

Package ‘viper’

November 13, 2024

Version 1.40.0

Date 2013-04-09

Title Virtual Inference of Protein-activity by Enriched Regulon analysis

Author Mariano J Alvarez <reef103@gmail.com>

Maintainer Mariano J Alvarez <reef103@gmail.com>

Depends R (>= 2.14.0), Biobase, methods

Imports mixtools, stats, parallel, e1071, KernSmooth

Suggests bcellViper

Description Inference of protein activity from gene expression data, including the VIPER and msVIPER algorithms

License file LICENSE

biocViews SystemsBiology, NetworkEnrichment, GeneExpression, FunctionalPrediction, GeneRegulation

Encoding UTF-8

RoxygenNote 6.1.1

git_url <https://git.bioconductor.org/packages/viper>

git_branch RELEASE_3_20

git_last_commit e859341

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-12

Contents

| | |
|-------------------------------------|---|
| aecdf | 2 |
| approxk2d | 3 |
| aracne2regulon | 4 |
| aracne2regulon4cnv | 5 |
| aREA | 6 |
| as.dist.signatureDistance | 6 |
| bootstrapmsviper | 7 |
| bootstrapTtest | 7 |
| bootstrapViper | 8 |

| | |
|-------------------------|-----------|
| comNames | 9 |
| distMode | 10 |
| fcvarna | 11 |
| filterColMatrix | 11 |
| filterCV | 12 |
| filterRowMatrix | 12 |
| frcv | 13 |
| frvarna | 13 |
| groupPwea3 | 14 |
| integrateSignatures | 15 |
| ledge | 15 |
| loadExpset | 16 |
| msviper | 16 |
| msviper-class | 18 |
| msviperAnnot | 18 |
| msviperClass | 19 |
| msviperCombinatorial | 20 |
| msviperSynergy | 21 |
| plot.msviper | 22 |
| pruneRegulon | 23 |
| pwea3NULLf | 24 |
| pwea3NULLgroups | 24 |
| regulon-class | 25 |
| rowTtest | 25 |
| scale.signatureDistance | 26 |
| scaleGroups | 26 |
| shadow | 27 |
| shadowRegulon | 28 |
| signatureDistance | 29 |
| signatureDistance-class | 30 |
| sigT | 30 |
| summary.msviper | 30 |
| ttestNull | 31 |
| viper | 32 |
| viperRPT | 33 |
| viperSignature | 34 |
| viperSignature-class | 35 |
| viperSimilarity | 36 |
| Index | 37 |

aecdf

Approximate empirical commulative distribution function

Description

This function generates an empirical null model that computes a normalized statistics and p-value

Usage

```
aecdf(dnull, symmetric = FALSE, n = 100)
```

Arguments

| | |
|------------------------|---|
| <code>dnull</code> | Numerical vector representing the null model |
| <code>symmetric</code> | Logical, whether the distribution should be treated as symmetric around zero and only one tail should be approximated |
| <code>n</code> | Integer indicating the number of points to evaluate the empirical cumulative probability function |

Value

function with two parameters, `x` and `alternative`

approxk2d

approxk2d

Description

This function uses a gaussian kernel to estimate the joint density distribution at the specified points

Usage

```
approxk2d(x, gridsize = 128, pos = x)
```

Arguments

| | |
|-----------------------|--|
| <code>x</code> | Matrix of x and y points |
| <code>gridsize</code> | number or vector indicating the size of the grid where to estimate the density |
| <code>pos</code> | Matrix of coordinates to evaluate the density |

Value

Vector of density estimates

Examples

```
x <- rnorm(500)
y <- x+rnorm(500)
kde2 <- approxk2d(cbind(x, y))
plot(x, y, pch=20, col=HSV(0, kde2/max(kde2), 1))
```

| | |
|----------------|--|
| aracne2regulon | <i>Regulon object generation from ARACNe results</i> |
|----------------|--|

Description

This function generates a regulon object from ARACNe results and the corresponding expression dataset

Usage

```
aracne2regulon(afile, eset, gene = FALSE, format = c("adj", "3col"),  
              verbose = TRUE)
```

Arguments

| | |
|---------|--|
| afile | Character string indicating the name of the ARACNe network file |
| eset | Either a character string indicating the name of the expression-dataset file, a ExpressionSet object or a gene expression matrix with genes (probes) in rows and samples in columns |
| gene | Logical, whether the probes should be collapsed at the gene level |
| format | Character string, indicating the format of the aracne file, either adj for adjacency matrixes generated by aracne, or 3col when the interactome is represented by a 3 columns text file, with regulator in the first column, target in the second and mutual information in the third column |
| verbose | Logical, whether progression messages should be printed in the terminal. |

Value

Regulon object

See Also

[msviper](#), [viper](#)

Examples

```
data(bcellViper, package="bcellViper")  
adjfile <- file.path(find.package("bcellViper"), "aracne", "bcellaracne.adj")  
regul <- aracne2regulon(adjfile, dset)  
print(regul)
```

aracne2regulon4cnv *Regulon object generation from ARACNe results corrected by cnv*

Description

This function generates a regulon object from ARACNe results and the corresponding expression dataset when correction for CNV have been applied

Usage

```
aracne2regulon4cnv(afile, eset, regeset, gene = FALSE,  
  format = c("adj", "3col"), verbose = TRUE)
```

Arguments

| | |
|---------|--|
| afile | Character string indicating the name of the ARACNe network file |
| eset | Either a character string indicating the name of the expression-dataset file, a ExpressionSet object or a gene expression matrix with genes (probes) in rows and samples in columns, where the expression was corrected by CNV |
| regeset | Either a character string indicating the name of the expression-dataset file, a ExpressionSet object or a gene expression matrix with genes (probes) in rows and samples in columns |
| gene | Logical, whether the probes should be collapsed at the gene level |
| format | Character string, indicating the format of the aracne file, either adj for adjacency matrixes generated by aracne, or 3col when the interactome is represented by a 3 columns text file, with regulator in the first column, target in the second and mutual information in the third column |
| verbose | Logical, whether progression messages should be printed in the terminal. |

