Package 'RTN'

October 12, 2016

Type Package

Title Reconstruction of transcriptional networks and analysis of master regulators

Version 1.10.0

Date 2016-03-11

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 RedeR, minet, snow, limma, data.table, ff, car, IRanges

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. Additionally, master regulators can be linked to genetic markers using eQTL/VSE analysis, taking advantage of the haplotype block structure mapped to the human genome in order to explore risk-associated SNPs identified in GWAS studies.

License Artistic-2.0

biocViews NetworkInference, NetworkAnalysis, NetworkEnrichment, GeneRegulation, GeneExpression, GraphAndNetwork, GeneSetEnrichment,GeneticVariability,SNP

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

Collate ClassUnions.R AllChecks.R AllClasses.R AllGenerics.R AllSupplementsTNA.R AllSupplementsTNI.R AllSupplementsAVS.R AllPlotsTNA.R AllPlotsAVS.R TNA-methods.R TNI-methods.R AVS-methods.R

LazyLoad yes

NeedsCompilation no

R topics documented:

RTN-package . . . 2
AVS-class . . . 4
avs.evse . . . 5
avs.get . . . 7
avs.plot1 . . . 8
avs.plot2 . . . 9
avs.preprocess . . . 10
avs.vse . . . 11
RTN.data . . . 12
TNA-class . . . 13
tna.get . . . 14
tna.graph . . . 16
tna.gsea1 . . . 17
tna.gsea2 . . . 19
tna.mra . . . 21
tna.overlap . . . 22
tna.plot.gsea1 . . . 23
tna.plot.gsea2 . . . 25
tna.shadow . . . 27
tna.synergy . . . 29
TNI-class . . . 30
tni.bootstrap . . . 32
tni.conditional . . . 33
tni.dpi.filter . . . 36
tni.get . . . 37
tni.graph . . . 39
tni.permutation . . . 40
tni.preprocess . . . 42
tni2tna.preprocess . . . 43

Index **45**

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

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. Additionally, master regulators can be linked to genetic markers using eQTL/VSE analysis, taking advantage of the haplotype block structure mapped to the human genome in order to explore risk-associated SNPs identified in GWAS studies.

Details

Package:	RTN
Type:	Package
Depends:	R (>= 2.15), methods, igraph
Imports:	RedeR, minet, snow, limma, data.table, ff, car, IRanges
Suggests:	HTSanalyzeR, RUnit, BiocGenerics
License:	Artistic-2.0
biocViews:	NetworkInference, GeneRegulation, GeneExpression, GraphsAndNetworks
Collate:	ClassUnions.R, AllChecks.R, AllClasses.R, AllGenerics.R, AllSupplements.R, AllPlots.R, TNA-methods.R, TN
LazyLoad:	yes

Index

TNI-class:	an S4 class for Transcriptional Network Inference.
tni.preprocess:	a preprocessing method for objects of class TNI.
tni.permutation:	inference of transcriptional networks.
tni.bootstrap:	inference of transcriptional networks.
tni.dpi.filter:	data processing inequality (DPI) filter.
tni.conditional:	conditional mutual information analysis.
tni.get:	get information from individual slots in a TNI object.
tni.graph:	compute a graph from TNI objects.
tni2tna.preprocess:	a preprocessing method for objects of class TNI.
TNA-class:	an S4 class for Transcriptional Network Analysis.
tna.mra:	master regulator analysis (MRA) over a list of regulons.
tna.overlap:	overlap analysis over a list of regulons.
tna.gsea1:	one-tailed gene set enrichment analysis (GSEA) over a list of regulons.
tna.gsea2:	two-tailed gene set enrichment analysis (GSEA) over a list of regulons.
tna.synergy:	synergy analysis over a list of regulons.
tna.shadow:	shadow analysis over a list of regulons.
tna.get:	get information from individual slots in a TNA object.
tna.plot.gsea1:	plot results from the one-tailed GSEA.
tna.plot.gsea2:	plot results from the two-tailed GSEA.
AVS-class:	an S4 class to do enrichment analyses in associated variant sets (AVSs).
avs.preprocess:	build associated variant sets for objects of class AVS.
avs.vse:	variant set enrichment analysis.
avs.evse:	an eQTL/VSE pipeline for variant set enrichment analysis.
avs.get:	get information from individual slots in an AVS object.
avs.plot1:	plot results from AVS methods, single plots.
avs.plot2:	plot results from AVS methods, multiple plots.

