Package 'RTN'

April 5, 2014

Type Package

Title Reconstruction of transcriptional networks and analysis of master regulators

Version 1.0.0

Date 2013-10-05

Author Mauro Castro, Xin Wang, Michael Fletcher, Florian Markowetz and Kerstin Meyer

Maintainer Mauro Castro <mauro.a.castro@gmail.com>

Depends R (>= 2.15), methods, igraph

Imports minet, snow, limma, RedeR (>= 1.8.1)

Suggests HTSanalyzeR, RUnit, BiocGenerics

Description This package provides classes and methods for transcriptional network inference and analysis. Modulators of transcription factor activity are assessed by conditional mutual information, and master regulators are mapped to phenotypes using different strategies, e.g., gene set enrichment, shadow and synergy analyses.

License Artistic-2.0

biocViews NetworkInference, NetworkAnalysis, NetworkEnrichment,GeneRegulation, GeneExpression, GraphsAndNetworks

URL http://dx.doi.org/10.1038/ncomms3464

Collate ClassUnions.R AllChecks.R AllClasses.R AllGenerics.R AllSupplements.R AllPlots.R TNA-methods.R TNI-methods.R

LazyLoad yes

R topics documented:

RTN-package *Reconstruction and Analysis of Transcriptional Networks.*

Description

This package provides classes and methods for transcriptional network inference and and analysis. Modulators of transcription factor activity are assessed by conditional mutual information, and master regulators are mapped to phenotypes using different strategies, e.g., gene set enrichment, shadow and synergy analyses.

Details

Package:	RTN
Type:	Package
Depends:	methods, igraph
Imports:	minet, snow, limma, RedeR
Suggests:	HTSanalyzeR
License:	Artistic-2.0
biocViews:	CellBasedAssays, Bioinformatics, MultipleComparisons
Collate:	ClassUnions.R, AllChecks.R, AllClasses.R, AllGenerics.R, AllSupplements.R, AllPlots.R, TNA-methods.R, TN
LazyLoad:	yes

Index

Further information is available in the vignettes by typing `vignette("RTN")`. Documented topics are also available in HTML by typing `help.start()` and selecting the RTN package from the menu.

Author(s)

Maintainer: Mauro Castro <mauro.a.castro@gmail.com>

References

Castro, M. A. A. et al. *RTN: Reconstruction and Analysis of Transcriptional Networks*. Journal Paper (in preparation), 2013.

`RTN.data` *A pre-processed dataset for the RTN package.*

Description

A minimum dataset used to demonstrate RTN main functions.

Usage

```
data(dt4rtn)
```

Format

`dt4rtn` List containing 6 R objects: 'gexp', 'gexpIDs', 'pheno', 'phenoIDs', 'hits' and 'tfs'.

Details

The dataset consists of 6 R objects used in the RTN vignettes. It should be regarded as a toy example for demonstration purposes only, despite being extracted, pre-processed and size-reduced from Fletcher et al. (2012) and Curtis et al. (2012).

dt4rtn$gexp a named gene expression matrix with 50 samples (reduced for demonstration purposes only).

dt4rtn$gexpIDs a data.frame of characters with probe ids matching a secundary annotation source (e.g. Probe-to-ENTREZ).

dt4rtn$pheno a named numeric vector with differential gene expression data.

dt4rtn$phenoIDs a data.frame of characters with probe ids matching a secundary annotation source (e.g. Probe-to-ENTREZ).

dt4rtn$hits a character vector with genes differentially expressed.

dt4rtn$tfs a named vector with transcriptions factors.

References

Fletcher M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Journal Paper (in preparation), 2012.

Curtis C. et al., *The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups.* Nature 486, 7403. 2012.

Examples

```
data(dt4rtn)
```

TNA-class *Class "TNA": an S4 class for Transcriptional Network Analysis.*

Description

This S4 class includes a series of methods to do enrichment, synergy, shadow and overlap analyses in transcriptional networks.

Objects from the Class

Objects can be created by calls of the form `new("TNA", referenceNetwork, transcriptionalNetwork, transcriptionF`

Slots

`referenceNetwork`: Object of class `"matrix"`, an optional partial co-expression matrix.

`transcriptionalNetwork`: Object of class `"matrix"`, a partial co-expression matrix.

`transcriptionFactors`: Object of class `"char_Or_NULL"`, a vector of transcription factors.

`phenotype`: Object of class `"num_Or_int"`, a numeric or integer vector of phenotypes named by gene identifiers.

`hits`: Object of class `"character"`, a character vector of gene identifiers for those considered as hits.

`annotation`: Object of class `"data.frame"`, a data frame with transcriptional network annotation.

