Package 'PANR'

May 25, 2024

Type Package

Title Posterior association networks and functional modules inferred from rich phenotypes of gene perturbations

Version 1.50.0

Date 2012-06-20

Author Xin Wang <xin_wang@hms.harvard.edu>

Maintainer Xin Wang <xin_wang@hms.harvard.edu>

Imports graphics, grDevices, MASS, methods, pvclust, stats, utils, RedeR

Depends R (>= 2.14), igraph

Suggests snow

Description This package provides S4 classes and methods for inferring functional gene networks with edges encoding posterior beliefs of gene association types and nodes encoding perturbation effects.

License Artistic-2.0

Collate classUnions.R assoScore.R panGraphics.R AllGenerics.R AllClasses.R BetaMixture-methods.R PAN-methods.R

LazyLoad yes

biocViews ImmunoOncology, NetworkInference, Visualization, GraphAndNetwork, Clustering, CellBasedAssays

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

git_branch RELEASE_3_19

git_last_commit 3b8ce2c

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-24

Contents

`assoScore` *Association scores for gene pairs*

Description

This function compute similarity scores to quantify associations between pairs of genes, given measured rich phenotypes.

Usage

```
assoScore(pheno, metric="cosine", upperTri=TRUE, transform=TRUE,
verbose=TRUE, ...)
```

Arguments

`pheno`	a numeric matrix of rich phenotypes with rows and columns specifying samples and genes, respectively.
`metric`	a character value specifying the metric to compute similarity scores. Currently, 'cosine' and 'correlation' are supported (see details for more).
`upperTri`	a logical value specifying whether (if `TRUE`) to take the upper triangular of the similarity matrix or not (if `FALSE`).

transform	a logical value specifying whether to transform (if TRUE) the range of association scores from [-1, 1] to [0, 1] or not (if FALSE).
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	other arguments for function cor

Details

This function aims at quantifying the associations between genes of interest given certain phenotyping measurements (e.g. gene expressions by microarray, cell viabilities, morphological phenotypes, etc.). For the current version of the package, the user can either choose 'cosine' or 'correlation'. When the latter is chosen, additional arguments (e.g. 'method') for the function cor are allowed.

Value

This function will return either a vector (if upperTri=TRUE) or a matrix (if upperTri=FALSE) of association scores for given phenotypes.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Roland F. Schwarz, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

cor, cosineSim

Examples

```
toydata<-matrix(rnorm(n=2000, mean=0, sd=4), nrow=100, ncol=20)
toyasso<-assoScore(t(toydata), "cosine", upperTri=FALSE, transform=FALSE)
##transform to [0, 1]
toyasso01<-assoScore(t(toydata), "cosine", upperTri=FALSE, transform=TRUE)
##transform to [0, 1] and return only the upper triangular
toyasso01upper<-assoScore(t(toydata), "cosine", upperTri=TRUE, transform=
TRUE)
##use spearman correlation
toyassoSp<-assoScore(t(toydata), "correlation", upperTri=FALSE, transform=
FALSE, method="spearman")
```

`BetaMixture-class` *An S4 class for beta mixture modelling of functional gene associations*

Description

This S4 class includes methods to do beta-mixture modelling of functional gene associations given rich phenotyping screens.

Objects from the Class

Objects of class `BetaMixture` can be created from `new("BetaMixture",metric, order, association, model, pheno, partition)` (see the example below for details).

Slots

`pheno`: a numeric matrix of rich phenotypes with rows and columns specifying genes and samples, respectively.

`metric`: a character value specifying the metric to compute similarity scores. Currently, 'cosine' and 'correlation' are supported (see `assoScore` for more details).

`order`: a numeric value specifying the order of the similarity score to be computed. Only 1 and 2 is supported for the current version. The first order (when `order=1`) similarities are used for quatification of the strength of functional associations between genes, whilst the second order (when `code=2`) ones are employed to compute the strength of modularity between genes.

`association`: a numeric vector providing all association scores between genes. This can be useful when `pheno` is not available or the user has a different way to compute functional associations.