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.

`AVS-class` *Class "AVS": an S4 class for variant set enrichment analysis.*

Description

This S4 class includes a series of methods to do enrichment analyses in Associated Variant Sets (AVSs).

Objects from the Class

Objects can be created by calls of the form `new("AVS", markers)`.

Slots

`markers`**:** Object of class `"character"`, a data frame, a 'BED file' format with rs# markers mapped to the same genome build of the LD source in the RTNdata package.

`validatedMarkers`**:** Object of class `"data.frame"`, a data frame with genome positions of the validated markers.

`variantSet`**:** Object of class `"list"`, an associated variant set.

`randomSet`**:** Object of class `"list"`, a random associated variant set.

`para`**:** Object of class `"list"`, a list of parameters for variant set enrichment analysis.

`results`**:** Object of class `"list"`, a list of results (see return values in the AVS methods).

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

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

Methods

avs.preprocess `signature(object = "AVS")`: see `avs.preprocess`

avs.vse `signature(object = "AVS")`: see `avs.vse`

avs.evse `signature(object = "AVS")`: see `avs.evse`

avs.get `signature(object = "AVS")`: see `avs.get`

Author(s)

Mauro Castro

See Also

TNA-class.

Examples

```
## Not run:
#This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMap.rel27)
data(bcarisk) #mapped to the same genome build of the RTNdata!
avs <- new("AVS", markers=bcarisk)

## End(Not run)
```

---

avs.evse *An eQTL/VSE pipeline for variant set enrichment analysis.*

Description

The VSE method (avs.vse) provides a robust framework to cope with the heterogeneous structure of haplotype blocks, and has been designed to test enrichment in cistromes and epigenomes. In order to extend the variant set enrichment to genes this pipeline implements an additional step using expression quantitative trait loci (eQTLs).

Usage

```
avs.evse(object, annotation, gxdata, snpdata, maxgap=250000, pValueCutoff=0.05,
boxcox=TRUE, lab="annotation", glist=NULL, minSize=100,  fineMapping=TRUE,
verbose=TRUE)
```

Arguments

object	an object. When this function is implemented as the S4 method of class AVS-class, this argument is an object of class 'AVS'.
annotation	a data frame with genomic annotations listing chromosome coordinates to which a particular property or function has been attributed. It should include the following columns: <CHROM>, <START>, <END> and <ID>. The <ID> column can be any genomic identifier, while values in <CHROM> should be listed in ['chr1', 'chr2', 'chr3' ..., 'chrX']. Both <START> and <END> columns correspond to chromosome positions mapped to the human genome assembly used to build the AVS object (see avs.preprocess).
gxdata	object of class "matrix", a gene expression matrix.
snpdata	either an object of class "matrix" or "ff", a single nucleotide polymorphism (SNP) matrix.
maxgap	a single integer value specifying the max distant (bp) between the AVS and the annotation used to compute the eQTL analysis.

pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
boxcox	a single logical value specifying to use Box-Cox procedure to find a transformation of the null that approaches normality (when boxcox=TRUE) or not (when boxcox=FALSE). See powerTransform and bcPower.
lab	a single character value specifying a name for the annotation dataset (this option is overrided if 'glist' is used).
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This option can be used to run a batch mode for gene sets and regulons.
minSize	if 'glist' is provided, this argument is a single integer or numeric value specifying the minimum number of elements for each gene set in the 'glist'. Gene sets with fewer than this number are removed from the analysis. if 'fineMapping=FALSE', an alternative min size value can be provided as a vector of the form c(minSize1, minSize2) used to space the null distributions (see 'fineMapping').
fineMapping	if 'glist' is provided, this argument is a single logical value specifying to compute individual null distributions, sized for each gene set (when fineMapping=TRUE). This option has a significant impact on the running time required to perform the computational analysis, especially for large gene set lists. When fineMapping=FALSE, a low resolution analysis is performed by pre-computing a fewer number of null distributions of different sizes (spaced by 'minSize'), and then used as a proxy of the nulls.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
#This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMap.rel27)
data(bcarisk)
avs <- new("AVS", markers=bcarisk)
avs <- avs.preprocess(avs, nrand=100)
avs.get(avs)

