Package: easyCODA (via r-universe)

October 27, 2024

Type Package

Version 0.40.2

Date 2024-08-25

Title Compositional Data Analysis in Practice

Author Michael Greenacre [aut, cre]

Imports ca (>= 0.7), vegan (>= 2.3), ellipse (>= 0.4.1)

Maintainer Michael Greenacre <`michael.greenacre@upf.edu`>

Description Univariate and multivariate methods for compositional data analysis, based on logratios. The package implements the approach in the book Compositional Data Analysis in Practice by Michael Greenacre (2018), where accent is given to simple pairwise logratios. Selection can be made of logratios that account for a maximum percentage of logratio variance. Various multivariate analyses of logratios are included in the package.

License GPL

URL `https://github.com/michaelgreenacre/CODAinPractice/`

NeedsCompilation no

Repository/R-Forge/Project easycoda

Repository/R-Forge/Revision 53

Repository/R-Forge/DateTimeStamp 2024-08-27 14:20:20

Date/Publication 2024-08-27 15:00:05 UTC

Depends R (>= 2.10)

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

RemoteUrl https://github.com/cran/easyCODA

RemoteRef HEAD

RemoteSha 3a9ea3953d8a6a7753c2c54298d93ef54558b3d2

Contents

easyCODA-package 2
ACLUST 4
ALR 6
BAR 7
CA 9
CIplot_biv 10
CLOSE 12
CLR 13
cups 14
DOT 14
DUMMY 16
FINDALR 16
fish 18
ILR 18
invALR 19
invCLR 20
invSLR 21
LR 22
LR.VAR 23
LRA 25
PCA 27
PLOT.CA 29
PLOT.LRA 30
PLOT.PCA 31
PLOT.RDA 33
PLR 34
RDA 35
SLR 37
STEP 39
STEPR 41
time 44
VAR 45
veg 46
WARD 46

Index **49**

`easyCODA-package` *Compositional Data Analysis in Practice*

Description

Univariate and multivariate methods for compositional data analysis, based on logratios. The package implements the approach in the book Compositional Data Analysis in Practice by Michael Greenacre (2018), where accent is given to simple pairwise logratios. Selection can be made of logratios that account for a maximum percentage of logratio variance. Various multivariate analyses of logratios are included in the package.

Details

The DESCRIPTION file:

Package:	easyCODA
Type:	Package
Version:	0.40.2
Date:	2024-08-25
Title:	Compositional Data Analysis in Practice
Authors@R:	person(given = "Michael", family = "Greenacre", role = c("aut", "cre"), email = "mich
Author:	Michael Greenacre [aut, cre]
Imports:	ca (>= 0.7), vegan (>= 2.3), ellipse (>= 0.4.1)
Maintainer:	Michael Greenacre <michael.greenacre@upf.edu>
Description:	Univariate and multivariate methods for compositional data analysis, based on logratio
License:	GPL
URL:	https://github.com/michaelgreenacre/CODAinPractice/
NeedsCompilation:	no
Packaged:	2024-08-25 20:08:18 UTC; u16368
Repository:	R-Forge
Repository/R-Forge/Project:	easycoda
Repository/R-Forge/Revision:	53
Repository/R-Forge/DateTimeStamp:	2024-08-27 14:20:20
Date/Publication:	2024-08-27 14:20:20

Index of help topics:

```
ACLUST                  Amalgamation clustering of the parts of a
                        compositional data matrix
ALR                     Additive logratios
BAR                     Compositional bar plot
CA                      Correspondence analysis
CIplot_biv              Bivariate confidence and data ellipses
CLOSE                   Closure of rows of compositional data matrix
CLR                     Centred logratios
DOT                     Dot plot
DUMMY                   Dummy variable (indicator) coding
FINDALR                 Find the best ALR transformation
ILR                     Isometric logratio
LR                      All pairwise logratios
LR.VAR                  Total logratio variance
LRA                     Logratio analysis
PCA                     Principal component analysis
PLOT.CA                 Plot the results of a correspondence analysis
PLOT.LRA                Plot the results of a logratio analysis
PLOT.PCA                Plot the results of a principal component
                        analysis
PLOT.RDA                Plot the results of a redundancy analysis
PLR                     Pivot logratios
```

RDA	Redundancy analysis
SLR	Amalgamation (summed) logratio
STEP	Stepwise selection of logratios
STEPR	Stepwise selection of pairwise logratios for generalized linear modelling
VAR	Variance of a vector of observations, dividing by n rather than n-1
WARD	Ward clustering of a compositional data matrix
cups	Dataset: RomanCups
easyCODA-package	Compositional Data Analysis in Practice
fish	Dataset: FishMorphology
invALR	Inverse of additive logratios
invCLR	Inverse of centred logratios
invSLR	Inverse of full set of amalgamation balances
time	Dataset: TimeBudget
veg	Dataset: Vegetables

Author(s)

Michael Greenacre [aut, cre]

Maintainer: Michael Greenacre <michael.greenacre@upf.edu>

References

Greenacre, Michael (2018) Compositional Data Analysis in Practice. Chapman & Hall / CRC Press

See Also

ca

Examples

```
# Roman cups glass compositions
data("cups")
# unweighted logratio analysis
cups.ulra <- LRA(cups, weight=FALSE)
PLOT.LRA(cups.ulra)
# weighted logratio analysis
cups.wlra <- LRA(cups)
PLOT.LRA(cups.wlra)
```

ACLUST *Amalgamation clustering of the parts of a compositional data matrix*

Description

This function clusters the parts of a compositional data matrix, using amalgamation of the parts at each step.

Usage

```
ACLUST(data, weight = TRUE, close = TRUE)
```

Arguments

`data`	Compositional data matrix, with the parts as columns
`weight`	`TRUE` (default) for weighting using part averages of closed compositions, `FALSE` for unweighted analysis, or a vector of user-defined column weights
`close`	`TRUE` (default) will close the rows of `data` prior to clustering, `FALSE` leaves `data` as it is

Details

The function `ACLUST` performs amalgamation hierarchical clustering on the parts (columns) of a given compositional data matrix, as proposed by Greenacre (2019). At each step of the clustering two clusters are amalgamated that give the least loss of explained logratio variance.

Value

An object which describes the tree produced by the clustering process on the n objects. The object is a list with components:

`merge`	an n-1 by 2 matrix. Row i of `merge` describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then observation -j was merged at this stage. If j is positive then the merge was with the cluster formed at the (earlier) stage j of the algorithm. Thus negative entries in `merge` indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
`height`	a set of n-1 real values (non-decreasing for ultrametric trees). The clustering height: that is, the value of the criterion associated with the clustering method for the particular agglomeration.
`order`	a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches
`labels`	a vector of column labels, the column names of `data`

Author(s)

Michael Greenacre

References

Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC.

Greenacre, M. (2019), Amalgamations are valid in compositional data analysis, can be used in agglomerative clustering, and their logratios have an inverse transformation. Applied Computing and Geosciences, open access.

See Also

hclust, WARD,CLR, LR.VAR, CLOSE

Examples

```
data(cups)

# amalgamation clustering    (weighted parts)
cups.aclust <- ACLUST(cups)
plot(cups.aclust)

# reproducing Figure 2(b) of Greenacre (2019) (unweighted parts))
# dataset Aar is in the compositions package
# aar is a subset of Aar
# code given here within the '\dontrun' environment since external package 'compositions' required
## Not run:
  library(compositions)
  data(Aar)
  aar <- Aar[,c(3:12)]
  aar.aclust <- ACLUST(aar, weight=FALSE)
# the maximum height is the total variance
# convert to percents of variance NOT explained
  aar.aclust$height <- 100 * aar.aclust$height / max(aar.aclust$height)
  plot(aar.aclust, main="Parts of Unexplained Variance", ylab="Variance (percent)")

## End(Not run)
```

ALR *Additive logratios*

Description

Computation of additive logratios (ALRs) with respect to a specified part.

Usage

```
ALR(data, denom=ncol(data), weight=TRUE, stats=FALSE)
```

Arguments

`data`	A compositional data frame or matrix
`denom`	Number of part used in the denominator
`weight`	Logical indicating if varying weights are returned(default:`TRUE`). If `FALSE`, unweighted (equal) weights are returned. Alternatively a set of positive weights can be specified.
`stats`	Logical indicating if means, variances and total variance of the ALRs are returned (default:`FALSE`)

Details

The function `ALR` computes a set of additive logratios (ALRs) with respect to a specified part (by default, the last part).

