gsi.add *Internal functions: Parallel operations of single and multiple datasets*

Description

The given operations are performed in parallel for multiple datasets or for two single datasets, or for multiple datasets with a single dataset.

Usage

```
gsi.add(x,y)
gsi.sub(x,y)
gsi.mul(x,y)
gsi.div(x,y)
```

Arguments

x	a vector or a matrix
y	a vector or a matrix

Details

All operations +,-,*,/ are performed on unclassed objects.

Value

a vector or a matrix with the operated values

Note

It is better not to use gsi.* functions directly since they are internal functions of the package

See Also

Examples

```
tmp1 <- matrix(1:12,ncol=3)
tmp2 <- 1:3
gsi.add(tmp1,tmp2)
gsi.sub(tmp1,tmp2)
gsi.mul(tmp1,tmp2)
gsi.div(tmp1,tmp2)

gsi.add(tmp2,tmp2)
gsi.sub(tmp2,tmp2)
gsi.mul(tmp2,tmp2)
gsi.div(tmp2,tmp2)

gsi.add(tmp1,tmp1)
gsi.sub(tmp1,tmp1)
gsi.mul(tmp1,tmp1)
gsi.div(tmp1,tmp1)
```

`clr` *Centered log ratio transform*

Description

Compute the centered log ratio transform of a (dataset of) composition(s) and its inverse.

Usage

```
clr( x )
clr.inv( z )
```

Arguments

`x`	a composition or a data matrix of compositions, not necessarily closed
`z`	the clr-transform of a composition or a data matrix of clr-transforms of compositions, not necessarily centered (i.e. summing up to zero)

Details

The clr-transform maps a composition in the D-part Aitchison-simplex isometrically to a D-1 dimensonal euclidian vector subspace: consequently, the transformation is not injective and only yields vectors which elements sum up to 0. Thus resulting covariance matrices are always singular.

The data can then be analysed in this transformation by all classical multivariate analysis tools not relying on a full rank of the covariance. See `ilr` and `alr` for alternatives. The interpretation of the results is relatively easy since the relation between each original part and a transformed variable is preserved.

The centered logratio transform is given by

$$clr(x) := \left(\ln x_i - \frac{1}{D} \sum_{j=1}^{D} \ln x_j \right)_i$$

The image of the `clr` is given by the vectors with entries summing to 0. This hyperplane is also called the clr-plane.

Value

`clr` gives the centered log ratio transform, `clr.inv` gives closed compositions with the given clr-transforms

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

`ilr`,`alr`,`apt`

Examples

```
(tmp <- clr(c(1,2,3)))
clr.inv(tmp)
clr.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(clr(cdata))
```

`gsi.diagGenerate` *Internal functions: Generate a diagonal matrix*

Description

Generate a diagonal matrix from a vector of the diagonal entries like.

Usage

```
gsi.diagGenerate(x)
```

Arguments

`x`	a vector

Details

The difference to original `diag` is that it always gives a diagonal matrix and does nothing flawed in case of a length one vector.

Value

a diagonal matrix.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi.diagExtract, diag

Examples

```
diag(1:3)
gsi.diagGenerate(1:3)
gsi.diagGenerate(3)
diag(3)
```

---

cor.acomp   *Correlations of amounts and compositions*

Description

Computes the correlation matrix in the various approaches of compositional and amount data analysis.

Usage

```
          cor(x,y=NULL,...)
          ## Default S3 method:
          cor(x, y=NULL, use="all.obs", method=c("pearson",
    "kendall", "spearman"),...)
          ## S3 method for class 'acomp':
          cor(x,y=NULL,...)
          ## S3 method for class 'rcomp':
          cor(x,y=NULL,...)
          ## S3 method for class 'aplus':
          cor(x,y=NULL,...)
          ## S3 method for class 'rplus':
          cor(x,y=NULL,...)
          ## S3 method for class 'rmult':
          cor(x,y=NULL,...)
```

Arguments

`x`	a dataset, eventually of amounts or compositions
`y`	a second dataset, eventually of amounts or compositions
`use`	see cor
`method`	see cor
`...`	further arguments to cor e.g. `use`

Details

The correlation matrix does not make much sense for compositions.

In R versions older than v2.0.0, `cor` was defined in package “base” instead of in “stats”. This might produce some misfunction.

Value

The correlation matrix.

See Also

`var.acomp`

Examples

```
data(SimulatedAmounts)
mean.col(sa.lognormals)
cor(acomp(sa.lognormals5[,1:3]),acomp(sa.lognormals5[,4:5]))
cor(rcomp(sa.lognormals5[,1:3]),rcomp(sa.lognormals5[,4:5]))
cor(aplus(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
cor(rplus(sa.lognormals5[,1:3]),rplus(sa.lognormals5[,4:5]))
cor(acomp(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
```

`segments` *Draws straight lines from point to point.*

Description

The function draws lines from a points x to a point y in the given geometry.

Usage

```
## S3 method for class 'acomp':
segments(x0,y,...,steps=30)
## S3 method for class 'rcomp':
segments(x0,y,...,steps=30)
## S3 method for class 'aplus':
segments(x0,y,...,steps=30)
## S3 method for class 'rplus':
segments(x0,y,...,steps=30)
## S3 method for class 'rmult':
segments(x0,y,...,steps=30)
```

Arguments

`x0`	dataset of points of the given type to draw the line from
`y`	dataset of points of the given type to draw the line to
`...`	further graphical parameters
`steps`	the number of discretisation points to draw the segments not straight on the monitor.

Details

The functions add lines to the graphics generated with the corresponding plot functions.

Adding to multipaneled plots, redraws the plot completely and is only possible, when the plot has been created with the plotting routines from this library.

See Also

`plot.acomp`,`lines.acomp`

Examples

```
data(SimulatedAmounts)

plot(acomp(sa.lognormals))
segments.acomp(acomp(c(1,2,3)),acomp(c(2,3,1)),col="red")
segments.rcomp(acomp(c(1,2,3)),acomp(c(2,3,1)),col="blue")

plot(aplus(sa.lognormals[,1:2]))
segments.aplus(aplus(c(10,20)),aplus(c(20,10)),col="red")
segments.rplus(rplus(c(10,20)),rplus(c(20,10)),col="blue")

plot(rplus(sa.lognormals[,1:2]))
segments.aplus(aplus(c(10,20)),aplus(c(20,10)),col="red")
segments.rplus(rplus(c(10,20)),rplus(c(20,10)),col="blue")
```

`matmult` *inner product for matrices and vectors*

Description

Multiplies two matrices, if they are conformable. If one argument is a vector, it will be coerced to either a row or a column matrix to make the two arguments conformable. If both are vectors it will return the inner product.

Usage

```
x %*% y
## S3 method for class 'default':
x %*% y
```

Arguments

`x,y`	numeric or complex matrices or vectors

Details

This is a copy of the `%*%` function. The function is made generic to allow the definition of specific methods.

Value

The matrix product. Uses 'drop' to get rid of dimensions which have only one level.

See Also

`%*%.rmult`

Examples

```
M <- matrix(c(
0.2,0.1,0.0,
0.1,0.2,0.0,
0.0,0.0,0.2),byrow=TRUE,nrow=3)
x <- c(1,1,2)
M %*% x
x %*% M
x %*% x
M %*% M
t(x) %*% M
```

`cpt` *Centered planar transform*

Description

Compute the centered planar transform of a (dataset of) compositions and its inverse.

Usage

```
cpt( x )
cpt.inv( z )
```

Arguments

`x`	a composition or a data.matrix of compositions, not necessarily closed
`z`	the cpt-transform of a composition or a data matrix of cpt-transforms of compositions. It is checked that the z sum up to 0.

Details

The cpt-transform maps a composition in the D-part real-simplex isometrically to a D-1 dimensional euclidian vector space, identified with a plane parallel to the simplex but passing through the origin. However the transformation is not injective and does not even reach the whole plane. Thus resulting covariance matrices are always singular.

The data can then be analysed in this transformed space by all classical multivariate analysis tools not relying on a full rank of the covariance matrix. See `ipt` and `apt` for alternatives. The interpretation of the results is relitevly easy since the relation of each transformed component to the original parts is preserved.

The centered planar transform is given by

$$cpt(x)_i := clo(x)_i - \frac{1}{D}$$

Value

`cpt` gives the centered planar transform, `cpt.inv` gives closed compositions with the given cpt-transforms.

References

See Also

`clr`,`apt`,`ipt`

Examples

```
(tmp <- cpt(c(1,2,3)))
cpt.inv(tmp)
cpt.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(cpt(cdata))
```

as.data.frame — *Convert ”compositions” classes to data frames*

Description

Convert a compositional vector to a dataframe

Usage

```
## S3 method for class 'acomp':
as.data.frame(x,...)
## S3 method for class 'rcomp':
as.data.frame(x,...)
## S3 method for class 'aplus':
as.data.frame(x,...)
## S3 method for class 'rplus':
as.data.frame(x,...)
## S3 method for class 'rmult':
as.data.frame(x,...)
```

Arguments

`x`	an object to be converted to a dataframe
`...`	additional arguments are not used

Value

a dataframe containing the given data.

Examples

```
data(SimulatedAmounts)
as.data.frame(acomp(sa.groups))
data.frame(acomp(sa.groups),groups=sa.groups.area)
```

gsi.addclass — *Internal function: give an object a derived subclass*

Description

This function just extends the class of an object by the given class.

Usage

```
gsi.addclass( x , cls)
```

Arguments

`x`	The object
`cls`	the new additional class

Value

The object x with additional class attached.

Note

Do not use gsi.* functions directly since they are internal functions of the package

Examples

```
gsi.addclass(1:10,"goofy")
```

`gsi.drop` *Internal functions: A conditional drop*

Description

drop, if drop is needed.

Usage

```
gsi.drop(X,drop)
```

Arguments

`X`	an array needing dimensions dropped
`drop`	a logical whether to drop dimensions

Details

Value

`X` or `drop(X)`

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`drop`, `gsi`

Examples

`norm` *Vector space norm*

Description

Each of the considered space structures has an associated norm, which is computed for each element by these functions.

Usage

```
norm(x,...)
## Default S3 method:
norm(x,...)
## S3 method for class 'acomp':
norm(x,...)
## S3 method for class 'rcomp':
norm(x,...)
## S3 method for class 'aplus':
norm(x,...)
## S3 method for class 'rplus':
norm(x,...)
## S3 method for class 'rmult':
norm(x,...)
```

Arguments

`x`	a dataset or a single vector of some type
`...`	currently not used, intended to select a different norm

Value

The norms of the given vectors.

See Also

`normalize`

Examples

```
data(SimulatedAmounts)
tmp <- acomp(sa.lognormals)
mvar(tmp)
sum(norm( tmp - mean(tmp) )^2)/(nrow(tmp)-1)
```

`powerofpsdmatrix` *power transform of a matrix*

Description

Computes a power of a positive semidefinite symmetric matrix.

Usage

```
powerofpsdmatrix( M , p,...)
```

Arguments

`M`	a Matrix, preferably symmetric
`p`	a single number giving the power
`...`	further arguments to the singular value decomposition

Details

for a symmetric matrix the computed result can actually be considered as a version of the given power of the matrix fullfilling the relation:

$$M^p M^q = M^{p+q}$$

The symmetry of the matrix is not checked.

Value

`U%*% D^p %*% t(P)` where the `UDP` is a singular value decomposition of M.

References

See Also

Examples

```
data(SimulatedAmounts)
d <- ilr(sa.lognormals)
var( d %*% powerofpsdmatrix(var(d),-1/2)) # Unit matrix
```

princomp.acomp — *Principal component analysis for Aitchison compositions*

Description

A principal component analysis is done in the Aitchison geometry (i.e. clr-transform) of the simplex. Some gimics simplify the interpretation of the computed components as compositional perturbations.

Usage

```
## S3 method for class 'acomp':
princomp(x,...,scores=TRUE)
## S3 method for class 'princomp.acomp':
print(x,...)
## S3 method for class 'princomp.acomp':
plot(x,y=NULL,...,
npcs=min(10,length(x$sdev)),
type=c("screeplot","variance","biplot","loadings","relative"),
main=NULL,
scale.sdev=1)
## S3 method for class 'princomp.acomp':
predict(object,newdata,...)
```

Arguments

`x`	a acomp-dataset (in princomp) or a result from princomp.acomp
`y`	not used
`scores`	a logical indicating whether scores should be computed or not
`npcs`	the number of components to be drawn in the scree plot
`type`	type of the plot: `"screeplot"` is a lined screeplot, `"variance"` is a boxplot like screeplot, `"biplot"` is a biplot, `"loadings"` displayes the Loadings as a `barplot.acomp`
`scale.sdev`	the multiple of sigma to use plotting the loadings
`main`	headline of the plot
`object`	a fitted princomp.acomp object
`newdata`	another compositional dataset of class acomp
`...`	further arguments to pass to internally-called functions

Details

As a metric euclidean space the Aitchison simplex has its own principal component analysis, that should be performed in terms of the covariance matrix and not in terms of the meaningless correlation matrix.
To aid the interpretation we added some extra functionality to a normal `princomp(clr(x))`.

First of all the result contains as additional information the compositional representation of the returned vectors in the space of the data: the center as a composition `Center`, and the loadings in terms of a composition to perturbe with positively (`Loadings`) or negatively (`DownLoadings`). The Up- and DownLoadings are normalized to the number of parts in the simplex and not to one to simplify the interpretation. A value of about one means no change in the specific component. To avoid confusion the meaningless last principal component is removed.
The plot routine provides screeplots (`type = "s"`,`type= "v"`), biplots (`type = "b"`), plots of the effect of loadings (`type = "b"`) in `scale.sdev*sdev`-spread, and loadings of pairwise (log-)ratios (`type = "r"`).
The interpretation of a screeplot does not differ from ordinary screeplots. It shows the eigenvalues of the covariance matrix, which represent the portions of variance explained by the principal components.
The interpretation of the biplot strongly differs from one. The relevant variables are not the drawn arrows of the components, but rather the links or differences between two arrow-heads, which can be interpreted as log-ratios between the two components represented by the arrows.
The compositional loading plot is introduced with this package. The loadings of all component can be seen as an orthogonal basis in the space of clr-transformed data. These vectors are displayed by a barplot with their corresponding composition. For a better interpretation the total of these compositons is set to the number of parts in the composition, such that a portion of one means no effect. This is similar to (but not exactly the same as) a zero loading in a real principal component analysis.
The loadings plot can work in two different modes: if `scale.sdev` is set to `NA` it displays the composition beeing represented by the unit vector of loadings in the clr-transformed space. If `scale.sdev` is numeric we use this composition scaled by the standard deviation of the respective component.
The relative plot displays the `relativeLoadings` as a barplot. The deviation from a unit bar shows the effect of each principal component on the respective ratio.

Value

`princomp` gives an object of type `c("princomp.acomp","princomp")` with the following content:

`sdev`	the standard deviation of the principal components
`loadings`	the matrix of variable loadings (i.e., a matrix which columns contain the eigenvectors). This is of class `"loadings"`. The last eigenvector is removed since it should contain the irrelevant scaling.
`center`	the clr-transformed vector of means used to center the dataset
`Center`	the `acomp` vector of means used to center the dataset
`scale`	the scaling applied to each variable
`n.obs`	number of observations
`scores`	if `scores = TRUE`, the scores of the supplied data on the principal components and the information was available. Scores are coordinates in a basis given by the principal components and thus not compositions
`call`	the matched call

`na.action`	not clearly understood
`Loadings`	compositions that represent a perturbation with the vectors represented by the loadings of each of the factors
`DownLoadings`	compositions that represent a perturbation with the inverse of the vectors represented by the loadings of each of the factors

`predict` returns a matrix of scores of the observations in the `newdata` dataset
. The other routines are mainly called for their side effect of plotting or printing and return the object `x`.

References

Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

Aitchison, J. and M. Greenacre (2002) *Biplots for Compositional Data* Journal of the Royal Statistical Society, Series C (Applied Statistics) **51** (4) 375-392

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

http://ima.udg.es/Activitats/CoDaWork03

http://ima.udg.es/Activitats/CoDaWork05

See Also

`clr`,`acomp`, `relativeLoadings` `princomp.aplus`, `princomp.rcomp`, `barplot.acomp`, `mean.acomp`, `var.acomp`

Examples

```
data(SimulatedAmounts)
pc <- princomp(acomp(sa.lognormals5))
pc
summary(pc)
plot(pc)           #plot(pc,type="screeplot")
plot(pc,type="v")
plot(pc,type="biplot")
plot(pc,choice=c(1,3),type="biplot")
plot(pc,type="loadings")
plot(pc,type="loadings",scale.sdev=-1) # Downward
plot(pc,type="relative",scale.sdev=NA) # The directions
plot(pc,type="relative",scale.sdev=1) # one sigma Upward
plot(pc,type="relative",scale.sdev=-1) # one sigma Downward
biplot(pc)
screeplot(pc)
loadings(pc)
```

```
relativeLoadings(pc,mult=FALSE)
relativeLoadings(pc)
relativeLoadings(pc,scale.sdev=1)
relativeLoadings(pc,scale.sdev=2)

pc$Loadings
pc$DownLoadings
barplot(pc$Loadings)
pc$sdev^2
cov(predict(pc,sa.lognormals5))
```

`geometricmean` *The geometric mean*

Description

Computes the geometric mean.

Usage

```
geometricmean(x,...)
geometricmean.row(x,...)
geometricmean.col(x,...)
```

Arguments

`x`	a numeric vector or matrix of data
`...`	further arguments to compute the mean

Details

The geometric mean is defined as:

$$geometricmean(x) := \left(\prod_{i=1}^{n} x_i \right)^{1/n}$$

The geometric mean is actually computed by `exp(mean(log(c(unclass(x))),...))`.

Value

The geometric means of x as a whole (geometricmean), its rows (geometricmean.row) or its columns (geometricmean.col).

See Also

`mean.rplus`

Examples

```
geometricmean(1:10)
```

---

acomp *Aitchison compositions*

Description

A class providing the means to analyse compositions in the philosophical framework of the Aitchison Simplex.