`model`: a character value specifying whether the original (if `global`) or extended (if `stratified`) model is used.

`partition`: a numeric of gene partition labels (e.g. c(rep(1, 100), rep(2, 20), rep(3, 80)) is a valid vector of partition labels for a vector of associations falling into three categories of interaction types 1, 2 and 3).

`result`: a list storing results from S4 methods of this class.

`summary`: a list of summary information for available results.

Methods

An overview of methods (More detailed introduction can be found in help for each specific function.):

`permNULL` do permutations for input rich phenotyping screens (`'pheno'`).

`fitNULL` fit the permuted association scores to a beta distribution.

`fitBM` fit the functional association scores computed from input screens to a three-beta mixture model.

`p2SNR` Translate p-values to Signal-to-Noise Ratios.

`SNR2p` Translate Signal-to-Noise Ratios to p-values.

`view` view the fitting results (a histogram of the original data and fitted probability density curves) for NULL and real data.

`summarize` summarize results including input data and parameters, NULL fitting and beta mixture fitting.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

`permNULL` `fitNULL` `fitBM` `view` `summarize`

Examples

```
## Not run:
data(Bakal2007)
bm1<-new("BetaMixture", pheno=Bakal2007, metric="cosine",
model="global", order=1)
bm1<-fitNULL(bm1, nPerm=10, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="all", verbose=TRUE)
bm1<-fitBM(bm1, para=list(zInit=NULL, thetaInit=c(alphaNeg=2, betaNeg=4,
alphaNULL=bm1@result$fitNULL$thetaNULL[["alphaNULL"]],
betaNULL=bm1@result$fitNULL$thetaNULL[["betaNULL"]],
alphaPos=4, betaPos=2), gamma=NULL),
ctrl=list(fitNULL=FALSE, tol=1e-1), verbose=TRUE, gradtol=1e-3)
view(bm1, "fitNULL")
view(bm1, "fitBM")
bm1

## End(Not run)
```

`buildPAN` *Build an igraph or RedeR graph for PAN*

Description

The function builds a graph for the inferred PAN so that it can be visualize in `igraph` or `RedeR`

Usage

```
buildPAN(object, engine="igraph", para=list(nodeColor=NULL, nodeSize=NULL,
edgeWidth=NULL, edgeColor=NULL, nodeSumCols=1, nodeSumMethod="none",
hideNeg=TRUE), verbose=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
engine	a character value specifying which graphics engine to use: 'igraph' or 'RedeR'.
para	a list of parameters specifying graph attributes (see details)
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	all the other arguments accepted by the function pvclust.

Details

Here are the introductions for the detailed graph attributes that the user can specify:

'nodeColor' and 'nodeSize' - a vector of node colors or sizes. Please note that the order of color or size must be concordant with the gene ids in PANR:the argument pheno of the object of BetaMixture.

'nodeSumCols' and 'nodeSumMethod' - these two arguments are used to scale the colors of nodes by phenotypes. The former argument is a numeric vector specifying the columns in slot pheno of the object of class BetaMixture; while the latter one is a character value giving the method to summarize these columns of phenotypes: either 'mean' or 'median'.

'edgeColor' and 'edgeWidth' - a vector of edge colors or width.

'pValCutoff' - the argument is only used when what='module' and for module searching based on pvclust. Only significant modules will be displayed.

'minSize' and 'maxSize' - two arguments controlling the size of modules which will be used to filtered out modules that are too small or too large.

'hideNeg' - a logical value specifying whether or not to hide edges with negative associations

Value

This function will return an object of class PAN with inferred gene modules (modules$clusters) and corresponding p-values (modules$pval) updated in slot 'modules'.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

R. Suzuki and H. Shimodaira. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12):1540, 2006.

Examples

```
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
```

`cosineSim` *Compute cosine similarities or distances between pairs of genes*

Description

This function compute cosine similarities or distances between pairs of genes, given measured rich phenotypes.

Usage

