Package 'SARP.compo'

May 15, 2021

Type Package

Title Network-Based Interpretation of Changes in Compositional Data

Version 0.1.5

Date 2021-05-15

Maintainer Emmanuel Curis <emmanuel.curis@parisdescartes.fr>

License Artistic-2.0

Description Provides a set of functions to interpret changes in compositional data based on a network representation of all pairwise ratio comparisons: computation of all pairwise ratio, construction of a p-value matrix of all pairwise tests of these ratios between conditions, conversion of this matrix to a network.

Suggests lme4

Depends igraph

Imports car

Encoding UTF-8

NeedsCompilation no

Author Emmanuel Curis [aut, cre] (<https://orcid.org/0000-0001-8382-1493>)

Repository CRAN

Date/Publication 2021-05-15 19:10:02 UTC

R topics documented:

arbre.Mp 2
BpLi 4
calc.rapports 5
choisir.seuil 6
choisir.seuil.equiv 9
conversions 11
coupures.Mp 12
creer_data.frame 14
creer_graphe 16

creer_matrice 18
distances 20
equivalence 22
fichiers 23
GADL1 26
generef 27
modele 28
poteries 30
puissance 32
SCH23390 35
tests 37

Index **40**

`arbre.Mp` *Grouping composants by changes in cut-off separation*

Description

These functions construct a tree based on the successive disjunctions between nodes of the graph when increasing the cut-off value.

Usage

```
arbre.Mp( Mp, en.log = FALSE, reference = NA, complement = FALSE )

## S3 method for class 'Arbre'
plot(x, seuil.p = 0.05,
                      xlab = "Composant",
                      ylab = if ( TRUE == en.log ) "-log seuil" else "Seuil",
                      col.seuil = "red"  , lwd.seuil = 1, lty.seuil = 1,
                      horiz = FALSE, center = TRUE, edge.root = TRUE,
                      ...)
```

Arguments

`Mp`	A square, symmetric matrix containing p-values. Element in row i and line j should contain the p-value for testing the $\frac{i}{j}$ ratio. The diagonal is ignored.
`en.log`	If `TRUE`, p-values are log-transformed (using decimal logarithm) to construct the tree. It does not change the tree structure, it only helps visualisation of the small p-value part of the tree.
`reference`	Either `NA` (the default) or a vector giving the names of reference genes. Corresponding leaves will then be drawn in orange, whereas leaves for genes of interests will be drawn in palegreen (like graphs).
`complement`	A logical. If TRUE, the tree is built using the complement of the graph, as when using equivalence test to build the graph.
`x`	The tree to be drawn

`seuil.p`	Selected cut-off for analysis. Can also be a `SARPcompo.H0` object, as returned by `choisir.seuil`, in which case the bounds of the confidence interval are also drawn, with dashed lines by default.
`xlab,ylab`	Legends for the axes
`col.seuil, lwd.seuil, lty.seuil`	Graphical parameters for drawing the analysis cut-off
`horiz, center, edge.root`	Options from `plot.dendrogram` with different defaults or needed for complementary plottings. If `TRUE`, the tree is drawn, respectivally, horizontally instead of vertically, with edges "centered", and with edge to the root node. See the documentation from `plot.dendrogram` for details.
`...`	Additionnal parameters for `plot.dendrogram`, which is used internally.

Details

By increasing the cut-off from 0 to 1, more and more edges between nodes are removed, and disjoint subgraphs appear. This can be used to build a tree of the composants, with nodes of the tree corresponding to the apparition of a new distinct subgraph. Leafs of the tree are the individual components.

Value

The `arbre.Mp` function returns a dendrogram.

Author(s)

Emmanuel Curis (<`emmanuel.curis@parisdescartes.fr`>)

See Also

`creer.Mp` to create a matrix of *p*-values for all possible ratios of a compositional vector.

`grf.Mp` to convert such a matrix to a graph, once a cut-off is selected.

`coupures.Mp` to obtain the set of *p*-values corresponding to the nodes of the tree, that is to the apparition of new sets of composants.

`plot.dendrogram` and `as.dendrogram` for more details on dendrogram drawing and structure.

Examples

```
# load the potery data set
data( poteries )

# Compute one-way ANOVA p-values for all ratios in this data set
Mp <- creer.Mp( poteries, c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
                f.p = anva1.fpc, v.X = 'Site' )

# Build the tree (in log scale, p-values are all < 0.01)
arbre <- arbre.Mp( Mp, en.log = TRUE )
```

```
# It is a dendrogram as defined in the cluster package
str( arbre )
class( arbre )

# Drawing this tree
plot( arbre )
```

BpLi — *Circadian Genes Expression in Bipolar Disorder Patients*

Description

These two datasets give the expression level of main circadian genes in lymphoblastoid cells from bipolar disorder patients, as determined by qRT-PCR. Results are expressed in cycle thresholds (CT) units.

Usage

```
data(BpLi_J2)
data(BpLi_J4)
```

Format

Each dataset is a data frame with 78 rows and 26 columns. Each row give the RNA quantification of circadian and control genes for lymphoblastoid cells of a given patient, either with or without lithium in the culture medium.

Phenotype	factor	Patient phenotype, either responding (R) or not (NR) to lithium
Patient	integer	Patient code
Li	factor	Is lithium present (Oui) or not (Non) in the culture medium of the cells

Other columns are the different circadian genes and reference genes expression levels, expressed as quantities in arbitrary units after quantitative reverse transcription PCR assays, using the Syber Green technology. Three sets of assays were done, using three different dilutions according to the explored gene: 1/20 (PER3, BHLHE41, NR1D1, DBP), 1/100 (GSK3b, RORA, PER1, PER2, CLOCK, ARNTL, CRY2, BHLHE40) and 1/200 (ARNTL2, TIMELESS, CRY1, CSNK1E). The numerical suffixe after the reference gene name (SDHA or HPRT) gives this dilution level – for instance, SDHA_20 is for the 1/20 dilution level, HRPT_100 for the 1/100 level...

Patients were classified as presenting a good response (R) or a lack of responce (NR) to lithium treatment based on the ALDA scale, see the original publication for details. Lymphoblastoid cells from each patients, obtained from blood samples, were cultivated for 2 (BpLi_J2) or 4 (BpLi_J4) days either with or without LiCl.

Source

Data courtesy allowed to be included in the package, by Cynthia Marie-Claire.

References

Geoffrey, P. A., Curis E., Courtin, C., Moreira, J., Morvillers, T., Etain, B., Laplanche, J.-L., Bellivier, F. and Marie-Claire, C. (2017). Lithium response in bipolar disorders and core clock genes expression. World J Biol Psychiatry, doi: 10.1080/15622975.2017.1282174.

`calc.rapports` *Compute all pairwise ratios of a set of variables*

Description

This function computes all pairwise ratios or differences of a set of variables in a given data frame

Usage

```
calc.rapports( d, noms, log = FALSE, isoler = FALSE )
```

Arguments

`d`	The data frame that contains the variables. Other objects will be coerced as data frames using `as.data.frame`
`noms`	A character vector containing the column names of the compositional variables to be used for ratio computations. Names absent from the data frame will be ignored with a warning. Optionnally, an integer vector containing the column numbers can be given instead. They will be converted to column names before further processing.
`log`	If `TRUE`, values in the columns are assumed to be log-transformed, and consequently ratios are computed as differences of the columns. The result is in the log scale. If `FALSE`, values are assumed to be raw data and ratios are computed directly.
`isoler`	If `TRUE`, the result data frame will not include the original values.

Details

Use this function to compute all pairwise ratio of a set of numerical variables. If non-numerical variables are given in the list of variables, they will be ignores with a warning.

Since the ratio of variables i and j is the inverse of the ratio of variables j and i, only one of them is computed. The order is determined by the order of the variables in `noms`. In matrix notations, only the upper right matrix is computed, withour the diagonal.

Value

These function returns the original data.frame with additional columns corresponding to all pairwise ratios added as the last columns of the data.frame.

These variables have their name constructed as the concatenation of the names of the two variables used, the first one being at the numerator, separated with a dot and with the additional suffix .r (or .r.log is working on difference of logarithms).

Their order is determined by the order given in noms: the first variable of the list, V1, is used to compute ratios with all others (V1/V2, V1/V3 and so on). Then the second one is used for ratios further ones (V2/V3 and so on), and so on until the last one.

Note

This function is mainly for designing a step-by-step analysis or control purposes. To avoid waste of memory, most of the functions in the package actually compute "on fly" the ratios when constructing the matrix or the data frame of p-values.

Author(s)

Emmanuel Curis (<emmanuel.curis@parisdescartes.fr>)

See Also

creer.Mp to create a matrix of p-values for all pairwise tests of ratio changes.