Usage

```
acomp(X,parts=1:NCOL(oneOrDataset(X)),total=1)
```

Arguments

`X`	composition or dataset of compositions
`parts`	vector containing the indices xor names of the columns to be used
`total`	the total amount to be used, typically 1 or 100

Details

Many multivariate datasets essentially describe amounts of D different parts in a whole. This has some important implications justifying to regard them as a scale for its own, called a composition. This scale was in-depth analysed by Aitchison (1986) and the functions around the class `"acomp"` follow his approach.
Compositions have some important properties: Amounts are always positive. The amount of every part is limited to the whole. The absolute amount of the whole is noninformative since it is typically due to artifacts on the measurement procedure. Thus only relative changes are relevant. If the relative amount of one part increases, the amounts of other parts must decrease, introducing spurious anticorrelation (Chayes 1960), when analysed directly. Often parts (e.g H2O, Si) are missing in the dataset leaving the total amount unreported and longing for analysis procedures avoiding spurious effects when applied to such subcompositions. Furthermore, the result of an analysis should be indepent of the units (ppm, g/l, vol.%, mass.%, molar fraction) of the dataset.
From these properties Aitchison showed that the analysis should be based on ratios or log-ratios only. He introduced several transformations (e.g. `clr`,`alr`), operations (e.g. `perturbe`, `power.acomp`), and a distance (`dist`) which are compatible with these properties. Later it was found that the set of compostions equiped with perturbations as addition and powertransform as scalar multiplication and the `dist` as distance form a D-1 dimensional euclidean vector space (Billheimer, Fagan and Guttorp, 2001), which can be mapped isometrically to a usual real vector space by `ilr` (Pawlowsky-Glahn and Egozcue, 2001).
The general approach in analysing acomp objects is thus to performe classical multivariate analysis on clr/alr/ilr-transformed coordinates and to backtransform or display the results

in such a way that they can be interpreted in terms of the original compositional parts. A side effect of the procedure is to force the compositions to sum up to a *total*, which is done by the closure operation `clo` .

Value

a vector of class `"acomp"` representing one closed composition or a matrix of class `"acomp"` representing multiple closed compositions each in one row.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

Billheimer, D., P. Guttorp, W.F. and Fagan (2001) Statistical interpretation of species composition, *Journal of the American Statistical Association*, **96** (456), 1205-1214

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

Pawlowsky-Glahn, V. and ??? (2003) ???

http://ima.udg.es/Activitats/CoDaWork03

http://ima.udg.es/Activitats/CoDaWork05

See Also

`clr`,`rcomp`, `aplus`, `princomp.acomp`, `plot.acomp`, `boxplot.acomp`, `barplot.acomp`, `mean.acomp`, `var.acomp`, `variation.acomp`, `cov.acomp`, `msd`

Examples

```
data(SimulatedAmounts)
plot(acomp(sa.lognormals))
```

`rcomparithm` *Arithmetic operations for composition in real geometry*

Description

The real compositions form a manifold of real vector space. The induced operations +,-,*,/ give results valued in the real vector space, but possibly outside the simplex.

Usage

```
convex.rcomp(x,y,alpha=0.5)
## Methods for class "rcomp"
##   x+y
##   x-y
##   -x
##   x*r
##   r*x
##   x/r
```

Arguments

`x`	an rcomp composition or dataset of compositions
`y`	an rcomp composition or dataset of compositions
`r`	a numeric vector of size 1 or nrow(x)
`alpha`	a numeric vector of size 1 or nrow(x) with values between 0 and 1

Details

The functions behave quite like `+.rmult`.
The convex combination is defined as: `x*alpha + (1-alpha)*y`

Value

`rmult`-objects containing the given operations on the simplex as subset of the R^D. Only the convex combination `convex.rcomp` results in an `rcomp`-object again, since only this operation is closed.

Note

For * the arguments x and y can be exchanged.

See Also

`+.rmult`, `+.acomp`,`cpt`, `rcomp`, `rmult`

Examples

```
rcomp(1:5)* -1 + rcomp(1:5)
data(SimulatedAmounts)
cdata <- rcomp(sa.lognormals)
plot( tmp <- (cdata-mean(cdata))/msd(cdata) )
class(tmp)
mean(tmp)
msd(tmp)
var(tmp)
plot(convex.rcomp(rcomp(c(1,1,1)),sa.lognormals,0.1))
```

`Hydrochem` *Hydrochemical composition data set of Llobregat river basin water (NE Spain)*

Description

Contains the hydrochemical compositional data set obtained from several rivers in the Llobregat river basin, in northeastern Spain.

Usage

```
data(Hydrochem)
```

Format

Data matrix with 485 cases and 19 variables.

Details

This hydrochemical data set contains measurements of 14 components, H, Na, K, Ca, Mg, Sr, Ba, NH_4, Cl, HCO_3, NO_3, SO_4, PO_4, TOC. From them, hydrogen was derived by inverting the relationship between its stable form in water, H_3O^+, and pH. Details can be found in Otero et al. (2005). Each of these parameters is measured approximately once each month during 2 years in 31 stations, placed along the rivers and main tributaries of the Llobregat river, one of the medium rivers in northeastern Spain.

The Llobregat river drains an area of 4948.2 km^2, and it is 156.6 km long, with two main tributaries, Cardener and Anoia. The headwaters of Llobregat and Cardener are in a rather unpolluted area of the Eastern Pyrenees. Mid-waters these rivers flow through a densely populated and industrialized area, where potash mining activity occurs and there are large salt mine tailings stored with no water proofing. There, the main land use is agriculture and stockbreeding. The lower course flows through one of the most densely populated areas of the Mediterranean region (around the city of Barcelona) and the river receives large inputs from industry and urban origin, while intensive agriculture activity is again present in the Llobregat delta. Anoia is quite different. Its headwaters are in an agricultural area, downwaters it flows through an industrialized zone (paper mills, tannery and textile industries), and near the confluence with Llobregat the main land use is agriculture again, mainly vineyards, with a decrease in industry and urban contribution. Given this variety in geological background and human activities, the sample has been splitted in four groups (higher Llobregat course, Cardener, Anoia and lower Llobregat course), which in turn are splitted into main river and tributaries (Otero et al, 2005). Information on these groupings, the sampling locations and sampling time is included in 5 complementary variables.

Source

The datasets are also accessible in Otero et al. (2005), and are here included under the GNU Public Library Licence Version 2 or newer.

References

Otero, N.; R. Tolosana-Delgado, A. Soler, V. Pawlowsky-Glahn and A. Canals (2005). Relative vs. absolute statistical analysis of compositions: A comparative study of surface waters of a Mediterranean river. Water Research, in press. doi: 10.1016/j.watres.2005.01.012

Examples

```
data(Hydrochem)
biplot(princomp(rplus(Hydrochem)))
biplot(princomp(rcomp(Hydrochem)))

biplot(princomp(aplus(Hydrochem)))
biplot(princomp(acomp(Hydrochem)))
```

`straight` *Draws infinite straight lines.*

Description

The function draws lines in a given direction d through points x.

Usage

```
straight(x,...)
## S3 method for class 'acomp':
straight(x,d,...,steps=30)
## S3 method for class 'rcomp':
straight(x,d,...,steps=30)
## S3 method for class 'aplus':
straight(x,d,...,steps=30)
## S3 method for class 'rplus':
straight(x,d,...,steps=30)
## S3 method for class 'rmult':
straight(x,d,...,steps=30)
```

Arguments

`x`	dataset of points of the given type to draw the line through
`d`	dataset of directions of the line
`...`	further graphical parameters
`steps`	the number of discretisation points to draw the segments not straight on the monitor.

Details

The functions add lines to the graphics generated with the corresponding plot functions.
Adding to multipaneled plots, redraws the plot completely and is only possible, when the plot has been created with the plotting routines from this library.
Lines end, when they leave the space (e.g. the simplex), which sometimes leads to the impression of premature end.

See Also

`plot.acomp`,`lines.acomp`

Examples

```
data(SimulatedAmounts)

plot(acomp(sa.lognormals))
straight(mean(acomp(sa.lognormals)),princomp(acomp(sa.lognormals))$Loadings[1,],col="red")
straight(mean(rcomp(sa.lognormals)),princomp(rcomp(sa.lognormals))$loadings[,1],col="blue")

plot(aplus(sa.lognormals[,1:2]))
straight(mean(aplus(sa.lognormals[,1:2])),princomp(aplus(sa.lognormals[,1:2]))$Loadings[1,],col="red")
straight(mean(rplus(sa.lognormals[,1:2])),princomp(rplus(sa.lognormals[,1:2]))$loadings[,1],col="blue")

plot(rplus(sa.lognormals[,1:2]))
straight(mean(aplus(sa.lognormals[,1:2])),princomp(aplus(sa.lognormals[,1:2]))$Loadings[1,],col="red")
straight(mean(rplus(sa.lognormals[,1:2])),princomp(rplus(sa.lognormals[,1:2]))$loadings[,1],col="blue")
```

`ratioLoadings` *Loadings of relations of two amounts*

Description

In a compositional dataset the relation of two objects can be interpreted better than a single amount. These functions compute, display and plot the corresponding pair-information for the various principal component analysis results.

Usage

```
relativeLoadings(x,...)
## S3 method for class 'princomp.acomp':
relativeLoadings(x,...,log=FALSE,scale.sdev=NA,cutoff=0.1)
## S3 method for class 'princomp.aplus':
relativeLoadings(x,...,log=FALSE,scale.sdev=NA,cutoff=0.1)
## S3 method for class 'princomp.rcomp':
relativeLoadings(x,...,scale.sdev=NA,cutoff=0.1)
## S3 method for class 'princomp.rplus':
```

```
relativeLoadings(x,...,scale.sdev=NA,cutoff=0.1)
## S3 method for class 'relativeLoadings.princomp.acomp':
print(x,...,cutoff=attr(x,"cutoff"),
                                                 digits=2
                                                 )
## S3 method for class 'relativeLoadings.princomp.aplus':
print(x,...,cutoff=attr(x,"cutoff"),
                                                 digits=2
                                                 )
## S3 method for class 'relativeLoadings.princomp.rcomp':
print(x,...,cutoff=attr(x,"cutoff"),
                                                 digits=2
                                                 )
## S3 method for class 'relativeLoadings.princomp.rplus':
print(x,...,cutoff=attr(x,"cutoff"),
                                                 digits=2
                                                 )
## S3 method for class 'relativeLoadings.princomp.acomp':
plot(x,...)
## S3 method for class 'relativeLoadings.princomp.aplus':
plot(x,...)
## S3 method for class 'relativeLoadings.princomp.rcomp':
plot(x,...)
## S3 method for class 'relativeLoadings.princomp.rplus':
plot(x,...)
```

Arguments

`x`	a result from an amount PCA `princomp.acomp`/`princomp.aplus`/`princomp.rcomp`/`princomp.rp`
`log`	a logical indicating to use log-ratios instead of ratios
`scale.sdev`	If not `NA`, a number specifying the multiple of a standard deviation the component is to be multiplied with.
`cutoff`	A single number. Changes under that (log)-cutoff are not displayed.
`digits`	The number of digits to be displayed
`...`	further parameters to internally-called functions

Details

The relative loadings of components allow a direct interpretation of the effects of principal components. For acomp/aplus classes the relation is induced by a ratio, which can optionally be log-transformed. For the rcomp/rplus-classes the relation is induced by a difference, which is quite meaningless when the units are different.

Value

The value is a matrix of type `"relativeLoadings.princomp.*"`, containing the ratios in the compositions represented by the loadings (optionally scaled by the standard deviation of the components and `scale.sdev`).

See Also

princomp.acomp, princomp.aplus, barplot

Examples

```
data(SimulatedAmounts)
pc <- princomp(acomp(sa.lognormals5))
pc
summary(pc)
relativeLoadings(pc,log=TRUE)
relativeLoadings(pc)
relativeLoadings(pc,scale.sdev=1)
relativeLoadings(pc,scale.sdev=2)

plot(relativeLoadings(pc,log=TRUE))
plot(relativeLoadings(pc))
plot(relativeLoadings(pc,scale.sdev=1))
plot(relativeLoadings(pc,scale.sdev=2))
```

---

gsiinternal1 *Internal functions of the compositions package*

Description

Internal functions

Usage

```
gsi.closespread(spread)
```

Arguments

`spread`

Details

Value

like spread but projected to the orthognonal complement of `c(1,...,1)`

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi

Examples

`gsi.simshape` *Internal function: Reshape an object to the shape type of another*

Description

Reshape an object to the shape type of another

Usage

```
gsi.simshape(x,oldx)
```

Arguments

`x`	the data to be returned
`oldx`	a data of the intended shape

Details

Value

x is shaped as oldx

Note

Do not use gsi.* functions directly since they are internal functions of the package

References

See Also

Examples

```
gsi.simshape(matrix(1:4,nrow=1),1:3)
```

dist	*Distances in variouse approaches*

Description

Calculates a distance matrix from a dataset.

Usage

```
dist(x,...)
## Default S3 method:
dist(x,...)
```

Arguments

`x`	a dataset
`...`	further arguments to `dist`

Details

The distance is computed based on `cdt`

Value

a distance matrix

See Also

`aplus`

Examples

```
data(iris)
dist(iris[,1:4])
data(SimulatedAmounts)
dist(acomp(sa.lognormals),method="manhattan")
dist(rcomp(sa.lognormals))
dist(aplus(sa.lognormals))
dist(rplus(sa.lognormals))
```

`summary.aplus` *Summaries of amounts*

Description

Usage

```
## S3 method for class 'aplus':
summary( object, ... ,digits=max(3, getOption("digits")-3))
## S3 method for class 'rplus':
summary( object, ... )
## S3 method for class 'rmult':
summary( object, ... )
```

Arguments

`object`	an `aplus`/`rplus` set of amounts
`digits`	the number of significant digits to be used. The argument can also be used with rplus/rmult.
`...`	not used, only here for generics

Details

Value

A matrix containing summary statistics. The obtained value is the same as for the classical summary `summary`, although in the case of `aplus` objects, the statistics have been computed in a logarithmic geometry and exponentiated.

References

See Also

`aplus`,`rplus`,`summary.acomp`, `summary.rcomp`

Examples

```
data(SimulatedAmounts)
summary(aplus(sa.lognormals))
summary(rplus(sa.lognormals))
summary(rmult(sa.lognormals))
```

Description

Compute the metric variance, covariance, correlation or standard deviation.

Usage

```
mvar(x,...)
mcov(x,...)
mcor(x,...)
msd(x,...)
mvar.default(x,y=NULL,...)
mcov.default(x,y=x,...)
mcor.default(x,y,...)
msd.default(x,y=NULL,...)
```

Arguments

`x`	a dataset, eventually of amounts or compositions
`y`	a second dataset, eventually of amounts or compositions
`...`	further arguments to `var` or `cov` e.g. `use`

Details

The metric variance (`mvar`) is defined by the trace of the variance in the natural geometry of the data, or also by the generalized variance in natural geometry. The natural geometry is equivalently given by the `cdt` or `idt` transforms.

The metric standard deviation (`msd`) is not the square root of the metric variance, but the square root of the mean of the eigenvalues of the variance matrix. In this way it can be interpreted in units of the original natural geometry, as the diameter of a 1-sigma sperical ball around the mean.

The metric covariance (`mvar`) is the sum over the absolute singular values of the covariance of two datasets in their respective geometries. It is always positive. The metric covariance of a dataset with itself is its metric variance. The interpretation of a metric covariance is quite difficult.

The metric correlation (`mcor`) is the metric covariance of the datasets in their natural geometry normalized to unit variance matrix. It is a number between 0 and the smaller dimension of both natural spaces. A number of 1 means perfect correlation in 1 dimension, but only partial correlations in higher dimensions.

Value

a scalar number, informing of the degree of variation/covariation of one/two datasets.

References

Daunis-i-Estadella, J., J.J. Egozcue, and V. Pawlowsky-Glahn (2002) Least squares regression in the Simplex on the simplex, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

See Also

`var`, `cov`, `mean.acomp`, `acomp`, `rcomp`, `aplus`, `rplus`

Examples

```
data(SimulatedAmounts)
mvar(acomp(sa.lognormals))
mvar(rcomp(sa.lognormals))
mvar(aplus(sa.lognormals))
mvar(rplus(sa.lognormals))

msd(acomp(sa.lognormals))
msd(rcomp(sa.lognormals))
msd(aplus(sa.lognormals))
msd(rplus(sa.lognormals))

mcov(acomp(sa.lognormals5[,1:3]),acomp(sa.lognormals5[,4:5]))
mcor(acomp(sa.lognormals5[,1:3]),acomp(sa.lognormals5[,4:5]))
mcov(rcomp(sa.lognormals5[,1:3]),rcomp(sa.lognormals5[,4:5]))
mcor(rcomp(sa.lognormals5[,1:3]),rcomp(sa.lognormals5[,4:5]))

mcov(aplus(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
mcor(aplus(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
mcov(rplus(sa.lognormals5[,1:3]),rplus(sa.lognormals5[,4:5]))
mcor(rplus(sa.lognormals5[,1:3]),rplus(sa.lognormals5[,4:5]))

mcov(acomp(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
mcor(acomp(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
```

`boxplot` *Displaying compositions and amounts by boxplots*

Description

For the different interpretations of amount data a different type of boxplot is feasible. Thus different boxplots are drawn.

Usage

```
## S3 method for class 'acomp':
boxplot(x,fak=NULL,...,
                         xlim=x.lim,ylim=c(minq,maxq),log=TRUE,
                         panel=vp.logboxplot,dots=!boxes,boxes=TRUE)
## S3 method for class 'rcomp':
boxplot(x,fak=NULL,...,
                         xlim=x.lim,ylim=c(0,1),log=FALSE,
                         panel=vp.boxplot,dots=!boxes,boxes=TRUE)
## S3 method for class 'aplus':
boxplot(x,fak=NULL,...,log=TRUE)
## S3 method for class 'rplus':
boxplot(x,fak=NULL,...,ylim=c(0,max(x)),log=FALSE)
vp.boxplot(x,y,...,dots=FALSE,boxes=TRUE,xlim,ylim,log,notch=FALSE)
vp.logboxplot(x,y,...,dots=FALSE,boxes=TRUE,xlim,ylim,log,notch=FALSE)
```

Arguments

`x`	a dataset
`fak`	a factor to split the dataset, not yet implemented in aplus and rplus
`xlim`	x-limits of the plot
`ylim`	y-limits of the plot
`log`	logical indicating whether ploting should be done on log scale
`panel`	the panel function to be used or a list of multiple panel functions
`...`	further graphical parameters
`dots`	a logical indicating whether the points should be drawn
`boxes`	a logical indicating the boxes should be drawn
`y`	used by pairs
`notch`	should the boxes be notched

Details

`vp.boxplot` and `vp.logboxplot` are only used as panel functions.

See Also

`plot.acomp`, `qqnorm.acomp`

Examples

```
data(SimulatedAmounts)
boxplot(acomp(sa.lognormals))
boxplot(rcomp(sa.lognormals))
boxplot(aplus(sa.lognormals))
boxplot(rplus(sa.lognormals))
```

ellipses	*Draw ellipses*

Description

Draws ellipses from a mean and a variance into a plot.

Usage

```
ellipses(mean,...)
## S3 method for class 'acomp':
ellipses(mean,var,r=1,...,steps=360)
## S3 method for class 'rcomp':
ellipses(mean,var,r=1,...,steps=360)
## S3 method for class 'aplus':
ellipses(mean,var,r=1,...,steps=360)
## S3 method for class 'rplus':
ellipses(mean,var,r=1,...,steps=360)
## S3 method for class 'rmult':
ellipses(mean,var,r=1,...,steps=360)
```

Arguments

`mean`	a dataset/value of means or midpoints of the ellipses
`var`	a variance matrix or a set of variance matrices given by `var[i,,]`. The principle axis of the variance give the axis of the ellipses. The the square-root of the eigenvalues times r give the large and small halfdiameter of the ellipse.
`r`	A scaling of the radius
`...`	further graphical parameters
`steps`	the numer of discretisation points to draw the ellipses.

