Package 'Cascade'

November 28, 2022

Type Package

Title Selection, Reverse-Engineering and Prediction in Cascade Networks

Version 2.1

Date 2022-11-28

Depends R (>= 3.5.0)

biocViews

Imports abind, animation, cluster, grid, igraph, lars, lattice, limma, magic, methods, nnls, splines, stats4, survival, tnet, VGAM

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`>), Laurent Vallat [ctb], Nicolas Jung [ctb]

Maintainer Frederic Bertrand <`frederic.bertrand@utt.fr`>

Description A modeling tool allowing gene selection, reverse engineering, and prediction in cascade networks. Jung, N., Bertrand, F., Bahram, S., Vallat, L., and Maumy-Bertrand, M. (2014) <doi:10.1093/bioinformatics/btt705>.

License GPL (>= 2)

Encoding UTF-8

Collate Cascade-package.R global.R micro_array.R network.R micro_array-network.R micropredict.R datasets.R

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

VignetteBuilder R.rsp

RoxygenNote 7.2.1

URL `https://fbertran.github.io/Cascade/`, `https://github.com/fbertran/Cascade/`

BugReports `https://github.com/fbertran/Cascade/issues/`

NeedsCompilation no

Repository CRAN
Date/Publication 2022-11-28 12:30:06 UTC

R topics documented:

`Cascade-package` *The Cascade Package: Selection, Reverse-Engineering and Prediction in Cascade Networks*

Description

A modeling tool allowing gene selection, reverse engineering, and prediction in cascade networks. Jung, N., Bertrand, F., Bahram, S., Vallat, L., and Maumy-Bertrand, M. (2014) <doi:10.1093/bioinformatics/btt705>.

Author(s)

This package has been written by Frédéric Bertrand, Myriam Maumy-Bertrand and Nicolas Jung with biological insights from Laurent Vallat. Maintainer: Frédéric Bertrand <frederic.bertrand@math.unistra.fr>

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.

---

analyze_network,network-method

Analysing the network

Description

Calculates some indicators for each node in the network.

Usage

```
## 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`	(optionnal) the name of the genes

Value

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

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

```
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)
```

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 individuals. This matrix should be created using cbind(M1,M2,...) with M1 a N*T matrix with the measurements for individual 1, M2 a N*T matrix with the measurements for individual 2.
`time`	A vector. The time points measurements.
`subject`	The number of subjects.

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)
}
```

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

Description

Allows comparison between actual and inferred network.

Usage

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

Methods

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

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

```
data(Net)
data(Net_inf)

#Comparing true and inferred networks
F_score=NULL

#Here are the cutoff level tested
test.seq<-seq(0,max(abs(Net_inf@network*0.9)),length.out=200)
for(u in test.seq){
F_score<-rbind(F_score,Cascade::compare(Net,Net_inf,u))
}
matplot(test.seq,F_score,type="l",ylab="criterion value",xlab="cutoff level",lwd=2)
```

`cutoff,network-method` *Choose the best cutoff*

Description

Allows estimating the best cutoff, in function of the scale-freeness of the network. For a sequence of cutoff, the corresponding p-value is then calculated.

Usage

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

Arguments

`Omega`	a network object
`sequence`	(optional) a vector corresponding to the sequence of cutoffs that will be tested.
`x_min`	(optional) 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, 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

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

---

dim *Dimension of the data*

Description

Dimension of the data

Usage

```
## 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.

Examples

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

evolution,network-method

See the evolution of the network with change of cutoff

Description

See the evolution of the network with change of cutoff. This function may be usefull to see if the global topology is changed while increasing the cutoff.

Usage

```
## S4 method for signature 'network'
evolution(
  net,
  list_nv,
  gr = NULL,
  color.vertex = NULL,
  fix = TRUE,
  gif = TRUE,
  taille = c(2000, 1000),
  label_v = 1:dim(net@network)[1],
  legend.position = "topleft",
  frame.color = "black",
  label.hub = FALSE
)
```

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
`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 ? Defaults to TRUE.
`gif`	logical, TRUE
`taille`	vector giving the size of the plot. Default to c(2000,1000)
`label_v`	(optional) the name of the genes
`legend.position`	(optional) the position of the legend, defaults to "topleft"
`frame.color`	(optional) the color of the frame, defaults to "black"
`label.hub`	(optional) boolean, defaults to FALSE

Value

A HTML page with the evolution of the network.

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

```
data(network)
sequence<-seq(0,0.2,length.out=20)
#setwd("inst/animation")
#evolution(network,sequence)
```

---

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,
  ini = NULL,
  frame.color = "white",
  label.hub = FALSE,
  graph = TRUE,
  names = FALSE
)
```

