par(mfrow)` and `split.screen`.

### Value

Returns a list containing the following objects:

<code>box</code>	Vector of 4 values giving the coordinates of the left, right, bottom and top of the plotting area, in relative units.
------------------	---

usr	Ranges of the plotting area, in data units.
yvar	Variable used for the y-axis.
ylim	Limits of the y-axis.

**Author(s)**

Steve Juggins

**See Also**[chclust](#).**Examples**

```
library(vegan) ## decorana
data(RLGH)
## Not run:
# create appropriately sized graphics window
windows(width=12, height=7) # quartz() on Mac, X11 on linux

## End(Not run)
# remove less abundant taxa
mx <- apply(RLGH$spec, 2, max)
spec <- RLGH$spec[, mx > 3]
depth <- RLGH$depth$Depth
#basic stratigraphic plot
strat.plot(spec, y.rev=TRUE)
#scale for percentage data
strat.plot(spec, y.rev=TRUE, scale.percent=TRUE)
# plot by sample depth
strat.plot(spec, yvar = depth, y.rev=TRUE, scale.percent=TRUE,
title="Round Loch of Glenhead", ylabel="Depth (cm)")
# add a dendrogram from constrained cluster analysis
diss <- dist(sqrt(RLGH$spec/100)^2)
clust <- chclust(diss, method="coniss")
# broken stick model suggest 3 significant zones
bstick(clust)
x <- strat.plot(spec, yvar = depth, y.rev=TRUE,
scale.percent=TRUE, title="Round Loch of Glenhead", ylabel="Depth (cm)",
clust=clust)
# add zones
addClustZone(x, clust, 3, col="red")
# use fig to control diagram size and position
x <- strat.plot(spec, xRight = 0.7, yvar = depth, y.rev=TRUE,
scale.percent=TRUE, title="Round Loch of Glenhead", ylabel="Depth (cm)")
# add curves for first two DCA components of diatom data
dca <- decorana(spec, iweigh=1)
sc <- scores(dca, display="sites", choices=1:2)
strat.plot(sc, xLeft = 0.7, yvar = depth, y.rev=TRUE, xRight=0.99,
y.axis=FALSE, clust=clust, clust.width=0.08, add=TRUE)
```

---

strat.plot.simple      *Plot a simple stratigraphic diagram*

---

### Description

Plots a simple stratigraphic diagram using `lattice`. The function is most useful for visualising the effect of interpolating and / or smoothing a dataset.

### Usage

```
strat.plot.simple(y1, x1, y2=NULL, x2=NULL, col=c("blue", "red"),
  sort.vars=c("original", "wa", "alphabetical"),
  ylim=range(x1), y.rev=FALSE, type=c("b", "l"),
  subset=c(1:ncol(y1)), ...)
```

### Arguments

<code>y1, y2, x1, x2</code>	data frames to be plotted and corresponding x-values. If <code>y2</code> and <code>x2</code> are specified columns of <code>y2</code> will be plotted overlain on those of <code>y1</code> .
<code>col, type</code>	colours and line types for the plots.
<code>sort.vars</code>	how to order variables in the plot).
<code>ylim</code>	limits for the y-axis.
<code>y.rev</code>	logical to reverse y-axis (which is actually the x-variable).
<code>subset</code>	a vector of column indices or a logical vector giving a subset of columns.
<code>...</code>	additional arguments to <code>xypLOT</code> .

### Details

`strat.plot.simple` plots all or a subset of variables in a data frame (`y`) against a single secondary variable (`x`). The function is appropriate for plotting stratigraphic data, with the x-variable is plotted on the y-axis and variables in `y` arrayed across the page. The function is primarily intended to visualise the effect of smoothing by plotting the original data with the interpolated or smoothed version superimposed.

The number of rows and columns in the plot may be controlled using the layout argument passed to `link{xypLOT}`. For example `layout=c(10,1)` will produce a plot with 1 row and 10 columns.

By default, the x-axis has the same scale for all variables. To allow independent x-scales for each variable include the additional argument: `scales=list(x="free")`.

The function uses `lattice` to produce the plots so if you call it from a loop or using source you will need to wrap it in a `print` or `plot` statement.

### Value

Function `strat.plot.simple` returns an object of class `link{trellis}`.