`listOfReferenceRegulons`: Object of class `"list"`, a list of regulons derived from the referenceNetwork.

`listOfRegulons`: Object of class `"list"`, a list of regulons derived from the transcriptionalNetwork (a 'regulon' is a vector of genes or potential transcription factor targets).

`listOfModulators`: Object of class `"list"`, a list of modulators derived from the `tni.conditional` analysis.

`para`: Object of class `"list"`, a list of parameters for transcriptional network analysis. These parameters are those listed in the functions `tna.mra`, `tna.overlap`, `tna.gsea1`, `tna.synergy`, `tna.shadow` and `tna.gsea2`.

`results`: Object of class `"list"`, a list of results (see return values in the functions `tna.mra`,`tna.gsea1`, `tna.overlap`, `tna.synergy`, `tna.shadow` and `tna.gsea2`)

`summary`: Object of class `"list"`, a list of summary information for `transcriptionalNetwork`, `transcriptionFactors`, `phenotype`,`listOfRegulons`, `para`, and `results`.

`status`: Object of class `"character"`, a single character value specifying the status of the TNI object based on the available methods.

Methods

tna.mra `signature(object = "TNA")`: see `tna.mra`

tna.overlap `signature(object = "TNA")`: see `tna.overlap`

tna.gsea1 `signature(object = "TNA")`: see `tna.gsea1`

tna.gsea2 `signature(object = "TNA")`: see `tna.gsea2`

tna.synergy `signature(object = "TNA")`: see `tna.synergy`

tna.shadow `signature(object = "TNA")`: see `tna.shadow`

tna.get `signature(object = "TNA")`: see `tna.get`

tna.graph `signature(object = "TNA")`: see `tna.graph`

Author(s)

Mauro Castro

See Also

`TNI-class`.

Examples

```
data(dt4rtn)

rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs)
```

`tna.get` *Get information from individual slots in a TNA object.*

Description

Get information from individual slots in a TNA object. Available results from a previous analysis can be selected either by pvalue cutoff (default) or top significance.

Usage

```
tna.get(object, what = "summary", order = TRUE, ntop = NULL, reportNames = TRUE)
```

Arguments

`object`	an object of class 'TNA' `TNA-class`.
`what`	a single character value specifying which information should be retrieved from the slots. Options: "tnet", "refnet", "tfs", "pheno", "regulons","refregulons", "para", "mra", "gsea", "cmap","overlap", "synergy", "shadow", "summary" and "status". Regulons can also be retrieved mapped to the available phenotype vector ("regulons.and.pheno" or "refregulons.and.pheno") or mapped to the assiged mode of action ("regulons.and.mode" or "refregulons.and.mode").
`order`	a single logical value specifying whether or not the output data should be ordered by significance. Valid only for "gsea", "overlap", "synergy" or "shadow" options.
`ntop`	a single integer value specifying to select how many results of top significance from "gsea", "overlap", "synergy" or "shadow" options.
`reportNames`	a single logical value specifying to reporte regulons with 'names' (when reportNames=TRUE) or not (when reportNames=FALSE). This option is effective only if transcription factors were named with alternative identifiers in the pre-processing analysis. It takes effect for the options "mra", "gsea", "overlap", "synergy" and "shadow".

Value

get the slot content from an object of class 'TNA' `TNA-class`.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)
tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

# run MRA analysis pipeline
rtna <- tna.mra(rtna)

# check summary
tna.get(rtna,what="summary")

# get results, e.g., MRA analysis
tna.get(rtna,what="mra")

## End(Not run)
```

---

tna.graph *Compute a graph from TNA objects.*

Description

Extract results from a TNA object and compute a graph.

Usage

```
tna.graph(object, tnet = "dpi", gtype="rmap", minRegulonSize=15, tfs=NULL, amapFilter="quantile", amap
```

Arguments

object	an object of class 'TNA' TNA-class.
tnet	a single character value specifying which network information should be used to compute the graph. Options: "ref" and "dpi".
gtype	a single character value specifying the graph type. Options: "rmap" and "amap". The "rmap" option returns regulatory maps represented by TFs and targets (regulons) and "amap" computes association maps among regulons (estimates the overlap using the Jaccard Coefficient).
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Regulons with fewer than this number are removed from the graph.

tfs	a vector with transcription factor identifiers.
amapFilter	a single character value specifying which method should be used to filter association maps (only when gtype="amap"). Options: "phyper","quantile" and "custom".
amapCutoff	a single numeric value (>=0 and <=1) specifying the cutoff for association map filter. When amapFilter="phyper", amapCutoff corresponds to a pvalue cutoff; when amapFilter="quantile", amapCutoff corresponds to a quantile threshold; and when amapFilter="custom", amapCutoff is a JC threshold.

Value

a graph object.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni, eps=0.05)

#run MRA analysis pipeline
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)
rtna <- tna.mra(rtna)

# compute regulatory maps
g<-tna.graph(rtna, tnet="dpi", gtype="rmap", tfs=tfs4test)

# option: plot the igraph object using RedeR
#library(RedeR)
#rdp<-RedPort()
#calld(rdp)
#addGraph(rdp,g)
#relax(rdp,p1=50,p5=20)

# compute association maps
g<-tna.graph(rtna, tnet="ref", gtype="amap", tfs=tfs4test)


## End(Not run)
```

