

Package: GSVA (via r-universe)

October 22, 2026

Version 2.7.25

Title Gene Set Variation Analysis for Microarray and RNA-Seq Data

Depends R (>= 4.0.0)

Imports methods, stats, utils, graphics, tools, BiocGenerics,
MatrixGenerics, S4Vectors, S4Arrays, HDF5Array, rhdf5,
SparseArray, DelayedArray, IRanges, Biobase,
SummarizedExperiment, GSEABase, Matrix (>= 1.5-0),
DelayedMatrixStats, BiocParallel, SingleCellExperiment,
SpatialExperiment, sparseMatrixStats, cli, memuse

Suggests RUnit, BiocStyle, knitr, rmarkdown, limma, RColorBrewer,
arrow, duckdb, DBI, org.Hs.eg.db, genefilter, edgeR, GSVAdata,
sva, ExperimentHub, TENxPBMCDData, TENxVisiumData, spatialLIBD,
scraper, bluster, igraph, ggspavis, patchwork, ggplot2, shiny,
shinydashboard, data.table, plotly, future, promises,
shinybusy, shinyjs, batchtools, foreach, doParallel

LinkingTo cli

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 Artistic-2.0

VignetteBuilder knitr

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

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

Encoding UTF-8

biocViews FunctionalGenomics, Microarray, RNASeq, Pathways,
GeneSetEnrichment

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libmagick++-dev gsfonts libicu-dev libpng-dev libxml2-dev libssl-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-10-22 17:55:18 UTC

RemoteUrl <https://github.com/bioc/GSVA>

RemoteRef HEAD

RemoteSha 9db994d166da271490ac5214b9935625c51e18e5

Contents

avgParam-class	3
computeGeneSetsOverlap	6
deduplicateGeneSets	7
filterGeneSets	8
geneIdsToGeneSetCollection	9
geneSets	10
gsva,avgParam-method	12
gsvaAnnotation	15
gsvaEnrichment	17
GsvaExprData-class	19
GsvaGeneSets-class	20
gsvaMap	21
GsvaMethodParam-class	26
gsvaParam-class	28
gsvaRowNorm	34
guessGeneIdType	38
igsva	39
plageParam-class	40
readGMT	42
saveHDF5GSVA	44
spatCor	48
ssgseaParam-class	49
zscoreParam-class	53

Index	56
--------------	-----------

avgParam-class	avgParam class
----------------	----------------

Description

S4 class for average scores method parameter objects.

Objects of class avgParam contain the parameters for running the average method.

Usage

```
avgParam(
  exprData,
  geneSets,
  assay = NA_character_,
  annotation = NULL,
  minSize = 1,
  maxSize = Inf,
  method = "mean",
  checkNA = c("auto", "yes", "no"),
  use = c("everything", "all.obs", "na.rm"),
  ondisk = c("auto", "yes", "no"),
  verbose = TRUE
)

## S4 method for signature 'avgParam'
anyNA(x, recursive = FALSE)
```

Arguments

exprData	The expression data set. Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
geneSets	The gene sets. Must be one of the classes supported by GsvaGeneSets . For a list of these classes, see its help page using <code>help(GsvaGeneSets)</code> .
assay	Character vector of length 1. The name of the assay to use in case exprData is a multi-assay container, otherwise ignored. By default, an assay called 'log-counts' will be used if present, otherwise the first assay is used.
annotation	<p>An object of class <code>GeneIdentifierType</code> from package <code>GSEABase</code> describing the gene identifiers used as the row names of the expression data set. See <code>GeneIdentifierType</code> for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets.</p> <p>If the default value <code>NULL</code> is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as <code>exprData</code> (by calling gsvaAnnotation on it). If still not successful, the <code>NullIdentifier()</code> will be used as the gene identifier type, gene identifier mapping will be disabled and</p>

	gene identifiers used in expression data set and gene sets can only be matched directly.
minSize	Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.
maxSize	Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is Inf.
method	character vector of length 1. Specific average method employed in the calculation of the average scores. By now, the default and only available value is method="mean", which calculates the average scores as the arithmetic mean of the expression values of the genes in each gene set at each column sample or cell.
checkNA	Character vector of length 1 specifying whether the input expression data should be checked for the presence of missing values (NA or NaN). This must be one of the strings "auto" (default), "yes", or "no". The default value "auto" means that the software will perform that check only when the input expression data is provided as a base matrix, an ExpressionSet or a SummarizedExperiment object, while every other type of input expression data container (e.g., SingleCellExperiment, etc.) will not be checked. If checkNA="yes", then the input expression data will be checked for missing values irrespective of the object class of the data container, and if checkNA="no", then that check will not be performed.
use	Character vector of length 1 specifying a policy for dealing with missing values (NA or NaN) in the input expression data argument exprData. It only applies when either checkNA="yes", or checkNA="auto" (see the checkNA parameter. The argument value must be one of the strings "everything" (default), "all.obs", or "na.rm". The policy of the default value "everything" consists of propagating missing values so that the resulting enrichment score will be NA, whenever one or more of its contributing values is missing, giving a warning when that happens. When use="all.obs", the presence of NAs in the input expression data will produce an error. Finally, when use="na.rm", missing values in the input expression data will be removed from calculations, giving a warning when that happens, and giving an error if no values are left after removing the missing values.
ondisk	Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value ondisk="auto" will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces de memory footprint. When ondisk="yes" it will attempt to work with an on-disk data structure, while when ondisk="no" it will attempt to load all the data in main memory.
verbose	Logical vector of length 1. It gives information about some decisions made by the software during parameter object construction when verbose=TRUE (default) and remains silent otherwise.
x	An object of class avgParam .
recursive	Not used with x being an object of class avgParam .

Details

Since the average scores method does not take any method-specific parameters, this class does not add any slots to the common slots inherited from GsvaMethodParam. In addition to the common parameter slots inherited from [GsvaMethodParam], this class has slots for a specifying the average method employed and as well as four more slots for implementing a missing value policy.

The average method takes a number of parameters shared with all methods implemented by package GSVA but does not take any method-specific parameters.

Value

A new [avgParam](#) object.

Slots

method character vector of length 1. Specific average method employed in the calculation of the average scores. By now, the default and only available value is `method="mean"`, which calculates the average scores as the arithmetic mean of the expression values of the genes in each gene set at each column sample or cell.

checkNA Character vector of length 1. One of the strings "auto" (default), "yes", or "no", which refer to whether the input expression data should be checked for the presence of missing (NA) values.

didCheckNA Logical vector of length 1, indicating whether the input expression data was checked for the presence of missing (NA) values.

anyNA Logical vector of length 1, indicating whether the input expression data contains missing (NA) values.

use Character vector of length 1. One of the strings "everything" (default), "all.obs", or "na.rm", which refer to three different policies to apply in the presence of missing values in the input expression data; see [ssgseaParam](#).

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [GsvaMethodParam](#), [zscoreParam](#), [plageParam](#), [ssgseaParam](#), [gsvaParam](#)
[GeneIdentifierType](#)

Examples

```
suppressPackageStartupMessages({
  library(GSEABase)
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)

## for simplicity, use only a subset of the sample data
se <- geneExpCostaEtAl2021[1:1000, ]
gsc <- c2BroadSets[1:100]
```

```
avgp1 <- avgParam(se, gsc)
avgp1
```

```
computeGeneSetsOverlap
```

Compute gene-sets overlap

Description

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

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 `minSize` and `maxSize`. 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.

Usage

```
## S4 method for signature 'list,character'
computeGeneSetsOverlap(gSets, uniqGenes, minSize = 1, maxSize = Inf)

## S4 method for signature 'GeneSetCollection,character'
computeGeneSetsOverlap(gSets, uniqGenes, minSize = 1, maxSize = Inf)
```

Arguments

<code>gSets</code>	Gene sets given either as a list or a <code>GeneSetCollection</code> object.
<code>uniqGenes</code>	Vector of unique genes to be considered when calculating the overlaps.
<code>minSize</code>	Minimum size.
<code>maxSize</code>	Maximum size.

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)))
```

deduplicateGeneSets *Handling of Duplicated Gene Set Names*

Description

Offers a choice of ways for handling duplicated gene set names that may not be suitable as input to other gene set analysis functions.

Usage

```
deduplicateGeneSets(
  geneSets,
  deduplUse = c("first", "drop", "union", "smallest", "largest")
)
```

Arguments

- | | |
|-----------|--|
| geneSets | A named list of gene sets represented as character vectors of gene IDs as e.g. returned by readGMT . |
| deduplUse | <p>A character vector of length 1 specifying one of several methods to handle duplicated gene set names. Duplicated gene set names are explicitly forbidden by the GMT file format specification but can nevertheless be encountered in the wild. The available choices are:</p> <ul style="list-style-type: none"> • first (the default): drops all gene sets whose names are duplicated according to the base R function and retains only the first occurrence of a gene set name. • drop: removes <i>all</i> gene sets that have a duplicated name, including its first occurrence. • union: replaces gene sets with duplicated names by a single gene set containing the union of all their gene IDs. • smallest: drops gene sets with duplicated names and retains only the smallest of them, i.e. the one with the fewest gene IDs. If there are several smallest gene sets, the first will be selected. • largest: drops gene sets with duplicated names and retains only the largest of them, i.e. the one with the most gene IDs. If there are several largest gene sets, the first will be selected. |

Value

A named list of gene sets represented as character vectors of gene IDs.

Examples

```
library(GSVA)

gsets <- list(gs1=LETTERS[1:3], gs2=LETTERS[4:6], gs2=LETTERS[5:8])
gsets

deduplicateGeneSets(gsets)
deduplicateGeneSets(gsets, deduplUse="drop")
deduplicateGeneSets(gsets, deduplUse="union")
deduplicateGeneSets(gsets, deduplUse="smallest")
deduplicateGeneSets(gsets, deduplUse="largest")

fname <- system.file("extdata", "c2.subsetdups.v7.5.symbols.gmt.gz",
                      package="GSVAdata")

## readGMT() calls internally deduplicateGeneSets() and it takes the
## parameter 'deduplUse' which is passed to the internal call
c2.dupgenesets <- readGMT(fname, deduplUse="union")
c2.dupgenesets
any(duplicated(names(c2.dupgenesets)))
```

filterGeneSets	<i>Filter gene sets</i>
----------------	-------------------------

Description

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

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

Usage

```
## S4 method for signature 'list'
filterGeneSets(gSets, minSize = 1, maxSize = Inf)

## S4 method for signature 'GeneSetCollection'
filterGeneSets(gSets, minSize = 1, maxSize = Inf)
```

Arguments

gSets	Gene sets given either as a list or a GeneSetCollection object.
minSize	Minimum size.
maxSize	Maximum 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, minSize=5)
```

geneIdsToGeneSetCollection

Construct a GeneSetCollection object from a list of character vectors

Description

This function is essentially the reverse of `GSEABase::geneIds()`, i.e., it takes as input a named list of character vectors representing gene sets and returns the corresponding `GeneSetCollection` object.

Usage

```
geneIdsToGeneSetCollection(
  geneIdsList,
  geneIdType = "auto",
  collectionType = NullCollection()
)
```

Arguments

geneIdsList	A named list of character vectors like the ones returned by <code>geneIds()</code> . Names must be unique; otherwise see <code>deduplicateGeneSets()</code> for a number of strategies to resolve this issue.
-------------	---

- geneIdType

By default a character vector of length 1 with the special value "auto" or an object of a subclass of GeneIdentifierType. If set to "auto", the function will try to derive the gene ID type from argument geneIdsList using guessGeneIdType. Other values, including NULL, will be ignored with a warning and geneIdType=NULLIdentifier() will be used instead. The gene ID type of all GeneSet objects in the resulting GeneSetCollection will be set to this value.
- collectionType

An object of class CollectionType. The collection type of all GeneSet objects in the resulting GeneSetCollection will be set to this value but can afterwards be modified for individual GeneSets if necessary.