**Author(s)**

Steve Juggins

**See Also**

[strat.plot](#) for better looking stratigraphics diagrams and [interp.dataset](#) for interpolation and / or smoothing.

**Examples**

```
data(RLGH)
spec <- RLGH$spec
depth <- RLGH$depth$Depth

# interpolate new dataset to every 0.5 cm
# using default method (linear)
x.new <- seq(0, 20, by=0.5)
sp.interp <- interp.dataset(y=spec, x=depth, xout=x.new)
## Not run:
# examine the results and compare to original data
strat.plot.simple(spec, depth, sp.interp, x.new)

## End(Not run)
```

---

 SWAP

---

*SWAP surface sediment diatom data and lake-water pH.*


---

**Description**

SWAP (Surface Water Acidification Programme) surface sediment diatom data from Birks et al. (1990) and Stevenson et al. (1990). Dataset is a list with the following named elements: (spec) diatom relative abundances for 277 taxa in 167 surface samples, (pH) associated lake-water pH. Column names in spec are short, 6-character alphanumeric codes for each diatom taxon. SWAP\$names contains the full names for each taxon, in the correct order).

**Usage**

```
data(SWAP)
```

**Source**

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

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., & Renberg, I. (1991) *The Surface Waters Acidification Project Palaeolimnology Programme: Modern Diatom / Lake-Water Chemistry Data-Set* ENSIS Ltd, London.

**Examples**

```
data(SWAP)
names(SWAP$spec)
hist(SWAP$pH)
```

---

 utils

*Utility functions.*


---

**Description**

Utility functions to perform simple computations, transformations, formatting etc.

**Usage**

```
make.dummy(fact)

dummy2factor(x)

Hill.N2(df, margin=2)

site.summ(y, max.cut=c(2, 5, 10, 20))

sp.summ(y, n.cut=c(5, 10, 20))

write.list.Excel(x, fName)
```

**Arguments**

fact	a factor to convert to a matrix of dummy variables.
x	a matrix or data frame of dummy variables to convert to a factor, or a list to save as an Excel file.
df	a data frame of species abundance data.
margin	margin to calculate over: 1 = by rows, 2 = by columns.
fName	file name for Excel file.
y	data frame or matrix of species by sites data.
n.cut	cut levels of abundance for species summary (see below).
max.cut	cut levels of occurrence for species summary.

## Details

Function `make.dummy` converts a factor into a matrix of dummy (1/0) variables. `dummy2factor` converts a matrix or data frame of dummy variables into a factor.

Function `Hill.N2` returns Hill's N2 values for species or samples for a given species by sites dataset (Hill 1973).

Function `write.list.Excel` saves the data frames in a list object as a series of names worksheets in Excel format. Not that this function is only available on Windows. See package `WriteXLS` for a Linux / MacOS X alternative.

## Value

`make.dummy` returns a matrix of dummy variables. `dummy2factor` returns a factor.

`Hill.N2` returns a numeric vector of N2 values.

`sp.summ` returns a matrix with columns for the number of occurrences, Hill's N2 and maximum abundance of each species, and the number of occurrences at abundance greater than the cut levels given in `n.cut`.

`sam.summ` returns a matrix with columns for the number of taxa, Hill's N2, maximum value and site total of each site (sample), and the number of taxa in each site with abundance greater than the cut levels given in `max.cut`.

## Author(s)

Steve Juggins

## References

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

---

WA

*Weighted averaging (WA) regression and calibration*

---

## Description

Functions for reconstructing (predicting) environmental values from biological assemblages using weighted averaging (WA) regression and calibration.

## Usage

```
WA(y, x, mono=FALSE, tolDW = FALSE, use.N2=TRUE, tol.cut=.01,  
   check.data=TRUE, lean=FALSE)
```

```
WA.fit(y, x, mono=FALSE, tolDW=FALSE, use.N2=TRUE, tol.cut=.01,  
       lean=FALSE)
```

```

## S3 method for class 'WA'
predict(object, newdata=NULL, sse=FALSE, nboot=100,
        match.data=TRUE, verbose=TRUE, ...)

## S3 method for class 'WA'
crossval(object, cv.method="loo", verbose=TRUE,
        ngroups=10, nboot=100, ...)

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

## S3 method for class 'WA'
rand.t.test(object, n.perm=999, ...)

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

## S3 method for class 'WA'
summary(object, full=FALSE, ...)

## S3 method for class 'WA'
plot(x, resid=FALSE, xval=FALSE, tolDW=FALSE, deshrink="inverse",
     xlab="", ylab="", ylim=NULL, xlim=NULL, add.ref=TRUE,
     add.smooth=FALSE, ...)

## S3 method for class 'WA'
residuals(object, cv=FALSE, ...)

## S3 method for class 'WA'
coef(object, ...)

## S3 method for class 'WA'
fitted(object, ...)

