Package ‘Patterns’
October 12, 2022

**Type**  Package  
**Title**  Deciphering Biological Networks with Patterned Heterogeneous Measurements  
**Version**  1.3  
**Date**  2021-03-19  
**Depends**  R (>= 3.5.0)

**biocViews**  
**Imports**  abind, gplots, graphics, grDevices, grid, igraph, jetset, KernSmooth, lars, lattice, limma, magic, methods, Mfuzz, movMF, msgps, nls, plotrix, SelectBoost, splines, stats4, survival, tnet, VGAM, WGCNA

**Suggests**  animation, Biobase, c060, cluster, elasticnet, glmnet, pixmap, biomaRt, CascadeData, repmis, knitr, R.rsp, spls, rmarkdown

**Author**  Frederic Bertrand [cre, aut] (<https://orcid.org/0000-0002-0837-8281>), Myriam Maumy-Bertrand [aut] (<https://orcid.org/0000-0002-4615-1512>)

**Maintainer**  Frederic Bertrand <frederic.bertrand@math.unistra.fr>

**Description**  A modeling tool dedicated to biological network modeling (Bertrand and others 2020, <doi:10.1093/bioinformatics/btaa855>). It allows for single or joint modeling of, for instance, genes and proteins. It starts with the selection of the actors that will be the used in the reverse engineering upcoming step. An actor can be included in that selection based on its differential measurement (for instance gene expression or protein abundance) or on its time course profile. Wrappers for actors clustering functions and cluster analysis are provided. It also allows reverse engineering of biological networks taking into account the observed time course patterns of the actors. Many inference functions are provided and dedicated to get specific features for the inferred network such as sparsity, robust links, high confidence links or stable through resampling links. Some simulation and prediction tools are also available for cascade networks (Jung and others 2014, <doi:10.1093/bioinformatics/btt705>). Example of use with microarray or RNA-Seq data are provided.

**License**  GPL (>= 2)

**Encoding**  UTF-8
micro_array-network.R micropredict.R
Classification/MSC 62J05, 62J07, 62J99, 92C42
VignetteBuilder knitr
RoxygenNote 7.1.1
URL https://fbertran.github.io/Patterns/,
https://github.com/fbertran/Patterns/
BugReports https://github.com/fbertran/Patterns/issues/
NeedsCompilation no
Repository CRAN
Date/Publication 2021-03-21 10:00:07 UTC

R topics documented:

Patterns-package .................................................. 3
analyze_network.network-method ............................. 4
as.micro_array .................................................. 4
CascadeFinit ................................................... 6
CascadeFshape .................................................. 6
CLL ............................................................. 7
clustExploration.micro_array-method ...................... 8
clustInference.micro_array.numeric-method ............... 9
compare-methods ............................................... 10
cutoff.network-method ....................................... 11
dim .......................................................... 12
evolution.network-method ................................ 12
geneNeighborhood.network-method ....................... 14
geneSelection ............................................... 16
gene_expr_simulation.network-method .................... 19
head.micro_array-method .................................. 20
IndicFinit ..................................................... 21
IndicFshape .................................................. 22
inference ..................................................... 23
infos ........................................................ 26
M ............................................................. 26
micropredict-class ......................................... 27
micro_array-class ........................................... 27
Net .......................................................... 28
network ....................................................... 28
network-class ................................................. 28
network2gp .................................................. 29
networkCascade ............................................... 30
network_random ............................................. 30
Net_inf_PL ..................................................... 31
**The Patterns Package**

**Description**

A modeling tool dedicated to biological network modeling. It allows for single or joint modeling of, for instance, genes and proteins. It starts with the selection of the actors that will be the used in the reverse engineering upcoming step. An actor can be included in that selection based on its differential measurement (for instance gene expression or protein abundance) or on its time course profile. Wrappers for actors clustering functions and cluster analysis are provided. It also allows reverse engineering of biological networks taking into account the observed time course patterns of the actors. Many inference functions are provided and dedicated to get specific features for the inferred network such as sparsity, robust links, high confidence links or stable through resampling links. Some simulation and prediction tools are also available for cascade networks. Example of use with microarray or RNA-Seq data are provided.

**Author(s)**

This package has been written by Frederic Bertrand in collaboration with Myriam Maumy-Bertrand. Maintainer: <frederic.bertrand@math.unistra.fr>

**References**


analyze_network(network-method)

*Analysing the network*

**Description**

Calculates some indicators for each node in the network.

**Usage**

```r
## S4 method for signature 'network'
analyze_network(Omega, nv, label_v = NULL)
```

**Arguments**

- **Omega**: a network object
- **nv**: the level of cutoff at which the analysis should be done
- **label_v**: (optional) the name of the genes

**Value**

A matrix containing, for each node, its betweenness, its degree, its output, its closeness.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
data(network)
analyze_network(network, nv=0)
```

---

**as.micro_array**

*Coerce a matrix into a micro_array object.*

**Description**

Coerce a matrix into a micro_array object.
**Usage**

```r
as.micro_array(
M,
  time,
  subject,
  name_probe = NULL,
  gene_ID = NULL,
  group = 0,
  start_time = 0
)
```

**Arguments**

- **M**: A matrix. Contains the microarray measurements. Should be of size N * K, with N the number of genes and K=T*P with T the number of time points, and P the number of subjects. This matrix should be created using `cbind(M1,M2,...)` with M1 a N*T matrix with the measurements for patient 1, M2 a N*T matrix with the measurements for patient 2.

- **time**: A vector. The time points measurements.

- **subject**: The number of subjects.

- **name_probe**: Vector with the row names of the micro array.

- **gene_ID**: Vector with the actors’ IDs of the row names of the micro array.

- **group**: Vector with the actors’ groups of the row names of the micro array.

- **start_time**: Vector with the actors’ starting time (i.e. the time it is thought to begin to have an effect on another actor in the network).

**Value**

A micro_array object.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
if(require(CascadeData)){
  data(micro_US, package="CascadeData")
  micro_US<-as.micro_array(micro_US[1:100,],time=c(60,90,210,390),subject=6)
  plot(micro_US)
}
```
CascadeFinit

Create initial F matrices for cascade networks inference.

Description
This is an helper function to create initial values F matrices for cascade networks.

Usage
CascadeFinit(sqF, ngrp, low.trig = TRUE)

Arguments

- sqF  Size of an F cell
- ngrp Number of groups
- low.trig Fill the lower trigonal matrices with ones

Value
An array of sizes c(sqF, sqF, ngrp).