Details

The ellipse drawn is given by the solutions of

$$(x - mean)^t var^{-1}(x - mean) = r^2$$

in the respective geometry of the parameter space. Currently the compositional scales (acomp and rcomp) can be used only for three part compositions and plot into a ternary diagram and the not sum constrained scales (aplus, rplus and rmult) can only be used with two parts and draw into any scatterplot.
TODO: Adding to multi-panel plots

See Also

`plot.acomp`,

Examples

```
data(SimulatedAmounts)

plot(acomp(sa.lognormals))
tt<-acomp(sa.lognormals); ellipses(mean(tt),var(tt),r=2,col="red")
tt<-rcomp(sa.lognormals); ellipses(mean(tt),var(tt),r=2,col="blue")

plot(aplus(sa.lognormals[,1:2]))
tt<-aplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="red")
tt<-rplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="blue")

plot(rplus(sa.lognormals[,1:2]))
tt<-aplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="red")
tt<-rplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="blue")
tt<-rmult(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="green")
```

meanrow *The arithmetic mean of rows or columns*

Description

Computes the arithmetic mean.

Usage

```
mean.row(x,..., na.action=get(getOption("na.action")))
mean.col(x,..., na.action=get(getOption("na.action")))
```

Arguments

`x`	a numeric vector or matrix of data
`...`	arguments to `mean`
`na.action`	The na.action to be used: one of `na.omit`,`na.fail`,`na.pass`

Details

Value

The arithmetic means of the rows (mean.row) or columns (mean.col) of x.

See Also

`mean.rplus`

Examples

```
data(SimulatedAmounts)
mean.col(sa.tnormals)
mean.row(sa.tnormals)
```

---

gsi.margin   *Internal function: Compute a desired compositional margin*

Description

This generic function should select the selected type of margin based on the class of the dataset and the specified margin type.

Usage

```
gsi.margin(X,...)
gsi.margin.acomp(X,what,...,margin="acomp")
gsi.margin.rcomp(X,what,...,margin="rcomp")
gsi.margin.aplus(X,what,...)
gsi.margin.rplus(X,what,...)
```

Arguments

`X`	The dataset to take the margin from.
`what`	The indices xor column names to be kept.
`margin`	The type of marginalisation to be used. Possible values are: '"sub"', '"rcomp"', '"acomp"' and an index xor a name of a variable in the dataset.
`...`	other arguments

Details

Value

Some marginal dataset or vector still containing the variables given by `what` and optionally one additional part named '"+"', '"*"' or *margin*.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi

Examples

```
data(SimulatedAmounts)
plot(gsi.margin(acomp(sa.lognormals5),c("Cd","Cu")))
```

---

barplot.acomp *Barcharts of amounts*

Description

Compositions and amounts can

Usage

```
## S3 method for class 'acomp':
barplot(height,...,legend.text=TRUE,beside=FALSE,total=1)
## S3 method for class 'rcomp':
barplot(height,...,legend.text=TRUE,beside=FALSE,total=1)
## S3 method for class 'aplus':
barplot(height,...,legend.text=TRUE,beside=TRUE)
## S3 method for class 'rplus':
barplot(height,...,legend.text=TRUE,beside=TRUE)
```

Arguments

`height`	an acomp, rcomp, aplus, or rplus object giving amounts to be displayed by a barplot
`...`	further graphical parameters as in `barplot`
`legend.text`	same as legend.text in `barplot`
`beside`	same as beside in `barplot`
`total`	The total to be used in displaying the composition, typically 1, 100 or the number of parts

Details

The functions are essentially light weighted warpers for `barplot` just adding a adequate default behavior for each of the scales.

Value

A numeric vector (or matrix, when `beside = TRUE`) giving the coordinates of all the bar midpoints drawn, as in `barchart`

See Also

`acomp`, `rcomp`, `rplus` `aplus`, `plot.acomp`, `boxplot.acomp`

Examples

```
data(SimulatedAmounts)
barplot(mean(acomp(sa.lognormals[1:10,])))
barplot(mean(rcomp(sa.lognormals[1:10,])))
barplot(mean(aplus(sa.lognormals[1:10,])))
barplot(mean(rplus(sa.lognormals[1:10,])))

barplot(acomp(sa.lognormals[1:10,]))
barplot(rcomp(sa.lognormals[1:10,]))
barplot(aplus(sa.lognormals[1:10,]))
barplot(rplus(sa.lognormals[1:10,]))
```

`princomp.rplus` *Principal component analysis for real amounts*

Description

A principal component analysis is done in real geometry (i.e. using iit-transform).

Usage

```
## S3 method for class 'rplus':
princomp(x,...,scores=TRUE)
## S3 method for class 'princomp.rplus':
print(x,...)
## S3 method for class 'princomp.rplus':
plot(x,y=NULL,...,
npcs=min(10,length(x$sdev)),
type=c("screeplot","variance","biplot","loadings","relative"),
main=NULL,
scale.sdev=1)
## S3 method for class 'princomp.rplus':
predict(object,newdata,...)
```

Arguments

`x`	an rplus-dataset (for princomp) or a result from princomp.rplus
`y`	not used
`scores`	a logical indicating whether scores should be computed or not
`npcs`	the number of components to be drawn in the scree plot
`type`	type of the plot: `"screeplot"` is a lined screeplot, `"variance"` is a boxplot like the screeplot, `"biplot"` is a biplot, `"loadings"` displays the loadings as a `barplot`
`scale.sdev`	the multiple of sigma to use plotting the loadings
`main`	headline of the plot

`object`	a fitted princomp.rplus object
`newdata`	another amount dataset of class rcomp
`...`	further arguments to pass to internally-called functions

Details

Mainly a `princomp(iit(x))` is performed.
The plot routine provides screeplots (`type = "s"`,`type= "v"`), biplots (`type = "b"`), plots of the effect of loadings (`type = "b"`) in `scale.sdev*sdev`-spread, and loadings of pairwise differences (`type = "r"`).
The interpretation of a screeplot does not differ from ordinary screeplots. It shows the eigenvalues of the covariance matrix, which represent the portions of variance explained by the principal components.
The interpretation of the these biplot uses additionally to the classical interperation biplots a compositional concept: The differences between two arrowheads can be interpreted as the shift of mass between the two components represented by the arrows.
The amount loading plot is more or less a standard loadings plot. The loadings are displayed by a barplot as positive and negative changes of amounts.
The loadings plot can work in two different modes: If `scale.sdev` is set to `NA` it displays the amount vector being represented by the unit vector of loadings in the iit-transformed space. If `scale.sdev` is numeric we use this amount vector scaled by the standard deviation of the respective component.
The relative plot displays the `relativeLoadings` as a barplot. The deviation from a unit bar shows the effect of each principal component on the respective differences.

Value

`princomp` gives an object of type `c("princomp.rcomp","princomp")` with the following content:

`sdev`	the standard deviation of the principal components
`loadings`	the matrix of variable loadings (i.e., a matrix which columns contain the eigenvectors). This is of class `"loadings"`
`Loadings`	The loadings as an `"rmult"`-object
`center`	the iit-transformed vector of means used to center the dataset
`Center`	the `rplus` vector of means used to center the dataset
`scale`	the scaling applied to each variable
`n.obs`	number of observations
`scores`	if `scores = TRUE`, the scores of the supplied data on the principal components and the information available. Scores are coordinates in a basis given by the principal components and thus not compositions
`call`	the matched call
`na.action`	not clearly understood

`predict` returns a matrix of scores of the observations in the `newdata` dataset.
. The other routines are mainly called for their side effect of plotting or printing and return the object `x`.

See Also

iit,rplus, relativeLoadings princomp.rcomp, princomp.aplus,

Examples

```
data(SimulatedAmounts)
pc <- princomp(rplus(sa.lognormals5))
pc
summary(pc)
plot(pc)           #plot(pc,type="screeplot")
plot(pc,type="v")
plot(pc,type="biplot")
plot(pc,choice=c(1,3),type="biplot")
plot(pc,type="loadings")
plot(pc,type="loadings",scale.sdev=-1) # Downward
plot(pc,type="relative",scale.sdev=NA) # The directions
plot(pc,type="relative",scale.sdev=1) # one sigma Upward
plot(pc,type="relative",scale.sdev=-1) # one sigma Downward
biplot(pc)
screeplot(pc)
loadings(pc)
relativeLoadings(pc,mult=FALSE)
relativeLoadings(pc)
relativeLoadings(pc,scale.sdev=1)
relativeLoadings(pc,scale.sdev=2)

pc$sdev^2
cov(predict(pc,sa.lognormals5))
```

`gsi.expandrcomp` *Internal function: Scaling rcomp*

Description

This functions tries to compute something similar to a scaling of an acomp object in the context of the rcomp-geometry.

Usage

```
gsi.expandrcomp(x,alpha)
```

Arguments

`x`	an rcomp object
`alpha`	a number or a numeric vector between 0 and 1

Value

an rcomp-object

Note

It is better not to use gsi.* functions directly since they are internal functions of the package

Examples

```
gsi.expandrcomp(rcomp(1:3),0.5)
```

`acomparith` *Power transform in Aitchisons simplex*

Description

The Aitchison Simplex with its two operations perturbation as + and power transform as * is a vector space. This vector space is represented by these operations.

Usage

```
power.acomp(x,s)
## Methods for class "acomp"
## x*y
## x/y
```

Arguments

x	an acomp composition or dataset of compositions (or a number or a numeric vector)
y	a numeric vector of size 1 or nrow(x)
s	a numeric vector of size 1 or nrow(x)

Details

The power transform is the basic multiplication operation of the Aitchison simplex seen as a vector space. It is defined as:

$$(x * y)_i := clo((x_i^{y_i})_i)_i$$

The division operation is just the multiplication with $1/y$.

Value

An `"acomp"` vector or matrix.

Note

For `*` the arguments x and y can be exchanged. Note that this definition generalizes the power by a scalar, since `y` or `s` may be given as a scalar, or as a vector with as many components as the composition in `acomp` `x`. The result is then a matrix where each row corresponds to the composition powered by one of the scalars in the vector.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

http://ima.udg.es/Activitats/CoDaWork03

http://ima.udg.es/Activitats/CoDaWork05

See Also

ilr,clr, alr,

Examples

```
acomp(1:5)* -1 + acomp(1:5)
data(SimulatedAmounts)
cdata <- acomp(sa.lognormals)
plot( tmp <- (cdata-mean(cdata))/msd(cdata) )
class(tmp)
mean(tmp)
msd(tmp)
var(tmp)
```

---

scale *Normalizing datasets by centering and scaling*

Description

The dataset is standardized by optional scaling and centering.

Usage

```
## S3 method for class 'acomp':
scale(x,center=TRUE, scale=TRUE)
## S3 method for class 'rcomp':
scale(x,center=TRUE, scale=TRUE)
## S3 method for class 'aplus':
scale(x,center=TRUE, scale=TRUE)
## S3 method for class 'rplus':
```

```
scale(x,center=TRUE, scale=TRUE)
## S3 method for class 'rmult':
scale(x,center=TRUE, scale=TRUE)
```

Arguments

`x`	a dataset or a single vector of some type
`center`	logical value
`scale`	logical value

Details

scaling is defined in various ways for the different data types. It is always performed as an operation in the enclosing vector space. Not in all cases an independent scaling of the different coordinates is appropriate. This is only done for rplus and rmult.

Value

a vector or data matrix, as x and with the same class, but acordingly transformed.

See Also

`split{base}`

Examples

```
  data(SimulatedAmounts)
  plot(scale(acomp(sa.groups)))
  ## Not run:
plot(scale(rcomp(sa.groups)))
## End(Not run)
  plot(scale(aplus(sa.groups)))
  ## Not run:
plot(scale(rplus(sa.groups)))
## End(Not run)
  plot(scale(rmult(sa.groups)))
```

`rplusarithm` *Arithmetik of rplus-scale*

Description

The positive quadrant forms a manifold of the real vector space. The induced operations +,-,*,/ give results valued in this real vector space (not necessarily inside the manifold).

Usage

```
mul.rplus(x,r)
## Methods for class rplus
##   x+y
##  x-y
##   -x
##   x*r
##   r*x
##   x/r
```

Arguments

`x`	an rplus composition or dataset of compositions
`y`	an rplus composition or dataset of compositions
`r`	a numeric vector of size 1 or nrow(x)

Details

The functions behave quite like `+.rmult`.

Value

`rmult`-objects containing the given operations on the rcomp manifold as subset of the R^D. Only the addition and multiplication with positive numbers are internal operation and results in an `rplus`-object again.

Note

For `*` the arguments x and y can be exchanged.

See Also

`+.rmult`, `+.acomp`,`cpt`, `rcomp`, `rmult`

Examples

```
rplus(1:5)* -1 + rplus(1:5)
data(SimulatedAmounts)
cdata <- rplus(sa.lognormals)
plot( tmp <- (cdata-mean(cdata))/msd(cdata) )
class(tmp)
mean(tmp)
msd(tmp)
var(tmp)
```

`rplus` *Amounts i.e. positive numbers analysed as objects of the real vector space*

Description

A class to analyse positive amounts in a classical (non-logarithmic) framework.

Usage

```
rplus(X,parts=1:NCOL(oneOrDataset(X)),total=NA)
```

Arguments

`X`	vector or dataset of positive numbers considered as amounts
`parts`	vector containing the indices xor names of the columns to be used
`total`	a numeric vectors giving the total amount of each dataset.

Details

Many multivariate datasets essentially describe amounts of D different parts in a whole. When the whole is large in relation to the considered parts, such that they do not exclude each other, and when the total amount of each componenten is actually determined by the phenomenon under investigation and not by sampling artifacts (such as dilution or sample preparation) then the parts can be treated as amounts rather than as a composition (cf. `rcomp`, `aplus`).
In principle, amounts are just real-scaled numbers with the single restriction that they are nonnegative. Thus they can be analysed by any multivariate analysis method. This class provides a simple access interface to do so. It tries to keep in mind the positivity property of amounts and the special point zero. However there are strong arguments why an analyis based on log-scale might be much more adapted to the problem. This log-approach is provided by the class `aplus`.

The classes rcomp, acomp, aplus, and rplus are designed in a fashion as similar as possible in order to allow direct comparison between results obtained by the different approaches. In particular, the aplus logistic transform `ilt` is mirrored by the simple identity transform `iit`. In terms of computer science, this identity mapping is actually mapping an object of type "rplus" to a class-less datamatrix.

Value

a vector of class `"rplus"` representing a vector of amounts or a matrix of class `"rplus"` representing multiple vectors of amounts, by rows.

References

See Also

iit,rcomp, aplus, princomp.rplus, plot.rplus, boxplot.rplus, barplot.rplus, mean.rplus, var.rplus, variation.rplus, cov.rplus, msd

Examples

```
data(SimulatedAmounts)
plot(rplus(sa.lognormals))
```

`acompscalarproduct` *inner product for datasets with a vector space structure*

Description

acomp and aplus objects are considered as (sets of) vectors. The `%*%` is considered as the inner multiplication. An inner multiplication with another vector is the scalar product. An inner multiplication with a matrix is a matrix multiplication, where the vectors are either considered as row or as column vector.

Arguments

```
x %*% y
x %*% A
A %*% x
x %*% y
x %*% A
A %*% x
```

`x`	a acomp or aplus object
`y`	a acomp or aplus object
`A`	a matrix interpreted in clr, ilr or ilt coordinates

Details

The operators try to mimic the behavior of `%*%` on `c()`-vectors as inner product, applied in parallel to all row-vectors of the dataset. Thus the product of a vector with a vector of the same type results in the scalar product of both. For the multiplication with a matrix each vector is considered as a row or column, whatever is more appropriate. The matrix itself is considered as representing a linear mapping (endomorphism) of the vector space to a space of the same type. The mapping is represented in clr, ilr or ilt coordinates. Which of the aforementioned coordinate systems is used is judged from the type of x and from the dimensions of the A.

Value

Either a numeric vector containing the scalar products, or an object of type acomp or aplus containing the vectors transformed with the given matrix.

See Also

`%*%.rmult`

Examples

```
x <- acomp(matrix( sqrt(1:12), ncol= 3 ))
x%*%x
A <- matrix( 1:9,nrow=3)
x %*% A %*% x
x %*% A
A %*% x
A <- matrix( 1:4,nrow=2)
x %*% A %*% x
x %*% A
A %*% x
x <- aplus(matrix( sqrt(1:12), ncol= 3 ))
x%*%x
A <- matrix( 1:9,nrow=3)
x %*% A %*% x
x %*% A
A %*% x
```

`gsi.isSingleRow` *Internal function: Can something be considered as a single multivariate data item?*

Description

Checks wether something can be regarded as a single multivariate item, being a matrix or a vector, which is only a row or a column.

Usage

```
gsi.isSingleRow(X)
```

Arguments

`X`	the matrix or vector to be checked

Details

It is defined as `gsi.isSingleRow <- function(X) { return( NROW(X) == 1 || NCOL(X) ==1 ) }`

Value

a logical value

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`

Examples

```
gsi.isSingleRow(1:10)
```

`summary.rcomp` *Summary of compositions in real geometry*

Description

Compute a summary of a composition based on real geometry.

Usage

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

Arguments

`object`	an `rcomp` dataset of compositions
`...`	further arguments to `summary`

Details

The data is applied a `clo` operation before the computation.

Value

A matrix containing summary statistics. The value is the same as for the classical summary `summary` applied to a closed dataset.

