Package 'RTN'

October 16, 2019

Type Package

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

Version 2.8.5

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 (>= 3.5.0), methods

Imports RedeR, minet, viper, mixtools, snow, limma, data.table, IRanges, igraph, S4Vectors, SummarizedExperiment

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 TNA-methods.R TNI-methods.R AVS-methods.R TNI-pruning.R

LazyLoad yes

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

git_branch RELEASE_3_9

git_last_commit b8c66dc

git_last_commit_date 2019-09-15

Date/Publication 2019-10-15

R topics documented:

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

Description

This package provides classes and methods for transcriptional network inference and analysis. Modulators of transcription factor activity are assessed by conditional mutual information, and master regulators are mapped to phenotypes using different strategies, e.g., gene set enrichment, shadow and synergy analyses. Additionally, master regulators can be linked to genetic markers using eQTL/VSE analysis, taking advantage of the haplotype block structure mapped to the human genome in order to explore risk-associated SNPs identified in GWAS studies.

Details

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

Index

avs.plot1: plot results from AVS methods, single plots.
avs.plot2: plot results from AVS methods, multiple plots.

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

Author(s)

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

References

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.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, maxgap=250, pValueCutoff=0.05, pAdjustMethod="bonfer
boxcox=TRUE, lab="annotation", glist=NULL, minSize=100,  fineMapping=TRUE,
verbose=TRUE)
```

Arguments

object	an object. When this function is implemented as the S4 method of class AVS-class, this argument is an object of class 'AVS'.
annotation	a data frame with genomic annotations listing chromosome coordinates to which a particular property or function has been attributed. It should include the following columns: <CHROM>, <START>, <END> and <ID>. The <ID> column can be any genomic identifier, while values in <CHROM> should be listed in ['chr1', 'chr2', 'chr3' ..., 'chrX']. Both <START> and <END> columns correspond to chromosome positions mapped to the human genome assembly used to build the AVS object.

gxdata	object of class "matrix", a gene expression matrix.
snpdata	either an object of class "matrix" or "ff", a single nucleotide polymorphism (SNP) matrix.
maxgap	a single integer value specifying the max distant (kb) between the AVS and the annotation used to compute the eQTL analysis.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
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.
lab	a single character value specifying a name for the annotation dataset (this option is overrided if 'glist' is used).
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This option can be used to run a batch mode for gene sets and regulons.
minSize	if 'glist' is provided, this argument is a single integer or numeric value specifying the minimum number of elements for each gene set in the 'glist'. Gene sets with fewer than this number are removed from the analysis. if 'fineMapping=FALSE', an alternative min size value can be provided as a vector of the form c(minSize1, minSize2) used to space the null distributions (see 'fineMapping').
fineMapping	if 'glist' is provided, this argument is a single logical value specifying to compute individual null distributions, sized for each gene set (when fineMapping=TRUE). This option has a significant impact on the running time required to perform the computational analysis, especially for large gene set lists. When fineMapping=FALSE, a low resolution analysis is performed by pre-computing a fewer number of null distributions of different sizes (spaced by 'minSize'), and then used as a proxy of the nulls.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
# This example requires the RTNdata package! (currently available under request)
library(RTNdata.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 just 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")
data(toy_gxdata, package="RTNdata.LDHapMapRel27")

#--- 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)

### NOTE REGARDING THIS EXAMPLE ####
#- This example is for demonstration purposes only. Despite the toy datasets,
#- both the AVS and regulons are derived from true observations. So, 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', 'pevse', 'annotation.vse', 'annotation.evse', '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, maxgap=250, pValueCutoff=0.05, pAdjustMethod="bonferroni",
boxcox=TRUE, lab="annotation", glist=NULL, minSize=100, 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>.
maxgap	a single integer value specifying the max distant (kb) used to assign the eQTLs in the 'eqtls' object.
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.
lab	a single character value specifying a name for the annotation dataset (this option is overrided if 'glist' is used).
glist	an optional list with character vectors mapped to the 'annotation' data via <ID> column. This option can be used to run a batch mode for gene sets and regulons.
minSize	if 'glist' is provided, this argument is a single integer or numeric value specifying the minimum number of elements for each gene set in the 'glist'. Gene sets with fewer than this number are removed from the analysis. if 'fineMapping=FALSE', an alternative min size value can be provided as a vector of the form c(minSize1, minSize2) used to space the null distributions (see 'fineMapping').
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

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.

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, rmargin=1,
bxseq=seq(-4,8,2), decreasing=TRUE, ylab="Annotation",
xlab="Clusters of risk-associated and linked SNPs", tfs=NULL)
```