Author(s)
Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```R
CascadeFinit(3,2)
CascadeFinit(4,3)
plotF(CascadeFinit(4,3),choice = "F")
CascadeFinit(3,2,low.trig=FALSE)
CascadeFinit(4,3,low.trig=FALSE)
plotF(CascadeFinit(4,3,low.trig=FALSE),choice = "F")
```

CascadeFshape

Create F matrices shaped for cascade networks inference.

Description
This is an helper function to create F matrices with special shape used for cascade networks.

Usage
CascadeFshape(sqF, ngrp)
**Arguments**

- sqF: Size of an F cell
- ngrp: Number of groups

**Value**

An array of sizes c(sqF, sqF, ngrp).

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
CascadeFshape(3,2)
plotF(CascadeFshape(3,2),choice = "Fshape")
CascadeFshape(4,3)
plotF(CascadeFshape(4,3),choice = "Fshape")
```

---

**CLL**  
*Expression data from healthy and malignant (chronic lymphocytic leukemia, CLL) human B-lymphocytes after B-cell receptor stimulation (GSE 39411 dataset)*

**Description**

B-cells were negatively selected from healthy donors and previously untreated CLL patients. BCR stimulated and unstimulated control B-cells were treated at four time points after stimulation for total RNA extraction and hybridization on Affymetrix microarrays.

**Format**

The format is: chr "CLL"

**Details**

The dataset provided with package is the first five lines of the full dataset. The full dataset can be downloaded from the github repository of the package (https://raw.githubusercontent.com/fbertran/Patterns/master/add_data/CLL.RData). Three different cell populations (6 healthy B-lymphocytes, 6 leukemic CLL B-lymphocyte of indolent form and 5 leukemic CLL B-lymphocyte of aggressive form) were stimulated in vitro with an anti-IgM antibody, activating the B-cell receptor (BCR). We analyzed the gene expression at 4 time points (60, 90, 210 and 390 minutes). Each gene expression measurement is performed both in stimulated cells and in control unstimulated cells. For one aggressive CLL case, we silenced expression of DUSP1 by transfecting DUSP1-specific RNAi and, as a control, transfected cells with a non-targeting RNAi. We then stimulated the BCR of these cells and analyzed the gene expression at the same time points in stimulated cells and in control unstimulated cells.
Author(s)
Bertrand Frederic, Myriam Maumy-Bertrand.

Source

References

Examples

data(CLL)
str(CLL)

CLLfile <- "https://raw.githubusercontent.com/fbertran/Patients/master/add_data/CLL.RData"
load(CLLfile)
str(CLL)

---

clustExploration,micro_array-method

A function to explore a dataset and cluster its rows.

Description
Based on soft clustering performed by the Mfuzz package.

Usage

```r
## S4 method for signature 'micro_array'
clustExploration(microarray, new.window = FALSE)
```

Arguments

- `microarray`: A microarray to cluster
- `new.window`: Boolean. New X11 window for plots. Defaults to FALSE.

Value

A data.frame of `nrows(microarray)` observations of 3 variables (name, cluster, maj.vote.index).
**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
library(Patterns)
if(require(CascadeData)){
data(micro_S, package="CascadeData")
D<-Patterns::as.micro_array(micro_S[1:100,],1:4,6)
a<-clustExploration(D)
a
}
```

---

**clustInference,micro_array,numeric-method**

_A function to explore a dataset and cluster its rows._

**Description**

Based on soft clustering performed by the Mfuzz package.

**Usage**

```r
## S4 method for signature 'micro_array,numeric'
clustInference(microarray, vote.index, new.window = FALSE)
```

**Arguments**

- **microarray**: A microarray to cluster
- **vote.index**: Option for cluster attribution
- **new.window**: Boolean. New X11 window for plots. Defaults to FALSE.

**Value**

A list of two elements:

- **res.matrix**: A data.frame of nrows(microarray) observations of 3 variables (name, cluster, maj.vote.index).
- **prop.matrix**: Additionnal info.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.
**Examples**

```r
library(Patterns)
if(require(CascadeData)){
data(micro_S, package="CascadeData")
D<-Patterns::as.micro_array(micro_S[1:20,,1:4,6])
b<-Patterns::clustInference(D,0.5)
b
}
```

**Description**

Allows comparison between actual and inferred network.

**Usage**

```r
## S4 method for signature 'network, network, numeric'
compare(Net, Net_inf, nv = 1)
```

**Arguments**

- **Net** A network object containing the actual network.
- **Net_inf** A network object containing the inferred network.
- **nv** A number that indicates at which level of cutoff the comparison should be done.

**Value**

A vector containing: sensitivity, predictive positive value, the usual F-score \((2*ppv*sens/(ppv+pe+sens))\), the \(1/2\) ponderated Fscore \(((1+0.5^2)*ppv*sens/(ppv/4+sens))\) and the 2 ponderated Fscore \(((1+2^2)*ppv*sens/(ppv*4+sens))\).

**Methods**

```r
list("signature(Net = "network", Net_inf = "network", nv = "numeric")")
```

- **Net** A network object containing the actual network.
- **Net_inf** A network object containing the inferred network.
- **nv** A number that indicates at which level of cutoff the comparison should be done.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.
Examples

data(Net)
data(Net_inf_PL)

#Comparing true and inferred networks
Crit_values=NULL

#Here are the cutoff level tested
test.seq<-seq(0,max(abs(Net_inf_PL@network*0.9)),length.out=200)
for(u in test.seq){
  Crit_values<-rbind(Crit_values,Patterns::compare(Net,Net_inf_PL,u))
}
matplot(test.seq,Crit_values,type="l",ylab="Criterion value",xlab="Cutoff level",lwd=2)
legend(x="topleft", legend=colnames(Crit_values), lty=1:5,col=1:5,ncol=2,cex=.9)

Description

Allows estimating the best cutoff. For a sequence of cutoff, the p value corresponding to each cutoff value of the sequence. Mainly recommended for single time cascade networks. To achieve more sparsity in other settings, please use a fitting function based on the stability selection or selectboost algorithms.

Usage

## S4 method for signature 'network'
cutoff(Omega, sequence = NULL, x_min = 0)

Arguments

Omega              a network object
sequence           a vector corresponding to the sequence of cutoffs that will be tested.
x_min              an integer ; only values over x_min are further retained for performing the test.

Value

A list containing two objects :
p.value            the p values corresponding to the sequence of cutoff
p.value.inter      the smoothed p value vector, using the loess function

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.
Examples

data(network)
cutoff(network)
#See vignette for more details

---

dim 

*Dimension of the data*

Description

Dimension of the data

Usage

```r
## S4 method for signature 'micro_array'
dim(x)
```

Arguments

- `x` an object of class `micro_array`.

Methods

- `list("signature(x = \"micro_array\")")` Gives the dimension of the matrix of measurements.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

---

evolution, network-method

*See the evolution of the network with change of cutoff*

Description

See the evolution of the network with change of cutoff
Usage