See Also

`rcomp`, `summary.aplus`, `summary.acomp`

Examples

```
data(SimulatedAmounts)
summary(rcomp(sa.lognormals))
```

`plot.aplus` *Displaying amounts in scatterplots*

Description

Usage

```
## S3 method for class 'aplus':
plot(x,...,labels=colnames(X),cn=colnames(X),aspanel=FALSE,id=FALSE,idlabs=NULL,idcol=2,center=
## S3 method for class 'rplus':
plot(x,...,labels=colnames(X),cn=colnames(X),aspanel=FALSE,id=FALSE,idlabs=NULL,idcol=2,center=
## S3 method for class 'rmult':
plot(x,...,labels=colnames(X),cn=colnames(X),aspanel=FALSE,id=FALSE,idlabs=NULL,idcol=2,center=
```

Arguments

`x`	a dataset of an amount class
`...`	further graphical parameters passed (see `par`)
`add`	a logical indicating whether the information should just be added to an existing plot. In case of false a new plot is created.
`col`	The color to plot the data.
`labels`	The names of the parts
`cn`	The names of the parts to be used in a single panel. Internal use only.
`aspanel`	Logical indicating that only a single panel should be drawn and not the whole plot. Internal use only.
`id`	A logical. If true one can identify the points like with the `identify` command.
`idlabs`	A character vector providing the labels to be used with the identification, when `id=TRUE`
`idcol`	color of the idlabs-labels
`center`	a logical indicating whether a the data should be centered prior to the plot. Centering is done in the choosen philosophy. See `scale`
`scale`	a logical indicating whether a the data should be scaled prior to the plot. Scaling is done in the choosen philosophy. See `scale`

`pca`	A logical indicating whether the first principle component should be displayed in the plot. Currently direction of the principle component of the displayed subcomposition is displayed as a line. Later on a the principle componenent of the whole dataset should be displayed.
`col.pca`	The color to draw the principle component.
`logscale`	logical indication, whether logscale should be used
`xlim`	2xncol(x)-matrix giving the xlims for the columns of x
`ylim`	2xncol(x)-matrix giving the ylims for the columns of x

Details

TO DO: fix pca bug

See Also

`plot.aplus`, `qqnorm.acomp`,`boxplot.acomp`

Examples

```
data(SimulatedAmounts)
plot(aplus(sa.lognormals))
plot(rplus(sa.lognormals))
plot(aplus(sa.lognormals5))
plot(rplus(sa.lognormals5))
```

`gsi.diagExtract` *Internal functions: Get the diagonal of a matrix*

Description

Get the main diagonal of a matrix, even if the matrix is 1x1.

Usage

```
gsi.diagExtract(x)
```

Arguments

`x`	a matrix

Details

The difference to original `diag` is that it always gives the diagonal and does nothing flawed in case of a 1x1 matrix or a single number considered as such matrix.

Value

a vector containing the main diagonal entries of x.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi.diagGenerate`, `diag`

Examples

```
data(SimulatedAmounts)
gsi.diagExtract(var(acomp(sa.lognormals,c(1,2))))
gsi.diagExtract(var(ilr(acomp(sa.lognormals,c(1,2)))))
gsi.diagExtract(var(ilt(aplus(sa.lognormals,c(1)))))
```

`acompmargin` *Marginal compositions in Aitchison Compositions*

Description

Compute marginal compositions of selected parts, by computing the rest as the geometric mean of the non-selected parts.

Usage

```
acompmargin(X,d=c(1,2),name="*",pos=length(d)+1)
```

Arguments

`X`	composition or dataset of compositions
`d`	vector containing the indices xor names of the columns selected
`name`	The new name of the amalgamation column
`pos`	The position where the new amalgamation column should be stored. This defaults to the last column.

Details

The amalgamation column is simply computed by taking the geometric mean of the non-selected components. This is consistent with the `acomp` approach and gives clear ternary diagrams. However, this geometric mean is difficult to interpret.

Value

A closed compositions with class `"acomp"` containing the variables given by `d` and the the amalgamation column.

References

Vera Pawlowsky-Glahn (2003) personal communication. Seminaris d'Estadística, Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona.

See Also

`rcompmargin`, `acomp`

Examples

```
data(SimulatedAmounts)
plot.acomp(sa.lognormals5,margin="acomp")
plot.acomp(acompmargin(sa.lognormals5,c("Pb","Zn")))
plot.acomp(acompmargin(sa.lognormals5,c(1,2)))
```

`rnorm` *Normal distributions on special spaces*

Description

`rnorm.`X generates multivariate normal random variates in the space X.

Usage

```
rnorm.acomp(n,mean,var)
rnorm.rcomp(n,mean,var)
rnorm.aplus(n,mean,var)
rnorm.rplus(n,mean,var)
rnorm.rmult(n,mean,var)
dnorm.acomp(x,mean,var)
dnorm.aplus(x,mean,var)
dnorm.rmult(x,mean,var)
```

Arguments

`n`	number of datasets to be simulated
`mean`	The mean of the dataset to be simulated
`var`	The variance covariance matrix
`x`	vectors in the sampling space

Details

The normal distributions in the variouse spaces dramatically differ. The normal distribution in the `rmult` space is the commonly known multivariate joint normal distribution. For `rplus` this distribution has to be somehow truncated at 0. This is here done by setting negative values to 0.
The normal distribution of `rcomp` is seen as a normal distribution within the simplex as a geometrical portion of the real vector space. The variance is thus forced to be singular and restricted to the affine subspace generated by the simplex. The necessary truncation of negative values is currently done by setting them explicitly to zero and reclosing afterwards.
The `"acomp"` and `"aplus"` are itself metric vector spaces and thus a normal distribution is defined in them just as in the real space. The resulting distribution corresponds to a multivariate lognormal in the case of `"aplus"` and in Aitchisons normal distribution in the simplex in the case of `"acomp"` (TO DO: Is that right??).
For the vector spaces `rmult`, `aplus`, `acomp` it is further possible to provide densities wiht repect to their Lebesgue measure. In the other cases this is not possible since the resulting distributions are not absolutly continues with respect to such a measure due to the truncation.

Value

a random dataset of the given class generated by a normal distribution with the given mean and variance in the given space.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

See Also

`runif.acomp`, `rlnorm.rplus`, `rDirichlet.acomp`

Examples

```
MyVar <- matrix(c(
0.2,0.1,0.0,
0.1,0.2,0.0,
0.0,0.0,0.2),byrow=TRUE,nrow=3)
MyMean <- c(1,1,2)

plot(rnorm.acomp(100,MyMean,MyVar))
plot(rnorm.rcomp(100,MyMean,MyVar))
plot(rnorm.aplus(100,MyMean,MyVar))
plot(rnorm.rplus(100,MyMean,MyVar))
plot(rnorm.rmult(100,MyMean,MyVar))
x <- rnorm.aplus(5,MyMean,MyVar)
dnorm.acomp(x,MyMean,MyVar)
dnorm.aplus(x,MyMean,MyVar)
dnorm.rmult(x,MyMean,MyVar)
```

`normalize` *Normalize vectors to norm 1*

Description

Normalize vectors to norm 1.

Usage

```
normalize(x,...)
## Default S3 method:
normalize(x,...)
```

Arguments

`x`	a dataset or a single vector of some type
`...`	currently not used, intended to select a different norm

Value

The vectors given, but normalized to norm 1.

See Also

`norm`

Examples

```
data(SimulatedAmounts)
normalize(c(1,2,3))
normalize(acomp(c(1,2,3)))
norm(normalize(acomp(sa.groups)))
```

`is.acomp` *Check for compositional data type*

Description

`is.`*XXX* returns `TRUE` if and only if its argument is of type *XXX*

Usage

```
is.acomp(x)
is.rcomp(x)
is.aplus(x)
is.rplus(x)
is.rmult(x)
```

Arguments

`x`	any object to be checked

Details

These functions only check for the class of the object.

Value

TRUE or FALSE

See Also

`acomp`, `rcomp` `aplus`, `rplus`

Examples

```
is.acomp(1:3)
is.acomp(acomp(1:3))
is.rcomp(acomp(1:3))
is.acomp(acomp(1:3)+acomp(1:3))
```

`runif` *The uniform distribution on the simplex*

Description

Generates random compositions with a uniform distribution on the (rcomp) simplex.

Usage

```
runif.acomp(n,D)
runif.rcomp(n,D)
```

Arguments

`n`	number of datasets to be simulated
`D`	number of parts

Value

a generated random dataset of class `"acomp"` or `"rcomp"` with drawn from a uniform distribution on the simplex of D parts.

Note

The only difference between both routines is the type of the returned dataset.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

`rDirichlet.acomp`

Examples

```
plot(runif.acomp(10,3))
plot(runif.rcomp(10,3))
```

`rmultmatmult`	*inner product for datasets with vector scale*

Description

An rmult object is considered as a sequence of vectors. The `%*%` is considered as the inner multiplication. An inner multiplication with another vector is the scalar product. an inner multiplication with a matrix is a matrix multiplication, where the rmult-vectors are either considered as row or as column vector.

Arguments

```
 x %*% y
x %*% v
v %*% x
x %*% A
A %*% x
w %*% A
A %*% w
w %*% x
x %*% w
```

`x`	an rmult vector or dataset of vectors
`y`	an rmult vector or dataset of vectors
`v`	a numeric vector of length `gsi.getD(x)`
`w`	a numeric vector of length `gsi.getD(x)`
`A`	a matrix

Details

The operators try to minic the behavior of `%*%` on `c()`-vectors as inner product applied in parallel to all vectors of the dataset. Thus the product of a vector with another `rmult` object or unclassed vector *v* results in the scalar product. For the multiplication with a matrix each vector is considered as a row or column, whatever is more appropriate.

Value

an object of class `"rmult"` or a numeric vector containing the result of the corresponding inner products.

Note

The product `x %*% A %*% y` is associative.

See Also

`rmult`, `%*%.rmult`

Examples

```
x <- rmult(matrix( sqrt(1:12), ncol= 3 ))
x%*%x
A <- matrix( 1:9,nrow=3)
x %*% A %*% x
x %*% A
A %*% x
x %*% 1:3
x %*% 1:3
1:3 %*% x
```

---

gsipairs *Internal functions of the compositions package*

Description

Creates a paired plot like pairs but allows to add additional panels afterwards

Usage

```
gsi.pairs(x, labels, panel = points, ..., main = NULL, oma = NULL,
    font.main = par("font.main"), cex.main = par("cex.main"),
    lower.panel = panel, upper.panel = panel, diag.panel = NULL,
    text.panel = textPanel, label.pos = 0.5 + has.diag/3, cex.labels = NULL,
    font.labels = 1, row1attop = TRUE, gap = 1,add=list(),xlim=apply(x,2,range),ylim=apply(x,2,
gsi.add2pairs(x,panel,...,noplot=FALSE)
gsi.plots
```

Arguments

x	a multivariate dataset
labels	The names of the variables
panel	The function to performe the actual pairwise plots.
...	see pairs
main	see pairs
oma	see pairs
font.main	see pairs
cex.main	see pairs
lower.panel	see pairs
upper.panel	see pairs
diag.panel	see pairs
text.panel	see pairs
label.pos	see pairs

`cex.labels`	see `pairs`
`font.labels`	see `pairs`
`row1attop`	see `pairs`
`gap`	see `pairs`
`add`	additional parameter containing a list of additional panels
`xlim`	additional 2x`ncol(x)`-matrix parameter giving in `xlim[,j]` the xlims of the j-th column
`ylim`	additional 2x`ncol(x)`-matrix parameter giving in `ylim[,i]` the ylims of the j-th column
`log`	additional parameter with possible values like in `plot` allowing to log some plots, without a warning
`noplot`	Logical indicating wether the plotting should be suppressed. This is usefull for plotting single page postscripts.

Details

`gsi.pairs` essentially copies the functionality of `pairs`. However it additionally stores its own parameters in the dev.cur() position of gsi.plots and allows to modify the parameters and redo a modified plot afterwards. This is mainly done by `gsi.add2pairs` by modifying the additional add parameter, that specifies more panels. This meachnism showed not be used directly since it is planed to replace the whole meachnism by a more rigor solution soon.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`,

Examples

`perturbe` *Perturbation of compositions*

Description

The perturbation is the addition operation in the Aitchison geometry of the simplex.

Usage

```
perturbe(x,y)
## Methods for class "acomp"
## x + y
## x - y
##   - x
```

Arguments

x	compositions of class acomp
y	compositions of class acomp

Details

The perturbation is the basic addition operation of the Aitichson simplex as a vector space. It is defined by:

$$(x + y)_i = clo((x_i y_i)_i)_i$$

`permute` and `+` compute this operation. The only difference is that `+` checks the class of its argument, while `permute` does not check the type of the arguments and can thus directly be applied to a composition in any form (unclassed, acomp, rcomp).
The `-` operation is the inverse of the addition in the usual way and defined by:

$$(x - y)_i := clo((x_i / y_i)_i)_i$$

and as unary operation respectively as:

$$(-x)_i := clo((1/y_i)_i)_i$$

Value

An acomp vector or matrix.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

acomp, *.aplus, +.rplus

Examples

```
tmp <- -acomp(1:3)
tmp + acomp(1:3)
```

`variation` *Variation matrices of amounts and compositions*

Description

Compute the variation matrix in the various approaches of compositional and amount data analysis. Pay attention that this is not computing the variance or covariance matrix!

Usage

```
variation(x,...)
## S3 method for class 'acomp':
variation(x, ...)
## S3 method for class 'rcomp':
variation(x, ...)
## S3 method for class 'aplus':
variation(x, ...)
## S3 method for class 'rplus':
variation(x, ...)
## S3 method for class 'rmult':
variation(x, ...)
```

Arguments

`x`	a dataset, eventually of amounts or compositions
`...`	currently unused

Details

The variation matrix was defined in the acomp context for analysis of compositions as the matrix of variances of all possible log-ratios among components (Aitchison, 1986). The generalization to rcomp objects is simply to reproduce the variance of all possible differences between components. The amount and rmult objects should not be treated with variation matrices, because this implies always the existence of a closure.

Value

The variation matrix of x.

See Also

`cdt`, `clrvar2ilr`, `clo`, `mean.acomp`, `acomp`, `rcomp`, `aplus`, `rplus`

Examples

```
data(SimulatedAmounts)
mean.col(sa.lognormals)
variation(acomp(sa.lognormals))
variation(rcomp(sa.lognormals))
variation(aplus(sa.lognormals))
variation(rplus(sa.lognormals))
variation(rmult(sa.lognormals))
```

`ult` *Uncentered log transform*

Description

Compute the uncentered log ratio transform of a (dataset of) composition(s) and its inverse.

Usage

```
ult( x )
ult.inv( z )
Kappa( x )
```

Arguments

`x`	a composition or a data matrix of compositions, not necessarily closed
`z`	the ult-transform of a composition or a data matrix of clr-transforms of compositions, not necessarily centered

Details

The ult-transform is simply the elementwise log of the closed composition. The ult has some important properties in the scope of Information Theory.

Value

`ult` gives the uncentered log transform,
`ult.inv` gives closed compositions with the given ult-transforms
`Kappa` gives the difference between the clr and the ult transform. It is quite linked to information measures.

See Also

`ilr`,`alr`,`apt`

Examples

```
(tmp <- ult(c(1,2,3)))
ult.inv(tmp)
ult.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(ult(cdata))
Kappa(c(1,2,3))
```

`qqnorm` *Normal quantile plots for compositions and amounts*

Description

The plots allow to check the normal distribution of multiple univaritate marginals by normal-quantile-quantile plots. For the different interpretations of amount data a different type of normality is assumed and checked. When an alpha-level is given the marginal displayed in each panel is checked for normality.

Usage

```
## S3 method for class 'acomp':
qqnorm(y,fak=NULL,...,panel=vp.qqnorm,alpha=NULL)
## S3 method for class 'rcomp':
qqnorm(y,fak=NULL,...,panel=vp.qqnorm,alpha=NULL)
## S3 method for class 'aplus':
qqnorm(y,fak=NULL,...,panel=vp.qqnorm,alpha=NULL)
## S3 method for class 'rplus':
qqnorm(y,fak=NULL,...,panel=vp.qqnorm,alpha=NULL)
vp.qqnorm(x,y,...,alpha=NULL)
```

Arguments

`y`	a dataset
`fak`	a factor to split the dataset, not yet implemented in aplus and rplus
`panel`	the panel function to be used or a list of multiple panel functions
`alpha`	The alpha level of a test for normality to be performed for each of the displayed marginals. The levels are adjusted for multiple testing with a Bonferroni-correction (i.e. dividing each of the the alpha-level by the number of test performed)
`...`	further graphical parameters
`x`	used by pairs only

Details

`qqnorm.rplus` and `qqnorm.rcomp` displays qqnorm-plots of individual amounts (on the diagonal), of pairwise differences of amounts (above the diagonal) and of pairwise sums of amounts (below the diagonal).
`qqnorm.aplus` displays qqnorm-plots of individual log-amounts (on the diagonal), of pairwise log-rations of amounts (above the diagonal) and of pairwise sums of log amount (below the diagonal).
`qqnorm.aplus` displays qqnorm-plots of pairwise log-ratios of amounts in all of diagonal panels. Nothing is displayed on the diagonal.
In all cases a joint normality of the original data in the selected framework would imply normality in all displayed marginal distributions.
The marginal normality can be checked in each of the plots using a `shapiro.test`, by specifying an alpha level. The alpha level are corrected for multiple testing. Plots displaying a marginal distribution significantly deviating from a normal distribution are marked by a red exclamation mark.
`vp.qqnorm` is used as a panel function to make high dimensional plots.

See Also

`plot.acomp`, `boxplot.acomp`, `rnorm.acomp`, `rnorm.rcomp`, `rnorm.aplus`, `rnorm.rplus`

Examples

```
data(SimulatedAmounts)
qqnorm(acomp(sa.lognormals),alpha=0.05)
qqnorm(rcomp(sa.lognormals),alpha=0.05)
qqnorm(aplus(sa.lognormals),alpha=0.05)
qqnorm(rplus(sa.lognormals),alpha=0.05)
```

`princomp.aplus` *Principal component analysis for amounts in log geometry*

Description

A principal component analysis is done in the Aitchison geometry (i.e. ilt-transform). Some gimics simplify the interpretation of the computed components as amount perturbations.

Usage

```
## S3 method for class 'aplus':
princomp(x,...,scores=TRUE)
## S3 method for class 'princomp.aplus':
print(x,...)
## S3 method for class 'princomp.aplus':
plot(x,y=NULL,...,
npcs=min(10,length(x$sdev)),
type=c("screeplot","variance","biplot","loadings","relative"),
main=NULL,
scale.sdev=1)
## S3 method for class 'princomp.aplus':
predict(object,newdata,...)
```

Arguments

`x`	an aplus dataset (for princomp) or a result from princomp.aplus
`y`	not used
`scores`	a logical indicating whether scores should be computed or not
`npcs`	the number of components to be drawn in the scree plot
`type`	type of the plot: `"screeplot"` is a lined screeplot, `"variance"` is a boxplot like screeplot, `"biplot"` is a biplot, `"loadings"` displayes the Loadings as a `barplot.acomp`
`scale.sdev`	the multiple of sigma to use plotting the loadings
`main`	headline of the plot
`object`	a fitted princomp.aplus object
`newdata`	another amount dataset of class aplus
`...`	further arguments to pass to internally-called functions

Details

As a metric euclidean space, the positive real space described in `aplus` has its own principal component analysis, that should be performed in terms of the covariance matrix and not in terms of the meaningless correlation matrix.
To aid the interpretation we added some extra functionality to a normal `princomp(ilt(x))`. First of all the result contains as additional information the amount representation of returned vectors in the space of the data: the center as amount `Center`, and the loadings in terms of amounts to perturbe with positively (`Loadings`) or negativly (`DownLoadings`). The Up- and DownLoadings are normalized to the number of parts and not to one to simplify the interpretation. A value of about one means no change in the specific component.
The plot routine provides screeplots (`type = "s"`,`type= "v"`), biplots (`type = "b"`), plots of the effect of loadings (`type = "b"`) in `scale.sdev*sdev`-spread, and loadings of pairwise (log-)ratios (`type = "r"`).
The interpretation of a screeplot does not differ from ordinary screeplots. It shows the eigenvalues of the covariance matrix, which represent the portions of variance explained by the principal components.
The interpretation of the the biplot uses additionally to the classical one a compositional concept: The differences between two arrowheads can be interpreted as log-ratios between the two components represented by the arrows.
The amount loading plot is introduced with this package. The loadings of all component can be seen as an orthogonal basis in the space of `ilt`-transformed data. These vectors are displayed by a barplot with their corresponding amounts. A portion of one means no change of this part. This is equivalent to a zero loading in a real principal component analysis.
The loadings plot can work in two different modes: If `scale.sdev` is set to `NA` it displays the amount vector being represented by the unit vector of loadings in the ilt-transformed

space. If `scale.sdev` is numeric we use the this amount vector scaled by the standard deviation of the respective component.
The relative plot displays the `relativeLoadings` as a barplot. The deviation from a unit bar shows the effect of each principal component on the respective ratio. The interpretation of the ratios plot may only be done in an Aitchison-compositional framework.