Value

Regulon object

See Also

[msviper](#), [viper](#)

Examples

```
data(bcellViper, package="bcellViper")  
adjfile <- file.path(find.package("bcellViper"), "aracne", "bcellaracne.adj")  
regul <- aracne2regulon(adjfile, dset)  
print(regul)
```

aREA *analytic Rank-based Enrichment Analysis*

Description

This function performs wREA enrichment analysis on a set of signatures

Usage

```
aREA(eset, regulon, method = c("auto", "matrix", "loop"), minsize = 20,
      cores = 1, wm = NULL, verbose = FALSE)
```

Arguments

| | |
|---------|---|
| eset | Matrix containing a set of signatures, with samples in columns and traits in rows |
| regulon | Regulon object |
| method | Character string indicating the implementation, either auto, matrix or loop |
| minsize | Integer indicating the minimum allowed size for the regulons |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| wm | Optional numeric matrix of weights (0; 1) with same dimension as eset |
| verbose | Logical, whether a progress bar should be shown |

Value

List of two elements, enrichment score and normalized enrichment score

as.dist.signatureDistance
Distance matrix from signatureDistance objects

Description

This function transforms a signatureDistance object into a dist object

Usage

```
## S3 method for class 'signatureDistance'
as.dist(m, diag = FALSE, upper = FALSE)
```

Arguments

| | |
|-------|--------------------------------------|
| m | signatureDistance object |
| diag | parameter included for compatibility |
| upper | parameter included for compatibility |

Value

Object of class dist

| | |
|------------------|---------------------------------------|
| bootstrapmsviper | <i>msviper bootstraps integration</i> |
|------------------|---------------------------------------|

Description

This function integrates the bootstrap msviper results

Usage

```
bootstrapmsviper(mobj, method = c("mean", "median", "mode"))
```

Arguments

| | |
|--------|--|
| mobj | msviper object |
| method | Character string indicating the method to use, either mean, median or mode |

Value

msviper object

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
sig <- bootstrapTtest(dset, "description", c("CB", "CC"), "N")
mra <- msviper(sig, regulon)
plot(mra, cex=.7)
```

| | |
|----------------|---|
| bootstrapTtest | <i>Bootstrapped signature by t-test</i> |
|----------------|---|

Description

This function generates a bootstrapped signature matrix by t-test

Usage

```
bootstrapTtest(x, ...)

## S4 method for signature 'matrix'
bootstrapTtest(x, y, per = 100, seed = 1,
  cores = 1, verbose = TRUE)

## S4 method for signature 'ExpressionSet'
bootstrapTtest(x, pheno, group1, group2,
  per = 100, seed = 1, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| x | Matrix containing the test dataset |
| ... | Additional parameters added to keep compatibility |
| y | Matrix containing the reference dataset |
| per | Integer indicating the number of permutations |
| seed | Integer indicating the seed for the permutations, 0 for disable it |
| cores | Integer indicating the number of cores to use (set to 1 in Windows-based systems) |
| verbose | Logical whether progress should be reported |
| pheno | Character string indicating the phenotype data to use |
| group1 | Vector of character strings indicating the category from phenotype pheno to use as test group |
| group2 | Vector of character strings indicating the category from phenotype pheno to use as control group |

Value

Matrix of z-scores with genes in rows and permutations in columns

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
sig <- bootstrapTtest(d1[, 1:10], d1[, 11:20], per=100)
dim(sig)
plot(density(sig[1907, ]))
data(bcellViper, package="bcellViper")
sig <- bootstrapTtest(dset, "description", "CB", "N", per=100)
dim(sig)
plot(density(sig[1907, ]))
```

bootstrapViper

bootstrapsViper

Description

This function performs a viper analysis with bootstraps

Usage

```
bootstrapViper(eset, regulon, nes = TRUE, bootstraps = 10,
  eset.filter = FALSE, adaptive.size = TRUE, minsize = 20,
  mvws = 1, cores = 1, verbose = TRUE)
```


Arguments

| | |
|---------------|---|
| eset | ExpressionSet object or Numeric matrix containing the expression data, with samples in columns and genes in rows |
| regulon | Object of class regulon |
| nes | Logical, whether the enrichment score reported should be normalized |
| bootstraps | Integer indicating the number of bootstraps iterations to perform. Only the scale method is implemented with bootstraps. |
| eset.filter | Logical, whether the dataset should be limited only to the genes represented in the interactome |
| adaptive.size | Logical, whether the weighting scores should be taken into account for computing the regulon size |
| minsize | Integer indicating the minimum number of targets allowed per regulon |
| mvws | Number or vector indicating either the exponent score for the metaViper weights, or the inflection point and trend for the sigmoid function describing the weights in metaViper |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

A list containing a matrix of inferred activity for each regulator gene in the network across all samples and the corresponding standard deviation computed from the bootstrap iterations.

See Also

[viper](#)

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
res <- viper(d1[, 1:50], regulon, bootstraps=10) # Run only on 50 samples to reduce computation time
dim(d1)
d1[1:5, 1:5]
regulon
dim(res$nes)
res$nes[1:5, 1:5]
res$sd[1:5, 1:5]
```

comNames

Combinatorial annotation

Description

This function converts combinatorial annotations

Usage

```
comNames(x, annot)
```

Arguments

| | |
|-------|---|
| x | Character vector of gene name combinations, where the combinations are separated by – |
| annot | Vector of gene names with geneID as names attribute |

Value

Converted annotations

See Also

[msviper](#)

| | |
|----------|---|
| distMode | <i>Mode of continuous distributions</i> |
|----------|---|

Description

This function computes the mode for continuous distributions

Usage

```
distMode(x, adj = 1)
```

Arguments

| | |
|-----|---|
| x | Numeric data vector |
| adj | Number indicating the adjustment for the kernel bandwidth |