Value

`LR`	The additive logratios (ALRs)
`LR.wt`	The weights assigned to the ALRs
`denom`	The index of the denominator used in the computation of the ALRs
`part.names`	The part names in the data, i.e. column names
`part.wt`	The part weights
`means`	The means of the ALRs (only returned if `stats = TRUE`)
`vars`	The variances of the ALRs (only returned if `stats = TRUE`)
`totvar`	The total variance of the ALRs (only returned if `stats = TRUE`)

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`invALR`, `LR`, `CLR`, `invCLR`, `LR.VAR`

Examples

```
data(veg)
ALR(veg, denom=2)
```

`BAR` *Compositional bar plot*

Description

Horizontal bar plot of compositional data

Usage

```
BAR(data, cols=rainbow(ncol(data)), col.names=colnames(data),
    row.names=rownames(data), order.column=NA, eps=0.5, main="", ylab="",
    ylim=c(0,nrow(data)), xlim=c(0,100), cex=1, truncate=NA)
```

Arguments

`data`	Compositional data matrix or data frame with compositions in rows, parts in columns
`cols`	Colours of points for each part, default rainbow
`col.names`	Part names, if modified
`row.names`	Sample names, if modified
`order.column`	By default parts are taken in order of columns, but can be re-ordered using this option
`eps`	Small space between bars, can be modified
`main`	Heading
`ylab`	Vertical axis label
`ylim`	Vertical axis limits (default is the number of rows in data)
`xlim`	Horizontal axis limits (default c(0,100))
`cex`	Character size scaling factor for labels
`truncate`	Truncate part (column) names to this number of characters for legend

Details

The function `BAR` makes a BAR plot for specified groups of points, which can be in columns of a matrix or data frame.

Author(s)

Michael Greenacre

References

Greenacre, M. (2016), Data reporting and visualization in ecology, Polar Biology: 39, 2189-2205.

See Also

`DOT`

Examples

```
# Vegetables data set: order samples by carbohydrates
data(veg)
BAR(veg, order.column=2)
data(time)
# TimeBudget data set: put domestic work in first column and order by it
BAR(time[,c(2,1,3,4,5,6)], order.column=1, main="Time Budget")
```

`CA` *Correspondence analysis*

Description

Computation of correspondence analysis on a table of nonnegative data.

Usage

```
CA(data, nd = 2, suprow = NA, supcol = NA)
```

Arguments

`data`	A data frame or matrix of nonnegative data (no negative values)
`nd`	Number of dimensions for summary solution if not 2 (default)
`suprow`	Indices of rows that are supplementary points
`supcol`	Indices of columns that are supplementary points

Details

The function `CA` is a simple wrapper for the `ca` function in the **ca** package (Nenadic and Greenacre, 2007), for compatibility within the **easyCODA** package.

Supplementary rows and columns can be declared (also known as passive points) – these do not contribute to the solution but are positioned on the solution axes.

The function borrows the structure and functions of the `ca` package, which is required, and produces a `ca` object, and the same `print`, `summary` and `plot` methods can be used, as for a `ca` object. It additionally exports the principal coordinates of both the rows and columns, not presently found in the `ca` package.

Value

`sv`	Singular values
`nd`	Number of dimensions in solution results
`rownames`	Row names
`rowmass`	Row weights
`rowdist`	Row logratio distances to centroid
`rowinertia`	Row inertias
`rowcoord`	Row standard coordinates
`rowpcoord`	Row principal coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column weights

coldist	Column logratio distances to centroid
colinertia	Column inertias
colcoord	Column standard coordinates
colpcoord	Column principal coordinates
N	The compositional data table

Author(s)

Michael Greenacre

References

Nenadic, O. and Greenacre, M. (2007). Correspondence analysis in R, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, **20 (3)**, https://www.jstatsoft.org/v20/i03/

See Also

PLOT.CA, plot.ca, summary.ca, print.ca

Examples

```
# CA of the Roman cups data (symmetric map)
data("cups")
PLOT.CA(CA(cups))
```

CIplot_biv *Bivariate confidence and data ellipses*

Description

Draws confidence and data ellipses in bivariate scatterplots

Usage

```
CIplot_biv(x, y, group, wt=rep(1/length(x),length(x)),
           varnames=c("x","y"), groupnames=sort(unique(group)),
           groupcols=rainbow(length(unique(group))),
           shownames=TRUE, xlim=c(NA,NA), ylim=c(NA,NA),
           lty=1, lwd=1, add=FALSE, alpha=0.95, ellipse=0,
           shade=FALSE, alpha.f=0.2, frac=0.01, cex=1)
```

Arguments

`x`	x-variable (horizontal) of scatterplot
`y`	y-variable (vertical) of scatterplot
`group`	Grouping variable
`wt`	Set of weights on the cases (operates when ellipse=1)
`varnames`	Vector of two labels for the axes (default is x and y)
`groupnames`	Vector of labels for the groups (default is 1, 2, etc...)
`groupcols`	Vector of colours for the groups
`shownames`	Whether to show group names at group centroids or not (default is `TRUE`)
`xlim`	Possible new x-limits for plot
`ylim`	Possible new y-limits for plot
`lty`	Line type for the ellipses (default is 1)
`lwd`	Line width for the ellipses (default is 1)
`add`	=`TRUE` if ellipses/intervals are added to existing plot (default is `FALSE`)
`alpha`	Confidence level of ellipses (default is 0.95)
`ellipse`	Type of ellipse (see Details below; default is 0 for normal-based ellipses)
`shade`	=`TRUE` for ellipse shading (default=`FALSE`)
`alpha.f`	Shading fraction (default is 0.2)
`frac`	Proportional part defining the width of the bars at the edges of confidence intervals (for `ellipse=3` and 4)
`cex`	Character expansion factor for group names

Details

The function `CIplot_biv` makes various types of confidence and data ellipses, according to option `ellipse`. Set `ellipse<0` for regular data-covering ellipses. Set `ellipse=0` (default) for normal-theory confidence ellipses. Set`ellipse=1` for bootstrap confidence ellipses. The option `ellipse=2` for the delta method is not implemented yet. Set `ellipse=3` for normal-theory confidence error bars lined up with axes. Set `ellipse=4` for bootstrap confidence error bars along axes. The package `ellipse` is required.

Author(s)

Michael Greenacre

References

Greenacre, M. (2016), Data reporting and visualization in ecology, Polar Biology, 39:2189-2205.

See Also

`DOT`

Examples

```
# Generate some bivariate normal data in three groups with different means
# Means (1,0), (0,1) and (0,0)
means  <- matrix(c(1,0,0,1,0,0), ncol=3)
data   <- matrix(nrow=300, ncol=2)
groups <- sample(rep(c(1,2,3), 100))
for(i in 1:300) data[i,] <- rnorm(c(1,1), mean=means[,groups[i]])
# Plot confidence ellipses with shading
CIplot_biv(data[,1], data[,2], group=groups, shade=TRUE)
```

CLOSE *Closure of rows of compositional data matrix*

Description

This function closes (or normalizes) the rows of a compositional data matrix, resulting in rows summing to 1.

Usage

```
CLOSE(x)
```

Arguments

x	Compositional data matrix.

Details

Compositional data carry relative information. It is sometimes required to close the data so that each row of observations sums to 1. The function `CLOSE` performs the closure.

Value

The closed compositional data matrix.

Author(s)

Michael Greenacre

References

Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC.

Examples

```
data(cups)
apply(cups, 2, sum)
cups <- CLOSE(cups)
apply(cups, 2, sum)
```

`CLR` *Centred logratios*

Description

Computation of centred logratios (CLRs).

Usage

```
CLR(data, weight=TRUE)
```

Arguments

`data`	A compositional data frame or matrix
`weight`	Logical indicating if varying weights are returned(default:`TRUE`). If `FALSE`, unweighted (equal) weights are returned. Alternatively a set of positive weights can be specified.

Details

The function `CLR` computes the set of centred logratios (CLRs).

Value

`LR`	The centred logratios (CLRs)
`LR.wt`	The weights assigned to the CLRs

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`invCLR`, `ALR`, `invALR`, `LR`, `LR.VAR`

Examples

```
data(veg)
CLR(veg)
```

cups *Dataset: RomanCups*

Description

This data set consists of the compositions of 11 oxides in 47 Roman cups found at an archaeological site in eastern England. Compositions are expressed as percentages.

Usage

```
data(cups)
```

Format

Data frame containing the 47 x 11 matrix.

Source

Baxter MJ, Beardah CC, Cool HEM and Jackson CM (2005) Compositional data analysis of some alkaline glasses. Mathematical Geology 37: 183-196.

