Package ‘coda.base’

May 14, 2020

Type Package
Title A Basic Set of Functions for Compositional Data Analysis
Version 0.3.1
Date 2020-05-14
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.

URL https://mcomas.github.io/coda.base,
https://github.com/mcomas/coda.base

Depends R (>= 3.0.4)
Imports Rcpp (>= 0.12.12), stats
LinkingTo Rcpp, RcppArmadillo
License GPL
Encoding UTF-8
LazyData true
NeedsCompilation yes
RoxygenNote 7.1.0
Suggests knitr, rmarkdown, testthat (>= 2.1.0)
VignetteBuilder knitr
Author Marc Comas-Cufí [aut, cre] (<https://orcid.org/0000-0001-9759-0622>)
Maintainer Marc Comas-Cufí <mcomas@imae.udg.edu>
Repository CRAN
Date/Publication 2020-05-14 17:10:29 UTC
R topics documented:

alr_basis ....................................................... 2
basis .............................................................. 3
cbalance_approx ................................................. 3
cc_basis ............................................................ 4
cdp_basis .......................................................... 4
cdp_partition ...................................................... 5
clr_basis ........................................................... 5
coda.base .......................................................... 6
composition ........................................................ 6
coordinates .......................................................... 7
dist ................................................................. 8
ilr_basis ............................................................. 9
parliament2017 ..................................................... 10
pb_basis ............................................................. 10
pc_basis ............................................................. 12
print.coda ........................................................ 12
sbp_basis ........................................................... 13
variation_array .................................................... 14

Index 15

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


Examples

```r
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))
```

basis Coordinates basis

<table>
<thead>
<tr>
<th>basis</th>
<th>Coordinates basis</th>
</tr>
</thead>
</table>

Description

Obtain coordinates basis

Usage

```r
basis(H)
```

Arguments

- `H` coordinates for which basis should be shown

Value

basis used to create coordinates H

cbalance_approx Balance generated from the first canonical correlation component

Description

Balance generated from the first canonical correlation component

Usage

```r
cbalance_approx(Y, X)
```

Arguments

<table>
<thead>
<tr>
<th>Y</th>
<th>compositional dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>explanatory dataset</td>
</tr>
</tbody>
</table>
# cc_basis

**Isometric log-ratio basis based on canonical correlations**

## Description

Isometric log-ratio basis based on canonical correlations

## Usage

```r
cc_basis(Y, X)
```

## Arguments

- `Y`: compositional dataset
- `X`: explanatory dataset

## Value

matrix

---

# cdp_basis

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

## Description

The function return default balances used in CoDaPack software.

## Usage

```r
cdp_basis(dim)
```

## Arguments

- `dim`: dimension to build the ILR basis based on balanced balances

## Value

matrix
Description

Compute the default binary partition used in CoDaPack’s software

Usage

cdp_partition(ncomp)

Arguments

ncomp number of parts

Value

matrix

Examples

cdp_partition(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

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

coda.base

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.

Author(s)
Marc Comas-Cufí

composition

Description
Get composition from coordinates w.r.t. an specific 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 = ifelse(is.character(basis), basis, "h"),
  basis_return = TRUE
)

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', 'pc', 'pb' and 'cdp'. If basis is a matrix, it is expected to have log-ratio basis given in columns.
label  name given to the coordinates
basis_return  Should the basis be returned as attribute? (default: TRUE)

Details

coordinates function calculates the coordinates of a composition 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', 'pc', 'pb' and 'cdp', for the additive log-ratio, centered log-ratio, isometric log-ratio, clr principal components, clr principal balances or default's CoDaPack balances 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))
# Default transformation improves performance.
N = 100
K = 1000
X = matrix(exp(rnorm(N*K)), nrow=N, ncol=K)
system.time(coordinates(X, alr_basis(K)))
system.time(coordinates(X, 'alr'))

---

dist

Distance Matrix Computation (including Aitchison distance)

Description

This function overwrites dist function to contain Aitchison distance between compositions.

