Package 'RTN'

November 12, 2026

Type Package

Title RTN: Reconstruction of Transcriptional regulatory Networks and analysis of regulons

Version 2.36.0

Author
Clarice Groeneveld [ctb], Gordon Robertson [ctb], Xin Wang [aut], Michael Fletcher [aut], Florian Markowetz [aut], Kerstin Meyer [aut], and Mauro Castro [aut]

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

Depends R (>= 3.6.3), methods,

Imports RedeR, minet, viper, mixtools, snow, stats, limma, data.table, IRanges, igraph, S4Vectors, SummarizedExperiment, car, pwr, pheatmap, grDevices, graphics, utils

Suggests RUnit, BiocGenerics, BiocStyle, knitr, rmarkdown

Description A transcriptional regulatory network (TRN) consists of a collection of transcription factors (TFs) and the regulated target genes. TFs are regulators that recognize specific DNA sequences and guide the expression of the genome, either activating or repressing the expression the target genes. The set of genes controlled by the same TF forms a regulon. This package provides classes and methods for the reconstruction of TRNs and analysis of regulons.

License Artistic-2.0

biocViews Transcription, Network, NetworkInference, NetworkEnrichment, GeneRegulation, GeneExpression, GraphAndNetwork, GeneSetEnrichment, GeneticVariability

VignetteBuilder knitr

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 AllPlotsTNI.R TNA-methods.R TNI-methods.R AVS-methods.R TNI-pruning.R TNI-annotation.R TNI-subgroups.R

LazyLoad yes

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

git_branch RELEASE_3_23

git_last_commit 451bf4e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

RTN-package . . . 3
AVS-class . . . 4
avs.evse . . . 5
avs.get . . . 8
avs.pevse . . . 9
avs.plot1 . . . 12
avs.plot2 . . . 13
avs.rvse . . . 14
avs.vse . . . 16
RTN.data . . . 19
TNA-class . . . 20
tna.get . . . 22
tna.gsea1 . . . 24
tna.gsea2 . . . 26
tna.mra . . . 28
tna.plot.gsea1 . . . 29
tna.plot.gsea2 . . . 31
TNI-class . . . 33
tni.alpha.adjust . . . 34
tni.annotate.regulons . . . 35
tni.annotate.samples . . . 37
tni.area3 . . . 38
tni.bootstrap . . . 40
tni.conditional . . . 41
tni.constructor . . . 44
TNI.data . . . 45
tni.dpi.filter . . . 46
tni.get . . . 47
tni.graph . . . 49
tni.gsea2 . . . 50
tni.overlap.genesets . . . 52
tni.permutation . . . 54
tni.plot.checks . . . 56
tni.plot.sre . . . 57
tni.preprocess . . . 59
tni.prune . . . 60
tni.regulon.summary . . . 61
tni.replace.samples . . . 62

tni.sre 63
tni2tna.preprocess 64
upgradeTNA 66
upgradeTNI 66

Index **68**

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

Description

A transcriptional regulatory network (TRN) consists of a collection of transcription factors (TFs) and the regulated target genes. TFs are regulators that recognize specific DNA sequences and guide the expression of the genome, either activating or repressing the expression the target genes. The set of genes controlled by the same TF forms a regulon. This package provides classes and methods for the reconstruction of TRNs and analysis of regulons.

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.
tni.gsea2:	compute regulon activity.
tni.prune:	prune regulons to remove redundant targets for regulon activity analysis.
tni.sre:	subgroup regulon difference analysis.
tni.plot.sre:	plot subgroup regulon rnrichment .
tni.regulon.summary:	return a summary of network and regulons.
tni.plot.checks:	plot regulon target counts.
tni.alpha.adjust:	adjust the significance level for two datasets.
tni.replace.samples:	replace samples of an existing TNI-class 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.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.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.vse:	variant set enrichment analysis.
avs.evse:	an eQTL/VSE pipeline for variant set enrichment analysis.

avs.pevse:	an EVSE pipeline using precomputed eQTLs.
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 by typing `help.start()` and selecting the RTN package from the menu.

Author(s)

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

References

Fletcher M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Nature Communications, 4:2464, 2013.

Castro M.A.A. et al., *Regulators of genetic risk of breast cancer identified by integrative network analysis.* Nature Genetics, 48:12-21, 2016.

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

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

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

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

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.LDHapMapRel27.hg18)
data(bcarisk, package = "RTNdata.LDHapMapRel27.hg18")
avs <- avs.preprocess.LDHapMapRel27.hg18(bcarisk, nrand=100)

## 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, glist=NULL, maxgap=250, minSize=100,
pValueCutoff=0.05, pAdjustMethod="bonferroni", boxcox=TRUE,
fineMapping=TRUE, verbose=TRUE)
```

Arguments

object	an object of class AVS-class.
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.
gxdata	object of class "matrix", a gene expression matrix.
snpdata	either an object of class "matrix" or "ff", a single nucleotide polymorphism (SNP) matrix.
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This list is used to run a batch mode for gene sets and regulons.
maxgap	a single integer value specifying the max distant (kb) between the AVS and the annotation used to compute the eQTL analysis.
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').
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).
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.
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).

Value

a data frame in the slot "results", see 'what' options in avs.get.

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
# This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMapRel27.hg18)
library(Fletcher2013b)
library(TxDb.Hsapiens.UCSC.hg18.knownGene)

##################################################
### Build AVS and random AVSs (mapped to hg18)
##################################################

#--- step 1: load 'risk SNPs' data (e.g. BCa risk SNPs from the GWAS catalog)
data(bcarisk, package = "RTNdata.LDHapMapRel27.hg18")

#--- step 2: build an AVS and 1000 matched random AVSs for the input 'risk SNPs'
bcavs <- avs.preprocess.LDHapMapRel27.hg18(bcarisk, nrand=1000)

##################################################
### Example of EVSE analysis for sets of genomic
### annotations (e.g. regulons, gene sets, etc.)
##################################################

#--- step 1: load a precomputed AVS (same 'bcavs' object as above!)
data(bcavs, package="RTNdata.LDHapMapRel27.hg18")

#--- step 2: load genomic annotation for all genes
genemap <- as.data.frame(genes(TxDb.Hsapiens.UCSC.hg18.knownGene))
genemap <- genemap[,c("seqnames","start","end","gene_id")]
colnames(genemap) <- c("CHROM","START","END","ID")

#--- step 3: load a TNI object, or any other source of regulons (e.g. gene sets)
#--- and prepare a gene set list
#--- (gene ids should be the same as in the 'genemap' object)
data("rtni1st")
glist <- tni.get(rtni1st,what="refregulons",idkey="ENTREZ")
glist <- glist[ c("FOXA1","GATA3","ESR1") ] #reduce the list for demonstration!

#--- step 4: input matched variation and gene expression datasets!
#--- here we use two "toy" datasets for demonstration purposes only.
data(toy_snpdata, package="RTNdata.LDHapMapRel27.hg18")
data(toy_gxdata, package="RTNdata.LDHapMapRel27.hg18")

#--- step 5: run the avs.evse pipeline
bcavs<-avs.evse(bcavs, annotation=genemap, gxdata=toy_gxdata,
                snpdata=toy_snpdata, glist=glist, pValueCutoff=0.01)

#--- step 6: generate the EVSE plots
avs.plot2(bcavs,"evse", height=2.5, width.panels=c(1,2), rmargin=0)
```

```
### NOTE REGARDING THIS EXAMPLE ####
#- This example is for demonstration purposes only, using toy datasets.
#- Any eventual positive/negative associations derived from these datasets
#- are not comparable with the original studies that described the method
#- (doi: 10.1038/ng.3458; 10.1038/ncomms3464).
####################################


## 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', 'rvse', 'pevse', 'annotation.vse', 'annotation.evse', 'annotation.rvse', 'annotation.pevse', '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.LDHapMapRel27)
data(bcarisk, package="RTNdata.LDHapMapRel27")
bcavs <- avs.preprocess.LDHapMapRel27(bcarisk, nrand=1000)
avs.get(avs)

## End(Not run)
```

`avs.pevse` *An EVSE pipeline using precomputed eQTLs.*

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 precomputed expression quantitative trait loci (eQTLs).

Usage

```
avs.pevse(object, annotation, eqtls, glist, maxgap=250, minSize=100,
pValueCutoff=0.05, pAdjustMethod="bonferroni", boxcox=TRUE,
verbose=TRUE)
```

Arguments

`object`	an object of class 'AVS' (see `AVS-class`).
`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.
`eqtls`	object of class 'data.frame' with at least two columns, including the following column names: <RSID> and <GENEID>.
`glist`	a list with character vectors mapped to the 'annotation' data via <ID> column. This list is used to run a batch mode for gene sets and regulons.
`maxgap`	a single integer value specifying the max distant (kb) used to assign the eQTLs in the 'eqtls' object.
`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').
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).
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.
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 'what' options in avs.get.