```
cosineSim(x)
cosineDist(x)
```

Arguments

`x`	a numeric matrix of rich phenotypes with rows and columns specifying samples and genes, respectively.

Value

This function will return either a numeric matrix of cosine similarities or an object of `dist`.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

PANR:assoScore

Examples

```
toydata<-matrix(rnorm(n=2000, mean=0, sd=4), nrow=100, ncol=20)
toycossim<-cosineSim(t(toydata))
toycosdist<-cosineDist(t(toydata))
```

`data-Bakal2007` *Rich morphological phenotypes for gene overexpression and RNA interference screens*

Description

The data set we use here comes from quantitative morphological screening for 249 gene-overexpression or RNAi knock-down experiments. For each individual cell, 145 different geometric features were computed by imaging analysis, and are subsequently scored with NNs trained to discriminate seven reference TCs with distinctive morphologies. For each TC, NN z-scores were computed from all scored cells in this TC (more details in Bakal 2007).

`Bakal2007`: a matrix of NN z-scores with rows and columns corresponding to 249 TCs and seven reference TCs.

`Bakal2007Cluster`: unsupervised hierarchical clustering results by Bakal et al.

`nodeColor`: colors scaled according to the clustering results by Bakal et al.

`bm1`: an object of `BetaMixture`, which includes data and results for beta-mixture modelling on the association densities based on first-order cosine similarities of the phenotyping screens.

Usage

```
##see example for details
```

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Bakal, C. and Aach, J. and Church, G. and Perrimon, N. (2007). Quantitative morphological signatures define local signaling networks regulating cell morphology. Science, 316(5832), 1753.

Examples

```
data(Bakal2007)
dim(Bakal2007)
data(bm)
bm1
```

edgeWeight *Compute edge weights for posterior association networks*

Description

This is an internal function to compute edge weights before inferring a posterior association network.

Usage

```
edgeWeight(object, which="bm1", type="SNR", log=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
which	a character value specifying which BetaMixture modelling result to use: first-order (if 'bm1') or second-order (if 'bm2').
type	a character value giving the type of edge weight to compute: signal- to-noise ratio (if 'SNR'), posterior probability odd (if 'PPR') or posterior probability (if 'PP').
log	a logical value specifying whether or not to compute logrithms for edge weights.

Details

This function will be called by infer to compute edge weights for posterior association networks. When inferring a signed PAN, signal-to-noise ratios are suggested to use; while inferring a PAN of only positive associations, posterior probability odds or posterior probabilities are preferred.

Value

This function will return a numeric adjacency matrix of edge weights.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

infer

exportPAN *Export inferred PAN or module graphs to files*

Description

Powered by function write.graph in package igraph, this function writes the inferred PAN or module graphs to files in a variety of formats that are supported by igraph.

Usage

```
exportPAN(object, file="pan", what="graph", moduleID=1, format="gml",
verbose=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
file	the name of the file to write the graph or module (no need to put a filename extension).
what	a character value specifying what to write: 'graph' or 'pvclustModule'.
moduleID	a numeric or integer vector of modules to write (see details in sigModules).
format	a character value specifying the format to write (see more details in write.graph).
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	not in use, but only for further extension.

Value

a numeric vector of ids for significant gene modules

Author(s)

Xin Wang <xw264@cam.ac.uk>

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
exportPAN(pan, file="pan", what="graph", format="ncol")

## End(Not run)
```

fitBM *Fit a three-beta mixture model to densities of functional gene associations*

Description

The function fits a three-beta mixture model to densities of functional gene associations computed from rich phenotyping screens.

Usage

```
fitBM(object, para=list(zInit=NULL, thetaInit=c(alphaNeg=2, betaNeg=4,
alphaNULL=4, betaNULL=4, alphaPos=4, betaPos=2), gamma=NULL),
ctrl=list(fitNULL=FALSE, tol=1e-3, maxIter=NULL), verbose=TRUE, ...)
```

Arguments

object	an object of S4 class BetaMixture.
para	a list of initial values for parameter estimation in fitting a three-beta mixture model (see 'details').
ctrl	a list of control parameters for the mixture model fitting (see 'details').
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	other arguments of the function nlm.