Value

An object of class GeneSetCollection with all its GeneSet objects using the gene ID and collection types specified by the corresponding arguments. Applying function geneIds() to this object should return a list identical to the geneIdsList argument.

See Also

[GeneSetCollection](#), [GeneIdentifierType](#), [geneIds](#), [deduplicateGeneSets](#), [guessGeneIdType](#), [GeneSet](#)

Examples

```
library(GSVA)

gsets <- list(INNATE_RESPONSE=c("AIM2", "ALPK1", "AP3B1"),
              ADAPTIVE_RESPONSE=c("CD27", "CD70", "EBAG9"))

gsets
geneIdsToGeneSetCollection(gsets)
```

geneSets	<i>Retrieve or Determine Gene Sets</i>
----------	--

Description

Retrieves or determines the gene sets that have been used or would be used in a gsva() gene set analysis. These are not necessarily the same as the input gene sets. See Details.

Usage

```
## S4 method for signature 'GsvaMethodParam'
geneSets(obj)

## S4 method for signature 'SummarizedExperiment'
geneSets(obj)
```

```
## S4 method for signature 'SingleCellExperiment'  
geneSets(obj)  
  
## S4 method for signature 'SpatialExperiment'  
geneSets(obj)  
  
## S4 method for signature 'GsvaExprData'  
geneSets(obj)  
  
## S4 method for signature 'GsvaMethodParam'  
geneSetSizes(obj)  
  
## S4 method for signature 'GsvaExprData'  
geneSetSizes(obj)
```

Arguments

obj An object of one of the following classes:

- An expression data object of one of the classes described in [GsvaExprData](#) that is the return value of a call to `gsva()`.
- A parameter object of one of the classes described in [GsvaMethodParam](#) that could be used in a call to `gsva()`.

Details

The gene sets used in a `gsva()` gene set analysis, or just their sizes, may be a valuable input to subsequent analyses. However, they are not necessarily the same as the original input gene sets, or their sizes: based on user choices, the gene annotation used, or presence/absence of genes in gene sets and expression data set, `gsva()` may have to modify them during the preparation of an analysis run. In order to make use of these gene sets or their sizes, you can either

- retrieve them from the object returned by `gsva()` by passing this object to `geneSets()` or `geneSetSizes()`, or
- predict them by calling `geneSets()` or `geneSetSizes()` on the parameter object that would also be passed to `gsva()`. This is much slower and should only be done if you do not intend to run an actual gene set analysis.

`geneSetSizes()` is a convenience wrapper running `lengths()` on the list of gene sets returned by `geneSets()`.

Value

The `geneSets()` methods return a named list of character vectors where each character vector contains the gene IDs of a gene set. The `geneSetSizes()` methods return a named integer vector of gene set sizes.

Examples

```
library(GSVA)
```

```

p <- 10 ## number of genes
n <- 30 ## number of samples

gsets <- list(set1=paste0("g", 1:3),
              set2=paste0("g", 4:6),
              set3=paste0("g", 7:10),
              set4=paste0("g", 10:13)) ## genes not in the expression data
gsets

y <- matrix(rnorm(n*p), nrow=p, ncol=n,
            dimnames=list(paste("g", seq_len(p), sep=""),
                          paste("s", seq_len(n), sep="")))

gsvapar <- gsvaParam(y, gsets)
geneSets(gsvapar)

es <- gsva(gsvapar)

geneSets(es)

```

gsva,avgParam-method *Gene Set Variation Analysis*

Description

Estimates GSVA enrichment scores.

Usage

```

## S4 method for signature 'avgParam'
gsva(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

## S4 method for signature 'gsvaParam'
gsva(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

## S4 method for signature 'plageParam'
gsva(

```

```

    param,
    verbose = TRUE,
    BPPARAM = SerialParam(progressbar = verbose),
    maxmem = "auto"
)

## S4 method for signature 'ssgseaParam'
gsva(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

## S4 method for signature 'zscoreParam'
gsva(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

```

Arguments

param	<p>A parameter object of one of the following classes:</p> <ul style="list-style-type: none"> • A gsvaParam object built using the constructor function gsvaParam. This object will trigger <code>gsva()</code> to use the GSVA algorithm by Hänzelmann et al. (2013). • A plageParam object built using the constructor function plageParam. This object will trigger <code>gsva()</code> to use the PLAGE algorithm by Tomfohr et al. (2005). • A zscoreParam object built using the constructor function zscoreParam. This object will trigger <code>gsva()</code> to use the combined z-score algorithm by Lee et al. (2008). • A ssgseaParam object built using the constructor function ssgseaParam. This object will trigger <code>gsva()</code> to use the ssGSEA algorithm by Barbie et al. (2009).
verbose	Gives information about each calculation step. Default: TRUE.
BPPARAM	An object of class <code>BiocParallelParam</code> specifying parameters related to the parallel execution of some of the tasks and calculations within this function.
maxmem	A vector of length 1 either specifying a number in bytes, or a character string with either the word <code>auto</code> (default), or a number followed by a suffix indicating kilobytes (K), megabytes (M), gigabytes (G) or terabytes (T), which GSVA will use to attempt bounding the maximum amount of main memory used across all threads of execution to that given quantity. By default <code>maxmem="auto"</code> , indicating that the maximum memory will be the 90% of the total main memory, as calculated by Sys.meminfo() or, when it is smaller, of the memory limit of

the job or container where R runs, such as a job of the SLURM workload manager. To avoid setting any bound on the maximum memory, use `maxmem=Inf`. Note that the amount of main memory used in an R session or script may depend on other commands and packages used in that same session or script. In particular, each R process running GSVA, the main one and each of its parallel workers in `BPPARAM`, takes about 0.8 GB by itself for loading GSVA and the packages it depends on, including forked workers, such as those of a `MulticoreParam` object, which end up taking most of that memory as their own while they run. The memory that GSVA estimates includes the expression data given to it, with all the assays of a `SummarizedExperiment` object held in main memory, but not other objects of the R session, such as other large matrices or data sets, which reduce the memory actually available to GSVA; when such objects are present, consider setting `maxmem` to the memory that they leave. Before starting its calculations, GSVA gives a warning when the memory it estimates they require exceeds `maxmem`, suggesting to use fewer parallel workers or a larger `maxmem`. The memory taken by each R process can be set with the option `GSVA.workermem`, in bytes, and this check can be disabled with `options(GSVA.check_memory=FALSE)`.

Value

A gene-set by sample matrix of GSVA enrichment scores stored in a container object of the same type as the input expression data container, except for the fact that enrichment scores are always dense, irrespective of whether the input is sparse, such as in single-cell data. If the input was a base matrix, a `dgCMatrix`, a `SVT_SparseMatrix`, or a `DelayedMatrix` object, then the output will be either a base matrix object or a `DelayedMatrix`, with the gene sets employed in the calculations stored in an attribute called `geneSets` of that object. If the input was an `ExpressionSet` object, then the output will be also an `ExpressionSet` object with the gene sets employed in the calculations stored in an attribute called `geneSets`. If the input was an object of either class `SummarizedExperiment`, `SingleCellExperiment`, or `SpatialExperiment`, then the output will be of the same class, where enrichment scores will be stored in an assay called `es` and the gene sets employed in the calculations will be stored in the `rowData` slot of the object under the column name `gs`.

References

- Barbie, D.A. et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(5):108-112, 2009. doi:[10.1038/nature08460](https://doi.org/10.1038/nature08460)
- Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013. doi:[10.1186/14712105147](https://doi.org/10.1186/14712105147)
- Lee, E. et al. Inferring pathway activity toward precise disease classification. *PLoS Comp Biol*, 4(11):e1000217, 2008. doi:[10.1371/journal.pcbi.1000217](https://doi.org/10.1371/journal.pcbi.1000217)
- Tomfohr, J. et al. Pathway level analysis of gene expression using singular value decomposition. *BMC Bioinformatics*, 6:225, 2005. doi:[10.1186/147121056225](https://doi.org/10.1186/147121056225)

See Also

`gsvaParam`, `plageParam`, `zscoreParam`, `ssgseaParam`, `BiocParallelParam`, `gsvaRowNorm`, `gsvaColRanks`, `gsvaColScores`

Examples

```

library(GSVA)
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", seq_len(p), sep=""),
                          paste("s", seq_len(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")

## build GSVA parameter object
gsvapar <- gsvaParam(y, geneSets)

## estimate GSVA enrichment scores for the three sets
gsva_es <- gsva(gsvapar)

## 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")

```

Description

Methods for storing and retrieving annotation metadata in expression data objects that support it. If gene sets and expression data are using different but known gene identifier types and an appropriate annotation database is available, gene set identifiers can be mapped to expression data identifiers without manual user intervention, e.g. from an MSigDb gene set using ENTREZ IDs or gene symbols to an expression data set using ENSEMBL IDs.

Usage

```
## S4 method for signature 'GsvaExprData'
gsvaAnnotation(object)

## S4 replacement method for signature 'GsvaExprData, GeneIdentifierType'
gsvaAnnotation(object) <- value

## S4 method for signature 'ExpressionSet'
gsvaAnnotation(object)

## S4 replacement method for signature 'ExpressionSet, character'
gsvaAnnotation(object) <- value

## S4 replacement method for signature 'ExpressionSet, GeneIdentifierType'
gsvaAnnotation(object) <- value

## S4 method for signature 'SummarizedExperiment'
gsvaAnnotation(object)

## S4 replacement method for signature 'SummarizedExperiment, GeneIdentifierType'
gsvaAnnotation(object) <- value

## S4 method for signature 'SingleCellExperiment'
gsvaAnnotation(object)

## S4 replacement method for signature 'SingleCellExperiment, GeneIdentifierType'
gsvaAnnotation(object) <- value

## S4 method for signature 'SpatialExperiment'
gsvaAnnotation(object)

## S4 replacement method for signature 'SpatialExperiment, GeneIdentifierType'
gsvaAnnotation(object) <- value

## S4 method for signature 'list'
gsvaAnnotation(object)

## S4 replacement method for signature 'list, GeneIdentifierType'
gsvaAnnotation(object) <- value
```

```
## S4 method for signature 'GeneSetCollection'  
gsvaAnnotation(object)
```

Arguments

object	An expression data object of one of the classes described in GsvaExprData . Simple matrix and dgCMatix objects are not capable of storing annotation metadata and will return NULL.
value	For the replacement methods, the annotation metadata to be stored in the object. For ExpressionSet objects, this must be a character of length 1 specifying the name of the annotation database to be used. For SummarizedExperiment and its subclasses, this must be a GeneIdentifierType created by one of the constructors from package GSEABase where the annotation argument is typically the name of an organism or annotation database, e.g. org.Hs.eg.db. Simple matrix and dgCMatix objects are not capable of storing annotation metadata and the attempt to do so will result in an error.

Value

For the retrieval methods, the annotation metadata stored in the object or NULL. For the replacement methods, the updated object.

