Package 'Patterns'

April 24, 2019

Type Package

Title Deciphering Biological Networks with Patterned Heterogeneous Measurements

Version 0.5

Date 2019-04-21

Depends R (>= 2.10)

Imports abind, animation, Biobase, c060, cluster, elasticnet, glmnet, gplots, graphics, grDevices, grid, igraph, jetset, KernSmooth, lars, lattice, limma, magic, methods, Mfuzz, movMF, msgps, nnls, plotrix, SelectBoost, splines, spls, stats4, survival, tnet, VGAM, WGCNA

Suggests R.rsp, CascadeData, knitr

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. 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.

Encoding UTF-8

Collate global.R micro_array.R network.R micro_array-network.R micropredict.R

Classification/MSC 62J05, 62J07, 62J99, 92C42

VignetteBuilder R.rsp

RoxygenNote 6.1.1

URL http://www-irma.u-strasbg.fr/~fbertran/, https://github.com/fbertran/Patterns

BugReports https://github.com/fbertran/Patterns/issues

NeedsCompilation no

R topics documented:

Patterns-package *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.

Details

Package:	Patterns
Type:	Package
Version:	1.0
Date:	2019-04-20
License:	GNU 2.0

Author(s)

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

analyze_network *Analysing the network*

Description

Calculates some indicators for each node in the network.

Usage

```
analyze_network(Omega,nv,...)
```

Arguments

Omega	a network object
nv	the level of cutoff at which the analysis should be done
...	Optional arguments

Value

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

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
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

```
as.micro_array(M, time, subject, name_probe = NULL, gene_ID = NULL)
```

Arguments

M	A matrix. Contains the microarray measurements. Should 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 gene IDs of the row names of the micro array.

Value

A micro_array object.

Author(s)

Bertrand Frederic, Myriam Maumy-Bertrand.

Examples

```
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).

Examples

```
CascadeFinit(3,2)
CascadeFinit(4,3)
CascadeFinit(3,2,low.trig=FALSE)
CascadeFinit(4,3,low.trig=FALSE)
```

`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).

Examples

```
CascadeFshape(4,3)
CascadeFshape(4,3)
```

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.

Usage

```
data("CLL")
```

Format

The format is: chr "CLL"

Details

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.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE39411

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. Proceedings of the National Academy of Sciences of the United States of America, 110(2), 459–464.

Examples

```
data(CLL)
str(CLL)
```

clustExploration *A function to explore a dataset and cluster its rows.*

Description

Based on soft clustering performed by the Mfuzz package.

Usage

```
clustExploration(microarray,...)
```

Arguments

microarray	A microarray to cluster
...	Optional arguments:
	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).

Examples

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

clustInference *A function to explore a dataset and cluster its rows.*

Description

Based on soft clustering performed by the Mfuzz package.

Usage

```
clustInference(microarray, vote.index, ...)
```

Arguments

microarray	A microarray to cluster
vote.index	Option for cluster attribution
...	Optional arguments:
	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

Examples

```
  if(require(CascadeData)){
data(micro_S, package="CascadeData")
D<-as.micro_array(micro_S[1:100,],1:4,6)
b<-clustInference(D,0.5)
}
```

`compare` *Some basic criteria of comparison between actual and inferred network.*

Description

Allows comparison between actual and inferred network.

Usage

```
compare(Net, Net_inf, nv)
```

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 : sensibility, predictive positive value, and the F-score

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
#See vignette
```

cutoff *Choose the best cutoff*

Description

Allows estimating the best cutoff. For a sequence of cutoff, the p value corresponding

Usage

```
cutoff(Omega,...)
```

Arguments

Omega	a network object
...	Optional arguments:
	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)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

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

dim *Dimension of the data*

Description

Dimension of the data

Methods

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

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

Description

See the evolution of the network with change of cutoff

Usage

```
evolution(net,list_nv,...)
```

Arguments

`net`	a network object
`list_nv`	a vector of cutoff at which the network should be shown
`...`	Optionnal arguments:
	gr a vector giving the group of each gene
	color.vertex a vector giving the color of each node
	fix logical, should the position of the node in the network be calculated once at the beginning ? Defaut to TRUE.
	taille vector giving the size of the plot. Default to c(2000,1000)

Value

A HTML page with the evolution of the network.

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

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

`geneNeighborhood` *Find the neighborhood of a set of nodes.*

Description

Find the neighborhood of a set of nodes.