Value

Number

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
mean(d1[, 1])
median(d1[, 1])
distMode(d1[, 1])
plot(density(d1[, 1]))
abline(v=c(mean(d1[, 1]), median(d1[, 1]), distMode(d1[, 1])), col=c("green", "red", "blue"))
legend("topleft", c("Mean", "Median", "Mode"), col=c("green", "red", "blue"), lwd=4)
```

| | |
|---------|--|
| fcvarna | <i>Variance of columns for arrays with NA values</i> |
|---------|--|

Description

This function computes the variance by columns ignoring NA values

Usage

```
fcvarna(x)
```

Arguments

| | |
|---|----------------|
| x | Numeric matrix |
|---|----------------|

Value

1-column matrix with the variance by column results

Examples

```
data(bcellViper, package="bcellViper")
tmp <- exprs(dset)[, 1:10]
tmp[round(runif(100, 1, length(tmp)))] <- NA
fcvarna(tmp)
```

| | |
|-----------------|---|
| filterColMatrix | <i>Filter for columns of a matrix with no loss of col and row names</i> |
|-----------------|---|

Description

This function filters the columns of a matrix returning always a two dimensional matrix

Usage

```
filterColMatrix(x, filter)
```

Arguments

| | |
|--------|---------------------------------------|
| x | Matrix |
| filter | Logical or numerical index of columns |

Value

Matrix

| | |
|----------|--|
| filterCV | <i>Coefficient of variation filter</i> |
|----------|--|

Description

This function filter redundant probes based on the highest coefficient of variation

Usage

```
filterCV(expset, ...)

## S4 method for signature 'matrix'
filterCV(expset)

## S4 method for signature 'ExpressionSet'
filterCV(expset)
```

Arguments

| | |
|--------|---|
| expset | Expression set or Matrix containing the gene expression data, with samples in columns and probes in rows. The colnames attribute should contain the sample names and the rownames attribute should contain the unique geneIDs |
| ... | Additional parameters added to keep compatibility |

Value

CV filtered dataset

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
tmp <- rownames(d1)
tmp[round(runif(10, 1, length(tmp)))] <- tmp[1]
rownames(d1) <- tmp
dim(d1)
d1 <- filterCV(d1)
dim(d1)
```

| | |
|-----------------|--|
| filterRowMatrix | <i>Filter for rows of a matrix with no loss of col and row names</i> |
|-----------------|--|

Description

This function filters the rows of a matrix returning always a two dimensional matrix

Usage

```
filterRowMatrix(x, filter)
```

Arguments

x Matrix
filter Logical or numerical index of rows

Value

Matrix

frcv *Coefficient of variations for rows*

Description

This function computes the coefficient of variation (CV) by rows

Usage

```
frcv(x)
```

Arguments

x Numeric matrix

Value

1-column matrix with the coefficient of variation by row results

Examples

```
data(bcellViper, package="bcellViper")  
tmp <- exprs(dset)[1:10, ]  
tmp[round(runif(100, 1, length(tmp)))] <- NA  
frcv(tmp)
```

frvarna *Variance of rows for arrays with NA values*

Description

This function computes the variance by rows ignoring NA values

Usage

```
frvarna(x)
```

Arguments

x Numeric matrix

Value

1-column matrix with the variance by row results

Examples

```
data(bcellViper, package="bcellViper")
tmp <- exprs(dset)[1:10, ]
tmp[round(runif(100, 1, length(tmp)))] <- NA
frvarna(tmp)
```

groupPwea3

Proportionally Weighted Enrichment Analysis for gene-set groups

Description

This function performs a Proportionally Weighted Enrichment Analysis on groups of gene-sets

Usage

```
groupPwea3(rlist, groups, nullpw = NULL, alternative = c("two.sided",
  "less", "greater"), per = 0, minsize = 5, cores = 1,
  verbose = TRUE)
```

Arguments

| | |
|-------------|--|
| rlist | Named vector containing the scores to rank the expression profile or matrix where columns contains bootstrapped signatures |
| groups | List of gene-sets (regulons), each component is a list of two vectors: <i>TFmode</i> containing the TFMoA index (-1; 1) and <i>likelihood</i> containing the interaction relative likelihood |
| nullpw | Numerical matrix representing the null model, with genes as rows (geneID as rownames) and permutations as columns |
| alternative | Character string indicating the alternative hypothesis, either two.sided, greater or less |
| per | Integer indicating the number of permutations for the genes in case "nullpw" is omitted |
| minsize | Integer indicating the minimum size for the regulons |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

A list containing four matrices:

es Enrichment score
nes Normalized Enrichment Score
size Regulon size
p.value Enrichment p.value

integrateSignatures *Integrate signatures*

Description

This function integrates signatures represented as columns in the input matrix using self-weighting average

Usage

```
integrateSignatures(signature, score = 1)
```

Arguments

| | |
|-----------|--|
| signature | Numeric matrix containing the signatures as z-scores or NES, genes in rows and signatures in columns |
| score | Number indicating the exponent score for the weight |

Value

Vector containing the integrated signatures

Examples

```
data(bcellViper, package="bcellViper")
sig <- bootstrapTtest(dset, "description", "CB", "N", per=100)
isig <- integrateSignatures(sig)
plot(density(sig))
lines(density(isig, adj=1.5), col="red")
```

ledge *Leading-edge analysis*

Description

This function performs a Leading-Edge analysis on an object of class `msviper`

Usage

```
ledge(mobj)
```

Arguments

| | |
|------|-----------------------------------|
| mobj | <code>msviper</code> class object |
|------|-----------------------------------|

Value

`msviper` object updated with a ledge slot

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", "CB", "N")$statistic
mra <- msviper(sig, regulon)
mra <- ledge(mra)
summary(mra)
```

loadExpset

Loading expression sets

Description

This function load an expression file into a matrix

Usage

```
loadExpset(filename)
```

Arguments

filename Character string indicating the name of the expression file

Value

List containing a numeric matrix of expression data with samples in columns and probes in rows; and a vector of gene mapping annotations

msviper

msVIPER

Description

This function performs MAster Regulator INference Analysis

Usage