```r
## S4 method for signature 'network'

function

net,
list_nv,
\text{gr} = \text{NULL},
color.vertex = \text{NULL},
color.edge = \text{NULL},
fix = \text{TRUE},
size = \text{c}(2000, 1000),
label_v = \text{1:dim(net@network)[1]},
\text{legend.position} = \text{"topleft"},
frame.color = \text{"black"},
label.hub = \text{FALSE},
\text{outdir},
type.ani = \text{"html"}
```

Arguments

- `net`: a network object
- `list_nv`: a vector of cutoff at which the network should be shown
- `gr`: a vector giving the group of each gene. Defaults to `NULL`
- `color.vertex`: a vector giving the color of each node. Defaults to `NULL`
- `color.edge`: a vector giving the color of each edge. Defaults to `NULL`
- `fix`: logical, should the position of the node in the network be calculated once at the beginning? Defaults to `TRUE`.
- `size`: vector giving the size of the plot. Defaults to `c(2000, 1000)`
- `label_v`: vector giving the labels of each vertex. Defaults to `1:dim(net@network)[1]`
- `legend.position`: string giving the position of the legend. Defaults to "topleft"
- `frame.color`: string giving the color of the frame of the plot. Defaults to "black"
- `label.hub`: label hubs. Defaults to `FALSE`
- `outdir`: Directory to save the animation. No default value since it must be specified by the user.
- `type.ani`: Type of animation. Defaults to "html"
- `legend`: string giving the position of the legend. Defaults to "topleft"

Details

Several types of outputs are available using the `type.ani` option.

- `html`
- `latex` (requires latex)
• swf (requires swftools)
• video (requires ffmpeg)
• gif
• manual_gif

Value
A HTML page with the evolution of the network.

Author(s)
Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(network)
sequence<-seq(0,0.2,length.out=20)

#Change the destdir to have the animation created where you want.
destdir = tempdir()

#Example of use of the evolution method with an html output.
evolution(network,sequence,type.ani = "html",outdir=destdir)

#Example of use of the evolution method with an animated gif output.
evolution(network,sequence,type.ani = "gif",outdir=destdir)

geneNeighborhood, network-method

*Find the neighborhood of a set of nodes.*

Description
Find the neighborhood of a set of nodes.

Usage

```
## S4 method for signature 'network'
geneNeighborhood(
  net,
  targets,
  nv = 0,
  order = length(net@time_pt) - 1,
  label_v = NULL,
```
Arguments

net          a network object
targets      a vector containing the set of nodes
nv           the level of cutoff. Default to 0.
order        of the neighborhood. Default to ‘length(net@time_pt)-1’.
label_v      vector defining the vertex labels.
ini          using the “position” function, you can fix the position of the nodes.
frame.color  color of the frames.
label.hub    logical ; if TRUE only the hubs are labeled.
graph        plot graph of the network. Defaults to ‘TRUE’.
names        return names of the neighbors. Defaults to ‘FALSE’.

Value

The neighborhood of the targeted genes.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(Selection)
data(infos)
#Find probesets for EGR1
pbst_EGR1 = infos[infos$hgnc_symbol=="EGR1", "affy_hg_u133_plus_2"]
gene_IDs = infos[match(Selection$name, infos$affy_hg_u133_plus_), "hgnc_symbol"]
data(network)
#A nv value can chosen using the cutoff function
nv=.11
EGR1=which(is.element(Selection$name,pbst_EGR1))
P<-position(network,nv=nv)
geneNeighborhood(network,targets=EGR1,nv=nv,ini=P,
label_v=gene_IDs)
Methods for selecting genes

Description
Selection of differentially expressed genes.

Usage
## S4 method for signature 'micro_array,micro_array,numeric'
geneSelection(
  x,
  y,
  tot.number,
  data_log = TRUE,
  wanted.patterns = NULL,
  forbidden.patterns = NULL,
  peak = NULL,
  alpha = 0.05,
  Design = NULL,
  lfc = 0
)

## S4 method for signature 'list,list,numeric'
geneSelection(
  x,
  y,
  tot.number,
  data_log = TRUE,
  alpha = 0.05,
  cont = FALSE,
  lfc = 0,
  f.asso = NULL,
  return.diff = FALSE
)

## S4 method for signature 'micro_array,numeric'
genePeakSelection(
  x,
  peak,
  y = NULL,
  data_log = TRUE,
  durPeak = c(1, 1),
  abs_val = TRUE,
  alpha_diff = 0.05
)
Arguments

x  either a micro_array object or a list of micro_array objects. In the first case, the micro_array object represents the stimulated measurements. In the second case, the control unstimulated data (if present) should be the first element of the list.

y  either a micro_array object or a list of strings. In the first case, the micro_array object represents the stimulated measurements. In the second case, the list is the way to specify the contrast:

First element: condition, condition&time or pattern. The condition specification is used when the overall is to compare two conditions. The condition&time specification is used when comparing two conditions at two precise time points. The pattern specification allows to decide which time point should be differentially expressed.

Second element: a vector of length 2. The two conditions which should be compared. If a condition is used as control, it should be the first element of the vector. However, if this control is not measured through time, the option cont=TRUE should be used.

Third element: depends on the first element. It is no needed if condition has been specified. If condition&time has been specified, then this is a vector containing the time point at which the comparison should be done. If pattern has been specified, then this is a vector of 0 and 1 of length T, where T is the number of time points. The time points with desired differential expression are provided with 1.

tot.number  an integer. The number of selected genes. If tot.number <0 all differentially genes are selected. If tot.number > 1, tot.number is the maximum of differentially genes that will be selected. If 0<tot.number<1, tot.number represents the proportion of differentially genes that are selected.

data_log  logical (default to TRUE); should data be logged ?

wanted.patterns  a matrix with wanted patterns [only for geneSelection].

forbidden.patterns  a matrix with forbidden patterns [only for geneSelection].

peak  interger. At which time points measurements should the genes be selected [optional for geneSelection].

alpha  float; the risk level. Default to ‘alpha=0.05’

Design  the design matrix of the experiment. Defaults to ‘NULL’.

lfc  log fold change value used in limma’s ‘topTable’. Defaults to 0.

cont  use contrasts. Defaults to ‘FALSE’.

f.asso  function used to assess the association between the genes. The default value ‘NULL’ implies the use of the usual ‘mean’ function.

return.diff  [FALSE] if TRUE then the function returns the stimulated expression of the differentially expressed genes

durPeak  vector of size 2 (default to c(1,1)) ; the first elements gives the length of the peak at the left, the second at the right. [only for genePeakSelection]
abs_val logical (default to TRUE) ; should genes be selected on the basis of their absolute value expression? [only for genePeakSelection]

alpha_diff float; the risk level

Value
A micro_array object.

Author(s)
Frédéric Bertrand, Myriam Maumy-Bertrand.

Examples

if(require(CascadeData)){
data(micro_US)
micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
data(micro_S)
micro_S<-as.micro_array(micro_S,time=c(60,90,210,390),subject=6)

#Basically, to find the 50 more significant expressed genes you will use:
Selection_1<-geneSelection(x=micro_S,y=micro_US,
tot.number=50,data_log=TRUE)
summary(Selection_1)

#If we want to select genes that are differentially at time t60 or t90:
Selection_2<-geneSelection(x=micro_S,y=micro_US,tot.number=30,
wanted.patterns=rbind(c(0,1,0,0),c(1,0,0,0),c(1,1,0,0)))
summary(Selection_2)

#To select genes that have a differential maximum of expression at a specific time point.
Selection_3<-genePeakSelection(x=micro_S,y=micro_US,peak=1,
abs_val=FALSE,alpha_diff=0.01)
summary(Selection_3)}

if(require(CascadeData)){
data(micro_US)
micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
data(micro_S)
micro_S<-as.micro_array(micro_S,time=c(60,90,210,390),subject=6)

#Genes with differential expression at t1
Selection1<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns=rbind(c(1,0,0,0)))

#Genes with differential expression at t2
Selection2<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns=rbind(c(0,1,0,0)))

#Genes with differential expression at t3
Selection3<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns=rbind(c(0,0,1,0)))
```r
# Genes with differential expression at t4
Selection4<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns= rbind(c(0,0,0,1)))