`tna.gsea1` *One-tailed Gene Set Enrichment Analysis (GSEA) over a list of regulons.*

Description

This function takes a TNA object and returns the results of the GSEA analysis over a list of regulons in a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.gsea1(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, nPermutations=1000,
     exponent=1, tnet="dpi", orderAbsValue=TRUE, stepFilter=TRUE, tfs=NULL, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`nPermutations`	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
`exponent`	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
`tnet`	a single character value specifying which transcriptional network should to used to compute the GSEA analysis. Options: "dpi" and "ref".
`orderAbsValue`	a single logical value indicating whether the values should be converted to absolute values and then ordered (if TRUE), or ordered as they are (if FALSE).
`stepFilter`	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from `tna.mra` (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
`tfs`	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'gsea1' option in `tna.get`.

Author(s)

Mauro Castro, Xin Wang

See Also

TNA-class

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run GSEA1 analysis pipeline
rtna <- tna.gsea1(rtna,stepFilter=FALSE)

#get results
tna.get(rtna,what="gsea1")

# run parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtna <- tna.gsea1(rtna)
stopCluster(getOption("cluster"))

## End(Not run)
```

---

tna.gsea2 *Two-tailed Gene Set Enrichment Analysis (GSEA) over a list of regulons.*

Description

This function takes a TNA object and returns a CMAP-like analysis obtained by two-tailed GSEA over a list of regulons in a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.gsea2(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, nPermutations=1000,
        exponent=1, tnet="dpi", stepFilter=TRUE, tfs=NULL, verbose=TRUE)
```

Arguments

object	a preprocessed object of class 'TNA' TNA-class.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
pAdjustMethod	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
nPermutations	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
exponent	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
tnet	a single character value specifying which transcriptional network should to used to compute the GSEA analysis. Options: "dpi" and "ref".
stepFilter	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from tna.mra (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
tfs	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'gsea2' option in tna.get.

Author(s)

Mauro Castro

See Also

TNA-class

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
```

```
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run GSEA2 analysis pipeline
rtna <- tna.gsea2(rtna,stepFilter=FALSE)

#get results
tna.get(rtna,what="gsea2")

# run parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtna <- tna.gsea2(rtna)
stopCluster(getOption("cluster"))

## End(Not run)
```

`tna.mra` *Master Regulator Analysis (MRA) over a list of regulons.*

Description

This function takes a TNA object and returns the results of the RMA analysis over a list of regulons from a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.mra(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, tnet="dpi", verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`tnet`	a single character value specifying which transcriptional network should to used to compute the MRA analysis. Options: "dpi" and "ref".
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'rma' option in `tna.get`.

Author(s)

Mauro Castro

See Also

TNA-class

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run MRA analysis pipeline
rtna <- tna.mra(rtna)

#get results
tna.get(rtna,what="mra")


## End(Not run)
```

---

tna.overlap *Overlap analysis over a list of regulons.*

Description

This function takes a TNA object and returns the results of the overlap analysis among regulons in a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.overlap(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, tnet="ref",
            tfs=NULL, verbose=TRUE)
```

Arguments

object	a preprocessed object of class 'TNA' TNA-class.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
pAdjustMethod	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
tnet	a single character value specifying which transcriptional network should to used to compute the overlap analysis. Options: "dpi" and "ref".
tfs	an optional vector with transcription factor identifiers.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'overlap' option in tna.get.

Author(s)

Mauro Castro

See Also

TNA-class

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run overlap analysis pipeline
rtna <- tna.overlap(rtna)

#get results
tna.get(rtna,what="overlap")

## End(Not run)
```

`tna.plot.gsea1` *Plot enrichment analyses from TNA objects.*

Description

This function takes a TNA object and plots the one-tailed GSEA results for individual regulons.