```
msviper(ges, regulon, nullmodel = NULL, pleiotropy = FALSE,
        minsize = 25, adaptive.size = FALSE, ges.filter = TRUE,
        synergy = 0, level = 10, pleiotropyArgs = list(regulators = 0.05,
        shadow = 0.05, targets = 10, penalty = 20, method = "adaptive"),
        cores = 1, verbose = TRUE)
```


Arguments

| | |
|-----------------------------|--|
| <code>ges</code> | Vector containing the gene expression signature to analyze, or matrix with columns containing bootstrapped signatures |
| <code>regulon</code> | Object of class <code>regulon</code> |
| <code>nullmodel</code> | Matrix of genes by permutations containing the NULL model signatures. A parametric approach equivalent to shuffle genes will be used if <code>nullmodel</code> is omitted. |
| <code>pleiotropy</code> | Logical, whether correction for pleiotropic regulation should be performed |
| <code>minsize</code> | Number indicating the minimum allowed size for the regulons |
| <code>adaptive.size</code> | Logical, whether the weight (likelihood) should be used for computing the regulon size |
| <code>ges.filter</code> | Logical, whether the gene expression signature should be limited to the genes represented in the interactome |
| <code>synergy</code> | Number indicating the synergy computation mode: (0) for no synergy computation; (0-1) for establishing the p-value cutoff for individual TFs to be included in the synergy analysis; (>1) number of top TFs to be included in the synergy analysis |
| <code>level</code> | Integer, maximum level of combinatorial regulation |
| <code>pleiotropyArgs</code> | list of 5 numbers for the pleiotropy correction indicating: regulators p-value threshold, pleiotropic interaction p-value threshold, minimum number of targets in the overlap between pleiotropic regulators, penalty for the pleiotropic interactions and the pleiotropy analysis method, either absolute or adaptive |
| <code>cores</code> | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| <code>verbose</code> | Logical, whether progression messages should be printed in the terminal |

Value

A `msviper` object containing the following components:

signature The gene expression signature

regulon The final regulon object used

es Enrichment analysis results including regulon size, normalized enrichment score and p-value

param `msviper` parameters, including `minsize`, `adaptive.size`

See Also

[viper](#)

Examples

```
data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
dnull <- ttestNull(dset, "description", c("CB", "CC"), "N", per=100) # Only 100 permutations to reduce computat
mra <- msviper(sig, regulon, dnull)
plot(mra, cex=.7)
```

| | |
|---------------|--------------------------|
| msviper-class | <i>The msviper class</i> |
|---------------|--------------------------|

Description

This class contains the results generated by the msviper function

Slots

signature: Matrix containing the gene expression signature

regulon: Object of class regulon

es: List containing 6 objects:

es\$es: Named vector of class numeric containing the enrichment scores

es\$nes: Named vector of class numeric containing the normalized enrichment scores

es\$nes.se: Named vector of class numeric containing the standard error for the normalized enrichment score

es\$size: Named vector of class numeric containing the size -number of target genes- for each regulator

es\$p.value: Named vector of class numeric containing the enrichment p-values

es\$nes.bt: Matrix containing the normalized enrichment score if the msviper test is performed with bootstraps

param: List containing 3 elements:

param\$minsize: Integer indicating the minimum allowed size for the regulons

param\$adaptive.size: Logical indicating whether the weight (likelihood) should be used for computing the regulon size

param\$iterative: Logical indicating whether a two step analysis with adaptive redundancy estimation should be performed

nullmodel: Matrix of genes by permutations containing the NULL model signatures

ledge: List containing the leading edge genes for each regulator. This slot is added by the ledge function

shadow: Two columns matrix containing the gene names for the shadow pairs. The first column contain the most probable regulator and the second column the one that was identified because a shadow effect

| | |
|--------------|----------------------------------|
| msviperAnnot | <i>msVIPER annotation change</i> |
|--------------|----------------------------------|

Description

This function changes the annotation of genes in msviper objects

Usage

```
msviperAnnot(mobj, annot, complete = TRUE)
```

Arguments

| | |
|----------|--|
| obj | msviper object generated by msviper function |
| annot | Vector of character strings containing the gene names and gene identifiers as vector names attribute |
| complete | Logical, whether the signature and target names should be also transformed |

Value

msviper object with updated annotations

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", "CB", "N")$statistic
mra <- msviper(sig, regulon)
tmp <- unique(c(names(mra$regulon), rownames(mra$signature)))
annot <- 1:length(tmp)
names(annot) <- tmp
plot(mra, cex=.7)
mra <- msviperAnnot(mra, annot)
plot(mra, cex=.7)
```

msviperClass

msVIPER class

Description

This function generates an instance of the msviper class from a signature, NES signature and regulon object

Usage

```
msviperClass(nes, signature, regulon, nullmodel = NULL)
```

Arguments

| | |
|-----------|--|
| nes | Numeric vector of NES values |
| signature | Numeric vector of gene expression signature |
| regulon | Instance of class regulon |
| nullmodel | Optional matrix containing the signatures for the null model |

Value

msviper class object

Examples

```

data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
mra <- msviper(sig, regulon)
mra1 <- msviperClass(mra$es$nes, sig, regulon)
summary(mra1)
plot(mra1)

```

msviperCombinatorial *msviper combinatorial analysis*

Description

This function performs combinatorial analysis for msviper objects

Usage

```

msviperCombinatorial(mobj, regulators = 100, nullmodel = NULL,
  minsize = NULL, adaptive.size = NULL, level = 10, cores = 1,
  processAll = FALSE, verbose = TRUE)

```

Arguments

| | |
|---------------|--|
| mobj | msviper object generated by msviper function |
| regulators | Either a number between 0 and 1 indicating the p-value cutoff for individual TFs to be included in the combinations analysis; (>1) indicating the number of top TFs to be included in the combinations analysis; or a vector of character strings indicating the TF IDs to be included in the analysis |
| nullmodel | Matrix of genes by permutations containing the NULL model signatures. Taken from mobj by default |
| minsize | Number indicating the minimum allowed size for the regulons, taken from mobj by default |
| adaptive.size | Logical, whether the weight (likelihood) should be used for computing the size, taken from mobj by default |
| level | Integer, maximum level of combinatorial regulation |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| processAll | Logical, whether all pairs, even if not significant, should be processed for synergy |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

