Package 'GSVA'

October 10, 2021

Version 1.40.1

Title Gene Set Variation Analysis for microarray and RNA-seq data

Depends R (>= 3.5.0)

Imports methods, stats, utils, graphics, S4Vectors, IRanges, Biobase, SummarizedExperiment, GSEABase, Matrix, parallel, BiocParallel, SingleCellExperiment, sparseMatrixStats, DelayedArray, DelayedMatrixStats, HDF5Array, BiocSingular

Suggests BiocGenerics, RUnit, BiocStyle, knitr, rmarkdown, limma, RColorBrewer, org.Hs.eg.db, genefilter, edgeR, GSVAdata, shiny, shinydashboard, ggplot2, data.table, plotly, future, promises, shinybusy, shinyjs

Description Gene Set Variation Analysis (GSVA) is a non-parametric, unsupervised method for estimating variation of gene set enrichment through the samples of a expression data set. GSVA performs a change in coordinate systems, transforming the data from a gene by sample matrix to a gene-set by sample matrix, thereby allowing the evaluation of pathway enrichment for each sample. This new matrix of GSVA enrichment scores facilitates applying standard analytical methods like functional enrichment, survival analysis, clustering, CNV-pathway analysis or cross-tissue pathway analysis, in a pathway-centric manner.

License GPL (>= 2)

VignetteBuilder knitr

URL https://github.com/rcastelo/GSVA

BugReports https://github.com/rcastelo/GSVA/issues

Encoding latin1

biocViews FunctionalGenomics, Microarray, RNASeq, Pathways, GeneSetEnrichment

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

git_branch RELEASE_3_13

git_last_commit 61b842e

git_last_commit_date 2021-06-03

Date/Publication 2021-10-10

Author Justin Guinney [aut, cre],
Robert Castelo [aut],
Alexey Sergushichev [ctb],
Pablo Sebastian Rodriguez [ctb]

Maintainer Justin Guinney <justin.guinney@sagebase.org>

R topics documented:

`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

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013.

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

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013.

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 'SingleCellExperiment,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'HDF5Array,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'dgCMatrix,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'SummarizedExperiment,GeneSetCollection'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
```

```
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'SummarizedExperiment,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'ExpressionSet,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'ExpressionSet,GeneSetCollection'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'matrix,GeneSetCollection'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
```

