Package 'bnem'

January 23, 2026

Type Package

Title Training of logical models from indirect measurements of perturbation experiments

Version 1.19.0

Description bnem combines the use of indirect measurements of Nested Effects Models (package mnem) with the Boolean networks of CellNOptR. Perturbation experiments of signalling nodes in cells are analysed for their effect on the global gene expression profile. Those profiles give evidence for the Boolean regulation of down-stream nodes in the network, e.g., whether two parents activate their child independently (OR-gate) or jointly (AND-gate).

Depends R (>= 4.1)

License GPL-3

Encoding UTF-8

biocViews Pathways, SystemsBiology, NetworkInference, Network, GeneExpression, GeneRegulation, Preprocessing

Imports CellNOptR, matrixStats, snowfall, Rgraphviz, cluster, flexclust, stats, RColorBrewer, epiNEM, mnem, Biobase, methods, utils, graphics, graph, affy, binom, limma, sva, vsn, rmarkdown

VignetteBuilder knitr

Suggests knitr, BiocGenerics, MatrixGenerics, BiocStyle, RUnit

BugReports https://github.com/MartinFXP/bnem/issues

URL https://github.com/MartinFXP/bnem/

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/bnem

git_branch devel

git_last_commit c656bcb

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-01-23

Author Martin Pirkl [aut, cre]

Maintainer Martin Pirkl <martinpirkl@yahoo.de>

Contents

`absorption` *Absorption*

Description

applies absorption law to a disjuncitve normal form

Usage

```
absorption(bString, model = NULL)
```

Arguments

`bString`	a disjunctive normal form or binary vector according to model
`model`	Model object including the search space, if available. See CellNOptR::preprocessing.

Value

bString after absorption law

Author(s)

Martin Pirkl

Examples

```
graph <- c("A+B=C", "A=C")
absorption(graph)
```

`absorptionII` *Inverse absorption*

Description

applies "inverse" absorption law to a disjuncitve normal form

Usage

```
absorptionII(bString, model = NULL)
```

Arguments

`bString`	a disjunctive normal form or binary vector according to model
`model`	Model object including the search space, if available. See CellNOptR::preprocessing.

Value

bString after "inverse" absorption law

Author(s)

Martin Pirkl

Examples

```
graph <- c("A+B=C", "A=C")
absorptionII(graph)
```

addNoise *Add noise*

Description

Adds noise to simulated data

Usage

```
addNoise(sim, sd = 1)
```

Arguments

sim	bnemsim object from simBoolGtn
sd	standard deviation for the rnorm function

Value

noisy fold-change matrix

Author(s)

Martin Pirkl

Examples

```
sim <- simBoolGtn(Sgenes = 10, maxEdges = 10, negation=0.1,layer=1)
fc <- addNoise(sim,sd=1)
```

bcr *B-Cell receptor signalling perturbations*

Description

Processed data from experiments with a stimulated B-Cell receptor (bcr) and perturbed signalling genes. The raw data is available at https://www.ncbi.nlm.nih.gov/geo/ with accession id GSE68761. For the process steps we refer to the publication Martin Pirkl, Elisabeth Hand, Dieter Kube, Rainer Spang, Analyzing synergistic and non-synergistic interactions in signalling pathways using Boolean Nested Effect Models, Bioinformatics, Volume 32, Issue 6, 15 March 2016, Pages 893–900, https://doi.org/10.1093/bioinform
Alternatively see also the function processDataBCR for details and for reproduction.