Usage

```
tna.plot.gsea1(object, regulon.order="size", ntop=NULL, tfs=NULL, filepath=".", ylimPanels=c(0.0,3.5,
heightPanels=c(1,1,3), width=6, height=5, ylabPanels=c("Phenotype","Regulon","Enrichment score"),
xlab="Position in the ranked list of genes", labPheno="tna_test", alpha=0.5, sparsity=10,
splitcor=FALSE, autoformat=TRUE, ...)
```

Arguments

`object`	an object of class 'TNA' TNA-class.
`regulon.order`	a single character value specifying whether regulons should be ordered by 'size', 'score', 'pvalue', 'adj.pvalue' and 'name' (or 'none' to keep the input ordering).
`ntop`	a single integer value specifying how many regulons of top significance will be plotted.
`tfs`	an optional vector with transcription factor identifiers (this option overrides the 'ntop' argument).
`filepath`	a single character value specifying where to store GSEA figures.
`ylimPanels`	a numeric vector of length=4 specifying y coordinates ranges of the 1st and 3th plots (i.e. ylim for 'Phenotypes' and 'Running enrichment score').
`heightPanels`	a numeric vector of length=3 specifying the relative height of each panel in the plot.
`width`	a single numeric value specifying the width of the graphics region in inches.
`height`	a single numeric value specifying the height of the graphics region in inches.
`ylabPanels`	a character vector of length=3 specifying the the title for the y axes.
`xlab`	a single character specifying the the title for the x axis.
`labPheno`	a single character specifying a label for the phenotype (will also be used as the name of output file).
`alpha`	a single numeric value in [0,1] specifying the transparency of the hits in the ranked list.
`sparsity`	a single integer value (>1) specifying the density of the dots representing the running score.
`splitcor`	a single logical value specifying to divide regulon genes into positive and negative correlation groups.
`autoformat`	a single logical value specifying to set the graph format using predefined themes. This option overrides the "ylimPanels" argument.
`...`	other arguments used by the function pdf.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

# run GSEA analysis pipeline
rtna <- tna.gsea(rtna, stepFilter=FALSE)

# plot available GSEA results
tna.plot.gsea1(rtna, width=6, height=4, heightPanels=c(1,0.7,3),
              ylimPanels=c(0,3.5,0,0.9), labPheno="test")

## End(Not run)
```

`tna.plot.gsea2` *Plot enrichment analyses from TNA objects.*

Description

This function takes a TNA object and plots the two-tailed GSEA results for individual regulons.

Usage

```
tna.plot.gsea2(object, regulon.order="size", ntop=NULL, tfs=NULL, filepath=".", ylimPanels=c(-1.0,3.0
heightPanels=c(1.5,0.7,5.0), width=5, height=4, ylabPanels=c("Phenotype","Regulon","Enrichment score"
xlab="Position in the ranked list of genes", labPheno="tna_test", alpha=1.0, sparsity=10, autoformat=TR
```

Arguments

`object`	an object of class 'TNA' `TNA-class`.
`regulon.order`	a single character value specifying whether regulons should be ordered by 'size', 'score', 'pvalue', 'adj.pvalue' and 'name' (or 'none' to keep the input ordering).
`ntop`	a single integer value specifying how many regulons of top significance will be plotted.

tfs	an optional vector with transcription factor identifiers (this option overrides the 'ntop' argument).
filepath	a single character value specifying where to store GSEA2 figures.
ylimPanels	a numeric vector of length=4 specifying y coordinates ranges of the 1st and 3th plots (i.e. ylim for 'Phenotypes' and 'Running enrichment score').
heightPanels	a numeric vector of length=3 specifying the relative height of each panel in the plot.
width	a single numeric value specifying the width of the graphics region in inches.
height	a single numeric value specifying the height of the graphics region in inches.
ylabPanels	a character vector of length=3 specifying the the title for the y axes.
xlab	a single character specifying the the title for the x axis.
labPheno	a single character specifying a label for the phenotype (will also be used as the name of output file).
alpha	a single numeric value in [0,1] specifying the transparency of the hits in the ranked list.
sparsity	a single integer value (>1) specifying the density of the dots representing the running score.
autoformat	a single logical value specifying to set the graph format using predefined themes. This option overrides the "ylimPanels" argument.
...	other arguments used by the function pdf.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

# run GSEA2 analysis pipeline
rtna <- tna.gsea2(rtna, stepFilter=FALSE)

# plot available GSEA2 results
tna.plot.gsea2(rtna, width=6, height=4, heightPanels=c(1,0.7,3),
              ylimPanels=c(0,3.5,0,0.9), labPheno="test")
```

```
## End(Not run)
```

