Package 'mnem'

October 17, 2020

Type Package

Title Mixture Nested Effects Models

Version 1.4.5

Author Martin Pirkl

Maintainer Martin Pirkl <martin.pirkl@bsse.ethz.ch>

Description Mixture Nested Effects Models (mnem) is an extension of Nested Effects Models and allows for the analysis of single cell perturbation data provided by methods like Perturb-Seq (Dixit et al., 2016) or Crop-Seq (Datlinger et al., 2017). In those experiments each of many cells is perturbed by a knock-down of a specific gene, i.e. several cells are perturbed by a knock-down of gene A, several by a knock-down of gene B, ... and so forth. The observed read-out has to be multi-trait and in the case of the Perturb-/Crop-Seq gene are expression profiles for each cell. mnem uses a mixture model to simultaneously cluster the cell population into k clusters and and infer k networks causally linking the perturbed genes for each cluster. The mixture components are inferred via an expectation maximization algorithm.

Depends R (>= 3.6)

License GPL-3

Encoding UTF-8

LazyData true

biocViews Pathways, SystemsBiology, NetworkInference, Network, RNASeq, PooledScreens, SingleCell, CRISPR, ATACSeq, DNASeq, GeneExpression

RoxygenNote 7.1.1

Imports cluster, graph, Rgraphviz, flexclust, lattice, naturalsort, snowfall, stats4, tsne, methods, graphics, stats, utils, Linnorm, data.table, Rcpp, RcppEigen, matrixStats, grDevices, e1071

LinkingTo Rcpp, RcppEigen

VignetteBuilder knitr

Suggests knitr, devtools, rmarkdown, BiocGenerics, RUnit, epiNEM

NeedsCompilation yes

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

git_branch RELEASE_3_11

git_last_commit 27a4b6c

git_last_commit_date 2020-09-08

Date/Publication 2020-10-16

R topics documented:

`app` *Processed scRNAseq from pooled CRISPR screens*

Description

Example data: mnem results for the Dixit et al., 2016 and Datlinger et al., pooled CRISPR screens. For details see the vignette or function createApp().

Usage

```
app
```

References

Datlinger, P., Rendeiro, A., Schmidl, C., Krausgruber, T., Traxler, P., Klughammer, J., Schuster, L. C., Kuchler, A., Alpar, D., and Bock, C. (2017). Pooled crispr screening with single-cell transcriptome readout. Nature Methods, 14, 297-301.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., Adamson, B., Norman, T. M., Lander, E. S., Weissman, J. S., Friedman, N., and Regev, A. (2016). Perturb-seq: Dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. Cell, 167(7), 1853-1866.e17.

Examples

```
data(app)
```

`bootstrap` *Bootstrap.*

Description

Run bootstrap simulations on the components (phi) of an object of class mnem.

Usage

```
bootstrap(x, size = 1000, p = 1, logtype = 2, complete = FALSE, ...)
```

Arguments

`x`	mnem object
`size`	size of the booststrap simulations
`p`	percentage of samples (e.g. for 100 E-genes p=0.5 means sampling 50)
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`complete`	if TRUE, complete data log likelihood is considered (for very large data sets, e.g. 1000 cells and 1000 E-genes)
`...`	additional parameters for hte nem function

Value

returns bootstrap support for each edge in each component (phi); list of adjacency matrices

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
boot <- bootstrap(result, size = 2)
```

`clustNEM` *Cluster NEM.*

Description

This function clusters the data and performs standard nem on each cluster.

Usage

```
clustNEM(
  data,
  k = 2:10,
  cluster = NULL,
  starts = 1,
  logtype = 2,
  nem = TRUE,
  getprobspars = list(),
  getaffinitypars = list(),
  Rho = NULL,
  ...
)
```

Arguments

`data`	data of log ratios with cells in columns and features in rows
`k`	number of clusters to check
`cluster`	given clustering has to correspond to the columns of data
`starts`	number of random starts for the kmeans algorithm
`logtype`	logarithm type of the data
`nem`	if FALSE only clusters the data
`getprobspars`	list of parameters for the getProbs function
`getaffinitypars`	list of parameters for the getAffinity function
`Rho`	perturbation matrix with dimensions nxl with n S-genes and l samples; either as probabilities with the sum of probabilities for a sample less or equal to 1 or discrete with 1s and 0s
`...`	additional arguments for standard nem function

Value

family of nems; the first k list entries hold full information of the standard nem search

`comp`	list of all adjacency matrices phi
`mw`	vector of mixture weights
`probs`	fake cell probabilities (see mw: mixture weights)

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
resulst <- clustNEM(data, k = 2:3)
```