Usage

```
bcr
```

References

Martin Pirkl, Elisabeth Hand, Dieter Kube, Rainer Spang, Analyzing synergistic and non-synergistic interactions in signalling pathways using Boolean Nested Effect Models, Bioinformatics, Volume 32, Issue 6, 15 March 2016, Pages 893–900, https://doi.org/10.1093/bioinformatics/btv680

Examples

```
data(bcr)
```

---

bnem *Boolean Nested Effects Model main function*

Description

This function takes a prior network and normalized perturbation data as input and trains logical functions on that prior network

Usage

```
bnem(
  search = "greedy",
  fc = NULL,
  expression = NULL,
  egenes = NULL,
  pkn = NULL,
  design = NULL,
  stimuli = NULL,
  inhibitors = NULL,
  signals = NULL,
  CNOlist = NULL,
  model = NULL,
  sizeFac = 10^-10,
  NAFac = 1,
  parameters = list(cutOffs = c(0, 1, 0), scoring = c(0.1, 0.2, 0.9)),
  parallel = NULL,
  method = "cosine",
  relFit = FALSE,
  verbose = TRUE,
  reduce = TRUE,
  parallel2 = 1,
  initBstring = NULL,
  popSize = 100,
  pMutation = 0.5,
  maxTime = Inf,
  maxGens = Inf,
  stallGenMax = 10,
  relTol = 0.01,
```

```
  priorBitString = NULL,
  selPress = c(1.2, 1e-04),
  fit = "linear",
  targetBstring = "none",
  elitism = NULL,
  inversion = NULL,
  selection = c("t"),
  type = "SOCK",
  exhaustive = FALSE,
  delcyc = FALSE,
  seeds = 1,
  maxSteps = Inf,
  node = NULL,
  absorpII = TRUE,
  draw = TRUE,
  prior = NULL,
  maxInputsPerGate = 2
)
```

Arguments

search	Type of search heuristic. Either "greedy", "genetic" or "exhaustive". "greedy" uses a greedy algorithm to move through the local neighbourhood of a initial hyper-graph. "genetic" uses a genetic algorithm. "exhaustive" searches through the complete search space and is not recommended.
fc	m x l matrix of foldchanges of gene expression values or equivalent input (normalized pvalues, logodds, ...) for m E-genes and l contrasts. If left NULL, the gene expression data is used to calculate naive foldchanges.
expression	Optional normalized m x l matrix of gene expression data for m E-genes and l experiments.
egenes	list object; each list entry is named after an S-gene and contains the names of egenes which are potential children
pkn	Prior knowledge network as output by CellNOptR::readSIF.
design	Optional n x l design matrix with n S-genes and l experiments. If available. If kept NULL, bnem needs either stimuli, inhibitors or a CNOlist object.
stimuli	Character vector of stimuli names.
inhibitors	Character vector of inhibitors.
signals	Optional character vector of signals. Signals are S-genes, which can directly regulate E-genes. If left NULL, all stimuli and inhibitors are defined as signals.
CNOlist	CNOlist object (see package CellNOptR), if available.
model	Model object including the search space, if available. See CellNOptR::preprocessing.
sizeFac	Size factor penelizing the hyper-graph size.
NAFac	factor penelizing NAs in the data.