Value

`princomp` gives an object of type `c("princomp.acomp","princomp")` with the following content:

`sdev`	the standard deviation of the principal components
`loadings`	the matrix of variable loadings (i.e., a matrix which columns contain the eigenvectors). This is of class `"loadings"`.
`center`	the ilt-transformed vector of means used to center the dataset
`Center`	the `aplus` vector of means used to center the dataset
`scale`	the scaling applied to each variable
`n.obs`	number of observations
`scores`	if `scores = TRUE`, the scores of the supplied data on the principal components and the information available. Scores are coordinates in a basis given by the principal components and thus not compositions
`call`	the matched call
`na.action`	not clearly understood
`Loadings`	vectors of amounts that represent a perturbation with the vectors represented by the loadings of each of the factors
`DownLoadings`	vectors of amounts that represent a perturbation with the inverses of the vectors represented by the loadings of each of the factors

`predict` returns a matrix of scores of the observations in the `newdata` dataset
. The other routines are mainly called for their side effect of plotting or printing and return the object `x`.

See Also

`ilt`,`aplus`, `relativeLoadings` `princomp.acomp`, `princomp.rplus`, `barplot.aplus`, `mean.aplus`,

Examples

```
data(SimulatedAmounts)
pc <- princomp(aplus(sa.lognormals5))
pc
summary(pc)
plot(pc)           #plot(pc,type="screeplot")
plot(pc,type="v")
plot(pc,type="biplot")
plot(pc,choice=c(1,3),type="biplot")
plot(pc,type="loadings")
plot(pc,type="loadings",scale.sdev=-1) # Downward
plot(pc,type="relative",scale.sdev=NA) # The directions
plot(pc,type="relative",scale.sdev=1) # one sigma Upward
plot(pc,type="relative",scale.sdev=-1) # one sigma Downward
biplot(pc)
screeplot(pc)
loadings(pc)
relativeLoadings(pc,mult=FALSE)
relativeLoadings(pc)
relativeLoadings(pc,scale.sdev=1)
relativeLoadings(pc,scale.sdev=2)

pc$Loadings
pc$DownLoadings
barplot(pc$Loadings)
pc$sdev^2
cov(predict(pc,sa.lognormals5))
```

`scalar` *Parallel scalar products*

Description

Usage

```
scalar(x,y)
## Default S3 method:
scalar(x,y)
```

Arguments

x	a vector or a matrix with rows considered as vectors
y	a vector or a matrix with rows considered as vectors

Details

The scalar product of two vectors is defined as:

$$scalar(x,y) := \sum(x_i y_i)$$

Value

a numerical vector containing the scalar products of the vectors given by x and y. If both `x` and `y` contain more than one vector the function uses parallel operation like it would happen with an ordinary product of vectors.

Note

The computation of the scalar product implicitly applies the cdt transform, which implies that the scalar products corresponding to the given geometries are returned for acomp, rcomp, aplus, rplus-objects. Even a useful scalar product for factors is defined.

Examples

```
scalar
```

---

oneOrDataset *Treating single compositions as one-row datasets*

Description

A dataset is converted to a data matrix. A single data item (i.e. a simple vector) is converted to a one-row data matrix.

Usage

```
oneOrDataset(W,B=NULL)
```

Arguments

W	a vector, matrix or dataframe
B	an optional second vector, matrix or data frame having the intended number of rows.

Value

A data matrix containing the same data as W. If W is a vector it is interpreded as a single row. If `B` is given and `length(dim(B))!= 2` and `W` is a vector, then `W` is repeated `nrow(B)` times.

See Also

Examples

```
oneOrDataset(c(1,2,3))
oneOrDataset(c(1,2,3),matrix(1:12,nrow=4))
oneOrDataset(data.frame(matrix(1:12,nrow=4)))
```

`rDirichlet` *Dirichlet distribution*

Description

The Dirichlet distribution on the simplex.

Usage

```
rDirichlet.acomp(n,alpha)
rDirichlet.rcomp(n,alpha)
```

Arguments

`n`	number of datasets to be simulated
`alpha`	parameters of the Dirichlet distribution

Details

TO DO!!!

Value

a generated random dataset of class `"acomp"` or `"rcomp"` with drawn from a Dirichlet distribution with the given parameter `alpha`. The names of `alpha` are used to name the parts.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

`rnorm.acomp`

Examples

```
tmp <- rDirichlet.acomp(10,alpha=c(A=2,B=0.2,C=0.2))
plot(tmp)
```

`aplus` *Amounts analysed in log-scale*

Description

A class to analyse positive amounts in a logistic framework.

Usage

```
aplus(X,parts=1:NCOL(oneOrDataset(X)),total=NA)
```

Arguments

`X`	vector or dataset of positive numbers
`parts`	vector containing the indices xor names of the columns to be used
`total`	a numeric vectors giving the total amounts of each dataset.

Details

Many multivariate datasets essentially describe amounts of D different parts in a whole. When the whole is large in relation to the considered parts, such that they do not exclude each other, or when the total amount of each componenten is indeed determined by the phenomenon under investigation and not by sampling artifacts (such as dilution or sample preparation), then the parts can be treated as amounts rather than as a composition (cf. `acomp`, `rcomp`).

Like compositions, amounts have some important properties. Amounts are always positive. An amount of exactly zero essentially means that we have a substance of an other quality. Different amounts - spanning different orders of magnitude - are often given in different units (ppm, ppb, %) and conversion factors need not to be fixed (e.g. for ppm, g/l, vol.%, mass %, molar fraction). Often, these amounts are also taken as indicators of other non-measured components (e.g. K as indicator for potassium feldspar), which might be proportional to the measured amount. However, in contrast to compositions, amounts themselves do matter. Amounts are typically heavily skewed and in many practical cases a log-transform makes their distribution roughly symmetric, even normal.

In full analogy to Aitchison's compositions, we introduce vector space operations for amounts: the perturbation `perturbe.aplus` as a vector space addition (corresponding to change of units), the power transformation `power.aplus` as scalar multiplication describing the law of mass action, and a distance `dist` which is independent of the chosen units. The induced vector space is mapped isometrically to a classical R^D by a simple log-transformation called `ilt`, resembling classical log transform approaches.

The general approach in analysing aplus objects is thus to performe classical multivariate analysis on ilt-transformed coordinates and to backtransform or display the results in such a way that they can be interpreted in terms of the original amounts.

The class aplus is complemented by the `rplus`, allowing to analyse amounts directly as real numbers, and by the classes `acomp` and `rcomp` to analyse the same data as compositions disregarding the total amounts, focusing on relative amounts only.

The classes rcomp, acomp, aplus, and rplus are designed as similar as possible in order to allow direct comparison between results achieved by the different approaches. Especially the acomp simplex transforms `clr`, `alr`, `ilr` are mirrored in the aplus class by the single bijective isometric transform `ilt`

Value

a vector of class `"aplus"` representing a vector of amounts or a matrix of class `"aplus"` representing multiple vectors of amounts, each vector in one row.

References

See Also

`ilt`,`acomp`, `rplus`, `princomp.aplus`, `plot.aplus`, `boxplot.aplus`, `barplot.aplus`, `mean.aplus`, `var.aplus`, `variation.aplus`, `cov.aplus`, `msd`

Examples

```
data(SimulatedAmounts)
plot(aplus(sa.lognormals))
```

`gsiinternal` *Internal functions of the compositions package*

Description

Internal function to compute 2D marginal compositions for plots

Usage

```
gsi.plotmargin(X,d,margin)
```

Arguments

`X`	A multivariate compositional dataset
`d`	a numeric or character vector of two elements specifying the margin
`margin`	a character specifying the type of margin to be choosen. Possible values are ”acomp”, ”rcomp” or a column name from the dataset.

Details

Value

a composition of three elements.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`

Examples

`alr` *Additive log ratio transform*

Description

Compute the additive log ratio transform of a (dataset of) composition(s), and its inverse.

Usage

```
alr( x  )
alr.inv( z )
```

Arguments

`x`	a composition, not necessarily closed
`z`	the alr-transform of a composition, thus a (D-1)-dimensional real vector

Details

The alr-transform maps a composition in the D-part Aitchison-simplex non-isometrically to a D-1 dimensonal euclidian vector, treating the last part as common denominator of the others. The data can then be analysed in this transformation by all classical multivariate analysis tools not relying on a distance. The interpretation of the results is relatively simple, since the relation to the original D-1 first parts is preserved. However distance is an extremely relevant concept in most types of analysis, where a `clr` or `ilr` transformation should be preferred.

The additive logratio transform is given by

$$alr(x)_i := \ln \frac{x_i}{x_D}$$

.

Value

`alr` gives the additive log ratio transform; accepts a compositional dataset `alr.inv` gives a closed composition with the given alr-transform; accepts a dataset

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

clr,alr,apt, http://ima.udg.es/Activitats/CoDaWork03

Examples

```
(tmp <- alr(c(1,2,3)))
alr.inv(tmp)
unclass(alr.inv(tmp)) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(alr(cdata))
```

`princomp.rmult` *Principle component analysis for Real data*

Description

Performes a principle component analysis for datasets of type rmult.

Usage

```
## S3 method for class 'rmult':
princomp(x,...)
```

Arguments

`x`	a rmult-dataset
`...`	Further arguments to call of princomp.default

Details

The function just does `princomp(unclass(x),...,scale=scale)` and is only here for convenience.

Value

An object of type `princomp` with the following fields

`sdev`	the standard deviation of the principle components.
`loadings`	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class `"loadings"`. The last eigenspace is removed since it should contain the irrelevant scaling.
`center`	the clr of the means that was substracted

`scale`	the scaling applied to each variable
`n.obs`	number of observations
`scores`	if `scores = TRUE`, the scores of the supplied data on the principle components and the information was available. Scores are coordinates in a basis given by the principle components and thus not compositions.
`call`	the matched call
`na.action`	Not clearly understood

See Also

`princomp.rplus`

Examples

```
data(SimulatedAmounts)
pc <- princomp(rmult(sa.lognormals5))
pc
summary(pc)
plot(pc)
screeplot(pc)
screeplot(pc,type="l")
biplot(pc)
biplot(pc,choice=c(1,3))
loadings(pc)
plot(loadings(pc))
pc$sdev^2
cov(predict(pc,sa.lognormals5))
```

`totals` *Total sum of amounts*

Description

Calculates the total amount by summing the individual parts.

Usage

```
totals(x,...)
## S3 method for class 'acomp':
totals(x,...)
## S3 method for class 'rcomp':
totals(x,...)
## S3 method for class 'aplus':
totals(x,...)
## S3 method for class 'rplus':
totals(x,...)
```

Arguments

`x`	an amount/amount dataset
`...`	not used, only here for generics

Value

a numeric vector containing the total amounts

See Also

aplus

Examples

```
data(SimulatedAmounts)
totals(acomp(sa.lognormals))
totals(rcomp(sa.lognormals,total=100))
totals(aplus(sa.lognormals))
totals(rplus(sa.lognormals))
aplus(acomp(sa.lognormals),total=totals(aplus(sa.lognormals)))
```

`lines` *Draws connected lines from point to point.*

Description

Functions taking coordinates given in various ways and joining the corresponding points with line segments.

Usage

```
## S3 method for class 'acomp':
lines(x,...,steps=30)
## S3 method for class 'rcomp':
lines(x,...,steps=30)
## S3 method for class 'aplus':
lines(x,...,steps=30)
## S3 method for class 'rplus':
lines(x,...,steps=30)
## S3 method for class 'rmult':
lines(x,...,steps=30)
```

Arguments

`x`	a dataset of the given type
`...`	further graphical parameters
`steps`	the numer of discretisation points to draw the segments not straight on the monitor.

Details

The functions add lines to the graphics generated with the corresponding plot functions.

Adding to multipaneled plots, redraws the plot completely and is only possible, when the plot has been created with the plotting routines from this library.

See Also

`plot.acomp`, `straight`

Examples

```
data(SimulatedAmounts)

plot(acomp(sa.lognormals))
lines(acomp(sa.lognormals),col="red")
lines(rcomp(sa.lognormals),col="blue")

plot(aplus(sa.lognormals[,1:2]))
lines(aplus(sa.lognormals[,1:2]),col="red")
lines(rplus(sa.lognormals)[,1:2],col="blue")

plot(rplus(sa.lognormals[,1:2]))
tt<-aplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="red")
tt<-rplus(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="blue")
tt<-rmult(sa.lognormals[,1:2]); ellipses(mean(tt),var(tt),r=2,col="green")
```

`summary.acomp` *Summarizing a compositional dataset in terms of ratios*

Description

Summaries in terms of compositions are quite different from classical ones. Instead of analysing each variable individually, we must analyse each pairwise ratio in a log geometry.

Usage

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

Arguments

`object`	a data.matrix of compositions, not necessarily closed
`...`	not used, only here for generics

Details

It is quite difficult to summarize a composition in a consistent and interpretable way. We tried to provide such a summary here.

Value

The result is an object of type `"summary.acomp"`

`mean`	The `mean.acomp` composition
`mean.ratio`	A matrix containing the geometric mean of the pairwise ratios
`variation`	The variation matrix of the dataset ({`variation.acomp`})
`expsd`	A matrix containing the one-sigma factor for each ratio, computed as `exp(sqrt(variation.acomp(W)))`. To obtain two-sigma-factor it needs to be squared. To obtain the reverse bound we compute 1/expsd
`min`	A matrix containing the minimum of each of the pairwise ratios
`q1`	A matrix containing the 1-Quartile of each of the pairwise ratios
`median`	A matrix containing the median of each of the pairwise ratios
`q1`	A matrix containing the 3-Quartile of each of the pairwise ratios
`max`	A matrix containing the maximum of each of the pairwise ratios

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

`acomp`

Examples

```
data(SimulatedAmounts)
summary(acomp(sa.lognormals))
```

`rmult` *Simple treatment of real vectors*

Description

A class to analyse real multivariate vectors.

Usage

```
            rmult(X,parts=1:NCOL(oneOrDataset(X)))
```

Arguments

`X`	vector or dataset of numbers considered as elements of a R-vector
`parts`	vector containing the indices xor names of the columns to be used

Details

The `rmult` class is a simple convenience class to treat data in the scale of real vectors just like data in the scale of real numbers. A major aspect to take into account is that the internal arithmetic of R is different for these vectors.

Value

a vector of class `"rmult"` representing one vector or a matrix of class `"rmult"`, representing multiple vectors by rows.

See Also

`+.rmult`, `scalar`, `norm`, `%*%.rmult`, `rplus`, `acomp`,

Examples

```
plot(rnorm.rmult(30,mean=0:4,var=diag(1:5)+10))
```

`gsi.eps` *Internal variable: Negligible differences*

Description

fixes or yields the internal value for differences to be considered as zero.

Usage

```
gsi.eps
```

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`

Examples

apt *Additive planar transform*

Description

Compute the additive planar transform of a (dataset of) compositions and its inverse.

Usage

```
apt( x )
apt.inv( z )
```

Arguments

x	a composition or a matrix of compositions, not necessarily closed
z	the apt-transform of a composition or a matrix of alr-transforms of compositions

Details

The apt-transform maps a composition in the D-part real-simplex linearly to a D-1 dimensional euclidian vector. Although the transformation does not reach the whole R^{D-1}, resulting covariance matrices are typically of full rank.

The data can then be analysed in this transformation by all classical multivariate analysis tools not relying on distances. See `cpt` and `ipt` for alternatives. The interpretation of the results is easy since the relation to the first D-1 original variables is preserved.

The additive planar transform is given by

$$apt(x)_i := clo(x)_i, i = 1, \ldots, D-1$$

Value

`apt` gives the centered planar transform, `apt.inv` gives closed compositions with the given apt-transforms

References

See Also

`alr`,`cpt`,`ipt`

Examples

```
(tmp <- apt(c(1,2,3)))
apt.inv(tmp)
apt.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(apt(cdata))
```

`idt` *Isometric default transform*

Description

Compute the isometric default transform of a vector (or dataset) of compositions or amounts in the selected class.

Usage

```
idt(x)
## Default S3 method:
idt( x )
## S3 method for class 'acomp':
idt( x )
## S3 method for class 'rcomp':
idt( x )
## S3 method for class 'aplus':
idt( x )
## S3 method for class 'rplus':
idt( x )
## S3 method for class 'rmult':
idt( x )
## S3 method for class 'factor':
idt( x )
```

Arguments

`x`	a classed amount or composition, to be transformed with its isometric default transform

Details

The general idea of this package is to analyse the same data with different geometric concepts, in a fashion as similar as possible. For each of the four concepts there exists an isometric transform expressing the geometry in a full-rank euclidean vector space. Such a transformation is computed by `idt`. For `acomp` the transform is `ilr`, for `rcomp` it is `ipt`, for `aplus` it is `ilt`, and for `rplus` it is `iit`. Keep in mind that the transform does not keep the variable names, since there is no guaranteed one-to-one relation between the original parts and each transformed variable.

Value

A corresponding matrix of row-vectors containing the transforms.

References

See Also

`cdt`, `ilr`, `ipt`, `ilt`, `iit`

Examples

```
## Not run:
# the idt is defined by
idt         <- function(x) UseMethod("idt",x)
idt.default <- function(x) x
idt.acomp   <- function(x) ilr(x)
idt.rcomp   <- function(x) ipt(x)
idt.aplus   <- ilt
idt.rplus   <- iit
## End(Not run)
idt(acomp(1:5))
idt(rcomp(1:5))
```

`endpointCoordinates` *Amounts in barytic-coordinates*

Description

Computes the convex combination of amounts given by `endpoints` to explain `X` as good as possible.