createApp *Creating app data.*

Description

This function is for the reproduction of the application results in the vignette and publication. See the publication Pirkl & Beerenwinkel (2018) on how to download the data files: GSE92872_CROP-seq_Jurkat_TCR.digital_expression.csv k562_both_filt.txt GSM2396861_k562_ccycle_cbc_gbc_dict.csv GSM2396858_k562_tfs_7_cbc_gbc_dict.csv

Usage

```
createApp(
  sets = seq_len(3),
  m = NULL,
  n = NULL,
  o = NULL,
  maxk = 5,
  parallel = NULL,
  path = "",
  types = c("data", "lods", "mnem"),
  allcrop = FALSE,
  multi = FALSE,
  file = NULL,
  ...
)
```

Arguments

`sets`	numeric vector with the data sets: 1 (CROPseq), 2, 3 (both PERTURBseq); default is all three
`m`	number of Sgenes (for testing)
`n`	number of most variable E-genes (for testing)
`o`	number of samples per S-gene (for testing)
`maxk`	maximum number of component in mnem inference (default: 5)
`parallel`	number of threads for parallelisation
`path`	path to the data files path/file.csv: "path/"
`types`	types of data/analysis; "data" creates the gene expression matrix, "lods" includes the log odds, "mnem" additionally performes the mixture nem analysis; default c("data", "lods", "mnem")
`allcrop`	if TRUE, does not restrict and uses the full CROPseq dataset
`multi`	if TRUE, includes cells with more than one perturbed gene
`file`	path and filename of the rda file with the raw data from the command "data <- createApp(..., types = "data")"
`...`	additional parameters for the mixture nem function

Value

app data object

Author(s)

Martin Pirkl

Examples

```
## recreate the app data object (takes very long, i.e. days)
## Not run:
createApp()

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

---

fitacc *Simulation accuracy.*

Description

Computes the accuracy of the fit between simulated and inferred mixture.

Usage

```
fitacc(x, y, strict = FALSE, unique = TRUE, type = "ham")
```

Arguments

`x`	mnem object
`y`	simulation object or another mnem object
`strict`	if TRUE, accounts for over/underfitting, i.e. the number of components
`unique`	if TRUE, phis of x and y are made unique each (FALSE if strict is TRUE)
`type`	type of accuracy. "ham" for hamming, "sens" for sensitivity and "spec for Specificity"

Value

plot of EM convergence

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
fitacc(result, sim)
fitacc(result, sim, type = "sens")
fitacc(result, sim, type = "spec")
fitacc(result, sim, strict = TRUE, type = "sens")
fitacc(result, sim, strict = TRUE, type = "spec")
```

fuzzyindex *Calculate fuzzy ground truth.*

Description

Calculates responsibilities and mixture weights based on the ground truth and noisy data.

Usage

```
fuzzyindex(x, data, logtype = 2, complete = FALSE, ...)
```

Arguments

`x`	mnemsim object
`data`	noisy data matrix
`logtype`	logarithm type of the data
`complete`	if TRUE, complete data log likelihood is considered (for very large data sets, e.g. 1000 cells and 1000 E-genes)
`...`	additional parameters for the function getAffinity

Value

list with cell log odds mixture weights and log likelihood

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- sim$data
data[which(sim$data == 1)] <- rnorm(sum(sim$data == 1), 1, 1)
data[which(sim$data == 0)] <- rnorm(sum(sim$data == 0), -1, 1)
fuzzy <- fuzzyindex(sim, data)
```

getAffinity *Calculate responsibilities.*

Description

This function calculates the responsibilities of each component for all cells from the expected log distribution of the hidden data.

Usage