DOT *Dot plot*

Description

Simple dot plot of original data

Usage

```
DOT(data, cols=NA, names=NA, groups=NA, pch=NA, horizon=FALSE, jitter=1,
     xscale=NA, xscalefac=1, yaxis=TRUE, shownames=TRUE, main="", ylab="",
     xlim=c(NA,NA), ylim=c(NA, NA), cex=1)
```

Arguments

`data`	Matrix or data frame with data groups in columns; alternatively, a single vector but then groups (if any) have to specified with the `groups` option
`cols`	Colours of points for each sample, default rainbow
`names`	Labels for variables, by default the column names of data, or group names
`groups`	Group codes to split the data vector into separate plots
`pch`	Point character
`horizon`	`TRUE` if horizontal gray dashed lines required at "nice" y-values (default FALSE, not implemented yet)

`jitter`	1 by default, increase or decrease slightly for more jitter
`xscale`	User-supplied positions of points on horizontal axis
`xscalefac`	1 by default, rescale the positions on horizontal axis
`yaxis`	TRUE by default, FALSE to suppress and optionally add afterwards
`shownames`	`TRUE` by default; `FALSE` to not show group names and add them externally
`main`	Heading
`ylab`	Vertical axis label
`xlim`	Horizontal axis limits
`ylim`	Vertical axis limits
`cex`	Character size adjustment for labels

Details

The function `DOT` makes a dot plot for specified groups of points, which can be in columns of a matrix or data frame, or in a single vector with group codes specified separately.

Author(s)

Michael Greenacre

References

Greenacre, M. (2016), Data reporting and visualization in ecology, Polar Biology, 39:2189-2205.

See Also

`BAR`

Examples

```
# Dot plot of columns of Vegetables data set
data(veg)
DOT(veg)
# Dot plot of domestic work column of TimeBudget data set, split by sex
data(time)
DOT(time[,2], groups=substr(rownames(time),3,3), cols=c("blue","red"), ylim=c(0,20),
    jitter=2, main="Percentage of Domestic Work")
```

DUMMY *Dummy variable (indicator) coding*

Description

Convert categorical variable to dummy (0/1) coding

Usage

```
DUMMY(x, catnames=NA)
```

Arguments

`x`	Variable (vector) of categorical data to be coded
`catnames`	Category names

Details

The function `DUMMY` takes a categorical variable and converts it to a set of dummy variables (zeros and ones), where the ones indicate the corresponding category. There are as many columns in the result as there are unique categories in the input vector.

Author(s)

Michael Greenacre

Examples

```
# Indicator (dummy) coding of sex in FishMorphology data set
data(fish)
sex   <- fish[,1]
sex.Z <- DUMMY(sex, catnames=c("F","M"))
```

FINDALR *Find the best ALR transformation*

Description

Searching over every possible reference part for choosing an optimal ALR transformation.

Usage

```
FINDALR(data, weight=FALSE)
```

Arguments

`data`	Compositional data matrix, with the parts as columns
`weight`	FALSE (default) for equally weighted parts, TRUE when weights are in data list object, or a vector of user-defined part weights

Details

The function `FINDALR` considers every possible set of additive logratio (ALR) transformations, by trying each of the references. For each set the closeness to isometry is measured by the Procrustes correlation. In addition, the variance of the log-transformed reference is also computed. The reference with highest Procrustes correlation and the reference with the lowest variance of its log-transform are identified. The number of ALRs computed is equal to 1 less than the number of rows or columns, whichever is the smallest.

Value

An object which describes the tree produced by the clustering process on the n objects. The object is a list with components:

`totvar`	Total logratio variance
`procrust.cor`	The Procrustes correlations of the ALRs using each reference
`procrust.max`	The value of the highest Procrustes correlation
`procrust.ref`	The reference corresponding to the highest correlation
`var.log`	Variances of the log-transformed references
`var.min`	The value of the lowest variance
`var.ref`	The reference corresponding to the lowest variance

Author(s)

Michael Greenacre

References

Greenacre, M., Martinez-Alvaro, M. and Blasco, A. (2021), Compositional data analysis of microbiome and any-omics datasets: a validation of the additive logratio transformation, Frontiers in Microbiology 12: 2625
Gower, J. and Dijksterhuis, G.B. (2004), Procrustes Problems. Oxford University Press
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC

See Also

`ALR`

Examples

```
# For the fish morphometric data, first close (normalize)
# then loop over the 26 possible references
data(fish)
FINDALR(CLOSE(fish[,4:29]))
# Note that for the default option weight=FALSE closing the data is not necessary
```

`fish` *Dataset: FishMorphology*

Description

This data set consists of the sex, habitat, mass and then 26 morphometric measurements on 75 fish (Arctic charr)

Usage

```
data(fish)
```

Format

Data frame containing the 75 x 29 matrix. Column 1 contains sex (1=female, 2=male). Column 2 contains habitat (1=litoral, 2=pelagic). Column 3 contains the mass in grams. Columns 4 to 29 contain the 26 morphometric measurements.

Source

Greenacre, M and Primicerio, R (2010) Multivariate Analysis of Ecological Data. BBVA Foundation, Bilbao. Free download at www.multivariatestatistics.org

---

`ILR` *Isometric logratio*

Description

Computation of a single isometric logratio (ILR)

Usage

```
ILR(data, numer=NA, denom=NA, weight=TRUE)
```

Arguments

`data`	A compositional data frame or matrix
`numer`	Vector of parts in the numerator
`denom`	Vector of parts in the denominator
`weight`	Logical indicating if a varying weight is returned (default:`TRUE`). If `FALSE`, a weight based on equally-weighted parts is returned. Alternatively a positive weight can be specified.

Details

The function `ILR` computes a single isometric logratio based on the specified numerator and denominator parts that define the two geometric means in the ratio.

Value

`LR`	The isometric logratio (ILR)
`LR.wt`	The weight assigned to the ILR

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`SLR`, `ALR`, `PLR`, `LR`

Examples

```
data(veg)
ILR(veg, numer=1, denom=2:3)
```

`invALR` *Inverse of additive logratios*

Description

Given additive logratios (ALRs) with respect to a specified part, compute the inverse (i.e. original parts)

Usage

```
invALR(ALRmatrix, part.names=paste("part",1:(ncol(ALRmatrix)+1),sep=""), denom=NA)
```

Arguments

`ALRmatrix`	A matrix of additive logratios (ALRs) with respect to a specified part)
`part.names`	Part names in the reconstructed compositional data matrix
`denom`	The index of the denominator used in the computation of the ALRs (default: last part))

Details

The function `invALR` computes the original parts, given the additive logratios (ALRs)

Value

`parts`	The reconstructed parts (they add up to 1)

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`ALR`, `LR`, `CLR`, `invCLR`, `LR.VAR`

Examples

```
data(veg)
# compute additive logratios with respect to second part
veg.ALR <- ALR(veg, denom=2)$LR
# recover original parts (to get same order, specify the denominator used originally)
invALR(veg.ALR, denom=2)
```

`invCLR` *Inverse of centred logratios*

Description

Given centred logratios (CLRs), compute the inverse (i.e. recover the original parts)

Usage

```
invCLR(CLRmatrix, part.names=colnames(CLRmatrix))
```

Arguments

`CLRmatrix`	A matrix of centred logratios
`part.names`	Part names in the reconstructed compositional data matrix

Details

The function `invCLR` computes the original parts, given the centred logratios (CLRs)

Value

`parts`	The reconstructed parts (they add up to 1)

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`CLR`, `ALR`, `invALR`, `LR.VAR`

Examples

```
data(veg)
# compute centred logratios
veg.CLR <- CLR(veg)$LR
# invert back to original parts (parts closed to sum to 1)
invALR(veg.CLR)
```

`invSLR` *Inverse of full set of amalgamation balances*

Description

Given a full set of amalgamation (or summation) balances (SLRs), compute the inverse (i.e. recover the original parts)

Usage

```
invSLR(SLRmatrix, part.names=NA, ratio.names=colnames(SLRmatrix))
```

Arguments

`SLRmatrix`	A matrix of amalgamation logratios, one less column than the number of parts
`part.names`	Part names in the reconstructed compositional data matrix
`ratio.names`	Definition of the amalgamation logratios

Details

The function `invSLR` computes the original parts, given the amalgamation logratios (CLRs). The amalgamation logratios are specified in `ratio.names` in the format `num/den` where `num` and `den` are the numerator and denominator amalgamations respectively. An amalgamation is specified as `"p1&p2&..."`, where `p1`, `p2`, etc. are the parts summed in the amalgamation. For example, an SLR of the ratio MnO/(CaO+P2O5) would be names as `"MnO/CaO&P2O5"`.