`tna.shadow` *shadow analysis over a list of regulons.*

Description

This function takes a TNA object and returns the results of the shadow analysis over a list of regulons in a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.shadow(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, minIntersectSize=1,
           nPermutations=1000, exponent=1, tnet="ref", orderAbsValue=TRUE, stepFilter=TRUE,
                 tfs=NULL, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`minIntersectSize`	a single integer or numeric value specifying the minimum number of elements in the intersect between any two regulons in the shadow analysis (as percentage value).
`nPermutations`	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
`exponent`	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
`tnet`	a single character value specifying which transcriptional network should to used to compute the shadow and shadow analyses. Options: "dpi" and "ref".
`orderAbsValue`	a single logical value indicating whether the values should be converted to absolute values and then ordered (if TRUE), or ordered as they are (if FALSE).
`stepFilter`	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from `tna.gsea1` (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
`tfs`	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'shadow' in tna.get.

Author(s)

Mauro Castro

See Also

TNA-class tna.shadow

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run overlap analysis pipeline
rtna <- tna.overlap(rtna)

#run shadow analysis pipeline
rtna <- tna.shadow(rtna,stepFilter=FALSE)

#get results
tna.get(rtna,what="shadow")

# run parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtna <- tna.shadow(rtna)
stopCluster(getOption("cluster"))

## End(Not run)
```

---

tna.synergy *Synergy analysis over a list of regulons.*

Description

This function takes a TNA object and returns the results of the synergy analysis over a list of regulons in a transcriptional network (with multiple hypothesis testing corrections).

Usage

```
tna.synergy(object, pValueCutoff=0.05, pAdjustMethod="BH", minRegulonSize=15, minIntersectSize=1,
            nPermutations=1000, exponent=1, tnet="ref", orderAbsValue=TRUE, stepFilter=TRUE,
                  tfs=NULL, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`minIntersectSize`	a single integer or numeric value specifying the minimum number of elements in the intersect between any two regulons in the synergy analysis (as percentage value).
`nPermutations`	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
`exponent`	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
`tnet`	a single character value specifying which transcriptional network should to used to compute the synergy and shadow analyses. Options: "dpi" and "ref".
`orderAbsValue`	a single logical value indicating whether the values should be converted to absolute values and then ordered (if TRUE), or ordered as they are (if FALSE).
`stepFilter`	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from `tna.gsea1` (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
`tfs`	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a data frame in the slot "results", see 'synergy' in `tna.get`.

Author(s)

Mauro Castro

See Also

TNA-class tna.shadow

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

#run synergy analysis pipeline
rtna <- tna.synergy(rtna,stepFilter=FALSE)

#get results
tna.get(rtna,what="synergy")

# run parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtna <- tna.synergy(rtna)
stopCluster(getOption("cluster"))

## End(Not run)
```

TNI-class *Class "TNI": an S4 class for Transcriptional Network Inference.*

Description

This S4 class includes a series of methods to do transcriptional network inference for high-throughput gene expression.

Objects from the Class

Objects can be created by calls of the form `new("TNI", gexp, transcriptionFactors)`.

Slots

`gexp`: Object of class `"matrix"`, a gene expression matrix.

`transcriptionFactors`: Object of class `"char_Or_NULL"`, a vector with transcription factor identifiers.

`modulators`: Object of class `"char_Or_NULL"`, a vector with modulator identifiers.

`annotation`: Object of class `"data.frame"`, a data frame with probe-to-gene information.

`para`: Object of class `"list"`, a list of parameters for transcriptional network inference. These parameters are those listed in the functions `tni.permutation`, `tni.bootstrap` and `tni.dpi.filter`.

`results`: Object of class `"list"`, a list of results (see the returned values in the functions `tni.permutation`).

`summary`: Object of class `"list"`, a list of summary information for `gexp`, `transcriptionFactors`, `para`, and `results`.

`status`: Object of class `"character"`, a single character value specifying the status of the TNI object based on the available methods.

Methods

tni.preprocess `signature(object = "TNI")`: see `tni.preprocess`

tni.permutation `signature(object = "TNI")`: see `tni.permutation`

tni.bootstrap `signature(object = "TNI")`: see `tni.bootstrap`

tni.dpi.filter `signature(object = "TNI")`: see `tni.dpi.filter`

tni.conditional `signature(object = "TNI")`: see `tni.conditional`

tni.get `signature(object = "TNI")`: see `tni.get`

tni.graph `signature(object = "TNI")`: see `tni.graph`

tni2tna.preprocess `signature(object = "TNI")`: see `tni2tna.preprocess`

Author(s)

Mauro Castro

See Also

`TNA-class`

Examples

```
data(dt4rtn)

rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs)
```

tni.bootstrap *Inference of transcriptional networks.*

Description

This function takes a TNI object and returns the consensus transcriptional network.

Usage

