Package 'coda.base'

May 9, 2018

Type Package

Title A Basic Set of Functions for Compositional Data Analysis

Version 0.1.9

Date 2018-05-03

Description A minimum set of functions to perform compositional data analysis using the log-ratio approach introduced by John Aitchison (1982) <http://www.jstor.org/stable/2345821>. Main functions have been implemented in c++ for better performance.

Depends R (>= 3.0.4)

Imports Rcpp (>= 0.12.12), MASS

LinkingTo Rcpp, RcppArmadillo

License GPL

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 6.0.1

Author Marc Comas-Cufí [aut, cre]

Maintainer Marc Comas-Cufí <mcomas@imae.udg.edu>

Repository CRAN

Date/Publication 2018-05-09 09:20:27 UTC

R topics documented:

`alr_basis` *Additive log-ratio basis*

Description

Compute the transformation matrix to express a composition using the oblique additive log-ratio coordinates.

Usage

```
alr_basis(dim, denominator = dim, numerator = which(denominator != 1:dim))
```

Arguments

`dim`	number of parts
`denominator`	part used as denominator (default behaviour is to use last part)
`numerator`	parts to be used as numerator. By default all except the denominator parts are chosen following original order.

Value

matrix

References

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

Examples

```
alr_basis(5)
# Third part is used as denominator
alr_basis(5, 3)
# Third part is used as denominator, and
# other parts are rearranged
alr_basis(5, 3, c(1,5,2,4))
```

clr_basis *Centered log-ratio basis*

Description

Compute the transformation matrix to express a composition using the linearly dependant centered log-ratio coordinates.

Usage

```
clr_basis(dim)
```

Arguments

dim	number of parts

Value

matrix

References

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

Examples

```
(B <- clr_basis(5))
# CLR coordinates are linearly dependant coordinates.
(clr_coordinates <- coordinates(c(1,2,3,4,5), B))
# The sum of all coordinates equal to zero
sum(clr_coordinates) < 1e-15
```

composition *Get composition from coordinates w.r.t. an specific basis*

Description

Calculate a composition from coordinates with respect a given basis

Usage

```
composition(H, basis = NULL, label = "x", sparse_basis = FALSE)
```

Arguments

H	coordinates of a composition. Either a matrix, a data.frame or a vector
basis	basis used to calculate the coordinates
label	name given to the coordinates
sparse_basis	Is the given matrix basis sparse? If TRUE calculation are carried taking into an account sparsity (default 'FALSE')

Value

coordinates with respect the given basis

See Also

See functions ilr_basis, alr_basis, clr_basis, sbp_basis to define different compositional basis. See function coordinates to obtain details on how to calculate coordinates of a given composition.

---

coordinates *Get coordinates from compositions w.r.t. an specific basis*

Description

Calculate the coordinates of a composition with respect a given basis

Usage

```
coordinates(X, basis = "ilr", label = "x", sparse_basis = FALSE)
```

Arguments

X	compositional dataset. Either a matrix, a data.frame or a vector
basis	basis used to calculate the coordinates. basis can be either a string or a matrix. Accepted values for strings are: 'ilr' (default), 'clr', 'alr' and 'pc'. If basis is a matrix, it is expected to have log-ratio basis given in columns.
label	name given to the coordinates
sparse_basis	Is the given matrix basis sparse? If TRUE calculation are carried taking into an account sparsity (default 'FALSE')

Details

coordinates function calculates the coordinates of a compositiona w.r.t. a given basis. 'basis' parameter is used to set the basis, it can be either a matrix defining the log-contrasts in columns or a string defining some well-known log-contrast: 'alr' 'clr', 'ilr' or 'pc' for the additive log-ratio, centered log-ratio, isometric log-ratio or clr principal components respectively.

Value

Coordinates of composition `X` with respect the given `basis`.

See Also

See functions `ilr_basis`, `alr_basis`, `clr_basis`, `sbp_basis` to define different compositional basis. See function `composition` to obtain details on how to calculate a compositions from given coordinates.

Examples

```
coordinates(c(1,2,3,4,5))
# basis is shown if 'coda.base.basis' option is set to TRUE
options('coda.base.basis' = TRUE)
coordinates(c(1,2,3,4,5))
# Setting sparse_basi to TRUE can improve performance if log-ratio basis is sparse.
N = 100
K = 1000
X = matrix(exp(rnorm(N*K)), nrow=N, ncol=K)
system.time(coordinates(X, alr_basis(K), sparse_basis = FALSE))
system.time(coordinates(X, alr_basis(K), sparse_basis = TRUE))
system.time(coordinates(X, 'alr'))
```

`dist` *Distance Matrix Computation (including Aitchison distance)*

Description

This function overwrite `dist` function to contain Aitchison distance between compositions.

Usage

```
dist(x, method = "euclidean", ...)
```

Arguments

`x`	compositions method
`method`	the distance measure to be used. This must be one of "aitchison", "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
`...`	arguments passed to `dist` function

Value

`dist` returns an object of class "dist".

See Also

See functions dist.