Value

`parts`	The reconstructed parts (they add up to 1)

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`CLR`, `ALR`, `invALR`, `LR.VAR`

Examples

```
data(veg)
# compute centred logratios
veg.CLR <- CLR(veg)$LR
# invert back to original parts (parts closed to sum to 1)
invALR(veg.CLR)
```

`LR` *All pairwise logratios*

Description

Computation of all pairwise logratios.

Usage

```
LR(data, ordering=1:ncol(data), weight=TRUE)
```

Arguments

`data`	A compositional data frame or matrix
`ordering`	A permutation of the columns (default: the original ordering)
`weight`	Logical indicating if varying weights are returned (default:`TRUE`). If `FALSE`, unweighted (equal) weights are returned. Alternatively a set of positive weights can be specified.

Details

The function `LR` computes the complete set of pairwise logratios, in the order [1,2], [1,3], [2,3], [1,4], [2,4], [3,4], etc.

Value

`LR`	The pairwise logratios as columns of a data matrix
`LR.wt`	The weights assigned to the respective logratios

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`ALR`, `invALR`, `CLR`, `invCLR`, `LR.VAR`

Examples

```
data(veg)
LR(veg)
```

`LR.VAR` *Total logratio variance*

Description

Computation of total (weighted)logratio variance.

Usage

```
LR.VAR(LRdata, row.wt = NA, weight=TRUE, vars=FALSE)
```

Arguments

`LRdata`	Matrix of logratios, either a vector or preferably the logratio object resulting from one of the functions ALR, CLR, PLR or LR
`row.wt`	Optional set of row weights (default: equal weights)
`weight`	Logical indicating if varying weights are returned(default:`TRUE`). If `FALSE`, unweighted (equal) weights are returned. Alternatively a set of positive weights can be specified.
`vars`	If `TRUE`, output individual variances as well (default `FALSE`)

Details

The function `LR.VAR` computes the sum of the logratio variances provided as input, using the weights in the logratio data object.

Value

`LRtotvar`	The total logratio variance
`LRvars`	(optional, if vars=TRUE, the individual logratio variances composing the total)

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`LR`, `ALR`, `invALR`, `CLR`, `invCLR`

Examples

```
data(cups)
# These give identical total logratio variances (weighted, by default)
LR.VAR(CLR(cups))
LR.VAR(LR(cups))
# Summing over all sets of ALRs gives twice the variance
totvar <- 0
for(j in 1:ncol(cups)) totvar <- totvar + LR.VAR(ALR(cups, denom=j))
totvar/2
```

LRA *Logratio analysis*

Description

Computation of weighted or unweighted logratio analysis of a samples-by-parts compositional data table.

Usage

```
LRA(data, nd = 2, weight = TRUE, suprow = NA, row.wt = NA, amalg = NA, supamalg = FALSE)
```

Arguments

`data`	A data frame or matrix of compositional data, with no zero values
`nd`	Number of dimensions for summary solution if not 2 (default)
`weight`	TRUE (default) for part weighting, FALSE for unweighted analysis, or a vector of user-defined part weights
`suprow`	Indices of rows that are supplementary points
`row.wt`	Optional user-defined set of positive weights for the rows (samples) (default: equal weights)
`amalg`	Optional list of amalgamated parts
`supamalg`	FALSE (default) when amalgamations are active and their subparts supplementary, TRUE when amalgamations are supplementary and their parts active

Details

The function `LRA` computes a log-ratio analysis of a table of compositional data based on the singular value decomposition. By default the weighted log-ratio analysis is computed (Greenacre & Lewi 2009). For the unweighted logratio analysis (Aitchison & Greenacre 2002), specify the option `weight=FALSE`.

User-specified weights can be supplied, for the rows and/or the columns. Usually row weights are not specified, and are equal unless intentional weighting of the samples is desired. Default column weights (if `weight = TRUE`) are the part means of the true compositional table, thus summing to 1. User-specified part weights can be provided using the same `weight` option.

Supplementary rows can be declared (also known as passive points) – these do not contribute to the solution but are positioned on the solution axes.

Amalgamations can be defined and can either replace their constituent parts (default) or be declared supplementary using the `supamalg` option: `supamalg = FALSE` (default), `= TRUE` if all declared amalgamations are supplementary.

The function borrows the structure and functions of the `ca` package, which is required, and produces a `ca` object, and the same `print`, `summary` and `plot` methods can be used, as for a `ca` object.

Value

`sv`	Singular values
`nd`	Number of dimensions in solution results
`rownames`	Row names
`rowmass`	Row weights
`rowdist`	Row logratio distances to centroid
`rowinertia`	Row inertias
`rowcoord`	Row standard coordinates
`rowpcoord`	Row principal coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column weights
`coldist`	Column logratio distances to centroid
`colinertia`	Column inertias
`colcoord`	Column standard coordinates
`colpcoord`	Column principal coordinates
`N`	The compositional data table

Author(s)

Michael Greenacre

References

Aitchison, J. and Greenacre, M. (2002), Biplots of compositional data, Applied Statistics 51, 375-392.
Greenacre, M. and Lewi, P.J. (2009), Distributional equivalence and subcompositional coherence in the analysis of compositional data, contingency tables and ratio scale measurements. Journal of Classification 26, 29-54. Greenacre, M. (2020), Amalgamations are valid in compositional data analysis, can be used in agglomerative clustering, and their logratios have an inverse transformation. Applied Computing and Geosciences 5, 100017.

See Also

`plot.ca`, `summary.ca`, `print.ca`

Examples

```
# (weighted) LRA of the RomanCups data set, showing default symmetric map
data("cups")
PLOT.LRA(LRA(cups))
# (unweighted) LRA of the RomanCups data set, showing default symmetric map
# the solution is completely different
PLOT.LRA(LRA(cups, weight=FALSE))
```

PCA *Principal component analysis*

Description

Computation of weighted or unweighted principal component analysis of a matrix of interval-scale data (e.g. a matrix of logratios).

Usage

```
PCA(data, nd = 2, weight = TRUE, row.wt = NA, suprow = NA)
```

Arguments

`data`	A data frame or matrix of interval-scale data, or logratio object from functions `ALR`, `CLR` or `LR`
`nd`	Number of dimensions for summary solution if not 2 (default)
`weight`	TRUE (default) for column weighting, FALSE for unweighted analysis, or a vector of user-defined column weights
`row.wt`	Optional user-defined set of positive weights for the rows (samples) (default: equal weights)
`suprow`	Indices of rows that are supplementary points (NOTE: this option is not implemented in this version)

Details

The function `PCA` computes an unstandardized principal component analysis, based on the singular value decomposition, of a matrix of interval-scale data, usually a matrix of logratios in the context of this package (but it can be used for general data as well). For general usage the unweighted option `weight = FALSE` might be preferred, but the default is weighted in the present context of compositional data.

User-specified weights can be supplied, for the rows and/or the columns. Usually row weights are not specified, and are equal unless intentional weighting of the samples is desired. User-specified part weights can be provided using the `weight` option.

Supplementary rows and columns can be declared (also known as passive points) – these do not contribute to the solution but are positioned on the solution axes. Notice that this optyion is not implemented in the present version, but will appear in the next one.

The function borrows the structure and functions of the `ca` package, which is required, and produces a `ca` object, and the same `print`, `summary` and `plot` methods can be used, as for a `ca` object.

Value

`sv`	Singular values
`nd`	Dimenson of the solution
`rownames`	Row names

`rowmass`	Row weights
`rowdist`	Row logratio distances to centroid
`rowinertia`	Row variances
`rowcoord`	Row standard coordinates
`rowpcoord`	Row principal coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column weights
`coldist`	Column logratio distances to centroid
`colinertia`	Column variances
`colcoord`	Column standard coordinates
`colpcoord`	Column principal coordinates
`N`	The data table

Author(s)

Michael Greenacre

References

Aitchison, J. and Greenacre, M. (2002), Biplots of compositional data, Applied Statistics 51, 375-392.
Greenacre, M. (2010), Biplots in Practice, BBVA Foundation, Bilbao. Free download from www.multivariatestatistics.org

See Also

`PLOT.PCA`, `plot.ca`, `summary.ca`, `print.ca`

Examples

```
# compute logratios of Vegetables data set
data("veg")
veg.LR <- LR(veg)
# unweighted PCA biplot of the results
veg.pca <- PCA(veg.LR$LR, weight=FALSE)
PLOT.PCA(veg.pca, map="asymmetric")
```

`PLOT.CA` *Plot the results of a correspondence analysis*

Description

Various maps and biplots of the results of a correspondence analysis using function `CA`.