Arguments

`net`	a network object
`targets`	a vector containing the set of nodes
`nv`	the level of cutoff. Defaut to 0.
`order`	of the neighborhood. Defaut 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)

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

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

geneNeighborhood(network,targets=EGR1,nv=nv,ini=P,
label_v=network@name)
```

geneSelection *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
)

## 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 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.
`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 [optionnal 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.
`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,network-method

Simulates microarray data based on a given network.

Description

Simulates microarray data based on a given network.

Usage

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

Arguments

`network`	A network object.
`time_label`	a vector containing the time labels.
`subject`	the number of subjects
`level_peak`	the mean level of peaks.

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

```
data(Net)
set.seed(1)

#We simulate gene expression according to the network Net
Msim<-gene_expr_simulation(
network=Net,
time_label=rep(1:4,each=25),
subject=5,
level_peak=200)
head(Msim)
```

head,micro_array-method

Overview of a micro_array object

Description

Overview of a micro_array object.

Usage

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

Arguments

`x`	an object of class ‘micro_array‘.
`...`	additional parameters

Methods

list("signature(x = \"ANY\")") Gives an overview.

list("signature(x = \"micro_array\")") Gives an overview.

Examples

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

inference,micro_array-method

Reverse-engineer the network

Description

Reverse-engineer the network.

Usage

```
## 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"
)
```

Arguments

`M`	a micro_array object.
`tour.max`	maximal number of steps. Defaults to ‘tour.max=30‘
`g`	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`	convergence criterion. Defaults to ‘conv=10e-3‘
`cv.subjects`	should the cross validation be done removing the subject one by one ? Defaults to ‘cv.subjects=TRUE‘.
`nb.folds`	Relevant only if cv.subjects is FALSE. The number of folds in cross validation. Defaults to ‘NULL‘.
`eps`	machine zero. Defaults to ‘10e-5‘.
`type.inf`	"iterative" or "noniterative" : should the algorithm be computed iteratively. Defaults to ‘"iterative"‘.

Value

A network 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

```
#With simulated data
data(M)
infM <- inference(M)
str(infM)

#With selection of genes from GSE39411
data(Selection)
infSel <- inference(Selection)
str(infSel)
```

`M` *Simulated M data for examples.*

Description

Simulated M microarray.

Examples

```
data(M)
head(M)
```

`micropredict-class` *Class* `"micropredict"`

Description

The `"micropredict"` class

Objects from the Class

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

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", ...)`.

Examples

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

`Net` *Simulated network data for examples.*

Description

Simulated network.

Examples

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

`network` *A network object data.*

Description

A network object. It is the same as the result in the vignette for the inference of the network.

Examples

```
data(network)
plot(network)
print(network)
```

`network-class` *Class* `"network"`

Description

The `"network"` class

Objects from the Class

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

Examples

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

`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, 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

```
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` *Reverse-engineered network of the simulated data.*

Description

The reverse-engineered network of the simulated data (M and Net).

Examples

```
data(Net_inf)
str(Net_inf)
```

---

plot-methods *Plot*

Description

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

Usage

```
## 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,
  video = TRUE,
  weight.node = NULL,
  ani = FALSE,
  taille = c(2000, 1000),
  label_v = 1:dim(x@network)[1],
  horiz = TRUE,
  legend.position = "topleft",
  frame.color = "black",
  label.hub = FALSE,
  ...
)

## S4 method for signature 'micropredict,ANY'
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. Defaut 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.
`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?
`taille`	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.
`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. 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 animated plot?

size 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.

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

list() Optionnal arguments: see plot for network

Examples

```
data(Net)
plot(Net)

data(M)
plot(M)

data(Selection)
data(network)
nv<-0.11
plot(network,choice="network",gr=Selection@group,nv=nv,label_v=Selection@name,
edge.arrow.size=0.9,edge.thickness=1.5)
```

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

Description

Returns the position of edges in the network

Usage

```
## 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

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.

Examples

```
data(Net)
position(Net)
```

`predict,micro_array-method`
Prediction of the gene expressions after a knock-out experience `predict`

Description

Prediction of the gene expressions after a knock-out experience

Usage

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

Arguments

`object`	a micro_array object
`Omega`	a network object.
`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 instead of one column matrices.

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

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

Usage

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

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

Arguments

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

Examples

```
data(Net)
print(Net)

data(M)
print(M)
```

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.

Examples

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

summary-methods *Methods for Function* `summary`

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.

Examples

```
data(M)
summary(M)
```

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

Description

Makes the union between two micro_array objects.

Usage

```
## 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.

Examples

```
data(M)
#Create another microarray object with 100 genes
Mbis<-M
#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))
```

Index