```

### Arguments

<code>y</code>	a data frame or matrix of biological abundance data.
<code>x</code> , <code>object</code>	a vector of environmental values to be modelled or an object of class WA.
<code>newdata</code>	new biological data to be predicted.
<code>mono</code>	logical to perform monotonic curvilinear deshrinking.
<code>tolDW</code>	logical to include regressions and predictions using tolerance downweighting.
<code>use.N2</code>	logical to adjust tolerance by species N2 values.
<code>tol.cut</code>	tolerances less than <code>tol.cut</code> are replaced by the mean tolerance.
<code>check.data</code>	logical to perform simple checks on the input data.
<code>lean</code>	logical to exclude some output from the resulting models (used when cross-validating to speed calculations).

<code>full</code>	logical to show head and tail of output in summaries.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to FALSE if you are sure the column names match exactly.
<code>resid</code>	logical to plot residuals instead of fitted values.
<code>xval</code>	logical to plot cross-validation estimates.
<code>xlab, ylab, xlim, ylim</code>	additional graphical arguments to <code>plot.WA</code> .
<code>deshrink</code>	deshrinking type to show in plot.
<code>add.ref</code>	add 1:1 line on plot.
<code>add.smooth</code>	add loess smooth to plot.
<code>cv.method</code>	cross-validation method, either "loo", "lgo" or "bootstrap".
<code>verbose</code>	logical or integer to show feedback during cross-validation. If TRUE print feedback every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation.
<code>sse</code>	logical indicating that sample specific errors should be calculated.
<code>n.perm</code>	number of permutations for randomisation t-test.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

## Details

Function `WA` performs weighted average (WA) regression and calibration. Weighted averaging has a long history in ecology and forms the basis of many biotic indices. It was popularised in palaeolimnology by ter Braak and van Dam (1989) and Birks et al. (1990) following ter Braak & Barendregt (1986) and ter Braak and Looman (1986) who demonstrated its theoretical properties in providing a robust and simple alternative to species response modelling using Gaussian logistic regression. Function `WA` predicts environmental values from sub-fossil biological assemblages, given a training dataset of modern species and environmental data. It calculates estimates using inverse and classical deshrinking, and, optionally, with taxa downweighted by their tolerances. Prediction errors and model complexity (simple or tolerance downweighted WA) can be estimated by cross-validation using `crossval` which implements leave-one out, leave-group-out, or bootstrapping. With leave-group out one may also supply a vector of group memberships for more carefully designed cross-validation experiments.

Function `predict` predicts values of the environmental variable for `newdata` or returns the fitted (predicted) values from the original modern dataset if `newdata` is NULL. Variables are matched between training and `newdata` by column name (if `match.data` is TRUE). Use [compare.datasets](#) to assess conformity of two species datasets and identify possible no-analogue samples.

Function `rand.t.test` performs a randomisation t-test to test the significance of the difference in cross-validation RMSE between tolerance-downweighted and simple WA, after van der Voet (1994).

`WA` has methods `fitted` and `residuals` that return the fitted values (estimates) and residuals for the training set, `performance`, which returns summary performance statistics (see below), `coef`

which returns the species coefficients (optima and tolerances), and `print` and `summary` to summarise the output. WA also has a `plot` method that produces scatter plots of predicted vs observed measurements for the training set.

### Value

Function WA returns an object of class WA with the following named elements:

`coefficients` species coefficients ("optima" and, optionally, "tolerances").  
`deshrink.coefficients` deshrinking coefficients.  
`tolDW` logical to indicate tolerance downweighted results in model.  
`fitted.values` fitted values for the training set.  
`call` original function call.  
`x` environmental variable used in the model.

If function `predict` is called with `newdata=NULL` it returns the fitted values of the original model, otherwise it returns a list with the following named elements:

`fit` predicted values for `newdata`.