See Also

[ExpressionSet](#), [SummarizedExperiment](#), [GeneIdentifierType](#), [dgCMatix](#)

Examples

```
library(GSEABase)  
library(GSVA)  
library(GSVAdata)  
  
data(geneprotExpCostaEtAl2021)  
se <- geneExpCostaEtAl2021  
se  
  
gsvaAnnotation(se)  
gsvaAnnotation(se) <- EntrezIdentifier("org.Hs.eg.db")  
gsvaAnnotation(se)
```

gsvaEnrichment

GSVA enrichment data and visualization

Description

Extract and plot enrichment data from GSVA scores.

Usage

```
gsvaEnrichment(
  rankExprData,
  column = 1,
  geneSet = 1,
  plot = c("auto", "base", "ggplot", "no"),
  ...
)
```

Arguments

rankExprData	A column-rank expression data set obtained with gsvaColRanks . Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
column	The column for which we want to retrieve the enrichment data. This parameter is only available in the <code>gsvaEnrichment()</code> method.
geneSet	Either a single positive integer number between 1 and the number of available gene sets in parameter object stored in <code>rankExprData</code> , or a single character string with the name of one of the gene sets available in that object, or a vector of integers or character strings with the index values or names of rows in <code>rankExprData</code> that should be considered as the gene set for which the enrichment data should be retrieved.
plot	A character string indicating whether an enrichment plot should be produced using either base R graphics (<code>plot="base"</code>) or the <code>ggplot2</code> package (<code>plot="ggplot"</code>), or not (<code>plot="no"</code>). In the latter case, the enrichment data will be returned. By default <code>plot="auto"</code> , which implies that if this method is called from an interactive session, a plot using base R graphics will be produced and, otherwise, the enrichment data is returned.
...	Further arguments passed to the <code>plot()</code> function when the previous parameter <code>plot="base"</code> .

Value

When `plot="no"`, this method returns the enrichment data. When `plot="ggplot"`, this method returns a `ggplot` object. When `plot="base"` no value is returned.

References

Hänzelmann, S., Castelo, R. and Guinney, J. GSEA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013. doi:[10.1186/14712105147](https://doi.org/10.1186/14712105147)

See Also

[gsvaColRanks](#), [GsvaExprData](#)

Examples

```

library(GSVA)

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(gset1=paste0("g", 1:3),
                 gset2=paste0("g", 4:6),
                 gset3=paste0("g", 7:10))

## 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", seq_len(p), sep=""),
                          paste("s", seq_len(n), sep="")))

## build GSVA parameter object
gsvapar <- gsvaParam(y, geneSets)

## calculate GSVA ranks
gsvarownorm <- gsvaRowNorm(gsvapar)
gsvaranks <- gsvaColRanks(gsvarownorm)

## by default the enrichment data for the first column and the first
## gene set in the input parameter object, are retrieved
gsvaEnrichment(gsvaranks)

## we can calculate the enrichment data for any of the gene sets given
## in the input parameter object
gsvaEnrichment(gsvaranks, geneSet="gset2")

## we can calculate the enrichment data for a new gene set that did not
## form part of the input parameter object
gsvaEnrichment(gsvaranks, geneSet=c("g1", "g4", "g7"))

```

GsvaExprData-class	GsvaExprData <i>class</i>
--------------------	---------------------------

Description

Virtual superclass of expression data classes supported by GSVA.

Details

GSVA supports expression data matrices in a growing number of containers and representations. This class union allows to store any of these in a slot of another class as well as defining common methods for all of them. The current list of supported classes is:

- `matrix`
- `dgCMatrx` from package `Matrix`
- `SVT_SparseMatrix` from package `SparseArray`
- `DelayedMatrix` from package `DelayedArray`
- `HDF5Matrix` from package `HDF5Array`
- `ExpressionSet` from package `Biobase`
- `SummarizedExperiment` from package `SummarizedExperiment`
- `SingleCellExperiment` from package `SingleCellExperiment`
- `SpatialExperiment` from package `SpatialExperiment`

See Also

[matrix](#), [dgCMatrx](#), [SVT_SparseMatrix](#), [DelayedMatrix](#), [HDF5Array](#), [ExpressionSet](#), [SummarizedExperiment](#), [SingleCellExperiment](#), [SpatialExperiment](#)

GsvaGeneSets-class

GsvaGeneSets *class*

Description

Virtual superclass of gene set classes supported by GSVA.

Details

GSVA supports gene sets consisting of gene identifiers as either a named list of character vectors or an object of class [GeneSetCollection](#). Alternatively, gene sets may be specified as a named list of integer vectors in the range of 1:nrow(X) that are indices to the rows of the corresponding expression data matrix X. This class union allows to store any of these in a slot of another class as well as defining common methods for them.

See Also

[list](#), [GeneSetCollection](#)

Examples

```
library(GSVA)

gsetslst <- list(INNATE_RESPONSE=c("AIM2", "ALPK1", "AP3B1"),
                ADAPTIVE_RESPONSE=c("CD27", "CD70", "EBAG9"))

gsetslst
gsetsgsc <- geneIdsToGeneSetCollection(gsetslst)
gsetsgsc

class(gsetslst)
class(gsetsgsc)
```

```
is(gsetslst, "GsvaGeneSets")
is(gsetsgsc, "GsvaGeneSets")
```

gsvaMap

MapReduce parallelization for HPC environments

Description

The functions `gsvaMap()` and `gsvaReduce()` allow one to run GSVA calculations across multiple compute nodes in a high-performance computing (HPC) environment. The `gsvaMap()` function is used to launch the GSVA calculations on each compute node, while the `gsvaReduce()` function is used to combine the results from all nodes into a single object.

The specific HPC backend used for parallelization is determined by the `BTPARAM` argument to the `gsvaMap()` function, which should be an object of class `BatchtoolsParam`. The function `gsvaBatchtoolsSlurmParam()` is provided to create a `BatchtoolsParam` object with some sensible defaults for running GSVA calculations on a Slurm cluster. The calculations across independent compute nodes without shared memory are enabled by using on-disk data structures that store enrichment scores and intermediate results in HDF5 files located in a filesystem path specified in the `dir` argument of `gsvaBatchtoolsSlurmParam()`, which defaults to a directory named "GSVAOUTPUT" in the current working directory from where the R session calling `gsvaMap()` was launched. The user must ensure that this path is reachable by all compute nodes in the HPC environment, and must manually delete its contents after the GSVA calculations are finished. Each job saves its results under a temporary name ending in `.partial`, which it renames once they are complete, so files with that ending left in that path belong to jobs that did not finish, and can be deleted once no job of those calculations is running.

Usage

```
gsvaMap(
  FUN,
  inputData,
  output = c("object", "HDF5", "Parquet"),
  verbose = TRUE,
  BTPARAM = BatchtoolsParam(workers = 2, progressbar = verbose),
  MAPREDO = NULL
)

gsvaReduce(mapOutput, verbose = TRUE)

gsvaBatchtoolsSlurmParam(
  dir = "GSVAOUTPUT",
  partition,
  walltime = NULL,
  nodes = 2,
  ncpus_per_task = 2,
```

```

mem = "10G",
arrayjobs = FALSE,
template = system.file("templates", "slurm-gsva.tmpl", package = "GSVA")
)

```

Arguments

FUN	In <code>gsvaMap()</code> , function to map to the data in the <code>inputData</code> argument.
inputData	In <code>gsvaMap()</code> , input data for performing the calculations in parallel. It should be an object either of class <code>gsvaParam</code> , or one of the classes supported by <code>GsvaExprData</code> as output from <code>gsvaRowNorm</code> or <code>gsvaColRanks</code> ; for a list of these classes consult class <code>?GsvaExprData</code> . It can also be a list object with the output of <code>gsvaMap()</code> itself, to proceed through the three-step pipeline of calculating row-normalized expression values, column ranks, and GSVA scores without having to call to <code>gsvaReduce()</code> in between.
output	In <code>gsvaMap()</code> , a character string specifying what each worker returns: either the resulting object ("object"), or the path to that object after saving it in the working directory of the registry of BTPARAM with <code>saveHDF5GSVA</code> ("HDF5") or with <code>saveParquetGSVA</code> ("Parquet"). In the latter two cases, the output of <code>gsvaMap()</code> is a list of paths instead of a list of objects. Note that, by default, the working directory of the registry is the current working directory. Saving in Apache Parquet format requires the package <code>arrow</code> . When this argument is not given, its default value is "HDF5" if BTPARAM sends the jobs to a workload manager, such as Slurm, and "object" otherwise, i.e., when the <code>cluster</code> field of BTPARAM is "socket", "multicore" or "interactive". When MAPREDO is given, its default value is the one used to produce MAPREDO. The names of the saved files depend only on FUN, inputData, output and the chunk boundaries, and <code>gsvaMap()</code> also writes in that directory a manifest file, ending in <code>_manifest.rds</code> , which allows the MAPREDO argument to find them. For this reason, <code>gsvaMap()</code> gives an error when that directory already has results of a previous call with the same FUN, inputData and output arguments, unless MAPREDO is given.
verbose	Gives information about the progress of the calculations. Default: TRUE.
BTPARAM	In <code>gsvaMap()</code> , an object of class <code>BatchtoolsParam</code> specifying parameters for parallel execution in an HPC environment. By default, it is set to a <code>BatchtoolsParam</code> object with 2 workers and a progress bar enabled, and this will start a multicore execution using CPU cores in the compute node where <code>gsvaMap()</code> has been called, i.e., by default it will not deploy an HPC environment. For that purpose, users should either create a <code>BatchtoolsParam</code> object themselves with appropriate arguments or, if an SLURM HPC environment is available, they may use the wrapper function <code>gsvaBatchtoolsSlurmParam()</code> , which is provided to create a <code>BatchtoolsParam</code> object with some sensible defaults for running GSVA calculations on a SLURM cluster. GSVA uses the memory element of the resources of BTPARAM as the maximum main memory of each job, shared by its R process and its parallel workers, while the template of the job scripts determines how much memory the workload manager allocates to the job; e.g., the default template of batchtools for Slurm allocates that amount for each CPU core, see the template argument of <code>gsvaBatchtoolsSlurmParam()</code> .