Usage

```
endpointCoordinates(X,...)
endpointCoordinatesInv(K,endpoints,...)
## Default S3 method:
endpointCoordinates(X,endpoints=diag(gsi.getD(X)), ...)
## S3 method for class 'acomp':
endpointCoordinates(X,endpoints=clr.inv(diag(gsi.getD(X))),...)
## S3 method for class 'aplus':
endpointCoordinates(X,endpoints,...)
## S3 method for class 'rplus':
endpointCoordinates(X,endpoints,...)
## S3 method for class 'rmult':
```

```
endpointCoordinatesInv(K,endpoints,...)
## S3 method for class 'acomp':
endpointCoordinatesInv(K,endpoints,...)
## S3 method for class 'rcomp':
endpointCoordinatesInv(K,endpoints,...)
## S3 method for class 'aplus':
endpointCoordinatesInv(K,endpoints,...)
## S3 method for class 'rplus':
endpointCoordinatesInv(K,endpoints,...)
```

Arguments

`X`	a dataset of amounts or compositions, to be represented in as convex combination of the endpoints in the given geometry
`K`	Konvex combination weights to the endpoints
`endpoints`	a dataset of extremal compositions from the same space as X. The number of endpoints given must not exceed the dimension of the space plus one.
`...`	currently unused

Details

The convex combination is performed in the respective geometry. This means that for rcomp positivity of the result is only guaranteed with extermal endmembers and that in acomp-geometry it is not possible to give extremal endmembers.
The main idea behind this functions is that the actually observed composition came from a convex combination of some extremal compositions specified by endpoints. Strictly speaking this is meaningfull in strictly this sense only in rplus-geometry and under some special circumstances in rcomp geometry. It is not meaningfull in terms of mass conservation in acomp- and aplus-geometry due to the non mass-balancing character of the geometry. In rcomp-geometry it dependent on unit of measurements and different for volume and mass % and only valid if the whole composition is observed.

Value

The `endpointCoordinates` functions give a `"rmult"`-dataset giving the convex weights, which allow to combine `X` from `endpoints` as good as possible. The result is an `"rmult"` since there is guarantee that the resulting weights are positive.
The `endpointCoordinates` functions reconstruct the convex combination from coordinates `K` and the given `endpoints`. The class of `endpoints` determines the geometry chosen and the class of the result.

References

Shurtz, Robert F., 2003. Compositional geometry and mass conservation. Mathematical Geology 35 (8), 972–937.

Examples

```
data(SimulatedAmounts)
ep <- aplus(rbind(c(2,1,2),c(2,2,1),c(1,2,2)))
dat <- endpointCoordinatesInv(acomp(sa.lognormals),acomp(ep))
plot(dat)
plot( acomp(endpointCoordinates(dat,acomp(ep))))

dat <- endpointCoordinatesInv(rcomp(sa.lognormals),rcomp(ep))
plot(dat)
plot( rcomp(endpointCoordinates(dat,rcomp(ep))))

dat <- endpointCoordinatesInv(aplus(sa.lognormals),aplus(ep))
plot(dat)
plot( endpointCoordinates(dat,aplus(ep)))

dat <- endpointCoordinatesInv(rplus(sa.lognormals),rplus(ep))
plot(dat)
plot(endpointCoordinates(rplus(dat),rplus(ep)))
```

`gsi2.invperm` *Internal function: Invert a permutation*

Description

Finds the inverse of a permutation given as a vector of indices.

Usage

```
gsi2.invperm( i,n )
```

Arguments

`i`	a sequence of different integers in `1:n` considered as a permutation given by `p=unique(c(i,1:n))`
`n`	the number of elements to be permuted

Details

The inverse permutation is defined by `p[`
`var{v}]==1:n` and `v[`
`var{p}]==1:n`.

Value

an integer vector *v* describing the inverse permutation of p.

Note

Do not use gsi.* functions directly since they are internal functions of the package

References

See Also

gsi

Examples

```
gsi2.invperm(c(2,3),10)
```

---

gsiinternal2 *Internal functions of the compositions package*

Description

Internal functions

Usage

```
gsi.pairrelativeMatrix(names)
```

Arguments

names	a character vector provinding names

Details

Value

a matrix containing pairwise contrasts

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi

Examples

```
gsi.pairrelativeMatrix(c("a","b","c"))
```

`rmultarithm` *vectorial arithmetic for datasets in a classical vector scale*

Description

vector space operations computed for multiple vectors in parallel

Usage

```
## Methods for class "rmult"
## x+y
## x-y
## -x
## x*r
## r*x
## x/r
```

Arguments

x	an rmult vector or dataset of vectors
y	an rmult vector or dataset of vectors
r	a numeric vector of size 1 or nrow(x)

Details

The operators try to mimic the parallel operation of R on vectors of real numbers on vectors of vectors represented as matrices containing the vectors as rows.

Value

an object of class `"rmult"` containing the result of the corresponding operation on the vectors.

See Also

`rmult`, `%*%.rmult`

Examples

```
x <- rmult(matrix( sqrt(1:12), ncol= 3 ))
x
x+x
x + rmult(1:3)
x * 1:4
1:4 * x
x / 1:4
x / 10
```

`iit` *Isometric identity transform*

Description

Compute the isometric identity transform of a vector (dataset) of amounts and its inverse.

Usage

```
iit( x )
iit.inv( z  )
```

Arguments

`x`	a vector or data matrix of amounts
`z`	the iit-transform of a vector or data.matrix of iit-transforms of amounts

Details

The iit-transform maps D amounts (considered in a real geometry) isometrically to a D dimensonal euclidian vector. The `iit` is part of the `rplus` framework. Despite its trivial operation, it is present to achieve maximal analogy between the `aplus` and the `rplus` framework.
The data can then be analysed in this transformated space by all classical multivariate analysis tools. The interpretation of the results is easy since the relation to the original variables is preserved. However results may be inconsistent, since the multivariate analysis tools disregard the positivity condition and the inner laws of amounts.

The isometric identity transform is a simple identity given by

$$iit(x)_i := x_i$$

Value

`ilt` gives the isometric identity transform, i.e. simply the input striped of the "rplus" class attribute, `ipt.inv` gives amounts with class "rplus" with the given iit, i.e. simply the argument checked to be a valid "rplus" object, and with this class attribute.

Note

`iit` can be used to unclass amounts.

References

See Also

ilt, ilr, rplus

Examples

```
(tmp <- iit(c(1,2,3)))
iit.inv(tmp)
iit.inv(tmp) - c(1,2,3) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(iit(cdata))
```

---

gsi.mapin01 *Internal functions: Storing integers as reals*

Description

An integer number is stored in a dataset with a given range.

Usage

```
gsi.mapin01(i,min=0,max=1)
gsi.mapfrom01(x)
gsi.mapmin(x)
gsi.mapmax(x)
```

Arguments

`i`

`x`

`max`

`min`

the minimum of the created dataset

Details

The function is used to get full control over the graphic ranges in pair plots and to pass the used column to panel functions.

Value

`gsi.mapin01` gives a vector *x* with `range(`
`var{x})==c(min,max)` and `gsi.mapfrom01(`
`var{x})`, `gsi.mapmin(`
`var{x})`, `gsi.mapmax(`
`var{x})` result in `i`, `max` and `min`.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi, plot.acomp

Examples

```
gsi.mapin01(5)
```

---

gsiinternal3 *Internal functions of the compositions package*

Description

Internal functions

Usage

```
gsi.spreadToIsoSpace(spread)
```

Arguments

`spread`	a matrix or a dataset of matrices

Details

Value

Converts a clr covariance matrix to a ilr covariance matrix

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi

Examples

`gsi.textpanel` *Internal function: A panel displaying a label only*

Description

A function useful as a text.panel in pairs.

Usage

```
gsi.textpanel(x,y,lab,...)
```

Arguments

`x`	discarded
`y`	discarded
`lab`	text to be plotted to the middle of the panel
`...`	further graphical parameters passed to `text`

Details

The function is used against log-scale problems in pairs called by function `boxplot.acomp`.

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`

Examples

```
data(iris)
pairs(iris,text.panel=gsi.textpanel)
```

`split` *Spliting datasets in groups given by factors*

Description

Each of the given scale levels has an associated norm, which is computed for each element by these functions.

Usage

```
## S3 method for class 'acomp':
split(x,f)
## S3 method for class 'rcomp':
split(x,f)
## S3 method for class 'aplus':
split(x,f)
## S3 method for class 'rplus':
split(x,f)
## S3 method for class 'rmult':
split(x,f)
```

Arguments

`x`	a dataset or a single vector of some type
`f`	a factor that defines the grouping or a list of factors

Value

a list of objects of the same type as x.

See Also

`split`

Examples

```
data(SimulatedAmounts)
split(acomp(sa.groups),sa.groups.area)
```

`gsi.plain` *Internal function: Convert to plain vector or matrix*

Description

The dataset is converted into a plain vector or matrix: data.frames are converted to data matrices and class attributes are removed.

Usage

```
gsi.plain( x )
```

Arguments

`x`	The dataset to be converted

Details

Value

unclassed object, typically a vector or matrix.

Note

Do not use gsi.* functions directly since they are internal functions of the package

References

See Also

Examples

```
gsi.plain(acomp(c(12,3,4)))
```

`gsiinternal` *Internal functions of the compositions package*

Description

Internal functions

Usage

```
gsi()
```

Arguments

None

Details

Value

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

gsi

Examples

`ilr` *Isometric log ratio transform*

Description

Compute the isometric log ratio transform of a (dataset of) composition(s) and its inverse.

Usage

```
ilr( x , V = ilrBase(x) )
ilr.inv( z , V = ilrBase(z=z))
```

Arguments

`x`	a composition, not necessarily closed
`z`	the ilr-transform of a composition
`V`	a matrix, with columns giving the chosen basis of the clr-plane

Details

The ilr-transform maps a composition in the D-part Aitchison-simplex isometrically to a D-1 dimensonal euclidian vector. The data can then be analysed in this transformation by all classical multivariate analysis tools. However the interpretation of the results may be difficult, since there is no one-to-one relation between the original parts and the transformed variables.

The isometric logratio transform is given by

$$ilr(x) := V^t clr(x)$$

with clr(x) the centred log ratio transform and $V \in R^{d\times(d-1)}$ a matrix which columns form an orthonormal basis of the clr-plane. A default matrix V is given by `ilrBase( var{D})`.

Value

`ilr` gives the isometric log ratio transform, `ilr.inv` gives closed compositions with the given ilr-transforms

References

Egozcue J.J., V. Pawlowsky-Glahn, G. Mateu-Figueras and C. Barcel'o-Vidal (2003) Isometric logratio transformations for compositional data analysis. *Mathematical Geology*, **35**(3) 279-300
Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

See Also

`clr`,`alr`,`apt`, `ilrBase`, `http://ima.udg.es/Activitats/CoDaWork03`

Examples

```
(tmp <- ilr(c(1,2,3)))
ilr.inv(tmp)
ilr.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(ilr(cdata))
ilrBase(D=3)
```

---

ilt *Isometric log transform*

Description

Compute the isometric log transform of a vector (dataset) of amounts and its inverse.

Usage

```
ilt( x )
ilt.inv( z  )
```

Arguments

`x`	a vector or data matrix of amounts
`z`	the ilt-transform of a vector or data matrix of ilt-transforms of amounts

Details

The ilt-transform maps D amounts (considered in log geometry) isometrically to a D dimensional euclidean vector. The `ilt` is part of the `aplus` framework.
The data can then be analysed in this transformation by all classical multivariate analysis tools. The interpretation of the results is easy since the relation to the original variables is preserved.

The isometric log transform is given by

$$ilt(x)_i := \ln x_i$$

Value

`ilt` gives the isometric log transform, i.e. simply the log of the argument, `ilt.inv` gives amounts with the given ilt, i.e. simple the exp of the argument

References

See Also

`ilr`, `iit`, `aplus`

Examples

```
(tmp <- ilt(c(1,2,3)))
ilt.inv(tmp)
ilt.inv(tmp) - c(1,2,3) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(ilt(cdata))
```

`princomp.rcomp` *Principal component analysis for real compositions*

Description

A principal component analysis is done in real geometry (i.e. cpt-transform) of the simplex. Some gimics simplify the interpretation of the obtained components.

Usage

```
## S3 method for class 'rcomp':
princomp(x,...,scores=TRUE)
## S3 method for class 'princomp.rcomp':
print(x,...)
## S3 method for class 'princomp.rcomp':
plot(x,y=NULL,...,
npcs=min(10,length(x$sdev)),
type=c("screeplot","variance","biplot","loadings","relative"),
main=NULL,
scale.sdev=1)
## S3 method for class 'princomp.rcomp':
predict(object,newdata,...)
```

Arguments

`x`	an rcomp dataset (for princomp) or a result from princomp.rcomp
`y`	not used
`scores`	a logical indicating whether scores should be computed or not
`npcs`	the number of components to be drawn in the scree plot
`type`	type of the plot: `"screeplot"` is a lined screeplot, `"variance"` is a boxplot like screeplot, `"biplot"` is a biplot, `"loadings"` displays the loadings as a `barplot`
`scale.sdev`	The multiple of sigma to use plotting the loadings
`main`	headline of the plot
`object`	a fitted princomp.rcomp object
`newdata`	another compositional dataset of class rcomp
`...`	further arguments to pass to internally-called functions

Details

Mainly a `princomp(cpt(x))` is performed. To avoid confusion the meaningless last principal component is removed.
The plot routine provides screeplots (`type = "s"`,`type= "v"`), biplots (`type = "b"`), plots of the effect of loadings (`type = "b"`) in `scale.sdev*sdev`-spread, and loadings of pairwise differences (`type = "r"`).
The interpretation of a screeplot does not differ from ordinary screeplots. It shows the eigenvalues of the covariance matrix, which represent the portions of variance explained by the principal components.
The interpretation of the the biplot differs from classical one. The relevant variables are not the drawn arrows of the components, but rather the differences between two arrowheads, which can be interpreted as transfer of mass from one component to the other.
The compositional loading plot is more or less a standard one. The loadings are displayed by a barplot as positve and negative changes of amounts.
The loading plot can work in two different modes: If `scale.sdev` is set to `NA` it displays the composition being represented by the unit vector of loadings in cpt-transformed space. If `scale.sdev` is numeric we use this composition scaled by the standard deviation of the respective component.
The relative plot displays the `relativeLoadings` as a barplot. The deviation from a unit bar shows the effect of each principal component on the respective differences.

Value

`princomp` gives an object of type `c("princomp.rcomp","princomp")` with the following content:

`sdev`	the standard deviation of the principal components.
`loadings`	the matrix of variable loadings (i.e., a matrix which columns contain the eigenvectors). This is of class `"loadings"`. The last eigenvalue is removed since it should contain the irrelevant scaling.
`Loadings`	The loadings as an rmult-object

`center`	the cpt-transformed vector of means used to center the dataset
`Center`	the `rcomp` vector of means used to center the dataset
`scale`	the scaling applied to each variable
`n.obs`	number of observations
`scores`	if `scores = TRUE`, the scores of the supplied data on the principal components and the information available. Scores are coordinates in a basis given by the principal components and thus not compositions
`call`	the matched call
`na.action`	not clearly understood

`predict` returns a matrix of scores of the observations in the `newdata` dataset
. The other routines are mainly called for their side effect of plotting or printing and return the object `x`.

See Also

`cpt`,`rcomp`, `relativeLoadings` `princomp.acomp`, `princomp.rplus`,

Examples

```
data(SimulatedAmounts)
pc <- princomp(rcomp(sa.lognormals5))
pc
summary(pc)
plot(pc)           #plot(pc,type="screeplot")
plot(pc,type="v")
plot(pc,type="biplot")
plot(pc,choice=c(1,3),type="biplot")
plot(pc,type="loadings")
plot(pc,type="loadings",scale.sdev=-1) # Downward
plot(pc,type="relative",scale.sdev=NA) # The directions
plot(pc,type="relative",scale.sdev=1) # one sigma Upward
plot(pc,type="relative",scale.sdev=-1) # one sigma Downward
biplot(pc)
screeplot(pc)
loadings(pc)
relativeLoadings(pc,mult=FALSE)
relativeLoadings(pc)
relativeLoadings(pc,scale.sdev=1)
relativeLoadings(pc,scale.sdev=2)

pc$sdev^2
cov(predict(pc,sa.lognormals5))
```

`cdt` *Centered default transform*

Description

Compute the centered default transform of a (dataset of) composition or amount.

Usage

```
cdt(x)
## Default S3 method:
cdt( x )
## S3 method for class 'acomp':
cdt( x )
## S3 method for class 'rcomp':
cdt( x )
## S3 method for class 'aplus':
cdt( x )
## S3 method for class 'rplus':
cdt( x )
## S3 method for class 'rmult':
cdt( x )
## S3 method for class 'factor':
cdt( x )
```

Arguments

`x`	a classed amount or composition (or a matrix of), to be transformed with its centered default transform

Details

The general idea of this package is to analyse the same data with different geometric concepts as similar as possible. For each of the four concepts there exists a unique transform expressing the geometry in a linear subspace, keeping the relation to the variables. This unique transformation is computed by `cdt`. For `acomp` the transform is `clr`, for `rcomp` it is `cpt`, for `aplus` it is `ilt`, and for `rplus` it is `iit`. Each component of the result is identified with a unit vector in the direction of the corresponding component of the original composition or amount. Keep in mind that the transform is not necessarily surjective and thus variances in the image space might be singular.

Value

A corresponding matrix or vector containing the transforms.

References

See Also

idt, clr, cpt, ilt, iit

Examples

```
## Not run:
# the cdt is defined by
cdt         <- function(x) UseMethod("cdt",x)
cdt.default <- function(x) x
cdt.acomp   <- clr
cdt.rcomp   <- cpt
cdt.aplus   <- ilt
cdt.rplus   <- iit
## End(Not run)
cdt(acomp(1:5))
cdt(rcomp(1:5))
```

`var.acomp` *Variances and covariances of amounts and compositions*

Description

Compute the (co)variance matrix in the several approaches of compositional and amount data analysis.

Usage

```
          var(x,...)
          ## Default S3 method:
          var(x, y=NULL, na.rm=FALSE, use, ...)
          ## S3 method for class 'acomp':
          var(x,y=NULL,...)
          ## S3 method for class 'rcomp':
          var(x,y=NULL,...)
          ## S3 method for class 'aplus':
          var(x,y=NULL,...)
          ## S3 method for class 'rplus':
          var(x,y=NULL,...)
          ## S3 method for class 'rmult':
          var(x,y=NULL,...)
          cov(x,y=x,...)
          ## Default S3 method:
          cov(x, y=NULL, use="all.obs", method=c("pearson",
    "kendall", "spearman"), ...)
          ## S3 method for class 'acomp':
          cov(x,y=NULL,...)
          ## S3 method for class 'rcomp':
```

```
cov(x,y=NULL,...)
## S3 method for class 'aplus':
cov(x,y=NULL,...)
## S3 method for class 'rplus':
cov(x,y=NULL,...)
## S3 method for class 'rmult':
cov(x,y=NULL,...)
```

Arguments

`x`	a dataset, eventually of amounts or compositions
`y`	a second dataset, eventually of amounts or compositions
`na.rm`	see `var`
`use`	see `var`
`method`	see `cov`
`...`	further arguments to `var` e.g. `use`

Details

The basic functions of `var`, `cov` are turned to S3-generics. The original versions are copied to the default method. This allows us to introduce generic methods to handle variances and covariances of other datatypes such as amounts or compositions.