Examples

```
# load the potery data set
data( poteries )

# Compute all ratios in the potery data set
d.r <- calc.rapports( d = poteries, noms = c( 'Al', 'Fe', 'Mg', 'Ca',
'Na' ) )
names( d.r )
head( d.r )

identical( d.r$Al.Fe.r, d.r$Al / d.r$Fe )
```

---

choisir.seuil *Cut-off selection by simulations*

Description

Obtaining the optimal p-value cut-off for individual tests to achieve a given Type I error level of obtaining disjoint components of the graph

Usage

```
choisir.seuil( n.genes,
               taille.groupes = c( 10, 10 ),
               alpha.cible = 0.05,
               seuil.p = (5:30)/100,
               B = 3000, conf.level = 0.95,
               f.p = student.fpc, frm = R ~ Groupe,
               normaliser = FALSE, en.log = TRUE,
               n.quantifies = n.genes, masque,
               n.coeurs = 1,
               ... )
```

Arguments

`n.genes`	Number of genes to be quantified simultaneously
`taille.groupes`	An integer vector containing the sample size for each group. The number of groups is determined by the length of this vector. Unused if `masque` is provided.
`alpha.cible`	The target type I error level of obtaining disjoint subnetworks under the null hypothesis that gene expressions are the same in all groups. Should be between 0 and 1.
`seuil.p`	A numeric vector of candidate cutoffs. Values outside the [0,1] interval are automatically removed. The default (from 0.05 to 0.30) is suited for a target type I error of 0.05 and less than 30 genes, roughly.
`B`	How many simulations to do.
`conf.level`	The confidence level of the interval given as a result (see Details).
`f.p`	The function to use for individual tests of each ratio. See `creer.Mp` for details.
`frm`	The formula to use. The default is suited for the structure of the simulated data, with R the ratio and Groupe the variable with group membership.
`normaliser`	Should the simulated data by normalised, that is should their sum be equal to 1? Since ratio are insensitive to the normalisation (by contrast with individual quantities), it is a useless step for usual designs, hence the default.
`en.log`	If `TRUE`, generated data are seen as log of quantities, hence the normalisation step is performed after exponentiation of the data and data are converted back in log. The option is also used in the call of `creer.Mp`.
`n.quantifies`	The number of quantified genes amongst the `n.genes` simulated. Must be at most equal to `n.genes`, which is the default.
`masque`	A data.frame containing the values of needed covariates for all replicates. If missing, a one-column data.frame generated using `taille.groupes`, the column (named ‘Groupe’) containing values ‘G1’ repeated `taille.groupes[ 1 ]` times, ‘G2’ repeated `taille.groupes[ 2 ]` times and so on.
`n.coeurs`	The number of CPU cores to use in computation, with parallelization using forks (does not work on Windows) with the help of the parallel package.
`...`	additional arguments, to be used by the analysis function f.p

Details

The `choisir.seuil` function simulates `B` datasets of `n.genes` “quantities” measured several times, under the null hypothesis that there is only random variations between samples. For each of these `B` datasets, `creer.Mp` is called with the provided test function, then converted to a graph using in turn all cut-offs given in `seuil.p` and the number of components of the graph is determined. Having more than one is a type I error.

For each cut-off in `seuil.p`, the proportion of false-positive is then determined, along with its confidence interval (using the exact, binomial formula). The optimal cut-off to achieve the target type I error is then found by linear interpolation.

Simulation is done assuming a log-normal distribution, with a reduced, centered Gaussian on the log scale. Since under the null hypothesis nothing changes between the groups, the only needed informations is the total number of values for a given gene, which is determined from the number of rows of `masque`. All columns of `masque` are transfered to the analysis function, so simulation under virtually any experimental design should be possible, as far as a complete null hypothesis is wanted (not any effect of any covariate).

Value

`choisir.seuil` returns a data.frame with four columns, corresponding to the candidate cut-offs, the corresponding estimated type-I error and its lower and upper confidence bounds, and attributes giving the estimated optimal cut-off, its confidence interval and details on simulation condition. This data.frame has the additional class `SARPcompo.H0`, allowing specific `print` and `plot` methods to be used.

Warning

The simulated ratios are stored in a column called R, appended to the simulated data.frame. For this reason, do not use any column of this name in the provided masque: it would be overwritten during the simulation process.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

`creer.Mp`.

Examples

```
# What would be the optimal cut-off for 10 genes quantified in two
#  groups of 5 replicates?
# For speed reason, only 50 simulations are done here,
#  but obviously much more are needed to have a good estimate f the cut-off.

seuil <- choisir.seuil( 10, c( 5, 5 ), B = 50 )
seuil

# Get the cut-off and its confidence interval
```

```
attr( seuil, "seuil" )

# Plot the results
plot( seuil )
```

`choisir.seuil.equiv` *Cut-off selection by simulations, in the context of equivalence tests*

Description

Obtaining the optimal p-value cut-off for individual tests to achieve a given Type I error level of obtaining connected nodes in the graph

Usage

```
choisir.seuil.equiv( n.genes, taille.groupes,
                     mu = 10, sigma = 0.5, Delta = 0.5,
                     alpha.cible = 0.05,
                     seuil.p = (10:40)/100,
                     B = 3000, conf.level = 0.95,
                     f.p = equiv.fpc,
                     en.log = TRUE,
                     n.coeurs = 1,
                     ... )
```

Arguments

`n.genes`	Number of genes to be quantified simultaneously
`taille.groupes`	An integer vector containing the sample size for each group. The number of groups is determined by the length of this vector. Unused if `masque` is provided.
`mu`	A numeric vector giving the mean amount for each component in the first condition, in the log scale (μ\). If a single value is provided, it is used for each component. Otherwise, the length of the vector must be equal to the number of components. It can also be a two-lines matrix giving the mean amounts for each component (columns) in the first (firt row) and second (second row) condition.
`sigma`	A numeric vector giving the standard deviation for the amount of each component in both conditions, in the log scale (σ\). If a single value is provided, it is used for each component. Otherwise, the length of the vector must be equal to the number of components. It can also be a two-lines matrix giving the mean amounts for each component (columns) in the first (firt row) and second (second row) condition.
`Delta`	The limit for the equivalence region, Δ, in the log scale.
`alpha.cible`	The target type I error level of obtaining disjoint subnetworks under the null hypothesis that gene expressions are the same in all groups. Should be between 0 and 1.

`seuil.p`	A numeric vector of candidate cutoffs. Values outside the [0,1] interval are automatically removed. The default (from 0.05 to 0.30) is suited for a target type I error of 0.05 and less than 30 genes, roughly.
`B`	How many simulations to do.
`conf.level`	The confidence level of the interval given as a result (see Details).
`f.p`	The function to use for individual tests of each ratio. See `creer.Mp` for details.
`en.log`	If `TRUE`, generated data are seen as log of quantities. The option is used in the call of `creer.Mp`.
`n.coeurs`	The number of CPU cores to use in computation, with parallelization using forks (does not work on Windows) with the help of the parallel package.
`...`	additional arguments, to be used by the analysis function f.p

Details

The `choisir.seuil.equiv` function simulates `B` datasets of `n.genes` "quantities" measured several times, under the null hypothesis that variations between samples of two conditions are given by the difference between the two rows of the μ matrix. If μ was given as a single row (or a single value), the second row is defined as $(\mu, \mu + \Delta, \mu + 2\Delta \ldots)$ – correspondong to the null hypothesis that all components have a different change between the two conditions, and that this change is equal to the equivalence region limit (Δ). For each of these `B` datasets, `creer.Mp` is called with the provided test function, then converted to a graph using in turn all cut-offs given in `seuil.p` and the number of edges of the graph is determined. Having at least one edge is a type I error, since under the null hypothesis there is no couple of genes having the same change.

For each cut-off in `seuil.p`, the proportion of false-positive is then determined, along with its confidence interval (using the exact, binomial formula). The optimal cut-off to achieve the target type I error is then found by linear interpolation.

Data are generated using a normal (Gaussian) distribution, independantly for each component and each condition.

Value

`choisir.seuil.equiv` returns a data.frame with four columns, corresponding to the candidate cut-offs, the corresponding estimated type-I error and its lower and upper confidence bounds, and attributes giving the estimated optimal cut-off, its confidence interval and details on simulation condition. This data.frame has the additional class `SARPcompo.H0`, allowing specific `print` and `plot` methods to be used.

Author(s)

Emmanuel Curis (<`emmanuel.curis@parisdescartes.fr`>)

See Also

`creer.Mp`, `equiv.fpc`.

See `choisir.seuil` for the case of difference tests and disjoing subgraphs.