MAPREDO	In <code>gsvaMap()</code> , the output of a previous call to <code>gsvaMap()</code> with the same <code>FUN</code> and <code>inputData</code> arguments, in which some of the chunks failed. When this argument is given, only the failed chunks are computed again, using the same chunk boundaries as in the previous call, and the result combines them with the chunks that did not fail. This makes it possible to resubmit the failed chunks with a different <code>BTPARAM</code> argument, for instance with more memory or a longer wall time. When the previous call saved its results to disk (see the <code>output</code> argument), this argument can also be the path to the directory where they were saved, or to the manifest file that <code>gsvaMap()</code> writes there, and the chunks whose results are not found in that directory are computed again and saved in it. This allows one to resubmit chunks whose jobs were killed by the workload manager, or that were running when the R session calling <code>gsvaMap()</code> ended. If that directory has results of several calls matching <code>FUN</code> and <code>inputData</code> , saved in different formats, the <code>output</code> argument selects one of them. Note that <code>inputData</code> is matched by its dimensions, dimension names and GSVA parameters, but not by its values. Default: <code>NULL</code> , which computes all chunks.
mapOutput	In <code>gsvaReduce()</code> , the output of <code>gsvaMap()</code> , which can be a list of objects or a list of paths to GSVA output saved with <code>saveHDF5GSVA</code> or <code>saveParquetGSVA</code> , such as the one returned by <code>gsvaMap()</code> with <code>output="HDF5"</code> or <code>output="Parquet"</code> .
dir	In <code>gsvaBatchtoolsSlurmParam()</code> , path to a directory where the output of the GSVA calculations will be saved. In this directory, <code>gsvaBatchtoolsSlurmParam()</code> also writes a file <code>gsvainit.R</code> , which is run at the start of each job, and a configuration file for the package <code>batchtools</code> , <code>gsvabatchtools.conf.R</code> , which makes <code>gsvaMap()</code> check the state of the jobs at most every 30 seconds, instead of up to every two minutes, to notice sooner jobs killed by the workload manager. This file also loads the batchtools configuration file of the user, if any, whose settings, including the interval between checks, take precedence. Default: <code>"GSAOUTPUT"</code> in the current working directory.
partition	In <code>gsvaBatchtoolsSlurmParam()</code> , name of the Slurm partition to use for the GSVA calculations. No default value, the user must provide a valid partition name for the Slurm cluster.
walltime	In <code>gsvaBatchtoolsSlurmParam()</code> , either <code>NULL</code> or the maximum wall time of each job sent to Slurm, after which Slurm ends it, given as in the <code>--time</code> option of the Slurm command <code>sbatch</code> : either a number of minutes, or a character string with one of the formats <code>"minutes"</code> , <code>"minutes:seconds"</code> , <code>"hours:minutes:seconds"</code> , <code>"days-hours"</code> , <code>"days-hours:minutes"</code> and <code>"days-hours:minutes:seconds"</code> , e.g., <code>"90"</code> or <code>"1:30:00"</code> for one hour and a half, or <code>"2-0"</code> for two days, which Slurm rounds up to minutes, or <code>"UNLIMITED"</code> , for the maximum time limit of the partition. <code>gsvaMap()</code> splits the input data into chunks, whose number depends on its size and on the memory of each job, see <code>mem</code> , and distributes them among nodes jobs, each of which processes its chunks one after another, so that <code>walltime</code> should be long enough for about the number of chunks divided by nodes; with <code>arrayjobs=TRUE</code> , it is instead the maximum wall time of each chunk. When <code>NULL</code> , no time limit is requested, and the jobs take the default time limit of the partition, which its maximum time limit bounds; e.g., the Slurm command <code>scontrol show partition</code> shows both of them in the fields <code>DefaultTime</code> and <code>MaxTime</code> . Chunks of jobs ended by Slurm can be resubmitted with the <code>MAPREDO</code> argument of <code>gsvaMap()</code> . Default: <code>NULL</code> .

nodes	In <code>gsvaBatchtoolsSlurmParam()</code> , number of independent compute nodes to distribute the GSVA calculations (tasks) across, which is the maximum number of jobs running at the same time. Default: 2.
ncpus_per_task	In <code>gsvaBatchtoolsSlurmParam()</code> , number of CPU cores to use for each independent task executed within a compute node. Default: 2.
mem	In <code>gsvaBatchtoolsSlurmParam()</code> , amount of memory to allocate for each independent task executed within a compute node, which GSVA uses as its maximum main memory, see the <code>maxmem</code> argument of <code>gsva()</code> . Each task runs one R process and, with more than one CPU core per task, as many parallel workers, and each of these R processes takes about 0.8 GB by itself for loading GSVA and the packages it depends on, so that <code>mem</code> should be at least $(1 + \text{ncpus_per_task})$ times 0.8 GB, plus the memory needed by the calculations on each chunk of data. <code>gsvaBatchtoolsSlurmParam()</code> gives a warning when <code>mem</code> is smaller than the former, while <code>gsvaMap()</code> , before submitting any job, gives an error in that case, because such jobs would be killed by the workload manager, and a warning when <code>mem</code> is smaller than the memory it estimates that the calculations on each chunk of data require. Default: "10G".
arrayjobs	In <code>gsvaBatchtoolsSlurmParam()</code> , logical vector of length <ol style="list-style-type: none"> 1. When TRUE, each of the nodes jobs is sent to Slurm as an array job, whose tasks process one chunk each, one after another, instead of as a single job processing all its chunks, so that <code>walltime</code> applies to each chunk, the scheduler may start the shorter tasks sooner, and a task ended by Slurm, e.g., for exceeding its memory or time limit, does not prevent the next tasks from processing their chunks. At most nodes tasks run at the same time. Default: FALSE.
template	In <code>gsvaBatchtoolsSlurmParam()</code> , path to the template of the Slurm job scripts, see <code>batchtools::makeClusterFunctionsSlurm()</code> . By default, the template provided by GSVA, which requests <code>mem</code> for the whole job (<code>--mem</code>), instead of the default template of batchtools for Slurm, which requests that amount for each CPU core (<code>--mem-per-cpu</code>), i.e., <code>ncpus_per_task</code> times the memory that GSVA uses in each job. A different template, e.g., one that loads modules or sets an account required by the cluster, should also request the memory given in the memory element of the resources for the whole job, and to support the arguments <code>walltime</code> and <code>arrayjobs</code> , the time limit in the <code>walltime</code> element only when it is given, which is either a number of seconds, as in the templates of batchtools, or the character string "UNLIMITED", and in array jobs, at most as many tasks at the same time as given in the <code>array.throttle</code> element, see the template provided by GSVA. Default: <code>system.file("templates", "slurm-gsva.tmpl", package="GSVA")</code> .

Value

The `gsvaMap()` function returns either a list of objects with the results of the GSVA calculations for each compute node, or a list of file paths where the results are saved. If the calculations in some of the chunks failed, `gsvaMap()` gives a warning and the corresponding elements of this list are condition objects describing the errors. This partial result can be given to the `MAPREDO` argument of a new call to `gsvaMap()` to compute only the failed chunks, but not to `gsvaReduce()`, which

gives an error. When results are saved to disk, chunks of jobs that did not deliver any result, such as jobs killed by the workload manager, are also reported as failed, unless no chunk was completed, in which case gsvaMap() gives an error. The gsvaReduce() function returns a single object that combines the results from all compute nodes. The gsvaBatchtoolsSlurmParam() function returns a BatchtoolsParam object with some sensible defaults for running GSVA calculations on a SLURM cluster.

Examples

```
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(gset1=paste0("g", 1:3),
                 gset2=paste0("g", 4:6),
                 gset3=paste0("g", 7:10))

## 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", seq_len(p), sep=""),
                          paste("s", seq_len(n), sep="")))

## build GSVA parameter object
gsvapar <- gsvaParam(y, geneSets)

## to keep this example short, the calculations below are split in chunks
## that run in this R session, by using a single worker. See further below
## how to distribute them across multiple compute nodes in a
## high-performance computing (HPC) environment
library(BiocParallel)
btpar <- BatchtoolsParam(workers=1, resources=list(ncpus=1, memory="1G"))

## calculate row-normalized expression values
gsvarnorm <- gsvaReduce(gsvaMap(gsvaRowNorm, gsvapar, BTPARAM=btpar))

## calculate column GSVA ranks
gsvaranks <- gsvaReduce(gsvaMap(gsvaColRanks, gsvarnorm, BTPARAM=btpar))

## calculate column GSVA scores
gsvaes <- gsvaReduce(gsvaMap(gsvaColScores, gsvaranks, BTPARAM=btpar))

## the example below assumes that a SLURM HPC environment is available
## with a partition named 'short' and that the user has write access to a
## filesystem path called 'GSVAOUTPUT' in the current working directory
## where this script is run and where the GSVA calculations will be saved.
## The user must ensure that this path is reachable by all compute nodes in
## the HPC environment, and must manually delete its contents after the GSVA
## calculations are finished.

gsvabtpar <- gsvaBatchtoolsSlurmParam(partition="short")
```

```

gsvaes <- gsvaReduce(gsvaMap(gsvaColScores, gsvaranks, BTPARAM=gsvabtpar))

## if some of the chunks fail, gsvaMap() gives a warning and returns a
## partial result, whose failed chunks can be resubmitted, for instance
## with more memory per task, without computing again the other chunks
gsvamapes <- gsvaMap(gsvaColScores, gsvaranks, BTPARAM=gsvabtpar)
gsvabtpar2 <- gsvaBatchtoolsSlurmParam(partition="short", mem="20G")
gsvamapes <- gsvaMap(gsvaColScores, gsvaranks, BTPARAM=gsvabtpar2,
                    MAPRED0=gsvamapes)
gsvaes <- gsvaReduce(gsvamapes)

## if the R session calling gsvaMap() ended before the jobs finished, the
## chunks without results can be resubmitted from a new R session using
## the directory where gsvaMap() saved its results
gsvamapes <- gsvaMap(gsvaColScores, gsvaranks, BTPARAM=gsvabtpar,
                    MAPRED0="GSVAOUTPUT")

```

GsvaMethodParam-class GsvaMethodParam class

Description

Virtual superclass of method parameter classes supported by GSVA.

A virtual superclass of the GSVA packages' method-specific parameter classes. The method 'details()' provides a detailed summary of the parameter values stored in this class and its subclasses.

Usage

```

## S4 method for signature 'GsvaMethodParam'
details(object)

## S4 method for signature 'avgParam'
details(object)

## S4 method for signature 'gsvaParam'
details(object)

## S4 method for signature 'ssgseaParam'
details(object)

```

Arguments

object An object of class GsvaMethodParam or one of its subclasses.

Details

GSVA implements four single-sample gene set analysis methods: PLAGE, combined z-scores, average, ssGSEA, and GSVA. All of them take at least an expression data matrix and one or more gene sets as input. Further common parameters include an assay name for use with multi-assay expression data containers, the gene ID type used by the expression data set, and a minimum and maximum size for gene sets to limit the range of gene set sizes used in an analysis. This virtual class provides the necessary slots for this shared parameter set and serves as the parent class for all GSVA method parameter classes.

The GSVA package implements four single-sample gene set analysis methods (PLAGE, combined z-scores, ssGSEA, and GSVA) and a respective method-specific parameter class that is used to invoke each of them with a matching set of parameters.

Value

The `details()` method returns a detailed summary of the parameter values stored in the `GsvaMethodParam` object and its subclasses.

Slots

`exprData` The expression data set. Must be one of the classes supported by `GsvaExprData`. For a list of these classes, see its help page using `help(GsvaExprData)`.

`geneSets` The gene sets. Must be one of the classes supported by `GsvaGeneSets`. For a list of these classes, see its help page using `help(GsvaGeneSets)`.

`assay` Character vector of length 1. The name of the assay to use in case `exprData` is a multi-assay container, otherwise ignored. By default, the first assay is used.

`annotation` An object of class `GeneIdentifierType` from package `GSEABase` describing the gene identifiers used as the row names of the expression data set. See `GeneIdentifierType` for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets. By default, this slot has value `NullIdentifier` and gene identifiers used in expression data set and gene sets are matched directly.

`minSize` Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.

`maxSize` Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is `Inf`.

`nzcount` Numeric vector of length 1. Number of non-zero values in the selected assay, if there is more than one, of the `'exprData'` slot.

`ondisk` Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value `ondisk="auto"` will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces the memory footprint. When `ondisk="yes"` it will attempt to work with an on-disk data structure, while when `ondisk="no"` it will attempt to load all the data in main memory.

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [zscoreParam](#), [avgParam](#), [plageParam](#), [ssgseaParam](#), [gsvaParam](#),
[GeneIdentifierType](#)
[plageParam](#), [zscoreParam](#), [ssgseaParam](#), [gsvaParam](#)

Examples

```
library(GSVA)

p <- 10 ## number of genes
n <- 30 ## number of samples

gsets <- list(set1=paste0("g", 1:3),
              set2=paste0("g", 4:6),
              set3=paste0("g", 7:10),
              set4=paste0("g", 10:13)) ## genes not in the expression data
gsets

y <- matrix(rnorm(n*p), nrow=p, ncol=n,
            dimnames=list(paste("g", seq_len(p), sep=""),
                          paste("s", seq_len(n), sep="")))

gsvapar <- gsvaParam(y, gsets)

class(gsvapar)
is(gsvapar, "GsvaMethodParam")

suppressPackageStartupMessages({
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)

gsvapar <- gsvaParam(geneExpCostaEtAl2021, c2BroadSets)
gsvapar
details(gsvapar)
class(gsvapar)
is(gsvapar, "GsvaMethodParam")
```

gsvaParam-class

gsvaParam *class*

Description

S4 class for GSVA method parameter objects.