Examples

```
X = exp(matrix(rnorm(10*50), ncol=50, nrow=10))

(d <- dist(X, method = 'aitchison'))
plot(hclust(d))

# In contrast to Euclidean distance
dist(rbind(c(1,1,1), c(100, 100, 100)), method = 'euc') # method = 'euclidean'
# using Aitchison distance, only relative information is of importance
dist(rbind(c(1,1,1), c(100, 100, 100)), method = 'ait') # method = 'aitchison'
```

---

ilr_basis *Default Isometric log-ratio basis*

Description

Build an isometric log-ratio basis for a composition with k+1 parts

$$h_i = \sqrt{\frac{i}{i+1}} \log \frac{\sqrt[i]{\prod_{j=1}^{i} x_j}}{x_{i+1}}$$

for $iin1 \dots k$

Usage

```
ilr_basis(dim)
```

Arguments

dim	number of components

Value

matrix

References

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

Examples

```
ilr_basis(5)
```

pb_basis *Isometric log-ratio basis based on Principal Balances.*

Description

Different approximations to approximate the principal balances of a compositional dataset.

Usage

```
pb_basis(X, method, rep = 0, ordering = TRUE, ...)
```

Arguments

X	compositional dataset
method	method to be used with Principal Balances. Methods available are: 'exact', 'lsearch' or method to be passed to hclust function (for example 'ward.D' or 'ward.D2' to use Ward method).
rep	Number of restartings to be used with the local search algorithm. If zero is supplied (default), one local search is performed using an starting point close to the principal component solution.
ordering	should the principal balances found be returned ordered? (first column, first principal balance and so on)
...	parameters passed to hclust function

Value

matrix

References

Pawlowsky-Glahn, V., Egozcue, J.J., Tolosana-Delgado R. (2011). *Principal balances*. in proceeding of the 4th International Workshop on Compositional Data Analysis (CODAWORK'11) (available online at http://www-ma3.upc.edu/users/ortego/codawork11-Proceedings/Admin/Files/FilePaper/p55.pdf)

Examples

```
set.seed(1)
X = matrix(exp(rnorm(5*100)), nrow=100, ncol=5)
# Optimal variance obtained with Principal components
(v1 <- apply(coordinates(X, 'pc'), 2, var))
# Optimal variance obtained with Principal balances
(v2 <- apply(coordinates(X,pb_basis(X, method='exact')), 2, var))
# Solution obtained using a hill climbing algorithm from pc approximation
apply(coordinates(X,pb_basis(X, method='lsearch')), 2, var)
# Solution obtained using a hill climbing algorithm using 10 restartings
apply(coordinates(X,pb_basis(X, method='lsearch', rep=10)), 2, var)
```

```
# Solution obtained using Ward method
(v3 <- apply(coordinates(X,pb_basis(X, method='ward.D2')), 2, var))
# Solution obtained using Old Ward function (in R versions <= 3.0.3)
apply(coordinates(X,pb_basis(X, method='ward.D')), 2, var)
# Plotting the variances
barplot(rbind(v1,v2,v3), beside = TRUE, legend = c('PC','PB','Ward'), args.legend = list(cex = 0.8))
```

`print.coda` *Printing coordinates*

Description

The function hides the basis attribute. An option is included to show such basis.

Usage

```
## S3 method for class 'coda'
print(x, ..., basis = getOption("coda.base.basis"))
```

Arguments

`x`	coordinates
`...`	parameters passed to print function
`basis`	boolean to show or not the basis with the output

`sbp_basis` *Isometric log-ratio basis based on Balances Build an* `ilr_basis` *using a sequential binary partition or a generic coordinate system based on balances.*

Description

Isometric log-ratio basis based on Balances Build an `ilr_basis` using a sequential binary partition or a generic coordinate system based on balances.

Usage

```
sbp_basis(..., data, silent = F)
```

Arguments

`...`	balances to consider
`data`	composition from where name parts are extracted
`silent`	inform about orthgonality

Value

matrix

Examples

```
X = data.frame(a=1:2, b=2:3, c=4:5, d=5:6, e=10:11, f=100:101, g=1:2)
sbp_basis(b1 = a~b+c+d+e+f+g,
          b2 = b~c+d+e+f+g,
          b3 = c~d+e+f+g,
          b4 = d~e+f+g,
          b5 = e~f+g,
          b6 = f~g, data = X)
sbp_basis(b1 = a~b,
         b2 = b1~c,
         b3 = b2~d,
         b4 = b3~e,
         b5 = b4~f,
         b6 = b5~g, data = X)
# A non-orthogonal basis can also be calculated.
sbp_basis(b1 = a+b+c~e+f+g,
          b2 = d~a+b+c,
          b3 = d~e+g,
          b4 = a~e+b,
          b5 = b~f,
          b6 = c~g, data = X)
```

---

variation_array *Variation array is returned.*

Description

Variation array is returned.

Usage

```
variation_array(X, only_variation = FALSE)
```

Arguments

X	Compositional dataset
only_variation	if TRUE only the variation matrix is calculated

Value

variation array matrix

Examples

```
set.seed(1)
X = matrix(exp(rnorm(5*100)), nrow=100, ncol=5)
variation_array(X)
variation_array(X, only_variation = TRUE)
```

Index