Examples

```
# What would be the optimal cut-off for 5 genes quantified in two
#  groups of 5 replicates?
# Null hypothesis : mean = 0, sd = 1, Delta = 2
# For speed reason, only 50 simulations are done here,
#  but obviously much more are needed to have a good estimate f the cut-off.

seuil <- choisir.seuil.equiv( 5, c( 5, 5 ),
                              mu = 1, sigma = 1, Delta = 1,
                              B = 50 )
seuil

# Get the cut-off and its confidence interval
attr( seuil, "seuil" )

# Plot the results
plot( seuil )
```

`conversions` *Convert between matrix and data-frame format*

Description

These functions convert all pairwise ratio tests between the matrix format and the data.frame format

Usage

```
Mp.DFp(DFp, col.noms = c( 1, 2 ), col.p = 'p')
```

Arguments

`DFp`	Results in the data.frame format.
`col.noms`	A length two character vector giving the columns in the data.frame format that contain the names of the two components of which the ratio is tested. If converting from a data.frame, can be given by number.
`col.p`	A length one character vector giving the column containing the p-values of the test in the data.frame format. If converting from a data.frame, can be given by number.

Details

The matrix format is more convenient for manipulations like finding cut-off probabilities or building the hierarchical tree of the components. However, it can only store one value per couple and is more memory consuming.

The data.frame format is more efficient for computations, since it allows to store several results at once. It can also be easily saved and read using text-files and `read.table` or `write.table`. However, finding the hierarchical tree of components or building the graph is not so straightforward.

These utilitary functions allow to convert between the two formats.

Value

The results in the other format: a matrix for `Mp.DFp` and a 3-columns data.frame for `DFp.Mp`.

In the matrix form, components will be sorted by alphabetical order.

Warning

When converting a data.frame to a matrix, there is no control that all possible combinations are present once, and only once, in the data.frame. Missing combinations will have 0 in the matrix; combinations present several time will have the value of the last replicate.

When converting a matrix to a data.frame, the diagonal is not included in the data.frame. The matrix is expected to be symmetric, and only the upper right part is used.

Author(s)

Emmanuel Curis (<emmanuel.curis@parisdescartes.fr>)

See Also

`creer.Mp` to create a matrix of *p*-values for all possible ratios of a compositional vector.

`creer.DFp` to create a data.frame of *p*-values for all possible ratios of a compositional vector.

Examples

```
  # load the potery data set data( poteries )
```

`coupures.Mp` *Finding cut-offs for graph disjonctions*

Description

These functions detect the experimental cut-offs to create distinct subgraphs, and propose adapted graphical representation.

Usage

```
coupures.Mp( Mp )

## S3 method for class 'Coupures'
plot(x, seuil.p = 0.05, en.log = TRUE,
                        xlab = "Seuil de p", ylab = "Nombre de composantes",
                        col.trait = "black", lwd.trait = 1, lty.trait = 1,
                        col.seuil = "red"  , lwd.seuil = 1, lty.seuil = 1,
                        pch.fin = 19, cex.fin = 1, col.fin ="darkgreen",
                        pch.deb = ")", cex.deb = 1, col.deb = "darkgreen",
                        ...)
```

Arguments

`Mp`	A square, symmetric matrix containing *p*-values. Element in row i and line j should contain the *p*-value for testing the $\frac{i}{j}$ ratio. The diagonal is ignored.
`x`	The set of critical values, as obtained by `coupures.Mp`
`seuil.p`	Selected cut-off for analysis. Can also be a `SARPcompo.H0` object, as returned by `choisir.seuil`, in which case the bounds of the confidence interval are also drawn, with dashed lines by default.
`en.log`	If `TRUE`, the *p*-values axis uses a decimal logarithm scale. It may help visualisation of the small critical *p*-values.
`xlab,ylab`	Legends for the axes
`col.trait, lwd.trait, lty.trait, pch.fin, cex.fin, col.fin, pch.deb, cex.deb, col.deb`	Graphical parameters for drawing the number of components in function of the cut-off. 'trait' refers to the function itself, 'deb' to the first point of a region of constant components number (that does not belong to it: the function is right-discontinuous) and 'fin' to the last point of this region (that belongs to it)
`col.seuil, lwd.seuil, lty.seuil`	Graphical parameters for drawing the analysis cut-off
`...`	Additionnal parameters for `plot`, which is used internally.

Details

By increasing the cut-off from 0 to 1, more and more edges between nodes are removed, and disjoint subgraphs appear. This function detects in a matrix of *p*-values which are the "critical" ones, that is the one for which the number of components changes.

Because the edge removal is defined by $p < cut - off$, the cut-off returned for a given number of components is to be understand as the maximal one that gives this number of components.

The `plot` method allows to visualize the evolution of the number of components with the cut-off, and writes the critical cut-off values.

Value

The `coupures.Mp` function returns a data.frame with additionnal class 'Coupures'. It contains three columns: one with the *p*-value cut-offs, one with the opposite of their decimal logarithm and one with the number of components when using exactly this cut-off. The additionnal class allows to provide a `plot` method.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

`creer.Mp` to create a matrix of *p*-values for all possible ratios of a compositional vector.

`grf.Mp` to convert such a matrix to a graph, once a cut-off is selected.

`arbre.Mp` to convert such a matrix to a classification tree of the components of the compositional vector.

Examples

```
# load the potery data set
data( poteries )

# Compute one-way ANOVA p-values for all ratios in this data set
Mp <- creer.Mp( poteries, c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
                f.p = anva1.fpc, v.X = 'Site' )

# Where would be the cut-offs?
seuils <- coupures.Mp( Mp )
seuils

# Drawing this, in log10 scale
plot( seuils, en.log = TRUE )
```

creer_data.frame — *Create p-values data-frame from pairwise tests of all possible ratios of a compositional vector*

Description

This function performs hypothesis testing on all possible pairwise ratios or differences of a set of variables in a given data frame, and store their results in a data.frame

Usage

```
creer.DFp( d, noms, f.p = student.fpc,
           log = FALSE, en.log = !log,
           nom.var = 'R',
           noms.colonnes = c( "Cmp.1", "Cmp.2", "p" ),
           add.col = "delta", n.coeurs = 1,
           ... )
```

Arguments

`d`	The data frame that contains the compositional variables. Other objects will be coerced as data frames using `as.data.frame`
`noms`	A character vector containing the column names of the compositional variables to be used for ratio computations. Names absent from the data frame will be ignored with a warning. Optionnally, an integer vector containing the column numbers can be given instead. They will be converted to column names before further processing.
`f.p`	An R function that will perform the hypothesis test on a single ratio (or log ratio, depending on `log` and `en.log` values). This function should return a numeric vector, of which the first one will typically be the p-value from the test — see `creer.Mp` for details. Such functions are provided for several common situations, see references at the end of this manual page.

`log`	If `TRUE`, values in the columns are assumed to be log-transformed, and consequently ratios are computed as differences of the columns. The result is in the log scale. If `FALSE`, values are assumed to be raw data and ratios are computed directly.
`en.log`	If `TRUE`, the ratio will be log-transformed before applying the hypothesis test computed by `f.p`. Don't change the default unless you really know what you are doing.
`nom.var`	A length-one character vector giving the name of the variable containing a single ratio (or log-ratio). No sanity check is performed on it: if you experience strange behaviour, check you gave a valid column name, for instance using `make.names`.
`noms.colonnes`	A length-three character vector giving the names of, respectively, the two columns of the data frame that will contain the components identifiers and of the column that will contain the p-value from the test (the first value returned by `f.p`).
`add.col`	A character vector giving the names of additional columns of the data.frame, used for storing additional return values of `f.p` (all but the first one).
`n.coeurs`	The number of CPU cores to use in computation, with parallelization using forks (does not work on Windows) with the help of the parallel package.
`...`	additional arguments to `f.p`, passed unchanged to it.

Details

This function constructs a data.frame with $n \times (n-1)/2$ rows, where `n = length( noms )` (after eventually removing names in `noms` that do not correspond to numeric variables). Each line of the data.frame is the result of the `f.p` function when applied on the ratio of variables whose names are given in the first two columns (or on its log, if either `(log == TRUE) && (en.log == FALSE)` or `(log == FALSE) && (en.log == TRUE)`).

Value

These function returns the data.frame obtained as described above.

Note

Creating a data.frame seems slightly less efficient (in terms of speed) than creating a dense matrix, so for compositionnal data with only a few components and simple stastitical analysis were only a single p-value is needed, consider using `creer.Mp` instead.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

Predefined `f.p` functions: `anova1.fpc` for one-way analysis of variance; `kw.fpc` for the non-parametric equivalent (Kruskal-Wallis test).