Usage

```
PLOT.CA(obj, map="symmetric", rescale=1, dim=c(1,2), axes.inv = c(1,1), main="",
       cols=c("blue","red"), colarrows = "pink", cexs=c(0.8,0.8), fonts=c(2,4))
```

Arguments

`obj`	A CA object created using function `CA`
`map`	Choice of scaling of rows and columns: `"symmetric"` (default), `"asymmetric"` or `"contribution"`
`rescale`	A rescaling factor applied to column coordinates (default is 1 for no rescaling)
`dim`	Dimensions selected for horizontal and vertical axes of the plot (default is c(1,2))
`main`	Title for plot
`axes.inv`	Option for reversing directions of horizontal and vertical axes (default is c(1,1) for no reversing, change one or both to -1 for reversing)
`cols`	Colours for row and column labels (default is c("blue","red"))
`colarrows`	Colour for arrows in asymmetric and contribution biplots (default is "pink")
`cexs`	Character expansion factors for row and column labels (default is c(0.8,0.8))
`fonts`	Fonts for row and column labels (default is c(2,4))

Details

The function `PLOT.CA` makes a scatterplot of the results of a correspondence analysis (computed using function `CA`), with various options for scaling the results and changing the direction of the axes. By default, dimensions 1 and 2 are plotted on the horizontal and vertical axes, and it is assumed that row points refer to samples and columns to variables.

By default, the symmetric scaling is used, where both rows and columns are in principal coordinates and have the same amount of weighted variance (i.e. inertia) along the two dimensions. The other options are biplots: the asymmetric option, when columns are in standard coordinates, and the contribution option, when columns are in contribution coordinates. In cases where the row and column displays occupy widely different extents, the column coordinates can be rescaled using the `rescale` option.

Author(s)

Michael Greenacre

References

Greenacre, M. (2013), Contribution biplots, Journal of Computational and Graphical Statistics, 22, 107-122.

See Also

`CA`, `plot.ca`

Examples

```
data("cups")
cups.ca <- CA(cups)
PLOT.CA(cups.ca, map="contribution", rescale=0.2)
# Compare the above plot with that of a weighted LRA -- practically the same
cups.lra <- LRA(cups)
PLOT.LRA(cups.lra, map="contribution", rescale=0.2)
```

PLOT.LRA *Plot the results of a logratio analysis*

Description

Various maps and biplots of the results of a logratio analysis using function `LRA`.

Usage

```
PLOT.LRA(obj, map="symmetric", rescale=1, dim=c(1,2), axes.inv = c(1,1), main="",
        cols=c("blue","red"), colarrows = "pink", cexs=c(0.8,0.8), fonts=c(2,4))
```

Arguments

`obj`	An LRA object created using function `LRA`
`map`	Choice of scaling of rows and columns: `"symmetric"` (default), `"asymmetric"` or `"contribution"`
`rescale`	A rescaling factor applied to column coordinates (default is 1 for no rescaling)
`dim`	Dimensions selected for horizontal and vertical axes of the plot (default is c(1,2))
`axes.inv`	Option for reversing directions of horizontal and vertical axes (default is c(1,1) for no reversing, change one or both to -1 for reversing)
`main`	Title for plot
`cols`	Colours for row and column labels (default is c("blue","red"))
`colarrows`	Colour for arrows in asymmetric and contribution biplots (default is "pink")
`cexs`	Character expansion factors for row and column labels (default is c(0.8,0.8))
`fonts`	Fonts for row and column labels (default is c(2,4))

Details

The function PLOT.LRA makes a scatterplot of the results of a logratio analysis (computed using function LRA), with various options for scaling the results and changing the direction of the axes. By default, dimensions 1 and 2 are plotted on the horizontal and vertical axes, and it is assumed that row points refer to samples and columns to compositional parts.

By default, the symmetric scaling is used, where both rows and columns are in principal coordinates and have the same amount of weighted variance along the two dimensions. The other options are the asymmetric option, when columns are in standard coordinates, and the contribution option, when columns are in contribution coordinates. In cases where the row and column displays occupy widely different extents, the column coordinates can be rescaled using the rescale option.

Author(s)

Michael Greenacre

References

Greenacre, M. (2013), Contribution biplots, Journal of Computational and Graphical Statistics, 22, 107-122.

See Also

plot.ca

Examples

```
# perform LRA on the vegetable compositions (unweighted form)
data("veg")
veg.lra <- LRA(veg, weight=FALSE)
PLOT.LRA(veg.lra)
# perform LRA on the Roman cups data set and plot the results (weighted form)
data("cups")
cups.lra <- LRA(cups)
PLOT.LRA(cups.lra, map="contribution", rescale=0.2)
```

PLOT.PCA *Plot the results of a principal component analysis*

Description

Various maps and biplots of the results of a principal component analysis using function PCA.

Usage

```
PLOT.PCA(obj, map="symmetric", rescale=1, dim=c(1,2), axes.inv = c(1,1),
         main="", cols=c("blue","red"), colarrows = "pink", cexs=c(0.8,0.8),
         fonts=c(2,4))
```

Arguments

`obj`	An LRA object created using function `LRA`
`map`	Choice of scaling of rows and columns: `"symmetric"` (default), `"asymmetric"` or `"contribution"`
`rescale`	A rescaling factor applied to column coordinates (default is 1 for no rescaling)
`dim`	Dimensions selected for horizontal and vertical axes of the plot (default is c(1,2))
`axes.inv`	Option for reversing directions of horizontal and vertical axes (default is c(1,1) for no reversing, change one or both to -1 for reversing)
`main`	Title for plot
`cols`	Colours for row and column labels (default is c("blue","red"))
`colarrows`	Colour for arrows in asymmetric and contribution biplots (default is "pink")
`cexs`	Character expansion factors for row and column labels (default is c(0.8,0.8))
`fonts`	Fonts for row and column labels (default is c(2,4))

Details

The function `PLOT.PCA` makes a scatterplot of the results of a logratio analysis (computed using function `PCA`), with various options for scaling the results and changing the direction of the axes. By default, dimensions 1 and 2 are plotted on the horizontal and vertical axes, and it is assumed that row points refer to samples and columns to variables.

By default, the symmetric scaling is used, where both rows and columns are in principal coordinates and have the same amount of weighted variance along the two dimensions. The other options are biplots: the asymmetric option, when columns are in standard coordinates, and the contribution option, when columns are in contribution coordinates. In cases where the row and column displays occupy widely different extents, the column coordinates can be rescaled using the `rescale` option.

Author(s)

Michael Greenacre

References

Greenacre, M. (2013), Contribution biplots, Journal of Computational and Graphical Statistics, 22, 107-122.

See Also

`plot.ca`

Examples

```
# perform weighted PCA on the ALRs of the Roman cups data set
# where the first oxide silica is chosen as the denominator
data("cups")
cups.alr <- ALR(cups, denom=1)
cups.pca <- PCA(cups.alr)
PLOT.PCA(cups.pca, map="contribution", rescale=0.2, axes.inv=c(1,-1))
```

PLOT.RDA	*Plot the results of a redundancy analysis*

Description

Various maps and biplots/triplots of the results of a redundancy analysis using function `RDA`.