parameters	parameters for discrete case (not recommended); has to be a list with entries cutOffs and scoring: cutOffs = c(a,b,c) with a (cutoff for real zeros), b (cutoff for real effects), c = -1 for normal scoring, c between 0 and 1 for keeping only relevant between -1 and 0 for keeping only a specific quantile of E-genes, and c > 1 for keeping the top c E-genes; scoring = c(a,b,c) with a (weight for real effects), c (weight for real zeros), b (multiplicator for effects/zeros between a and c);
parallel	Parallelize the search. An integer value specifies the number of threads on the local machine or a list object as in list(c(1,2,3), c("machine1", "machine2", "machine3")) specifies the threads distributed on different machines (local or others).
method	Scoring method can be "cosine", a correlation, or a distance measure. See ?cor and ?dist for details.
relFit	if TRUE a relative fit for each E-gene is computed (not recommended)
verbose	TRUE for verbose output
reduce	if TRUE reduces the search space for exhaustive search
parallel2	if TRUE parallelises the starts and not the search itself
initBstring	Binary vector for the initial hyper-graph.
popSize	Population size (only "genetic").
pMutation	Probability between 0 and 1 for mutation (only "genetic").
maxTime	Define a maximal time (seconds) for the search.
maxGens	Maximal number of generations (only "genetic").
stallGenMax	Maximum number of stall generations (only "genetic").
relTol	Score tolerance for networks defined as optimal but with a lower score as the real optimum (only "genetic").
priorBitString	Binary vector defining hyper-edges which are added to every hyper-graph. E.g. if you know hyper-edge 55 is definitly there and to fix that, set priorBitString[55] <- 1 (only "genetic").
selPress	Selection pressure between 1 and 2 (if fit="linear") and greater 2 (for fit "nonlinear") for the stochastic universal sampling (only "genetic").
fit	"linear" or "nonlinear fit for stochastic universal sampling
targetBstring	define a binary vector representing a network; if this network is found, the computation stops
elitism	Number of best hyper-graphs transferred to the next generation (only "genetic").
inversion	Number of worst hyper-graphs for which their binary strings are inversed (only "genetic").
selection	"t" for tournament selection and "s" for stochastic universal sampling (only "genetic").
type	type of the paralellisation on multpile machines (default: "SOCK")
exhaustive	If TRUE an exhaustive search is conducted if the genetic algorithm would take longer (only "genetic").
delcyc	If TRUE deletes cycles in all hyper-graphs (not recommended).

seeds	how many starts for the greedy search? (default: 1); uses the n-dimensional cube (n = number of S-genes) to maximize search space coverage
maxSteps	Maximal number of steps (only "greedy").
node	vector of S-gene names, which are used in the greedy search; if node = NULL all nodes are considered
absorpII	Use inverse absorption (default: TRUE).
draw	If TRUE draws the network evolution.
prior	Binary vector. A 1 specifies hyper-edges which should not be optimized (only "greedy").
maxInputsPerGate	If no model is supplied, one is created with maxInputsPerGate as maximum number of parents for each hyper-edge.

Value

List object including the optimized hyper-graph, its corresponding binary vector for full hyper-graph and optimized scores.

Author(s)

Martin Pirkl

See Also

nem

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1,
maxInhibit = 2, signals = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
initBstring <- rep(0, length(model$reacID))
res <- bnem(search = "greedy", model = model, CNOlist = CNOlist,
fc = fc, pkn = PKN, stimuli = "A", inhibitors = c("B","C","D"),
parallel = NULL, initBstring = initBstring, draw = FALSE, verbose = FALSE,
maxSteps = Inf)
```

bnemBs *Bootstraped Network*

Description

Runs Bootstraps on the data

Usage

```
bnemBs(fc, x = 10, f = 0.5, replace = TRUE, startString = NULL, ...)
```

Arguments

fc	m x l matrix of foldchanges of gene expression values or equivalent input (normalized pvalues, logodds, ...) for m E-genes and l contrasts. If left NULL, the gene expression data is used to calculate naive foldchanges.
x	number of bootstraps
f	percentage to sample, e.g. f = 0.5 samples only 50 the amount of E-genes as the original data
replace	if TRUE classical bootstrap, if FALSE sub-sampling without replacement
startString	matrix with each row being a string denoting a network to start inference several times with a specific network
...	additional parameters for the bnem function

Value

list with the accumulation of edges in x and the number of bootstraps in n

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1,
maxInhibit = 2, signals = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
```

```
initBstring <- rep(0, length(model$reacID))
res <- bnemBs(search = "greedy", model = model, CNOlist = CNOlist,
fc = fc, pkn = PKN, stimuli = "A", inhibitors = c("B","C","D"),
parallel = NULL, initBstring = initBstring, draw = FALSE, verbose = FALSE,
maxSteps = Inf)
```

---

computeFc *Compute differential effects*

Description

computes differential effects given an activation pattern (absolute gene expression or truth table)