# Genes with global differential expression
Selection5<-geneSelection(x=micro_S,y=micro_US,20)

# We then merge these selections:
Selection<-unionMicro(list(Selection1,Selection2,Selection3,Selection4,Selection5))
print(Selection)

# Prints the correlation graphics Figure 4:
summary(Selection,3)

## Uncomment this code to retrieve geneids.
library(org.Hs.eg.db)

# ff<-function(x){substr(x, 1, nchar(x)-3)}
# ff<-Vectorize(ff)

## Here is the function to transform the probeset names to gene ID.
# library("hgu133plus2.db")

# probe_to_id<-function(n){
# x <- hgu133plus2Symbol
# mp<-mappedkeys(x)
# xx <- unlist(as.list(x[mp]))
# genes_all = xx[(n)]
# genes_all[is.na(genes_all)]<"unknown"
# return(genes_all)
# }
# Selection@name<-probe_to_id(Selection@name)
```

---

**gene_expr_simulation**, **network-method**

_Simulates microarray data based on a given network._

### Description

Simulates microarray data based on a given network.

### Usage

```r
## S4 method for signature 'network'
gene_expr_simulation(
  network,
  time_label = 1:4,
  subject = 5,
```
peak_level = 100,
act_time_group = 1:4
)

Arguments

network A network object.
time_label a vector containing the time labels.
subject the number of subjects
peak_level the mean level of peaks.
act_time_group [NULL] vector ; at which time the groups (defined by sort(unique(group))) are activated?

Value

A micro_array object.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(Net)
set.seed(1)

#We simulate gene expressions according to the network Net
Msim<-Patterns::gene_expr_simulation(
    network=Net,
time_label=rep(1:4,each=25),
subject=5,
peak_level=200)
head(Msim)
**IndicFinit**

Create initial F matrices using specific intergroup actions for network inference.

**Description**

This is an helper function to create initial values F matrices for networks.

**Usage**

```r
IndicFinit(sqF, ngrp, Indic, low.trig = TRUE)
```

**Arguments**

- `sqF`: Size of an F cell
- `ngrp`: Number of groups
- `Indic`: Matrix to specify where there is an interaction from one group to another
- `low.trig`: Fill the lower trigonal matrices with ones

**Value**

An array of size `(sqF, sqF, ngrp)`.
**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

\[ \text{IndicFshape}(3, 2, \text{matrix}(1,2,2)-\text{diag}(2)) \]

---

**IndicFshape**  
Create F matrices using specific intergroup actions for network inference.

**Description**

This is an helper function to create values F matrices using specific intergroup actions for network inference.

**Usage**

\[ \text{IndicFshape}(\text{sqF}, \text{ngrp}, \text{Indic}) \]

**Arguments**

- **sqF**  
  Size of an F cell
- **ngrp**  
  Number of groups
- **Indic**  
  Matrix to specify where there is an interaction from one group to another

**Value**

An array of size (sqF, sqF, ngrp).

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

\[ \text{IndicFshape}(3, 2, \text{matrix}(1,2,2)-\text{diag}(2)) \]
Reverse-engineer the network.

## S4 method for signature 'micro_array'

inference(
  M,
  tour.max = 30,
  g = function(x) { 1/x },
  conv = 0.001,
  cv.subjects = TRUE,
  nb.folds = NULL,
  eps = 10^-5,
  type.inf = "iterative",
  Fshape = NULL,
  Finit = NULL,
  Omega = NULL,
  fitfun = "LASSO",
  use.Gram = TRUE,
  error.stabsel = 0.05,
  pi_thr.stabsel = 0.6,
  priors = NULL,
  mc.cores = getOption("mc.cores", 2L),
  intercept.stabpath = TRUE,
  steps.seq = 0.95,
  limselect = 0.95,
  use.parallel = TRUE,
  verbose = TRUE,
  show.error.messages = FALSE
)

Arguments

- **M**: a micro_array object.
- **tour.max**: [30] tour.max + 1 = maximal number of steps.
- **g**: After each step, the new solution is choosen as (the old solution + g(x) * the new solution)/(1+g(x)) where x is the number of steps. Defaults to 'g=function(x) 1/x'
- **conv**: [0.001] Convergence criterion.
- **cv.subjects**: [TRUE] Subjectwise cross validation: should the cross validation be done by removing the subject one by one?
nb.folds [NULL] Relevant only if no subjectwise cross validation (i.e. cv.subjects=FALSE). The number of folds in cross validation.

eps [10^-5] Threshold for rounding coefficients to 0 (i.e. machine zero).

type.inf ["iterative"] "iterative" or "noniterative" : should the algorithm be computed iteratively or only for one step? For highly homogeneous clusters, the "noniterative" option is sufficient.

Fshape [NULL] Shape of the F matrix.

Finit [NULL] Init values of the F matrix.

Omega [NULL] Init values for the Omega matrix.

fitfun ["LASSO"] Function to infer the Omega matrix at each step.


error.stabsel [0.05] Optional parameter for the stability selection algorithm in the ‘c060’ package.

pi_thr.stabsel [0.6] Optional parameter for the stability selection algorithm in the ‘c060’ package.

priors [NULL] A priori weights for the links between the actors. 0 means that an actor is always included in the predictive model, 1 is a neutral weighting and +infinity that the actor is never used in the model. For a given predictive model, the weighting vector is normalized so that its sum is equal to the number of predictors in the model.

mc.cores [getOption("mc.cores", 2L)] Number of cores.

intercept.stabpath [TRUE] Use intercept in stability selection models?

steps.seq [.95] Optional parameter for the SelectBoost algorithm in the ‘SelectBoost’ package.

limselect [.95] Optional parameter for the SelectBoost algorithm in the ‘SelectBoost’ package.

use.parallel [TRUE] Use parallel computing?

verbose [TRUE] Info on the completion of the fitting process

show.error.messages [FALSE] Should the error messages of the Omega estimating function be returned?

Details

The fitting built-in fitting functions (‘fitfun’) provided with the ‘Patterns’ package are :

**LASSO** from the ‘lars’ package (default value)

**LASSO2** from the ‘glmnet’ package

**SPLS** from the ‘spls’ package

**ELASTICNET** from the ‘elasticnet’ package

**stability.c060** from the ‘c060’ package implementation of stability selection
**stability.c060.weighted** a new weighted version of the 'c060' package implementation of stability selection

**robust lasso** from the 'lars' package with light random Gaussian noise added to the explanatory variables

**selectboost.weighted** a new weighted version of the 'selectboost' package implementation of the selectboost algorithm to look for the more stable links against resampling that takes into account the correlated structure of the predictors. If no weights are provided, equal weights are for all the variables (=non weighted case).

The weights are viewed as a penalty factors in the penalized regression model: it is a number that multiplies the lambda value in the minimization problem to allow differential shrinkage, [Friedman et al. 2010](https://web.stanford.edu/~hastie/Papers/glmnet.pdf), equation 1 page 3. If equal to 0, it implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables. Infinity means that the variable is excluded from the model. Note that the weights are rescaled to sum to the number of variables.

**Value**

A network object.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
# With simulated data, default shaped F matrix and default LASSO from the lars package
# as fitting function
data(M)
inference(M)
str(infM)
plot(infM, choice="F", nround=0)
plot(infM, choice="F", nround=1)

# With simulated data, cascade network shaped F matrix (1 group per time measurement case)
# and default LASSO from the lars package as fitting function
infM <- inference(M, Finit=CascadeFinit(4,4), Fshape=CascadeFshape(4,4))
str(infM)
plot(infM, choice="F", nround=0)
plot(infM, choice="F", nround=1)

# With selection of genes from GSE39411
data(Selection)
inference(Selection, Finit=CascadeFinit(4,4), Fshape=CascadeFshape(4,4))
str(infSel)
plot(infSel, choice="F", nround=0)
plot(infSel, choice="F", nround=1)
```
infos