Usage

```
geneNeighborhood(net,targets,...)
```

Arguments

`net`	a network object
`targets`	a vector containing the set of nodes
`...`	Optional arguments. See plot options.

Value

The neighborhood of the targeted genes.

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
#See vignette
```

genePeakSelection *Methods for selecting genes*

Description

Selection of differentially expressed genes.

Usage

```
geneSelection(x,y,tot.number,...)
genePeakSelection(x,peak,...)
```

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 throught 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 diffenrtially genes that will be selected. If 0<tot.number<1, tot.number represents the proportion of diffenrentially genes that are selected.
peak	interger. At which time points measurements should the genes be selected [optionnal for geneSelection].
...	Optional arguments: **M2** a micro_array object. The unstimulated measurements. **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]. **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)

Nicolas Jung, Frédéric Bertrand , Myriam Maumy-Bertrand.

References

Jung, N., Bertrand, F., Bahram, S., Vallat, L., and Maumy-Bertrand, M. (2014). Cascade: a R-package to study, predict and simulate the diffusion of a signal through a temporal gene network. *Bioinformatics*, btt705.

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

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)))
#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` *Simulates microarray data based on a given network.*

Description

Simulates microarray data based on a given network.

Usage

```
gene_expr_simulation(network,...)
```

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.

Value

A micro_array object.

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
#See vignette
```

---

head *Overview of a micro_array object*

Description

Overview of a micro_array object.

Methods

`signature(x = "ANY")` Gives an overview.

`signature(x = "micro_array")` Gives an overview.

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

```
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).

Examples

```
IndicFinit(3, 2, matrix(1,2,2))
```

`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

```
IndicFshape(sqF, ngrp, 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).

Examples

```
IndicFshape(3, 2, matrix(1,2,2))
```

`inference` *Reverse-engineer the network*

Description

Reverse-engineer the network.

Usage

```
inference(M,...)
```

Arguments

M	a micro_array object.
...	Optional arguments:
	tour.max=30 maximal number of steps.
	g=function(x) 1/x the new solution is choosen as (the old solution + g(x) * the new solution)/(1+g(x)) where x is the number of steps.
	conv=10e-3 convergence criterion.
	cv.subjects=TRUE should the cross validation be done removing the subject one by one ?
	nb.folds=NULL Relevant only if cv.subjects is FALSE. The number of folds in cross validation.
	eps=10e-5 machine zero
	type.inf="iterative" "iterative" or "noniterative" : should the algorithm be computed iteratively

Value

A network object.

Author(s)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
#data(micro_US)
#inference(micro_US)
#See vignette for more details
```

---

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), ensembl_gene_id, entrezgene, hgnc_symbol, chromosome_name, start_position, end_position and band.

Usage

```
data("infos")
```

Format

The format is: chr "infos"

Details

Data.frame with 8859 rows and 8 variables.

Examples

```
data(infos)
```

---

micropredict-class *Class* "micropred"

Description

2254

Objects from the Class

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

Examples

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

---

micro_array-class *Class* "micro_array"

Description

The Class

Objects from the Class

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

Slots

`microarray`: Object of class "matrix" ~~

`name`: Object of class "vector" ~~

`gene_ID`: Object of class "vector" ~~

`group`: Object of class "vector" ~~

`start_time`: Object of class "vector" ~~

`time`: Object of class "vector" ~~

`subject`: Object of class "numeric" ~~

Methods

dim `signature(x = "micro_array")`: ...

genePeakSelection `signature(M1 = "micro_array", M2 = "micro_array", peak = "numeric")`: ...

geneSelection `signature(M1 = "micro_array", M2 = "micro_array", tot.number = "numeric")`: ...

head `signature(x = "micro_array")`: ...

inference `signature(M = "micro_array")`: ...

plot `signature(x = "micro_array", y = "ANY")`: ...

plot `signature(x = "network", y = "micro_array")`: ...

predict `signature(object = "micro_array")`: ...

print `signature(x = "micro_array")`: ...

summary `signature(object = "micro_array")`: ...

unionMicro `signature(M1 = "micro_array", M2 = "micro_array")`: ...

Examples

```
showClass("micro_array")
```

---

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.

Usage

```
data("network")
```

Format

The format is: chr "network"

Details

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

Examples

```
data(network)
str(network)
plot(network)
```

network-class *Class* "network"

Description

2254

Objects from the Class

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

Slots

network: Object of class "matrix" ~~

name: Object of class "vector" ~~

F: Object of class "array" ~~

convF: Object of class "matrix" ~~

convO: Object of class "vector" ~~

time_pt: Object of class "vector" ~~

Methods

analyze_network signature(Omega = "network"): ...

cutoff signature(Omega = "network"): ...

evolution signature(net = "network"): ...

geneNeighborhood signature(net = "network"): ...

plot signature(x = "network", y = "ANY"): ...

plot signature(x = "network", y = "micro_array"): ...

position signature(net = "network"): ...

print signature(x = "network"): ...

Examples

```
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.