```
getAffinity(
  x,
  affinity = 0,
  norm = TRUE,
  logtype = 2,
  mw = NULL,
  data = matrix(0, 2, ncol(x)),
  complete = FALSE
)
```

Arguments

`x`	log odds for l cells and k components as a kxl matrix
`affinity`	0 for standard soft clustering, 1 for hard clustering during inference (not recommended)
`norm`	if TRUE normalises to probabilities (recommended)
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`mw`	mixture weights of the components
`data`	data in log odds
`complete`	if TRUE, complete data log likelihood is considered (for very large data sets, e.g. 1000 cells and 1000 E-genes)

Value

responsibilities as a kxl matrix (k components, l cells)

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
resp <- getAffinity(result$probs, mw = result$mw, data = data)
```

---

getIC *Calculate negative penalized log likelihood.*

Description

This function calculates a negative penalized log likelihood given a object of class mnem. This penalized likelihood is based on the normal likelihood and penalizes complexity of the mixture components (i.e. the networks).

Usage

```
getIC(
  x,
  man = FALSE,
  degree = 4,
  logtype = 2,
  pen = 2,
  useF = FALSE,
  Fnorm = FALSE
)
```

Arguments

`x`	mnem object
`man`	logical. manual data penalty, e.g. man=TRUE and pen=2 for an approximation of the Akaike Information Criterion
`degree`	different degree of penalty for complexity: positive entries of transitively reduced phis or phi^r (degree=0), phi^r and mixture components minus one k-1 (1), phi^r, k-1 and positive entries of thetas (2), positive entries of transitively closed phis or phi^t, k-1 (3), phi^t, theta, k-1 (4, default), all entries of phis, thetas and k-1 (5)
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`pen`	penalty weight for the data (e.g. pen=2 for approximate Akaike Information Criterion)
`useF`	use F (see publication) as complexity instead of phi and theta
`Fnorm`	normalize complexity of F, i.e. if two components have the same entry in F, it is only counted once

Value

penalized log likelihood

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
pen <- numeric(3)
result <- list()
for (k in seq_len(2)) {
    result[[k]] <- mnem(data, k = k, starts = 1)
    pen[k] <- getIC(result[[k]])
}
print(pen)
```

`hamSim` *Accuracy for two phis.*

Description

This function uses the hamming distance to calculate an accuracy for two networks (phi).

Usage

```
hamSim(a, b, diag = 1, symmetric = TRUE)
```

Arguments

`a`	adjacency matrix (phi)
`b`	adjacency matrix (phi)
`diag`	if 1 includes diagonal in distance, if 0 not
`symmetric`	comparing a to b is asymmetrical, if TRUE includes comparison b to a

Value

normalized hamming accuracy for a and b

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
similarity <- hamSim(sim$Nem[[1]], sim$Nem[[2]])
```

`mnem` *Mixture NEMs - main function.*

Description

This function simultaneously learns a mixture of causal networks and clusters of a cell population from single cell perturbation data (e.g. log odds of fold change) with a multi-trait readout. E.g. Pooled CRISPR scRNA-Seq data (Perturb-Seq. Dixit et al., 2016, Crop-Seq. Datlinger et al., 2017).

Usage

```
mnem(
  D,
  inference = "em",
  search = "greedy",
  phi = NULL,
  theta = NULL,
  mw = NULL,
  method = "llr",
  parallel = NULL,
  reduce = FALSE,
  runs = 1,
  starts = 3,
  type = "networks",
  complete = FALSE,
  p = NULL,
  k = NULL,
  kmax = 10,
  verbose = FALSE,
  max_iter = 100,
  parallel2 = NULL,
  converged = -Inf,
  redSpace = NULL,
  affinity = 0,
  evolution = FALSE,
  lambda = 1,
  subtopoX = NULL,
  ratio = TRUE,
  logtype = 2,
  domean = TRUE,
  modulesize = 5,
  compress = FALSE,
  increase = TRUE,
  fpfn = c(0.1, 0.1),
  Rho = NULL,
  ksel = c("kmeans", "silhouette", "cor")
)
```