## End(Not run)
```

avs.get *Get information from individual slots in an AVS object.*

Description

Get information from individual slots in an AVS object.

Usage

```
avs.get(object, what="summary", report=FALSE, pValueCutoff=NULL)
```

Arguments

object	an object of class 'AVS' AVS-class.
what	a single character value specifying which information should be retrieved from the slots. Options: 'markers', 'validatedMarkers', 'variantSet', 'randomSet', 'linkedMarkers', 'randomMarkers', 'vse', 'evse', 'summary' and 'status'.
report	a single logical value indicating whether to return results from 'vse', 'evse' as a consolidated table (if TRUE), or as they are (if FALSE).
pValueCutoff	an optional single numeric value specifying the cutoff to retrive results for p-values considered significant.

Value

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

Author(s)

Mauro Castro

Examples

```
## Not run:
#This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMap.rel27)
data(bcarisk)
avs <- new("AVS", markers=bcarisk)
avs <- avs.preprocess(avs, nrand=100)
avs.get(avs)

## End(Not run)
```

avs.plot1 *Plot results from AVS methods, single plots.*

Description

This function takes an AVS object and plots results from the VSE and EVSE methods.

Usage

```
avs.plot1(object, what="vse", fname=what, ylab="genomic annotation",
xlab="Number of clusters mapping to genomic annotation", breaks="Sturges",
maxy=200, pValueCutoff=1e-2, width=8, height=3)
```

Arguments

object	an object of class 'AVS' AVS-class.
what	a character value specifying which analysis should be used. Options: "vse" and "evse".
fname	a character value specifying the name of output file.
ylab	a character value specifying the y-axis label.
xlab	a character value specifying the x-axis label.
breaks	breaks in the histogram, see hist function.
maxy	a numeric value specifying the max y-limit.
pValueCutoff	a numeric value specifying the cutoff for p-values considered significant.
width	a numeric value specifying the width of the graphics region in inches.
height	a numeric value specifying the height of the graphics region in inches.

Author(s)

Mauro Castro

Examples

```
###
```

avs.plot2 *Plot results from AVS methods, multiple plots.*

Description

This function takes an AVS object and plots results from the VSE and EVSE methods.

Usage

```
avs.plot2(object, what="evse", fname=what, width=14, height=2.5, rmargin=1,
bxseq=seq(-4,8,2), decreasing=TRUE, ylab="Annotation",
xlab="Clusters of risk-associated and linked SNPs", tfs=NULL)
```

Arguments

object	an object of class 'AVS' AVS-class.
what	a single character value specifying which analysis should be used. Options: "vse" and "evse".
fname	a character value specifying the name of output file.
height	a numeric value specifying the height of the graphics region in inches.
width	a numeric value specifying the width of the graphics region in inches.
rmargin	a numeric value specifying the right margin size in inches.
bxseq	a numeric vector specifying which x-axis tickpoints are to be drawn.
decreasing	a logical value, used to sort by EVSE scores.
ylab	a character value specifying the y-axis label.
xlab	a character value specifying the x-axis label (on the top of the grid image).
tfs	an optional vector with annotation identifiers (e.g. transcription factor).

Author(s)

Mauro Castro

Examples

```
###
```

`avs.preprocess` *A preprocessing function for objects of class AVS.*

Description

This is a generic function.

Usage