A msviper object

See Also

[msviper](#)

Examples

```

data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
dnull <- ttestNull(dset, "description", c("CB", "CC"), "N", per=100) # Only 100 permutations to reduce computat
mra <- msviper(sig, regulon, dnull)
mra <- msviperCombinatorial(mra, 20)
plot(mra, cex=.7)

```

msviperSynergy

*msviper synergy analysis***Description**

This function performs a synergy analysis for combinatorial regulation

Usage

```

msviperSynergy(mobj, per = 1000, seed = 1, cores = 1,
  verbose = TRUE)

```

Arguments

| | |
|---------|--|
| mobj | msviper object containing combinatorial regulation results generated by msviperCombinatorial |
| per | Integer indicating the number of permutations |
| seed | Integer indicating the seed for the permutations, 0 for disable it |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

Updated msviper object containing the synergy p-value

See Also

[msviper](#)

Examples

```

data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
dnull <- ttestNull(dset, "description", c("CB", "CC"), "N", per=100) # Only 100 permutations to reduce computat
mra <- msviper(sig, regulon, dnull)
mra <- msviperCombinatorial(mra, 20)
mra <- msviperSynergy(mra)
summary(mra)

```

| | |
|--------------|-----------------------------|
| plot.msviper | <i>Plot msviper results</i> |
|--------------|-----------------------------|

Description

This function generate a plot for msviper results showing the enrichment of the target genes for each significant master regulator on the gene expression signature

Usage

```
## S3 method for class 'msviper'
plot(x, mrs = 10, color = c("cornflowerblue",
  "salmon"), pval = NULL, bins = 500, cex = 0, density = 0,
  smooth = 0, sep = 0.2, hybrid = TRUE, include = c("expression",
  "activity"), gama = 2, ...)
```

Arguments

| | |
|---------|---|
| x | msviper object produced by msviper function |
| mrs | Either an integer indicating the number of master regulators to include in the plot, or a character vector containing the names of the master regulators to include in the plot |
| color | Vector of two components indicating the colors for the negative and positive parts of the regulon |
| pval | Optional matrix of p-values to include in the plot |
| bins | Number of bins to split the vector of scores in order to compute the density color of the bars |
| cex | Number indicating the text size scaling, 0 indicates automatic scaling |
| density | Integer indicating the number of steps for the kernel density. Zero for not plotting it |
| smooth | Number indicating the proportion of point for smoothing the density distribution. Zero for not using the smoother |
| sep | Number indicating the separation from figure and text |
| hybrid | Logical, whether the 3-tail approach used for computing the enrichment should be reflected in the plot |
| include | Vector indicating the information to include as heatmap to the right of the msviper plot: expression and activity |
| gama | Positive number indicating the exponential transformation for the activity and expression color scale |
| ... | Given for compatibility to the plot generic function |

Value

Nothing, a plot is generated in the default output device

See Also

[msviper](#)

Examples

```

data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
dnull <- ttestNull(dset, "description", c("CB", "CC"), "N", per=100) # Only 100 permutations to reduce computat
mra <- msviper(sig, regulon, dnull)
plot(mra, cex=.7)

```

pruneRegulon

*Prune Regulons***Description**

This function limits the maximum size of the regulons

Usage

```

pruneRegulon(regulon, cutoff = 50, adaptive = TRUE,
             eliminate = FALSE, wm = NULL)

```

Arguments

| | |
|-----------|---|
| regulon | Object of class regulon |
| cutoff | Number indicating the maximum size for the regulons (maximum number of target genes) |
| adaptive | Logical, whether adaptive size should be used (i.e. $\text{sum}(\text{likelihood}^2)$) |
| eliminate | Logical whether regulons smaller than cutoff should be eliminated |
| wm | Optional numeric vector of weights (0; 1) for the genes |

Value

Pruned regulon

See Also

[viper](#), [msviper](#)

Examples

```

data(bcellViper, package="bcellViper")
hist(sapply(regulon, function(x) sum(x$likelihood)/max(x$likelihood)), nclass=20)
preg <- pruneRegulon(regulon, 400)
hist(sapply(preg, function(x) sum(x$likelihood)/max(x$likelihood)), nclass=20)

```

pwea3NULLf *Null model function*

Description

This function generates the NULL model function, which computes the normalized enrichment score and associated p-value

Usage

```
pwea3NULLf(pwnull, cores = 1, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| pwnull | Object generated by pwea3NULLgroups function |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

List of function to compute NES and p-value

pwea3NULLgroups *Regulon-specific NULL model*

Description

This function generates the regulon-specific NULL models

Usage

```
pwea3NULLgroups(pwnull, groups, cores = 1, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| pwnull | Numerical matrix representing the null model, with genes as rows (geneID as rownames) and permutations as columns |
| groups | List containing the regulons |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

A list containing two elements:

groups Regulon-specific NULL model containing the enrichment scores
ss Direction of the regulon-specific NULL model

| | |
|---------------|--------------------------|
| regulon-class | <i>The regulon class</i> |
|---------------|--------------------------|

Description

This class contains interactome data

Slots

List of regulators with the following slots:

tfmode: Named vector of class `numeric` containing the regulator mode of action scores, with target genes as name attribute

likelihood: Vector of class `numeric` containing the relative likelihood for each target gene

| | |
|----------|----------------------------------|
| rowTtest | <i>Student's t-test for rows</i> |
|----------|----------------------------------|

Description

This function performs a Student's t-test on each row of a matrix

Usage

```
rowTtest(x, ...)

## S4 method for signature 'matrix'
rowTtest(x, y = NULL, mu = 0,
         alternative = "two.sided")

## S4 method for signature 'ExpressionSet'
rowTtest(x, pheno, group1, group2 = NULL,
         mu = 0, alternative = "two.sided")
```