```
tni.bootstrap(object, estimator="pearson", nBootstraps=100, consensus=95, parChunks=10, verbose=TRUE)
```

Arguments

object	a processed object of class 'TNI' TNI-class evaluated by the method tni.permutation.
estimator	a character string indicating which estimator to be used for mutual information computation. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
nBootstraps	a single integer or numeric value specifying the number of bootstraps for deriving a consensus between every TF-target association inferred in the mutual information analysis. If running in parallel, nBootstraps should be greater and multiple of parChunks.
consensus	a single integer or numeric value specifying the consensus fraction (in percentage) under which a TF-target association is accepted.
parChunks	an optional single integer value specifying the number of bootstrap chunks to be used in the parallel analysis.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

a matrix in the slot "results" containing a reference transcriptional network, see 'tn.ref' option in tni.get.

Author(s)

Mauro Castro

See Also

TNI-class makeCluster

Examples

```
data(dt4rtn)

# just a few TFs for quick demonstration!
tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")

# create a new TNI object
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

# preprocessing
rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)

# linear version!
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)

# parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtni<-tni.permutation(rtni,parChunks=50)
rtni<-tni.bootstrap(rtni,parChunks=10)
stopCluster(getOption("cluster"))

## End(Not run)
```

`tni.conditional` *Modulators of transcription factor (TF) activity assessed by conditional mutual information analysis.*

Description

This function takes a TNI object and a list of candidate modulators, and computes the conditional mutual information over the TF-target interactions in a transcriptional network (with multiple hypothesis testing corrections). For each TF, the method measures the change in the mutual information between the TF and its targets conditioned on the gene expression of a modulator.

Usage

```
tni.conditional(object, modulators=NULL, tfs=NULL, sampling=35, pValueCutoff=0.01,
pAdjustMethod="bonferroni", statFilter="phyper", statUniverse="all", minRegulonSize=15,
minIntersectSize=5, miThreshold=NULL, parChunks=10, verbose=TRUE)
```

Arguments

`object`	a processed object of class 'TNI' `TNI-class` evaluated by the methods `tni.permutation`, `tni.bootstrap` and `tni.dpi.filter`.

modulators	a vector with gene identifiers for those considered as candidate modulators. If NULL, the function will assess all TFs in the network, provided that the candidate passes all filtering steps.
tfs	a vector with TF identifiers. If NULL, the function will assess all TFs in the network.
sampling	a single integer value specifying the percentage of the available samples that should be included in the analysis. For example, for each TF-target interaction of a given hub, 'sampling = 35' means that the conditional mutual information will be computed from the top and bottom 35% of the samples ranked by the gene expression of a given candidate modulator.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
pAdjustMethod	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
statFilter	a single character value specifying the statistics used to test whether the number of modulated elements observed in a regulon is greater than would be expected by chance. Options: "phyper" and "pchisq" (see "phyper" and "pchisq" for details).
statUniverse	a single character value specifying the universe of the statistical test. Options: "all" (all possible interactions) and "tnet" (only the observed interactions in the transcriptional network).
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Gene sets with fewer than this number are removed from the analysis.
minIntersectSize	a single integer or numeric value specifying the minimum number of observed modulated elements in a regulon (as percentage value).
miThreshold	an optional single numeric value specifying the mutual information threshold used in the conditional analysis. If NULL, 'miThreshold' will be estimated by permutation analysis.
parChunks	an optional single integer value specifying the number of permutation chunks to be used in the parallel analysis.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

a data frame in the slot "results", see 'cdt' option in `tni.get`.

Author(s)

Mauro Castro

References

Wang, K. et al. *Genome-wide identification of post-translational modulators of transcription factor activity in human B cells*. Nat Biotechnol, 27(9):829-39, 2009.

Castro, M.A.A. et al. *RTN: Reconstruction and Analysis of Transcriptional Networks*. Journal Paper (in preparation), 2012.

See Also

TNI-class

Examples

```
data(dt4rtn)

# a few TFs for quick demonstration!
tfs4test<-dt4rtn$tfs[c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")]

# create a new TNI object
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=tfs4test)

## Not run:

# preprocessing
rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)

# permutation analysis (infers the reference/relevance network)
rtni<-tni.permutation(rtni)

# dpi filter (infers the transcriptional network)
rtni<-tni.dpi.filter(rtni)

# ..and a few candidate modulators for demonstration!
mod4test<-rownames(rtni@gexp)[sample(1:nrow(rtni@gexp),200)]

# conditional analysis
rtni<-tni.conditional(rtni, modulators=mod4test, pValueCutoff=1e-3)

#get results
cdt<-tni.get(rtni,what="cdt")