Details

This function fits a beta-mixture model to functional gene associations using the Expectation-Maximization algorithm. The function allows various parameter settings to perform fitting by the original (if model='global') or stratified (if model='stratified') beta-mixture model (the model should be specified when creating a new object of BetaMixture).

The initial values of the beta distributions can be set by thetaInit, is a numeric vector including the two shape parameters for the '-' (negative), 'x' (NULL) and '+' (positive) distributions. Please note that if ctrl$NULL is TRUE, meaning that the NULL distribution has already been fitted, then para$alphaNULL and para$betaNULL are supposed to be filled in the estimated NULL parameters by the function fitNULL).

zInit is a matrix of posterior probabilities for gene associations following the three mixture components.

The hyper-parameters for the dirichlet priors for the mixture components can also be set by para$gamma, which is a numeric matrix with rows and columns correponding to association partitions and the three beta mixture components.

The other arguments to control the fitting algorithm are tol and maxIter, which are covergence tolerence and the maximal iterations.

Since the estimation of shape parameters of beta distributions are realized by the function nlm numerically, additional arguments for nlm are allowed by ....

Value

This function will return an updated object of class `BetaMixture`.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

`fitNULL`

Examples

```
## Not run:
data(Bakal2007)
bm1<-new("BetaMixture", pheno=Bakal2007, model="global", order=1)
bm1<-fitNULL(bm1, nPerm=10, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="all", verbose=TRUE)
bm1<-fitBM(bm1, para=list(zInit=NULL, thetaInit=c(alphaNeg=2, betaNeg=4,
alphaNULL=bm1@result$fitNULL$thetaNULL[["alphaNULL"]],
betaNULL=bm1@result$fitNULL$thetaNULL[["betaNULL"]],
alphaPos=4, betaPos=2), gamma=NULL),
ctrl=list(fitNULL=FALSE, tol=1e-1), verbose=TRUE, gradtol=1e-3)

## End(Not run)
```

`fitNULL` *Fit the NULL component of a three-beta mixture model for functional gene associations*

Description

The function performs permutations to the input rich phenotyping screens, and subsequently fit a beta distribution to the densities.

Usage

```
fitNULL(object, nPerm=20, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="keepRep", verbose=TRUE, ...)
```

Arguments

object	an object of S4 class BetaMixture.
nPerm	a positive numeric or integer value specifying the number of permutations.
thetaNULL	a list of numeric values giving the initial values for estimating the two shape parameters (see dbeta for more details).
sumMethod	a character value specifying how to summarize estimated parameters from multiple permutations. The current version only supports 'median' and 'mean'.
permMethod	a character value of the method to permute, either 'keepRep' (keep the order of replicates) or 'all' (regardless of replicates).
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	other arguments for function fitdistr.

Details

This function is a step prior to the three-beta mixture model fitting to functional gene association scores. The fitted parameters are then used as a fixed parameters in the three-beta mixture model for further fitting to the real screens. The NULL fitting is performed using the function fitdistr, so other arguments for the function fitdistr are also allowed by the argument ....

Value

This function will return an updated object of class BetaMixture.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

fitBM

Examples

```
data(Bakal2007)
bm1<-new("BetaMixture", pheno=Bakal2007, model="global", order=1)
bm1<-fitNULL(bm1, nPerm=10, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="all", verbose=TRUE)
```

`infer` *Infer a posterior association network*

Description

The function infers a posterior association network from beta-mixture modelling of functional associations computed from rich phenotyping screens.

Usage

```
infer(object, para=list(type='SNR', log=TRUE, sign=TRUE, cutoff=0),
filter=FALSE, verbose=TRUE, ...)
```

Arguments

`object`	an object of S4 class `PAN`.
`para`	a list of parameters to perform inference (see details).
`filter`	a logical value specifying whether or not to filter out genes without any significant association with all the other genes.
`verbose`	a logical value to switch on (if `TRUE`) or off if `FALSE` detailed run-time message.
`...`	not in use, only for further extension.