Arguments

| | |
|--------------------------|---|
| <code>x</code> | ExpressionSet object or Numerical matrix containing the test samples |
| <code>...</code> | Additional parameters added to keep compatibility |
| <code>y</code> | Optional numerical matrix containing the reference samples. If omitted <code>x</code> will be tested against mean = <code>mu</code> |
| <code>mu</code> | Number indicating the alternative hypothesis when <code>y</code> is omitted |
| <code>alternative</code> | Character string indicating the tail for the test, either <code>two.sided</code> , <code>greater</code> or <code>lower</code> |
| <code>pheno</code> | Character string indicating the phenotype data to use |
| <code>group1</code> | Vector of character strings indicating the category from phenotype <code>pheno</code> to use as test group |
| <code>group2</code> | Vector of character strings indicating the category from phenotype <code>pheno</code> to use as control group |

Value

List of Student-t-statistic (`statistic`) and p-values (`p.value`)

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
res <- rowTtest(d1[, 1:10], d1[, 11:20])
res$statistic[1:5, ]
res$p.value[1:5, ]
data(bcellViper, package="bcellViper")
res <- rowTtest(dset, "description", "CB", "N")
res$statistic[1:5, ]
res$p.value[1:5, ]
```

`scale.signatureDistance`

Scaling of signatureDistance objects

Description

This function scales the `signatureDistance` so its range is (-1, 1)

Usage

```
## S3 method for class 'signatureDistance'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

| | |
|---------------------|--|
| <code>x</code> | signatureDistance object |
| <code>center</code> | Not used, given for compatibility with the generic function <code>scale</code> |
| <code>scale</code> | Not used, given for compatibility with the generic function <code>scale</code> |

Value

Scaled `signatureDistance` object

`scaleGroups`

Signatures with grouping variable

Description

`scaleGroups` compares each group vs. the remaining groups using a Student's t-test

Usage

```
scaleGroups(x, groups)
```

Arguments

| | |
|--------|--|
| x | Numerical matrix with genes in rows and samples in columns |
| groups | Vector of same length as columns has the dset containing the labels for grouping the samples |

Details

This function compute signatures using groups information

Value

Numeric matrix of signatures (z-scores) with genes in rows and groups in columns

Examples

```
data(bcellViper, package="bcellViper")
res <- scaleGroups(exprs(dset)[, 1:20], rep(1:4, rep(5, 4)))
res[1:5, ]
```

shadow

Shadow analysis for msviper objects

Description

This function performs shadow analysis on msviper objects

Usage

```
shadow(mobj, regulators = 0.01, targets = 10, shadow = 0.01,
       per = 1000, nullmodel = NULL, minsize = NULL,
       adaptive.size = NULL, iterative = NULL, seed = 1, cores = 1,
       verbose = TRUE)
```

Arguments

| | |
|---------------|--|
| mobj | msviper object generated by msviper |
| regulators | This parameter represent different ways to select a subset of regulators for performing the shadow analysis, it can be either a p-value cutoff, the total number of regulons to be used for computing the shadow effect, or a vector of regulator ids to be considered |
| targets | Integer indicating the minimum number of common targets to compute shadow analysis |
| shadow | Number indicating the p-value threshold for the shadow effect |
| per | Integer indicating the number of permutations |
| nullmodel | Null model in marix format |
| minsize | Integer indicating the minimum size allowed for the regulons |
| adaptive.size | Logical, whether the target weight should be considered when computing the regulon size |

| | |
|-----------|--|
| iterative | Logical, whether a two step analysis with adaptive redundancy estimation should be performed |
| seed | Integer indicating the seed for the permutations, 0 for disable it |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

An updated msviper object with an additional slot (shadow) containing the shadow pairs

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
sig <- rowTtest(dset, "description", c("CB", "CC"), "N")$statistic
dnull <- ttestNull(dset, "description", c("CB", "CC"), "N", per=100) # Only 100 permutations to reduce computat
mra <- msviper(sig, regulon, dnull)
mra <- shadow(mra, regulators=10)
summary(mra)
```

| | |
|---------------|----------------------------------|
| shadowRegulon | <i>Correction for pleiotropy</i> |
|---------------|----------------------------------|

Description

This function penalize the regulatory interactions based on pleiotropy analysis

Usage

```
shadowRegulon(ss, nes, regul, regulators = 0.05, shadow = 0.05,
  targets = 10, penalty = 2, method = c("absolute", "adaptive"))
```

Arguments

| | |
|------------|---|
| ss | Named vector containing the gene expression signature |
| nes | Named vector containing the normalized enrichment scores |
| regul | Regulon object |
| regulators | Number indicating the number of top regulators to consider for the analysis or the p-value threshold for considering significant regulators |
| shadow | Number indicating the p-value threshold for considering a significant shadow effect |
| targets | Integer indicating the minimal number of overlapping targets to consider a pair of regulators for pleiotropy analysis |
| penalty | Number higher than 1 indicating the penalty for the pleiotropic interactions. 1 = no penalty |
| method | Character string indicating the method to use for computing the pleiotropy, either absolute or adaptive |

Value

Corrected regulon object

| | |
|-------------------|---------------------------|
| signatureDistance | <i>Signature Distance</i> |
|-------------------|---------------------------|

Description

This function computes the similarity between columns of a data matrix

Usage

```
signatureDistance(dset1, dset2 = NULL, nn = NULL, groups = NULL,
  scale. = TRUE, two.tails = TRUE, ws = 2)
```

Arguments

| | |
|-----------|--|
| dset1 | Dataset of any type in matrix format, with features in rows and samples in columns |
| dset2 | Optional Dataset. If provided, distance between columns of dset and dset2 are computed and reported as rows and columns, respectively; if not, distance between all possible pairs of columns from dset are computed |
| nn | Optional size for the signature, default is either the full signature or 10 percent of it, depending on whether ws=0 or not |
| groups | Optional vector indicating the group ID of the samples |
| scale. | Logical, whether the data should be scaled |
| two.tails | Logical, whether a two tails, instead of 1 tail test should be performed |
| ws | Number indicating the exponent for the weighting the signatures, the default of 0 is uniform weighting, 1 is weighting by SD |

Value

Object of class signatureDistance as a matrix of normalized enrichment scores

Examples