#get summary on a graph object
g<-tni.graph(rtni,gtype="mmap")

###-------------------------------------------
### optional: plot the igraph object using RedeR
library(RedeR)

#--load reder interface
rdp<-RedPort()
calld(rdp)

#---add graph and legends
addGraph(rdp,g)
addLegend.shape(rdp,g)
addLegend.size(rdp,g)
addLegend.color(rdp,g,type="edge")
relax(rdp,p1=50,p5=20)
```

```
## End(Not run)
```

`tni.dpi.filter` *Data Processing Inequality (DPI) filter.*

Description

This function takes a TNI object and returns the transcriptional network filtered by the data processing inequality algorithm.

Usage

```
tni.dpi.filter(object, eps=0, verbose=TRUE)
```

Arguments

`object`	a processed object of class 'TNI' `TNI-class` evaluated by the methods `tni.permutation` and `tni.bootstrap`.
`eps`	a single numeric value specifying the threshold under which Aracne algorithm should apply the dpi filter. For additional detail see `aracne`.
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

a mutual information matrix in the slot "results" containing a dpi-filtered transcriptional network, see 'tn.dpi' option in `tni.get`.

Author(s)

Mauro Castro

See Also

`TNI-class`

Examples

```
data(dt4rtn)

# just a few TFs for quick demonstration!
tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")

# create a new TNI object
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])
```

```
## Not run:

# preprocessing
rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)

# permutation analysis (infers the reference/relevance network)
rtni<-tni.permutation(rtni)

# dpi filter (infers the transcriptional network)
rtni<-tni.dpi.filter(rtni)


## End(Not run)
```

---

tni.get *Get information from individual slots in a TNI object.*

Description

Get information from individual slots in a TNI object and any available results from a previous analysis.

Usage

```
tni.get(object, what="summary", order=TRUE, ntop=NULL, reportNames=TRUE)
```

Arguments

object	an object of class 'TNI' TNI-class.
what	a single character value specifying which information should be retrieved from the slots. Options: "gexp", "tfs", "para", "refnet", "tnet","refregulons","regulons","cdt", "summary" and "status". Regulons can also be retrieved mapped to the assiged mode of action ("regulons.and.mode" or "refregulons.and.mode").
order	a single logical value specifying whether or not the output data should be ordered by significance. Valid only for "cdt" option.
ntop	a single integer value specifying to select how many results of top significance from "cdt" option.
reportNames	a single logical value specifying to reporte regulators with 'names' (when reportNames=TRUE) or not (when reportNames=FALSE). This option is effective only if regulators were named with alternative identifiers. It takes effect for the option "cdt".

Value

get the slot content from an object of class 'TNI' TNI-class.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)

# check summary
tni.get(rtni,what="summary")

# get reference/relevance network
refnet<-tni.get(rtni,what="refnet")

# get transcriptional network
tnet<-tni.get(rtni,what="tnet")

# get status of the pipeline
tni.get(rtni,what="status")

## End(Not run)
```

---

tni.graph *Compute a graph from TNI objects.*

Description

Extract results from a TNI object and compute a graph.

Usage

```
tni.graph(object, tnet = "dpi", gtype="rmap", minRegulonSize=15, tfs=NULL, amapFilter="quantile", amap
```

Arguments

object	an object of class 'TNI' TNI-class.
tnet	a single character value specifying which network information should be used to compute the graph. Options: "ref" and "dpi".

gtype	a single character value specifying the graph type. Options: "rmap", "amap" and "mmap". The "rmap" option returns regulatory maps represented by TFs and targets (regulons); "amap" computes association maps among regulons (estimates the overlap using the Jaccard Coefficient); "mmap" returns modulated maps derived from the tni.conditional function.
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Regulons with fewer than this number are removed from the graph.
tfs	a vector with transcription factor identifiers.
amapFilter	a single character value specifying which method should be used to filter association maps (only when gtype="amap"). Options: "phyper","quantile" and "custom".
amapCutoff	a single numeric value (>=0 and <=1) specifying the cutoff for association map filter. When amapFilter="phyper", amapCutoff corresponds to a pvalue cutoff; when amapFilter="quantile", amapCutoff corresponds to a quantile threshold; and when amapFilter="custom", amapCutoff is a JC threshold.

Value

a graph object.