Details

This function employs different edge weights to infer a posterior association network (see `edgeWeight` for more details). Multiple parameters are provided for the user to specify the network:

'type' - a character value giving the type of edge weights: signal-to- noise ratio ('SNR'), posterior probability ratio ('PPR') or posterior probability ('PP').

'log' - a logical value specifying whether or not to compute logrithms for edge weights.

'sign' - a logical value specifying whether a signed graph should be inferred. It is only valid when `type='SNR'`.

'cutoff' - a numeric value giving the threshold to tell the significance of an edge.

Value

This function will return an object of class `PAN` with inferred PAN updated in slot 'graph'.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

edgeWeight

Examples

```
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
```

---

p2SNR *Translate p-values to Signal-to-Noise Ratios*

Description

The function translate p-values to Signal-to-Noise Ratios based on the fitted mixture model.

Usage

```
p2SNR(object, pval, ...)
```

Arguments

object	an object of S4 class BetaMixture.
pval	a numeric or integer specifying the p-value to translate
...	not in use, only for further extension.

Value

The function will return a data frame including p-values, lower and upper quantiles and corresponding Signal-to-Noise Ratios.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

p2SNR

Examples

```
## Not run:
data(Bakal2007)
bm1<-new("BetaMixture", pheno=Bakal2007, model="global", order=1)
bm1<-fitNULL(bm1, nPerm=10, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="all", verbose=TRUE)
bm1<-fitBM(bm1, para=list(zInit=NULL, thetaInit=c(alphaNeg=2, betaNeg=4,
alphaNULL=bm1@result$fitNULL$thetaNULL[["alphaNULL"]],
betaNULL=bm1@result$fitNULL$thetaNULL[["betaNULL"]],
alphaPos=4, betaPos=2), gamma=NULL),
ctrl=list(fitNULL=FALSE, tol=1e-1), verbose=TRUE, gradtol=1e-3)
p2SNR(bm1, pval=0.01)

## End(Not run)
```

PAN-class *An S4 class for inferring a posterior association network*

Description

This S4 class includes methods to infer posterior association networks and enriched modules of functional gene interactions from rich phenotyping screens.

Objects from the Class

Objects of class PAN can be created from `new("PAN", bm1, bm2)` (see the example below for details).

Slots

bm1: an object of S4 class `BetaMixture`, which models the first- order similarities between genes (see `BetaMixture`).

bm2: an object of S4 class `BetaMixture`, which models the second- order similarities between genes (modularity).

edgeWt: a weighted adjacency matrix computed from the posterior probabilities for gene associations to belong to mixture components (see `edgeWeight`).

engine: the graphics visualization engine for PAN.

graph: a weighted adjacency matrix with edge weights satisfying certain constraints specified by the user (see `infer`).

modules: a list summarizing inferred enriched functional gene modules (see `pvclustModule`.

iPAN: an igraph object for storing the inferred PAN.

legend: a list of legends for built PAN graph.

summary: a list of summary information for available results.

Methods

An overview of methods (More detailed introduction can be found in help for each specific function.):

`edgeWeight` compute edge weights by signal-to-noise ratio, posterior odd or posterior probabilities (more details in `edgeWeight`).

`infer` infer a posterior association network given the beta-mixture model(s) fitted to first- and/or second-order similarities (more details in `infer`).

`pvclustModule` search significantly enriched functional gene modules by hierarchical clustering with bootstrap resampling based on the package `pvclust` (more details in `pvclustModule`).

`exportPAN` export the inferred PAN or modules to file(s) in a variety of formats (more details in `exportPAN`).

`sigModules` retrieve significant gene modules that satisfy the given p-value cutoff and module size range (more details in `sigModules`).

`viewNestedModules` view a nested structure for gene modules searched by hierarchical clustering (more details in `viewNestedModules`).