`grf.DFp to create a graphe from the obtained matrix.`

Examples

```
# load the Circadian Genes Expression dataset, at day 4
data( "BpLi_J4" )
ng <- names( BpLi_J4 )[ -c( 1:3 ) ] # Name of the genes

# analysis function (complex design)
#   1. the formula to be used
frm <- R ~  (1 | Patient) + Phenotype + Li + Phenotype:Li

#   2. the function itself
#    needs the lme4 package
if ( TRUE == require( "lme4" ) ) {
  f.p <- function( d, variable, ... ) {
       # Fit the model
       md <- lmer( frm, data = d )

       # Get coefficients and standard errors
       cf <- fixef( md )
       se <- sqrt( diag( vcov( md ) ) )

       # Wald tests on these coefficients
       p <- 2 * pnorm( -abs( cf ) / se )

       # Sending back the 4 p-values
       p
  }

  # CRAN does not like 'long' computations
  # => analyse only the first 6 genes
  #  (remove for real exemple!)
  ng <- ng[ 1:6 ]

  # Create the data.frame with all results
  DF.p <- creer.DFp( d = BpLi_J4, noms = ng,
                     f.p = f.p, add.col = c( 'p.NR', 'p.Li', 'p.I' ) )

  # Make a graphe from it and plot it
  #  for the interaction term, at the p = 0.2 threshold
  plot( grf.DFp( DF.p, p = 0.20, col.p = 'p.I' ) )
}
```

`creer_graphe` *Create a graph using a set of p-values from pairwise tests*

Description

These functions construct an undirected graph, using the igraph package, to represent and investigate the results of all testing of all ratios of components of a compositional vector.

Usage

```
grf.Mp( Mp, p = 0.05, reference = NULL, groupes = NULL,
        complement = FALSE )

grf.DFp( DFp, col.noms = c( 1, 2 ), p = 0.05, col.p = 'p',
         reference = NULL, groupes = NULL )
```

Arguments

`Mp`	A square, symmetric matrix containing *p*-values. Element in row i and line j should contain the *p*-value for testing the $\frac{i}{j}$ ratio. The diagonal is ignored.
`DFp`	A data frame containing at least three columns: two with component names and one with *p*-values for the corresponding ratio.
`col.noms`	A length two vector giving the names of the two columns containing the components names.
`p`	The *p*-value cutoff for adding an edge between two nodes. See details.
`col.p`	The name of the column containing the *p*-values to use to create the graph.
`reference`	A character vector giving the names of nodes that should be displayed with a different color in the created graph. These names should match column names used in Mp. Typical use would be for reference genes in qRT-PCR experiments. By default, all nodes are displayed in palegreen; reference nodes, if any, will be displayed in orange.
`groupes`	
`complement`	A logical. If TRUE, the complement of the graph is returned. It allows to construct graphs using equivalence tests, instead of difference tests (that is, to keep an edge between two nodes if the test is significant, instead of non-significant).

Details

Consider a compositional vector of n components. These n are seen as the nodes of a graph. Nodes i and j will be connected if and only if the *p*-value for the test of the $\frac{i}{j}$ ratio is higher than the cutoff, `p` – that is, if the test is **not** significant at the level p.

Strongly connected sets of nodes will represent components that share a similar behaviour between the conditions tested, whereas unrelated sets of nodes will have a different behaviour.

Value

These function returns the created graph. It is an igraph object on which any igraph function can be applied, including plotting, and searching for graph components, cliques or communities.

Author(s)

Emmanuel Curis (<`emmanuel.curis@parisdescartes.fr`>)

See Also

creer.Mp to create a matrix of *p*-values for all possible ratios of a compositional vector.

creer.DFp to create a data.frame of *p*-values for all possible ratios of a compositional vector.

Examples

```
# load the potery data set
data( poteries )

# Compute one-way ANOVA p-values for all ratios in this data set
Mp <- creer.Mp( poteries, c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
                f.p = anova1.fpc, v.X = 'Site' )
Mp

# Make a graphe from it and plot it
plot( grf.Mp( Mp ) )
```

---

creer_matrice *Create p-values matrix from pairwise tests of all possible ratios of a compositional vector*

Description

This function performs hypothesis testing on all possible pairwise ratios or differences of a set of variables in a given data frame, and store their results in a (symmetric) matrix

Usage

```
creer.Mp( d, noms, f.p, log = FALSE, en.log = !log,
          nom.var = 'R', n.coeurs = 1, ... )
```

Arguments

d	The data frame that contains the compositional variables. Other objects will be coerced as data frames using as.data.frame
noms	A character vector containing the column names of the compositional variables to be used for ratio computations. Names absent from the data frame will be ignored with a warning. Optionnally, an integer vector containing the column numbers can be given instead. They will be converted to column names before further processing.
f.p	An R function that will perform the hypothesis test on a single ratio (or log ratio, depending on log and en.log values). This function should return a single numerical value, typically the p-value from the test. This function must accept at least two named arguments: d that will contain the data frame containing all required variables and variable that will contain the

	name of the column that contains the (log) ratio in this data frame. All other needed arguments can be passed through `...`. Such functions are provided for several common situations, see references at the end of this manual page.
`log`	If `TRUE`, values in the columns are assumed to be log-transformed, and consequently ratios are computed as differences of the columns. The result is in the log scale. If `FALSE`, values are assumed to be raw data and ratios are computed directly.
`en.log`	If `TRUE`, the ratio will be log-transformed before applying the hypothesis test computed by `f.p`. Don't change the default unless you really know what you are doing.
`nom.var`	A length-one character vector giving the name of the variable containing a single ratio (or log-ratio). No sanity check is performed on it: if you experience strange behaviour, check you gave a valid column name, for instance using `make.names`.
`n.coeurs`	The number of CPU cores to use in computation, with parallelization using forks (does not work on Windows) with the help of the parallel package.
`...`	additional arguments to `f.p`, passed unchanged to it.

Details

This function constructs a $n \times n$ matrix, where `n = length( noms )` (after eventually removing names in `noms` that do not correspond to numeric variables). Term (i, j) in this matrix is the result of the `f.p` function when applied on the ratio of variables `noms[ i ]` and `noms[ j ]` (or on its log, if either `(log == TRUE) && (en.log == FALSE)` or `(log == FALSE) && (en.log == TRUE)`).

The `f.p` function is always called only once, for $i < j$, and the other term is obtained by symmetry.

The diagonal of the matrix is filled with 1 without calling `f.p`, since corresponding ratios are always identically equal to 1 so nothing useful can be tested on.

Value

These function returns the matrix obtained as described above, with row an column names set to the names in `noms` (after conversion into column names and removing all non-numeric variables).

Note

Since the whole matrix is stored and since it is a dense matrix, memory consumption (and computation time) increases as n^2. For compositional data with a large number of components, like in RNA-Seq data, consider instead creating a file.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

Predefined `f.p` functions: `anova1.fpc` for one-way analysis of variance; `kw.fpc` for the non-parametric equivalent (Kruskal-Wallis test).

`grf.Mp to create a graphe from the obtained matrix.`

Examples

```
# load the potery data set
data( poteries )

# Compute one-way ANOVA p-values for all ratios in this data set
Mp <- creer.Mp( poteries, c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
                f.p = anva1.fpc, v.X = 'Site' )
Mp

# Make a graphe from it and plot it
plot( grf.Mp( Mp ) )
```

`distances` *Simulate the distribution of maximal minimal distances in a random graph*

Description

This function simulates the distribution of the maximum of the minimal distances between nodes of a random graph, for a given cut-off threshold.

Usage

```
distrib.distances( n.genes,
                   taille.groupes = c( 10, 10 ), masque,
                   me.composition = 0, cv.composition = 1, en.log = TRUE,
                   seuil.p = 0.05,
                   B = 3000, conf.level = 0.95,
                   f.p = student.fpc, frm = R ~ Groupe,
                   n.coeurs = 1 )
```

Arguments

`conf.level`	The confidence level for the exact confidence intervals of estimated probabilities of maximal minimal distances in the graph.
`n.genes`	Number of components in the system (of nodes in the total graph). Ignored if `me.composition` is a matrix.
`me.composition`	The expected median quantity of each component, in the log scale. Can be either a single value, used for two conditions and `n.genes` components (hence, assuming the null hypothesis that no change occurs), or a matrix with one row by experimental condition and one column by component.
`cv.composition`	The expected coefficient of variation of the quantified amounts. Should be either a single value, that will be used for all components and all conditions, or a matrix with the same structure than `me.composition`: one row for each condition, one column for each component, in the same order and with the same names. Coefficients of variations are expected in the amount scale, in raw form (that is, give 0.2 for a 20% coefficient of variation).

`en.log`	If `TRUE`, the values in the matrices are given in the log scale.
`taille.groupes`	The sample size for each condition. Unused if `masque` is given. If a single value, it will be used for all conditions. Otherwise, should have the same length that the number of rows in the provided matrices.
`masque`	A data.frame that will give the dataset design for a given experiment. Should contain at least one column containing the names of the conditions, with values being in the conditions names in `composition`. If not provided, it is generated from `taille.groupes` as a single column named 'Condition'.
`f.p, frm`	The function used to analyse the dataset, and its parameter. See `creer.Mp` for details.
`seuil.p`	The p-value cut-off to be used when creating the graph. Should be between 0 and 1. See `grf.Mp` for details.
`B`	The number of simulations to be done.
`n.coeurs`	The number of CPU cores to use to parallelize the simulation.

Details

In an undirected graph, minimal distance between two nodes is the minimal number of edges to cross to go from one node to the other. The maximal minimal distance is the largest of all possible minimal distances in a given graph.