Arguments

`D`	data with cells indexing the columns and features (E-genes) indexing the rows
`inference`	inference method "em" for expectation maximization
`search`	search method for single network inference "greedy", "exhaustive" or "modules" (also possible: "small", which is greedy with only one edge change per M-step to make for a smooth convergence)
`phi`	a list of n lists of k networks for n starts of the EM and k components
`theta`	a list of n lists of k attachment vector for the E-genes for n starts of the EM and k components
`mw`	mixture weights; if NULL estimated or uniform
`method`	"llr" for log ratios or foldchanges as input (see ratio)

`parallel`	number of threads for parallelization of the number of em runs
`reduce`	logical - reduce search space for exhaustive search to unique networks
`runs`	number of runs for greedy search
`starts`	number of starts for the em
`type`	initialize with responsibilities either by "random", "cluster" (each S-gene is clustered and the different S-gene clustered differently combined for several starts), "cluster2" (clustNEM is used to infer reasonable phis, which are then used as a start for one EM run), "cluster3" (global clustering as a start), or "networks" (initialize with random phis)
`complete`	if TRUE, optimizes the expected complete log likelihood of the model, otherwise the log likelihood of the observed data
`p`	initial probabilities as a k (components) times l (cells) matrix
`k`	number of components
`kmax`	maximum number of components when k=NULL is inferred
`verbose`	verbose output
`max_iter`	maximum iteration, if likelihood does not converge
`parallel2`	if parallel=NULL, number of threads for single component optimization
`converged`	absolute distance for convergence between new and old log likelihood; if set to -Inf, the EM stops if neither the phis nor thetas were changed in the most recent iteration
`redSpace`	space for "exhaustive" search
`affinity`	0 is default for soft clustering, 1 is for hard clustering
`evolution`	logical. If TRUE components are penelized for being different from each other.
`lambda`	smoothness value for the prior put on the components, if evolution set to TRUE
`subtopoX`	hard prior on theta as a vector with entry i equal to j, if E-gene i is attached to S-gene j
`ratio`	logical, if true data is log ratios, if false foldchanges
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`domean`	average the data, when calculating a single NEM (speed improvment)
`modulesize`	max number of S-genes per module in module search
`compress`	compress networks after search (warning: penelized likelihood not interpretable)
`increase`	if set to FALSE, the algorithm will not stop if the likelihood decreases
`fpfn`	numeric vector of length two with false positive and false negative rates for discrete data
`Rho`	perturbation matrix with dimensions nxl with n S-genes and l samples; either as probabilities with the sum of probabilities for a sample less or equal to 1 or discrete with 1s and 0s
`ksel`	character vector of methods for the inference of k; can combine "hc" (hierarchical clustering) or "kmeans" with "silhouette", "BIC" or "AIC"; can also include "cor" for correlation distance (preferred) instead of euclidean

Value

object of class mnem

`comp`	list of the component with each component being a list of the causal network phi and the E-gene attachment theta
`data`	input data matrix
`limits`	list of results for all indpendent searches
`ll`	log likelihood of the best model
`lls`	log likelihood ascent of the best model search
`mw`	vector with mixture weights
`probs`	kxl matrix containing the cell log likelihoods of the model

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
```

`mnemh` *Hierarchical mixture.*

Description

This function does a hierarchical mixture. That means it uses the approximate BIC to check, if there are more than one component. It recursively splits the data if there is evidence for k > 1 components.

Usage

```
mnemh(data, k = 2, logtype = 2, getprobspars = list(), ...)
```

Arguments

`data`	data matrix either binary or log odds
`k`	number of maximal components for each hierarchy leaf
`logtype`	log type of the data
`getprobspars`	list of parameters for the getProbs function
`...`	additional parameters for the mnem function

Value

object of class mnem

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnemh(data, starts = 1, k = 1)
```

---

mnemk — *Learn the number of components K and optimize the mixture.*

Description

High level function for learning the number of components k, if unknown.