```
avs.preprocess(object, nrand=1000, mergeColinked=TRUE, reldata="RTNdata.LDHapMap.rel27", snpop="all"
```

Arguments

`object`	an object. When this function is implemented as the S4 method of class `AVS-class`, this argument is an object of class 'AVS'.
`nrand`	a single integer value specifying the size to the random variant set.
`mergeColinked`	a single logical value specifying to merge co-linked markers eventually present in more than one cluster in the AVS (when mergeColinked=TRUE) or not (when mergeColinked=FALSE).
`reldata`	a single character value specifying the RTNdata package used to build the associated variant set. Current options are "RTNdata.LDHapMap.rel27" and "RTNdata.LD1000g.rel20130502" (for additional details on the Linkage Disequilibrium data, see the RTNdata package documentation).
`snpop`	a single character value specifying the universe size used to build the null distributions for the VSE/EVSE analysis (i.e. random AVSs). Options: 'all' or 'ld'. The 1st option represents all the SNPs genotyped in the Linkage Disequilibrium (LD) dataset, while the 2nd represents all the SNPs observed after the LD threshold is applyed. Alternativelly, 'snpop' can be a customized 'BED file' data frame with rs# identifiers.
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

`AVS-class`

Examples

```
## Not run:
#This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMap.rel27)
```

```
data(bcarisk)
avs <- new("AVS", markers=bcarisk)
avs <- avs.preprocess(avs, nrand=100)
avs.get(avs)

## End(Not run)
```

---

avs.vse *Variant set enrichment (VSE) analysis.*

Description

The VSE method tests the enrichment of an AVS for a particular trait in a genomic annotation.

Usage

```
avs.vse(object, annotation, maxgap=0, pValueCutoff=0.05, boxcox=TRUE,
lab="annotation", glist=NULL, minSize=100, verbose=TRUE)
```

Arguments

object	an object. When this function is implemented as the S4 method of class AVS-class, this argument is an object of class 'AVS'.
annotation	a data frame with genomic annotations listing chromosome coordinates to which a particular property or function has been attributed. It should include the following columns: <CHROM>, <START>, <END> and <ID>. The <ID> column can be any genomic identifier, while values in <CHROM> should be listed in ['chr1', 'chr2', 'chr3' ..., 'chrX']. Both <START> and <END> columns correspond to chromosome positions mapped to the human genome assembly used to build the AVS object (see avs.preprocess).
maxgap	a single integer value specifying the max distant (bp) between the AVS and the annotation used to compute the enrichment analysis.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
boxcox	a single logical value specifying to use Box-Cox procedure to find a transformation of the null that approaches normality (when boxcox=TRUE) or not (when boxcox=FALSE). See powerTransform and bcPower.
lab	a single character value specifying a name for the annotation dataset (this option is overrided if 'glist' is used).
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This option can be used to run a batch mode for gene sets and regulons.
minSize	if 'glist' is provided, this argument is a single integer or numeric value specifying the minimum number of elements for each gene set in the 'glist'. Gene sets with fewer than this number are removed from the analysis.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
#This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMap.rel27)
data(bcarisk)
avs <- new("AVS", markers=bcarisk)
avs <- avs.preprocess(avs, nrand=100)
avs.get(avs)

## End(Not run)
```

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, transcription`

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 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`. `tni2tna.preprocess`.

Examples

```
## see 'tni2tna.preprocess' method!
```

---

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,
idkey=NULL)
```

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', 'gsea1', 'gsea2','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 'gsea1', 'gsea2', '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 report 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 preprocessing analysis. It takes effect on 'mra', 'gsea', 'overlap', 'synergy' and 'shadow' options.
idkey	an optional single character value specifying an ID name from the available 'TNA' annotation to be used as alias for data query outputs (obs. it has no effect on consolidated tables).

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", amapCutoff=NULL, ...)
```

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 the 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.
...	additional arguments passed to tna.graph function.

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 tna.plot.gsea1

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(3, "SOCK"))
rtna <- tna.gsea1(rtna,stepFilter=FALSE)
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 tna.plot.gsea2