`viewPAN` view the inferred PAN or modules in `igraph` or `RedeR` (more details in `viewPAN`).

`buildPAN` build a PAN graph for visualization in `igraph` or `RedeR` (more details in `viewPAN`).

`viewLegend` View the legends for the graph built for PAN.

`summarize` summarize results including input data and parameters, inferred graph and modules.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

`edgeWeight` `infer` `pvclustModule` `exportPAN` `sigModules` `viewPAN` `viewNestedModules` `summarize`

Examples

```
## Not run:
data(bm, package="PANR")
##create an object of `PAN'
pan<-new("PAN", bm1=bm1)
##infer a PAN
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
##build a PAN graph for RedeR, hide negative edges
##using colors scaled based on the clustering results from Bakal et al. 2007
data(Bakal2007Cluster)
```

```
pan<-buildPAN(pan, engine="RedeR", para=list(nodeColor=nodeColor, hideNeg=TRUE))
##view PAN in RedeR
library(RedeR)
viewPAN(pan, what="graph")
##print a summary of results
summarize(pan, "ALL")

## End(Not run)
```

---

permNULL *Do permutations for input rich phenotyping screens.*

Description

This is an internal S4 method for class PAN to permute the original inputted phenotyping screens to further fit the dissociation component of the beta-mixture model.

Usage

```
permNULL(object, permMethod="keepRep", ...)
```

Arguments

object	an object of BetaMixture.
permMethod	a character value of the method to permute, either 'keepRep' (keep the order of replicates) or 'all' (regardless of replicates).

Value

a matrix of permuted data.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

fitNULL, fitBM, BetaMixture

Examples

```
data(bm)
bm.perm<-permNULL(bm1)
```

pvclustModule *Search enriched functional gene modules by pvclust*

Description

The function employs the R package pvclust to search significant functional gene modules using hierarchical clustering with bootstrap resampling.

Usage

```
pvclustModule(object, nboot=1000, metric="cosine", hclustMethod="average",
filter=TRUE, verbose=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
nboot	a numeric value giving the number of bootstraps for pvclust.
metric	a character value specifying which distance metric to use for the hierarchical clustering: 'correlation', 'cosine', 'abscor' or those allowed by the argument 'method' in dist.
hclustMethod	the agglomerative method used in hierarchical clustering: 'average', 'ward', 'single', 'complete', 'mcquitty', 'median' or 'centroid' (see the argument method in hclust for more details).
filter	a logical value specifying whether or not to filter out screening data of genes without significant associations with all the other genes.
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	all the other arguments accepted by the function pvclust.

Details

This function performs hierarchical clustering with bootstrap resampling to quantify significance of gene clusters (modules) based on the package pvclust.

Value

This function will return an object of class PAN with inferred gene modules (modules$clusters) and corresponding p-values (modules$pval) updated in slot 'modules'.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

R. Suzuki and H. Shimodaira. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12):1540, 2006.

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
##need pvclust to search modules
library(pvclust)
pan<-pvclustModule(pan, nboot=1000, metric="cosine",
hclustMethod="average", filter=TRUE, verbose=TRUE, r=c(5:12/7))

## End(Not run)
```

---

sigModules *Retrieve ids for significant gene modules searched by pvclust*

Description

The function retrieve ids for significant gene modules that satisfy the given p-value cutoff and module size range.

Usage

```
sigModules(object, pValCutoff=0.01, minSize=3, maxSize=100, sortby="size",
decreasing=FALSE, ...)
```

Arguments

object	an object of S4 class PAN.
pValCutoff	a numeric value specifying the p-value cutoff to tell the significance of a gene module.
minSize	a numeric or integer value giving the minimal size of gene modules.
maxSize	a numeric or integer value giving the maximal size of gene modules.
sortby	a character value specitying how to sort the list of gene modules: by 'size' (module size) or 'pval' (pvclust p-value).

decreasing	a logical value specifying whether or not the sorting will be conducted decreasingly.
...	not in use, but only for further extension.