Usage

```
PLOT.RDA(obj, map="symmetric", indcat=NA, rescale=1, dim=c(1,2), axes.inv=c(1,1),
         main="", rowstyle=1, cols=c("blue","red","forestgreen"),
         colarrows=c("pink","lightgreen"), colrows=NA, pchrows=NA, colcats=NA,
         cexs=c(0.8,0.8,0.8), fonts=c(2,4,4))
```

Arguments

`obj`	An RDA object created using function `RDA`
`map`	Choice of scaling of rows and columns: `"symmetric"` (default), `"asymmetric"` or `"contribution"`
`indcat`	A vector indicating which of the covariates are dummy (or fuzzy) variables
`rescale`	A rescaling factor applied to column coordinates(default is 1 for no rescaling). If rescale is a vector with two values, the first applies to the column coordinates and the second to the covariate coordinates.
`dim`	Dimensions selected for horizontal and vertical axes of the plot (default is c(1,2))
`axes.inv`	Option for reversing directions of horizontal and vertical axes (default is c(1,1) for no reversing, change one or both to -1 for reversing)
`main`	Title for plot
`rowstyle`	Scaling option for row coordinates, either 1 (SVD coordinates, default) or 2 (as supplementary points)
`cols`	Colours for row and column and covariate labels (default is c("blue","red","forestgreen"))
`colarrows`	Colour for arrows in asymmetric or contribution biplots, for columns and co-variates (default is c("pink","lightgreen"))
`colrows`	Optional vector of colours for rows
`pchrows`	Optional vector of point symbols for rows
`colcats`	Optional vector of colours for covariate categories (dummy variables)
`cexs`	Vector of character expansion factors for row and column and covariate labels (default is c(0.8,0.8,0.8))
`fonts`	Vector of font styles for row and column and covariate labels (default is c(2,4,4))

Details

The function `PLOT.RDA` makes a scatterplot of the results of a redundancy analysis (computed using function `RDA`), with various options for scaling the results and changing the direction of the axes. By default, dimensions 1 and 2 are plotted on the horizontal and vertical axes, and it is assumed that row points refer to samples and columns to compositional parts. Covariates are plotted according to their regression coefficients with the RDA dimensions, and if they contain dummy (or fuzzy) variables these are indicated by the option indcat, and hence plotted as centroids not arrows.

By default, the symmetric scaling is used, where both rows and columns are in principal coordinates and have the same amount of weighted variance along the two dimensions. The other options are the asymmetric option, when columns are in standard coordinates, and the contribution option, when columns are in contribution coordinates. In cases where the row and column displays as well as the covariate positions occupy widely different extents, the column and covariate coordinates can be rescaled using the `rescale` option.

Author(s)

Michael Greenacre

See Also

`RDA`

Examples

```
# see the use of PLOT.RDA in the example of the RDA function
```

`PLR` *Pivot logratios*

Description

Computation of the set of pivot logratios(PLRs) based on the specified ordering of parts

Usage

```
PLR(data, ordering=1:ncol(data), weight=TRUE)
```

Arguments

`data`	A compositional data frame or matrix
`ordering`	The ordering of the parts to be used in the PLRs (by default, the original ordering of the columns)
`weight`	Logical indicating if varying weights are returned (default:`TRUE`). If `FALSE`, weights based on equally-weighted parts are returned. Alternatively a set of positive weights can be specified.

Details

The function `PLR` computes the set of pivot logratios according to the ordering of the parts.

Value

`LR`	The pivot logratios (PLRs)
`LR.wt`	The weights assigned to the PLRs

Author(s)

Michael Greenacre

References

Hron K., Filzmoser P., de Caritat P., Fiserova E., Gardlo A. (2017). Weighted pivot coordinates for copositional data and their application to geochemical mapping. Mathematical Geosciences 49, 777-796.

See Also

`ILR`, `SLR`, `CLR`, `ALR`, `LR`

Examples

```
data(veg)
PLR(veg, ordering=c(1,3,2))
```

`RDA` *Redundancy analysis*

Description

Computation of weighted or unweighted redundancy analysis of a samples-by-parts compositional data table, given a set of covariates.

Usage

```
RDA(data, cov=NA, nd = NA, weight = TRUE, suprow = NA, row.wt = NA)
```

Arguments

`data`	A data frame or matrix of interval-scale data, e.g. logratios (which are preferably in a list object with weights)
`cov`	List of covariates for constraining solution
`nd`	Number of dimensions for summary output, by default the number of constraining dimensions

`weight`	TRUE (default) when weights are in data list object, FALSE for unweighted analysis, or a vector of user-defined part weights
`suprow`	Indices of rows that are supplementary (passive) points (NOTE: this option is not implemented in this version)
`row.wt`	Optional user-defined set of positive weights for the rows (samples) (default: equal weights)

Details

The function `RDA` computes a redundancy analysis of a matrix of interval-scaled data, constrained by a matrix of covariates, using the singular value decomposition. By default weights are assumed in the data list object. For the unweighted logratio analysis, specify the option `weight=FALSE`. If `weight = TRUE` (the default) it is assumed that the weights are included in the data object, which comes from one of the logratio functions. User-specified part weights can be provided using the same `weight` option.

Usually row weights are not specified, they are equal unless intentional weighting of the samples is desired. Supplementary rows can be declared (also known as passive points) – these do not contribute to the solution but are positioned on the solution axes. This option will be available in the next release of the package.

Value

`sv`	Singular values
`nd`	Number of dimensions in the solution output
`rownames`	Row names
`rowmass`	Row weights
`rowdist`	Row distances to centroid
`rowinertia`	Row variances
`rowcoord`	Row standard coordinates
`rowpcoord`	Row principal coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column weights
`coldist`	Column logratio distances to centroid
`colinertia`	Column variances
`colcoord`	Column standard coordinates
`colpcoord`	Column principal coordinates
`covcoord`	Regression coordinates of constraining variables
`covnames`	Names of constraining variables
`N`	The data table (usually logratios in this package)
`cov`	The table of covariates

Author(s)

Michael Greenacre

References

Van den Wollenbergh, A. (1977), Redundancy analysis. An alternative to canonical correlation analysis, Psychometrika 42, 207-219.
Greenacre, M. (2013), Contribution biplots, Journal of Computational and Graphical Statistics 22, 107-122.

See Also

PLOT.RDA, CLR, LR, DUMMY

Examples

```
# Data frame fish has sex, habitat and mass in first columns,
# then morphometric data in remaining columns
data("fish")
sex     <- fish[,1]
habitat <- fish[,2]
mass    <- fish[,3]
fishm   <- as.matrix(fish[,4:29])
# Convert to compositional data matrix
fishm   <- CLOSE(fishm)
# Compute logarithm of mass and interaction of sex (F/M) and habitat (L/P) categories
logmass <- log(mass)
sexhab  <- 2*(sex-1)+habitat
sexhab.names <- c("FL","FP","ML","MP")
rownames(fishm) <- sexhab.names[sexhab]
# Create dummy variables for sexhab and create matrix of covariates
sexhab.Z <- DUMMY(sexhab, catnames=sexhab.names)
vars     <- cbind(logmass, sexhab.Z)
# Perform RDA on centred logratios
require(ca)
fish.rda <- RDA(CLR(fishm), cov=vars)
# Plot results
# (for more options see Appendix of Compositional Data Analysis in Practice)
PLOT.RDA(fish.rda, map="contribution", rescale=0.05, indcat=2:5,
         colrows=rainbow(4, start=0.1, end=0.8)[sexhab], cexs=c(0.8,0.8,1))
```

SLR *Amalgamation (summed) logratio*

Description

Computation of a single amalgamation (summed) logratio

Usage

```
SLR(data, numer=NA, denom=NA, weight=TRUE)
```

Arguments

`data`	A compositional data frame or matrix
`numer`	Vector of parts in the numerator
`denom`	Vector of parts in the denominator
`weight`	Logical indicating if a varying weight is returned (default:`TRUE`). If `FALSE`, a weight based on equally-weighted parts is returned. Alternatively a positive weight can be specified.

Details

The function `SLR` computes a single amalgamation logratio based on the specified numerator and denominator parts that define the two summations in the ratio.

Value

`LR`	The amalgamation (summed)) logratio (SLR)
`LR.wt`	The weight assigned to the SLR

Author(s)

Michael Greenacre

References

Aitchison, J. (1986), The Statistical Analysis of Compositional Data, Chapman & Hall.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC Press.

See Also

`ILR`, `ALR`, `CLR`, `PLR`, `LR`

Examples

```
data(veg)
SLR(veg, numer=1, denom=2:3)
```

STEP *Stepwise selection of logratios*

Description

Stepwise selection of pairwise logratios that explain maximum variance in a target matrix.

Usage

```
STEP(data, datatarget=data, previous=NA, previous.wt=NA, weight=TRUE,
     random=FALSE, nsteps=min(ncol(data), ncol(datatarget))-1, top=1)
```

Arguments

`data`	A data frame or matrix of compositional data on which pairwise logratios are computed
`datatarget`	A matrix of interval-scale data, with as many rows as `data`, which serves as the target matrix whose variance is to be explained (by default it is the same matrix as data, in which case total logratio variance is to be explained)
`previous`	A vector or matrix of variables to be forced in before logratios are sought
`previous.wt`	Possible weights of the variable(s) forced in before logratios are sought (if not specified, weights of 1 are assumed)
`weight`	`TRUE` (default) when weights are in data list object, `FALSE` for unweighted analysis, or a vector of user-defined part weights
`random`	`TRUE` if a random selection is made of tied logratios; `FALSE` (default) if logratio that maximizes Procrustes correlation is chosen
`nsteps`	Number of steps to take (by default, one less than the number of columns of data and of datatarget, whichever is smaller)
`top`	Number of top variance-explaining logratios returned after last step (by default, 1, i.e. the best)

Details

The function `STEP` sequentially computes the logratios in a data matrix (usually compositional) that best explain the variance in a second matrix, called the target matrix. By default, the target matrix is the same matrix, in which case the logratios that best explain the logratio variance in the same matrix are computed. In this case, weights for the data matrix are assumed by default, proportional to part means of the compositional data matrix. For the unweighted logratio variance, specify the option `weight=FALSE`. User-specified weights on the columns of the data matrix (usually compositional parts) can be provided using the same `weight` option.

