GSVA

October 25, 2011

`computeGeneSetsOverlap`
Compute gene-sets overlap

Description

Calculates the overlap among every pair of gene-sets given as input.

Usage

```
## S4 method for signature 'list,character'
computeGeneSetsOverlap(gSets, uniqGenes, min.sz=1, max.sz=Inf)
## S4 method for signature 'list,ExpressionSet'
computeGeneSetsOverlap(gSets, uniqGenes, min.sz=1, max.sz=Inf)
## S4 method for signature 'GeneSetCollection,character'
computeGeneSetsOverlap(gSets, uniqGenes, min.sz=1, max.sz=Inf)
## S4 method for signature 'GeneSetCollection,ExpressionSet'
computeGeneSetsOverlap(gSets, uniqGenes, min.sz=1, max.sz=Inf)
```

Arguments

`gSets`	Gene sets given either as a `list` or a `GeneSetCollection` object.
`uniqGenes`	Vector of unique genes to be considered when calculating the overlaps.
`min.sz`	Minimum size.
`max.sz`	Maximum size.

Details

This function calculates the overlap between every pair of gene sets of the input argument `gSets`. Before this calculation takes place, the gene sets in `gSets` are firstly filtered to discard genes that do not match to the identifiers in `uniqGenes`. Secondly, they are further filtered to meet the minimum and/or maximum size specified with the arguments `min.sz` and `max.sz`. The overlap between two gene sets is calculated as the number of common genes between the two gene sets divided by the smallest size of the two gene sets.

Value

A gene-set by gene-set matrix of the overlap among every pair of gene sets.

Author(s)

J. Guinney

References

H\"anzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene Set Variation Analysis, *submitted*

See Also

`filterGeneSets`

Examples

```
geneSets <- list(set1=as.character(1:4), set2=as.character(4:10))

computeGeneSetsOverlap(geneSets, unique(unlist(geneSets)))
```

`filterGeneSets` *Filter gene sets*

Description

Filters gene sets through a given minimum and maximum set size.

Usage

```
## S4 method for signature 'list'
filterGeneSets(gSets, min.sz=1, max.sz=Inf)
## S4 method for signature 'GeneSetCollection'
filterGeneSets(gSets, min.sz=1, max.sz=Inf)
```

Arguments

`gSets`	Gene sets given either as a `list` or a `GeneSetCollection` object.
`min.sz`	Minimum size.
`max.sz`	Maximum size.

Details

This function filters the input gene sets according to a given minimum and maximum set size.

Value

A collection of gene sets that meet the given minimum and maximum set size.

Author(s)

J. Guinney

References

H\"anzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene Set Variation Analysis, *submitted*

See Also

computeGeneSetsOverlap

Examples

```
geneSets <- list(set1=as.character(1:4), set2=as.character(4:10))

filterGeneSets(geneSets, min.sz=5)
```

---

gsva *Gene Set Variation Analysis*

Description

Estimates GSVA enrichment scores.

Usage

```
## S4 method for signature 'ExpressionSet,list'
gsva(expr, gset.idx.list,
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    no.bootstraps=0,
    bootstrap.percent = .632,
    parallel.sz=0,
    parallel.type="SOCK",
    verbose=TRUE,
    mx.diff=TRUE)
## S4 method for signature 'ExpressionSet,GeneSetCollection'
gsva(expr, gset.idx.list,
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    no.bootstraps=0,
    bootstrap.percent = .632,
    parallel.sz=0,
    parallel.type="SOCK",
    verbose=TRUE,
    mx.diff=TRUE)
## S4 method for signature 'matrix,list'
gsva(expr, gset.idx.list,
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
```

```
    no.bootstraps=0,
    bootstrap.percent = .632,
    parallel.sz=0,
    parallel.type="SOCK",
    verbose=TRUE,
    mx.diff=TRUE)
```