Usage

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

Arguments

- x: compositions
- 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'
ilr_basis

# 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

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

for \( i \in 1 \ldots k \).

Usage

ilr_basis(dim, type = "default")

Arguments

dim number of components

type if different than 'pivot' (pivot balances) or 'cdp' (codapack balances) default balances are returned, which computes a triangular Helmert matrix as defined by Egozcue et al., 2013.

Details

Modifying parameter type (pivot or cdp) other ilr basis can be generated

Value

matrix

References


Examples

ilr_basis(5)
parliament2017  

Results of catalan parliament elections in 2017 by regions.

Description  
Results of catalan parliament elections in 2017 by regions.

Usage  
parliament2017

Format  
A data frame with 42 rows and 9 variables:

- **com**  Region
- **cs**  Votes to Ciutadans party
- **jxcat**  Votes to Junts per Catalunya party
- **erc**  Votes to Esquerra republicana de Catalunya party
- **psc**  Votes to Partit socialista de Catalunya party
- **catsp**  Votes to Catalunya si que es pot party
- **cup**  Votes to Candidatura d’unitat popular party
- **pp**  Votes to Partit popular party
- **other**  Votes to other parties

Source  
http://www.idescat.cat/tema/elecc

---

pb_basis  

Isometric log-ratio basis based on Principal Balances.

Description  
Exact method to calculate the principal balances of a compositional dataset. Different methods to approximate the principal balances of a compositional dataset are also included.

Usage  

```r
pb_basis(
  X,
  method,
  constrained.complete_up = FALSE,
  cluster.method = "ward.D2",
  ordering = TRUE,
  ...
)
```
**Arguments**

- **X**
  - compositional dataset

- **method**
  - method to be used with Principal Balances. Methods available are: 'exact', 'constrained' or 'cluster'.

- **constrained.complete.up**
  - When searching up, should the algorithm try to find possible siblings for the current balance (TRUE) or build a parent directly forcing current balance to be part of the next balance (default: FALSE). While the first is more exhaustive and given better results the second is faster and can be used with high dimensional datasets.

- **cluster.method**
  - Method to be used with the hclust function (default: 'ward.D2') or any other method available in hclust function

- **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**


**Examples**

```r
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 constrained method
(v3 <- apply(coordinates(X,pb_basis(X, method='constrained')), 2, var))

# Solution obtained using Ward method
(v4 <- apply(coordinates(X,pb_basis(X, method='cluster')), 2, var))

# Plotting the variances
barplot(rbind(v1,v2,v3,v4), beside = TRUE, ylim = c(0,2),
        legend = c('Principal Components', 'PB (Exact method)',
                    'PB (Constrained)', 'PB (Ward approximation)'),
        names = paste0('Comp.', 1:4), args.legend = list(cex = 0.8), ylab = 'Variance')
```
**pc_basis**

*Isometric log-ratio basis based on Principal Components.*

**Description**

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

**Usage**

```r
cp_basis(X)
```

**Arguments**

- `X`: compositional dataset

**Value**

matrix

**print.coda**

*Printing coordinates*

**Description**

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

**Usage**

```r
## 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
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**

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

**Arguments**

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

**Value**

matrix

**Examples**

```r
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 Variations 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

*Topic datasets
  parliament2017, 10
  alr_basis, 2, 6, 7
  basis, 3
  cbalance_approx, 3
  cc_basis, 4
  cdp_basis, 4
  cdp_partition, 5
  clr_basis, 5, 6, 7
  coda.base, 6
  composition, 6, 7
  coordinates, 6, 7
  dist, 8, 8
  ilr_basis, 6, 7, 9, 13
  parliament2017, 10
  pb_basis, 10
  pc_basis, 12
  print.coda, 12
  sbp_basis, 6, 7, 13
  variation_array, 14