Details on some probesets of the affy_hg_u133_plus_2 platform.

Description

Dataset with information on the affy_hg_u133_plus_2 platform such as probeset name (affy_hg_u133_plus_2),
enssembl_gene_id, entrezgene, hgnc_symbol, chromosome_name, start_position, end_position and
band.

Format

The format is: chr "infos"

Details

Data.frame with 8859 rows and 8 variables.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(infos)

M

Simulated microarray.

Description

Simulated M, microarray.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(M)
head(M)
str(M)
**micropredict-class**  
*Class "micropredict"*

---

**Description**

The "micropredict" class

**Objects from the Class**

Objects can be created by calls of the form `new("micropred", ...)`.  

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```
showClass("micropredict")
```

---

**micro_array-class**  
*Class "micro_array"*

---

**Description**

The "micro_array" class

**Objects from the Class**

Objects can be created by calls of the form `new("micro_array", ...)`.  

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```
showClass("micro_array")
```
**Net**

*Simulated network for examples.*

---

**Description**

Simulated network.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
data(Net)
str(Net)
```

---

**network**

*A example of an inferred network (4 groups case).*

---

**Description**

This dataset is a network example with 102 nodes, 4 times and 4 groups.

**Format**

The format is: chr "network"

**Details**

A network class object [package "Patterns"] with 6 slots.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
data(network)
str(network)
plot(network)
```
Description

The "network" class

Objects from the Class

Objects can be created by calls of the form `new("network", ...)`.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```r
showClass("network")
```

network2gp

A example of an inferred cascade network (2 groups case).

Description

This dataset is a cascade network example with 53 nodes, 4 times and 2 groups.

Format

The format is: chr "network2gp"

Details

A network class object [package "Patterns"] with 6 slots.

Examples

```r
data(network2gp)
str(network2gp)
plot(network2gp)
```
**networkCascade**

*A example of an inferred cascade network (4 groups case).*

**Description**

This dataset is a cascade network example with 102 nodes, 4 times and 4 groups.

**Format**

The format is: chr "networkCascade"

**Details**

A network class object [package "Patterns"] with 6 slots.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
data(networkCascade)
str(networkCascade)
plot(networkCascade)
```

---

**network_random**

*Generates a network.*

**Description**

Generates a network.

**Usage**

```r
network_random(
    nb,
    time_label,
    exp,
    init,
    regul,
    min_expr,
    max_expr,
    casc_level
)
```
**Arguments**

- **nb**  
  Integer. The number of genes.

- **time_label**  
  Vector. The time points measurements.

- **exp**  
  The exponential parameter, as in the `barabasi.game` function in `igraph` package.

- **init**  
  The attractiveness of the vertices with no adjacent edges. See `barabasi.game` function.

- **regul**  
  A vector mapping each gene with its number of regulators.

- **min_expr**  
  Minimum of strength of a non-zero link

- **max_expr**  
  Maximum of strength of a non-zero link

- **casc.level**  
  ...  

**Value**

A network object.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
set.seed(1)
Net<-network_random(
  nb=100,
  time_label=rep(1:4,each=25),
  exp=1,
  init=1,
  regul=round(rexp(100,1))+1,
  min_expr=0.1,
  max_expr=2,
  casc.level=0.4 )
plot(Net)
```

---

**Net_inf_PL**  
Reverse-engineered network of the M and Net simulated data.

**Description**

The reverse-engineered network with the ‘Patterns’ package using the `fitfun="LASSO"` default function and a cascade network setting.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.
Examples

```r
data(Net_inf_PL)
str(Net_inf_PL)
```

---

### plot-methods

**Plot**

#### Description

Considering the class of the argument which is passed to `plot`, the graphical output differs.

#### Usage

```r
## S4 method for signature 'micro_array,ANY'
plot(x, y, ...)