Usage

```
computeFc(CNOlist, y)
```

Arguments

CNOlist	CNOlist object (see package CellNOptR), if available.
y	activation pattern according to the annotation in CNOlist

Value

numeric matrix with annotated response scheme

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signals = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
```

convertGraph *Convert normal form*

Description

converts a disjunctive normal form into a conjunctive normal form and vice versa; input graph as disjunctive normal form like that: c("A+B=D", "C=D", "G+F=U", ...); output is the dual element also in disjunctive normal form;

Usage

```
convertGraph(g)
```

Arguments

g	graph in normal form

Value

converted graph normal form

Author(s)

Martin Pirkl

Examples

```
g <- "A+B=C"
g2 <- convertGraph(g)
```

dummyCNOlist *Create dummy CNOlist*

Description

creates a general CNOlist object from meta information

Usage

```
dummyCNOlist(
  stimuli = NULL,
  inhibitors = NULL,
  maxStim = 0,
  maxInhibit = 0,
  signals = NULL
)
```

Arguments

stimuli	Character vector of stimuli names.
inhibitors	Character vector of inhibitors.
maxStim	maximal number of stimulated genes for a single experiment
maxInhibit	maximal number of inhibited genes for a single experiment
signals	Optional character vector of signals. Signals are S-genes, which can directly regulate E-genes. If left NULL, all stimuli and inhibitors are defined as signals.

Value

CNOlist object

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signals = c("A", "B","C","D"))
```

---

epiNEM2Bg *Switch between epiNEM and B-NEM*

Description

Convert epiNEM model into general Boolean graph. Only needed for comparing accuracy of inferred network for bnem and epiNEM.

Usage

```
epiNEM2Bg(t)
```

Arguments

t	full epiNEM model

Value

differential effects pattern

Author(s)

Martin Pirkl

See Also

CreateTopology

Examples

```
topology <- epiNEM::CreateTopology(3, 1, force = TRUE)
topology <- unlist(unique(topology), recursive = FALSE)
extTopology <- epiNEM::ExtendTopology(topology$model, 100)
b <- epiNEM2Bg(extTopology)
```

---

findResiduals *Compute residuals*

Description

calculates residuals (data and optimized network do not match) and visualizes them

Usage

```
findResiduals(
  bString,
  CNOlist,
  model,
  fc = NULL,
  expression = NULL,
  egenes = NULL,
  parameters = list(cutOffs = c(0, 1, 0), scoring = c(0.1, 0.2, 0.9)),
  method = "s",
  sizeFac = 10^-10,
  main = "residuals for decoupled vertices",
  sub = paste0("green residuals are added effects (left positive,",
    " right negative) and red residuals are deleted ", "effects"),
  cut = TRUE,
  parallel = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

bString	Binary vector denoting the network given a model
CNOlist	CNOlist object (see package CellNOptR), if available.
model	Model object including the search space, if available. See CellNOptR::preprocessing.

fc	m x l matrix of foldchanges of gene expression values or equivalent input (normalized pvalues, logodds, ...) for m E-genes and l contrasts. If left NULL, the gene expression data is used to calculate naive foldchanges.
expression	Optional normalized m x l matrix of gene expression data for m E-genes and l experiments.
egenes	list object; each list entry is named after an S-gene and contains the names of egenes which are potential children
parameters	parameters for discrete case (not recommended); has to be a list with entries cutOffs and scoring: cutOffs = c(a,b,c) with a (cutoff for real zeros), b (cutoff for real effects), c = -1 for normal scoring, c between 0 and 1 for keeping only relevant between -1 and 0 for keeping only a specific quantile of E-genes, and c > 1 for keeping the top c E-genes; scoring = c(a,b,c) with a (weight for real effects), c (weight for real zeros), b (multiplicator for effects/zeros between a and c);
method	Scoring method can be "cosine", a correlation, or a distance measure. See ?cor and ?dist for details.
sizeFac	Size factor penelizing the hyper-graph size.
main	Main title of the figure.
sub	Subtitle of the figure.
cut	If TRUE does not visualize experiments/S-genes which do not have any residuals.
parallel	Parallelize the search. An integer value specifies the number of threads on the local machine or a list object as in list(c(1,2,3), c("machine1", "machine2", "machine3")) specifies the threads distributed on different machines (local or others).
verbose	TRUE for verbose output
...	additional parameters for epiNEM::HeatmapOP

Value

numeric matrices indicating experiments and/or genes, where the network and the data disagree

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signal = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
```

```
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
initBstring <- rep(0, length(model$reacID))
res <- bnem(search = "greedy", CNOlist = CNOlist, fc = fc, model = model,
parallel = NULL, initBstring = initBstring, draw = FALSE, verbose = FALSE,
maxSteps = Inf)
rownames(fc) <- seq_len(nrow(fc))
## val <- validateGraph(CNOlist = CNOlist, fc = fc, model = model,
## bString = res$bString, Egenes = 10, Sgene = 4)
residuals <- findResiduals(res$bString, CNOlist, model, fc = fc)
```