The function simulates the distribution of the maximal minimal distance in a graph whose edges were removed according to the specified p-value cut-off. To avoid infinite distances, these distances are computed in the largest connected component of the graph.

In the observed graph, nodes that are at a largest minimal distance than probable maximal minimal distances may signal components belonging to different sets, that could not be disconnected because of some nodes having intermediate changes.

Value

A 4-columns data.frame, with additional attributes giving the number of simulations (`Nombre.simulations`) and their results (`Tirages`). The first column contains the maximal minimal distances, the second contains their observed frequencies in the simulated datasets, the third and fourth contain the limits of the confidence interval of the corresponding probability.

Confidence intervals are exacts, using the Clopper-Pearson method.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

`distances`, in package igraph, to compute the matrix of all minimal distances of a graphe.

`creer.Mp` and `grf.Mp`, which are used internally, for details about analysis functions and p-value cutoff.

`equivalence` *Utility function to obtain p-value for equivalence tests on individual ratios*

Description

These functions can be used in the functions to perform analysis on all pairwise ratios of a compositional dataset, using equivalence tests to ensure edge existence

Usage

```
equiv.fpc( d, variable, v.X, var.equal = TRUE, Delta = 0.5,
           pred = FALSE, ... )
```

Arguments

`d`	The data frame that contains the ratio to test, and all variables of the original data frame that where not used as compositional data.
`variable`	A length-one character vector containing the names of the variable corresponding to the ratio (or log-ratio) to test.
`v.X`	The **name** of the explanatory (independant, predictor) variable. This variable should be a factor for `equiv.fpc`.
`var.equal`	For `equiv.fpc`, shall we assume that variance are equals in the two groups (`TRUE`, the default) or not (`FALSE`). Same as in `t.test`.
`Delta`	The value giving the positive limit of the equivalence region. For symmetry reasons, the equivalence region will be [-Delta, Delta].
`pred`	If `FALSE`, a standard equivalence test of the mean difference is done. If `TRUE`, the p-value is computed assuming the variance of the difference, and not the difference of the means. This allows to take into account the sampling variability in the interval width, avoiding (with large sample sizes) too narrow intervals to fit in the equivalence region when large inter-sample variability does not allow to consider genes as reliable reference genes.
`...`	additional arguments

Details

These functions are only wrapper to some commonly used equivalence tests.

The basic idea underlying equivalence tests is to try to reject the null hypothesis that the difference between the two conditions is higher (in absolute value) than a predefined, fixed, value, given by `Delta`. Consequently, significant tests will mean that the edge between the two tested nodes should be kept, whereas non-significant tests will mean that the existence of the edge is uncertain.

As a consequence, conversion of the p-value matrix to graphs should be made using the complement of the graph, and the analysis of the graph should be made in terms of cliques instead of disjoint subgraphs. See example for an illustration.

Value

These function returns the *p*-value from the corresponding test.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

`student.fpc` for a, more usual, approach using difference tests.

`creer.Mp` to use these functions to create a matrix of *p*-values.

Examples

```
# load the reference genes data set
data( generef )

# compute equivalence test of all ratios
#  Genes are in column 3 to 35
#  Only the first 10 genes are used here, for speed reasons
#  Equivalence is defined as a change lower than 0.5 Cq (× 1.41)
#  Values are expressed as Cq : already in log
Mp <- creer.Mp( generef, names( generef )[ 3:13 ], log = TRUE,
                f.p = equiv.fpc, Delta = 0.5, v.X = 'Group' )

# Make the graph from it, and plot it
#  Threshold is set at 0.15, for 10 nodes...
#  (using the complement, to keep edges with p < threshold only )
plot( grf.Mp( Mp, p = 0.15, complement = TRUE ) )
#  => there is a single clique, of 3 genes : HPRT1, B2M, HSP90AB1
#     only these three genes are "proved" to have the same variation
```

`fichiers` *Create and read a file of p-values for all pairwise tests of all possible ratios of a compositional vector*

Description

These functions allow to perform hypothesis testing on all possible pairwise ratios or differences of a set of variables in a given data frame, and store or read their results in a file

Usage

```
creer.Fp( d, nom.fichier,
          noms, f.p = student.fpc,
          log = FALSE, en.log = !log,
          nom.var = 'R',
          noms.colonnes = c( "Cmp.1", "Cmp.2", "p" ),
          add.col = "delta",
          sep = ";", dec = ".", row.names = FALSE, col.names = TRUE,
          ... )

grf.Fp( nom.fichier, col.noms = c( 1, 2 ), p = 0.05, col.p = 'p',
        reference = NULL, groupes = NULL,
        sep = ";", dec = ".", header = TRUE,
        ... )
```

Arguments

`d`	The data frame that contains the compositional variables. Other objects will be coerced as data frames using `as.data.frame`
`nom.fichier`	A length-one character vector giving the name of the file
`noms`	A character vector containing the column names of the compositional variables to be used for ratio computations. Names absent from the data frame will be ignored with a warning. Optionnally, an integer vector containing the column numbers can be given instead. They will be converted to column names before further processing.
`f.p`	An R function that will perform the hypothesis test on a single ratio (or log ratio, depending on `log` and `en.log` values). This function should return a numeric vector, of which the first one will typically be the p-value from the test — see `creer.Mp` for details. Such functions are provided for several common situations, see links at the end of this manual page.
`log`	If `TRUE`, values in the columns are assumed to be log-transformed, and consequently ratios are computed as differences of the columns. The result is in the log scale. If `FALSE`, values are assumed to be raw data and ratios are computed directly.
`en.log`	If `TRUE`, the ratio will be log-transformed before applying the hypothesis test computed by `f.p`. Don't change the default unless you really know what you are doing.
`nom.var`	A length-one character vector giving the name of the variable containing a single ratio (or log-ratio). No sanity check is performed on it: if you experience strange behaviour, check you gave a valid column name, for instance using `make.names`.
`noms.colonnes`	A length-three character vector giving the names of, respectively, the two columns of the data frame that will contain the components identifiers and of the column that will contain the p-value from the test (the first value returned by `f.p`).
`add.col`	A character vector giving the names of additional columns of the data.frame, used for storing additional return values of `f.p` (all but the first one).
`sep, dec, row.names, col.names, header`	Options for controling the file format, used by `write.table` and `read.table`.
`col.noms`	A length-two vector giving the two columns that contain the two components of the ratio. Can be given either as column number or column name.

`col.p`	A length-one vector giving the column that contain the *p*-value of the ratio. Can be given either as column number or column name.
`p`	The *p*-value cut-off to be used when creating the graph, see `grf.Mp` for details.
`reference`	A character vector giving the names of nodes that should be displayed with a different color in the created graph. These names should match components names present un the file. Typical use would be for reference genes in qRT-PCR experiments. By default, all nodes are displayed in palegreen; reference nodes, if any, will be displayed in orange.
`groupes`	
`...`	additional arguments to `f.p`, passed unchanged to it.

Details

These functions are basically the same as the function that create data.frames (`creer.DFp`) and use data.frames to create a graph (`grf.DFp`), except thatthey work on text files. This allow to deal with compositionnal data including thousands of components, like RNA-Seq or microarray data.

Seeing the results as a matrix, computations are done in rows and the file is updated after each row. Only the upper-triangular part, without the diagonal, is stored in the file.

The function that creates the graphe from file is not very efficient and can take a lot of time for huge matrices. Making a first filter on the file using shell tools, like `gawk` or `perl`, or a dedicated C software and loading the resulting file as a data.frame before converting it into a graph is a better alternative, but may lose some isolated nodes.

Value

`creer.Fp` does not return anything. `grf.Fp` returns the result graph.

Note

Creating a file and working from a file is quite inefficient (in terms of speed), so for compositionnal data with only a few components, consider using `creer.DFp` that creates the data.frame directly in memory and `grf.DFp` that creates the graphe from a data.frame instead.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

Predefined `f.p` functions: `anva1.fpc` for one-way analysis of variance; `kw.fpc` for the non-parametric equivalent (Kruskal-Wallis test).

For directly creating and manipulating matrices, `creer.Mp` and `grf.Mp`.

Examples

```
# load the potery data set
data( poteries )

# Create the file name in R temporary directory
nom.fichier <- paste0( tempdir(), "/fichier_test.csv" )
nom.fichier

# Compute one-way ANOVA p-values for all ratios in this data set
#  and store them in a text file
creer.Fp( poteries, nom.fichier,
          c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
          f.p = anva1.fpc, v.X = 'Site',
          add.col = c( 'mu0', 'd.C', 'd.CoA', 'd.IT', 'd.L' ) )

# Make a graphe from it and plot it
plot( grf.Fp( nom.fichier ) )

# The file is a simple text-file that can be read as a data.frame
DFp <- read.table( nom.fichier, header = TRUE, sep = ";", dec = "," )
DFp
```

GADL1 *Gene expression change in bipolar disorder*

Description

This data set gives the mRNA quantification of a few genes compared between bipolar disorder patients and healthy volunteers.