Details

This function facilitates the user to retrieve significant gene modules found by pvclust and obtain their ids, which can be subsequently used for visualization (see viewPAN for details).

Value

a numeric vector of ids for significant gene modules

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

R. Suzuki and H. Shimodaira. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12):1540, 2006.

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
##need pvclust to search modules
library(pvclust)
pan<-pvclustModule(pan, nboot=1000, metric="cosine",
hclustMethod="average", filter=TRUE, verbose=TRUE, r=c(5:12/7))
inds<-sigModules(pan, pValCutoff=0.01, minSize=5, maxSize=100, sortby="size",
decreasing=FALSE)
pan@modules$clusters[inds]

## End(Not run)
```

SNR2p *Translate p-values to Signal-to-Noise Ratios*

Description

The function translate Signal-to-Noise Ratios to p-values based on the fitted mixture model.

Usage

```
SNR2p(object, SNR, ...)
```

Arguments

object	an object of S4 class BetaMixture.
SNR	a numeric or integer specifying the SNR to translate
...	not in use, only for further extension.

Value

The function will return a data frame including SNRs, lower and upper quantiles and corresponding p-values.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

SNR2p

Examples

```
## Not run:
data(Bakal2007)
bm1<-new("BetaMixture", pheno=Bakal2007, model="global", order=1)
bm1<-fitNULL(bm1, nPerm=10, thetaNULL=c(alphaNULL=4, betaNULL=4),
sumMethod="median", permMethod="all", verbose=TRUE)
bm1<-fitBM(bm1, para=list(zInit=NULL, thetaInit=c(alphaNeg=2, betaNeg=4,
alphaNULL=bm1@result$fitNULL$thetaNULL[["alphaNULL"]],
betaNULL=bm1@result$fitNULL$thetaNULL[["betaNULL"]],
alphaPos=4, betaPos=2), gamma=NULL),
ctrl=list(fitNULL=FALSE, tol=1e-1), verbose=TRUE, gradtol=1e-3)
```

```
SNR2p(bm1, SNR=10)

## End(Not run)
```

---

summarize *Summarize the object of S4 class 'BetaMixture' or 'PAN'*

Description

The function helps print a summary of an object of S4 class BetaMixture or PAN.

Usage

```
summarize(object, what='ALL', ...)
```

Arguments

object	an object of S4 class BetaMixture or PAN.
what	a character value specifying what to print (see details).
...	not in use, only for further extension.

Details

This function print a summary of an object of BetaMixture or PAN. The function is also called by S4 method show, which prints only a short message about the input parameters and data.

For an object of class BetaMixture:

If what='input', the function prints to screen a summary of input parameters; If what='fitNULL', the function prints to screen a summary of fitting results for the NULL distribution. If what='fitBM', the function prints to screen a summary of fitting results for the beta-mixture model. If what='ALL', all above messages will be printed.

For an object of class PAN:

If what='input', the function prints to screen a summary of input object(s) of class BetaMixture; If what='graph', the function prints to screen a summary of inferred posterior association network; If what='module', the function prints to screen a summary of functional gene modules; If what='ALL', all above messages will be printed.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

Examples

```
data(bm)
summarize(bm1, what='ALL')
```

`view` *View the results of beta-mixture model fitting*

Description

This function facillitate the user to view and check the fitting results of NULL and beta-mixture distributions.

Usage

```
view(object, what="fitBM", ...)
```

Arguments

`object`	an object of S4 class `BetaMixture`.
`what`	a character value specifying to show the fitting results of NULL (if `what='fitNULL'`) or beta-mixture model (if `what='fitBM'`).
`...`	not in use, only for further extension.

Details

The function help the user to view and check the fitting of NULL and three beta-mixture model to permuted and real phenotyping screens, respectively. For either fitting, a histogram of association scores and the fitted beta distribution(s) will be plotted. For beta-mixture fitting, the integrated probability density function for the mixed distribution will also be plotted.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

`fitNULL` `fitBM`

Examples