`plot.bnem` *plot bnem opbject*

Description

plots the boolen network as disjunctive normal form

Usage

```
## S3 method for class 'bnem'
plot(x, ...)
```

Arguments

`x`	bnemsim object
`...`	further arguments; see function mnem::plotDnf

Value

plot of boolean network

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1,
maxInhibit = 2, signals = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
```

```
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
initBstring <- rep(0, length(model$reacID))
res <- bnem(search = "greedy", model = model, CNOlist = CNOlist,
fc = fc, pkn = PKN, stimuli = "A", inhibitors = c("B","C","D"),
parallel = NULL, initBstring = initBstring, draw = FALSE, verbose = FALSE,
maxSteps = Inf, seeds = 10)
plot(res)
```

`plot.bnemBs` *Plot Bootstrap result*

Description

Shows the result of a Boostrap with either edge frequencies or confidence intervals

Usage

```
## S3 method for class 'bnemBs'
plot(
  x,
  scale = 3,
  shift = 0.1,
  cut = 0.5,
  dec = 2,
  ci = 0,
  cip = 0.95,
  method = "exact",
  ...
)
```

Arguments

`x`	bnemBs object
`scale`	numeric value for scaling the edgewidth
`shift`	numeric value for shifting the edgewidth
`cut`	shows only edges with a fraction larger than cut
`dec`	integer for function round
`ci`	if TRUE shows confidence intervals
`cip`	range for the confidence interval, e.g. 0.95
`method`	method to use for conidence interval computation (see function binom.confint from package binom)
`...`	additional parameters for the function mnem::plotDnf

Value

plot of the network from the bootstrap

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1,
maxInhibit = 2, signals = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
initBstring <- rep(0, length(model$reacID))
res <- bnemBs(search = "greedy", model = model, CNOlist = CNOlist,
fc = fc, pkn = PKN, stimuli = "A", inhibitors = c("B","C","D"),
parallel = NULL, initBstring = initBstring, draw = FALSE, verbose = FALSE,
maxSteps = Inf)
```

---

plot.bnemsim *plot simulation object*

Description

plots the boolen network from a simulation as disjunctive normal form

Usage

```
## S3 method for class 'bnemsim'
plot(x, ...)
```

Arguments

x	bnemsim object
...	further arguments; see function mnem::plotDnf

Value

plot of boolean network

Author(s)

Martin Pirkl

Examples

```
sim <- simBoolGtn()
plot(sim)
```

`processDataBCR` *BCR perturbation reproduction*

Description

Produce the application data from the BCR paper of Pirkl, et al., 2016, Bioinformatics. Raw data is available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE68761

Usage

```
processDataBCR(path = "", combsign = FALSE)
```

Arguments

`path`	path to the CEL.gz/Cel files
`combsign`	if TRUE includes all covariates in ComBat analysis to estimate batch effects.