```
data(bcellViper, package="bcellViper")
dd <- signatureDistance(exprs(dset))
dd[1:5, 1:5]
scale(dd)[1:5, 1:5]
as.matrix(as.dist(dd))[1:5, 1:5]
```

signatureDistance-class
signatureDistance

Description

This class contains the results generated by signatureDistance function.

Slots

Matrix of class numeric containing the similarity scores

sigT *Sigmoid transformation*

Description

This function transforms a numeric vector using a sigmoid function

Usage

```
sigT(x, slope = 20, inflection = 0.5)
```

Arguments

| | |
|------------|---|
| x | Numeric vector |
| slope | Number indicating the slope at the inflection point |
| inflection | Number indicating the inflection point |

Value

Numeric vector

summary.msviper *List msviper results*

Description

This function generates a table of msviper results

Usage

```
## S3 method for class 'msviper'
summary(object, mrs = 10, ...)
```

Arguments

| | |
|--------|--|
| object | msviper object |
| mrs | Either number of top MRs to report or vector containing the genes to display |
| ... | Given for compatibility with the summary generic function |

Value

Data.frame with results

| | |
|-----------|---|
| ttestNull | <i>Null model by sample permutation testing</i> |
|-----------|---|

Description

This function performs sample permutation and t-test to generate a null model

Usage

```
ttestNull(x, ...)

## S4 method for signature 'matrix'
ttestNull(x, y, per = 1000, repos = TRUE,
  seed = 1, cores = 1, verbose = TRUE)

## S4 method for signature 'ExpressionSet'
ttestNull(x, pheno, group1, group2, per = 1000,
  repos = TRUE, seed = 1, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| x | ExpressionSet object or Matrix containing the test dataset |
| ... | Additional parameters added to keep compatibility |
| y | Matrix containing the reference dataset |
| per | Integer indicating the number of permutations |
| repos | Logical, whether the permutations should be performed with reposition |
| seed | Integer indicating the seed for the permutations, 0 for disable it |
| cores | Integer indicating the number of cores to use (set to 1 in windows systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |
| pheno | Character string indicating the phenotype data to use |
| group1 | Vector of character strings indicating the category from phenotype pheno to use as test group |
| group2 | Vector of character strings indicating the category from phenotype pheno to use as control group |

Value

Matrix of z-scores with genes in rows and permutations in columns

See Also

[msviper](#), [viper](#)

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
dnull <- ttestNull(d1[, 1:10], d1[, 11:20], per=100)
dim(dnull)
plot(density(dnull))
data(bcellViper, package="bcellViper")
dnull <- ttestNull(dset, "description", "CB", "CC", per=100)
dim(dnull)
plot(density(dnull))
```

viper

VIPER

Description

This function performs Virtual Inference of Protein-activity by Enriched Regulon analysis

Usage

```
viper(eset, regulon, dnull = NULL, pleiotropy = FALSE, nes = TRUE,
      method = c("none", "scale", "rank", "mad", "ttest"), bootstraps = 0,
      minsize = 25, adaptive.size = FALSE, eset.filter = TRUE,
      mvws = 1, pleiotropyArgs = list(regulators = 0.05, shadow = 0.05,
      targets = 10, penalty = 20, method = "adaptive"), cores = 1,
      verbose = TRUE)
```

Arguments

| | |
|---------------|--|
| eset | ExpressionSet object or Numeric matrix containing the expression data or gene expression signatures, with samples in columns and genes in rows |
| regulon | Object of class regulon or list of objects of class regulon for metaVIPER analysis |
| dnull | Numeric matrix for the null model, usually generated by nullTtest |
| pleiotropy | Logical, whether correction for pleiotropic regulation should be performed |
| nes | Logical, whether the enrichment score reported should be normalized |
| method | Character string indicating the method for computing the single samples signature, either scale, rank, mad, ttest or none |
| bootstraps | Integer indicating the number of bootstraps iterations to perform. Only the scale method is implemented with bootstraps. |
| minsize | Integer indicating the minimum number of targets allowed per regulon |
| adaptive.size | Logical, whether the weighting scores should be taken into account for computing the regulon size |
| eset.filter | Logical, whether the dataset should be limited only to the genes represented in the interactome #' @param mvws Number or vector indicating either the exponent score for the metaViper weights, or the inflection point and trend for the sigmoid function describing the weights in metaViper |

| | |
|----------------|---|
| pleiotropyArgs | list of 5 numbers for the pleiotropy correction indicating: regulators p-value threshold, pleiotropic interaction p-value threshold, minimum number of targets in the overlap between pleiotropic regulators, penalty for the pleiotropic interactions and the method for computing the pleiotropy, either absolute or adaptive |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |

Value

A matrix of inferred activity for each regulator gene in the network across all samples

See Also

[msviper](#)

Examples

```
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
res <- viper(d1, regulon)
dim(d1)
d1[1:5, 1:5]
regulon
dim(res)
res[1:5, 1:5]
```

viperRPT

viperRPT

Description

This function computes residual post-translational activity

Usage

```
viperRPT(vipermat, expmat, weights = matrix(1, nrow(vipermat),
  ncol(vipermat), dimnames = list(rownames(vipermat), colnames(vipermat))),
  method = c("spline", "lineal", "rank"), robust = FALSE, cores = 1)
```

Arguments

| | |
|----------|---|
| vipermat | Numeric matrix containing the viper protein activity inferences |
| expmat | Numeric matrix or expressionSet containing the expression data |
| weights | List of numeric matrix of sample weights |
| method | Character string indicating the method to use, either rank, lineal or spline |
| robust | Logical, whether the contribution of outliers is down-weighted by using a gaussian kernel estimate for the join probability density |
| cores | Integer indicating the number of cores to use |

Value

Matrix of RPT-activity values

See Also

[viper](#)

Examples

```
data(bcellViper, package="bcellViper")
vipermat <- viper(dset, regulon)
rpt <- viperRPT(vipermat, dset)
rpt[1:5, 1:5]
```

| | |
|----------------|--|
| viperSignature | <i>Generic S4 method for signature and sample-permutation null model for VIPER</i> |
|----------------|--|

Description

This function generates a viperSignature object from a test dataset based on a set of samples to use as reference

Usage

```
viperSignature(eset, ...)