Usage

```
data("network2gp")
```

Format

The format is: chr "network2gp"

Details

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

Examples

```
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.

Usage

```
data("networkCascade")
```

Format

The format is: chr "networkCascade"

Details

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

Examples

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

`network_random` *Generates a network.*

Description

Generates a network.

Usage

```
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)

Nicolas Jung, Bertrand Frederic, Myriam Maumy-Bertrand.

References

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., ... & Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2), 459-464.

Examples

```
#See vignette
```

plot-methods *Plot*

Description

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

Methods

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

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

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

... 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. Defaut 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)

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.

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

... Optionnal arguments: see plot for network

Examples

```
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)
}
```

position *Retrieve network position for consistent plotting.*

Description

Utility function to plot networks.

Usage

```
position(net, ...)
```

Arguments

net	Network
...	Additionnal parameters

Value

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

Examples

```
data(network)
position(network)
```

---

position-methods *Returns the position of edges in the network*

Description

Returns the position of edges in the network

Methods

`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.

---

predict *Methods for Function* predict

Description

Prediction of the gene expressions after a knock-out experience

Usage

```
predict(object,...)
```

Arguments

object	a micro_array object
...	Other arguments:
	Omega a netowork object.
	nv [=0] numeric ; the level of the cutoff
	targets [NULL] vector ; which genes are knocked out ?

Examples

```
data(Selection)
data(network)
#A nv value can chosen using the cutoff function
nv=.11
EGR1<-which(match(Selection@gene_ID,"EGR1")==1)
P<-position(network,nv=nv)

#We predict gene expression modulations within the network if EGR1 is experimentaly knocked-out.
prediction_ko5<-predict(Selection,network@network,nv=nv,targets=EGR1)

#Then we plot the results. Here for example we see changes at time point t2:
plot(prediction_ko5,time=2,ini=P,label_v=Selection@name)
```

---

print-methods *~~ Methods for Function* print *~~*

Description

~~ Methods for function print ~~

Methods

signature(x = "ANY")

signature(x = "micro_array")

signature(x = "network")

---

probeMerge *Function to merge probesets*

Description

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

Usage

```
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

Examples

```
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)
```

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 abs(row(a) - col(a)) <= k

Value

A matrix (same size as a)

Examples

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

`replaceDown` *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 -(row(a) - col(a)) <= k

Value

A matrix (same size as a)

Examples

```
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

```
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)

Examples

```
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.

Usage

```
data(Selection)
```

Examples

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

summary-methods *~~ Methods for Function* summary *~~*

Description

~~ Methods for function summary ~~

Methods

```
signature(object = "ANY")
signature(object = "micro_array")
```

unionMicro-methods *Makes the union between two micro_array objects.*

Description

Makes the union between two micro_array objects.

Methods

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

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

Examples

```
if(require(CascadeData)){
data(micro_S, package="CascadeData")
#Create another microarray object with 100 genes
Mbis<-M<-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

Cluster a micro_array object: performs the clustering.

Description

Based on soft clustering performed by the Mfuzz package.

Usage

```
unsupervised_clustering(M1, clust, mestim, ...)
```

Arguments

M1	Object of micro_array class.
clust	Number of clusters.
mestim	Fuzzification parameter.
...	Additional parameters.

Value

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

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)
MwithGrp=unsupervised_clustering(M, 8, mc$m, screen=NULL, heatmap=FALSE)
# Other options
unsupervised_clustering(M, 8, mc$m, screen=c(3,3), heatmap=TRUE)
plot(MwithGrp)
}
```

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

Description

Based on soft clustering performed by the Mfuzz package.

Usage

```
unsupervised_clustering_auto_m_c(M1, ...)
```

Arguments

`M1`	Object of micro_array class.
`...`	Additional parameters.

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

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