Author(s)

Mauro Castro, Steve Booth

See Also

AVS-class

Examples

```
## Not run:

# This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMapRel27.hg18)
library(Fletcher2013b)
library(TxDb.Hsapiens.UCSC.hg18.knownGene)

##################################################
### Example of EVSE analysis for sets of genomic
### annotations (e.g. regulons, gene sets, etc.)
##################################################

#--- step 1: load a precomputed AVS
data(bcavs, package="RTNdata.LDHapMapRel27.hg18")

#--- step 2: load genomic annotation for all genes
genemap <- as.data.frame(genes(TxDb.Hsapiens.UCSC.hg18.knownGene))
genemap <- genemap[,c("seqnames","start","end","gene_id")]
colnames(genemap) <- c("CHROM","START","END","ID")

#--- step 3: load a TNI object (or any other source of regulons)
#--- and prepare a gene set list.
#--- Note: gene ids should be the same as in the 'genemap' object.
data("rtni1st")
glist <- tni.get(rtni1st,what="refregulons",idkey="ENTREZ")
```

```
glist <- glist[ c("FOXA1","GATA3","ESR1") ] #reduce the list for demonstration!

#--- step 4: load precomputed eQTLs
#--- Please note that the input data should represent eQTLs from genome-wide
#--- calls, that is, the universe size should cover 'all SNPs' vs. 'all genes'.
#--- The correct represetation of universe size is essential to build the
#--- null distributions. Or, to put it another way, the eQTL analysis
#--- should unbiasedly test linked and random markers from the AVS.
#--- In this example it represents 2029 + 966240 SNPs:

lkMarkers <- avs.get(bcavs,what="linkedMarkers")
length(lkMarkers) # i.e. 2029 risk associated and linked SNPs

rdMarkers <- avs.get(bcavs,what="randomMarkers")
length(rdMarkers) # i.e. 966240 random SNPs

#--- Now we prepare a 'toy' dataset for demonstration purposes only
#--- by picking (naively) SNPs within 250 kb window around the
#--- genomic annotation.

## load HapMap SNPs (release 27) mapped to hg18
data("popsnp2")

## map SNPs to the genomic annotation
query <- with(popsnp2,
              GRanges(chrom, IRanges(position, position),
              id=rsid)
              )
subject <- with(genemap,
                GRanges(CHROM, IRanges(START, END),
                id=ID)
                )
hits <- findOverlaps(query,subject, maxgap = 250000)

## reduce 'hits' just for demonstration
hits <- hits[sort(sample(length(hits), 50000))]

## build a 'toy_eqtls' data frame
toy_eqtls <- data.frame(rsid = popsnp2$rsid[from(hits)],
                        geneid = genemap$ID[to(hits)])

#--- step 5: run the 'avs.pevse' pipeline
#--- important: set 'maxgap' to the same searching window used
#--- in the dQTL analysis (e.g. 250kb)
bcavs <- avs.pevse(bcavs, annotation=genemap, glist=glist,
                   eqtls=toy_eqtls, maxgap = 250)

#--- step 6: generate the pEVSE plots
avs.plot2(bcavs,"pevse",height=2.5)


##################################################
#--- parallel version for 'step 5' with SNOW package!
```

```
# library(snow)
# options(cluster=snow::makeCluster(3, "SOCK"))
# bcavs <- avs.pevse(bcavs, annotation=genemap, glist=glist,
#          eqtls=toy_eqtls, maxgap = 250)
# stopCluster(getOption("cluster"))

## ps. as a technical note, the parallel version uses a
## slightly different overlap-based operation, which might
## bring slightly different counts depending on the data
## input organization


## 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.

Value

A plot showing results from the VSE and EVSE methods.

Author(s)

Mauro Castro

Examples

```
# see 'avs.vse' and 'avs.evse' methods.
```

`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,
width.panels=c(1,3), rmargin=1, at.x=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.
`width.panels`	a vector of the form c(width1, width2) specifying the proportional width of the 1st and 2nd panels of the plot, respectively.
`rmargin`	a numeric value specifying the right margin in inches.
`at.x`	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).

Value

A plot showing results from the VSE and EVSE methods.

Author(s)

Mauro Castro

Examples

```
# see 'avs.vse' and 'avs.evse' methods.
```

`avs.rvse` *An rQTL/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 regulon quantitative trait loci (rQTLs).

Usage

```
avs.rvse(object, annotation, regdata, snpdata, glist,
maxgap=250, minSize=100, pValueCutoff=0.05,
pAdjustMethod="bonferroni", boxcox=TRUE, verbose=TRUE)
```

Arguments

`object`	an object of class `AVS-class`.
`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.
`regdata`	object of class "matrix", a regulon activity matrix.
`snpdata`	either an object of class "matrix" or "ff", a single nucleotide polymorphism (SNP) matrix.
`glist`	a list with character vectors mapped to the 'annotation' data via <ID> column. This list is used to run a batch mode for gene sets and regulons.
`maxgap`	a single integer value specifying the max distant (kb) between the AVS and the annotation used to compute the eQTL analysis.
`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.
`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).
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.
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 'what' options in avs.get.

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
# This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMapRel27.hg18)
library(Fletcher2013b)
library(TxDb.Hsapiens.UCSC.hg18.knownGene)

##################################################
### Build AVS and random AVSs (mapped to hg18)
##################################################

#--- step 1: load 'risk SNPs' data (e.g. BCa risk SNPs from the GWAS catalog)
data(bcarisk, package = "RTNdata.LDHapMapRel27.hg18")

#--- step 2: build an AVS and 1000 matched random AVSs for the input 'risk SNPs'
bcavs <- avs.preprocess.LDHapMapRel27.hg18(bcarisk, nrand=1000)

##################################################
### Example of RVSE analysis
##################################################

#--- step 1: load a precomputed AVS (same 'bcavs' object as above!)
data(bcavs, package="RTNdata.LDHapMapRel27.hg18")

#--- step 2: load genomic annotation for all genes
genemap <- as.data.frame(genes(TxDb.Hsapiens.UCSC.hg18.knownGene))
genemap <- genemap[,c("seqnames","start","end","gene_id")]
colnames(genemap) <- c("CHROM","START","END","ID")

#--- step 3: load a TNI object and get a list with regulons
```

```
#--- (ids should be the same as in 'genemap')
data("rtni1st")
glist <- tni.get(rtni1st,what="refregulons",idkey="ENTREZ")
glist <- glist[ c("FOXA1","GATA3","ESR1") ] #reduce the list for demonstration!

#--- step 4: compute single-sample regulon activity
rtni1st <- tni.gsea2(rtni1st, regulatoryElements = c("FOXA1","GATA3","ESR1"))
regdata <- tni.get(rtni1st, what = "regulonActivity")$diff

#--- step 5: load a variation dataset matched with the regulon activity dataset
#--- here we use a "toy" dataset for demonstration purposes only.
data(toy_snpdata, package="RTNdata.LDHapMapRel27")

#--- step 6: check "snpdata"" and "regdata"" alignment (samples on cols)
regdata <- t(regdata)
all(colnames(toy_snpdata)==colnames(regdata))
#TRUE

#--- step 7: run the avs.rvse pipeline
bcavs <- avs.rvse(bcavs,
                  annotation=genemap,
                  regdata=regdata,
                  snpdata=toy_snpdata,
                  glist=glist,
                  pValueCutoff=0.01)

#--- step 8: generate the RVSE plot
avs.plot2(bcavs, "rvse", height=2.5)


## 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, glist=NULL, maxgap=0, minSize=100,
pValueCutoff=0.05, pAdjustMethod="bonferroni", boxcox=TRUE,
verbose=TRUE)
```

Arguments

`object`	an object of class `AVS-class`.

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.
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This list is used to run a batch mode for gene sets and regulons.
maxgap	a single integer value specifying the max distant (kb) between the AVS and the annotation used to compute the enrichment analysis.
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.
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).
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.
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 'what' options in avs.get.