If the target matrix is a different matrix, it is the logratio variance of that matrix that is to be explained. An option for the target matrix to be any response matrix will be in the next release.

If `nsteps` > 1 and `top=1` the results are in the form of an optimal set of logratios that sequentially add maximum explained variance at each step. If `top>1` then at the last step the ordered list of top variance-explaining logratios is returned, which allows users to make an alternative choice of the logratio based on substantive knowledge. Hence, if `nsteps=1` and `top=10`, for example, the procedure will move only one step, but list the top 10 logratios for that step. If `top=1` then all results with extension `.top` related to the top ratios are omitted because they are already given.

Value

`names`	Names of maximizing ratios in stepwise process
`ratios`	Indices of ratios
`logratios`	Matrix of logratios
`R2max`	Sequence of maximum cumulative explained variances
`pro.cor`	Corresponding sequence of Procrustes correlations
`names.top`	Names of "top" ratios at last step
`ratios.top`	Indices of "top" ratios
`logratios.top`	Matrix of "top" logratios
`R2.top`	Sequence of "top" cumulative explained variances (in descending order)
`pro.cor.top`	Corresponding sequence of "top" Procrustes correlations
`totvar`	Total logratio variance of target matrix

Author(s)

Michael Greenacre

References

Van den Wollenbergh, A. (1977), Redundancy analysis. An alternative to canonical correlation analysis, Psychometrika 42, 207-219.
Greenacre, M. (2018), Variable selection in compositional data analysis using pairwise logratios, Mathematical Geosciences, DOI: 10.1007/s11004-018-9754-x.
Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC

See Also

`PLOT.RDA`, `CLR`, `LR`, `ALR`

Examples

```
# Stepwise selection of ratios for RomanCups data set
data(cups)
# Set seed to obtain same results as in Appendix C of Greenacre (2018)
set.seed(2872)
STEP(cups, random=TRUE)
# Select best ratio, but output "top 5"
STEP(cups, nsteps=1, top=5)
```

STEPR	*Stepwise selection of pairwise logratios for generalized linear modelling*

Description

Three different algorithms for selecting pairwise logratios that best explain/predict a response variable, which could be continuous, binary or count

Usage

```
STEPR(data, y, method = NA, family = "gaussian", nsteps = ncol(data)-1,
      top = 1, previous = NA, criterion = "Bonferroni", alpha = 0.05,
      previousparts=NA, denom=NA)
```

Arguments

`data`	A data frame or matrix of compositional data on which the pairwise logratios will be constructed and selected
`y`	The response variable: a numeric variable for regression (default)), a binary factor for logistic regression or a numeric count for Poisson regression
`method`	The selection method: 1 (unrestricted selection of logratios), 2 (restricted to non-overlapping parts), 3 (additive logratios)
`family`	The distribution used in the generalized linear model family: "gaussian" (default, for multiple regression), "binomial" (for logistic regression of binary response), or "poisson" (for Poisson regression)
`nsteps`	The maximum number of steps taken, by default one less than the number of parts
`top`	When one step is taken (nsteps=1), the ordered list of top logratios with the highest improvements in the likelihood function, for selection based on domain knowledge
`previous`	For specifying variable(s) to be included before stepwise selection takes place; these can be non-compositional variables and/or specific pairwise logratios computed in previous runs of STEPR or by hand; the matrix (or vector for a single variable) of values must be supplied
`criterion`	Criterion for stopping the stepwise selection: "Bonferroni" (default), "AIC", "BIC", or NA for no stopping until maximum specified or permissible logratios entered
`alpha`	Overall significance level (default is 0.05)
`previousparts`	(For method 2) The sequence numbers of the logratios, if any, forced in using the previous option
`denom`	(For method 3) The sequence number of the part used in denominator; for use when additive logratios are forced in using previous option or to select a set of additive logratios with specific reference from the start

Details

The function `STEPR` performs stepwise selection of pairwise logratios with the objective of explaining/predicting a response variable, in the framework of generalized linear modelling where the response can be numeric continuous (regression analysis), or a binary factor (logistic regression), or a numeric count (Poisson regression). The corresponding family option has to be indicated if the regression is logistic or Poisson. The different method options for the stepwise selection are `method = 1` (unrestricted selection of logratios, any logratios can be selected irrespective of the previous ones), `method = 2` (restricted to non-overlapping parts, each part participates at most in one logratio, so that parts in previously selected logratios are excluded in subsequent steps; logratio effects can be interpreted as under orthogonality), `method = 3` (additive logratios, only logratios with the same denominator as the first selected logratio are c onsidered; the result is an additive logratio transformation on a subcomposition) Three alternative stopping criteria can be specified, otherwise the procedure executes as many steps as the value of nsteps. These are (in increasing strictness), "AIC", "BIC" and "Bonferroni" (the default).

Value

`rationames`	Names of the selected logratios
`ratios`	The sequence numbers of the selected parts in each ratio
`logratios`	Matrix of selected logratios
`logLik`	The -2*log-likelihood sequence for the steps
`deviance`	The deviance sequence for the steps
`AIC`	The AIC sequence for the steps
`BIC`	The BIC sequence for the steps
`Bonferroni`	The Bonferroni sequence for the steps
`null.deviance`	The null deviance for the regression

(Notice that for `logLik`, `AIC`, `BIC` and `Bonferroni`, the values for one more step are given, so that the stopping point can be confirmed.)

And the following if `top > 1`:

`ratios.top`	The top ratios and the sequence numbers of their parts
`logratios.top`	The matrix of top logratios
`logLik.top`	The set of top -2*log-likelihoods
`deviance.top`	The set of top deviances
`AIC.top`	The set of top AICs
`BIC.top`	The set of top BICs
`Bonferroni.top`	The set of top Bonferronis

Author(s)

Michael Greenacre

References

Coenders, G. and Greenacre, M. (2021), Three approaches to supervised learning for compositional data with pairwise logratios. aRxiv preprint. URL:https://arxiv.org/abs/2111.08953
Coenders, G. and Pawlowsky-Glahn, V. (2020), On interpretations of tests and effect sizes in regression models with a compositional predictor. SORT, 44:201-220
Greenacre, M. (2021), Compositional data analysis, Annual Review of Statistics and its Application, 8: 271-299

See Also

ALR, STEP, glm

Examples

```
# For the fish morphometric data, first close (normalize, although not necessary)
# then loop over the 26*25/2 = 325 possible logratios stepwise
data(fish)
habitat <- fish[,2]
morph <- CLOSE(fish[,4:29])
# predict habitat binary classification from morphometric ratios
fish.step1 <- STEPR(morph, as.factor(habitat), method=1, family="binomial")
# [1] "Criterion increases when 3-th ratio enters"
fish.step1$names
# [1] "Bac/Hg" "Hw/Jl"
# perform logistic regression with selected logratios
fish.glm   <- glm(as.factor(habitat) ~ fish.step1$logratios, family="binomial")
summary(fish.glm)
fish.pred1  <- predict(fish.glm)
table(fish.pred1>0.5, habitat)
#     habitat
#        1  2
# FALSE 56 11
# TRUE   3  5
# (Thus 61/75 correct predictions)
#
# force the sex variable in at the first step before selecting logratios
# and using more strict Bonferroni default
sex <- as.factor(fish[,1])
fish.step2 <- STEPR(morph, as.factor(habitat), method=1, previous=sex, family="binomial")
# [1] "Criterion increases when 3-th ratio enters"
fish.step2$names
# [1] "Bac/Hg" "Hw/Jl"
# perform logistic regression with sex and selected logratios
fish.glm   <- glm(as.factor(habitat) ~ sex + fish.step2$logratios, family="binomial")
summary(fish.glm)
# (sex not significant)
#
# check the top 10 ratios at Step 1 to allow domain knowledge to operate
fish.step3 <- STEPR(morph, as.factor(habitat), method=1, nsteps=1, top=10, family="binomial")
cbind(fish.step3$ratios.top, fish.step3$BIC.top)
#        row col
# Bac/Hg   8  19 67.93744
# Bp/Hg    7  19 69.87134
# Jl/Hg    6  19 70.31554
# Jw/Bp    5   7 71.53671
# Jw/Jl    5   6 71.57122
# Jw/Bac   5   8 71.69294
# Fc/Hg   10  19 72.38560
# Hw/Bac   1   8 73.25325
# Jw/Fc    5  10 73.48882
# Hw/Bp    1   7 73.55621
# Suppose 5th in list, Jw/Jl (Jaw width/Jaw length), preferred at the first step
fish.step4 <- STEPR(morph, as.factor(habitat), method=1,
                    previous=fish.step3$logratios.top[,5], family="binomial")
# [1] "Criterion increases when 2-th ratio enters"
fish.step4$names
# [1] "Bac/Hg"
# So after Jw/Jl forced in only Bac/Hg enters, the best one originally
```

time *Dataset: TimeBudget*

Description

This data set consists of the average percentage breakdown of time use into six categories, for 16 countries, split by males and females.

