Package ‘mnem’

January 26, 2024

Type Package
Title Mixture Nested Effects Models
Version 1.18.0
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 (>= 4.1)
License GPL-3
Encoding UTF-8
LazyData true
biocViews Pathways, SystemsBiology, NetworkInference, Network, RNASeq, PooledScreens, SingleCell, CRISPR, ATACSeq, DNaseq, GeneExpression
RoxygenNote 7.2.1
Imports cluster, graph, Rgraphviz, flexclust, lattice, naturalsort, snowfall, stats4, tsne, methods, graphics, stats, utils, Linnorm, data.table, Rcpp, RcppEigen, matrixStats, grDevices, e1071, ggplot2, wesanderson
LinkingTo Rcpp, RcppEigen
VignetteBuilder knitr
Suggests knitr, devtools, markdown, BiocGenerics, RUnit, epiNEM, BiocStyle
NeedsCompilation yes
BugReports https://github.com/cbg-ethz/mnem/issues
URL https://github.com/cbg-ethz/mnem/
git_url  https://git.bioconductor.org/packages/mnem

git_branch  RELEASE_3_18

git_last_commit  ad087ee

git_last_commit_date  2023-10-24

Repository  Bioconductor 3.18

Date/Publication  2024-01-26

Author  Martin Pirkl [aut, cre]

Maintainer  Martin Pirkl <martinpirkl@yahoo.de>

R topics documented:

app ................................................................. 3
bootstrap .......................................................... 3
clustNEM ......................................................... 4
createApp ....................................................... 5
fitacc ............................................................ 7
fuzzyindex ....................................................... 8
getAffinity ...................................................... 9
getIC ............................................................ 10
hamSim .......................................................... 11
mnem ............................................................ 12
mnemh ........................................................... 15
mnemk ........................................................... 16
moreboxplot ..................................................... 17
nem ............................................................. 18
plot.bootmnem .................................................. 20
plot.mnem ....................................................... 21
plot.mnem_mcmc ............................................... 23
plot.mnem_sim .................................................. 24
plotConvergence ............................................... 24
plotConvergence.mnem ....................................... 25
plotDnf ........................................................ 26
scoreAdj ....................................................... 29
simData ......................................................... 31
transitive.closure ............................................. 32
transitive.reduction ......................................... 33

Index 34
**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**

```r
app
```

**References**


**Examples**

```r
data(app)
```

---

**bootstrap**

**Bootstrap.**

**Description**

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

**Usage**

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

**Arguments**

- `x`: mnem object
- `size`: size of the bootstrap 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 the nem function
Value

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

Author(s)

Martin Pirkl

Examples

```r
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

```r
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
createApp

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, "lod" includes the log odds, "mnem" additionally performs the mixture nem analysis; default c("data", "lod", "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")"

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)
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, marginal = FALSE, ...)

Arguments

- **x**: mnem_sim 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)
- **marginal**: logical to compute the marginal likelihood (TRUE)
- **...**: additional parameters for the function getAffinity

Value

list with cell log odds mixture weights and log likelihood

Author(s)

Martin Pirkl

Examples

```r
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**

```r
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**

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

*Calculate negative penalized log likelihood.*

**Description**

This function calculates a negative penalized log likelihood given an 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**

```r
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

```r
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**

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

```r
mnem(
  D,
  inference = "em",
  search = "greedy",
  phi = NULL,
  theta = NULL,
  mw = NULL,
  method = "llr",
  marginal = FALSE,
  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,
  dmean = TRUE,
  modulesize = 5,
  compress = FALSE,
  increase = TRUE,
  fpfn = c(0.1, 0.1),
  Rho = NULL,
)```

mnem

k sel = c("kmeans", "silhouette", "cor"),
nullcomp = FALSE,
tree = FALSE,
burnin = 10,
hastings = TRUE,
node switch = TRUE,
post gaps = 10,
penalized = FALSE,
accept range = 1,
...

Arguments

- **D**: data with cells indexing the columns and features (E-genes) indexing the rows
- **inference**: inference method "em" for expectation maximization or "mcmc" for markov chain monte carlo sampling
- **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)
- **marginal**: logical to compute the marginal likelihood (TRUE)
- **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 or mcmc
- **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), inference='mcmc' only supports 'networks' and 'empty' for unconnected networks phi
- **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 iterations (moves for inference='mcmc'. adjust parameter burnin)
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 penalized 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)

domain

average the data, when calculating a single NEM (speed improvement)

modulesize

max number of S-genes per module in module search

compress

compress networks after search (warning: penalized 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 as the first two values "hc" (hierarchical clustering) or "kmeans" with "silhouette", "BIC" or "AIC"; the third value is either "cor" for correlation distance or any method accepted by the function 'dist'

nullcomp

if TRUE, adds a null component (k+1)

tree

if TRUE, restrict inference on trees (MCMC not included)

burnin

number of iterations to be discarded prior to analyzing the posterior distribution of the mcmc

hastings

if set to TRUE, the Hastings ratio is calculated

nodeswitch

if set to TRUE, node switching is allowed as a move, additional to the edge moves

postgaps

can be set to numeric. Determines after how many iterations the next Phi mixture is added to the Phi edge Frequency tracker in the mcmc

penalized

if set to TRUE, the penalized likelihood will be used for the mcmc. Per default this is FALSE, since no component learning is involved and sparsity is hence not enforced

accept_range

the random probability the acceptance probability is compared to (default: 1)

... arguments to function nem
**mnemh**

**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 independent 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**