## S4 method for signature 'network,ANY'
plot(
  x,
  y,
  choice = "network",
  nv = 0,
  gr = NULL,
  ini = NULL,
  color.vertex = NULL,
  color.edge = NULL,
  video = TRUE,
  weight.node = NULL,
  ani = FALSE,
  size = c(2000, 1000),
  label_v = 1:dim(x@network)[1],
  horiz = TRUE,
  legend.position = "topleft",
  frame.color = "black",
  label.hub = FALSE,
  nround = 2,
  ani.img.name = "Rplot",
  ani.imgdir = "images",
  ani.htmlfile = "index.html",
  outdir,
  ani.group.legend = "Cluster",
  layout = ini,
  alpha = 1,
  pixmap.color = terrain.colors(20),
  ...
```

## S4 method for signature 'micropredict,ANY'

```r
plot(
  x,
  time = NULL,
  label_v = NULL,
  frame.color = "white",
  ini = NULL,
  label.hub = FALSE,
  edge.arrow.size = 0.7,
  edge.thickness = 1
)
```

### Arguments

- `x`: a micro\_array object, a network object or a micropredict object
- `y`: optional and not used if x is an appropriate structure
- `...`: additional parameters
- `choice`: what graphic should be plotted: either "F" (for a representation of the matrices F) or "network".
- `nv`: the level of cutoff. Default to ‘0’.
- `gr`: a vector giving the group of each gene
- `ini`: using the “position” function, you can fix the position of the nodes.
- `color.vertex`: a vector defining the color of the vertex.
- `color.edge`: color of the edges
- `video`: if ani is TRUE and video is TRUE, the result of the animation is saved as an animated GIF.
- `weight.node`: nodes weighting. Defaults to ‘NULL’.
- `ani`: animated plot?
- `size`: vector giving the size of the plot. Default to ‘c(2000,1000)’.
- `label_v`: vector defining the vertex labels.
- `horiz`: landscape? Defaults to ‘TRUE’.
- `legend.position`: position of the legend.
- `frame.color`: color of the frames.
- `label.hub`: logical ; if TRUE only the hubs are labeled.
- `nround`: number of digits to display
- `ani.img.name`: name of image file for animations
- `ani.imgdir`: name of the image directory for animations
- `ani.htmlfile`: name of the html file for animations
- `outdir`: name of the outdir for animations
ani.group.legend
  legend for animations
layout
  layout of the graphs
alpha
  transparency of the graphs
pixmap.color
  color for pixmap graphs
time
  sets the time for plot of the prediction. Defaults to ‘NULL’
edge.arrow.size
  size of the arrows ; default to 0.7.
edge.thickness
  edge thickness ; default to 1.

Methods

`list("signature(x = "micro_array", y = "ANY",...)")` x a micro_array object

`list_nv` a vector of cutoff at which the network should be shown

`list("signature(x = "network", y = "ANY",...)")` x a network object

`list()` Optionnal arguments:

  gr a vector giving the group of each gene
  choice what graphic should be plotted: either "F" (for a representation of the matrices F) or "network".
  nv the level of cutoff. Default to 0.
  ini using the “position” function, you can fix the position of the nodes
  color.vertex a vector defining the color of the vertex
  ani vector giving the size of the plot. Default to c(2000,1000). The animation can only be created in the working directory. See the help page of the animation method.
  video if ani is TRUE and video is TRUE, the animation result is a GIF video
  label_v vector defining the vertex labels
  legend.position position of the legend
  frame.color color of the frames
  label.hub logical ; if TRUE only the hubs are labeled
  edge.arrow.size size of the arrows ; default to 0.7
  edge.thickness edge thickness ; default to 1.

`list("signature(x = "micropredict", y = "ANY",...)")` x a micropredict object

`list()` Optionnal arguments: see plot for network

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```r
if(require(CascadeData)){
  data(micro_US, package="CascadeData")
  micro_US<-as.micro_array(micro_US[1:100,],time=c(60,90,210,390),subject=6)
  plot(micro_US)
}
```
plotF

Plot functions for the F matrices.

Description

The graphical output will differ according to the option used.

Usage

plotF(x, choice = "Fshape", nround = 2, pixmap.color = terrain.colors(20))

Arguments

x

The F matrix.

choice

A string: either "F", "Fpixmap", "Fshape", or "Fshapepixmap"

nround

An integer. For numerical F matrices only. The number of decimal numbers to display.

pixmap.color

For pixmap plots.

Value

Nothing.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

# For numerical/inferred F matrices
plotF(CascadeFinit(4,4),choice="F", nround=1)
plotF(CascadeFinit(4,4),choice="Fpixmap")

# For theoretical F matrices
plotF(CascadeFshape(4,4),choice="Fshape")
plotF(CascadeFshape(4,4),choice="Fshapepixmap")
position-methods

Returns the position of edges in the network

Description

Returns the position of edges in the network. Retrieve network position for consistent plotting. Utility function to plot networks.

Usage

```r
## S4 method for signature 'network'
position(net, nv = 0)
```

Arguments

- `net`: a network object
- `nv`: the level of cutoff at which the analysis should be done

Value

Matrix with as many rows as the number of edges of network and three columns (name, xcoord, ycoord).