Arguments

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

Author(s)

Mauro Castro

Examples

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

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

Description

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

Usage

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

Arguments

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

Author(s)

Mauro Castro

See Also

AVS-class

Examples

```
## Not run:
# This example requires the RTNdata package! (currently available under request)
library(RTNdata.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$bdsites, pValueCutoff=0.001, lab="ERa")
bcavs <- avs.vse(bcavs, annotation=FOXA1bdsites$bdsites, pValueCutoff=0.001, lab="FOXA1")

#--- step 4: generate the VSE plots
avs.plot2(bcavs,"vse",height=2.2)

##################################################
### 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 just 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)

### 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 the 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.

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.

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)
```

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

Description

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

Objects from the Class

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

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.overlap`, `tna.gsea1`, `tna.synergy`, `tna.shadow` and `tna.gsea2`.

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

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

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

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

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

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

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

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

Author(s)

Mauro Castro

See Also

TNI-class. tni2tna.preprocess.

Examples

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

---

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

Description

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

Usage

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

Arguments

object	an object of class TNA-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', 'overlap', 'synergy', 'shadow'.
order	a single logical value specifying whether or not the output data should be ordered by significance. Valid only for 'gsea1', 'gsea2', 'overlap', 'synergy' or 'shadow' options.
ntop	a single integer value specifying to select how many results of top significance from 'gsea', 'overlap', 'synergy' or 'shadow' options.
reportNames	a single logical value specifying to report regulons with 'names' (when reportNames=TRUE) or not (when reportNames=FALSE). This option is effective only if transcription factors were named with alternative identifiers in the preprocessing analysis. It takes effect on 'mra', 'gsea', 'overlap', 'synergy' and 'shadow' options.
idkey	an optional single character value specifying an ID name from the available 'TNA' annotation to be used as alias for data query outputs (obs. it has no effect on consolidated tables).

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.

overlap A data frame with results from the `tna.overlap` analysis pipeline.

synergy A data frame with results from the `tna.synergy` analysis pipeline.

shadow A data frame with results from the `tna.shadow` 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.graph` *Compute a graph from TNA objects.*

Description

Extract results from a TNA object and compute a graph.

Usage

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

Arguments

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

amapCutoff	a single numeric value (>=0 and <=1) specifying the cutoff for the association map filter. When amapFilter="phyper", amapCutoff corresponds to a pvalue cutoff; when amapFilter="quantile", amapCutoff corresponds to a quantile threshold; and when amapFilter="custom", amapCutoff is a JC threshold.
...	additional arguments passed to tna.graph function.

Value

a graph object.

Author(s)

Mauro Castro

Examples

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

# get the regulatory map
g1 <- tna.graph(rtna, tnet="dpi", gtype="rmap", tfs=c("PTTG1","E2F2","FOXM1"))

# get the association map
g2 <- tna.graph(rtna, tnet="ref", gtype="amap")

# option: plot the igraph objects using RedeR
#library(RedeR)
#rdp <- RedPort()
#calld(rdp)
#addGraph(rdp,g1)
#relax(rdp, ps=TRUE)


## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Mauro Castro, Xin Wang

See Also

TNA-class tna.plot.gsea1

Examples

```
data(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, stepFilter=FALSE)

#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,stepFilter=FALSE)
stopCluster(getOption("cluster"))

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.

nPermutations	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
exponent	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
tnet	a single character value specifying which transcriptional network should to used to compute the GSEA analysis. Options: "dpi" and "ref".
stepFilter	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from tna.mra (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
tfs	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

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

Author(s)

Mauro Castro

See Also

TNA-class tna.plot.gsea2

Examples

```
data(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, stepFilter=FALSE)

#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,stepFilter=FALSE)
```

```
stopCluster(getOption("cluster"))

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Mauro Castro

See Also

`TNA-class`

Examples

```
data(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.overlap` *Overlap analysis over a list of regulons.*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Mauro Castro

See Also

TNA-class

Examples

```
data(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 overlap analysis pipeline
rtna <- tna.overlap(rtna)

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

## End(Not run)
```

---

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

Description

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

Usage

```
tna.plot.gsea1(object, labPheno="", 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.

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, stepFilter=FALSE)

# plot available GSEA results
tna.plot.gsea1(rtna, labPheno="test")

## End(Not run)
```

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

Description

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

Usage

```
tna.plot.gsea2(object, labPheno="", 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.