If sample specific errors were requested the list will also include:

`fit.boot` mean of the bootstrap estimates of `newdata`.  
`v1` squared standard error of the bootstrap estimates for each new sample.  
`v2` mean squared error for the training set samples, across all bootstrap samples.  
`SEP` standard error of prediction, calculated as the square root of `v1 + v2`.

Function `crossval` also returns an object of class WA and adds the following named elements:

`predicted` predicted values of each training set sample under cross-validation.  
`residuals.cv` prediction residuals.

Function `performance` returns a matrix of performance statistics for the WA model. See [performance](#), for a description of the summary.

### Author(s)

Steve Juggins

### References

- Birks, H.J.B., Line, J.M., Juggins, S., Stevenson, A.C., & ter Braak, C.J.F. (1990) Diatoms and pH reconstruction. *Philosophical Transactions of the Royal Society of London*, **B**, 327, 263-278.
- ter Braak, C.J.F. & Barendregt, L.G. (1986) Weighted averaging of species indicator values: its efficiency in environmental calibration. *Mathematical Biosciences*, 78, 57-72.
- ter Braak, C.J.F. & Looman, C.W.N. (1986) Weighted averaging, logistic regression and the Gaussian response model. *Vegetatio*, **65**, 3-11.

ter Braak, C.J.F. & van Dam, H. (1989) Inferring pH from diatoms: a comparison of old and new calibration methods. *Hydrobiologia*, **178**, 209-223.

van der Voet, H. (1994) Comparing the predictive accuracy of models using a simple randomization test. *Chemometrics and Intelligent Laboratory Systems*, **25**, 313-323.

### See Also

[WAPLS](#), [MAT](#), and [compare.datasets](#) for diagnostics.

### Examples

```
# pH reconstruction of core K05 from the Round Loch of Glenhead,
# Galloway, SW Scotland. This lake has become acidified over the
# last c. 150 years

data(SWAP)
data(RLGH)
spec <- SWAP$spec
pH <- SWAP$pH
core <- RLGH$spec
age <- RLGH$depth$Age

fit <- WA(spec, pH, toldW=TRUE)
# plot predicted vs. observed
plot(fit)
plot(fit, resid=TRUE)

# RLGH reconstruction
pred <- predict(fit, core)

#plot the reconstructio
plot(age, pred$fit[, 1], type="b")

# cross-validation model using bootstrapping
## Not run:
fit.xv <- crossval(fit, cv.method="boot", nboot=1000)
par(mfrow=c(1,2))
plot(fit)
plot(fit, resid=TRUE)
plot(fit.xv, xval=TRUE)
plot(fit.xv, xval=TRUE, resid=TRUE)

# RLGH reconstruction with sample specific errors
pred <- predict(fit, core, sse=TRUE, nboot=1000)

## End(Not run)
```

---

WAPLS	<i>Weighted averaging partial least squares (WAPLS) regression and calibration</i>
-------	--

---

### Description

Functions for reconstructing (predicting) environmental values from biological assemblages using weighted averaging partial least squares (WAPLS) regression and calibration.

### Usage

```

WAPLS(y, x, npls=5, iswapls=TRUE, standx=FALSE, lean=FALSE,
      check.data=TRUE, ...)

WAPLS.fit(y, x, npls=5, iswapls=TRUE, standx=FALSE, lean=FALSE)

## S3 method for class 'WAPLS'
predict(object, newdata=NULL, sse=FALSE, nboot=100,
       match.data=TRUE, verbose=TRUE, ...)

## S3 method for class 'WAPLS'
crossval(object, cv.method="loo", verbose=TRUE, ngroups=10,
        nboot=100, ...)

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

## S3 method for class 'WAPLS'
rand.t.test(object, n.perm=999, ...)

## S3 method for class 'WAPLS'
screplot(x, rand.test=TRUE, ...)

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

## S3 method for class 'WAPLS'
summary(object, full=FALSE, ...)

## S3 method for class 'WAPLS'
plot(x, resid=FALSE, xval=FALSE, npls=1,
     xlab="", ylab="", ylim=NULL, xlim=NULL, add.ref=TRUE,
     add.smooth=FALSE, ...)

## S3 method for class 'WAPLS'
residuals(object, cv=FALSE, ...)