Author(s)

Mauro Castro

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni, eps=0.05)

# compute regulatory maps
g<-tni.graph(rtni, tnet="dpi", gtype="rmap", tfs=tfs4test)

# option: plot the igraph object using RedeR
library(RedeR)
rdp<-RedPort()
calld(rdp)
addGraph(rdp,g)
addLegend.shape(rdp,g)
addLegend.color(rdp,g,type="edge")
relax(rdp,p1=50,p5=20)
```

```
# compute association maps
resetd(rdp)
g<-tni.graph(rtni, tnet="ref", gtype="amap", tfs=tfs4test)
addGraph(rdp,g)
addLegend.size(rdp,g)
addLegend.size(rdp,g,type="edge")


## End(Not run)
```

`tni.permutation` *Inference of transcriptional networks.*

Description

This function takes a TNI object and returns a transcriptional network inferred by mutual information (with multiple hypothesis testing corrections).

Usage

```
tni.permutation(object, pValueCutoff=0.01, pAdjustMethod="BH", globalAdjustment=TRUE,
        estimator="pearson",nPermutations=1000, pooledNullDistribution=TRUE,
        parChunks=50, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`globalAdjustment`	a single logical value specifying to run global p.value adjustments (when globalAdjustment=TRUE) or not (when globalAdjustment=FALSE).
`estimator`	a character string indicating which estimator to be used for mutual information computation. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
`nPermutations`	a single integer value specifying the number of permutations for deriving TF-target p-values in the mutual information analysis. If running in parallel, nPermutations should be greater and multiple of parChunks.
`pooledNullDistribution`	a single logical value specifying to run the permutation analysis with pooled regulons (when pooledNullDistribution=TRUE) or not (when pooledNullDistribution=FALSE).
`parChunks`	an optional single integer value specifying the number of permutation chunks to be used in the parallel analysis (effective only for "pooledNullDistribution = TRUE").

verbose — a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

a mutual information matrix in the slot "results" containing a reference transcriptional network, see 'tn.ref' option in tni.get.

Author(s)

Mauro Castro

See Also

TNI-class makeCluster

Examples

```
data(dt4rtn)

# just a few TFs for quick demonstration!
tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")

# create a new TNI object
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

# preprocessing
rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)

# linear version!
rtni<-tni.permutation(rtni)

# parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtni<-tni.permutation(rtni,parChunks=50)
stopCluster(getOption("cluster"))

## End(Not run)
```

---

tni.preprocess *A preprocessing function for objects of class TNI.*

Description

This is a generic function.

Usage

```
tni.preprocess(object, gexpIDs=NULL, cvfilter=TRUE, verbose=TRUE)
```

Arguments

object	an object. When this function is implemented as the S4 method of class TNI-class, this argument is an object of class 'TNI'.
gexpIDs	an optional data frame or matrix with probe-to-gene annotation. Column 1 must provide all probe ids listed in the 'gexp' matrix. Ideally, col1 = PROBEID, col2 = GENEID, and col3 = SYMBOL.
cvfilter	a single logical value specifying to remove duplicated genes in the gene expression matrix using the probe-to-gene annotation. In this case, 'gexpIDs' must be provided, with col1 = PROBEID and col2 = GENEID. The decision is made based on the maximum dinamic range (i.e. keeping the probes with max coefficient of variation across all samples).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

TNI-class

Examples

```
data(dt4rtn)

rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs)
rtni <- tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
```

---

tni2tna.preprocess *A preprocessing function for objects of class TNI.*

Description

This is a generic function.

Usage

```
tni2tna.preprocess(object, phenotype=NULL, hits=NULL, phenoIDs=NULL, duplicateRemoverMethod="max", ve
```

Arguments

object	a processed object of class 'TNI' TNI-class evaluated by the methods tni.permutation, tni.bootstrap and tni.dpi.filter.
phenotype	a numeric or integer vector of phenotypes named by gene identifiers. Required for gsea, synergy and shadow methods (see tna.gsea1).
hits	a character vector of gene identifiers for those considered as hits. Required for tna.mra and tna.overlap methods.
phenoIDs	an optional 2cols-matrix used to aggregate genes in the 'phenotype' (e.g. probe-to-gene ids; in this case, col 1 should correspond to probe ids).
duplicateRemoverMethod	a single character value specifying the method to remove the duplicates. The current version provides "min" (minimum), "max" (maximum), "average". Further details in 'duplicateRemover' function at the HTSanalyzeR package.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

TNI-class TNA-class

Examples

```
data(dt4rtn)

tfs4test<-c("PTTG1","E2F2","FOXM1","E2F3","RUNX2")
rtni <- new("TNI", gexp=dt4rtn$gexp, transcriptionFactors=dt4rtn$tfs[tfs4test])

## Not run:

rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
rtni<-tni.dpi.filter(rtni)
rtna<-tni2tna.preprocess(rtni, phenotype=dt4rtn$pheno, hits=dt4rtn$hits, phenoIDs=dt4rtn$phenoIDs)

## End(Not run)
```

Index