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, stepFilter=FALSE)

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

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Mauro Castro

See Also

TNA-class tna.shadow

Examples

```
data(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 overlap analysis pipeline
rtna <- tna.overlap(rtna)

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

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

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`object`	a preprocessed object of class 'TNA' `TNA-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`minRegulonSize`	a single integer or numeric value specifying the minimum number of elements in a regulon that must map to elements of the gene universe. Gene sets with fewer than this number are removed from the analysis.
`minIntersectSize`	a single integer or numeric value specifying the minimum number of elements in the intersect between any two regulons in the synergy analysis (as percentage value).

nPermutations	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA.
exponent	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
tnet	a single character value specifying which transcriptional network should to used to compute the synergy and shadow analyses. Options: "dpi" and "ref".
orderAbsValue	a single logical value indicating whether the values should be converted to absolute values and then ordered (if TRUE), or ordered as they are (if FALSE).
stepFilter	a single logical value specifying to use a step-filter algorithm removing non-significant regulons derived from tna.gsea1 (when stepFilter=TRUE) or not (when stepFilter=FALSE). It may have a substantial impact on the overall processing time.
tfs	an optional vector with transcription factor identifiers (this option overrides the 'stepFilter' argument).
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

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

Author(s)

Mauro Castro

See Also

TNA-class tna.shadow

Examples

```
data(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 synergy analysis pipeline
rtna <- tna.synergy(rtna, stepFilter=FALSE)

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

# run parallel version with SNOW package!
```

```
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtna <- tna.synergy(rtna,stepFilter=FALSE)
stopCluster(getOption("cluster"))

## End(Not run)
```

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

Description

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

Slots

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

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

`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`

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

Author(s)

Mauro Castro

See Also

TNA-class

Examples

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

---

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, doSizeFilter=FALSE, scale=FALSE, tnet="dpi",
tfs=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.
doSizeFilter	a logical value. If TRUE, negative and positive targets are independently verified by the 'minRegulonSize' argument.
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.
tfs	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
EScores <- tni.area3(rtni)


## 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=10, 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=NULL, tfs=NULL, sampling=35, pValueCutoff=0.01,
pAdjustMethod="bonferroni", minRegulonSize=15, minIntersectSize=5,
miThreshold="md", prob=0.99, pwtransform=FALSE, 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 gene identifiers for those considered as candidate modulators. If NULL, the function will assess all TFs in the network, provided that the candidate passes all filtering steps.
tfs	a vector with TF identifiers. If NULL, the function will assess all TFs in the network.
sampling	a single integer value specifying the percentage of the available samples that should be included in the analysis. For example, for each TF-target interaction of a given hub, 'sampling = 35' means that the conditional mutual information will be computed from the top and bottom 35% of the samples ranked by the gene expression of a given candidate modulator.
pValueCutoff	a single numeric value specifying the cutoff for p-values considered significant.
pAdjustMethod	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Gene sets with fewer than this number are removed from the analysis.
minIntersectSize	a single integer or numeric value specifying the minimum number of observed modulated elements in a regulon (as percentage value).
miThreshold	a single character value specifying the underlying distribution used to estimate the mutual information threshold. Options: 'md' and 'md.tf'. In the 1st case, 'miThreshold' is estimated from a pooled null distribution representing random modulators, while in the 2nd case a specific mutual information threshold is estimated for each TF conditioned on the random modulators. In the two options the 'miThreshold' is estimated by permutation analysis (see 'prob'). Alternatively, users can either provide a custom mutual information threshold or a numeric vector with lower (a) and upper (b) bounds for the differential mutual information analysis (e.g. 'c(a,b)').
prob	a probability value in [0,1] used to estimate the 'miThreshold' based on the underlying quantile distribution.
pwtransform	a single logical value specifying whether a Box-Cox power transformation should be used to improve data symmetry (this procedure might be carefully used as a complementary test to check the validity of the inferred associations).
medianEffect	a single logical value specifying whether to assess the median effect of each modulator. This global statistics does not affect the inferential process over single TF-target interactions. This method is still experimental, it can be used as a complementary analysis to chek the overall modulation effect onto all targets listed in a given regulon (this step may require substantial computation time).
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 analysis (infers the reference/relevance network)
rtni <- tni.permutation(rtni)

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

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

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

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

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

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

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

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


## End(Not run)
```

`tni.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).