```r
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)
```

**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

```r
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

```r
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 transitive 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 transitivity 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)

Description  
Plots a boxplots plus x-axis randomised scatter and mirrored densities to visualise a distribution.

Usage  

moreboxplot(
  x,  
  box = TRUE,  
  dens = TRUE,  
  scatter = "no",  
  polygon = TRUE,  
  sd = 0.1,  
  dcol = NULL,  
  scol = NULL,  
  dlty = 1,  
  dlwd = 1,  
  spch = 1,  
  gcol = rgb(0, 0, 0, 0.5),  
  glty = 2,  
  glen = 10,  
  gmin = NA,  
  gmax = NA,  
  ...
)
Arguments

- x: list, matrix or data.frame
- box: if TRUE, draws boxes
- dens: if TRUE, draws densities
- scatter: if set to "random", draws x-axis randomised scatter points
- polygon: if TRUE, fills the densities
- sd: standard deviation of the scatter
- dcol: color of the densities
- scol: color of the scatter points
- dlty: line type of the densities
- dlwd: line width of the densities
- spch: type of scatter points
- gcol: color of the grid
- glty: line type of the grid
- glen: length of the grid
- gmin: minimal point of the grid
- gmax: maximal point of the grid
- ...: optional parameters for boxplot or plot

Value

transitively closed matrix or graphNEL

Author(s)

Martin Pirkl

Examples

D <- matrix(rnorm(100*3), 100, 3)
moreboxplot(D)

---

nem

Implementation of the original NEM

Description

Infers a signalling pathway from perturbation experiments.
Usage

nem(
    D,                  # data matrix with observed genes as rows and knock-down experiments as columns
    search = "greedy", # either "greedy", "modules" or "exhaustive" (not recommended for more than five S-genes)
    start = NULL,      # either NULL ("null") or a specific network to start the greedy
    method = "llr",    # "llr" for log odds or p-values densities or "disc" for binary data
    marginal = FALSE,  # logical to compute the marginal likelihood (TRUE)
    parallel = NULL,   # NULL for no parallel optimization or an integer for the number of threads
    reduce = FALSE,    # reduce search space (TRUE) for exhaustive search
    weights = NULL,    # a numeric vector of weights for the columns of D
    runs = 1,          # the number of runs for the greedy search
    verbose = FALSE,   # for verbose output (TRUE)
    redSpace = NULL,   # reduced search space for exhaustive search; see result of exhaustive search with reduce = TRUE
    trans.close = TRUE, # if TRUE uses the transitive closure of adj
    ...)

Arguments

D                   data matrix with observed genes as rows and knock-down experiments as columns
search              either "greedy", "modules" or "exhaustive" (not recommended for more than five S-genes)
start               either NULL ("null") or a specific network to start the greedy
method              "llr" for log odds or p-values densities or "disc" for binary data
marginal            logical to compute the marginal likelihood (TRUE)
parallel            NULL for no parallel optimization or an integer for the number of threads
reduce              reduce search space (TRUE) for exhaustive search
weights             a numeric vector of weights for the columns of D
runs                the number of runs for the greedy search
verbose             for verbose output (TRUE)
redSpace            reduced search space for exhaustive search; see result of exhaustive search with reduce = TRUE
trans.close         if TRUE uses the transitive closure of adj
subtopo  optional matrix with the subtopology theta as adjacency matrix
prior    a prior network matrix for adj
ratio    if FALSE uses alternative distance for the model score
domain   if TRUE summarizes duplicate columns
modulesize the max number of S-genes included in one module for search = "modules"
fpfn     numeric vector of length two with false positive and false negative rates
Rho      optional perturbation matrix
logtype  log base of the log odds
modified if TRUE, assumes a preprocessed data matrix
tree     if TRUE forces tree; does not allow converging edges
learnRates if TRUE learns rates for false positives/negatives
stepSize numerical step size for learning rates
...     optional parameters for future search methods

Value

transitively closed matrix or graphNEL

Author(s)

Martin Pirkl

Examples

D <- matrix(rnorm(100*3), 100, 3)
colnames(D) <- 1:3
rownames(D) <- 1:100
adj <- diag(3)
colnames(adj) <- rownames(adj) <- 1:3
scoreAdj(D, adj)

plot.bootmnem  Plot bootstrap mnem result.

Description

Plot bootstrap mnem result.

Usage

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

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)

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",
marginal = FALSE,
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
genes show egene attachments, i.e. number of E-genes assigned to each S-gene
sep separate 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
marginal logical to compute the marginal likelihood (TRUE)
showweights if TRUE, shows mixture weights for all components
...
additional parameters

Value

visualization of mnem result with Rgraphviz
Author(s)

Martin Pirkl

Examples

```r
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.mnem_mcmc

Plot mnem_mcmc result.

