# Package 'viper'

October 8, 2014

**Version** 1.0.0

**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

**Suggests** bcellViper

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

**License** GPL (>=2)

**biocViews**
SystemsBiology, NetworkEnrichment, GeneExpression,FunctionalPrediction, GeneRegulation

**Collate** 'general.r' 'internal.r' 'ledge.r' 'msviper.r' 'plot.r'
'shadow.r' 'viper.r' 'aracne.r' 'signatureDistance.r' 'clases.r'

## R topics documented:

---

`aracne2regulon` *Regulon object generation from ARACNe results*

---

## 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)
```

---

## 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` *msviper bootstraps integration*

---

## 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 *Bootstrapped signature by t-test*

## Description

This function generates a bootstrapped signature matrix by t-test

## Usage

```
## S4 method for signature matrix
bootstrapTtest(x, y, per = 100, seed=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 |
| 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 #' |
| 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

## 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, ]))
```

## comNames *Combinatorial annotation*

### Description

This function convers 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 *Mode of continuous distributions*

### 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 *Variance of columns for arrays with NA values*

---

## 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 *Filter for columns of a matrix with no loss of col and row names*

---

## 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 *Coefficient of variation filter*

---

### Description

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

### Usage

```
## 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 |

### 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 *Filter for rows of a matrix with no loss of col and row names*

### 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 *Coeficient 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, tnorm = TRUE, minsize = 5, tw = 1, lw = 1,
  simsig = TRUE, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| rlist | Named vector containing the scores to rank the expression profile or matrix where columns contains bootstraped signatures |
| groups | List of gene-sets (regulons), each component is a list of two vectors: *TFmode* containing the TFMoA index (-1; 1) and *likelihood* 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 ommited |
| tnorm | Logical, whether the ranking vector should be normally transformed after the copula transformation |
| minsize | Integer indicating the minimum size for the regulons |
| tw | Number indicating the exponent for the target pleotropy weight |
| lw | Number indicating the exponent for the edge likelihood weight |
| simsig | Logical, whether a symetrical distribution of the signature is used |
| 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 | msviper 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, minsize = 25,
  adaptive.size = FALSE, iterative = FALSE,
  ges.filter = TRUE, synergy = 0, level = 10,
  verbose = TRUE)
```

### Arguments

| | |
|---|---|
| ges | Vector containing the gene expression signature to analyze, or matrix with columns containing bootstraped signatures |
| regulon | Object of class regulon |
| nullmodel | Matrix of genes by permutations containing the NULL model signatures. A parametric approach equivalent to shuffle genes will be used if nullmodel is ommitted. |
| minsize | Number indicating the minimum allowed size for the regulons |
| adaptive.size | Logical, whether the weight (likelihood) should be used for computing the regulon size |

| | |
|---|---|
| iterative | Logical, whether a two step analysis with adaptive redundancy estimation should be performed |
| ges.filter | Logical, whether the gene expression signature should be limited to the genes represented in the interactome |
| synergy | 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 |
| level | Integer, maximum level of combinatorial regulation |
| verbose | 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`, `iterative`

## 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=1000)
mra <- msviper(sig, regulon, dnull)
plot(mra, cex=.7)
```

---

# msviper-class *The msviper class*

---

## 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 probble regulator and the second column the one that was identified because a shadow effect

---

`msviperAnnot` *msVIPER annotation change*

---

## Description

This function changes the annotation of genes in msviper objects

## Usage

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

## Arguments

| | |
|---|---|
| `mobj` | msviper object generated by `msviper` function |
| `annot` | Vector os 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)
```

---

# 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,
  iterative = NULL, level = 10, 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 |
| iterative | Logical, whether a two step analysis with adaptive redundancy estimation should be performed, taken from mobj by default |
| level | Integer, maximum level of combinatorial regulation |
| 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")
mra <- msviper(sig, regulon, dnull)
mra <- msviperCombinatorial(mra, 50)
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,
  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 |
| verbose | Logical, whether progression messages should be printed in the terminal |

### Value

Updated msviper object containing the sygergy 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")
mra <- msviper(sig, regulon, dnull)
mra <- msviperCombinatorial(mra, 50)
mra <- msviperSynergy(mra)
summary(mra)
```