```
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
## S4 method for signature 'matrix,list'
gsva(expr, gset.idx.list, annotation,
    method=c("gsva", "ssgsea", "zscore", "plage"),
    kcdf=c("Gaussian", "Poisson", "none"),
    abs.ranking=FALSE,
    min.sz=1,
    max.sz=Inf,
    parallel.sz=1L,
    mx.diff=TRUE,
    tau=switch(method, gsva=1, ssgsea=0.25, NA),
    ssgsea.norm=TRUE,
    verbose=TRUE,
    BPPARAM=SerialParam(progressbar=verbose))
```

Arguments

`expr`	Gene expression data which can be given either as a `SummarizedExperiment`, `SingleCellExperiment` `ExpressionSet` object, or as a matrix of expression values where rows correspond to genes and columns correspond to samples. This matrix can be also in a sparse format, as a `dgCMatrix`, or as an on-disk backend representation, such as `HDF5Array` .
`gset.idx.list`	Gene sets provided either as a `list` object or as a `GeneSetCollection` object.
`annotation`	In the case of calling `gsva()` on a `SummarizedExperiment` or `SingleCellExperiment` object, the `annotation` argument can be used to select the assay containing the molecular data we want as input to the `gsva()` function, otherwise the first assay is selected. In the case of calling `gsva()` with expression data in a `matrix` and gene sets as a `GeneSetCollection` object, the `annotation` argument can be used to supply the name of the Bioconductor package that contains annotations for the class of gene identifiers occurring in the row names of the expression data matrix. In the case of calling `gsva()` on a `ExpressionSet` object, the `annotation` argument is ignored. See details information below.
`method`	Method to employ in the estimation of gene-set enrichment scores per sample. By default this is set to `gsva` (Hänzelmann et al, 2013) and other options are `ssgsea` (Barbie et al, 2009), `zscore` (Lee et al, 2008) or `plage` (Tomfohr et al, 2005). The latter two standardize first expression profiles into z-scores over the samples and, in the case of `zscore`, it combines them together as their sum divided by the square-root of the size of the gene set, while in the case of `plage` they are used to calculate the singular value decomposition (SVD) over

the genes in the gene set and use the coefficients of the first right-singular vector as pathway activity profile.

`kcdf` Character string denoting the kernel to use during the non-parametric estimation of the cumulative distribution function of expression levels across samples when `method="gsva"`. By default, `kcdf="Gaussian"` which is suitable when input expression values are continuous, such as microarray fluorescent units in logarithmic scale, RNA-seq log-CPMs, log-RPKMs or log-TPMs. When input expression values are integer counts, such as those derived from RNA-seq experiments, then this argument should be set to `kcdf="Poisson"`.

`abs.ranking` Flag used only when `mx.diff=TRUE`. When `abs.ranking=FALSE` (default) a modified Kuiper statistic is used to calculate enrichment scores, taking the magnitude difference between the largest positive and negative random walk deviations. When `abs.ranking=TRUE` the original Kuiper statistic that sums the largest positive and negative random walk deviations, is used. In this latter case, gene sets 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.

`parallel.sz` Number of threads of execution to use when doing the calculations in parallel. The argument `BPPARAM` allows one to set the parallel back-end and fine tune its configuration.

`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.

`tau` Exponent defining the weight of the tail in the random walk performed by both the `gsva` (Hänzelmann et al., 2013) and the `ssgsea` (Barbie et al., 2009) methods. By default, this `tau=1` when `method="gsva"` and `tau=0.25` when `method="ssgsea"` just as specified by Barbie et al. (2009) where this parameter is called `alpha`.

`ssgsea.norm` Logical, set to `TRUE` (default) with `method="ssgsea"` runs the SSGSEA method from Barbie et al. (2009) normalizing the scores by the absolute difference between the minimum and the maximum, as described in their paper. When `ssgsea.norm=FALSE` this last normalization step is skipped.

`verbose` Gives information about each calculation step. Default: `FALSE`.

`BPPARAM` An object of class `BiocParallelParam` specifiying parameters related to the parallel execution of some of the tasks and calculations within this function.

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.

By default, gsva() will try to match the identifiers in expr to the identifiers in gset.idx.list just as they are, unless the annotation argument is set.

The gsva() function first maps the identifiers in the gene sets in gset.idx.list to the identifiers in the input expression data expr. When the input gene sets in gset.idx.list is provided as a list object, gsva() will try to match the identifiers in expr directly to the identifiers in gset.idx.list just as they are. Because unmatching identifiers will be discarded in both, expr and gset.idx.list, gsva() may prompt an error if no identifiers can be matched as in the case of different types of identifiers (e.g., gene symbols vs Entrez identitifers).

However, then the input gene sets in gset.idx.list is provided as a GeneSetCollection object, gsva() will try to automatically convert those identifiers to the type of identifier in the input expression data expr. Such an automatic conversion, however, will only occur in three scenarios: 1. when expr is an ExpressionSet object with an appropriately set annotation slot; 2. when expr is a SummarizedExperiment or a SingleCellExperiment object with an appropriately set annotation slot in the metadata of expr; 3. when expr is a matrix or a dgCMatrix and the annotation argument of the gsva() function is set to the name of the annotation package that provides the relationships between the type of identifiers in expr and gset.idx.list.

The collection of gene sets resulting from the previous identifier matching, can be further filtered to require a minimun and/or maximum size by using the arguments min.sz and max.sz.

If you use GSVA in your research, please cite also the corresponding method as described in the method parameter.

Value

A gene-set by sample matrix (of matrix or dgCMatrix type, depending on the input) of GSVA enrichment scores.

Author(s)

J. Guinney and R. Castelo

References

Barbie, D.A. et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(5):108-112, 2009.

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013.

Lee, E. et al. Inferring pathway activity toward precise disease classification. *PLoS Comp Biol*, 4(11):e1000217, 2008.

Tomfohr, J. et al. Pathway level analysis of gene expression using singular value decomposition. *BMC Bioinformatics*, 6:225, 2005.

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 'nGrp1+1' to 'n' 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)

## 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")
```

---

igsva *Gene Set Variation Analysis*

Description

Starts an interactive GSVA shiny web app.

Usage

```
igsva()
```

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 `igsva()` function starts an interactive shiny web app that allows the user to configure the arguments of the `gsva()` function and runs it on the computer. Please see the manual page of the `gsva()` function for a description of the arguments and their default and alternative values.

The input data may be loaded from the users workspace or by selecting a CSV file for the expression data, and a GMT file for the gene sets data.

Value

A gene-set by sample matrix of GSVA enrichment scores after pressing the button 'Save & Close'. This result can be also downloaded as a CSV file with the 'Download' button.

Author(s)

J. Fernández and R. Castelo

References

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013.

See Also

`gsva`

Examples

```
## Not run:
res <- igsva() ## this will open your browser with the GSVA shiny web app

## End(Not run)
```

Index