If classed amounts or compositions are involved, they are transformed with their corresponding transforms, using the centered default transform (`cdt`). That implies that the variances have to be interpreded in a log scale level for `acomp` and `aplus`.
We should be aware that variance matrices of compositions are singular. They can be transformed to the correponding nonsingular variances of ilr or ipt -space by `clrvar2ilr`.

In R versions older than v2.0.0, `var` and `cov` were defined in package "base" instead of in "stats". This might produce some misfunction.

Value

The variance matrix of x or the covariance matrix of x and y.

See Also

`cdt`, `clrvar2ilr`, `clo`, `mean.acomp`, `acomp`, `rcomp`, `aplus`, `rplus`, `variation`

Examples

```
data(SimulatedAmounts)
mean.col(sa.lognormals)
var(acomp(sa.lognormals))
var(rcomp(sa.lognormals))
var(aplus(sa.lognormals))
var(rplus(sa.lognormals))
```

```
cov(acomp(sa.lognormals5[,1:3]),acomp(sa.lognormals5[,4:5]))
cov(rcomp(sa.lognormals5[,1:3]),rcomp(sa.lognormals5[,4:5]))
cov(aplus(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))
cov(rplus(sa.lognormals5[,1:3]),rplus(sa.lognormals5[,4:5]))
cov(acomp(sa.lognormals5[,1:3]),aplus(sa.lognormals5[,4:5]))

svd(var(acomp(sa.lognormals)))
```

rcomp   *Compositions as elements of the Simplex embedded in the D-dimensional real space*

Description

A class providing a way to analyse compositions in the philosophical framework of the Simplex as subset of the R^D.

Usage

```
rcomp(X,parts=1:NCOL(oneOrDataset(X)),total=1)
```

Arguments

`X`	composition or dataset of compositions
`parts`	vector containing the indices xor names of the columns to be used
`total`	the total amount to be used, typically 1 or 100

Details

Many multivariate datasets essentially describe amounts of D different parts in a whole. This has some important implications justifying to regard them as a scale on its own, called a composition. The functions around the class `"rcomp"` follow the traditional (but statistically inconsistent) approach regarding compositions simply as a multivariate vector of positive numbers summing up to 1. This space of D positive numbers summing to 1 is traditionally called the D-1-dimensional simplex.

The compositional scale was in-depth analysed by Aitchison (1986) and he found serious reasons why compositional data should be analysed with a different geometry. The functions around the class `"acomp"` follow his approach. However the Aitchison approach based on log-ratios is sometimes criticized (e.g. Rehder and Zier, 2002). It cannot deal with absent parts (i.e. zeros). It is sensitive to large measurement errors in small amounts. The Aitchison operations cannot represent simple mixture of different chemical compositions. The used transformations are not uniformly continuous. Straight lines and ellipses in Aitchison space look strangely in ternary diagrams. As all uncritical statistical analysis, blind application of logratio-based analysis is sometimes misleading. Therefore it is sometimes usefull to analyse compositional data directly as a multivariate dataset of portions summing to 1.

However a clear warning must be given that the utilisation of almost any kind of classical multivariate analysis introduce some kinds of artifacts (e.g. Chayes 1960) when applied to compositional data. So extra care and considerable expert knowlegde is needed for the proper interpretation of results achieved in this non-Aitchison approach. The package tries to lead the user around these artifacts as much as possible and gives hints to major pitfalls in the help. However meaningless results cannot be fully avoided in this (rather inconsistent) approach.
A side effect of the procedure is to force the compositions to sum to one, which is done by the closure operation `clo` .
The classes rcomp, acomp, aplus, and rplus are designed in a fashion as similar as possible, in order to allow direct comparison between results achieved by the different approaches. Especially the acomp logistic transforms `clr`, `alr`, `ilr` are mirrored by analogous linear transforms `cpt`, `apt`, `ipt` in the rcomp class framework.

Value

a vector of class `"rcomp"` representing a closed composition or a matrix of class `"rcomp"` representing multiple closed compositions, by rows.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Rehder, S. and U. Zier (2001) Letter to the Editor: Comment on "Logratio Analysis and Compositional Distance" by J. Aitchison, C. Barceló-Vidal, J.A. Martín-Fernández and V. Pawlowsky-Glahn, *Mathematical Geology*, **33** (7), 845-848.

Zier, U. and S. Rehder (2002) Some comments on log-ratio transformation and compositional distance, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

See Also

`cpt`, `apt`, `ipt`, `acomp`, `rplus`, `princomp.rcomp`, `plot.rcomp`, `boxplot.rcomp`, `barplot.rcomp`, `mean.rcomp`, `var.rcomp`, `variation.rcomp`, `cov.rcomp`, `msd`, `convex.rcomp`, `+.rcomp`

Examples

```
data(SimulatedAmounts)
plot(rcomp(sa.tnormals))
```

plot.acomp	*Displaying compositions in ternary diagrams*

Description

Usage

```
## S3 method for class 'acomp':
plot(x,...,labels=colnames(X),cn=colnames(X),aspanel=FALSE,id=FALSE,idlabs=NULL,idcol=2,center=
## S3 method for class 'rcomp':
plot(x,...,labels=colnames(X),cn=colnames(X),aspanel=FALSE,id=FALSE,idlabs=NULL,idcol=2,center=
```

Arguments

`x`	a dataset of a compositional class
`...`	further graphical parameters passed (see `par`)
`margin`	The type of marginalisation to be computed, when displaying the individual panels. Possible values are: `"acomp"`, `"rcomp"` and any of the variable names/column numbers in the composition. If one of the columns is selected each panel displays a subcomposition given by the row part, the column part and the given part. If one of the classes is given the corresponding margin `acompmargin` or `rcompmargin` is used.
`add`	a logical indicating whether the information should just be added to an existing plot. In case of false a new plot is created.
`triangle`	A logical indicating whether the triangle should be drawn.
`col`	The color to plot the data.
`labels`	The names of the parts
`cn`	The names of the parts to be used in a single panel. Internal use only.
`aspanel`	Logical indicating that only a single panel should be drawn and not the whole plot. Internal use only.
`id`	A logical. If true one can identify the points like with the `identify` command.
`idlabs`	A character vector providing the labels to be used with the identification, when `id=TRUE`
`idcol`	color of the idlabs-labels
`center`	a logical indicating whether a the data should be centered prior to the plot. Centering is done in the choosen philosophy. See `scale`
`scale`	a logical indicating whether a the data should be scaled prior to the plot. Scaling is done in the choosen philosophy. See `scale`

`pca`	A logical indicating whether the first principle component should be displayed in the plot. Currently direction of the principle component of the displayed subcomposition is displayed as a line. Later on a the principle componenent of the whole dataset should be displayed.
`col.pca`	The color to draw the principle component.

Details

The data is displayed in ternary diagrams. This does not work for two part compositions. Compositions of three parts are displayed in a single ternary diagram. For compositions of more than three components, the data is arrange in a scatterplot matrix through the command `pairs`.
The third component in each of the panels is than choosen according to setting of `margin=`. Possible values of `margin=` are: `"acomp"`, `"rcomp"` and any of the variable names/column numbers in the composition. If one of the columns is selected each panel displays a subcomposition given by the row part, the column part and the given part. If one of the classes is given the corresponding margin `acompmargin` or `rcompmargin` is used.
Ternary diagrams can be read in multiple ways. Each corner of the triangle corresponds to a composition only containing the single part displayed in that corner. Points on the edges correspond to compositions only containing the parts in the adjacent corners. The relative amounts are displayed by the distance to the opposite corner. The individual portions of general points can be infered by imaginatorily drawing a line parallel to the edge opposite to the corner of the part of interest through the point. The portion of the part of intrest is constant along the line. Thus we can read it on both crossing points of the line with the edges.
Relative portions of two parts can be inferred by imaginatorily drawing a line through the point and the corner of the unimportant component. This line intersects the edge between the two components of interest in the composition with the same relative portion of the two remaining components.
Exactly the lines parallel to one of the edges or going through one of the corners are straight lines as well in Aitchison and as in real geometry. They remain straight under an arbitrary perturbation.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

Billheimer, D., P. Guttorp, W.F. and Fagan (2001) Statistical interpretation of species composition, *Journal of the American Statistical Association*, **96** (456), 1205-1214

Pawlowsky-Glahn, V. and J.J. Egozcue (2001) Geometric approach to statistical analysis on the simplex. *SERRA* **15**(5), 384-398

http://ima.udg.es/Activitats/CoDaWork03

http://ima.udg.es/Activitats/CoDaWork05

See Also

plot.aplus, qqnorm.acomp,boxplot.acomp

Examples

```
data(SimulatedAmounts)
plot(acomp(sa.lognormals))
plot(rcomp(sa.lognormals))
plot(acomp(sa.lognormals5),pca=TRUE)
plot(rcomp(sa.lognormals5),pca=TRUE)
```

---

ipt *Isometric planar transform*

Description

Compute the isometric planar transform of a (dataset of) composition(s) and its inverse.

Usage

```
ipt( x , V = ilrBase(x) )
ipt.inv( z , V = ilrBase(z=z) )
ucipt.inv( z , V = ilrBase(z=z) )
```

Arguments

x	a composition or a data matrix of compositions, not necessarily closed
z	the ipt-transform of a composition or a data matrix of ipt-transforms of compositions
V	a matrix with columns giving the chosen basis of the clr-plane

Details

The ipt-transform maps a composition in the D-part real-simplex isometrically to a D-1 dimensonal euclidian vector. Although the transformation does not reach the whole R^{D-1}, resulting covariance matrices are typically of full rank.
The data can then be analysed in this transformation by all classical multivariate analysis tools. However, interpretation of results may be difficult, since the transform does not keep the variable names, given that there is no one-to-one relation between the original parts and each transformed variables. See cpt and apt for alternatives.

The isometric planar transform is given by

$$ipt(x) := V^t cpt(x)$$

with cpt(x) the centred planar transform and $V \in R^{d\times(d-1)}$ a matrix which columns form an orthonormal basis of the clr-plane. A default matrix V is given by `ilrBase(`
`var{D})`

Value

`ipt` gives the centered planar transform, `ipt.inv` gives closed compositions with with the given ipt-transforms, `ucipt.inv` unconstrained ipt.inv does the same as ipt.inv but sets illegal values to NA rather then giving an error. This is a workaround to allow procedures not honoring the constraints of the space.

References

See Also

`ilr`,`ilrBase`, `cpt`

Examples

```
(tmp <- ipt(c(1,2,3)))
ipt.inv(tmp)
ipt.inv(tmp) - clo(c(1,2,3)) # 0
data(Hydrochem)
cdata <- Hydrochem[,6:19]
pairs(ipt(cdata))
```

`gsicall` *Internal functions of the compositions package*

Description

Calls a function with a list of arguments.

Usage

```
gsi.call(fkt,...)
```

Arguments

`fkt`	The function to be called
`...`	The arguments to call the function with

Details

This is only useful in conjunction with do.call and allows to call anonymous functions with a parameters given by a list.

Value

Note

Do not use gsi.* functions directly since they are internal functions of the package

See Also

`gsi`

Examples

```
mypars <- list(x=3)
do.call("gsi.call",c(list(function(x){x}),mypars))
```

`SimulatedAmounts` *Simulated amount datasets*

Description

Several simulated datasets intended as reference examples for various conceptual and statistical models of compositions and amounts.

Usage

```
data(SimulatedAmounts)
```

Format

Data matrices with 60 cases and 3 or 5 variables.

Details

The statistical analysis of amounts and compositions is set to discussion. Four essentially different approaches are provided in this package around the classes ”rplus”, ”aplus”, ”rcomp”, ”acomp”. There is no absolutely ”right” approach, since there is a conection between these approaches and the processes originating the data. We provide here simulated standard datasets and the corresponding simulation procedures following these several models to provide “good” analysis examples and to show how these models actually look like in data.

The data sets are simulated according to correlated lognormal distributions (sa.lognormals, sa.lognormal5), winsorised correlated normal distributions (sa.tnormals, sa.tnormal5), Dirichlet distribution on the simplex (sa.dirichlet, sa.dirichlet5), uniform distribution on the simplex (sa.uniform, sa.uniform5), and a grouped dataset (sa.groups, sa.groups5) with three groups (given in sa.groups.area and sa.groups5.area) all distributed accordingly with a lognormal distribution with group dependent means.

We can imagine that amounts evolve in nature e.g. in part of the soil they are diluted and transported in a transport medium, usually water, which comes from independent source (the rain, for instance) and this new composition is normalized by taking a sample of standard size. For each of the datasets *sa.X* there is a corresponding *sa.X*`.dil` dataset which is build by simulating exactly that process on the corresponding *sa.X* dataset . The amounts in the *sa.X*`.dil` are given in ppm. This idea of a transport medium is a major argument for a compositional approach, because the total amount given by the sum of the parts is induced by the dilution given by the medium and thus uninformative for the original investigated process.

If we imagine now these amounts flowing into a river and sedimenting, the different contributions are accumulated along the river and renormalized to a unit portion on taking samples again. For each of the dataset *sa.X*`.dil` there is a corresponding *sa.X*`.mix` dataset which is build from the corresponding *sa.X* dataset by simulating exactly that accumulation process. Mixing of different compositions is a major argument against the log based approaches (`aplus`, `acomp`) since mixing is a highly nonlinear operation in terms of the logratios.

Source

The datasets are simulated for this package and are under the GNU Public Library Licence Version 2 or newer.