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
# This example requires the RTNdata package! (currently available under request)
library(RTNdata.LDHapMapRel27.hg18)
library(Fletcher2013b)
library(TxDb.Hsapiens.UCSC.hg18.knownGene)

################################################
### Build AVS and random AVSs (mapped to hg18)
################################################

#--- step 1: load 'risk SNPs' data (e.g. BCa risk SNPs from the GWAS catalog)
```

```
data(bcarisk, package = "RTNdata.LDHapMapRel27.hg18")

#--- step 2: build an AVS and 1000 matched random AVSs for the input 'risk SNPs'
bcavs <- avs.preprocess.LDHapMapRel27.hg18(bcarisk, nrand=1000)

################################################
### Example of VSE analysis for ERa and FOXA1
### cistromes (one genomic annotation each time)
################################################

#--- step 1: load a precomputed AVS (same 'bcavs' object as above!)
data(bcavs, package="RTNdata.LDHapMapRel27.hg18")

#--- step 2: load cistrome data from the Fletcher2013b package
#NOTE: Fletcher2013b is a large data package, but only two 'bed files'
#are used to illustrate this analysis (ESR1bdsites and FOXA1bdsites).
#these bed files provide ERa and FOXA1 binding sites mapped by
#ChIP-seq experiments
data(miscellaneous)

#--- step 3: run the avs.vse pipeline
bcavs <- avs.vse(bcavs, annotation=ESR1bdsites$bdsites,
                 pValueCutoff=0.001)
bcavs <- avs.vse(bcavs, annotation=FOXA1bdsites$bdsites,
                 pValueCutoff=0.001)

#--- step 4: generate the VSE plots
avs.plot2(bcavs,"vse",height=2.2, width.panels=c(1,2), rmargin=0)

################################################
### Example of VSE analysis for sets of genomic
### annotations (e.g. regulons, gene sets, etc.)
################################################

#--- step 1: load the precomputed AVS (same 'bcavs' object as above!)
data(bcavs, package="RTNdata.LDHapMapRel27.hg18")

#--- step 2: load genomic annotation for all genes
genemap <- as.data.frame(genes(TxDb.Hsapiens.UCSC.hg18.knownGene))
genemap <- genemap[,c("seqnames","start","end","gene_id")]
colnames(genemap) <- c("CHROM","START","END","ID")

#--- step 3: load a TNI object, or any other source of regulons (e.g. gene sets)
#--- and prepare a gene set list
#--- (gene ids should be the same as in the 'genemap' object)
data("rtni1st")
glist <- tni.get(rtni1st,what="refregulons",idkey="ENTREZ")
glist <- glist[ c("FOXA1","GATA3","ESR1") ] #reduce the list for demonstration!

#--- step 4: run the avs.vse pipeline
bcavs<-avs.vse(bcavs, annotation=genemap, glist=glist, pValueCutoff=0.05)

#--- step 5: generate the VSE plots
```

```
avs.plot2(bcavs,"vse", height=2.5, width.panels=c(1,2), rmargin=0)

### NOTE REGARDING THIS EXAMPLE ####
#- This example is for demonstration purposes only;
#- we recommend using the EVSE/eQTL approach when analysing genes/regulons.
#- Also, the AVS object here is not the same as the one used in the study that
#- extended the method (doi:10.1038/ng.3458), so the results are not comparable;
#- (here fewer risk SNPs are considered, and without an eQTL step).
####################################


## End(Not run)
```

RTN.data *Pre-processed datasets for the RTN package.*

Description

Datasets used to demonstrate RTN main functions.

Format

tniData, **tnaData**, and **tfsData**

Details

The **tniData** and **tnaData** datasets were extracted, pre-processed and size-reduced from Fletcher et al. (2013) and Curtis et al. (2012). They consist of two lists used in the RTN vignettes for demonstration purposes only. The **tniData** list contains the 'expData', 'rowAnnotation' and 'colAnnotation' R objects, while the **tnaData** list contains the 'phenotype', 'phenoIDs' and 'hits' R objects.

The **tfsData** consists of a list with gene annotation for human transcription factors (TFs), compiled from 6 resources (Lambert et. al 2018; Carro et al. 2010; Vaquerizas et al. 2009; D. L. Fulton et al. 2009; Yusuf et al. 2012; and Ravasi et al. 2010), and provided here in the form of 'data frame' objects that include ENTREZ and HGNC gene symbol annotation.

The **pksData** consists of a list with gene annotation for human protein kinases, retrived from the Swiss-Prot Protein Knowledgebase (https://www.uniprot.org/docs/pkinfam), release 2020_02 of 22-Apr-2020.

tniData$expData a named gene expression matrix with 120 samples (a subset from the Fletcher2013b).

tniData$rowAnnotation a data.frame of characters with gene annotation (a subset from the Fletcher2013b).

tniData$colAnnotation a data.frame of characters with sample annotation (a subset from the Fletcher2013b).

tnaData$phenotype a named numeric vector with differential gene expression data.

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

tnaData$hits a character vector with genes differentially expressed.

tfsData$Lambert2018 a data.frame listing TFs from Lambert et. al (2018).

tfsData$Yusuf2012 a data.frame listing TFs from Yusuf et al. (2012).

tfsData$Carro2010 a data.frame listing TFs from Carro et al. (2010).

tfsData$Ravasi2010 a data.frame listing TFs from Ravasi et al. (2010).

tfsData$Fulton2009 a data.frame listing TFs from Fulton et al. (2009).

tfsData$Vaquerizas2009 a data.frame listing TFs from Vaquerizas et al. (2009).

tfsData$all a data.frame listing all TFs.

pksData$pkinfam2020 a data.frame listing human protein kinases.

References

Carro, M.S. et al., *The transcriptional network for mesenchymal transformation of brain tumors.* Nature, 463(7279):318-325, 2010.

Curtis C. et al., *The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups.* Nature, 486(7403):346-352, 2012.

Fletcher, M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Nature Communications, 4:2464, 2013.

Fulton, D.L. et al., *TFCat: the curated catalog of mouse and human transcription factors.* Genome Biology, 10(3):R29, 2009.

Lambert, S.A. et al., *The Human Transcription Factors.* Cell, 172(4):650-665, 2018.

Ravasi, T. et al., *An atlas of combinatorial transcriptional regulation in mouse and man.* Cell, 140(5):744-752, 2010.

Vaquerizas, J.M. et al., *A census of human transcription factors: function, expression and evolution.* Nature Reviews Genetics, 10(4):252-263, 2009.

Yusuf, D. et al., *The Transcription Factor Encyclopedia.* Genome Biology, 13(3):R24, 2012.

Examples