```
data(bm)
view(bm1, what="fitNULL")
view(bm1, what="fitBM")
```

viewLegend *View the legends for the graph built for PAN*

Description

This function helps the user to view legends for the built PAN graph.

Usage

```
viewLegend(object, what="nodeColor", ...)
```

Arguments

object	an object of S4 class PAN.
what	a character value specifying which legend to view: 'nodeColor', 'nodeSize' or 'edgeWidth'.
...	not in use, but only for further extension.

Author(s)

Xin Wang <xw264@cam.ac.uk>

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
viewLegend(pan, "edgeWidth")

## End(Not run)
```

viewNestedModules *View the nested modules in a posterior association network in RedeR*

Description

The function displays the nested enriched functional gene modules found by pvclust in a powerful graphic visualization software RedeR.

Usage

```
viewNestedModules(object, pValCutoff=0.01, minSize=3, maxSize=100,
verbose=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
pValCutoff	a numeric value specifying the p-value cutoff to tell the significance of a gene module.
minSize	a numeric or integer value giving the minimal size of gene modules.
maxSize	a numeric or integer value giving the maximal size of gene modules.
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	not in use, but only for further extension.

Details

This function presents the searched enriched functional modules in RedeR - a bioconductor package for network visualization.

Please note that the user is expected to run buildPAN to build a graph and search modules using pvclustModule prior to visualize using this function.

Please also note that if 'RedeR' is selected as the graphics engine, it is suggested to manually organise the sizes and positions of containers (for nesting gene modules) run a dynamic layout to obtain the best structure for the network.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Roland F. Schwarz, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

addGraph, nestNodes, viewPAN, buildPAN

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
```

```
##need pvclust to search modules
library(pvclust)
pan<-pvclustModule(pan, nboot=10000, metric="cosine2",
hclustMethod="average", filter=TRUE, verbose=TRUE, r=c(5:12/7))
viewNestedModules(pan, pValCutoff=0.05, minSize=5, maxSize=100)

## End(Not run)
```

---

viewPAN *Show posterior association networks or modules in 'igraph' or 'RedeR'*

Description

The function display inferred posterior association networks or enriched functional gene modules in igraph or a powerful graphical visualization software RedeR.

Usage

```
viewPAN(object, what="graph", moduleID=1,
layout="layout.fruchterman.reingold", verbose=TRUE, ...)
```

Arguments

object	an object of S4 class PAN.
what	a character value specifying which to show: 'graph' or 'pvclustModule'.
moduleID	a numeric or integer vector of modules to view (see details in sigModules). This argument will be applied only when what='pvclustModule'.
layout	a character value specifying the layout method (see details in layout). This argument will be applied only when engine='igraph' when build a graph for PAN using buildPAN.
verbose	a logical value to switch on (if TRUE) or off if FALSE detailed run-time message.
...	not in use, but only for further extension.

Details

This function presents the inferred posterior association network or enriched functional modules in igraph or RedeR depending on the graphics engine used when building the graph for PAN (details in buildPAN).

Please note that when viewing a dense PAN in 'igraph', it could be very messy. Multiple modules can be viewed at the same time in 'RedeR' powered by its feature of 'containers', which are used to group gene modules. When view multiple modules in 'igraph',

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Xin Wang, Mauro Castro, Klaas W. Mulder and Florian Markowetz, Posterior association networks and enriched functional gene modules inferred from rich phenotypic perturbation screens, in preparation.

See Also

buildPAN, layout

Examples

```
## Not run:
data(bm, package="PANR")
pan<-new("PAN", bm1=bm1)
pan<-infer(pan, para=list(type="SNR", log=TRUE, sign=TRUE, cutoff=log(5)),
filter=FALSE, verbose=TRUE)
data(Bakal2007Cluster, package="PANR")
pan<-buildPAN(pan, engine="igraph", para=list(nodeColor=nodeColor,
hideNeg=TRUE), verbose=TRUE)
##view inferred PAN
viewPAN(pan, what='graph', layout="layout.fruchterman.reingold")

## End(Not run)
```

Index