Value

list with the full foldchanges and epxression matrix, a reduced foldchange matrix and the design matrix for the computations

Author(s)

Martin Pirkl

Examples

```
## Not run:
processDataBCR()

## End(Not run)
data(bcr)
```

randomDnf *sample normal form*

Description

creates a random normal form or hyper-graph

Usage

```
randomDnf(
  vertices = 10,
  negation = TRUE,
  max.edge.size = NULL,
  max.edges = NULL,
  dag = FALSE
)
```

Arguments

vertices	number of vertices
negation	if TRUE, negations (NOT gates) are allowed
max.edge.size	maximal number of inputs per edge
max.edges	maximal number of hyper-edges
dag	if TRUE, graph will be acyclic

Value

random hyper-graph in normal form

Author(s)

Martin Pirkl

Examples

```
g <- randomDnf(10)
```

reduceGraph *Reduce graph*

Description

reduces the size of a graph, if possible, to an equivalent sub-graph

Usage

```
reduceGraph(bString, model, CNOlist)
```

Arguments

bString	binary vector indicating the sub-graph given a model
model	Model object including the search space, if available. See CellNOptR::preprocessing.
CNOlist	CNOlist object (see package CellNOptR), if available.

Value

equivalent sub-graph denoted by a bString

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signal = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
bString <- reduceGraph(rep(1, length(model$reacID)), model, CNOlist)
```

scoreDnf *score a boolean network*

Description

computes the score of a boolean network given the model and data

Usage

```
scoreDnf(
  bString,
  CNOlist,
  fc,
  expression = NULL,
  model,
  method = "cosine",
  sizeFac = 10^-10,
  NAFac = 1,
  parameters = list(cutOffs = c(0, 1, 0), scoring = c(0.25, 0.5, 2)),
  NEMlist = NULL,
  relFit = FALSE,
  verbose = FALSE
)
```

Arguments

bString	binary string denoting the boolean network
CNOlist	CNOlist object (see package CellNOptR), if available.
fc	m x l matrix of foldchanges of gene expression values or equivalent input (normalized pvalues, logodds, ...) for m E-genes and l contrasts. If left NULL, the gene expression data is used to calculate naive foldchanges.
expression	Optional normalized m x l matrix of gene expression data for m E-genes and l experiments.
model	Model object including the search space, if available. See CellNOptR::preprocessing.
method	Scoring method can be "cosine", a correlation, or a distance measure. See ?cor and ?dist for details.
sizeFac	Size factor penelizing the hyper-graph size.
NAFac	factor penelizing NAs in the data.
parameters	parameters for discrete case (not recommended); has to be a list with entries cutOffs and scoring: cutOffs = c(a,b,c) with a (cutoff for real zeros), b (cutoff for real effects), c = -1 for normal scoring, c between 0 and 1 for keeping only relevant between -1 and 0 for keeping only a specific quantile of E-genes, and c > 1 for keeping the top c E-genes; scoring = c(a,b,c) with a (weight for real effects), c (weight for real zeros), b (multiplicator for effects/zeros between a and c);

NEMlist	NEMlist object (optional)
relFit	if TRUE a relative fit for each E-gene is computed (not recommended)
verbose	TRUE for verbose output

Value

numeric value (score)

Author(s)

Martin Pirkl

Examples

```
sim <- simBoolGtn()
scoreDnf(sim$bString, sim$CNOlist, sim$fc, model=sim$model)
```

---

simBoolGtn *Sample random network and simulate data*

Description

Draws a random prior network, samples a ground truth from the full boolean extension and generates data

Usage