```
data(tniData)
data(tnaData)
data(tfsData)
data(pksData)
```

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

Description

This S4 class includes a series of methods to do enrichment analyses in transcriptional networks.

Objects from the Class

Objects can be created by calls of the form `new("TNA", referenceNetwork, transcriptionalNetwork, regulatoryElements, phenotype, hits)`.

Slots

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

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

`regulatoryElements`: Object of class `"char_Or_NULL"`, a vector of regulatory elements (e.g. 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.

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

`rowAnnotation`: Object of class `"data.frame"`, a data frame with row annotation (e.g. probe-to-gene information).

`colAnnotation`: Object of class `"data.frame"`, a data frame with column annotation (e.g. sample information).

`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.gsea1`, and `tna.gsea2`.

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

`summary`: Object of class `"list"`, a list of summary information for `transcriptionalNetwork`, `regulatoryElements`, `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.gsea1 `signature(object = "TNA")`: see `tna.gsea1`

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

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

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-class`.
`what`	a single character value specifying which information should be retrieved from the slots. Options: 'summary', 'status', 'para', 'pheno', 'hits', 'regulatoryElements', 'tnet', 'refnet', 'regulons', 'refregulons', 'regulons.and.mode', 'refregulons.and.mode', 'rowAnnotation', 'colAnnotation', 'mra', 'gsea1', 'gsea2', 'gsea2summary'.
`order`	a single logical value specifying whether or not the output data should be ordered by significance. Valid only for 'mra', 'gsea1' and 'gsea2' options.
`ntop`	a single integer value specifying to select how many results of top significance from 'mra', 'gsea1' and 'gsea2' 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', 'gsea1' and 'gsea2' 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).

Details

Options for the 'what' argument retrieve the following types of information:

summary A list summarizing parameters and results available in the TNA object.

status A vector indicating the status of each available method in the pipeline.

para A list with the parameters used by each available method in the pipeline.

pheno A numeric vector of phenotypes named by gene identifiers (see `tni2tna.preprocess`).

hits A character vector of gene identifiers for those considered as hits (see tni2tna.preprocess).

regulatoryElements A vector of regulatory elements (e.g. transcription factors).

tnet A data matrix with MI values, evaluated by the DPI filter. MI values are computed between regulators and targets, with regulators on cols and targets on rows. Note that signals (+/-) are assigned to the inferred associations in order to represent the 'mode of action', which is derived from Pearson's correlation between regulators and targets.

refnet A data matrix with MI values (not evaluated by the DPI filter). MI values are computed between regulators and targets, with regulators on cols and targets on rows. Note that signals (+/-) are assigned to the inferred associations in order to represent the 'mode of action', which is derived from Pearson's correlation between regulators and targets.

regulons A list with regulons extracted from the 'tnet' data matrix.

refregulons A list with regulons extracted from the 'refnet' data matrix.

regulons.and.mode A list with regulons extracted from the 'tnet' data matrix, including the assiged 'mode of action'.

refregulons.and.mode A list with regulons extracted from the 'refnet' data matrix, including the assiged 'mode of action'.

rowAnnotation A data frame with probe-to-gene annotation.

colAnnotation A data frame with sample annotation.

mra A data frame with results from the tna.mra analysis pipeline.

gsea1 A data frame with results from the tna.gsea1 analysis pipeline.

gsea2 A data frame with results from the tna.gsea2 analysis pipeline.

Value

Get the slot content from a TNA-class object.

Author(s)

Mauro Castro

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

# run MRA analysis pipeline
```

```
rtna <- tna.mra(rtna)

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

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


## 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,
sizeFilterMethod="posORneg", nPermutations=1000, exponent=1, tnet="dpi",
signature=c("phenotype","hits"), orderAbsValue=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.
`sizeFilterMethod`	a single character value specifying the use of the 'minRegulonSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posPLUSneg". For "posANDneg", the number of both positive and negative targets should be > 'minRegulonSize'; for "posORneg", the number of either positive or negative targets should be > 'minRegulonSize'; and for "posPLUSneg", the number of all targets should be > 'minRegulonSize'.
`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".
signature	a single character value specifying which signature to use in the GSEA method. This could be the 'phenotype' already provided by the user (usually log2 differential expression values) or a 'phenotype' derived from the 'hits'; see tni2tna.preprocess for details on 'phenotype' and 'hits' parameters.
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).
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 'gsea1' option in tna.get.

Author(s)

Mauro Castro, Xin Wang

See Also

TNA-class tna.plot.gsea1

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

#run GSEA1 analysis pipeline
rtna <- tna.gsea1(rtna)

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

# run parallel version with SNOW package!
library(snow)
options(cluster=snow::makeCluster(3, "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,
sizeFilterMethod="posORneg", nPermutations=1000, exponent=1, tnet="dpi",
signature=c("phenotype","hits"), 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.
`sizeFilterMethod`	a single character value specifying the use of the 'minRegulonSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posORnegTrim", "posSUMneg". For "posANDneg", both the number of positive and the number of negative targets must be >= 'minRegulonSize'; for "posORneg", either the number of positive or the number of negative targets must be >= 'minRegulonSize' (both subsets are retained); for "posORnegTrim", either the number of positive or the number of negative targets must be >= 'minRegulonSize', and subsets with size < 'minRegulonSize' are discarded; and for "posSUMneg", the combined number of positive and negative targets must be >= 'minRegulonSize'.
`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".

signature	a single character value specifying which signature to use in the GSEA method. This could be the 'phenotype' already provided by the user (usually log2 differential expression values) or a 'phenotype' derived from the 'hits'; see tni2tna.preprocess for details on 'phenotype' and 'hits' parameters.
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 'gsea2' option in tna.get.

Author(s)

Mauro Castro

See Also

TNA-class tna.plot.gsea2

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

#run GSEA2 analysis pipeline
rtna <- tna.gsea2(rtna)

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

# run parallel version with SNOW package!
library(snow)
options(cluster=snow::makeCluster(3, "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", 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 MRA 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 'rma' option in `tna.get`.

Author(s)

Mauro Castro

See Also

`TNA-class`

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

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

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


## 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="", file="tna_gsea1",
filepath=".", regulon.order="size", ntop=NULL, tfs=NULL,
ylimPanels=c(0.0,3.5,0.0,0.8), heightPanels=c(1,1,3), width=4.4,
height=4, ylabPanels=c("Phenotype","Regulon","Enrichment score"),
xlab="Position in the ranked list of genes", alpha=0.5,
sparsity=10, autoformat=TRUE, plotpdf=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.
plotpdf	a single logical value specifying to whether to plot a PDF file or directly to Viewer.
...	other arguments used by the function pdf.

Value

A plot showing results from the 'tna.gsea1' method.

Author(s)

Mauro Castro

See Also

tna.gsea1

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
```

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

# run GSEA analysis pipeline
rtna <- tna.gsea1(rtna)

# 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="", file="tna_gsea2", 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=2.8, height=3.0,
ylabPanels=c("Phenotype","Regulon","Enrichment score"),
xlab="Position in the ranked list of genes", alpha=1.0,
sparsity=10, autoformat=TRUE, plotpdf=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.
plotpdf	a single logical value specifying to whether to plot a PDF file or directly to Viewer.
...	other arguments used by the function pdf.

Value

A plot showing results from the 'tna.gsea2' method.

Author(s)

Mauro Castro

See Also

tna.gsea2

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

# run GSEA2 analysis pipeline
rtna <- tna.gsea2(rtna)

# plot available GSEA2 results
tna.plot.gsea2(rtna, labPheno="test")

## 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.

Slots

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

`regulatoryElements`: Object of class `"character"`, a vector of regulatory elements (e.g. transcription factors).

`targetElements`: Object of class `"character"`, a vector of target elements (e.g. target genes).

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

`rowAnnotation`: Object of class `"data.frame"`, a data frame with row annotation (e.g. probe-to-gene information).

`colAnnotation`: Object of class `"data.frame"`, a data frame with column annotation (e.g. sample 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`, `regulatoryElements`, `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`

tni.gsea2 `signature(object = "TNI")`: see `tni.gsea2`

tni.area3 `signature(object = "TNI")`: see `tni.area3`

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

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

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

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

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

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

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

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

Author(s)

Mauro Castro

See Also

`TNA-class`

Examples

```
## see 'tni.constructor'!
```

`tni.alpha.adjust` *Adjust the significance level for two datasets.*

Description

When analyzing two datasets that have different numbers of samples, this function can be used to assist the choice of a p value threshold for the tni.permutation() function, in order that RTN will return regulon results that have been generated with similar tradeoffs between Type I and Type II errors for both datasets. Doing this should help ensure that it is reasonable to compare the regulons in the two datasets.

Usage

```
tni.alpha.adjust(nB, nA, alphaA, betaA = 0.2)
```

Arguments

`nB`	a single integer specifying the number of samples in dataset 'B'.
`nA`	a single integer specifying the number of samples in dataset 'A'.
`alphaA`	a single numeric value specifying alpha for dataset 'A' (Type I error probability).
`betaA`	a single numeric value specifying beta for dataset 'A' (Type II error probability).

Value

Significance level for 'nB', given 'nA', 'alphaA' and 'betaA'.

Note

The 'tni.alpha.adjust' function calls the pwr.r.test function, which uses the 'uniroot' function to solve a power equation. The 'uniroot' function aims to find a root in a given interval, searching from lower to upper end-points. As the upper end-point must be strictly larger than the lower end-point, in order to avoid an error when searching the root, 'nA' must be greater than or equal to 'nB' (i.e. 'nB' is expected to be the smallest data set). Also, please note that 'uniroot' eventually will not find the root for the input arguments, especially when searching thresholds of low stringency (we suggest to avoid setting 'alphaA' > 0.01).

Author(s)

Mauro Castro, Gordon Robertson

See Also

pwr.r.test

Examples

```
# estimate 'alphaB' for 'nB', given 'nA', 'alphaA' and 'betaA'
alphaB <- tni.alpha.adjust(nB = 100, nA = 300, alphaA = 1e-5, betaA = 0.2)
```

---

tni.annotate.regulons *Annotate regulons with external gene set collections.*

Description

This function calculates an enrichment score between gene sets and regulons.

Usage

```
tni.annotate.regulons(object, geneSetList, sampleSet = NULL,
regulatoryElements = NULL, minSetSize = 15, sizeFilterMethod="posORneg",
exponent = 1, verbose = TRUE)
```

Arguments

object	a preprocessed object of class 'TNI' TNI-class already evaluated by the tni.dpi.filter method.
geneSetList	a list with gene sets.
sampleSet	an optional vector of samples.
regulatoryElements	a vector of valid regulatory elements (e.g. transcription factors).
minSetSize	a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.

`sizeFilterMethod`
: a single character value specifying the use of the 'minSetSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posPLUSneg". For "posANDneg", the number of both positive and negative targets should be > 'minSetSize'; for "posORneg", the number of either positive or negative targets should be > 'minRegulonSize'; and for "posPLUSneg", the number of all targets should be > 'minSetSize'.