Usage

`data(GADL1)`

Format

A data frame with 13 columns and 56 rows. Each row gives the quantification results, expressed in amounts:

Groupe	factor	The group label, Ctrl for healthy volunteers and BPD for bipolar disorder patients
Phenotype	factor	the subgroup label, C for healthy volunteers, NR for patients not responding to treatment and R for patie
Sample	factor	the sample unique identifier

All other columns are numeric and give the amount, in arbitrary unit, of mRNA of the corresponding gene. The column name gives the name of the gene and the dilution used for the quantification. *hprt* and *sdha* are used as reference genes.

Source

Original data communicated by the authors of the experiment.

Examples

```
data( GADL1 )
# Optimal cut-off for ten genes and alpha = 0.05
#  is around 0.22

# First step, is there differences between healthy subjects
#   and patients ?
#
M.m <- creer.Mp( d = GADL1, noms = names( GADL1 )[ -c( 1:3 ) ],
                 f.p = student.fpc, v.X = 'Groupe' )

#  2) L'arbre associé
#  [reference gene for plotting purpose]
n.ref <- grep( 'HPRT|SDHA', names( GADL1 ), value = TRUE )
plot( arbre.Mp( M.m, reference = n.ref ),
      seuil.p = c( 0.218, 0.207, 0.230 ) )

#  3) Le graphe pour le seuil optimal
#    => only IGF1 seems to behave differently
#       (but it has missing values, so interpretation is difficult)
plot( grf.Mp( M.m, reference = n.ref, p = 0.22 ) )


# Second step, is there differences between patients
#   that respond or not respond to treatment?
d.R <- GADL1[ which( GADL1$Groupe == 'BPD' ), ]
M.R <- creer.Mp( d = d.R, noms = names( GADL1 )[ -c( 1:3 ) ],
                 f.p = student.fpc, v.X = 'Phenotype' )

#  2) L'arbre associé
plot( arbre.Mp( M.R, reference = n.ref ),
      seuil.p = c( 0.218, 0.207, 0.230 ) )

#  3) Le graphe pour le seuil optimal
#    => no sign of any difference
plot( grf.Mp( M.R, reference = n.ref, p = 0.22 ) )
```

`generef` *Expression level of candidate reference genes*

Description

This dataset gives the expression level of 30 different candidate reference genes, in control subjects and in subjects with bipolar disorder.

Usage

```
data(generef)
```

Format

A data frame with 40 rows and 32 columns. Each row gives the results and characteristics of a given patient. Columns 3 to 32 give the average expression level of the gene whose name is the column name, for the given patient. Columns are sorted in alphabetical order of the gene name. Values are expressed as Cq (Ct) and were obtained through qRT-PCR. RNAs were extracted on lymphoblastoid cell line culture. Cq are the average of three technical replicates, after eventual removal of outliers.

ID	factor	Patient identifier
Group	factor	Patient group (control or having bipolar disorders)
ATCB	numeric	Cq for gene ATCB
...	numeric	Cq for gene ...
YWHAZ	numeric	Cq for gene YWHAZ

Source

Experimental data kindly provided by Calypso Nepost \& Cynthia Marie-Claire, UMR-S 1144, INSERM-Paris Descartes-Paris Diderot

`modele` *Create a compositional model for simulations*

Description

These functions create and plot a model of compositionnal data for two or more conditions.

Usage

```
modele_compo( medianes, en.log = FALSE,
              noms = colnames( medianes ),
              conditions = rownames( medianes ),
              reference = NULL, total = 1 )

## S3 method for class 'SARPcompo.modele'
plot( x,
     xlab = "Composant",
     ylab.absolu = "Quantit\u00e9", ylab.relatif = "Fraction",
     taille.noeud = 50, ... )
```

Arguments

`medianes`	A matrix giving the medians of all components quantities in each condition. Each row of this matrix corresponds to a different condition; each column, to one of the components.
`en.log`	If `TRUE`, the values in the matrix are given in the log scale.
`noms`	Names of the components. If provided, should be a character vector whose length is equal to the number of columns of `medianes`.
`conditions`	Names of the different conditions. If provided, should be a character vector whose length is equal to the number of rows of `medianes`.
`reference`	A character vector giving the names of the components used as reference (typically, reference genes in qRT-PCR).
`total`	The total amount. The sum of amounts in each condition will equal this total, when the data are made compositionnal.
	Arguments for the plot method
`x`	The modele to be plotted
`xlab`	Legend for the X axis
`ylab.absolu`	Legend for the Y axis, in the amount scale (no constrain)
`ylab.relatif`	Legend for the Y axis, in the compositional scale.
`taille.noeud`	The plot size of nodes of the theoretical graph
`...`	Additionnal parameters for `plot`, which is used internally.

Details

The `modele_compo` function creates a compositionnal data model using the quantites provided: it converts amounts in fractions of the total amount for each condition, then computes the theoretical graph showing classes of equivalents components, that is components that have the same evolution between the two conditions. If more than two conditions are given, graphs correspond to comparison of each condition with the first one.

The `plot` methods represents the original quantities, the quantities after conversion in compositional data ant the theoretical graph.

Value

An object of class `SARPcompo.modele`, with a plot method. It is a list with the following elements:

`Absolue`	The matrix of quantities in amount scale
`Relative`	The matrix of quantities in compositional data scale
`Graphes`	A list of length `nrow(medianes) -1`. Each element of the list gives, for the corresponding condition, the matrix of all ratios of pairwise ratios between condition and the first condition (element `M.rapports`), the corresponding connectivity matrix (element `M.connexion`), the graph of component changes compared to the first condition (element `Graphe`, an igraph object) and the list of components of this graph (element `Connexe`, obtained from the `components` function.

It also stores a few informations as attributes.

Author(s)

Emmanuel Curis (<`emmanuel.curis@parisdescartes.fr`>)

See Also

`estimer.puissance` and `estimer.alpha` to use these models in simulations to study power and type I error of the method in a given situation.

Examples

```
## Create a toy example: four components, two conditions
##  components 1 and 2 do not change between conditions
##  component  3 is doubled
##  component  4 is halfed
me <- rbind( 'A' = c( 1, 1, 1, 1 ),
             'B' = c( 1, 1, 2, 0.5 ) )
colnames( me ) <- paste0( "C-", 1:4 )

md <- modele_compo( me )

## Plot it...
plot( md )

## What is approximately the power to detect that something changes
## between conditions A and B using a Student test
## with a CV of around 50 % ?
##  (only a few simulations for speed, should be increased )
puissance <- estimer.puissance( md, cv = 0.50, B = 50, f.p = student.fpc )
plot( puissance )
```

`poteries` *Composition of Roman poteries*

Description

This data set gives the oxide composition of several potteries found in five different archaelogic sites of the United Kingdom. Composition was obtained using atomic absorption spectrometry.

Usage

```
data(poteries)
```

Format

A data frame with 14 columns and 48 rows. Each row gives the composition of a pottery (columns 2 to 10), the archaelogical site where it was found (columns 6 and 7):

ID	factor	Pottery sample identifier (see original paper appendix)
Al	numeric	Percentage of aluminium oxide

Fe	numeric	Percentage of iron oxide
Mg	numeric	Percentage of magnesium oxide
Ca	numeric	Percentage of calcium oxide
Na	numeric	Percentage of natrium oxide
K	numeric	Percentage of kalium oxide
Ti	numeric	Percentage of titanim oxide
Mn	numeric	Percentage of manganese oxide
Ba	numeric	Percentage of baryum oxide
Site	factor	Kiln site
Pays	factor	Location of the kiln site
Couleur	factor	External color of the pottery
Date	factor	Approximate date of the pottery

Note

The DASL version of the dataset, as presented in the "Pottery stoty", does not include data on the poteries from the Gloucester site, neither the data on K, Ti, Mn and Ba oxides. It neither includes the color and date informations, and codes sites as their first letter only.

The DASL version of the dataset exists in the `car` package, as the `Pottery` dataset (with two locations differently spelled).

Source

Downloaded from the DASL (Data and Story Library) website, and completed from the original paper of Tubb et al.

References

A. Tubb, A. J. Parker, and G. Nickless (1980). The analysis of Romano-British pottery by atomic absorption spectrophotometry. *Archaeometry*, 22, 153-171.

Hand, D. J., Daly, F., Lunn, A. D., McConway, K. J., and E., O. (1994) *A Handbook of Small Data Sets*. Chapman and Hall – for the short version of the dataset.