## S4 method for signature 'ExpressionSet'
viperSignature(eset, pheno, refgroup,
  method = c("zscore", "ttest", "mean"), per = 100, bootstrap = TRUE,
  seed = 1, cores = 1, verbose = TRUE)

## S4 method for signature 'matrix'
viperSignature(eset, ref, method = c("zscore",
  "ttest", "mean"), per = 100, bootstrap = TRUE, seed = 1,
  cores = 1, verbose = TRUE)
```

Arguments

| | |
|-----------|--|
| eset | ExpressionSet object or numeric matrix containing the test dataset, with genes in rows and samples in columns |
| ... | Additional parameters added to keep compatibility |
| pheno | Character string indicating the phenotype data to use |
| refgroup | Vector of character string indicatig the category of pheno to use as reference group |
| method | Character string indicating how to compute the signature and null model, either ttest, zscore or mean |
| per | Integer indicating the number of sample permutations |
| bootstrap | Logical, whether null model should be estimated with bootstrap. In this case, only reference samples are used. |

| | |
|---------|---|
| seed | Integer indicating the seed for the random sample generation. The system default is used when set to zero |
| cores | Integer indicating the number of cores to use (only 1 in Windows-based systems) |
| verbose | Logical, whether progression messages should be printed in the terminal |
| ref | Numeric matrix containing the reference samples (columns) and genes in rows |

Value

viperSignature S3 object containing the signature and null model

Examples

```
data(bcellViper, package="bcellViper")
ss <- viperSignature(dset, "description", c("N", "CB", "CC"), per=100) # Only 100 permutations to reduce comput
res <- viper(ss, regulon)
dim(exprs(dset))
exprs(dset)[1:5, 1:5]
regulon
dim(res)
exprs(res)[1:5, 1:5]
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
pos <- pData(dset)[["description"]] %in% c("N", "CB", "CC")
ss <- viperSignature(d1[, !pos], d1[, pos], per=100) # Only 100 permutations to reduce computation time, but it
res <- viper(ss, regulon)
dim(d1)
d1[1:5, 1:5]
regulon
dim(res)
res[1:5, 1:5]
```

viperSignature-class *viperSignature*

Description

This class contains the results produced by the viperSignature function

Slots

signature: Matrix of class numeric with genes in rows and samples in columns containing the gene expression signatures

nullmodel: Matrix of class numeric with genes in rows and permutations in columns containing the sample-permutation based signatures to be used as NULL model

| | |
|-----------------|-------------------------|
| viperSimilarity | <i>VIPER similarity</i> |
|-----------------|-------------------------|

Description

If `ws` is a single number, weighting is performed using an exponential function. If `ws` is a 2 numbers vector, weighting is performed with a symmetric sigmoid function using the first element as inflection point and the second as trend.

Usage

```
viperSimilarity(x, nn = NULL, ws = c(4, 2), method = c("two.sided",  
"greater", "less"))
```

Arguments

| | |
|---------------------|---|
| <code>x</code> | Numeric matrix containing the VIPER results with samples in columns and regulators in rows |
| <code>nn</code> | Optional number of top regulators to consider for computing the similarity |
| <code>ws</code> | Number indicating the weighting exponent for the signature, or vector of 2 numbers indicating the inflection point and the value corresponding to a weighting score of .1 for a sigmoid transformation, only used if <code>nn</code> is omitted |
| <code>method</code> | Character string indicating whether the most active (<code>greater</code>), less active (<code>less</code>) or both tails (<code>two.sided</code>) of the signature should be used for computing the similarity |

Details

This function computes the similarity between VIPER signatures

Value

signatureDistance object

Examples

```
data(bcellViper, package="bcellViper")  
dd <- viperSimilarity(exprs(dset))  
dd[1:5, 1:5]  
scale(dd)[1:5, 1:5]  
as.matrix(as.dist(dd))[1:5, 1:5]
```

Index

aecdf, 2
approxk2d, 3
aracne2regulon, 4
aracne2regulon4cnv, 5
aREA, 6
as.dist.signatureDistance, 6

bootstrapmsviper, 7
bootstrapTtest, 7
bootstrapTtest, ExpressionSet-method
 (bootstrapTtest), 7
bootstrapTtest, matrix-method
 (bootstrapTtest), 7
bootstrapViper, 8

comNames, 9

distMode, 10

fcvarna, 11
filterColMatrix, 11
filterCV, 12
filterCV, ExpressionSet-method
 (filterCV), 12
filterCV, matrix-method (filterCV), 12
filterRowMatrix, 12
frcv, 13
frvarna, 13

groupPwea3, 14

integrateSignatures, 15

ledge, 15
loadExpset, 16

msviper, 4, 5, 7, 8, 10, 16, 16, 19–23, 28, 32, 33
msviper-class, 18
msviperAnnot, 18
msviperClass, 19
msviperCombinatorial, 20
msviperSynergy, 21

plot.msviper, 22

pruneRegulon, 23
pwea3NULLf, 24
pwea3NULLgroups, 24

regulon-class, 25
rowTtest, 25
rowTtest, ExpressionSet-method
 (rowTtest), 25
rowTtest, matrix-method (rowTtest), 25

scale.signatureDistance, 26
scaleGroups, 26
shadow, 27
shadowRegulon, 28
signatureDistance, 29
signatureDistance-class, 30
sigT, 30
summary.msviper, 30

ttestNull, 31
ttestNull, ExpressionSet-method
 (ttestNull), 31
ttestNull, matrix-method (ttestNull), 31

viper, 4, 5, 9, 17, 23, 32, 32, 34
viperRPT, 33
viperSignature, 34
viperSignature, ExpressionSet-method
 (viperSignature), 34
viperSignature, matrix-method
 (viperSignature), 34
viperSignature-class, 35
viperSimilarity, 36