Objects of class gsvaParam contain the parameters for running the GSVA method.

Usage

```
gsvaParam(
  exprData,
  geneSets,
  assay = NA_character_,
  annotation = NULL,
  minSize = 1,
  maxSize = Inf,
  rowNorm = c("ecdf", "clr", "none"),
  kcdf = c("auto", "Gaussian", "Poisson", "none"),
  kcdfNoneMinSampleSize = 200,
  tau = 1,
  maxDiff = TRUE,
  absRanking = FALSE,
  sparse = TRUE,
  checkNA = c("auto", "yes", "no"),
  use = c("everything", "all.obs", "na.rm"),
  filterRows = TRUE,
  ondisk = c("auto", "yes", "no"),
  verbose = TRUE
)

## S4 replacement method for signature 'gsvaParam,GsvaGeneSets'
geneSets(object) <- value

## S4 method for signature 'gsvaParam'
anyNA(x, recursive = FALSE)
```

Arguments

exprData	The expression data set. Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
geneSets	The gene sets. Must be one of the classes supported by GsvaGeneSets . For a list of these classes, see its help page using <code>help(GsvaGeneSets)</code> .
assay	Character vector of length 1. The name of the assay to use in case exprData is a multi-assay container, otherwise ignored. By default, an assay called 'log-counts' will be used if present, otherwise the first assay is used.
annotation	<p>An object of class <code>GeneIdentifierType</code> from package <code>GSEABase</code> describing the gene identifiers used as the row names of the expression data set. See <code>GeneIdentifierType</code> for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets.</p> <p>If the default value <code>NULL</code> is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as <code>exprData</code> (by calling gsvaAnnotation on it). If still not successful, the <code>NullIdentifier()</code> will be used as the gene identifier type, gene identifier mapping will be disabled and</p>

	gene identifiers used in expression data set and gene sets can only be matched directly.
minSize	Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.
maxSize	Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is Inf.
rowNorm	Character vector of length 1 denoting the method to use for row normalization of the input expression data. The default value rowNorm="ecdf" will perform a row normalization of the input expression data by transforming the expression values of each row using an empirical cumulative distribution function (ECDF) built from the same gene expression profile. The value rowNorm="none" will skip the row normalization step.
kcdf	Character vector of length 1 denoting the kernel to use during the non-parametric estimation of the empirical cumulative distribution function (ECDF) of expression levels across samples. The value kcdf="auto" will allow GSVA to automatically choose one of the possible values. The value kcdf="Gaussian" 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". When we do not want to use a kernel approach for the estimation of the ECDF, then we should set kcdf="none".
kcdfNoneMinSampleSize	Integer vector of length 1. When kcdf="auto", this parameter decides at what minimum sample size kcdf="none", i.e., the estimation of the empirical cumulative distribution function (ECDF) of expression levels across samples is performed directly without using a kernel. By default, this value is set to 200; see the kcdf slot.
tau	Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the GSVA (Hänzelmann et al., 2013) method. The default value is 1 as described in the paper.
maxDiff	Logical vector of length 1 which offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic. <ul style="list-style-type: none"> • FALSE: ES is calculated as the maximum distance of the random walk from 0. This approach produces a distribution of enrichment scores that is bimodal, but it can give large enrichment scores to gene sets whose genes are not concordantly activated in one direction only. • TRUE (the default): ES is calculated as the magnitude difference between the largest positive and negative random walk deviations. This default value gives larger enrichment scores to gene sets whose genes are concordantly activated in one direction only.
absRanking	Logical vector of length 1 used only when maxDiff=TRUE. When absRanking=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 absRanking=TRUE the original Kuiper statistic that sums the largest positive and negative random walk deviations is used.

sparse	Logical vector of length 1 used only when the input expression data in <code>exprData</code> is stored in a sparse matrix (e.g., a <code>dgCMatrix</code> or a <code>SingleCellExperiment</code> object storing the expression data in a <code>dgCMatrix</code>). In such a case, when <code>sparse=TRUE</code> (default), a sparse version of the GSVA algorithm will be applied. Otherwise, when <code>sparse=FALSE</code> , the classical version of the GSVA algorithm will be used.
checkNA	Character vector of length 1 specifying whether the input expression data should be checked for the presence of missing values (NA or NaN). This must be one of the strings "auto" (default), "yes", or "no". The default value "auto" means that the software will perform that check only when the input expression data is provided as a base matrix, an <code>ExpressionSet</code> or a <code>SummarizedExperiment</code> object, while every other type of input expression data container (e.g., <code>SingleCellExperiment</code> , etc.) will not be checked. If <code>checkNA="yes"</code> , then the input expression data will be checked for missing values irrespective of the object class of the data container, and if <code>checkNA="no"</code> , then that check will not be performed.
use	Character vector of length 1 specifying a policy for dealing with missing values (NA or NaN) in the input expression data argument <code>exprData</code> . It only applies when either <code>checkNA="yes"</code> , or <code>checkNA="auto"</code> (see the <code>checkNA</code> parameter). The argument value must be one of the strings "everything" (default), "all.obs", or "na.rm". The policy of the default value "everything" consists of propagating missing values so that the resulting enrichment score will be NA, whenever one or more of its contributing values is missing, giving a warning when that happens. When <code>use="all.obs"</code> , the presence of NAs in the input expression data will produce an error. Finally, when <code>use="na.rm"</code> , missing values in the input expression data will be removed from calculations, giving a warning when that happens, and giving an error if no values are left after removing the missing values.
filterRows	Logical vector of length 1, indicating whether the rows in, the input expression data, typically corresponding to transcripts, genes or proteins, should be filtered for constant expression across columns, typically corresponding to samples or cells, with respect to all available (nonmissing) values and to the non-zero values. By default, this slot is set to <code>TRUE</code> and the user may set it to <code>FALSE</code> when there is absolute certainty that no such rows exist in the input expression data, since this may save running time, especially with data sets with hundreds of thousands or millions of columns.
ondisk	Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value <code>ondisk="auto"</code> will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces the memory footprint. When <code>ondisk="yes"</code> it will attempt to work with an on-disk data structure, while when <code>ondisk="no"</code> it will attempt to load all the data in main memory.
verbose	Logical vector of length 1. It gives information about some decisions made by the software during parameter object construction when <code>verbose=TRUE</code> (default) and remains silent otherwise.
object	For the replacement method, an object of class <code>gsvaParam</code> .
value	For the replacement method, an object of the classes supported by <code>GsvaGeneSets</code> .

x	An object of class <code>gsvaParam</code> .
recursive	Not used with x being an object of class <code>gsvaParam</code> .

Details

In addition to the common parameter slots inherited from `[GsvaMethodParam]`, this class has slots for a number of method-specific parameters of the GSVA method described below.

In addition to a number of parameters shared with all methods implemented by package GSVA, GSVA takes six method-specific parameters. All of these parameters are described in detail below.

Value

A new `gsvaParam` object.

Slots

rowNorm Character vector of length 1 denoting the method to use for row normalization of the input expression data. The default value `rowNorm="ecdf"` will perform a row normalization of the input expression data by transforming the expression values of each row using an empirical cumulative distribution function (ECDF) built from the same gene expression profile. The value `rowNorm="none"` will skip the row normalization step.

kcdf Character vector of length 1 denoting the kernel to use during the non-parametric estimation of the empirical cumulative distribution function (ECDF) of expression levels across samples. The value `kcdf="auto"` will allow GSVA to automatically choose one of the possible values. The value `kcdf="Gaussian"` 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"`. When we do not want to use a kernel approach for the estimation of the ECDF, then we should set `kcdf="none"`.

kcdfNoneMinSampleSize Integer vector of length 1. When `kcdf="auto"`, this parameter decides at what minimum sample size `kcdf="none"`, i.e., the estimation of the empirical cumulative distribution function (ECDF) of expression levels across samples is performed directly without using a kernel; see the `kcdf` slot.

tau Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the GSVA (Hänzelmann et al., 2013) method.

maxDiff Logical vector of length 1 which offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic.

- FALSE: ES is calculated as the maximum distance from 0 of the random walk.
- TRUE: ES is calculated as the magnitude difference between the largest positive and negative random walk deviations.

absRanking Logical vector of length 1 used only when `maxDiff=TRUE`. When `absRanking=FALSE` a modified Kuiper statistic is used to calculate enrichment scores, taking the magnitude difference between the largest positive and negative random walk deviations. When `absRanking=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.

- sparse** Logical vector of length 1 used only when the input expression data in `exprData` is stored in a sparse matrix (e.g., a `dgCMatrx` or a container object, such as a `SingleCellExperiment`, storing the expression data in a `dgCMatrx`). In such a case, when `sparse=TRUE`, a sparse version of the GSVA algorithm will be applied. Otherwise, when `sparse=FALSE`, the classical version of the GSVA algorithm will be used.
- checkNA** Character vector of length 1. One of the strings "auto" (default), "yes", or "no", which refer to whether the input expression data should be checked for the presence of missing (NA) values.
- didCheckNA** Logical vector of length 1, indicating whether the input expression data was checked for the presence of missing (NA) values.
- anyNA** Logical vector of length 1, indicating whether the input expression data contains missing (NA) values.
- use** Character vector of length 1. One of the strings "everything" (default), "all.obs", or "na.rm", which refer to three different policies to apply in the presence of missing values in the input expression data; see `ssgseaParam`.
- filterRows** Logical vector of length 1, indicating whether the rows in, the input expression data, typically corresponding to transcripts, genes or proteins, should be filtered for constant expression across columns, typically corresponding to samples or cells, with respect to all available (nonmissing) values and to the non-zero values. By default, this slot is set to TRUE and the user may set it to FALSE when there is absolute certainty that no such rows exist in the input expression data, since this may save running time, especially with data sets with hundreds of thousands or millions of columns.

References

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013. doi:[10.1186/14712105147](https://doi.org/10.1186/14712105147)

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [GsvaMethodParam](#), [plageParam](#), [zscoreParam](#), [avgParam](#), [ssgseaParam](#)
[GeneIdentifierType](#), [matrix](#), [ExpressionSet](#), [SummarizedExperiment](#), [SingleCellExperiment](#),
[SpatialExperiment](#)

Examples

```
suppressPackageStartupMessages({
  library(GSEABase)
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)

## for simplicity, use only a subset of the sample data
se <- geneExpCostaEtAl2021[1:1000, ]
gsc <- c2BroadSets[1:100]
```

```
gp1 <- gsvaParam(se, gsc)
gp1
```

gsvaRowNorm

GSVA ranks and scores

Description

Calculate GSVA scores in three steps: (1) normalize values of expression by row; (2) calculate GSVA ranks by column from the previous row-normalized values; and (3) calculate GSVA scores by column from the previously calculated column ranks.