`exponent`
: a single integer or numeric value used in weighting phenotypes in GSEA (this parameter only affects the 'dES' option).

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

Details

Using the samples available in the provided TNI object, the 'tni.annotate.regulons' calculates the enrichment of each regulon for each gene set. First, the samples are split into two groups, one with high average gene-set expression (GS_high) and the other with low average gene-set expression (GS_low). Then a gene-wise differential expression (DEG) signature is generated by comparing the GS_high vs. GS_low groups. The DEG signature is regarded as the gene-set phenotype in the cohort. A GSEA-2T approach is used to calculate the activity score (dES) of each regulon in the phenotype (for additional details on GSEA-2T, please see section 2.2 of the RTN's vignette).

Value

A numeric matrix with dES scores between gene sets vs. regulons.

Author(s)

Mauro Castro

See Also

TNI-class

Examples

```
data(tniData)

## Not run:

#compute regulons
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#load a gene set collection
#here, we build three random gene sets for demonstration
```

```
geneset1 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset2 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset3 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneSetList <- list(geneset1=geneset1,
                    geneset2=geneset2,
                    geneset3=geneset3)

#compute regulon activity
dES <- tni.annotate.regulons(rtni, geneSetList)


## End(Not run)
```

`tni.annotate.samples` *Annotate samples with external gene set collections.*

Description

This function calculates an enrichment score between gene sets and samples.

Usage

```
tni.annotate.samples(object, geneSetList, minSetSize = 15,
    exponent = 1, samples=NULL, verbose = TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class`.
`geneSetList`	a list with gene sets.
`minSetSize`	a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`exponent`	a single integer or numeric value used in weighting phenotypes in GSEA (this parameter only affects the GSEA statistics).
`samples`	an optional string vector listing the sample names for which will be computed the GSEA statistics.
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Details

Using the samples available in the provided TNI object, the 'tni.annotate.samples' calculates the enrichment of each sample for each gene set. First, a gene-wise differential expression (DEG) signature is generated by comparing the expression of a given sample with the avarage expression of all samples. The DEG signature is regarded as a the sample phenotype, representing the relative expression of the sample's genes in the cohort. Then a single-sample Gene Set Enrichment Analysis (ssGSEA) is used to calculate the enrichment score (ES) of the sample for a given gene set.

Value

A numeric matrix with association statistics between gene sets vs. samples.

Author(s)

Mauro Castro

See Also

TNI-class

Examples

```
data(tniData)

## Not run:

#generate a TNI object
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#load a gene set collection
#here, we build three random gene sets for demonstration
geneset1 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset2 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset3 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneSetList <- list(geneset1=geneset1,
                    geneset2=geneset2,
                    geneset3=geneset3)

#compute single-sample GSEA
#note: regulons are not required for this function,
#as it will assess the samples in the TNI object
ES <- tni.annotate.samples(rtni, geneSetList)


## End(Not run)
```

---

tni.area3 *Compute regulon activity by calling aREA (analytic Rank-based Enrichment Analysis) algorithm*

Description

Uses aREA 3-tail algorithm to compute regulon activity for TNI-class objects.

Usage

```
tni.area3(object, minRegulonSize=15, sizeFilterMethod="posORneg", scale=FALSE, tnet="dpi",
regulatoryElements=NULL, samples=NULL, features=NULL, refsamp=NULL, log=FALSE, verbose=TRUE)
```

Arguments

object	a preprocessed object of class 'TNI' TNI-class.
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Regulons smaller than this number are removed from the analysis.
sizeFilterMethod	a single character value specifying the use of the 'minRegulonSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posPLUSneg". For "posANDneg", the number of both positive and negative targets should be > 'minRegulonSize'; for "posORneg", the number of either positive or negative targets should be > 'minRegulonSize'; and for "posPLUSneg", the number of all targets should be > 'minRegulonSize'.
scale	A logical value specifying if expression values should be centered and scaled across samples (when verbose=TRUE) or not (when verbose=FALSE).
tnet	can take values of 'refnet', 'dpi' or 'cdt'. It refers to the version of the regulatory network that will be used for GSEA analysis.
regulatoryElements	an optional vector with transcription factor identifiers.
samples	an optional string vector containing the sample names for which will be computed regulon activity.
features	a string vector containing features for feature selection.
refsamp	an optional string vector containing the names of the reference samples for differential expression calculations. If not provided, then the average of all samples will be used as reference.
log	a logical value. If TRUE, differential expression calculations will be computed in log space.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a list with enrichment scores for all samples in the TNI.

References

Alvarez et al. Functional characterization of somatic mutations in cancer using network-based inference of protein activity. Nature Genetics, 48(8):838-847, 2016.

See Also

TNI-class aREA

Examples

```
data(tniData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#run aREA algorithm
rtni <- tni.area3(rtni)

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



## End(Not run)
```

`tni.bootstrap` *Inference of consensus transcriptional networks.*

Description

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

Usage

```
tni.bootstrap(object, nBootstraps=100, consensus=95, parChunks=NULL, verbose=TRUE)
```

Arguments

`object`	a processed object of class 'TNI' `TNI-class` evaluated by the method `tni.permutation`.
`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

Examples

```
data(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

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

## parallel version with SNOW package!
#library(snow)
#options(cluster=snow::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, tfs=NULL, sampling=35, pValueCutoff=0.01,
pAdjustMethod="bonferroni", minRegulonSize=15, minIntersectSize=5,
miThreshold="md", prob=0.99, medianEffect=FALSE,
iConstraint=TRUE, 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 identifiers for those considered as candidate modulators.
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.
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).
iConstraint	a single logical value specifying whether to apply independence constraint between TFs and modulators (when verbose=TRUE) or not (when verbose=FALSE).

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(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

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

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

# get some 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.table")

# 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.constructor` *A constructor for objects of class TNI.*

Description

This function is the main entry point of the TNI pipeline.

Usage

```
tni.constructor(expData, regulatoryElements, rowAnnotation=NULL,
colAnnotation=NULL, cvfilter=FALSE, verbose=TRUE)
```

Arguments

`expData`	a gene expression matrix or 'SummarizedExperiment' object.
`regulatoryElements`	a vector of regulatory elements (e.g. transcription factors).
`rowAnnotation`	an optional data frame with gene annotation. Column 1 must provide all ids listed in the gene expression matrix. Ideally, col1 = <ID>, 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`).
`colAnnotation`	an optional data frame with sample annotation.

cvfilter	a single logical value specifying to remove duplicated genes in the gene expression matrix using the probe-to-gene annotation. In this case, 'rowAnnotation' must be provided, with col1 = <ID> and col2 = <GENEID>. Genes duplicated in col2 will be collapsed; the decision is made based on the maximum dinamic range (i.e. keeping the gene 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).

Value

A pre-processed TNI-class object.

Author(s)

Mauro Castro

See Also

TNI-class

Examples

```
data(tniData)

#--- run constructor
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
```

TNI.data *A pre-processed TNI for demonstration purposes only.*

Description

A minimum TNI object that can be used to demonstrate RTN functionalities.

Usage

```
data(stni)
```

Format

stni, A TNI-class with a subset of samples and genes from the Fletcher2013b package.

Details

The TNI consists of a TNI-class with a subsetted gene expression matrix and reduced list of transcription factors. It should be regarded as a toy example for demonstration purposes only, despite being extracted, pre-processed and size-reduced from Fletcher et al. (2013) and Curtis et al. (2012).

Value

a TNI-class.

References

Fletcher M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Nature Communications, 4:2464, 2013.

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

Examples

```
data(stni)
```

`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, sizeThreshold = TRUE, minRegulonSize = 15, verbose = TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class` evaluated by the methods `tni.permutation` and `tni.bootstrap`.
`eps`	a single numeric value (>= 0) specifying the threshold under which ARACNe algorithm should apply the dpi filter. If not available (i.e. 'eps = NA'), then the threshold is estimated from the empirical null distribution computed in the permutation and bootstrap steps. For additional details see `aracne`.
`sizeThreshold`	a logical value specifying if the 'minRegulonSize' argument should be used (when 'sizeThreshold = TRUE') or not (when 'sizeThreshold = FALSE'). It will have no effect when 'eps = NA'.
`minRegulonSize`	a single integer or numeric value. This argument prevents the DPI algorithm from removing additional targets from large unbalanced regulons, when the subset of either positive or negative targets is below the 'minRegulonSize'.
`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(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

# 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 available results from individual slots in a TNI object.

Usage

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

Arguments

object	an object of class TNI-class.
what	a single character value specifying which information should be retrieved from the slots. Options: 'summary', 'status', 'para', 'gexp','regulatoryElements', 'targetElements', 'modulators', 'tnet', 'refnet', 'regulons', 'refregulons', 'regulons.and.mode', 'refregulons.and.mode', 'rowAnnotation', 'colAnnotation', 'cdt.list', 'cdt.table', 'regulonSize','regulonActivity'.

