Package 'nempi'

January 3, 2025

Type Package

Title Inferring unobserved perturbations from gene expression data

Version 1.14.0

Depends R (>= 4.1), mnem

Description Takes as input an incomplete perturbation profile and differential gene expression in log odds and infers unobserved perturbations and augments observed ones. The inference is done by iteratively inferring a network from the perturbations and inferring perturbations from the network. The network inference is done by Nested Effects Models.

License GPL-3

Encoding UTF-8

LazyData true

biocViews Software, GeneExpression, DifferentialExpression, DifferentialMethylation, GeneSignaling, Pathways, Network, Classification, NeuralNetwork, NetworkInference, ATACSeq, DNASeq, RNASeq, PooledScreens, CRISPR, SingleCell, SystemsBiology

Imports e1071, nnet, randomForest, naturalsort, graphics, stats, utils, matrixStats, epiNEM

VignetteBuilder knitr

Suggests knitr, BiocGenerics, rmarkdown, RUnit, BiocStyle

BugReports https://github.com/cbg-ethz/nempi/issues

URL https://github.com/cbg-ethz/nempi/

RoxygenNote 7.1.1

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

git_branch RELEASE_3_20

git_last_commit daecb3f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-01-02

Author Martin Pirkl [aut, cre]

Maintainer Martin Pirkl <martinpirkl@yahoo.de>

Contents

`classpi` *Classification*

Description

Builds and uses different classifiers to infer perturbation profiles

Usage

```
classpi(
  D,
  unknown = "",
  full = TRUE,
  method = "svm",
  size = NULL,
  MaxNWts = 10000,
  ...
)
```

Arguments

`D`	either a binary effects matrix or log odds matrix as for Nested Effects Models (see package 'nem')
`unknown`	colname of samples without mutation data, E.g. ""
`full`	if FALSE, does not change the known profiles
`method`	either one of svm, nn, rf
`size`	parameter for neural network (see package 'nnet')
`MaxNWts`	parameters for neural network (see package 'nnet')
`...`	additional parameters for mnem::nem

Value

plot

Author(s)

Martin Pirkl

Examples

```
D <- matrix(rnorm(1000*100), 1000, 100)
colnames(D) <- sample(seq_len(5), 100, replace = TRUE)
Gamma <- matrix(sample(c(0,1), 5*100, replace = TRUE, p = c(0.9, 0.1)), 5,
100)
Gamma <- apply(Gamma, 2, function(x) return(x/sum(x)))
Gamma[is.na(Gamma)] <- 0
rownames(Gamma) <- seq_len(5)
result <- classpi(D)
```

---

nempi *Main function for NEM based perturbation imputation.*

Description

Infers perturbations profiles based on a sparse perturbation matrix and differential gene expression as log odds

Usage

```
nempi(
  D,
  unknown = "",
  Gamma = NULL,
  type = "null",
  full = TRUE,
  verbose = FALSE,
  logtype = 2,
  null = TRUE,
  soft = TRUE,
  combi = 1,
  converged = 0.1,
  complete = TRUE,
  mw = NULL,
  max_iter = 100,
  keepphi = TRUE,
  start = NULL,
  phi = NULL,
  ...
)
```

Arguments

D	either a binary effects matrix or log odds matrix as for Nested Effects Models (see package 'nem')
unknown	colname of samples without mutation data, E.g. ""
Gamma	matrix with expectations of perturbations, e.g. if you have a binary mutation matrix, just normalize the columns to have sum 1
type	"null": does not use the unknown samples for inference at the start, "random" uses them in a random fashion (not recommended)

full	if FALSE, does not change the known profiles
verbose	if TRUE gives more output during inference
logtype	log type for the log odds
null	if FALSE does not use a NULL node for uninformative samples
soft	if FALSE discretizes Gamma during the inference
combi	if combi > 1, uses a more complex algorithm to infer combinatorial perturbations (experimental)
converged	the absolute difference of log likelihood till convergence
complete	if TRUE uses the complete-data logliklihood (recommended for many E-genes)
mw	if NULL infers mixture weights, otherwise keeps them fixed
max_iter	maximum iterations of the EM algorithm
keepphi	if TRUE, uses the previous phi for the next inference, if FALSE always starts with start network (and empty and full)
start	starting network as adjacency matrix
phi	if not NULL uses only this phi and does not infer a new one
...	additional parameters for the nem function (see package mnem, function nem or mnem::nem)

Value

nempi object

Author(s)

Martin Pirkl

Examples

```
D <- matrix(rnorm(1000*100), 1000, 100)
colnames(D) <- sample(seq_len(5), 100, replace = TRUE)
Gamma <- matrix(sample(c(0,1), 5*100, replace = TRUE, p = c(0.9, 0.1)), 5,
100)
Gamma <- apply(Gamma, 2, function(x) return(x/sum(x)))
Gamma[is.na(Gamma)] <- 0
rownames(Gamma) <- seq_len(5)
result <- nempi(D, Gamma = Gamma)
```

---

nempibs *Bootstrapping function*

Description

Bootstrap algorithm to get a more stable result.