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(3, "SOCK"))
rtna <- tna.gsea2(rtna,stepFilter=FALSE)
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, labPheno="tna", file=labPheno, filepath=".", regulon.order="size",
ntop=NULL, tfs=NULL, ylimPanels=c(0.0,3.5,0.0,0.8), heightPanels=c(1,1,3),
width=5, height=4, ylabPanels=c("Phenotype","Regulon","Enrichment score"),
xlab="Position in the ranked list of genes", alpha=0.5,
sparsity=10, autoformat=TRUE, ...)
```

Arguments

object	an object of class 'TNA' TNA-class.
file	a character string naming a file.
filepath	a single character value specifying where to store GSEA figures.
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).
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 value specifying the the title for the x axis.
labPheno	a single character value 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

See Also

tna.gsea1

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.gsea1(rtna, stepFilter=FALSE)

# plot available GSEA results
tna.plot.gsea1(rtna, 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, labPheno="tna", file=labPheno, filepath=".", regulon.order="size",
ntop=NULL, tfs=NULL, ylimPanels=c(-3.0,3.0,-0.5,0.5), heightPanels=c(2.0,0.8,5.0), width=2.7,
height=3.0, ylabPanels=c("Phenotype","Regulon","Enrichment score"),
xlab="Position in the ranked list of genes", alpha=1.0,
sparsity=10, autoformat=TRUE, ...)
```

Arguments

`object`	an object of class 'TNA' TNA-class.
`file`	a character string naming a file.
`filepath`	a single character value specifying where to store GSEA2 figures.
`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).
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

See Also

tna.gsea2

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, 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 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(3, "SOCK"))
rtni<-tni.permutation(rtni)
rtni<-tni.bootstrap(rtni)
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 to the gene expression of a modulator.

Usage

```
tni.conditional(object, modulators=NULL, tfs=NULL, sampling=35, pValueCutoff=0.01,
pAdjustMethod="bonferroni", minRegulonSize=15, minIntersectSize=5,
miThreshold="md", prob=0.99, pwtransform=FALSE, medianEffect=FALSE,
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).
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	a single character value specifying the underlying distribution used to estimate the mutual information threshold. Options: 'md' and 'md.tf'. In the 1st case, 'miThreshold' is estimated from a pooled null distribution representing random modulators, while in the 2nd case a specific mutual information threshold is estimated for each TF conditioned on the random modulators. In the two options the 'miThreshold' is estimated by permutation analysis (see 'prob'). Alternatively, users can either provide a custom mutual information threshold or a numeric vector with lower (a) and upper (b) bounds for the differential mutual information analysis (e.g. 'c(a,b)').
prob	a probability value in [0,1] used to estimate the 'miThreshold' based on the underlying quantile distribution.
pwtransform	a single logical value specifying whether a Box-Cox power transformation should be used to improve data symmetry (this procedure might be carefully used as a complementary test to check the validity of the inferred associations).
medianEffect	a single logical value specifying whether to assess the median effect of each modulator. This global statistics does not affect the inferential process over single TF-target interactions. This method is still experimental, it can be used as a complementary analysis to chek the overall modulation effect onto all targets listed in a given regulon (this step may require substantial computation time).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)
...	additional arguments passed to tna.graph function.

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,
idkey=NULL)
```

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 report regulators with 'names' (when reportNames=TRUE) or not (when reportNames=FALSE). This option takes effect on 'cdt' option if regulators are named with alternative identifiers.
idkey	an optional single character value specifying an ID name from the available 'TNI' annotation to be used as alias for data query outputs (obs. it has no effect on consolidated tables).

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",
amapCutoff=NULL, ntop=NULL, ...)
```

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", "mmap" and "mmapDetailed". 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" and "mmapDetailed" return 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 the 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.
ntop	when gtype="mmapDetailed", ntop is an optional single integer value (>=1) specifying the number of TF's targets that should be used to compute the modulated map. The n targets is derived from the top ranked TF-target interations, as defined in the mutual information analysis used to construct the regulon set.
...	additional arguments passed to tni.graph function.

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(3, "SOCK"))
rtni<-tni.permutation(rtni)
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>. Additional annotation can be included in the data frame and will be passed to the resulting TNI object. Furthermore, in order to eventually use the TNI object in `AVS-class` methods, it should also include chromosome coordinates: columns <CHROM>, <START> and <END>. Values in <CHROM> should be listed in [chr1, chr2, chr3, ..., chrX], while <START> and <END> correspond to chromosome positions (see `avs.evse`).
`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", verbose=TRUE)
```