Usage

```
data(time)
```

Format

Data matrix containing the 32 x 6 matrix. Row samples are labelled by the two-character country code and m (male) or f (female).

Source

Greenacre M., Compositional Data Analysis in Practice, Chapman & Hall / CRC, 2018.

VAR *Variance of a vector of observations, dividing by n rather than n-1*

Description

This function computes the usual variance but divides by n, not by n-1.

Usage

```
VAR(x)
```

Arguments

`x`	Vector of values for which variance is computed

Details

To think of each of n observations weighted by 1/n this function `VAR` computes squared deviations from the mean and averages them. Thus, the sum of squared deviations is divided by n rather than by n-1, as for the unbiased estimate of the variance.

Value

The value of the variance.

Author(s)

Michael Greenacre

References

Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC.

See Also

`LR.VAR`, `CLOSE`

Examples

```
data(cups)
cups <- CLOSE(cups)

# variances using base R function var
apply(cups, 2, var)

# variances using easyCODA function VAR
apply(cups, 2, VAR)
```

veg *Dataset: Vegetables*

Description

This data set consists of the protein, carbohydrate and fat compositions of 10 different vegetables. Compositions are expressed as percentages.

Usage

```
data(veg)
```

Format

Data frame containing the 10 x 3 matrix.

Source

US Department of Agriculture, https://ndb.nal.usda.gov/ndb/nutrients/index

WARD *Ward clustering of a compositional data matrix*

Description

This function clusters the rows (or the columns, if the matrix is transformed) of a compositional data matrix, using weighted Ward clustering of the logratios.

Usage

```
WARD(LRdata, weight=TRUE, row.wt=NA)
```

Arguments

`LRdata`	Matrix of logratios, either a vector or preferably the logratio object resulting from one of the functions ALR, CLR, PLR or LR (usually CLRs will be used))
`weight`	`TRUE` (default) for weighted analysis (in which case weights are in the logratio object), `FALSE` for unweighted analysis, or a vector of user-defined column weights
`row.wt`	Optional set of row weights (default is equal weights when `row.wt=NA`)

Details

The function `WARD` performs a weighted WARD hierarchical clustering on the rows of an input set of logratios, usually CLR-transformed. (This would be equivalent to performing the clustering on all pairwise logratios). If the columns of the logratio matrix are unweighted, specify the option `weight=FALSE`: they will then get equal weights. The default `weight=TRUE` option implies that column weights are provided, either in the input list object LRdata, as `LRdata$LR.wt`, or as a vector of user-specified weights using the same `weight` option.

Value

An object which describes the tree produced by the clustering process on the n objects. The object is a list with components:

`merge`	an n-1 by 2 matrix. Row i of `merge` describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then observation -j was merged at this stage. If j is positive then the merge was with the cluster formed at the (earlier) stage j of the algorithm. Thus negative entries in `merge` indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
`height`	a set of n-1 real values (non-decreasing for ultrametric trees). The clustering height: that is, the value of the criterion associated with the clustering method for the particular agglomeration.
`order`	a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches

Author(s)

Michael Greenacre

References

Greenacre, M. (2018), Compositional Data Analysis in Practice, Chapman & Hall / CRC.

See Also

`hclust`, `CLR`, `LR.VAR`, `CLOSE`

Examples

```
# clustering steps for unweighted and weighted logratios
# for both row- and column-clustering
data(cups)
cups <- CLOSE(cups)

# unweighted logratios: clustering samples
cups.uclr  <- CLR(cups, weight=FALSE)
cups.uward <- WARD(cups.uclr, weight=FALSE)   # weight=FALSE not needed here,
                                              # as equal weights are in object
```

```
plot(cups.uward)
# add up the heights of the nodes
sum(cups.uward$height)
# [1] 0.02100676
# check against the total logratio variance
LR.VAR(cups.uclr, weight=FALSE)
# [1] 0.02100676

# unweighted logratios: clustering parts
tcups <- t(cups)
tcups.uclr  <- CLR(tcups, weight=FALSE)
tcups.uward <- WARD(tcups.uclr, weight=FALSE)  # weight=FALSE not needed here,
                                               # as equal weights are in object
plot(tcups.uward, labels=colnames(cups))
sum(tcups.uward$height)
# [1] 0.02100676
LR.VAR(tcups.uclr, weight=FALSE)
# [1] 0.02100676

# weighted logratios: clustering samples
cups.clr <- CLR(cups)
cups.ward <- WARD(cups.clr)
plot(cups.ward)
sum(cups.ward$height)
# [1] 0.002339335
LR.VAR(cups.clr)
# [1] 0.002339335

# weighted logratios: clustering parts
# weight=FALSE is needed here, since we want equal weights
# for the samples (columns of tcups)
tcups.clr  <- CLR(tcups, weight=FALSE)
tcups.ward <- WARD(tcups.clr, row.wt=colMeans(cups))
plot(tcups.ward, labels=colnames(cups))
   sum(tcups.ward$height)
# [1] 0.002339335
LR.VAR(tcups.clr, row.wt=colMeans(cups))
# [1] 0.002339335
```

Index

∗ **coding**
    DUMMY, 16
∗ **datasets**
    cups, 14
    fish, 18
    time, 44
    veg, 46
∗ **logratio**
    ALR, 6
    CLR, 13
    ILR, 18
    invALR, 19
    invCLR, 20
    invSLR, 21
    LR, 22
    LR.VAR, 23
    PLR, 34
    SLR, 37
∗ **multivariate**
    ACLUST, 4
    CA, 9
    FINDALR, 16
    LRA, 25
    PCA, 27
    PLOT.CA, 29
    PLOT.LRA, 30
    PLOT.PCA, 31
    PLOT.RDA, 33
    RDA, 35
    STEP, 39
    STEPR, 41
    WARD, 46
∗ **package**
    easyCODA-package, 2
∗ **variance**
    VAR, 45
∗ **visualization**
    BAR, 7
    CIplot_biv, 10
    DOT, 14

ACLUST, 4
ALR, 6, *13*, *17*, *19–24*, *35*, *38*, *40*, *43*

BAR, 7, *15*

CA, 9, *30*
ca, *4*
CIplot_biv, 10
CLOSE, *6*, 12, *45*, *47*
CLR, *6*, *7*, 13, *20–24*, *35*, *37*, *38*, *40*, *47*
cups, 14

DOT, *8*, *11*, 14
DUMMY, 16, *37*

easyCODA-package, 2

FINDALR, 16
fish, 18

glm, *43*

hclust, *6*, *47*

ILR, 18, *35*, *38*
invALR, *7*, *13*, 19, *21–24*
invCLR, *7*, *13*, *20*, 20, *23*, *24*
invSLR, 21

LR, *7*, *13*, *19*, *20*, 22, *24*, *35*, *37*, *38*, *40*
LR.VAR, *6*, *7*, *13*, *20–23*, 23, *45*, *47*
LRA, 25

PCA, 27
PLOT.CA, *10*, 29
plot.ca, *10*, *26*, *28*, *30–32*
PLOT.LRA, 30
PLOT.PCA, 31
PLOT.RDA, 33, *37*, *40*
PLR, *19*, 34, *38*

print.ca, *10*, *26*, *28*

RDA, *34*, 35

SLR, *19*, *35*, 37
STEP, 39, *43*
STEPR, 41
summary.ca, *10*, *26*, *28*

time, 44

VAR, 45
veg, 46

WARD, *6*, 46