```

```
## S3 method for class 'WAPLS'
coef(object, ...)
```

```
## S3 method for class 'WAPLS'
fitted(object, ...)
```

### Arguments

<code>y</code>	a data frame or matrix of biological abundance data.
<code>x, object</code>	a vector of environmental values to be modelled or an object of class <code>wa</code> .
<code>newdata</code>	new biological data to be predicted.
<code>iswapls</code>	logical logical to perform WAPLS or PLS. Defaults to TRUE = WAPLS.
<code>standx</code>	logical to standardise x-data in PLS, defaults to FALSE.
<code>npls</code>	number of pls components to extract.
<code>check.data</code>	logical to perform simple checks on the input data.
<code>match.data</code>	logical indicate the function will match two species datasets by their column names. You should only set this to FALSE if you are sure the column names match exactly.
<code>lean</code>	logical to exclude some output from the resulting models (used when cross-validating to speed calculations).
<code>full</code>	logical to show head and tail of output in summaries.
<code>resid</code>	logical to plot residuals instead of fitted values.
<code>xval</code>	logical to plot cross-validation estimates.
<code>xlab, ylab, xlim, ylim</code>	additional graphical arguments to <code>plot.wa</code> .
<code>add.ref</code>	add 1:1 line on plot.
<code>add.smooth</code>	add loess smooth to plot.
<code>cv.method</code>	cross-validation method, either "loo", "lgo" or "bootstrap".
<code>verbose</code>	logical or integer to show feedback during cross-validation. If TRUE print feedback every 50 cycles, if integer, use this value.
<code>nboot</code>	number of bootstrap samples.
<code>ngroups</code>	number of groups in leave-group-out cross-validation, or a vector contain leave-out group membership.
<code>sse</code>	logical indicating that sample specific errors should be calculated.
<code>rand.test</code>	logical to perform a randomisation t-test to test significance of cross validated components.
<code>n.perm</code>	number of permutations for randomisation t-test.
<code>cv</code>	logical to indicate model or cross-validation residuals.
<code>...</code>	additional arguments.

## Details

Function `WAPLS` performs partial least squares (PLS) or weighted averaging partial least squares (WAPLS) regression. WAPLS was first described in ter Braak and Juggins (1993) and ter Braak et al. (1993) and has since become popular in palaeolimnology for reconstructing (predicting) environmental values from sub-fossil biological assemblages, given a training dataset of modern species and environmental data. Prediction errors and model complexity (number of components) can be estimated by cross-validation using `crossval` which implements leave-one out, leave-group-out, or bootstrapping. With leave-group out one may also supply a vector of group memberships for more carefully designed cross-validation experiments.

Function `predict` predicts values of the environmental variable for `newdata` or returns the fitted (predicted) values from the original modern dataset if `newdata` is `NULL`. Variables are matched between training and `newdata` by column name (if `match.data` is `TRUE`). Use [compare.datasets](#) to assess conformity of two species datasets and identify possible no-analogue samples.

WAPLS has methods `fitted` and `rediduals` that return the fitted values (estimates) and residuals for the training set, `performance`, which returns summary performance statistics (see below), `coef` which returns the species coefficients, and `print` and `summary` to summarise the output. WAPLS also has a `plot` method that produces scatter plots of predicted vs observed measurements for the training set.

Function `rand.t.test` performs a randomisation t-test to test the significance of the cross-validated components after van der Voet (1994).

Function `screepplot` displays the RMSE of prediction for the training set as a function of the number of components and is useful for estimating the optimal number for use in prediction. By default `screepplot` will also carry out a randomisation t-test and add a line to scree plot indicating percentage change in RMSE with each component annotate with the p-value from the randomisation test.

## Value

Function `WAPLS` returns an object of class `WAPLS` with the following named elements:

<code>coefficients</code>	species coefficients (the updated "optima").
<code>meanY</code>	weighted mean of the environmental variable.
<code>iswapls</code>	logical indicating whether analysis was WAPLS ( <code>TRUE</code> ) or PLS ( <code>FALSE</code> ).
<code>T</code>	sample scores.
<code>P</code>	variable (species) scores.
<code>npls</code>	number of pls components extracted.
<code>fitted.values</code>	fitted values for the training set.
<code>call</code>	original function call.
<code>x</code>	environmental variable used in the model.
<code>standx, meanT, sdx</code>	additional information returned for a PLS model.