Methods

- `list("signature(net = \"network\")")` Returns a matrix with the position of the node. This matrix can then be used as an argument in the plot function.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```r
data(network)
position(network)
```
Description
Prediction of the gene expressions after a knock-out experience for cascade networks.

Usage
## S4 method for signature 'micro_array'
predict(
  object,
  Omega,
  act_time_group = NULL,
  nv = 0,
  targets = NULL,
  adapt = TRUE
)

Arguments
object a micro_array object.
Omega a network object.
act_time_group [NULL] vector; at which time the groups (defined by sort(unique(group))) are activated ?
nv [=0] numeric ; the level of the cutoff
targets [NULL] vector ; which genes are knocked out ?
adapt [TRUE] boolean; do not raise an error if used with vectors

Details
The plot of prediction of knock down experiments (i.e. targets<>NULL) is still in beta testing for the moment.

Author(s)
Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(Selection)
data(infos)
pbst_NR4A1 = infos[hgnc_symbol=="NR4A1", "affy_hg_u133_plus_2"]
pbst_EGR1 = infos[infos$hgnc_symbol=="EGR1", "affy_hg_u133_plus_2"]
gene_IDs = infos[match(Selection@name, infos$affy_hg_u133_plus_), "hgnc_symbol"]

data(networkCascade)
# A nv value can be chosen using the cutoff function
nv = .02
NR4A1<-which(is.element(Selection@name,pbst_NR4A1))
EGR1<-which(is.element(Selection@name,pbst_EGR1))
P<-position(networkCascade,nv=nv)

# We predict gene expression modulations within the network if NR4A1 is experimentaly knocked-out.
prediction_ko5_NR4A1<-predict(Selection,networkCascade,nv=nv,targets=NR4A1,act_time_group=1:4)

# Then we plot the results. Here for example we see changes at time points t2, t3 ans t4:
plot(prediction_ko5_NR4A1,time=2:4,ini=P,label_v=gene_IDs)

# We predict gene expression modulations within the network if EGR1 is experimentaly knocked-out.
prediction_ko5_EGR1<-predict(Selection,networkCascade,nv=nv,targets=EGR1,act_time_group=1:4)

# Then we plot the results. Here for example we see changes at time point t2, t3 ans t4:
plot(prediction_ko5_EGR1,time=2:4,ini=P,label_v=gene_IDs)

---

print-methods

Print methods

Description

Methods for generic function print

Usage

## S4 method for signature 'micro_array'
print(x, ...)

Arguments

x an object of class micro-array or network
... additional parameters

Methods

list("signature(x = \"ANY\")")
list("signature(x = \"micro_array\")") Print an object of class micro_array
list("signature(x = \"network\")") Print an object of class network

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.
probeMerge, micro_array-method

Function to merge probesets

Description

Used to collapse probesets using the collapseRows function of the WGCNA package.

Usage

## S4 method for signature 'micro_array'
probeMerge(x, ...)

Arguments

- **x**: Microarray
- **...**: Additionnal parameters to the collapseRows function of the WGCNA package.

Value

Formal class 'micro_array' [package "Patterns"] with 7 slots.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```r
if(require(CascadeData)){
data(micro_S)
D<-as.micro_array(micro_S[1:2000,],1:4,6)
D@gene_ID<-jetset::scores.hgu133plus2[D@name,"EntrezID"]
PM <- probeMerge(D)
}
```

replaceBand

Replace matrix values by band.

Description

F matrices utility function.

Usage

replaceBand(a, b, k)
replaceDown

Arguments

- `a` The matrix to be replaced
- `b` The matrix with the replacement values
- `k` The extend of the replacement: 0 (diagonal only), 1 (diagonal and first extra diagonal), in general an entry is replaced if \(|\text{row}(a) - \text{col}(a)| \leq k\)

Value

A matrix (same size as `a`)

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```r
a=matrix(1:9,3,3)
b=matrix(0,3,3)
replaceBand(a,b,0)
replaceBand(a,b,1)
replaceBand(a,b,2)
```

```r
f
Replace matrix values triangular lower part and by band for the upper part.
```

Description

F matrices utility function.

Usage

`replaceDown(a, b, k)`

Arguments

- `a` The matrix to be replaced
- `b` The matrix with the replacement values
- `k` The extend of the replacement: 0 (lower part and diagonal only), 1 (lower part and first extra diagonal), in general an entry is replaced if \((-\text{row}(a) - \text{col}(a)) \leq k\)

Value

A matrix (same size as `a`)
**replaceUp**

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
a = matrix(1:9,3,3)
b = matrix(1,3,3)
replaceDown(a,b,0)
replaceDown(a,b,1)
replaceDown(a,b,2)
```

---

**replaceUp** Replace matrix values triangular upper part and by band for the lower part.

**Description**

F matrices utility function.

**Usage**

```r
replaceUp(a, b, k)
```

**Arguments**

- `a` The matrix to be replaced
- `b` The matrix with the replacement values
- `k` The extend of the replacement: 0 (upper part only), 1 (upper part and first extra diagonal), in general an entry is replaced if (row(a) - col(a)) <= k

**Value**

A matrix (same size as `a`)

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
a = matrix(1:9,3,3)
b = matrix(1,3,3)
replaceUp(a,b,0)
replaceUp(a,b,1)
replaceUp(a,b,2)
```
Selection

Selection of genes.

Description

20 (at most) genes with differential expression at t1, 20 (at most) genes with differential expression at t2, 20 (at most) genes with differential expression at t3, 20 (at most) genes with differential expression at t4 et 20 (at most) genes with global differential expression were selected.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

data(Selection)
head(Selection)
summary(Selection,3)

Description

Methods for function summary

Usage

## S4 method for signature 'micro_array'
summary(object, nb.graph = NULL, ...)

Arguments

object an object of class micro-array
nb.graph (optionnal) choose the graph to plot. Displays all graphs by default.
... additional parameters.

Methods

list("signature(object = \"ANY\")")
list("signature(object = \"micro_array\")") method here~~

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.
unionMicro-methods

*Makes the union between two micro_array objects.*

**Description**

Makes the union between two micro_array objects.

**Usage**

```r
## S4 method for signature 'micro_array,micro_array'
unionMicro(M1, M2)
```

**Arguments**

- `M1`: a micro-array or a list of micro-arrays
- `M2`: a micro-array or nothing if `M1` is a list of micro-arrays

**Methods**

- `list("signature(M1 = "micro_array", M2 = "micro_array")")`: Returns a micro_array object which is the union of `M1` and `M2`.
- `list("signature(M1 = "list", M2 = "ANY")")`: Returns a micro_array object which is the union of the elements of `M1`.

**Author(s)**

Bertrand Frederic, Myriam Maumy-Bertrand.

**Examples**