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

stniA 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, verbose=TRUE)
```

Arguments

`object`	a processed object of class 'TNI' `TNI-class` evaluated by the methods `tni.permutation` and `tni.bootstrap`.
`eps`	a single numeric value specifying the threshold under which ARACNe algorithm should apply the dpi filter (eps>=0). 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`.
`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 information from individual slots in a TNI object and any available results from a previous analysis.

Usage

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

Arguments

`object`	an object of class `TNI-class`.
`what`	a single character value specifying which information should be retrieved from the slots. Options: 'summary', 'status', 'para', 'gexp','regulatoryElements', 'tnet', 'refnet', 'regulons', 'refregulons', 'regulons.and.mode', 'refregulons.and.mode', 'rowAnnotation', 'colAnnotation', 'cdt', 'cdtrev').
`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 retrieve the following types of information:

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).

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

cdtrev A data frame with the same results as retrieved above (using 'cdt'), but arranged in a different format.

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.*

Description

Extract results from a TNI object and compute a graph.

Usage

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

Arguments

object	an object of class 'TNI' TNI-class.
tnet	a single character value specifying which network information should be used to compute the graph. Options: "ref" and "dpi".
gtype	a single character value specifying the graph type. Options: "rmap", "amap", "mmap" and "mmapDetailed". The "rmap" option returns regulatory maps represented by TFs and targets (regulons); "amap" computes association maps among regulons (estimates the overlap using the Jaccard Coefficient); "mmap" and "mmapDetailed" return modulated maps derived from the tni.conditional function.
minRegulonSize	a single integer or numeric value specifying the minimum number of elements in a regulon. Regulons with fewer than this number are removed from the graph.
tfs	a vector with transcription factor identifiers.
amapFilter	a single character value specifying which method should be used to filter association maps (only when gtype="amap"). Options: "phyper","quantile" and "custom".
amapCutoff	a single numeric value (>=0 and <=1) specifying the cutoff for the association map filter. When amapFilter="phyper", amapCutoff corresponds to a pvalue cutoff; when amapFilter="quantile", amapCutoff corresponds to a quantile threshold; and when amapFilter="custom", amapCutoff is a JC threshold.
ntop	when gtype="mmapDetailed", ntop is an optional single integer value (>=1) specifying the number of TF's targets that should be used to compute the modulated map. The n targets is derived from the top ranked TF-target interations, as defined in the mutual information analysis used to construct the regulon set.
...	additional arguments passed to tni.graph function.

Value

a graph object.

Author(s)

Mauro Castro

Examples

```
data(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, eps=0)

# get the regulatory map
g <- tni.graph(rtni, tnet="dpi", gtype="rmap", tfs=c("PTTG1","E2F2","FOXM1"))

# option: plot the igraph object using RedeR
library(RedeR)
rdp <- RedPort()
calld(rdp)
addGraph(rdp,g)
addLegend.shape(rdp, g)
addLegend.color(rdp, g, type="edge")
relax(rdp, p1=20, p3=50, p4=50, p5=10, p8=50, ps=TRUE)
#...it should take some time to relax!

# get the association map
resetd(rdp)
g <- tni.graph(rtni, tnet="ref", gtype="amap")
addGraph(rdp,g)
addLegend.size(rdp, g)
addLegend.size(rdp, g, type="edge")


## End(Not run)
```

`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, doSizeFilter=FALSE, scale=FALSE, exponent=1,
tnet="dpi", tfs=NULL, samples=NULL, features=NULL, refsamp=NULL, log=FALSE,
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.
doSizeFilter	a logical value. If TRUE, negative and positive targets are independently verified by the 'minRegulonSize' argument.
scale	A logical value specifying if expression values should be centered and scaled across samples (when verbose=TRUE) or not (when verbose=FALSE).
exponent	a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function at HTSanalyzeR).
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.
tfs	an optional vector with transcription factor identifiers.
samples	an optional string vector containing the sample names for which will be computed the GSEA2.
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.
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.

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
EScores <- tni.gsea2(rtni)

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


## 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,
        parChunks=50, verbose=TRUE)
```

Arguments

`object`	a preprocessed object of class 'TNI' `TNI-class`.
`pValueCutoff`	a single numeric value specifying the cutoff for p-values considered significant.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details).
`globalAdjustment`	a single logical value specifying to run global p.value adjustments (when globalAdjustment=TRUE) or not (when globalAdjustment=FALSE).
`estimator`	a character string indicating which estimator to be used for mutual information computation. One of "pearson", "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).

`parChunks` an optional single integer value specifying the number of permutation chunks to be used in the parallel analysis (effective only for "pooledNullDistribution = TRUE").

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

Value

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

Author(s)

Mauro Castro

See Also

`TNI-class`

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.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. 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 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).

Author(s)

Mauro Castro

See Also

`TNI-class`

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

Author(s)

Mauro Castro

Examples

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

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

Author(s)

Mauro Castro

See Also

`TNI-class` `TNA-class`

Examples

```
data(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.

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.

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