Examples

```
data( poteries )
# Reconstruct the car version of this dataset
dcar <- poteries[ , c( 'Al', 'Fe', 'Mg', 'Ca', 'Na', 'Site' ) ]
dcar <- droplevels( dcar[ -which( dcar$Site == "College of Art" ), c( 6, 1:5 ) ] )
levels( dcar$Site )[ c( 1, 3, 4 ) ] <- c( "AshleyRails", "Islethorns", "Llanedyrn" )

# Reconstruct the DASL version of this dataset
ddasl <- dcar[ , c( 2:6, 1 ) ]
levels( ddasl$Site ) <- c( 'A', 'C', 'I', 'L' )
```

puissance *Estimate the power and the type-I error of the disjoint-subgraphs method*

Description

Estimate the power and the type-I error of the disjoint-graph method to detect a change in compositions between different conditions

Usage

```
estimer.puissance( composition, cv.composition,
                   taille.groupes = 10, masque,
                   f.p, v.X = 'Condition',
                   seuil.candidats = ( 5:30 ) / 100,
                   f.correct = groupes.identiques,
                   groupes.attendus = composition$Graphes[[ 1 ]]$Connexe,
                avec.classique = length( attr( composition, "reference" ) ) > 0,
                   f.correct.classique = genes.trouves,
                   genes.attendus,
                   B = 3000, n.coeurs = 1,
                   ... )

estimer.alpha( composition, cv.composition,
               taille.groupes = 10, masque,
               f.p, v.X = 'Condition',
               seuil.candidats = ( 5:30 ) / 100,
               avec.classique = length( attr( composition, "reference" ) ) > 0,
               B = 3000, n.coeurs = 1,
               ... )
```

Arguments

`composition`	A composition model, as obtained by `modele_compo`. For simulations under the null hypothesis (`estimer.alpha`), the first condition is duplicated to other conditions (but not the cv.composition, if provided as a matrix, allowing to explore some kinds of pseudo-null hypothesis).
`cv.composition`	The expected coefficient of variation of the quantified amounts. Should be either a single value, that will be used for all components and all conditions, or a matrix with the same structure than `composition$Absolue`: one row for each condition, one column for each component, in the same order and with the same names. Coefficients of variations are expected in the amount scale, in raw form (that is, give 0.2 for a 20% coefficient of variation).
`taille.groupes`	The sample size for each condition. Unused if `masque` is given. If a single value, it will be used for all conditions. Otherwise, should have the same length that the number of conditions in the provided model.

`masque`	A data.frame that will give the dataset design for a given experiment. Should contain at least one column containing the names of the conditions, with values being in the conditions names in `composition`. If not provided, it is generated from `taille.groupes` as a single column named ‘Condition’.
`f.p`	The function used to analyse the dataset. See `creer.Mp` for details.
`v.X`	The name of the column identifying the different conditions in `masque`.
`seuil.candidats`	A vector of p-value cut-offs to be tested. All values should be between 0 and 1.
`f.correct`	A function to determine if the result of the analysis is the expected one. Defaults to a function that compares the disjoint sub-graphs of a reference graph and the obtained one.
`groupes.attendus`	The reference graph for the above function. Defaults to the theoretical graph of the model, for the comparison between the first and the second conditions.
`avec.classique`	If `TRUE`, analysis is also done using an additive log-ratio (alr)-like method, using the geometric mean of the reference components as the “normalisation factor”. This correspond to the Delta-Delta-Ct method, or similar methods, in qRT-PCR. With this method, each non-reference component is tested in turn after division by the normalisation factor. If requested, the analysis is done with and without multiple testing correction (with Holm’s method). The “cut-off p-value” is used as the nominal type~I error level for the individual tests.
`f.correct.classique`	A function to determine if the alr-like method finds the correct answer. Defaults to a function that compares the set of significant tests with the set of expected components.
`genes.attendus`	A character vector giving the names of components expected to behave differently than the reference set.
`B`	The number of simulations to be done.
`n.coeurs`	The number of CPU cores to use in computation, with parallelization using forks (does not work on Windows) with the help of the parallel package.
`...`	Additionnal parameters for helper functions, including `f.p`, `f.correct` and `f.correct.classique`

Details

Use this function to simulate experiments and explore the properties of the disjoint graph method in a specified experimental context. Simulations are done using a log-normal model, so analysis is always done on the log scale. Coefficients of variation in the original scale hence directly translate into standard deviations in the log-scale.

For power analysis, care should be taken that any rejection of the null hypothesis “nothing is different between conditions” is counted as a success, even if the result does not respect the original changes. This is the reason for the additional correct-finding probability estimation. However, defining what is a correct, or at least acceptable, result may be not straightforward, especially for comparison with other analysis methods.

Note also that fair power comparisons can be done only for the same type I error level. Hence, for instance, power of the corrected alr-like method at p = 0.05 should be compared to the power of the disjoint-graph method at its “optimal” cut-off.

Value

An object of class `SARPcompo.simulation`, with a plot method. It is a data.frame with the following columns:

`Seuil`	The cut-offs used to build the graph
`Disjoint`	The number of simulations that led to disjoint graphs.
`Correct`	The number of simulations that led to the correct graph (as defined by the `f.correct` function).

If `avec.classique` is `TRUE`, it has additionnal columns:

`DDCt`	The number of simulations that led at least one significant test using the alr-like method.
`DDCt.H`	The number of simulations that led at least one significant test using the alr-like method, after multiple testing correction using Holm’s method.
`DDCt.correct`	The number of simulations that detected the correct components (as defined by the `f.correct.classique` function) using the alr-like method.
`DDCt.H.correct`	As above, but after multiple testing correction using Holm’s method.

It also stores a few informations as attributes, including the total number of simulations (attribute `n.simulations`).

Author(s)

Emmanuel Curis (<`emmanuel.curis@parisdescartes.fr`>)

See Also

`modele_compo` to create a compositional model for two or more conditions.

`creer.Mp`, which is used internally, for details about analysis functions.

`choisir.seuil` for a simpler interface to estimate the optimal cut-off.

Examples

```
## Create a toy example: four components, two conditions
##  components 1 and 2 do not change between conditions
##  components 3 and 4 are doubled
##  component  1 is a reference component
me <- rbind( 'A' = c( 1, 1, 1, 1 ),
             'B' = c( 1, 1, 2, 2 ) )
colnames( me ) <- paste0( "C-", 1:4 )

md <- modele_compo( me, reference = 'C-1' )
```

```
## How many simulations?
##  50 is for speed; increase for useful results...
B <- 50

## What is the optimal cut-off for this situation?
## (only a few simulations for speed, should be increased)
## (B = 3000 suggests a cut-off between 0.104 and 0.122)
seuil <- choisir.seuil( 4, B = B )

## What is approximately the type I error
## between conditions A and B using a Student test
## with a CV of around 50 % ?
##  (only a few simulations for speed, should be increased)
alpha <- estimer.alpha( md, cv = 0.50, B = B,
                        f.p = student.fpc )

# Plot it : darkgreen = the disjoint graph method
#           orange    = the alr-like method, Holm's corrected
#           salmon    = the alr-like method, uncorrected
plot( alpha )

## What is approximately the power to detect that something changes
## between conditions A and B using a Student test
## with a CV of around 50 % ?
##  (only a few simulations for speed, should be increased)
puissance <- estimer.puissance( md, cv = 0.50, B = B,
                                f.p = student.fpc,
                                genes.attendus = c( 'C-3', 'C-4' )  )

# Plot it : darkgreen = the disjoint graph method
#           orange    = the alr-like method, Holm's corrected
#           salmon    = the alr-like method, uncorrected
plot( puissance )

## Do we detect the correct situation in general?
##  (that is, exactly two sets: one with C-1 and C-2, the second with
##   C-3 and C-4 --- for the alr-like method, that only C-3 and C-4
##   are significant)
#           darkgreen = the disjoint graph method
#           orange    = the alr-like method, Holm's corrected
#           salmon    = the alr-like method, uncorrected
plot( puissance, correct = TRUE )
```

`SCH23390` *Effect of MDMA and SCH23390 on gene expression*

Description

This data set gives the mRNA quantification of several genes involved in the dopamin pathway in four different conditions: control, after addition of MDMA, after addition of SCH23390 and after addition of both.

Usage

```
data(SCH23390)
```

Format

A data frame with 8 columns and 48 rows. Each row gives the quantification results, expressed in cycle threshold (CT):

Groupe	factor	The group label
MDMA	factor	MDMA addition indicator (Oui=Yes, Non=No)
SCH23390	factor	SCH23390 addition indicator (as above)
Hprt	numeric	CT for the *hprt* gene, use as reference gene
Fos,Fosb,Egr1,Egr2	numeric	CT for the four genes of interest

Source

Original data communicated by the authors of the paper.