Arguments

`expr`	Gene expression data which can be given either as an `ExpressionSet` object or as a matrix of expression values where rows correspond to genes and columns correspond to samples.
`gset.idx.list`	Gene sets provided either as a `list` object or as a `GeneSetCollection` object.
`abs.ranking`	Flag to determine whether genes should be ranked according to their sign (flag=FALSE) or by absolute value (flag=TRUE). In the latter, pathways with genes enriched on either extreme (high or low) will be regarded as 'highly' activated.
`min.sz`	Minimum size of the resulting gene sets.
`max.sz`	Maximum size of the resulting gene sets.
`no.bootstraps`	Number of bootstrap iterations to perform.
`bootstrap.percent`	.632 is the ideal percent samples bootstrapped.
`parallel.sz`	Number of processors to use when doing the calculations in parallel. This requires to previously load either the `multicore` or the `snow` library. If `multicore` is loaded and this argument is left with its default value (`parallel.sz=0`) then it will use all available core processors unless we set this argument with a smaller number. If `snow` is loaded then we must set this argument to a positive integer number that specifies the number of processors to employ in the parallel calculation.
`parallel.type`	Type of cluster architecture when using `snow`.
`verbose`	Gives information about each calculation step. Default: `FALSE`.
`mx.diff`	Offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic. `mx.diff=FALSE`: ES is calculated as the maximum distance of the random walk from 0. `mx.diff=TRUE` (default): ES is calculated as the magnitude difference between the largest positive and negative random walk deviations.

Details

GSVA assesses the relative enrichment of gene sets across samples using a non-parametric approach. Conceptually, GSVA transforms a p-gene by n-sample gene expression matrix into a g-geneset by n-sample pathway enrichment matrix. This facilitates many forms of statistical analysis in the 'space' of pathways rather than genes, providing a higher level of interpretability.

The `gsva()` function first maps the identifiers in the gene sets to the identifiers in the input expression data leading to a filtered collection of gene sets. This collection can be further filtered to require a minimun and/or maximum size of the gene sets for which we want to calculate GSVA enrichment scores, by using the arguments `min.sz` and `max.sz`.

Value

A gene-set by sample matrix of GSVA enrichment scores.

Author(s)

J. Guinney

References

H\"anzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene Set Variation Analysis, *submitted*

See Also

`filterGeneSets` `computeGeneSetsOverlap`

Examples

```
library(limma)

p <- 10 ## number of genes
n <- 30 ## number of samples
nGrp1 <- 15 ## number of samples in group 1
nGrp2 <- n - nGrp1 ## number of samples in group 2

## consider three disjoint gene sets
geneSets <- list(set1=paste("g", 1:3, sep=""),
                 set2=paste("g", 4:6, sep=""),
                 set3=paste("g", 7:10, sep=""))

## sample data from a normal distribution with mean 0 and st.dev. 1
y <- matrix(rnorm(n*p), nrow=p, ncol=n,
            dimnames=list(paste("g", 1:p, sep="") , paste("s", 1:n, sep="")))

## genes in set1 are expressed at higher levels in the last 10 samples
y[geneSets$set1, (nGrp1+1):n] <- y[geneSets$set1, (nGrp1+1):n] + 2

## build design matrix
design <- cbind(sampleGroup1=1, sampleGroup2vs1=c(rep(0, nGrp1), rep(1, nGrp2)))

## fit linear model
fit <- lmFit(y, design)

## estimate moderated t-statistics
fit <- eBayes(fit)

## genes in set1 are differentially expressed
topTable(fit, coef="sampleGroup2vs1")

## estimate GSVA enrichment scores for the three sets
gsva_es <- gsva(y, geneSets, mx.diff=1)$es.obs

## fit the same linear model now to the GSVA enrichment scores
fit <- lmFit(gsva_es, design)

## estimate moderated t-statistics
```

```
fit <- eBayes(fit)

## set1 is differentially expressed
topTable(fit, coef="sampleGroup2vs1")
```

Index