```
simBoolGtn(
  Sgenes = 10,
  maxEdges = 25,
  stimGenes = 2,
  layer = 1,
  frac = 0.1,
  maxInDeg = 2,
  dag = TRUE,
  maxSize = 2,
  maxStim = 2,
  maxInhibit = 1,
  Egenes = 10,
  flip = 0.33,
  reps = 1,
  keepsif = FALSE,
  negation = 0.25,
  allstim = FALSE,
  and = 0.25,
  positive = TRUE,
  verbose = FALSE
)
```

Arguments

Sgenes	number of S-genes
maxEdges	number of maximum edges (upper limit) in the DAG
stimGenes	number of stimulated S-genes
layer	scaling factor for the sampling of next Sgene layerof the prior. high (5-10) mean more depth and low (0-2) means more breadth
frac	fraction of hyper-edges in the ground truth (GTN)
maxInDeg	maximum number of incoming hyper-edges
dag	if TRUE, graph will be acyclic
maxSize	maximum number of S-genes in a hyper-edge
maxStim	maximum of stimulated S-genes in an experiment (=data samples)
maxInhibit	maximum number of inhibited S-genes in an experiment (=data samples)
Egenes	number of E-genes per S-gene, e.g. 10 S-genes and 10 E-genes will return 100 E-genes overall
flip	fraction of inhibited E-genes
reps	number of replicates
keepsif	if TRUE does not delete sif file, which encodes the prior network
negation	sample probability for negative or NOT edges
allstim	full network in which all S-genes are also stimulated
and	probability for AND-gates in the GTN
positive	if TRUE, sets all stimulation edges to activation, else samples inhibitory edges by 'negation' probability
verbose	TRUE for verbose output

Value

list with the corresponding prior graph, ground truth network and data

Author(s)

Martin Pirkl

Examples

```
sim <- simBoolGtn()
plot(sim)
```

simulateStatesRecursive *Simulate states*

Description

simulates the activation pattern (truth table) of a hyper-graph and annotated perturbation experiments

Usage

```
simulateStatesRecursive(CNOlist, model, bString, NEMlist = NULL)
```

Arguments

CNOlist	CNOlist object (see package CellNOptR), if available.
model	Model object including the search space, if available. See CellNOptR::preprocessing.
bString	binary vector denoting the sub-graph given model
NEMlist	NEMlist object only for devel

Value

return the truth tables for certain perturbation experiments as a numeric matrix

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signal = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
states <- simulateStatesRecursive(CNOlist, model,
rep(1, length(model$reacID)))
```

transClose *transitive closure*

Description

calculates transitive closure of a hyper-graph

Usage

```
transClose(g, max.iter = NULL, verbose = FALSE)
```

Arguments

g	hyper-graph in normal form
max.iter	maximal iteration till convergence
verbose	TRUE for verbose output

Value

transitive closure in normal form

Author(s)

Martin Pirkl

Examples

```
g <- c("A=B", "B=C")
gclose <- transClose(g)
```

transRed *transitive reduction*

Description

calculates transitive reduction of a hyper-graph in normal form

Usage

```
transRed(g, max.iter = NULL, verbose = FALSE)
```

Arguments

g	hyper-graph in normal form
max.iter	maximal number of iterations till convergence
verbose	TRUE for verbose output

Value

transitive reduction of the hyper-graph in normal form

Author(s)

Martin Pirkl

Examples

```
g <- c("A=B", "A=C", "B=C", "B=D", "!A=D")
gred <- transRed(g)
```

`validateGraph` *validate graph*

Description

plotting the observed differential effects of an effect reporter and the expected differential effects of the regulating signalling gene

Usage

```
validateGraph(
  CNOlist,
  fc = NULL,
  expression = NULL,
  model,
  bString,
  Egenes = 25,
  Sgene = 1,
  parameters = list(cutOffs = c(0, 1, 0), scoring = c(0.1, 0.2, 0.9)),
  plot = TRUE,
  disc = 0,
  affyIds = TRUE,
  relFit = FALSE,
  xrot = 25,
  Rowv = FALSE,
  Colv = FALSE,
  dendrogram = "none",
  soft = TRUE,
  colSideColors = NULL,
  affychip = "hgu133plus2",
  method = "s",
  ranks = FALSE,
  breaks = NULL,
  col = "RdYlGn",
  sizeFac = 10^-10,
```

```
  order = "rank",
  verbose = TRUE,
  ...
)
```