Function `crossval` also returns an object of class `WAPLS` and adds the following named elements:

<code>predicted</code>	predicted values of each training set sample under cross-validation.
------------------------	--

`residuals.cv` prediction residuals.

If function `predict` is called with `newdata=NULL` it returns the fitted values of the original model, otherwise it returns a list with the following named elements:

`fit` predicted values for `newdata`.

if sample specific errors were generated the list will also include:

`fit.boot` mean of the bootstrap estimates of `newdata`.

`v1` squared standard error of the bootstrap estimates for each new sample.

`v2` mean squared error for the training set samples, across all bootstrap samples.

`SEP` standard error of prediction, calculated as the square root of `v1 + v2`.

Function `performance` returns a matrix of performance statistics for the WAPLS model. See [performance](#), for a description of the summary.

Function `rand.t.test` returns a matrix of performance statistics together with columns indicating the p-value and percentage change in RMSE with each higher component (see van der Veot (1994) for details).

### Author(s)

Steve Juggins

### References

ter Braak, C.J.F. & Juggins, S. (1993) Weighted averaging partial least squares regression (WAPLS): an improved method for reconstructing environmental variables from species assemblages. *Hydrobiologia*, **269/270**, 485-502.

ter Braak, C.J.F., Juggins, S., Birks, H.J.B., & Voet, H., van der (1993). Weighted averaging partial least squares regression (WAPLS): definition and comparison with other methods for species-environment calibration. In *Multivariate Environmental Statistics* (eds G.P. Patil & C.R. Rao), pp. 525-560. Elsevier Science Publishers.

van der Voet, H. (1994) Comparing the predictive accuracy of models using a simple randomization test. *Chemometrics and Intelligent Laboratory Systems*, **25**, 313-323.

### See Also

[WA](#), [MAT](#), [performance](#), and [compare.datasets](#) for diagnostics.

### Examples

```
data(IK)
spec <- IK$spec
SumSST <- IK$env$SumSST
core <- IK$core

fit <- WAPLS(spec, SumSST)
fit
# cross-validate model
```

```
fit.cv <- crossval(fit, cv.method="loo")
# How many components to use?
rand.t.test(fit.cv)
screplot(fit.cv)

#predict the core
pred <- predict(fit, core, npls=2)

#plot predictions - depths are in rownames
depth <- as.numeric(rownames(core))
plot(depth, pred$fit[, 2], type="b", ylab="Predicted SumSST", las=1)

# predictions with sample specific errors
## Not run:
pred <- predict(fit, core, npls=2, sse=TRUE, nboot=1000)
pred

## End(Not run)
```

# Index

## \*Topic **IO**

- read.C2Model, 31
- read.CEP, 32
- read.Tilia, 34

## \*Topic **aplot**

- gutils, 8

## \*Topic **cluster**

- chclust, 4

## \*Topic **datasets**

- aber, 3
- IK, 9
- Ponds, 28
- RLGH, 35
- SWAP, 43

## \*Topic **file**

- read.C2Model, 31
- read.CEP, 32
- read.Tilia, 34

## \*Topic **hplot**

- chclust, 4
- inkspot, 14
- interp.dataset, 16
- sp.plot, 36
- strat.plot, 38
- strat.plot.simple, 42

## \*Topic **models**

- IKFA, 9
- LWR, 17
- MAT, 19
- MLRC, 25
- WA, 45
- WAPLS, 50

## \*Topic **multivariate**

- IKFA, 9
- LWR, 17
- MAT, 19
- MLRC, 25
- WA, 45
- WAPLS, 50

## \*Topic **package**

- rioja-package, 2

## \*Topic **utilities**

- compare.datasets, 6
- Merge, 23
- PTF, 29
- utils, 44

aber, 3

addClustZone (strat.plot), 38

addZone (strat.plot), 38

approx, 17

bstick, 5

bstick.chclust (chclust), 4

chclust, 4, 41

chull, 8

coef.IKFA (IKFA), 9

coef.LWR (LWR), 17

coef.MLRC (MLRC), 25

coef.WA (WA), 45

coef.WAPLS (WAPLS), 50

communality (IKFA), 9

compare.datasets, 6, 12, 13, 19, 21, 22, 27, 28, 47, 49, 52, 53

crossval (PTF), 29

crossval.IKFA (IKFA), 9

crossval.LWR (LWR), 17

crossval.MAT (MAT), 19

crossval.MLRC (MLRC), 25

crossval.WA (WA), 45

crossval.WAPLS (WAPLS), 50

cutree, 5

dendrogram, 5

dot (utils), 44

dummy2factor (utils), 44

figCnvt (gutils), 8

fitted.IKFA (IKFA), 9

fitted.LWR (LWR), 17  
 fitted.MAT (MAT), 19  
 fitted.MLRC (MLRC), 25  
 fitted.WA (WA), 45  
 fitted.WAPLS (WAPLS), 50  
  