References

N. Benturquia, C. Courtin, F. Noble, and C. Marie-Claire (2008). Involvement of D1 dopamine receptor in MDMA-induced locomotor activity and striatal gene expression in mice. *Brain Research*, 1211, 1-5

Examples

```
data( SCH23390 )
# Optimal cut-off for five genes and alpha = 0.05
#  is around 0.13

# First step, experimental check
#
# MDMA should change expression levels of all genes but the reference
#  1) extract the data for the Ctrl vs MDMA groups comparison
d.MDMA <- SCH23390[ which( SCH23390$Groupe %in% c( 'Ctrl', 'MDMA' ) ), ]
M.MDMA <- creer.Mp( d = d.MDMA, noms = names( d.MDMA )[ 4:8 ], log = TRUE,
                    f.p = student.fpc, v.X = 'MDMA' )

#  2) L'arbre associé
plot( arbre.Mp( M.MDMA, reference = 'Hprt' ),
      seuil.p = c( 0.137, 0.128, 0.147 ) )

#  3) Le graphe pour le seuil optimal
#    => indeed, all genes are modified by MDMA
#       Fos and Fosb seems to have the same behavior
plot( grf.Mp( M.MDMA, reference = 'Hprt', p = 0.13 ) )
```

```
# Second step, experiment analysis
# Does SCH23390 modulate the MDMA effect?
#   => interaction term in a two-ways analysis of variance
M.I <- creer.Mp( d = SCH23390, noms = names( SCH23390 )[ 4:8 ], log = TRUE,
                 f.p = anva_SC.fpc,
                 frm = R ~ MDMA + SCH23390 + MDMA:SCH23390, SC = 3 )


#  2) L'arbre associé
plot( arbre.Mp( M.I, reference = 'Hprt' ),
      seuil.p = c( 0.137, 0.128, 0.147 ) )

#  3) Le graphe pour le seuil optimal
#    => no clear detection of interaction
plot( grf.Mp( M.I, reference = 'Hprt', p = 0.13 ) )
```

tests — *Utility functions to obtain p-values from tests on individual ratios*

Description

These functions can be used in the functions to perform analysis on all pairwise ratios of a compositional dataset

Usage

```
student.fpc( d, variable, v.X, var.equal = TRUE, ... )
anva1.fpc( d, variable, v.X, frm = NULL, ... )
anva1vi.fpc( d, variable, v.X, frm = NULL, ... )
rls.fpc( d, variable, v.X, frm = NULL, ... )
kw.fpc( d, variable, v.X, frm = NULL, ... )
anva_SC.fpc( d, variable, frm, SC = 1, type = 1, ... )
```

Arguments

`d`	The data frame that contains the ratio to test, and all variables of the original data frame that where not used as compositional data.
`variable`	A length-one character vector containing the names of the variable corresponding to the ratio (or log-ratio) to test.
`v.X`	The **name** of the explanatory (independant, predictor) variable. This variable should be a factor for `anva1.fpc`, `anva1vi.fpc` and `kw.fpc` and a numeric for `rls.fpc`.
`frm`	The formula to use. Defaults to `as.formula( paste0( variable,"~",v.X ) )` for `anva1.fpc`, `anva1vi.fpc` and `kw.fpc`. Providing the formula speeds up the computation, since it avoids repeating the construction step for each ratio. For `anva_SC.fpc`, giving the formula is mandatory and `variable` is unused. Beware of the term order to select the right sum of squares to test!

`SC`	For `anva_SC.fpc`, the number of the line to use in the analysis of variance table to get a p-value, see details.
`type`	For `anva_SC.fpc`, the kind of sums of square to be used when constructing the analysis of variance table, see details.
`var.equal`	For `student.fpc`, shall we assume that variance are equals in the two groups (`TRUE`, the default) or not (`FALSE`). Same as in `t.test`.
`...`	additional arguments

Details

These functions are only wrapper to some commonly used tests. The correspondance is as follow

`student.fpc`	Student's T-test	`t.test()$p.value`
`anva1.fpc`	One-way analysis of variance	`anova(lm())[ 1, 5 ]`
`rls.fpc`	Simple linear regression	`anova(lm())[ 1, 5 ]`
`anva1vi.fpc`	One-way analysis of variance, without equal variance assumption	`oneway.test()$p.value`
`kw.fpc`	Kruskal-Wallis test	`kruskal.test()$p.value`

`anva_SC.fpc` is a generic wrapper for `lm` using any formula. It then extracts the *p*-value of the line given by `SC` in the analysis of variance table. If `type = 1`, the table is built using `anova` and corresponds to type 1 (sequential sum of square). If `type = 2` or `type = 3`, the table is built using `car::Anova` and corresponds either to type 2 or type 3 sums of squares.

For Student's test (either with equal or unequal variances), instead of calling t.test, the computation is done internally, hopefully speeding up (less controls are done and only useful computations are done)

Value

These function returns the *p*-value from the corresponding test.

Note

`rls.fpc` is an exact synonym for `anva1.fpc`, since the underlying theory is the same. Distinction is made to help users without a formal statistical background to find the right test.

Author(s)

Emmanuel Curis (`<emmanuel.curis@parisdescartes.fr>`)

See Also

`kruskal.test`, `lm`, `anova`, `Anova`, `oneway.test`, for corresponding tests.

`equiv.fpc` for an approach using equivalence tests.

`creer.Mp` to use these functions to create a matrix of *p*-values.

Examples

```
# load the potery data set
data( poteries )

# Compute one-way ANOVA p-values for all ratios in this data set
Mp <- creer.Mp( poteries, c( 'Al', 'Na', 'Fe', 'Ca', 'Mg' ),
                f.p = anva1.fpc, v.X = 'Site', frm = R ~ Site )
Mp

# Make a graphe from it and plot it
plot( grf.Mp( Mp ) )
```

Index

∗ **compositional**
    arbre.Mp, 2
    choisir.seuil, 6
    choisir.seuil.equiv, 9
    conversions, 11
    coupures.Mp, 12
    creer_data.frame, 14
    creer_graphe, 16
    creer_matrice, 18
    distances, 20
    equivalence, 22
    fichiers, 23
    modele, 28
    puissance, 32
    tests, 37
∗ **datasets**
    BpLi, 4
    GADL1, 26
    generef, 27
    poteries, 30
    SCH23390, 35
∗ **equivalence**
    equivalence, 22

alpha *(*puissance*)*, 32
Anova, *38*
anova, *38*
anva1.fpc, *15*, *19*, *25*
anva1.fpc *(*tests*)*, 37
anva1vi.fpc *(*tests*)*, 37
anva_SC.fpc *(*tests*)*, 37
arbre.Mp, 2, *13*
as.data.frame, *5*, *14*, *18*, *24*
as.dendrogram, *3*

BpLi, 4
BpLi_J2 *(*BpLi*)*, 4
BpLi_J4 *(*BpLi*)*, 4

calc.rapports, 5
choisir.seuil, *3*, 6, *10*, *13*, *34*
choisir.seuil.equiv, 9
components, *29*
conversions, 11
coupures.Mp, *3*, 12
Cq.reference *(*generef*)*, 27
create.DFp *(*creer_data.frame*)*, 14
create.Fp *(*fichiers*)*, 23
create.graphe *(*creer_graphe*)*, 16
create.Mp *(*creer_matrice*)*, 18
creer.DFp, *12*, *18*, *25*
creer.DFp *(*creer_data.frame*)*, 14
creer.Fp *(*fichiers*)*, 23
creer.Mp, *3*, *6–8*, *10*, *12–15*, *18*, *21*, *23–25*, *33*, *34*, *38*
creer.Mp *(*creer_matrice*)*, 18
creer_data.frame, 14
creer_graphe, 16
creer_matrice, 18

DFp.Mp *(*conversions*)*, 11
distances, 20
distrib.distances *(*distances*)*, 20

equiv.fpc, *10*, *38*
equiv.fpc *(*equivalence*)*, 22
equivalence, 22
estimer.alpha, *30*
estimer.alpha *(*puissance*)*, 32
estimer.puissance, *30*
estimer.puissance *(*puissance*)*, 32

fichier *(*fichiers*)*, 23
fichiers, 23

GADL1, 26
gadl1 *(*GADL1*)*, 26
generef, 27
grf.DFp, *15*, *25*
grf.DFp *(*creer_graphe*)*, 16

grf.Fp *(fichiers)*, 23
grf.Mp, *3*, *13*, *19*, *21*, *25*
grf.Mp *(creer_graphe)*, 16

kruskal.test, *38*
kw.fpc, *15*, *19*, *25*
kw.fpc *(tests)*, 37

lm, *38*

make.names, *15*, *19*, *24*
modele, 28
modele_compo, *32*, *34*
modele_compo *(modele)*, 28
Mp.DFp *(conversions)*, 11

oneway.test, *38*

plot, *13*, *29*
plot.Arbre *(arbre.Mp)*, 2
plot.Coupures *(coupures.Mp)*, 12
plot.dendrogram, *3*
plot.SARPcompo.H0 *(choisir.seuil)*, 6
plot.SARPcompo.modele *(modele)*, 28
poteries, 30
print.SARPcompo.H0 *(choisir.seuil)*, 6
puissance, 32

read.table, *11*, *24*
rls.fpc *(tests)*, 37

SCH23390, 35
sch23390 *(SCH23390)*, 35
student.fpc, *23*
student.fpc *(tests)*, 37

t.test, *22*, *38*
tests, 37

write.table, *11*, *24*