Usage

```
gsvaRowNorm(
  param,
  verbose = TRUE,
  dropExistingAssays = FALSE,
  errorOnTooFewRows = TRUE,
  first = NA_real_,
  last = NA_real_,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

gsvaColRanks(
  rowNormExprData,
  verbose = TRUE,
  dropExistingAssays = FALSE,
  first = NA_real_,
  last = NA_real_,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)

gsvaColScores(
  rankExprData,
  geneSets,
  verbose = TRUE,
  first = NA_real_,
  last = NA_real_,
  recompute_nzcount = FALSE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)
```

Arguments

<code>param</code>	A gsvaParam object built using the constructor function gsvaParam .
<code>verbose</code>	Gives information about each calculation step. Default: TRUE.
<code>dropExistingAssays</code>	Logical vector of length 1. It only applies when the input expression data is stored using a SummarizedExperiment derivative, which allows one to store more than one matrix of expression values in different assay slots. By default <code>dropExistingAssays=FALSE</code> and the new assay with the row-normalized expression values or the column ranks will be stored as a new assay in the same input object. When <code>dropExistingAssays=TRUE</code> , any existing assay will be dropped before adding the new assay with the row-normalized expression values or the column ranks.
<code>errorOnTooFewRows</code>	Logical vector of length 1. When TRUE (default), an error will be thrown if the number of rows in the input expression data is less than 2 after filtering out rows with constant values across columns. When FALSE, a warning will be given instead, and the returned object will either have one or no rows.
<code>first</code>	Numeric vector of length 1. First row, in the case of <code>gsvaRowNorm()</code> , or first column, in the case of <code>gsvaColRanks()</code> and <code>gsvaColScores()</code> , to which calculations should be restricted. By default, <code>first=NA_real_</code> , which implies that calculations start at the first row or column of the input expression data.
<code>last</code>	Numeric vector of length 1. Last row, in the case of <code>gsvaRowNorm()</code> , or last column, in the case of <code>gsvaColRanks()</code> and <code>gsvaColScores()</code> , to which calculations should be restricted. By default, <code>last=NA_real_</code> , which implies that calculations end at the last row or column of the input expression data.
<code>BPPARAM</code>	An object of class <code>BiocParallelParam</code> specifying parameters related to the parallel execution of some of the tasks and calculations within this function.
<code>maxmem</code>	A vector of length 1 either specifying a number in bytes, or a character string with either the word <code>auto</code> (default), or a number followed by a suffix indicating kilobytes (K), megabytes (M), gigabytes (G) or terabytes (T), which GSVA will use to attempt bounding the maximum amount of main memory used across all threads of execution to that given quantity. By default <code>maxmem="auto"</code> , indicating that the maximum memory will be the 90% of the total main memory, as calculated by Sys.meminfo() or, when it is smaller, of the memory limit of the job or container where R runs, such as a job of the SLURM workload manager. To avoid setting any bound on the maximum memory, use <code>maxmem=Inf</code> . Note that the amount of main memory used in an R session or script may depend on other commands and packages used in that same session or script. In particular, each R process running GSVA, the main one and each of its parallel workers in <code>BPPARAM</code> , takes about 0.8 GB by itself for loading GSVA and the packages it depends on, including forked workers, such as those of a MulticoreParam object, which end up taking most of that memory as their own while they run. The memory that GSVA estimates includes the expression data given to it, with all the assays of a <code>SummarizedExperiment</code> object held in main memory, but not other objects of the R session, such as other large matrices or data sets, which reduce the memory actually available to GSVA;

when such objects are present, consider setting `maxmem` to the memory that they leave. Before starting its calculations, GSVA gives a warning when the memory it estimates they require exceeds `maxmem`, suggesting to use fewer parallel workers or a larger `maxmem`. The memory taken by each R process can be set with the option `GSVA.workermem`, in bytes, and this check can be disabled with `options(GSVA.check_memory=FALSE)`.

`rowNormExprData`

A row-normalized expression data set obtained with [gsvaRowNorm](#). It can be either a single character string with the path to a file containing the row-normalized data stored with [saveHDF5GSVA](#) or [saveParquetGSVA](#), an `s3://` or `gs://` URI, or an `http://` or `https://` URL to a file stored with [saveParquetGSVA](#), or an object of one of the classes supported by [GsvaExprData](#). For a list of these classes, see `class ? GsvaExprData`.

`rankExprData`

A column-rank expression data set obtained with [gsvaColRanks](#). It can be either a single character string with the path to a file containing the column-rank data stored with [saveHDF5GSVA](#) or [saveParquetGSVA](#), an `s3://` or `gs://` URI, or an `http://` or `https://` URL to a file stored with [saveParquetGSVA](#), or an object of one of the classes supported by [GsvaExprData](#). For a list of these classes, see `class ? GsvaExprData`.

`geneSets`

An object of the classes supported by [GsvaGeneSets](#). Currently, either a [GeneSetCollection](#) object or a list object.

`recompute_nzcount`

Logical vector of length 1. When TRUE, the number of non-zero rows in the input expression data will be recomputed, internally used only.

Details

Column ranks are only valid for the rows on which they were calculated, because GSVA scores are calculated from the ranking of all those rows. For this reason, rows cannot be removed from the output of `'gsvaColRanks()'`, and `'gsvaColScores()'` gives an error when the number of rows of its input differs from the number of rows on which the ranks were calculated. Rows should be removed before calling `'gsvaRowNorm()'` instead. Removing columns does not affect the validity of the ranks.

When the column ranks given to `'gsvaColScores()'` are stored in Apache Parquet format, they are processed from disk by blocks of columns, even if they would fit in main memory, unless `ondisk="no"` was set in the original [gsvaParam](#) object.

Value

In the case of `'gsvaRowNorm()'`, an object of the same class as the input expression data given in the argument `exprData` of the `gsvaParam` object, containing the row-normalized expression values. The resulting object will have metadata with a copy of the input `gsvaParam` object, except for the `exprData` slot, and with the version of GSVA that produced it, and in the case of being a derivative of a [SummarizedExperiment](#) object, an additional assay called `"gsvarnorm"` storing the row-normalized expression values.

In the case of `'gsvaColRanks()'`, an object of the same class as the input expression data given in the argument `exprData` of the `gsvaParam` object, containing the column rank values. The resulting

object will have metadata with a copy of the input gsvaParam object, except for the exprData slot, with the version of GSVA that produced it, and with the number of rows on which the ranks were calculated, and in the case of being a derivative of a [SummarizedExperiment](#) object, an additional assay called "gsvaranks" storing the column rank values.

In the case of 'gsvaColScores()', an object of the same class as the input expression data given in the argument exprData of the gsvaParam object, containing the enrichment scores for the given gene sets. Note that while it will have the same columns as the input expression data, the rows will correspond to the gene sets for which the enrichment scores were calculated.

References

Hänzelmann, S., Castelo, R. and Guinney, J. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013. doi:[10.1186/14712105147](https://doi.org/10.1186/14712105147)

See Also

[gsvaParam](#), [gsva](#), [gsvaEnrichment](#), [BiocParallelParam](#),

Examples

```
library(GSVA)

p <- 10 ## number of genes
n <- 30 ## number of samples

## consider three disjoint gene sets
geneSets <- list(gset1=paste0("g", 1:3),
                 gset2=paste0("g", 4:6),
                 gset3=paste0("g", 7:10))

## 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", seq_len(p), sep=""),
                          paste("s", seq_len(n), sep="")))

## build GSVA parameter object
gsvapar <- gsvaParam(y, geneSets)

## calculate row-normalized expression values
gsvarownormexpr <- gsvaRowNorm(gsvapar)

## calculate GSVA column ranks
gsvacolranks <- gsvaColRanks(gsvarownormexpr)

## calculate GSVA scores
gsva_es <- gsvaColScores(gsvacolranks)

## calculate now GSVA scores in a single step
gsva_es1 <- gsva(gsvapar)

## both approaches give the same result with the same input gene sets
```

```

all.equal(gsva_es1, gsva_es)

## however, results will be (obviously) different with different gene sets
geneSets2 <- list(gset1=paste0("g", 3:6),
                  gset2=paste0("g", c(1, 2, 7, 8)))

## note that there is no need to calculate the GSEA ranks again
## geneSets(gsva_rankspar) <- geneSets2
## gsvaScores(gsva_rankspar)

```

guessGeneIdType

Guess the gene identifier type from a list of character vectors

Description

This function tries to derive the type of gene IDs used in a named list of character vectors provided as input.

Usage

```
guessGeneIdType(geneIdsList)
```

Arguments

`geneIdsList` A named list of character vectors like the ones returned by `geneIds()`.

Details

In order to make this function useful and keep it as simple as possible, we limit ourselves to the most common types of gene identifiers: "Gene IDs" consisting of digits only are considered ENTREZ IDs, anything starting with 'ENS' an ENSEMBL identifier and anything else a HuGO gene symbol.

Value

An object of a subclass of `GeneIdentifierType` derived from the input.

See Also

[GeneIdentifierType](#)

Examples

```

library(GSEA)

gsets <- list(INNATE_RESPONSE=c("AIM2", "ALPK1", "AP3B1"),
              ADAPTIVE_RESPONSE=c("CD27", "CD70", "EBAG9"))

idtype <- guessGeneIdType(gsets)

```

```
idtype  
class(idtype)
```

igsva

Gene Set Variation Analysis

Description

Starts an interactive GSVA shiny web app.

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.

Usage

```
igsva()
```

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

```
res <- igsva() ## this will open your browser with the GSVA shiny web app
```

plageParam-class	plageParam class
------------------	------------------

Description

S4 class for PLAGE method parameter objects.

Objects of class `plageParam` contain the parameters for running the PLAGE method.

Usage

```
plageParam(
  exprData,
  geneSets,
  assay = NA_character_,
  annotation = NULL,
  minSize = 1,
  maxSize = Inf,
  ondisk = c("auto", "yes", "no"),
  verbose = TRUE
)

## S4 method for signature 'plageParam'
anyNA(x, recursive = FALSE)
```

Arguments

<code>exprData</code>	The expression data set. Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
<code>geneSets</code>	The gene sets. Must be one of the classes supported by GsvaGeneSets . For a list of these classes, see its help page using <code>help(GsvaGeneSets)</code> .
<code>assay</code>	Character vector of length 1. The name of the assay to use in case <code>exprData</code> is a multi-assay container, otherwise ignored. By default, an assay called 'log-counts' will be used if present, otherwise the first assay is used.
<code>annotation</code>	An object of class <code>GeneIdentifierType</code> from package <code>GSEABase</code> describing the gene identifiers used as the row names of the expression data set. See <code>GeneIdentifierType</code> for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets. If the default value <code>NULL</code> is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as <code>exprData</code> (by calling gsvaAnnotation on it). If still not successful, the <code>NullIdentifier()</code> will be used as the gene identifier type, gene identifier mapping will be disabled and gene identifiers used in expression data set and gene sets can only be matched directly.
<code>minSize</code>	Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.

maxSize	Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is Inf.
ondisk	Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value ondisk="auto" will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces the memory footprint. When ondisk="yes" it will attempt to work with an on-disk data structure, while when ondisk="no" it will attempt to load all the data in main memory.
verbose	Logical vector of length 1. It gives information about some decisions made by the software during parameter object construction when verbose=TRUE (default) and remains silent otherwise.
x	An object of class <code>plageParam</code> . Since the PLAGE method does not handle missing (NA) values, <code>anyNA()</code> always returns FALSE for this class, without checking the input expression data.
recursive	Not used with x being an object of class <code>plageParam</code> .

Details

Since method PLAGE does not take any method-specific parameters, this class does not add any slots to the common slots inherited from `GsvaMethodParam`.

PLAGE takes a number of parameters shared with all methods implemented by package GSEA but does not take any method-specific parameters. These parameters are described in detail below.