Usage

```
nempibs(D, bsruns = 100, bssize = 0.5, replace = TRUE, ...)
```

Arguments

D	either a binary effects matrix or log odds matrix as
bsruns	number of bootstraps
bssize	number of E-genes for each boostrap
replace	if TRUE, actual bootstrap, if False sub-sampling
...	additional parameters for the function nempi

Value

list with aggregate Gamma and aggregate causal network phi

Author(s)

Martin Pirkl

Examples

```
D <- matrix(rnorm(1000*100), 1000, 100)
colnames(D) <- sample(seq_len(5), 100, replace = TRUE)
Gamma <- matrix(sample(c(0,1), 5*100, replace = TRUE, p = c(0.9, 0.1)), 5,
100)
Gamma <- apply(Gamma, 2, function(x) return(x/sum(x)))
Gamma[is.na(Gamma)] <- 0
rownames(Gamma) <- seq_len(5)
result <- nempibs(D, bsruns = 3, Gamma = Gamma)
```

---

pifit *Accuracy computation*

Description

Compares the ground truth of a perturbation profile with the inferred profile

Usage

```
pifit(x, y, D, unknown = "", balanced = FALSE, propagate = TRUE, knowns = NULL)
```

Arguments

x	object of class nempi
y	object of class mnemsim
D	data matrix
unknown	label for the unlabelled samples
balanced	if TRUE, computes balanced accuracy
propagate	if TRUE, propagates the perturbation through the network
knowns	subset of P-genes that are known to be perturbed (the other are neglegted)

Value

list of different accuracy measures: true/false positives/negatives, correlation, area under the precision recall curve, (balanced) accuracy

Author(s)

Martin Pirkl

Examples

```
library(mnem)
seed <- 42
Pgenes <- 10
Egenes <- 10
samples <- 100
uninform <- floor((Pgenes*Egenes)*0.1)
Nems <- mw <- 1
noise <- 1
multi <- c(0.2, 0.1)
set.seed(seed)
simmini <- simData(Sgenes = Pgenes, Egenes = Egenes,
Nems = Nems, mw = mw, nCells = samples,
uninform = uninform, multi = multi,
badCells = floor(samples*0.1))
data <- simmini$data
ones <- which(data == 1)
zeros <- which(data == 0)
data[ones] <- rnorm(length(ones), 1, noise)
data[zeros] <- rnorm(length(zeros), -1, noise)
lost <- sample(1:ncol(data), floor(ncol(data)*0.5))
colnames(data)[lost] <- ""
res <- nempi(data)
fit <- pifit(res, simmini, data)
```

---

plot.nempi *Plotting nempi*

Description

Plot function for an object of class 'nempi'.

Usage

```
## S3 method for class 'nempi'
plot(x, barlist = list(), heatlist = list(), ...)
```

Arguments

x	object of class 'nempi'
barlist	additional arguments for function 'barplot' from package 'graphics'
heatlist	additional arguments for function 'HeatmapOP' from package 'epiNEM'
...	additional arguments for function 'plotDnf' from package 'mnem'

Value

Plots of the optimal network phi and perturbation matrix.

Author(s)

Martin Pirkl

Examples

```
D <- matrix(rnorm(1000*100), 1000, 100)
colnames(D) <- sample(seq_len(5), 100, replace = TRUE)
result <- nempi(D)
plot(result)
```

---

plotConvergence.nempi *Plot convergence of EM*

Description

Produces different convergence plots based on a nempi object

Usage

```
## S3 method for class 'nempi'
plotConvergence(x, type = "b", ...)
```

Arguments

x	nempi object
type	see ?plot.default
...	additional parameters for plot

Value

plot

Author(s)

Martin Pirkl

Examples

```
D <- matrix(rnorm(1000*100), 1000, 100)
colnames(D) <- sample(seq_len(5), 100, replace = TRUE)
Gamma <- matrix(sample(c(0,1), 5*100, replace = TRUE, p = c(0.9, 0.1)), 5,
100)
Gamma <- apply(Gamma, 2, function(x) return(x/sum(x)))
Gamma[is.na(Gamma)] <- 0
rownames(Gamma) <- seq_len(5)
result <- nempi(D, Gamma = Gamma)
par(mfrow=c(2,3))
plotConvergence(result)
```

Index