---

`plot.msviper` *Plot msviper results*

---

## 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` | Integrer 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 computingthe 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=1000)
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, eliminate = FALSE)
```

## Arguments

| | |
|---|---|
| regulon | Object of class regulon |
| cutoff | Number indicating the maximum size for the regulons (maximum number of target genes) |
| eliminate | Logical whether regulons smalles than cutoff should be eliminated |

## Value

Prunned 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, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| pwnull | Object generated by pwea3NULLgroups function |
| 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, tnorm = TRUE, tw = 1,
  lw = 1, simsig = TRUE, 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 |
| tnorm | Logical, whether the ranking vector should be normally transformed after the copula transformation |
| tw | Number indicating the exponent for the target pleotropy weight |
| lw | Number indicating the exponent for the edge likelihood weight |
| simsig | Logical, whether a symetrical distribution of the signature is used |
| 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 *The regulon class*

---

### 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 *Student's t-test for rows*

## Description

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

## Usage

```
## 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

| | |
|---|---|
| x | ExpressionSet object or Numerical matrix containing the test samples |
| y | Optional numerical matrix containing the reference samples. If ommited x will be tested against mean = mu |
| mu | Number indicating the alternative hypothesis when y is ommited |
| alternative | Character string indicating the tail for the test, either two.sided, greater or lower |
| 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

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

| | |
|---|---|
| x | signatureDistance object |
| center | Not used, given for compatibility with the generic funtion scale |
| scale | Not used, given for compatibility with the generic function scale |

## 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, 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 |
| 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=1000)
mra <- msviper(sig, regulon, dnull)
mra <- shadow(mra)
summary(mra)
```

---

# signatureDistance *Signature Distance*

---

## 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 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

---

# 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` *Null model by sample permutation testing*

## Description

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

## Usage

```
## S4 method for signature matrix
ttestNull(x, y, per = 1000, repos = TRUE, seed = 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 |
| 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 |
| 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, tw = 0.5, lw = 1,
  nes = TRUE,
  method = c("scale", "rank", "mad", "ttest", "none"),
  minsize = 25, adaptive.size = FALSE,
  eset.filter = TRUE, nes.method = "analytic",
  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 |
| dnull | Numeric matrix for the null model, usually generated by nullTtest |
| tw | Number indicating the power transformation for target weight |
| lw | Number indicating the power transformation for the likelihood weight |
| 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 |
| 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 |
| nes.method | Character string indicating the method for estimating the NES, either analytical, empirical or fit. Fit method is recommended only for large datasets (>500 samples) |
| 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]
```

---

`viperSignature` *Generic S4 method for signature and sample-permutation null model for VIPER*

---

## Description

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

## Usage

```
## S4 method for signature matrix
viperSignature(eset, ref, per = 1000, seed = 1, verbose = TRUE)
## S4 method for signature ExpressionSet
viperSignature(eset, pheno, refgroup, per = 1000, seed = 1, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `eset` | ExpressionSet object or numeric matrix containing the test dataset, with genes in rows and samples in columns |
| `pheno` | Character string indicating the phenotype data to use |
| `refgroup` | Vector of character string indicatig the category of `pheno` to use as reference group |
| `per` | Integer indicating the number of sample permutations |
| `seed` | Integer indicating the seed for the random sample generation. The system default is used when set to zero |
| `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"))
res <- viper(ss, regulon)
dim(exprs(dset))
exprs(dset)[1:5, 1:5]
regulon
dim(res)
res[1:5, 1:5]
data(bcellViper, package="bcellViper")
d1 <- exprs(dset)
ss <- viperSignature(d1[, -(1:5)], d1[, 1:5])
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 ontaining the sample-permutation based signatures to be used as NULL model

# Index