Usage

```
mnemk(
  D,
  ks = seq_len(5),
  man = FALSE,
  degree = 4,
  logtype = 2,
  pen = 2,
  useF = FALSE,
  Fnorm = FALSE,
  ...
)
```

Arguments

`D`	data with cells indexing the columns and features (E-genes) indexing the rows
`ks`	vector of number of components k to test
`man`	logical. manual data penalty, e.g. man=TRUE and pen=2 for an approximation of the Akaike Information Criterion
`degree`	different degree of penalty for complexity: positive entries of transitively reduced phis or phi^r (degree=0), phi^r and mixture components minus one k-1 (1), phi^r, k-1 and positive entries of thetas (2), positive entries of transitively closed phis or phi^t, k-1 (3), phi^t, theta, k-1 (4, default), all entries of phis, thetas and k-1 (5)
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`pen`	penalty weight for the data (e.g. pen=2 for approximate Akaike Information Criterion)
`useF`	use F (see publication) as complexity instead of phi and theta
`Fnorm`	normalize complexity of F, i.e. if two components have the same entry in F, it is only counted once
`...`	additional parameters for the mnem main function

Value

list containing the result of the best k as an mnem object and the raw and penalized log likelihoods

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnemk(data, ks = seq_len(2), starts = 1)
```

`plot.bootmnem` *Plot bootstrap mnem result.*

Description

Plot bootstrap mnem result.

Usage

```
## S3 method for class 'bootmnem'
plot(x, reduce = TRUE, ...)
```

Arguments

`x`	bootmnem object
`reduce`	if TRUE transitively reduces the graphs
`...`	additional parameters for the plotting function plotDNF

Value

visualization of bootstrap mnem result with Rgraphviz

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
boot <- bootstrap(result, size = 2)
plot(boot)
```

plot.mnem *Plot mnem result.*

Description

Plot mnem result.

Usage

```
## S3 method for class 'mnem'
plot(
  x,
  oma = c(3, 1, 1, 3),
  main = "M&NEM",
  anno = TRUE,
  cexAnno = 1,
  scale = NULL,
  global = TRUE,
  egenes = TRUE,
  sep = FALSE,
  tsne = FALSE,
  affinity = 0,
  logtype = 2,
  cells = TRUE,
  pch = ".",
  legend = FALSE,
  showdata = FALSE,
  bestCell = TRUE,
  showprobs = FALSE,
  shownull = TRUE,
  ratio = TRUE,
  method = "llr",
  showweights = TRUE,
  ...
)
```

Arguments

`x`	mnem object
`oma`	outer margin
`main`	main text
`anno`	annotate cells by their perturbed gene
`cexAnno`	text size of the cell annotations
`scale`	scale cells to show relative and not absolute distances
`global`	if TRUE clusters all cells, if FALSE clusters cells within a component
`egenes`	show egene attachments, i.e. number of E-genes assigned to each S-gene
`sep`	seperate clusters and not put them on top of each other for better visualization
`tsne`	if TRUE use tsne instead of pca

`affinity`	use hard clustering if TRUE
`logtype`	logarithm type of the data (e.g. 2 for log2 data or exp(1) for natural)
`cells`	show cell attachments, .i.e how many cells are assigned to each S-gene
`pch`	cell symbol
`legend`	show legend
`showdata`	show data if TRUE
`bestCell`	show probability of best fitting cell for each S-gene
`showprobs`	if TRUE, shows responsibilities for all cells and components
`shownull`	if TRUE, shows the null node
`ratio`	use log ratios (TRUE) or foldchanges (FALSE)
`method`	"llr" for ratios
`showweights`	if TRUE, shows mixture weights for all components
`...`	additional parameters

Value

visualization of mnem result with Rgraphviz

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
plot(result)
```

`plot.mnemsim` *Plot simulated mixture.*

Description

Plot simulated mixture.

Usage

```
## S3 method for class 'mnemsim'
plot(x, data = NULL, logtype = 2, fuzzypars = list(), ...)
```

Arguments

`x`	mnemsim object
`data`	noisy data matrix (optional)
`logtype`	logarithm type of the data
`fuzzypars`	list of parameters for the function fuzzyindex
`...`	additional parameters for the plotting function plotDNF

Value

visualization of simulated mixture with Rgraphviz

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
plot(sim)
```

`plotConvergence` *Plot convergence of EM*

Description

This function plots the convergence of the different EM iterations (four figures, e.g. par(mfrow=(2,2))).