Description

Plot mnem_mcmc result.

Usage

```r
## S3 method for class 'mnem_mcmc'
plot(x, starts = NULL, burnin = 0, ...)
```

Arguments

- `x`: mnem_mcmc object
- `starts`: restarts of mcmc as used in mnem function
- `burnin`: number of iteration to start from
- `...`: parameters for function ggplot2

Value

Visualization of mcmc result with Rgraphviz

Author(s)

Viktoria Brunner

Examples

```r
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.mnem_sim  
*Plot simulated mixture.*

**Description**
Plot simulated mixture.

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

**Arguments**
- `x`: mnem_sim 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**
```r
sim <- simData(Sgenes = 3, Egenes = 2, Nems = 2, mw = c(0.4, 0.6))
plot(sim)
```

---

plotConvergence  
*Plot convergence of EM*

**Description**
Generic function plotting convergence diagnostics for different methods.

**Usage**
```r
plotConvergence(x, ...)
```
plotConvergence.mnem

Arguments

x  
object with convergence statistics

...  
additional parameters for the specific object type

Value

plot of EM convergence

Author(s)

Martin Pirkl

Examples

```r
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)
```

Description

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

Usage

```r
## S3 method for class 'mnem'
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 for the plots/lines functions

Value

plot of EM convergence
Author(s)
Martin Pirkl

Examples
```r
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)
```

Description
This function visualizes a graph encoded as a disjunctive normal form. See the graphviz documentation for possible input arguments, like edgehead/tail: https://graphviz.org/docs/attr-types/arrowType/

Usage
```r
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,
)```

plotDnf   Plot disjunctive normal form.
plotDnf

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 Vertex label size.
labelsize Edge label size.
type Different plot types. 2 for Rgraphviz and 1 for graph.
lwd Line width of nodeborder.
edgelwd  Global 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  Vertex width.
height  Vertex 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.
lables  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 for individual edges.
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.
scoreAdj

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.
...

Value

Rgraphviz object

Author(s)

Martin Pirkl

Examples

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

scoreAdj

Description

Computes the fit (score of a network) of the data given a network matrix

Usage

scoreAdj(
  D,
  adj,
  method = "llr",
  marginal = FALSE,
  logtype = 2,
  weights = NULL,
  trans.close = TRUE,
  subtopo = NULL,
  prior = NULL,
  ratio = TRUE,
  fpfn = c(0.1, 0.1),
  Rho = NULL,
  dotopo = FALSE,
  P = NULL,
  oldadj = NULL,
  modified = TRUE
)

Arguments

- **D** data matrix; use modified = FALSE
- **adj** adjacency matrix of the network phi
- **method** either llr if D consists of log odds or disc, if D is binary
- **marginal** logical to compute the marginal likelihood (TRUE)
- **logtype** log base of the log odds
- **weights** a numeric vector of weights for the columns of D
- **trans.close** if TRUE uses the transitive closure of adj
- **subtopo** optional matrix with the subtopology theta as adjacency matrix
- **prior** a prior network matrix for adj
- **ratio** if FALSE uses alternative distance for the model score
- **fpfn** numeric vector of length two with false positive and false negative rates
- **Rho** optional perturbation matrix
- **dotopo** if TRUE computes and returns the subtopology theta (optional)
- **P** previous score matrix (only used internally)
- **oldadj** previous adjacency matrix (only used internally)
- **modified** if TRUE, assumes a preprocessed data matrix

Value

transitively closed matrix or graphNEL

Author(s)

Martin Pirkl

Examples

```R
D <- matrix(rnorm(100*3), 100, 3)
colnames(D) <- 1:3
rownames(D) <- 1:100
adj <- diag(3)
colnames(adj) <- rownames(adj) <- 1:3
scoreAdj(D, adj)
```
**simData**

*Simulate data.*

**Description**

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

**Usage**

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

**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 or a range `c(l,u)` with lower and upper bound
- **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
transitive.closure

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 provided by edgeprob
tree if TRUE, restricts dag to a tree
... additional parameters for the scale free network sampler (see 'nem' package)

Value

simulation object with meta information and data

Nem list of adjacency matrixes generating 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)
transitive.reduction

Value
transitively closed matrix or graphNEL

Author(s)
Martin Pirkl

Examples

```r
  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 an adjacency matrix or graphNEL object. Originally imported from the package 'nem'.

Usage

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

Arguments

- `g`: adjacency matrix or graphNEL object

Value
transitively reduced adjacency matrix

Author(s)
Holger Froehlich

References

Examples

```r
  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

app, 3
bootstrap, 3
clustNEM, 4
createApp, 5
fitacc, 7
fuzzyindex, 8
getAffinity, 9
getIC, 10
hamSim, 11
mnem, 12
mnemh, 15
mnemk, 16
moreboxplot, 17
nem, 18
plot.bootmnem, 20
plot.mnem, 21
plot.mnem_mcmc, 23
plot.mnem_sim, 24
plotConvergence, 24
plotConvergence.mnem, 25
plotDnf, 26
scoreAdj, 29
simData, 31
transitive_closure, 32
transitive.reduction, 33