The PLAGE score of a gene set is the first right singular vector of the standardized expression values of its genes, whose sign is arbitrary (Bro et al., 2008). GSEA orients it so that the scores correlate positively with the mean standardized expression of the genes in the gene set, in the same way as module eigengenes are aligned with the average expression of their module in the WGCNA package (Langfelder and Horvath, 2008). As a result, higher PLAGE scores tend to correspond to higher expression of the gene set, and the sign of the scores is the same across runs.

Value

A new `plageParam` object.

References

- Tomfohr, J. et al. Pathway level analysis of gene expression using singular value decomposition. *BMC Bioinformatics*, 6:225, 2005. doi:[10.1186/147121056225](https://doi.org/10.1186/147121056225)
- Bro, R., Acar, E. and Kolda, T.G. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, 22:135-140, 2008. doi:[10.1002/cem.1122](https://doi.org/10.1002/cem.1122)
- Langfelder, P. and Horvath, S. WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*, 9:559, 2008. doi:[10.1186/147121059559](https://doi.org/10.1186/147121059559)

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [GsvaMethodParam](#), [zscoreParam](#), [avgParam](#), [ssgseaParam](#), [gsvaParam](#)
[GeneIdentifierType](#)

Examples

```
suppressPackageStartupMessages({
  library(GSEABase)
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)

## for simplicity, use only a subset of the sample data
se <- geneExpCostaEtAl2021[1:1000, ]
gsc <- c2BroadSets[1:100]
pp1 <- plageParam(se, gsc)
pp1
```

readGMT

Import Gene Sets from a GMT File

Description

Imports a list of gene sets from a GMT (Gene Matrix Transposed) format file, offering a choice of ways to handle duplicated gene set names.

Usage

```
readGMT(
  con,
  sep = "\t",
  geneIdType = "auto",
  collectionType = NullCollection(),
  valueType = c("GeneSetCollection", "list"),
  deduplUse = c("first", "drop", "union", "smallest", "largest"),
  ...
)
```

Arguments

con	A connection object or a non-empty character string of length 1 containing e.g. the filename or URL of a (possibly compressed) GMT file.
sep	The character string separating members of each gene set in the GMT file.
geneIdType	By default a character vector of length 1 with the special value "auto" or an object of a subclass of <code>GeneIdentifierType</code> . If set to "auto", the function will try to derive the gene ID type from argument <code>geneIdsList</code> using guessGeneIdType . Other values, including NULL, will be ignored with a warning and <code>geneIdType=NullIdentifier()</code> will be used instead. Depending on

	the value of argument <code>valueType</code> , the gene ID type of the resulting list or of all <code>GeneSet</code> objects in the resulting <code>GeneSetCollection</code> will be set to this value.
<code>collectionType</code>	Only used when <code>valueType == "GeneSetCollection"</code> . See <code>getGmt</code> for more information.
<code>valueType</code>	A character vector of length 1 specifying the desired type of return value. It must be one of: <ul style="list-style-type: none"> • <code>GeneSetCollection</code> (the default): a <code>GeneSetCollection</code> object as defined and described by package <code>GSEABase</code>. • <code>list</code>: a named list of gene sets represented as character vectors of gene IDs. This format is much simpler and cannot store the metadata required for automatic mapping of gene IDs.
<code>deduplUse</code>	A character vector of length 1 specifying one of several methods to handle duplicated gene set names. Duplicated gene set names are explicitly forbidden by the GMT file format specification but can nevertheless be encountered in the wild. The available choices are: <ul style="list-style-type: none"> • <code>first</code> (the default): drops all gene sets whose names are duplicated according to the base R function and retains only the first occurrence of a gene set name. • <code>drop</code>: removes <i>all</i> gene sets that have a duplicated name, including its first occurrence. • <code>union</code>: replaces gene sets with duplicated names by a single gene set containing the union of all their gene IDs. • <code>smallest</code>: drops gene sets with duplicated names and retains only the smallest of them, i.e. the one with the fewest gene IDs. If there are several smallest gene sets, the first will be selected. • <code>largest</code>: drops gene sets with duplicated names and retains only the largest of them, i.e. the one with the most gene IDs. If there are several largest gene sets, the first will be selected.
<code>...</code>	Further arguments passed on to <code>readLines()</code>

Value

The gene sets imported from the GMT file, with duplicate gene sets resolved according to argument `deduplUse` and in the format determined by argument `valueType`.

See Also

[deduplicateGeneSets](#), [readLines](#), [GeneSetCollection](#), [GeneIdentifierType](#), [getGmt](#),

Examples

```
library(GSVA)
suppressPackageStartupMessages(library(GSVAdata))

fname <- file.path(system.file("extdata", package="GSVAdata"),
  "c2.subsetdups.v7.5.symbols.gmt.gz")
```

```
## by default, guess geneIdType from content and return a GeneSetCollection
genesets <- readGMT(fname)
genesets

## how to manually override the geneIdType
genesets <- readGMT(fname, geneIdType=NULLIdentifier())
genesets

## how to drop *all* gene sets with duplicated names (instead of ignoring
## only the duplicated one)
genesets <- readGMT(fname, deduplUse="drop")
genesets

## return a simple list instead of a GeneSetCollection
genesets <- readGMT(fname, valueType="list")
head(genesets, 2)

## the list has a geneIdType, too
gsvaAnnotation(genesets)
```

saveHDF5GSVA

Save/load GSVA output to disk using HDF5 or Apache Parquet format

Description

The functions `saveHDF5GSVA()` and `loadHDF5GSVA()` allow one to save and load the output from GSVA to/from disk. The `saveHDF5GSVA()` function takes the output of `gsvaRowNorm`, `gsvaColRanks` or `gsvaColScores` as input, and saves the output from these methods with the relevant metadata to a single file in HDF5 format. The `loadHDF5GSVA()` function reads the saved data from that file and returns an object with the corresponding GSVA row-normalized or rank expression values, or GSVA scores, and their corresponding metadata.

The functions `saveParquetGSVA()` and `loadParquetGSVA()` do the same using a single file in Apache Parquet format, which is organized to efficiently read blocks of columns, as `gsvaColRanks` and `gsvaColScores` do. These two functions require having installed additional packages by the user. More concretely, loading Parquet files from the local filesystem or lazily over Amazon S3 or Google Cloud Storage buckets, and saving them, require the CRAN package `arrow`, while loading Parquet files lazily over HTTP(S) URLs requires the CRAN packages `duckdb` and `DBI`.

Usage

```
saveHDF5GSVA(
  gsvaExprData,
  file,
  assay = "auto",
  replace = FALSE,
  verbose = FALSE
)
```

```
loadHDF5GSVA(file, assay = "auto", verbose = TRUE)

saveParquetGSVA(
  gsvaExprData,
  file,
  assay = "auto",
  colsPerRowGroup = "auto",
  replace = FALSE,
  verbose = FALSE
)

loadParquetGSVA(file, assay = "auto", verbose = TRUE)
```

Arguments

gsvaExprData	An object obtained with gsvaRowNorm , gsvaColRanks or gsvaColScores . Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
file	The path to the file where to save the GSVA output data in HDF5 or Apache Parquet format or, for loading it, that path or, in Apache Parquet format, also an <code>s3://</code> or <code>gs://</code> URI, or an <code>http://</code> or <code>https://</code> URL.
assay	A single character string specifying the assay that contains the GSVA output to be saved or loaded. By default, <code>assay="auto"</code> , which in the case of saving and a <code>gsvaExprData</code> object that is a <code>SummarizedExperiment</code> or one of its derivatives, it will look for an assay named <code>gsvaranks</code> , and if not found, it will look for an assay named <code>gsvarnorm</code> . If <code>gsvaExprData</code> is not a <code>SummarizedExperiment</code> or one of its derivatives, then the assay to be saved will be determined by the <code>assay</code> attribute of the <code>gsvaExprData</code> object.
replace	Logical vector of length 1. When <code>TRUE</code> , an existing file in <code>file</code> is replaced. By default, <code>replace=FALSE</code> .
verbose	Logical vector of length 1. In <code>saveHDF5GSVA()</code> and <code>saveParquetGSVA()</code> , when <code>TRUE</code> , a progress bar is shown while writing the file, and by default <code>verbose=FALSE</code> . In <code>loadHDF5GSVA()</code> and <code>loadParquetGSVA()</code> , when <code>TRUE</code> (default), messages inform about the steps of loading the file.
colsPerRowGroup	Either <code>"auto"</code> (default), or the number of matrix columns stored in each row group of the Apache Parquet file, which is the smallest unit of data that can be read from it. With <code>"auto"</code> , dense data is stored in row groups of at most 2^{20} values, which is also the maximum number of values allowed, while sparse data is stored in row groups of 500 columns. Values larger than the number of columns of the matrix store all its columns in a single row group.

Details

`saveHDF5GSVA()` stores the GSVA output values in a single HDF5 file, keeping only the non-zero values of sparse data, such as sparse GSVA ranks, in the compressed sparse column layout used by 10x Genomics, which takes less space and is read faster by blocks of columns, as [gsvaColRanks](#)

and `gsvaColScores` do, than storing all values. Dense data is stored in a dense HDF5 dataset whose chunks span all its rows. The rest of the object storing the GSVA output, such as its row and column names and data, and its GSVA metadata, is stored in the same file. The file does not store its own location, so it can be moved or copied. The object returned by `loadHDF5GSVA()` holds its values in a `DelayedMatrix` object that reads them from the file only when needed. GSVA output loaded with `loadHDF5GSVA()` can be saved in other formats of the `HDF5Array` package, e.g., with `saveHDF5SummarizedExperiment`.

`saveParquetGSVA()` stores sparse input, such as a `dgMatrix` or an `SVT_SparseMatrix` object, keeping only its non-zero values, and stores GSVA ranks as integer values. The object returned by `loadParquetGSVA()` holds its values in a `DelayedMatrix` object that reads them from the file only when needed, by blocks of columns, and `gsvaColScores` processes ranks stored in this way from disk. `loadParquetGSVA()` can also read files stored in Amazon S3 or Google Cloud Storage, given as `s3://` or `gs://` URIs, as long as the installed `arrow` package supports them; see [arrow_with_s3](#). Files in public Google Cloud Storage buckets should be given with URIs of the form `gs://anonymous@<bucket>/<path>`, which request anonymous access, since otherwise `arrow` first looks for Google Cloud credentials, and opening the file fails if they are not available. Files in private buckets require setting up such credentials, e.g., with `gcloud auth application-default login`. Requests to Google Cloud Storage are retried for at most 15 seconds, unless the URI sets another limit with the query parameter `retry_limit_seconds`, e.g., `gs://<bucket>/<path>?retry_limit_seconds=60`. Files in private Amazon S3 buckets require AWS credentials, which `arrow` reads from the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` (and `AWS_SESSION_TOKEN` for temporary credentials), or from the file `~/.aws/credentials`, whose profile can be selected with the environment variable `AWS_PROFILE`. Although `arrow` also accepts credentials within the URI, e.g., `s3://<key>:<secret>@<bucket>/<path>`, this is not recommended, because they may end up stored in scripts and in the R history; GSVA hides them in its messages.

`loadParquetGSVA()` can also lazily read files available at public `http://` or `https://` URLs using the API of the package `duckdb`, which only downloads the parts of the file that are needed, as with `s3://` and `gs://` URIs. This requires that the HTTP(S) server supports range requests, which most do. The first time, DuckDB downloads its extension `httpfs` from its own servers, which GSVA installs in the directory `file.path(tools::R_user_dir("GSVA", which="cache"), "duckdb")`. If DuckDB cannot verify the certificate of an HTTPS server, e.g., because the server does not send its intermediate certificates, a file with the certificates of the certification authorities can be given with `options(GSVA.ca_cert_file="<file>")`. This file replaces the certificates of the certification authorities that DuckDB trusts, so it should only contain certificates from trustworthy sources, including the root certificates needed by every HTTPS server accessed in the same R session.