Usage

```
plotConvergence(x, col = NULL, type = "b", convergence = 0.1, ...)
```

Arguments

`x`	mnem object
`col`	vector of colors for the iterations
`type`	see ?plot.default
`convergence`	difference of when two log likelihoods are considered equal; see also convergence for the function mnem()
`...`	additional parameters ofr the plots/lines functions

Value

plot of EM convergence

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
data <- (sim$data - 0.5)/0.5
data <- data + rnorm(length(data), 0, 1)
result <- mnem(data, k = 2, starts = 1)
par(mfrow=c(2,2))
plotConvergence(result)
```

`plotDnf` *Plot disjunctive normal form.*

Description

This function visualizes a graph encoded as a disjunctive nromal form.

Usage

```
plotDnf(
  dnf = NULL,
  freq = NULL,
  stimuli = c(),
  signals = c(),
  inhibitors = c(),
  connected = TRUE,
  CNOlist = NULL,
  cex = NULL,
  fontsize = NULL,
  labelsize = NULL,
  type = 2,
  lwd = 1,
  edgelwd = 1,
  legend = 0,
  x = 0,
  y = 0,
  xjust = 0,
  yjust = 0,
  width = 1,
  height = 1,
  layout = "dot",
  main = "",
  sub = "",
  cex.main = 1.5,
  cex.sub = 1,
  col.sub = "grey",
  fontcolor = NULL,
  nodestates = NULL,
  simulate = NULL,
  edgecol = NULL,
  labels = NULL,
  labelcol = "blue",
  nodelabel = NULL,
  nodecol = NULL,
  bordercol = NULL,
  nodeshape = NULL,
  verbose = FALSE,
  edgestyle = NULL,
  nodeheight = NULL,
  nodewidth = NULL,
  edgewidth = NULL,
  lty = NULL,
  hierarchy = NULL,
  showall = FALSE,
  edgehead = NULL,
  edgelabel = NULL,
  edgetail = NULL,
  bool = TRUE,
  draw = TRUE,
  ...
)
```

Arguments

`dnf`	Hyper-graph in disjunctive normal form, e.g. c("A=B", "A=C+D", "E=!B") with the child on the left and the parents on the right of the equation with "A=C+D" for A = C AND D. Alternatively, dnf can be an adjacency matrix, which is converted on the fly to a disjunctive normal form.
`freq`	Frequency of hyper-edges which are placed on the edges.
`stimuli`	Highlights vertices which can be stimulated.
`signals`	Highlights vertices which regulate E-genes.
`inhibitors`	Highlights vertices which can be inhibited.
`connected`	If TRUE, only includes vertices which are connected to other vertices.
`CNOlist`	CNOlist object. Optional instead of stimuli, inhibitors or signals. See package CellNOptR.
`cex`	Global font size.
`fontsize`	Vertice label size.
`labelsize`	Edge label size.
`type`	Different plot types. 2 for Rgraphviz and 1 for graph.
`lwd`	Line width.
`edgelwd`	Edgeline width.
`legend`	0 shows no legend. 1 shows legend as a graph. 2 shows legend in a standard box.
`x`	x coordinate of box legend.
`y`	y coordinate of box legend.
`xjust`	Justification of legend box left, right or center (-1,1,0).
`yjust`	Justification of legend box top, bottom or middle (-1,1,0).
`width`	Vertice width.
`height`	Vertice height.
`layout`	Graph layout. See graphvizCapabilities()$layoutTypes.
`main`	Main title.
`sub`	Subtitle.
`cex.main`	Main title font size.
`cex.sub`	Subtitle font size.
`col.sub`	Font color of subtitle.
`fontcolor`	Global font color.

`nodestates`	Binary state of each vertice.
`simulate`	Simulate stimulation and inhibition of a list of vertices. E.g. simulate = list(stimuli = c("A", "B"), inhibitors = c("C", "D")).
`edgecol`	Vector with colors for every edge of the graph (not hyper-graph). E.g. an AND gate consists of three distinct edges.
`labels`	Vector with labels for the edges.
`labelcol`	Vector with label colors for the edges.
`nodelabel`	List of vertices with labels as input. E.g. labels = list(A="test", B="label for B").
`nodecol`	List of vertices with colors as input.
`bordercol`	List of vertices with colors as input.
`nodeshape`	List of vertices with shapes (diamond, box, square,...).
`verbose`	Verbose output.
`edgestyle`	set the edge style like dashed, can be numerical
`nodeheight`	List of vertices with height as input.
`nodewidth`	List of vertices with width as input.
`edgewidth`	Vector with edge widths.
`lty`	Vector with edge styles (line, dotted,...).
`hierarchy`	List with the hierarchy of the vertices. E.g. list(top = c("A", "B"), bottom = c("C", "D")).
`showall`	See "connected" above.
`edgehead`	Vector with edge heads.
`edgelabel`	Vector with edge labels.
`edgetail`	Vector with edge tails.
`bool`	If TRUE, only shows normal graph and no AND gates.
`draw`	Do not plot the graph and only output the graphNEL object.
`...`	additional arguments

Value

Rgraphviz object

Author(s)

Martin Pirkl

Examples