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).

Details

Options for the 'what' argument:

summary A list summarizing parameters and results available in the TNI object (see `tni.regulon.summary` for a summary of the network and regulons).

status A vector indicating the status of each available method in the pipeline.

para A list with the parameters used by each available method in the pipeline.

gexp A gene expression matrix.

regulatoryElements A vector of regulatory elements (e.g. transcription factors).

targetElements A vector of target elements (e.g. TF targets).

modulators A vector of modulators (e.g. TF modulators).

tnet A data matrix with MI values, evaluated by the DPI filter. MI values are computed between regulators and targets, with regulators on cols and targets on rows. Note that signals (+/-) are assigned to the inferred associations in order to represent the 'mode of action', which is derived from Pearson's correlation between regulators and targets.

refnet A data matrix with MI values (not evaluated by the DPI filter). MI values are computed between regulators and targets, with regulators on cols and targets on rows. Note that signals (+/-) are assigned to the inferred associations in order to represent the 'mode of action', which is derived from Pearson's correlation between regulators and targets.

regulons A list with regulons extracted from the 'tnet' data matrix.

refregulons A list with regulons extracted from the 'refnet' data matrix.

regulons.and.mode A list with regulons extracted from the 'tnet' data matrix, including the assiged 'mode of action'.

refregulons.and.mode A list with regulons extracted from the 'refnet' data matrix, including the assiged 'mode of action'.

rowAnnotation A data frame with probe-to-gene annotation.

colAnnotation A data frame with sample annotation.

cdt.table A data frame with results from the `tni.conditional` analysis pipeline.

cdt.list A list with results from the `tni.conditional` analysis pipeline.

regulonSize A data frame with the number of targets annotated in each regulon.

regulonActivity A list with results from the `tni.gsea2` analysis pipeline.

Value

Get the slot content from a TNI-class object.

Author(s)

Mauro Castro

Examples

```
data(tniData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

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

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

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

## End(Not run)
```

---

tni.graph *Compute a graph from TNI objects (Deprecated).*

Description

Extract results from a TNI object and compute a graph (Deprecated).

Usage

```
tni.graph(x, ...)
```

Arguments

x	Deprecated arg.
...	Additional deprecated args.

Value

—

Examples

```
# deprecated function
```

`tni.gsea2` *Compute regulon activity by calling GSEA2 (two-tailed Gene Set Enrichment Analysis) algorithm*

Description

Uses GSEA2 algorithm to compute regulon activity for `TNI-class` objects.

Usage

```
tni.gsea2(object, minRegulonSize=15, sizeFilterMethod="posORneg", scale=FALSE,
exponent=1, tnet="dpi", regulatoryElements=NULL, features=NULL, samples=NULL,
refsamp=samples, log=TRUE, alternative=c("two.sided", "less", "greater"),
targetContribution=FALSE, additionalData=FALSE, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class`.
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon. Regulons smaller than this number are removed from the analysis.
`sizeFilterMethod`	a single character value specifying the use of the 'minRegulonSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posORnegTrim", "posSUMneg". For "posANDneg", both the number of positive and the number of negative targets must be >= 'minRegulonSize'; for "posORneg", either the number of positive or the number of negative targets must be >= 'minRegulonSize' (both subsets are retained); for "posORnegTrim", either the number of positive or the number of negative targets must be >= 'minRegulonSize', and subsets with size < 'minRegulonSize' are discarded; and for "posSUMneg", the combined number of positive and negative targets must be >= 'minRegulonSize'.s
`scale`	A logical value specifying if expression values should be centered and scaled across samples (when scale=TRUE) or not (when scale=FALSE).
`exponent`	a single integer or numeric value used in weighting phenotypes in GSEA.
`tnet`	can take values of 'ref', 'dpi' or 'cdt'. It refers to the version of the regulatory network that will be used for GSEA analysis.
`regulatoryElements`	an optional vector with transcription factor identifiers.

features	a string vector listing features for feature selection.
samples	an optional string vector listing the sample names for which will be computed the GSEA2.
refsamp	an optional string vector listing the names of the reference samples for differential expression calculations. If not provided, then the average of all samples will be used as reference.
log	a logical value. If TRUE, it will check whether the expression values are provided as logged data; if not, it will performe a log2 transformation on expression values before the differential expression calculations.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
targetContribution	This argument is used for internal calls. A single logical value specifying to return the contribution of each target in enrichment scores (when verbose=TRUE) or not (when verbose=FALSE).
additionalData	This argument is used for internal calls. A single logical value specifying to return the additional data objects (when verbose=TRUE) or not (when verbose=FALSE).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

a list with enrichment scores for all samples in the TNI. The list contains the following elements:

differential: A numeric `"matrix"` with differential enrichment scores (dES).

positive: A numeric `"matrix"` with enrichment scores from positive targets.

negative: A numeric `"matrix"` with enrichment scores from negative targets.

status: A numeric `"matrix"` with discretized scores derived from the dES values.

regulatoryElements: A character vector listing the regulatory elements assessed by the GSEA-2T algorithm.

sections: A single numeric value used in internal plots.

Author(s)

Mauro Castro

See Also

`TNI-class` `tna.gsea2` `tna.plot.gsea2`

Examples

```
data(tniData)

## Not run:
```

```
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#run GSEA2 analysis pipeline
rtni <- tni.gsea2(rtni)

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

#parallel version with SNOW package!
library(snow)
options(cluster=snow::makeCluster(3, "SOCK"))
rtni <- tni.gsea2(rtni)
stopCluster(getOption("cluster"))


## End(Not run)
```

`tni.overlap.genesets` *Associate regulons with external gene set collections.*

Description

This function tests the overlap between gene sets and regulons.

Usage

```
tni.overlap.genesets(object, geneSetList, regulatoryElements = NULL,
    minSetSize = 15, sizeFilterMethod="posORneg",
    method = c("HT","JC"), pValueCutoff = 0.05,
    pAdjustMethod = "BH", verbose = TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class` already evaluated by the `tni.dpi.filter` method.
`geneSetList`	a list with gene sets.
`regulatoryElements`	a vector of valid regulatory elements (e.g. transcription factors).
`minSetSize`	a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.

`sizeFilterMethod`
: a single character value specifying the use of the 'minSetSize' argument, which is applyed to regulon's positive and negative targets. Options: "posANDneg", "posORneg", "posPLUSneg". For "posANDneg", the number of both positive and negative targets should be > 'minSetSize'; for "posORneg", the number of either positive or negative targets should be > 'minRegulonSize'; and for "posPLUSneg", the number of all targets should be > 'minSetSize'.

`method`
: a string specifying the method used to assess the association between gene sets and regulons (see 'Details').

`pValueCutoff`
: a single numeric value specifying the cutoff for p-values considered significant (this parameter only affects the 'HT' option).

`pAdjustMethod`
: a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details) (this parameter only affects the 'HT' option).

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

Details

The 'HT' option assesses the overlap between gene sets and regulons using a hypergeometric test, and returns a data frame with the overlap statistics. The 'JC' option assesses the overlap between gene sets and regulons using the Jaccard Coefficient (JC), and retuns a matrix with JC values.

Value

Either a data frame or a numeric matrix with association statistics between gene sets vs. regulons.

Author(s)

Mauro Castro

See Also

TNI-class

Examples

```
data(tniData)

## Not run:

#compute regulons
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#load a gene set collection
#here, we build three random gene sets for demonstration
```

```
geneset1 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset2 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneset3 <- sample(tniData$rowAnnotation$SYMBOL,50)
geneSetList <- list(geneset1=geneset1,
                    geneset2=geneset2,
                    geneset3=geneset3)

#run the overlap analysis
ovstats <- tni.overlap.genesets(rtni, geneSetList, pValueCutoff = 1)


## 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="spearman", nPermutations=1000, pooledNullDistribution=TRUE,
        boxcox=TRUE, parChunks=NULL, 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 specifying the mutual information estimator. One of "pearson", "kendall", or "spearman" (default).
`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).

boxcox — a single logical value specifying to use Box-Cox procedure to find a transformation of inferred associations that approaches normality (when boxcox=TRUE) or not (when boxcox=FALSE). Dam et al. (2018) have acknowledged that different RNA-seq normalization methods introduce different biases in co-expression analysis, usually towards positive correlation, possibly affected by read-depth differences between samples and the large abundance of 0 values present in RNA-seq-derived expression matrices. In order to correct this positive correlation bias we suggest using this box-cox correction strategy. See powerTransform and bcPower.

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

References

Dam et al. Gene co-expression analysis for functional classification and gene-disease predictions. Brief Bioinform. 2018 Jul 20;19(4):575-592. doi: 10.1093/bib/bbw139.

See Also

TNI-class

Examples

```
data(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

# linear version (set nPermutations >= 1000)
rtni <- tni.permutation(rtni, nPermutations = 100)

## parallel version with SNOW package!
#library(snow)
#options(cluster=snow::makeCluster(3, "SOCK"))
```

```
#rtni<-tni.permutation(rtni)
#stopCluster(getOption("cluster"))

## End(Not run)
```

`tni.plot.checks` *Plot regulon target counts.*

Description

This funtion can help to check whether the numbers of positive and negative targets are reasonably well balanced in the regulons.

Usage

```
tni.plot.checks(object, minRegulonSize = 15, option = c("barplot","edf","points"))
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class` already evaluated by the `tni.dpi.filter` method.
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon (only affects the 'barplot' option).
`option`	plot option.