```r
if(require(CascadeData)){
  data(micro_S, package = "CascadeData")
  # Create another microarray object with 100 genes
  Mbis <- as.micro_array(micro_S[1:100,], 1:4, 6)
  # Rename the 100 genes
  Mbis@name <- paste(M@name, "bis")
  rownames(Mbis@microarray) <- Mbis@name
  # Union (merge without duplicated names) of the two microarrays.
  str(unionMicro(M, Mbis))
}
```
unsupervised_clustering, micro_array, numeric, numeric-method

Cluster a micro_array object: performs the clustering.

Description

Based on soft clustering performed by the Mfuzz package.

Usage

## S4 method for signature 'micro_array, numeric, numeric'
unsupervised_clustering(
  M1,
  clust,
  mestim,
  M2 = NULL,
  data_log = TRUE,
  screen = NULL,
  heatmap = TRUE,
  new.window = TRUE
)

Arguments

M1 Object of micro_array class.
clust Number of clusters.
mestim Fuzzification parameter.
M2 [NULL] Object of micro_array class,
data_log [TRUE] Should data be logged?
heatmap [TRUE] Plot heatmaps?
new.window [TRUE] Use new window?

Value

An object of class micro_array with the group slot updated by groups deduced from the soft clustering result.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.
Examples

if(require(CascadeData)){
  data(micro_S, package="CascadeData")
  M<-.as.micro_array(micro_S[51:100,],1:4,6)
  mc<-unsupervised_clustering_auto_m_c(M)
  MwithGrp=unsupervised_clustering(M, 4, mc$m, screen=NULL, heatmap=FALSE, new.window = FALSE)
  # Other options
  unsupervised_clustering(M, 4, mc$m, screen=c(2,2), heatmap=TRUE, new.window = FALSE)
  # Plot the clusters
  plot(MwithGrp)
}

unsupervised_clustering_auto_m_c,micro_array-method

Cluster a micro_array object: determine optimal fuzzification parameter and number of clusters.

Description

Based on soft clustering performed by the Mfuzz package.

Usage

## S4 method for signature 'micro_array'
unsupervised_clustering_auto_m_c(
  M1,
  clust = NULL,
  mestim = NULL,
  M2 = NULL,
  data_log = TRUE,
  screen = NULL,
  crange = NULL,
  repeats = NULL,
  cselect = TRUE,
  dminimum = FALSE
)

Arguments

M1 Object of micro_array class.
clust [NULL] Number of clusters.
mestim [NULL] Fuzzification parameter.
M2 [NULL] Object of micro_array class,
data_log [TRUE] Should data be logged?
crange  [NULL] Specify ‘crange’ parameter.
repeats [NULL] Specify ‘repeats’ parameter.
cselect [TRUE] Estimate ‘cselect’ parameter.

Value

m Estimate of the optimal fuzzification parameter.
c Estimate of the optimal number of clusters.
csearch More result from the cselection function of the Mfuzz package

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

if(require(CascadeData)){
  data(micro_S, package="CascadeData")
  M<-as.micro_array(micro_S[1:100,],1:4,6)
  mc<-unsupervised_clustering_auto_m_c(M)
}


Index

* classes
  micro_array-class, 27
  micropredict-class, 27
  network-class, 29
* cluster
  clustExploration,micro_array-method, 8
  clustInference,micro_array,numeric-method, 9
  unsupervised_clustering,micro_array,numeric-method, 44
  unsupervised_clustering_auto_m_c,micro_array-method, 45
* datasets
  CLL, 7
  infos, 26
  M, 26
  Net, 28
  Net_inf_PL, 31
  network, 28
  network2gp, 29
  networkCascade, 30
  Selection, 42
* dplots
  position-methods, 36
* dplot
  plotF, 35
* manip
  probeMerge,micro_array-method, 39
  replaceBand, 39
  replaceDown, 40
  replaceUp, 41
* methods
  analyze_network,network-method, 4
  cutoff,network-method, 11
  dim, 12
  evolution,network-method, 12
  geneNeighborhood,network-method, 14
  geneSelection, 16
  head,micro_array-method, 20
  inference, 23
  plot-methods, 32
  position-methods, 36
  predict,micro_array-method, 37
  print-methods, 38
  summary-methods, 42
  unionMicro-methods, 43
* models
  CascadeFinit, 6
  CascadeFshape, 6
  IndicFinit, 21
  IndicFshape, 22
* package
  Patterns-package, 3

analyze_network
  (analyze_network,network-method), 4
analyze_network,network-method, 4
analyze_network-methods
  (analyze_network,network-method), 4
as.micro_array, 4

CascadeFinit, 6
CascadeFshape, 6
CLL, 7
clustExploration
  (clustExploration,micro_array-method), 8
clustExploration,micro_array-method, 8
clustExploration-methods
  (clustExploration,micro_array-method), 8
clustInference
  (clustInference,micro_array,numeric-method), 9
INDEX

clustInference, micro_array, numeric-method, 9
clustInference-methods
  (clustInference, micro_array, numeric-method), 9
compare (compare-methods), 10
compare, network, network, numeric-method
  (compare-methods), 10
compare-methods, 10
cutoff (cutoff, network-method), 11
cutoff, network-method, 11
cutoff-methods (cutoff, network-method), 11
dim, 12
dim, micro_array-method (dim), 12
dim-methods (dim), 12
evolution (evolution, network-method), 12
evolution, network-method, 12
evolution-methods
  (evolution, network-method), 12
gene_expr_simulation
  (gene_expr_simulation, network-method), 19
gene_expr_simulation, network-method, 19
gene_expr_simulation-methods
  (gene_expr_simulation, network-method), 19
geneNeighborhood
  (geneNeighborhood, network-method), 14
geneNeighborhood, network-method, 14
geneNeighborhood-methods
  (geneNeighborhood, network-method), 14
genePeakSelection (geneSelection), 16
genePeakSelection, micro_array, numeric-method
  (geneSelection), 16
genePeakSelection-methods
  (geneSelection), 16
geneSelection, 16
geneSelection, list, list, numeric-method
  (geneSelection), 16
geneSelection, micro_array, micro_array, numeric-method
  (geneSelection), 16
geneSelection, micro_array, numeric-method
  (geneSelection), 16
head, ANY-method
  (head, micro_array-method), 20
head, micro_array-method, 20
head-methods (head, micro_array-method), 20
IndicFinit, 21
Indicfshape, 22
inference, 23
inference, micro_array-method
  (inference), 23
inference-methods (inference), 23
infos, 26
M, 26
methods (head, micro_array-method), 20
micro_array-class, 27
micropredict-class, 27
Net, 28
Net_inf_PL, 31
network, 28
network-class, 29
network2ggp, 29
network_random, 30
networkCascade, 30
Patterns (Patterns-package), 3
Patterns-package, 3
plot, micro_array, ANY-method
  (plot-methods), 32
plot, micropredict, ANY-method
  (plot-methods), 32
plot, network, ANY-method (plot-methods), 32
plot-methods, 32
plotF, 35
position (position-methods), 36
position, network-method
  (position-methods), 36
position-methods, 36
predict (predict, micro_array-method), 37
predict, ANY-method
predict.macro_array-method, 37
predict-methods
  (predict.macro_array-method), 37
print, ANY-method (print-methods), 38
print, macro_array-method
  (print-methods), 38
print, network-method (print-methods), 38
print-methods, 38
probeMerge
  (probeMerge, macro_array-method), 39
probeMerge, macro_array-method, 39
replaceBand, 39
replaceDown, 40
replaceUp, 41
Selection, 42
summary, ANY-method (summary-methods), 42
summary, macro_array-method
  (summary-methods), 42
summary-methods, 42
unionMicro (unionMicro-methods), 43
unionMicro, list, ANY-method
  (unionMicro-methods), 43
unionMicro, macro_array, macro_array-method
  (unionMicro-methods), 43
unionMicro-methods, 43
unsupervised_clustering
  (unsupervised_clustering, macro_array, numeric, numeric-method), 44
unsupervised_clustering, macro_array, numeric, numeric-method, 44
unsupervised_clustering_auto_m_c
  (unsupervised_clustering_auto_m_c, macro_array-method), 45
unsupervised_clustering_auto_m_c, macro_array-method, 45