References

http://statistic.boogaart.de/compositions/data

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

Zier Rehder

xxx Something recommending to use log-transforms

xxx Something warning against log-transforms

Examples

```
data(SimulatedAmounts)
plot.acomp(sa.lognormals)
plot.acomp(sa.lognormals.dil)
plot.acomp(sa.lognormals.mix)
plot.acomp(sa.lognormals5)
plot.acomp(sa.lognormals5.dil)
plot.acomp(sa.lognormals5.mix)

library(MASS)
plot.rcomp(sa.tnormals)
plot.rcomp(sa.tnormals.dil)
plot.rcomp(sa.tnormals.mix)
plot.rcomp(sa.tnormals5)
plot.rcomp(sa.tnormals5.dil)
plot.rcomp(sa.tnormals5.mix)

plot.acomp(sa.groups,col=as.numeric(sa.groups.area),pch=20)
plot.acomp(sa.groups.dil,col=as.numeric(sa.groups.area),pch=20)
plot.acomp(sa.groups.mix,col=as.numeric(sa.groups.area),pch=20)
plot.acomp(sa.groups5,col=as.numeric(sa.groups.area),pch=20)
plot.acomp(sa.groups5.dil,col=as.numeric(sa.groups.area),pch=20)
plot.acomp(sa.groups5.mix,col=as.numeric(sa.groups.area),pch=20)

plot.acomp(sa.uniform)
plot.acomp(sa.uniform.dil)
plot.acomp(sa.uniform.mix)
plot.acomp(sa.uniform5)
plot.acomp(sa.uniform5.dil)
plot.acomp(sa.uniform5.mix)

plot.acomp(sa.dirichlet)
plot.acomp(sa.dirichlet.dil)
plot.acomp(sa.dirichlet.mix)
plot.acomp(sa.dirichlet5)
plot.acomp(sa.dirichlet5.dil)
plot.acomp(sa.dirichlet5.mix)

# The data was simulated with the following commands:

library(MASS)
dilution <- function(x) {clo(cbind(x,exp(rnorm(nrow(x),5,1))))[,1:ncol(x)]*1E6}
seqmix   <- function(x) {clo(apply(x,2,cumsum))*1E6}

vars  <- c("Cu","Zn","Pb")
vars5 <- c("Cu","Zn","Pb","Cd","Co")

sa.lognormals <- structure(exp(matrix(rnorm(3*60),ncol=3) %*%
                               chol(matrix(c(1,0.8,-0.2,0.8,1,
                                              -0.2,-0.2,-0.2,1),ncol=3))+
                               matrix(rep(c(1:3),each=60),ncol=3)),
                           dimnames=list(NULL,vars))

plot.acomp(sa.lognormals)
pairs(sa.lognormals)

sa.lognormals.dil <- dilution(sa.lognormals)
plot.acomp(sa.lognormals.dil)
```

```
pairs(sa.lognormals.dil)

sa.lognormals.mix <- seqmix(sa.lognormals.dil)
plot.acomp(sa.lognormals.mix)
pairs(sa.lognormals.mix)

sa.lognormals5 <- structure(exp(matrix(rnorm(5*60),ncol=5) %*%
                               chol(matrix(c(1,0.8,-0.2,0,0,
                                             0.8,1,-0.2,0,0,
                                             -0.2,-0.2,1,0,0,
                                             0,0,0,5,4.9,
                                             0,0,0,4.9,5),ncol=5))+
                               matrix(rep(c(1:3,-2,-2),each=60),ncol=5)),
                           dimnames=list(NULL,vars5))

plot.acomp(sa.lognormals5)
pairs(sa.lognormals5)

sa.lognormals5.dil <- dilution(sa.lognormals5)
plot.acomp(sa.lognormals5.dil)
pairs(sa.lognormals5.dil)

sa.lognormals5.mix <- seqmix(sa.lognormals5.dil)
plot.acomp(sa.lognormals5.mix)
pairs(sa.lognormals5.mix)


sa.groups.area <- factor(rep(c("Upper","Middle","Lower"),each=20))
sa.groups <- structure(exp(matrix(rnorm(3*20*3),ncol=3) %*%
                               chol(0.5*matrix(c(1,0.8,-0.2,0.8,1,
                                             -0.2,-0.2,-0.2,1),ncol=3))+
                               matrix(rep(c(1,2,2.5,2,2.9,5,4,2,5),
                                          each=20),ncol=3)),
                           dimnames=list(NULL,c("clay","sand","gravel")))

plot.acomp(sa.groups,col=as.numeric(sa.groups.area),pch=20)
pairs(sa.lognormals,col=as.numeric(sa.groups.area),pch=20)

sa.groups.dil <- dilution(sa.groups)
plot.acomp(sa.groups.dil,col=as.numeric(sa.groups.area),pch=20)
pairs(sa.groups.dil,col=as.numeric(sa.groups.area),pch=20)

sa.groups.mix <- seqmix(sa.groups.dil)
plot.acomp(sa.groups.mix,col=as.numeric(sa.groups.area),pch=20)
pairs(sa.groups.mix,col=as.numeric(sa.groups.area),pch=20)


sa.groups5.area <- factor(rep(c("Upper","Middle","Lower"),each=20))
sa.groups5 <- structure(exp(matrix(rnorm(5*20*3),ncol=5) %*%
                               chol(matrix(c(1,0.8,-0.2,0,0,
                                             0.8,1,-0.2,0,0,
                                             -0.2,-0.2,1,0,0,
                                             0,0,0,5,4.9,
                                             0,0,0,4.9,5),ncol=5))+
                               matrix(rep(c(1,2,2.5,
                                            2,2.9,5,
                                            4,2.5,0,
                                            -2,-1,-1,
                                            -1,-2,-3),
                                          each=20),ncol=5)),
                           dimnames=list(NULL,
                             vars5))

plot.acomp(sa.groups5,col=as.numeric(sa.groups5.area),pch=20)
pairs(sa.groups5,col=as.numeric(sa.groups5.area),pch=20)

sa.groups5.dil <- dilution(sa.groups5)
plot.acomp(sa.groups5.dil,col=as.numeric(sa.groups5.area),pch=20)
pairs(sa.groups5.dil,col=as.numeric(sa.groups5.area),pch=20)

sa.groups5.mix <- seqmix(sa.groups5.dil)
plot.acomp(sa.groups5.mix,col=as.numeric(sa.groups5.area),pch=20)
pairs(sa.groups5.mix,col=as.numeric(sa.groups5.area),pch=20)


sa.tnormals <- structure(pmax(matrix(rnorm(3*60),ncol=3) %*%
                               chol(matrix(c(1,0.8,-0.2,0.8,1,
                                             -0.2,-0.2,-0.2,1),ncol=3))+
                               matrix(rep(c(0:2),each=60),ncol=3),0),
                           dimnames=list(NULL,c("clay","sand","gravel")))

plot.rcomp(sa.tnormals)
pairs(sa.tnormals)

sa.tnormals.dil <- dilution(sa.tnormals)
plot.acomp(sa.tnormals.dil)
pairs(sa.tnormals.dil)

sa.tnormals.mix <- seqmix(sa.tnormals.dil)
plot.acomp(sa.tnormals.mix)
pairs(sa.tnormals.mix)


sa.tnormals5 <- structure(pmax(matrix(rnorm(5*60),ncol=5) %*%
                               chol(matrix(c(1,0.8,-0.2,0,0,
                                             0.8,1,-0.2,0,0,
                                             -0.2,-0.2,1,0,0,
                                             0,0,0,0.05,0.049,
                                             0,0,0,0.049,0.05),ncol=5))+
                               matrix(rep(c(0:2,0.1,0.1),each=60),ncol=5),0),
                           dimnames=list(NULL,
                             vars5))

plot.rcomp(sa.tnormals5)
pairs(sa.tnormals5)
```

```
sa.tnormals5.dil <- dilution(sa.tnormals5)
plot.acomp(sa.tnormals5.dil)
pairs(sa.tnormals5.dil)

sa.tnormals5.mix <- seqmix(sa.tnormals5.dil)
plot.acomp(sa.tnormals5.mix)
pairs(sa.tnormals5.mix)


sa.dirichlet <- sapply(c(clay=0.2,sand=2,gravel=3),rgamma,n=60)
colnames(sa.dirichlet) <- vars

plot.acomp(sa.dirichlet)
pairs(sa.dirichlet)

sa.dirichlet.dil <- dilution(sa.dirichlet)
plot.acomp(sa.dirichlet.dil)
pairs(sa.dirichlet.dil)

sa.dirichlet.mix <- seqmix(sa.dirichlet.dil)
plot.acomp(sa.dirichlet.mix)
pairs(sa.dirichlet.mix)


sa.dirichlet5 <- sapply(c(clay=0.2,sand=2,gravel=3,humus=0.1,plant=0.1),rgamma,n=60)
colnames(sa.dirichlet5) <- vars5

plot.acomp(sa.dirichlet5)
pairs(sa.dirichlet5)

sa.dirichlet5.dil <- dilution(sa.dirichlet5)
plot.acomp(sa.dirichlet5.dil)
pairs(sa.dirichlet5.dil)

sa.dirichlet5.mix <- seqmix(sa.dirichlet5.dil)
plot.acomp(sa.dirichlet5.mix)
pairs(sa.dirichlet5.mix)

sa.uniform   <- sapply(c(clay=1,sand=1,gravel=1),rgamma,n=60)
colnames(sa.uniform) <- vars

plot.acomp(sa.uniform)
pairs(sa.uniform)

sa.uniform.dil <- dilution(sa.uniform)
plot.acomp(sa.uniform.dil)
pairs(sa.uniform.dil)

sa.uniform.mix <- seqmix(sa.uniform.dil)
plot.acomp(sa.uniform.mix)
pairs(sa.uniform.mix)
```

```
sa.uniform5   <- sapply(c(clay=1,sand=1,gravel=1,humus=1,plant=1),rgamma,n=60)
colnames(sa.uniform5) <- vars5

plot.acomp(sa.uniform5)
pairs(sa.uniform5)

sa.uniform5.dil <- dilution(sa.uniform5)
plot.acomp(sa.uniform5.dil)
pairs(sa.uniform5.dil)

sa.uniform5.mix <- seqmix(sa.uniform5.dil)
plot.acomp(sa.uniform5.mix)
pairs(sa.uniform5.mix)

objects(pattern="sa.*")
```

`rlnorm` *The multivariate lognormal distribution*

Description

Generates random amounts with a multivariate lognormal distribution .

Usage

```
rlnorm.rplus(n,meanlog,varlog)
dlnorm.rplus(x,meanlog,varlog)
```

Arguments

`n`	number of datasets to be simulated
`meanlog`	The mean-vector of the logs
`varlog`	The variance/covariance matrix of the logs
`x`	vectors in the sample space

Value

`rlnorm.rplus` gives a generated random dataset of class `"rplus"` following a lognormal distribution with logs having mean `meanlog` and variance `varlog`.
`dlnorm.rplus` gives the density of the distribution with respect to the Lesbesgue measure on R+ as a subset of R.

Note

The main difference between `rlnorm.rplus` and `rnorm.aplus` is that rlnorm.rplus needs a loged mean. The additional difference for the calculation of the density by `dlnorm.rplus` and `dnorm.aplus` is the reference measure.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

rnorm.acomp

Examples

```
MyVar <- matrix(c(
0.2,0.1,0.0,
0.1,0.2,0.0,
0.0,0.0,0.2),byrow=TRUE,nrow=3)
MyMean <- c(1,1,2)

plot(rlnorm.rplus(100,log(MyMean),MyVar))
plot(rnorm.aplus(100,MyMean,MyVar))
x <- rnorm.aplus(5,MyMean,MyVar)
dnorm.aplus(x,MyMean,MyVar)
dlnorm.rplus(x,log(MyMean),MyVar)
```

---

names *The names of the parts*

Description

The names function provide a transparent way to access the names of the parts regardless of the shape of the dataset or data item.

Usage

```
## S3 method for class 'acomp':
names(x)
## S3 method for class 'rcomp':
names(x)
## S3 method for class 'aplus':
names(x)
## S3 method for class 'rplus':
names(x)
## S3 method for class 'rmult':
names(x)
## S3 method for class 'acomp':
names(x) <- value
## S3 method for class 'rcomp':
names(x) <- value
```

```
## S3 method for class 'aplus':
names(x) <- value
## S3 method for class 'rplus':
names(x) <- value
## S3 method for class 'rmult':
names(x) <- value
```

Arguments

`x`	an amount/amount dataset
`value`	the new names of the parts
`...`	not used, only here for generics

Value

a character vector giving the names of the parts

See Also

aplus

Examples

```
data(SimulatedAmounts)
tmp <- acomp(sa.lognormals)
names(tmp)
names(tmp) <- c("x","y","z")
tmp
```

---

groupparts *Group amounts of parts*

Description

Groups parts by amalgamation or balancing of their amounts or proportions.

Usage

```
groupparts(x,...)
## S3 method for class 'acomp':
groupparts(x,...,groups=list(...))
## S3 method for class 'rcomp':
groupparts(x,...,groups=list(...))
## S3 method for class 'aplus':
groupparts(x,...,groups=list(...))
## S3 method for class 'rplus':
groupparts(x,...,groups=list(...))
```

Arguments

`x`	an amount/compositional dataset
`...`	further parameters to use (actually ignored)
`groups`	a list of numeric xor character vectors, each giving a group of parts

Details

In the real geometry grouping is done by amalgamation (i.e. adding the parts). In the Aitchison-geometry grouping is done by taking geometric means. The new parts are named by named formal arguments. Not-mentioned parts remain ungrouped.

Value

a new dataset of the same type with each group represented by a single column

References

Egozcue, J.J. and V. Pawlowsky-Glahn (2005) Groups of Parts and their Balances in Compositional Data Analysis, Mathematical Geology, in press

See Also

`aplus`

Examples

```
data(SimulatedAmounts)
groupparts(acomp(sa.lognormals5),A=c(1,2),B=c(3,4),C=5)
groupparts(aplus(sa.lognormals5),B=c(3,4),C=5)
groupparts(rcomp(sa.lognormals5),A=c("Cu","Pb"),B=c(2,5))
groupparts(rplus(sa.lognormals5),1:5)
```

`aplusarithm` *vectorial arithmetic for datasets with aplus class*

Description

The positive vectors equipped with the perturbation (defined as the element-wise product) as Abelian sum, and powertransform (defined as the element-wise powering with a scalar) as scalar multiplication forms a real vector space. These vector space operations are defined here in a similar way to `+.rmult`.

Usage

```
perturbe.aplus(x,y)
## Methods for aplus
##   x+y
##   x-y
##   -x
##   x*r
##   r*x
##   x/r
power.aplus(x,r)
```

Arguments

x	an aplus vector or dataset of vectors
y	an aplus vector or dataset of vectors
r	a numeric vector of size 1 or nrow(x)

Details

The operators try to mimic the parallel operation of R for vectors of real numbers to vectors of amounts, represented as matrices containing the vectors as rows and works like the operators for {rmult}

Value

an object of class "aplus" containing the result of the corresponding operation on the vectors.

See Also

rmult, %*%.rmult

Examples

```
x <- aplus(matrix( sqrt(1:12), ncol= 3 ))
x
x+x
x + aplus(1:3)
x * 1:4
1:4 * x
x / 1:4
x / 10
power.aplus(x,1:4)
```

gsi.getD — *Interal function: Get number of samples and number of parts in a compositional object*

Description

Get the number of samples N and the number of parts D of the compositions in an `acomp`, `rcomp`, `aplus`, `rplus` object.

Usage

```
gsi.getD(x)
gsi.getN(x)
```

Arguments

`x`	an `acomp`, `rcomp`, `aplus`, `rplus` object or something that could be cast to one of them

Value

an integer giving the number of parts D or the number of samples N.

Note

Do not use gsi.* functions directly since they are internal functions of the package

Examples

```
gsi.getD(1:5)
gsi.getN(1:5)
NCOL(1:5)
NROW(1:5)
data(SimulatedAmounts)
gsi.getD(sa.lognormals5)
gsi.getN(sa.lognormals5)
```

mean.acomp — *Mean amounts and mean compositions*

Description

Compute the mean in the several approaches of compositional and amount data analysis.

Usage

```
mean.acomp(x,..., na.action=get(getOption("na.action")))
mean.rcomp(x,..., na.action=get(getOption("na.action")))
mean.aplus(x,..., na.action=get(getOption("na.action")))
mean.rplus(x,..., na.action=get(getOption("na.action")))
mean.rmult(x,..., na.action=get(getOption("na.action")))
```

Arguments

`x`	a classed dataset of amounts or compositions
`...`	further arguments to `mean` e.g. `trim`
`na.action`	The na.action to be used: one of `na.omit`,`na.fail`,`na.pass`

Details

The different compositional approaches `acomp`, `rcomp`, `aplus`, `rplus` correpond to different geometries. The mean is calculated in the respective canonical geometry by applying a canonical transform (see `cdt`), taking ordinary `mean.col` and backtransforming.

The Aitchison geometries imply that `mean.acomp` and `mean.aplus` are geometric means, the first one closed. The real geometry implies that `mean.rcomp` and `mean.rplus` are arithmetic means, the first one resulting in a closed composition.

In all cases the mean is again an object of the same class.

Value

The mean is given as a composition or amount vector of the same class as the original dataset.

See Also

`clo`, `mean.col`, `geometricmean`, `acomp`, `rcomp`, `aplus`, `rplus`

Examples

```
data(SimulatedAmounts)
mean.col(sa.lognormals)
mean(acomp(sa.lognormals))
mean(rcomp(sa.lognormals))
mean(aplus(sa.lognormals))
mean(rplus(sa.lognormals))
mean(rmult(sa.lognormals))
```

`ilrBase`	*The canonical basis in the clr plane used for ilr and ipt transforms.*

Description

Compute the basis of a clr-plane, to use with isometric log-ratio or planar transform of a (dataset of) compositions.

Usage

```
ilrBase( x=NULL , z=NULL , D = NULL )
gsi.ilrBase(D)
```

Arguments

`x`	optional dataset or vector of compositions
`z`	optional dataset or vector containing ilr or ipt coordinates
`D`	number of parts of the simplex

Details

ilrBase is a wrapper catching the answers of gsi.ilrBase and is to be used as the more convenient function. Only one of the arguments is needs to determine the dimension of the simplex.

Value

Both methods give a matrix containing by columns the basis elements for the canonical basis of the clr-plane used for the ilr and ipt transform.

References

Egozcue J.J., V. Pawlowsky-Glahn, G. Mateu-Figueras and C. Barcel'o-Vidal (2003) Isometric logratio transformations for compositional data analysis. *Mathematical Geology*, **35**(3) 279-300

See Also

`clr`,`ilr`,`ipt`, `http://ima.udg.es/Activitats/CoDaWork03`

Examples

```
ilr(c(1,2,3))
ilrBase(D=2)
ilrBase(c(1,2,3))
ilrBase(z= ilr(c(1,2,3)) )
round(ilrBase(D=7),digits= 3)
```

`clr2ilr` *Convert between clr and ilr, and between cpt and ipt. Acts in vectors and in bilinear forms.*

Description

Compute the centered log ratio transform of a (dataset of) isometric log-ratio transform(s) and its inverse. Equivalently, compute centered and isometric planar transforms from each other.

Usage

```
clr2ilr( x , V=ilrBase(x) )
ilr2clr( z , V=ilrBase(z=z) )
clrvar2ilr( varx , V=ilrBase(D=ncol(varx)) )
ilrvar2clr( varz , V=ilrBase(D=ncol(varz)+1) )
```

Arguments

`x`	the clr/cpt-transform of composition(s)
`z`	the ilr/ipt-transform of composition(s)
`varx`	variance or covariance matrix of clr/cpt-transformed compositions
`varz`	variance or covariance matrix of ilr/ipt-transformed compositions
`V`	a matrix with columns giving the chosen basis of the clr-plane

Details

These functions perform a matrix multiplication with `V` in an appropriate way.

Value

`clr2ilr` gives the ilr/ipt transform of the same composition(s),
`ilr2clr` gives the clr/cpt transform of the same composition(s),
`clrvar2ilr` gives the variance-/covariance-matrix of the ilr/ipt transform of the same compositional data set,
`ilrvar2clr` gives the variance-/covariance-matrix of the clr/cpt transform of the same compositional data set.

References

Egozcue J.J., V. Pawlowsky-Glahn, G. Mateu-Figueras and C. Barcel'o-Vidal (2003) Isometric logratio transformations for compositional data analysis. *Mathematical Geology*, **35**(3) 279-300
Aitchison, J, C. Barcel'o-Vidal, J.J. Egozcue, V. Pawlowsky-Glahn (2002) A consise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis, *Terra Nostra*, Schriften der Alfred Wegener-Stiftung, 03/2003

See Also

ilr, ipt

Examples

```
data(SimulatedAmounts)
ilr.inv(clr2ilr(clr(sa.lognormals)))-clo(sa.lognormals)
clr.inv(ilr2clr(ilr(sa.lognormals)))-clo(sa.lognormals)
ilrvar2clr(var(ilr(sa.lognormals)))-var(clr(sa.lognormals))
clrvar2ilr(var(cpt(sa.lognormals)))-var(ipt(sa.lognormals))
```

---

rcompmargin *Marginal compositions in real geometry*

Description

Compute marginal compositions by amalgamating the rest (additively).

Usage

```
rcompmargin(X,d=c(1,2),name="+",pos=length(d)+1)
```

Arguments

X	composition or dataset of compositions
d	vector containing the indices xor names of the columns to be kept
name	The new name of the amalgamation column
pos	The position where the new amalgamation column should be stored. This defaults to the last column.

Details

The amalgamation column is simply computed by adding the non-selected components after closing the composition. This is consistent with the rcomp approach and is widely used because of its easy interpretation. However, it often leads to difficult-to-read ternary diagrams and is inconsistent with the acomp approach.

Value

A closed compositions with class `"rcomp"` containing the selected variables given by `d` and the the amalgamation column.

References

See Also

`acompmargin`, `rcomp`

Examples

```
data(SimulatedAmounts)
plot.rcomp(sa.tnormals5,margin="rcomp")
plot.rcomp(rcompmargin(sa.tnormals5,c("Cd","Zn")))
plot.rcomp(rcompmargin(sa.tnormals5,c(1,2)))
```

`clo` *Closure of a composition*

Description

Closes compositions to sum up to one (or an optional total), by dividing each part by the sum.

Usage

```
clo( X, parts=1:NCOL(oneOrDataset(X)),total=1)
```

Arguments

`X`	composition or dataset of compositions
`parts`	vector containing the indices xor names of the columns to be used
`total`	the total amount to which the compositions should be closed; either a single number, or a numeric vector of length `gsi.getN(X)` specifying a different total for each compositional vector in the dataset.

Details

The closure operation is given by

$$clo(x) := \left(x_i / \sum_{j=1}^{D} x_j \right)$$

`clo` makes a composition without assigning one of the compositional classes `acomp` or `rcomp`. Note that after computing the closed-to-one version, obtaining a version closed to any other value is done by simple multiplication.

Value

a composition or a datamatrix of compositions without compositional class. The individual compositions are forced to sum to 1 (or to the optionally-specified total). The result should have the same shape as the input (vector, row, matrix).

Note

`clo` can be used to unclass compositions.

References

Aitchison, J. (1986) *The Statistical Analysis of Compositional Data* Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London (UK). 416p.

See Also

`clr`,`acomp`,`rcomp`

Examples

```
(tmp <- clo(c(1,2,3)))
clo(tmp,total=100)
data(Hydrochem)
cdata <- Hydrochem[,6:19]
plot( clo(Hydrochem,8:9) ) # Giving points on a line
```