Value

A plot showing the distribution of regulons' positive and negative targets.

Note

We have observed that transcription factor (TF) regulons reconstructed from RTN exhibit different proportions of positive and negative targets. While the proportion can vary between different regulons, we have observed a consistent higher proportion of positive targets, especially when using RNA-seq data. RTN uses mutual information (MI) to assess TF-target associations, assigning the direction of the inferred associations by Spearman's correlations. Dam et al. (2018) have acknowledged that different RNA-seq normalization methods introduce different biases in co-expression analysis, usually towards positive correlation, possibly affected by read-depth differences between samples and the large abundance of 0 values present in RNA-seq-derived expression matrices. This funtion can help to check whether the numbers of positive and negative target genes are reasonably well balanced in the regulons.

Author(s)

Mauro Castro, Gordon Robertson

References

Dam et al. Gene co-expression analysis for functional classification and gene-disease predictions. Brief Bioinform. 2018 Jul 20;19(4):575-592. doi: 10.1093/bib/bbw139.

Examples

```
data(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

# compute regulons
rtni <- tni.permutation(rtni, nPermutations = 1000)
rtni <- tni.permutation(rtni)
rtni <- tni.dpi.filter(rtni)

# check target distribution
tni.plot.checks(rtni)


## End(Not run)
```

---

tni.plot.sre *Plot Subgroup Regulon Enrichment for TNI-class objects.*

Description

This method plots the results of the subgroup regulon enrichment analysis in a heatmap. The rows of the heatmap represent enriched regulons, while the columns show the subgroups. The plotted values correspond to average regulon activity for a regulon in a subgroup. Enriched values can be marked.

Usage

```
tni.plot.sre(object, nGroupsEnriched = NULL, nTopEnriched = NULL,
             colors = c("blue","white","red"), breaks = seq(-1.5, 1.5, 0.1),
             markEnriched = TRUE, ...)
```

Arguments

`object`	A TNI-class object.
`nGroupsEnriched`	a filter to keep 'nGroupsEnriched' regulons; a single integer specifying how many subgroups a regulon has to be enriched for it to appear in the rows of the heatmap (it must be use either 'nGroupsEnriched' or 'nTopEnriched').

nTopEnriched	a filter to keep 'nTopEnriched' regulons; a single integer specifying how many regulons will be shown for each group. The top regulons are chosen by significance (it must be use either 'nTopEnriched' or 'nGroupsEnriched').
colors	a vector of color for the 'pheatmap'.
breaks	a numerical vector of breaks for the 'pheatmap'.
markEnriched	a single logical value. If TRUE, asterisks are added to cells of heatmap that were found to be significant.
...	parameters passed to 'pheatmap' for customization.

Value

A heatmap of the subgroup regulon enrichment results.

See Also

tni.sre

Examples

```
# load tniData
data(tniData)

## Not run:

# preprocessing
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)

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

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

#run GSEA2 analysis pipeline
rtni <- tni.gsea2(rtni)

# set sample groups
colAnnotation <- tni.get(rtni, "colAnnotation")
sampleGroups <- list(G1=colAnnotation$ID[1:60],
                G2=colAnnotation$ID[61:90],
                G3=colAnnotation$ID[91:120])

# run subgroup regulon enrichment analysis
rtni <- tni.sre(rtni, sampleGroups)

# plot results
tni.plot.sre(rtni)

## End(Not run)
```

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

Description

This is a generic function, provides all preprocessing methods for the 'tni.constructor' function.

Usage

```
tni.preprocess(object, rowAnnotation=NULL, colAnnotation=NULL, cvfilter=FALSE,
    verbose=TRUE)
```

Arguments

object	this argument is an object of class TNI-class.
rowAnnotation	an optional data frame with gene annotation.
colAnnotation	an optional data frame with sample annotation.
cvfilter	a single logical value specifying to remove duplicated genes in the gene expression matrix using the gene annotation.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

A pre-processed TNI-class object.

Author(s)

Mauro Castro

See Also

tni.constructor

Examples

```
## see 'tni.constructor'!
```

tni.prune *Prune regulons to remove redundant targets for regulon activity analysis*

Description

Uses network pruning methods to compute a 'core' regulon that retains good correlation with original regulon activity.

Usage

```
tni.prune(object, regulatoryElements = NULL, minRegCor = 0.95,
tarPriorityMethod = "EC", minPrunedSize = 30, verbose = TRUE, ...)
```

Arguments

object	a preprocessed object of TNI-class.
regulatoryElements	an optional vector with regulatoryElements identifiers. If NULL, all regulatoryElements are used.
minRegCor	an numeric value between 0 and 1. The minimum correlation between the original activity values for a regulon and the activity after pruning.
tarPriorityMethod	method for prioritizing targets for the target backwards elimination. One of "EC" (expression correlation), "MI" (mutual information) or "TC" (target contribution).
minPrunedSize	a single integer or numeric value specifying the minimum number of elements in a regulon after pruning.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).
...	arguments passed to tni.gsea2

Value

a TNI-class object, with the pruned regulons.

Author(s)

Clarice Groeneveld

See Also

TNI-class tni.gsea2

Examples

```
data(tniData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

# prune the PTTG1 regulon
rtni_pruned <- tni.prune(rtni, "PTTG1", tarPriorityMethod = "TC")

#parallel version with SNOW package!
#library(snow)
#options(cluster=makeCluster(3, "SOCK"))
#rtni_pruned <- tni.prune(rtni, c("PTTG1", "E2F2"))
#stopCluster(getOption("cluster"))


## End(Not run)
```

---

tni.regulon.summary *Summary of regulon characteristics.*

Description

This function takes a TNI object and optionally a list of regulatory elements and returns a summary of the network (if no regulatory elements are given) or of the chosen regulon or regulons.

Usage

```
tni.regulon.summary(object, regulatoryElements = NULL, verbose = TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class` already evaluated by the `tni.dpi.filter` method.
`regulatoryElements`	a vector of valid regulatory elements (e.g. transcription factors).
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

It returns a print-out of the network summary (if verbose is TRUE) and invisibly returns a data.frame of network characteristics such as regulon size and regulon balance.

Author(s)

Clarice Groeneveld

See Also

TNI-class

Examples

```
data(tniData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

# Summary of the network
tni.regulon.summary(rtni)

# Summary of a regulon
tni.regulon.summary(rtni, regulatoryElements = "PTTG1")


## End(Not run)
```

---

tni.replace.samples *Entry point to assess new samples with previously calculated regulons.*

Description

This function replaces samples of an existing TNI-class objects.

Usage

```
tni.replace.samples(object, expData, rowAnnotation=NULL, colAnnotation=NULL,
removeRegNotAnnotated=TRUE, verbose=TRUE)
```

Arguments

object	an object of class TNI-class.
expData	a gene expression matrix or 'SummarizedExperiment' object.
rowAnnotation	an optional data frame with gene annotation. Column 1 must provide all ids listed in the gene expression matrix.
colAnnotation	an optional data frame with sample annotation.

removeRegNotAnnotated
: a single logical value specifying to remove 'regulatoryElements' not annotated in 'expData' (when removeRegNotAnnotated=TRUE) or not (when removeRegNotAnnotated=FALSE).

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

Value

A TNI-class object.

Author(s)

Mauro Castro

Examples

```
## please the package's vignette
```

---

tni.sre *Subgroup Regulon Enrichment for TNI-class objects.*

Description

This method evaluates which regulons are enriched in sample groups, given a grouping variable. It performs Fisher's Exact Test whether a regulon is positively or negatively enriched in a subgroup using regulon activity.

Usage