```
g <- c("!A+B+C=G", "C=G", "!D=G")
plotDnf(g)
```

simData	*Simulate data.*

Description

This function simulates single cell data from a random mixture of networks.

Usage

```
simData(
  Sgenes = 5,
  Egenes = 1,
  Nems = 2,
  reps = NULL,
  mw = NULL,
  evolution = FALSE,
  nCells = 1000,
  uninform = 0,
  unitheta = FALSE,
  edgeprob = 0.25,
  multi = FALSE,
  subsample = 1,
  scalefree = FALSE,
  badCells = 0,
  exactProb = TRUE,
  ...
)
```

Arguments

`Sgenes`	number of Sgenes
`Egenes`	number of Egenes
`Nems`	number of components
`reps`	number of replicates, if set (not realistic for cells)
`mw`	mixture weights (has to be vector of length Nems)
`evolution`	evolving and not purely random network, if set to TRUE
`nCells`	number of cells
`uninform`	number of uninformative Egenes
`unitheta`	uniform theta, if TRUE
`edgeprob`	edge probability, value between 0 and 1 for sparse or dense networks
`multi`	a vector with the percentages of cell with multiple perturbations, e.g. c(0.2,0.1,0) for 20 no quadruple knock-downs
`subsample`	range to subsample data. 1 means the full simulated data is used
`scalefree`	if TRUE, graph is scale free
`badCells`	number of cells, which are just noise and not connected to the ground truth network
`exactProb`	logical; if TRUE generates random network with exact fraction of edges
`...`	additional parameters for the scale free network sampler (see ’nem’ package)

Value

simulation object with meta information and data

`Nem`	list of adjacency matrixes generatign the data
`theta`	E-gene attachaments
`data`	data matrix
`index`	index for which Nem generated which cell (data column)
`mw`	vector of input mixture weights

Author(s)

Martin Pirkl

Examples

```
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4,0.6))
```

`transitive.closure` *Transitive closure of a directed acyclic graph (dag)*

Description

Computes the transitive closure of a dag or only of a deletion/addition of an edge

Usage

```
transitive.closure(g, u = NULL, v = NULL)
```

Arguments

`g`	graph as matrix or graphNEL object
`u`	index of the parent of an edge (optional)
`v`	index of the child of an edge (optional)

Value

transitively closed matrix or graphNEL

Author(s)

Martin Pirkl

Examples

```
g <- matrix(c(0,0,0,1,0,0,0,1,0), 3)
transitive.closure(g)
```

`transitive.reduction` *Transitive reduction*

Description

Computes the transitive reduction of a adjacency matrix or graphNEL object. Originally imported from the package 'nem'.

Usage

```
transitive.reduction(g)
```

Arguments

`g`	adacency matrix or graphNEL object

Author(s)

Holger Froehlich

References

R. Sedgewick, Algorithms, Pearson, 2002.

Examples

```
g <- matrix(c(0,0,0,1,0,0,0,1,0), 3)
rownames(g) <- colnames(g) <- seq_len(3)
g.tr <- transitive.reduction(g)
```

Index