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

∗Topic **GSEA2**
tna.plot.gsea2, 25
∗Topic **GSEA**
tna.get, 14
tna.gsea1, 17
tna.gsea2, 19
tna.plot.gsea1, 23
∗Topic **RMA**
tna.mra, 21
∗Topic **VSE**
avs.plot1, 8
avs.plot2, 9
∗Topic **classes**
AVS-class, 4
TNA-class, 13
TNI-class, 30
∗Topic **dataset**
RTN.data, 12
∗Topic **methods**
avs.evse, 5
avs.get, 7
avs.preprocess, 10
avs.vse, 11
tna.graph, 16
tni.bootstrap, 32
tni.conditional, 33
tni.dpi.filter, 36
tni.get, 37
tni.graph, 39
tni.permutation, 40
tni.preprocess, 42
tni2tna.preprocess, 43
∗Topic **overlap**
tna.overlap, 22
∗Topic **package**
RTN-package, 2
∗Topic **shadow**
tna.shadow, 27
∗Topic **synergy**
tna.synergy, 29

aracne, *36*
AVS-class, *3*, 4
avs.evse, *3, 4*, 5, *42*
avs.evse,AVS-method *(*AVS-class*)*, 4
avs.get, *3, 4*, 7
avs.get,AVS-method *(*AVS-class*)*, 4
avs.plot1, *3*, 8
avs.plot2, *3*, 9
avs.preprocess, *3–5*, 10, *11*
avs.preprocess,AVS-method *(*AVS-class*)*, 4
avs.vse, *3–5*, 11
avs.vse,AVS-method *(*AVS-class*)*, 4

bcPower, *6*, *11*

dt4rtn *(*RTN.data*)*, 12

hist, *8*

makeCluster, *32*, *41*

powerTransform, *6*, *11*

RTN *(*RTN-package*)*, 2
RTN-package, 2
RTN.data, 12

TNA-class, *3*, 13
tna.get, *3*, *14*, 14, *18*, *20, 21*, *23*, *28*, *30*
tna.get,TNA-method *(*TNA-class*)*, 13
tna.graph, *14*, 16
tna.graph,TNA-method *(*TNA-class*)*, 13
tna.gsea1, *3*, *13, 14*, 17, *24*, *27*, *29*, *43*
tna.gsea1,TNA-method *(*TNA-class*)*, 13
tna.gsea2, *3*, *13, 14*, 19, *26*
tna.gsea2,TNA-method *(*TNA-class*)*, 13
tna.mra, *3*, *13, 14*, *18*, *20*, 21, *43*
tna.mra,TNA-method *(*TNA-class*)*, 13
tna.overlap, *3*, *13, 14*, 22, *43*

tna.overlap,TNA-method *(*TNA-class*)*, 13
tna.plot.gsea1, *3*, *18*, 23
tna.plot.gsea2, *3*, *20*, 25
tna.shadow, *3*, *13, 14*, 27, *28*, *30*
tna.shadow,TNA-method *(*TNA-class*)*, 13
tna.synergy, *3*, *13, 14*, 29
tna.synergy,TNA-method *(*TNA-class*)*, 13
TNI-class, *3*, 30
tni.bootstrap, *3*, *31*, 32, *34*, *36*, *43*
tni.bootstrap,TNI-method *(*TNI-class*)*, 30
tni.conditional, *3*, *13*, *31*, 33, *39*
tni.conditional,TNI-method *(*TNI-class*)*, 30
tni.dpi.filter, *3*, *31*, *34*, 36, *43*
tni.dpi.filter,TNI-method *(*TNI-class*)*, 30
tni.get, *3*, *31, 32*, *35, 36*, 37, *41*
tni.get,TNI-method *(*TNI-class*)*, 30
tni.graph, *3*, *31*, 39
tni.graph,TNI-method *(*TNI-class*)*, 30
tni.permutation, *3*, *31, 32*, *34*, *36*, 40, *43*
tni.permutation,TNI-method *(*TNI-class*)*, 30
tni.preprocess, *3*, *31*, 42
tni.preprocess,TNI-method *(*TNI-class*)*, 30
tni2tna.preprocess, *3*, *14*, *31*, 43
tni2tna.preprocess,TNI-method *(*TNI-class*)*, 30