```
tni.sre(object, sampleGroups, regulatoryElements = NULL,
        pValueCutoff = 0.05, pAdjustMethod = "BH")
```

Arguments

object
: A TNI object.

sampleGroups
: either a list featuring sample groups or a string indicating a group varible available in the TNI object.

regulatoryElements
: an optional string vector specifying regulons to use for the analysis.

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).

Value

A TNI-class object with the results of the subgroup regulon enrichment added to the results slot. To recover the results, use tni.get(object, "regulonEnrichment")

See Also

tni.plot.sre

Examples

```
# load tniData
data(tniData)

## Not run:

# compute regulons
rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)

#run GSEA2 analysis pipeline
rtni <- tni.gsea2(rtni)

# set sample groups
colAnnotation <- tni.get(rtni, "colAnnotation")
sampleGroups <- list(G1=colAnnotation$ID[1:60],
                G2=colAnnotation$ID[61:90],
                G3=colAnnotation$ID[91:120])

# run subgroup regulon enrichment analysis
rtni <- tni.sre(rtni, sampleGroups)

# get results
tni.get(rtni, "subgroupEnrichment")

# for a heatmap representation, see the tni.plot.sre() function.

## End(Not run)
```

---

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 vector of phenotypes named by gene identifiers (usually log2 differential expression values). Required for the tna.gsea1 and tna.gsea2) methods.
hits	a character vector of gene identifiers for those considered as hits. Required for the tna.mra method.
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).

Value

A pre-processed TNA-class object.

Author(s)

Mauro Castro

See Also

TNI-class TNA-class

Examples

```
data(tniData)
data(tnaData)

## Not run:

rtni <- tni.constructor(expData=tniData$expData,
        regulatoryElements=c("PTTG1","E2F2","FOXM1","E2F3","RUNX2"),
        rowAnnotation=tniData$rowAnnotation)
rtni <- tni.permutation(rtni)
rtni <- tni.bootstrap(rtni)
rtni <- tni.dpi.filter(rtni)
rtna <- tni2tna.preprocess(rtni, phenotype=tnaData$phenotype,
        hits=tnaData$hits, phenoIDs=tnaData$phenoIDs)

## End(Not run)
```

upgradeTNA *Upgrade objects of class TNI.*

Description

This function provides compatibility checks for a TNI class object.

Usage

```
upgradeTNA(object)
```

Arguments

object	this argument is an object of class TNI-class.

Value

An updated TNA-class object.

Author(s)

Mauro Castro

Examples

```
### Objects of class TNA generated by RTN (version <= 1.15.2) can be upgraded
### to the latest version by calling upgradeTNA().
```

upgradeTNI *Upgrade objects of class TNI.*

Description

This function provides compatibility checks for a TNI class object.

Usage

```
upgradeTNI(object)
```

Arguments

object	this argument is an object of class TNI-class.

Value

An updated TNI-class object.

Author(s)

Mauro Castro

Examples

```
### Objects of class TNI generated by RTN (version <= 1.15.2) can be upgraded
### to the latest version by calling upgradeTNI().
```

Index

∗ **GSEA2**
tna.plot.gsea2, 31
∗ **GSEA**
tna.gsea1, 24
tna.gsea2, 26
tna.plot.gsea1, 29
tni.gsea2, 50
∗ **Prune**
tni.prune, 60
∗ **RMA**
tna.mra, 28
∗ **VSE**
avs.plot1, 12
avs.plot2, 13
∗ **aREA**
tni.area3, 38
∗ **annotate**
tni.annotate.regulons, 35
tni.annotate.samples, 37
tni.overlap.genesets, 52
∗ **classes**
AVS-class, 4
TNA-class, 20
TNI-class, 33
∗ **dataset**
RTN.data, 19
TNI.data, 45
∗ **methods**
avs.evse, 5
avs.get, 8
avs.pevse, 9
avs.rvse, 14
avs.vse, 16
tna.get, 22
tni.alpha.adjust, 34
tni.bootstrap, 40
tni.conditional, 41
tni.constructor, 44
tni.dpi.filter, 46
tni.get, 47
tni.graph, 49
tni.permutation, 54
tni.plot.checks, 56
tni.preprocess, 59
tni.replace.samples, 62
tni2tna.preprocess, 64
∗ **package**
RTN-package, 3
∗ **subgroups**
tni.sre, 63
∗ **summary**
tni.regulon.summary, 61
∗ **upgrade**
upgradeTNA, 66
upgradeTNI, 66

aracne, *46*
aREA, *38, 39*
AVS-class, *3*, 4
avs.evse, *3*, *5*, 5, *44*
avs.evse,AVS-method *(*AVS-class*)*, 4
avs.get, *4–6*, 8, *10*, *15*, *17*
avs.get,AVS-method *(*AVS-class*)*, 4
avs.pevse, *4, 5*, 9
avs.pevse,AVS-method *(*AVS-class*)*, 4
avs.plot1, *4*, 12
avs.plot2, *4*, 13
avs.rvse, *5*, 14
avs.rvse,AVS-method *(*AVS-class*)*, 4
avs.vse, *3*, *5*, *9*, *14*, 16
avs.vse,AVS-method *(*AVS-class*)*, 4

bcPower, *6*, *10*, *15*, *17*, *55*

hist, *12*

pksData *(*RTN.data*)*, 19
powerTransform, *6*, *10*, *15*, *17*, *55*
pwr.r.test, *35*

RTN (RTN-package), 3
RTN-package, 3
RTN.data, 19

stni (TNI.data), 45

tfsData (RTN.data), 19
TNA-class, *3*, 20
tna.get, *3*, *21*, 22, *25*, *27*, *28*
tna.get,TNA-method (TNA-class), 20
tna.gsea1, *3*, *21*, *23*, 24, *30*, *65*
tna.gsea1,TNA-method (TNA-class), 20
tna.gsea2, *3*, *21*, *23*, 26, *32*, *51*, *65*
tna.gsea2,TNA-method (TNA-class), 20
tna.mra, *3*, *21*, *23*, 28, *65*
tna.mra,TNA-method (TNA-class), 20
tna.plot.gsea1, *3*, *25*, 29
tna.plot.gsea2, *3*, *27*, 31, *51*
tnaData (RTN.data), 19
TNI, *57*, *63*
TNI-class, *3*, 33
tni.alpha.adjust, *3*, 34
tni.annotate.regulons, *34*, 35
tni.annotate.regulons,TNI-method (TNI-class), 33
tni.annotate.samples, *34*, 37
tni.annotate.samples,TNI-method (TNI-class), 33
tni.area3, *33*, 38
tni.area3,TNI-method (TNI-class), 33
tni.bootstrap, *3*, *33*, 40, *42*, *46*, *65*
tni.bootstrap,TNI-method (TNI-class), 33
tni.conditional, *3*, *21*, *33*, 41, *48*
tni.conditional,TNI-method (TNI-class), 33
tni.constructor, 44, *59*
TNI.data, 45
tni.dpi.filter, *3*, *33*, *35*, *42*, 46, *52*, *56*, *61*, *65*
tni.dpi.filter,TNI-method (TNI-class), 33
tni.get, *3*, *33*, *41*, *43*, *46*, 47, *55*
tni.get,TNI-method (TNI-class), 33
tni.graph, *3*, *33*, 49
tni.graph,TNI-method (TNI-class), 33
tni.gsea2, *3*, *33*, *48*, 50, *60*
tni.gsea2,TNI-method (TNI-class), 33
tni.overlap.genesets, *34*, 52
tni.overlap.genesets,TNI-method (TNI-class), 33
tni.permutation, *3*, *33*, *40*, *42*, *46*, 54, *65*
tni.permutation,TNI-method (TNI-class), 33
tni.plot.checks, *3*, 56
tni.plot.sre, *3*, 57, *64*
tni.preprocess, *3*, *33*, 59
tni.preprocess,TNI-method (TNI-class), 33
tni.prune, *3*, *33*, 60
tni.prune,TNI-method (TNI-class), 33
tni.regulon.summary, *3*, *33*, *48*, 61
tni.regulon.summary,TNI-method (TNI-class), 33
tni.replace.samples, *3*, *33*, 62
tni.replace.samples,TNI-method (TNI-class), 33
tni.sre, *3*, *34*, *58*, 63
tni.sre,TNI-method (TNI-class), 33
tni2tna.preprocess, *3*, *21–23*, *25*, *27*, *34*, 64
tni2tna.preprocess,TNI-method (TNI-class), 33
tniData (RTN.data), 19

upgradeTNA, 66
upgradeTNI, 66