 gutils, 8  
  
 hclust, 5  
 Hill.N2 (utils), 44  
 hulls (gutils), 8  
  
 IK, 9  
 IKFA, 9, 29, 30  
 inkspot, 14, 37  
 interp.dataset, 16, 43  
  
 lattice, 37  
 loess, 17  
 LWR, 17  
  
 make.dummy (utils), 44  
 make.names, 33  
 MAT, 13, 19, 19, 28, 29, 49, 53  
 Merge, 23  
 merge, 24  
 MLRC, 25, 29  
  
 paldist (MAT), 19  
 paldist2 (MAT), 19  
 performance, 13, 22, 27, 28, 48, 53  
 performance (PTF), 29  
 performance.IKFA (IKFA), 9  
 performance.LWR (LWR), 17  
 performance.MAT (MAT), 19  
 performance.MLRC (MLRC), 25  
 performance.WA (WA), 45  
 performance.WAPLS (WAPLS), 50  
 plclust, 5  
 plot.chclust (chclust), 4  
 plot.compare.datasets, 7  
 plot.compare.datasets  
     (compare.datasets), 6  
 plot.IKFA (IKFA), 9  
 plot.LWR (LWR), 17  
 plot.MAT (MAT), 19  
 plot.MLRC (MLRC), 25  
 plot.WA (WA), 45  
 plot.WAPLS (WAPLS), 50  
 Ponds, 28  
  
 predict.IKFA (IKFA), 9  
 predict.LWR (LWR), 17  
 predict.MAT (MAT), 19  
 predict.MLRC (MLRC), 25  
 predict.WA (WA), 45  
 predict.WAPLS (WAPLS), 50  
 print.C2 (read.C2Model), 31  
 print.IKFA (IKFA), 9  
 print.LWR (LWR), 17  
 print.MAT (MAT), 19  
 print.MLRC (MLRC), 25  
 print.WA (WA), 45  
 print.WAPLS (WAPLS), 50  
 PTF, 29  
  
 rand.t.test, 12, 13, 53  
 rand.t.test (PTF), 29  
 rand.t.test.IKFA (IKFA), 9  
 rand.t.test.WA (WA), 45  
 rand.t.test.WAPLS (WAPLS), 50  
 read.C2Model, 31  
 read.CEP, 31, 32, 34  
 read.cep, 33  
 read.Tilia, 34  
 residuals.IKFA (IKFA), 9  
 residuals.LWR (LWR), 17  
 residuals.MAT (MAT), 19  
 residuals.MLRC (MLRC), 25  
 residuals.WA (WA), 45  
 residuals.WAPLS (WAPLS), 50  
 rioja, 29  
 rioja (rioja-package), 2  
 rioja-package, 2  
 RLGH, 35  
  
 screeplot.IKFA (IKFA), 9  
 screeplot.MAT (MAT), 19  
 screeplot.WAPLS (WAPLS), 50  
 site.summ (utils), 44  
 smooth.spline, 17  
 sp.plot, 36  
 sp.summ (utils), 44  
 strat.plot, 38, 43  
 strat.plot.simple, 17, 42  
 summary.C2 (read.C2Model), 31  
 summary.IKFA (IKFA), 9  
 summary.LWR (LWR), 17  
 summary.MAT (MAT), 19  
 summary.MLRC (MLRC), 25

summary.WA (WA), [45](#)  
summary.WAPLS (WAPLS), [50](#)  
SWAP, [43](#)

utils, [44](#)

vegemite, [15](#)

WA, [13](#), [22](#), [28–30](#), [45](#), [53](#)  
WAPLS, [19](#), [22](#), [29](#), [30](#), [49](#), [50](#)  
write.CEP (read.CEP), [32](#)  
write.list.Excel, [7](#)  
write.list.Excel (utils), [44](#)

xyplot, [6](#), [37](#)