Arguments

CNOlist	CNOlist object (see package CellNOptR), if available.
fc	m x l matrix of foldchanges of gene expression values or equivalent input (normalized pvalues, logodds, ...) for m E-genes and l contrasts. If left NULL, the gene expression data is used to calculate naive foldchanges.
expression	Optional normalized m x l matrix of gene expression data for m E-genes and l experiments.
model	Model object including the search space, if available. See CellNOptR::preprocessing.
bString	Binary string denoting the hyper-graph.
Egenes	Maximal number of visualized E-genes.
Sgene	Integer denoting the S-gene. See colnames(getSignals(CNOlist)[[1]]) to match integer with S-gene name.
parameters	parameters for discrete case (not recommended); has to be a list with entries cutOffs and scoring: cutOffs = c(a,b,c) with a (cutoff for real zeros), b (cutoff for real effects), c = -1 for normal scoring, c between 0 and 1 for keeping only relevant between -1 and 0 for keeping only a specific quantile of E-genes, and c > 1 for keeping the top c E-genes; scoring = c(a,b,c) with a (weight for real effects), c (weight for real zeros), b (multiplicator for effects/zeros between a and c);
plot	Plot the heatmap. If FALSE, only corresponding information is printed.
disc	Discretize the data.
affyIds	Experimental. Turn Affymetrix Ids into HGNC gene symbols.
relFit	if TRUE a relative fit for each E-gene is computed (not recommended)
xrot	See function epiNEM::HeatmapOP
Rowv	See function epiNEM::HeatmapOP
Colv	See function epiNEM::HeatmapOP
dendrogram	See function epiNEM::HeatmapOP
soft	if TRUE, assigns weights to the expected pattern
colSideColors	See function epiNEM::HeatmapOP
affychip	Define Affymetrix chip used to generate the data (optional and experimental).
method	Scoring method can be "cosine", a correlation, or a distance measure. See ?cor and ?dist for details.
ranks	if TRUE, turns data into ranks
breaks	See function epiNEM::HeatmapOP
col	See function epiNEM::HeatmapOP
sizeFac	Size factor penelizing the hyper-graph size.

order	Order by "rank", "name" or "none"
verbose	TRUE for verbose output
...	additional arguments for epiNEM::HeatmapOP

Value

lattice object with matrix information

Author(s)

Martin Pirkl

Examples

```
sifMatrix <- rbind(c("A", 1, "B"), c("A", 1, "C"), c("B", 1, "D"),
c("C", 1, "D"))
temp.file <- tempfile(pattern="interaction",fileext=".sif")
write.table(sifMatrix, file = temp.file, sep = "\t",
row.names = FALSE, col.names = FALSE,
quote = FALSE)
PKN <- CellNOptR::readSIF(temp.file)
CNOlist <- dummyCNOlist("A", c("B","C","D"), maxStim = 1, maxInhibit = 2,
signal = c("A", "B","C","D"))
model <- CellNOptR::preprocessing(CNOlist, PKN, maxInputsPerGate = 100)
expression <- matrix(rnorm(nrow(slot(CNOlist, "cues"))*10), 10,
nrow(slot(CNOlist, "cues")))
fc <- computeFc(CNOlist, expression)
initBstring <- rep(0, length(model$reacID))
res <- bnem(search = "greedy", CNOlist = CNOlist, fc = fc,
model = model, parallel = NULL, initBstring = initBstring, draw = FALSE,
verbose = FALSE, maxSteps = Inf)
rownames(fc) <- seq_len(nrow(fc))
val <- validateGraph(CNOlist = CNOlist, fc = fc, model = model,
bString = res$bString, Egenes = 10, Sgene = 4)
```

Index