Loading GSVA output in HDF5 or Apache Parquet format restores R objects stored in the file, as `readRDS` does, and therefore, as with files read with `readRDS()`, files should only be loaded from trusted sources.

Value

For `saveHDF5GSVA()` and `saveParquetGSVA()`, the path to the file where the data has been saved is returned invisibly. For `loadHDF5GSVA()` and `loadParquetGSVA()`, an object is returned containing the corresponding loaded GSVA row-normalized or rank expression values, and their corresponding metadata. If the saved GSVA output was originally stored in a `SummarizedExperiment` object or one of its derived classes, then the returned object will be a `SummarizedExperiment`. Otherwise,

the returned object will be a [DelayedMatrix](#) object. This is also the case for GSVA output originally stored in an [ExpressionSet](#) object, of which only the values and the GSVA metadata are saved.

Examples

```
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(gset1=paste0("g", 1:3),
                 gset2=paste0("g", 4:6),
                 gset3=paste0("g", 7:10))

## 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", seq_len(p), sep=""),
                           paste("s", seq_len(n), sep="")))

## build GSVA parameter object
gsvapar <- gsvaParam(y, geneSets)

## calculate row-normalized expression values
gsvarownorm <- gsvaRowNorm(gsvapar)

## calculate GSVA column ranks
gsvacolranks <- gsvaColRanks(gsvarownorm)

## calculate GSVA scores
es <- gsvaColScores(gsvacolranks)

## save the GSVA row-normalized expression values to disk
rnormfile <- tempfile(fileext=".h5")
saveHDF5GSVA(gsvarownorm, rnormfile)

## save the GSVA rank values to disk
ranksfile <- tempfile(fileext=".h5")
saveHDF5GSVA(gsvacolranks, ranksfile)

## load the GSVA row-normalized values from disk
loaded_gsvarownorm <- loadHDF5GSVA(rnormfile)

## check that the loaded row-normalized values provide the
## same ranks as the ones calculated from the original values
gsvacolranks_from_loaded_gsvarownorm <- gsvaColRanks(loaded_gsvarownorm)

identical(gsvacolranks, gsvacolranks_from_loaded_gsvarownorm)

## load the GSVA ranks from disk
loaded_gsvacolranks <- loadHDF5GSVA(ranksfile)

## check that the loaded ranks provide the
```

```
## same scores as the original ranks
loaded_es <- gsvaColScores(loaded_gsvacolranks)
identical(es, loaded_es)

## the same using Apache Parquet files
if (requireNamespace("arrow", quietly=TRUE)) {
  ranksfile <- tempfile(fileext=".parquet")
  saveParquetGSVA(gsvacolranks, ranksfile)
  loaded_gsvacolranks <- loadParquetGSVA(ranksfile)
  loaded_es <- gsvaColScores(loaded_gsvacolranks)
  all.equal(es, loaded_es, check.attributes=FALSE)
}
```

spatCor

Compute Spatial Autocorrelation for SpatialExperiment objects

Description

Computes spatial autocorrelation using Moran's I statistic for a `SpatialExperiment` object, using an inverse squared distance weight matrix as default, or an inverse distance weight matrix as an alternative. It also tests for spatial autocorrelation assuming normality.

Usage

```
## S4 method for signature 'SpatialExperiment'
spatCor(
  spe,
  assay = NA_character_,
  na.rm = FALSE,
  alternative = "two.sided",
  squared = TRUE,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose)
)
```

Arguments

<code>spe</code>	An object of <code>SpatialExperiment</code> class.
<code>assay</code>	Character vector of length 1, specifying the name of the assay to use. By default, an assay called 'logcounts' will be used if present, otherwise the first assay is used.
<code>na.rm</code>	A logical indicating whether missing values should be removed.
<code>alternative</code>	A character string specifying the alternative hypothesis tested against the null hypothesis of no spatial autocorrelation; must be one of "two.sided", "less", or "greater", or any unambiguous abbreviation of these.

squared	A logical indicating whether the inverse distance weight matrix should be squared or not.
verbose	Gives information about each calculation step. Default: TRUE.
BPPARAM	An object of class BiocParallelParam specifying parameters related to the parallel execution of some of the tasks and calculations within this function.

Value

A data.frame with the same row names as the original SpatialExperiment object. Columns include the observed Moran's I statistic, the expected Moran's I statistic under no spatial autocorrelation, the expected standard deviation under no spatial autocorrelation, and the p-value of the test.

See Also

[BiocParallelParam](#)

Examples

```
suppressPackageStartupMessages({
  library(Matrix)
  library(GSVAdata)
})

spe <- HumanCerebellumNormSubset()

set.seed(123) ## for reproducibility of the random gene sets
## build two gene sets with 4 randomly chosen genes and one
## third gene set with a few microglia marker genes
gsets <- list(gset1=sample(rownames(spe), size=4, replace=FALSE),
              gset2=sample(rownames(spe), size=4, replace=FALSE),
              microglia=c("ENSG00000078808", "ENSG00000116251",
                          "ENSG00000142583", "ENSG00000173372"))

## calculate GSVA enrichment scores
gsvapar <- gsvaParam(spe, gsets, verbose=FALSE)
es <- gsva(gsvapar, verbose=FALSE)

## calculate spatial autocorrelation on the GSVA enrichment scores
spatCor(es, verbose=FALSE)
```

ssgseaParam-class

ssgseaParam *class*

Description

S4 class for ssGSEA method parameter objects.

Objects of class ssgseaParam contain the parameters for running the ssGSEA method.

Usage

```
ssgseaParam(
  exprData,
  geneSets,
  assay = NA_character_,
  annotation = NULL,
  minSize = 1,
  maxSize = Inf,
  alpha = 0.25,
  normalize = TRUE,
  checkNA = c("auto", "yes", "no"),
  use = c("everything", "all.obs", "na.rm"),
  ondisk = c("auto", "yes", "no"),
  verbose = TRUE
)

## S4 method for signature 'ssgseaParam'
anyNA(x, recursive = FALSE)
```

Arguments

exprData	The expression data set. Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
geneSets	The gene sets. Must be one of the classes supported by GsvaGeneSets . For a list of these classes, see its help page using <code>help(GsvaGeneSets)</code> .
assay	Character vector of length 1. The name of the assay to use in case exprData is a multi-assay container, otherwise ignored. By default, an assay called 'log-counts' will be used if present, otherwise the first assay is used.
annotation	<p>An object of class <code>GeneIdentifierType</code> from package <code>GSEABase</code> describing the gene identifiers used as the row names of the expression data set. See <code>GeneIdentifierType</code> for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets.</p> <p>If the default value <code>NULL</code> is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as <code>exprData</code> (by calling gsvaAnnotation on it). If still not successful, the <code>NullIdentifier()</code> will be used as the gene identifier type, gene identifier mapping will be disabled and gene identifiers used in expression data set and gene sets can only be matched directly.</p>
minSize	Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.
maxSize	Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is <code>Inf</code> .
alpha	Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the ssGSEA (Barbie et al., 2009) method. The default value is 0.25 as described in the paper.

normalize	Logical vector of length 1; if TRUE 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. Otherwise this final normalization step is skipped.
checkNA	Character vector of length 1 specifying whether the input expression data should be checked for the presence of missing values (NA or NaN). This must be one of the strings "auto" (default), "yes", or "no". The default value "auto" means that the software will perform that check only when the input expression data is provided as a base matrix, an ExpressionSet or a SummarizedExperiment object, while every other type of input expression data container (e.g., SingleCellExperiment, etc.) will not be checked. If checkNA="yes", then the input expression data will be checked for missing values irrespective of the object class of the data container, and if checkNA="no", then that check will not be performed.
use	Character vector of length 1 specifying a policy for dealing with missing values (NA or NaN) in the input expression data argument exprData. It only applies when either checkNA="yes", or checkNA="auto" (see the checkNA parameter. The argument value must be one of the strings "everything" (default), "all.obs", or "na.rm". The policy of the default value "everything" consists of propagating missing values so that the resulting enrichment score will be NA, whenever one or more of its contributing values is missing, giving a warning when that happens. When use="all.obs", the presence of NAs in the input expression data will produce an error. Finally, when use="na.rm", missing values in the input expression data will be removed from calculations, giving a warning when that happens, and giving an error if no values are left after removing the missing values.
ondisk	Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value ondisk="auto" will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces the memory footprint. When ondisk="yes" it will attempt to work with an on-disk data structure, while when ondisk="no" it will attempt to load all the data in main memory.
verbose	Logical vector of length 1. It gives information about some decisions made by the software during parameter object construction when verbose=TRUE (default) and remains silent otherwise.
x	An object of class ssgseaParam .
recursive	Not used with x being an object of class ssgseaParam .

Details

In addition to the common parameter slots inherited from `[GsvaMethodParam]`, this class has slots for the two method-specific parameters of the ssGSEA method described below as well as four more slots for implementing a missing value policy.

In addition to a number of parameters shared with all methods implemented by package GSEA, ssGSEA takes two method-specific parameters as well as two more parameters for implementing a missing value policy. All of these parameters are described in detail below.

Value

A new `ssgseaParam` object.

Slots

`alpha` Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the ssGSEA (Barbie et al., 2009) method.

`normalize` Logical vector of length 1. If TRUE 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. Otherwise this final normalization step is skipped.

`checkNA` Character vector of length 1. One of the strings "auto" (default), "yes", or "no", which refer to whether the input expression data should be checked for the presence of missing (NA) values.

`didCheckNA` Logical vector of length 1, indicating whether the input expression data was checked for the presence of missing (NA) values.

`anyNA` Logical vector of length 1, indicating whether the input expression data contains missing (NA) values.

`use` Character vector of length 1. One of the strings "everything" (default), "all.obs", or "na.rm", which refer to three different policies to apply in the presence of missing values in the input expression data; see `ssgseaParam`.

References

Barbie, D.A. et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(5):108-112, 2009. doi:[10.1038/nature08460](https://doi.org/10.1038/nature08460)

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [GsvaMethodParam](#), [plageParam](#), [zscoreParam](#), [avgParam](#), [gsvaParam](#), [GeneIdentifierType](#), [matrix](#), [ExpressionSet](#), [SummarizedExperiment](#), [SingleCellExperiment](#)

Examples

```
suppressPackageStartupMessages({
  library(GSEABase)
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)

## for simplicity, use only a subset of the sample data
se <- geneExpCostaEtAl2021[1:1000, ]
gsc <- c2BroadSets[1:100]
sp1 <- ssgseaParam(se, gsc)
sp1
```

zscoreParam-class zscoreParam *class*

Description

S4 class for combined z-scores method parameter objects.

Objects of class `zscoreParam` contain the parameters for running the combined z-scores method.

Usage

```
zscoreParam(
  exprData,
  geneSets,
  assay = NA_character_,
  annotation = NULL,
  minSize = 1,
  maxSize = Inf,
  ondisk = c("auto", "yes", "no"),
  verbose = TRUE
)

## S4 method for signature 'zscoreParam'
anyNA(x, recursive = FALSE)
```

Arguments

<code>exprData</code>	The expression data set. Must be one of the classes supported by GsvaExprData . For a list of these classes, see its help page using <code>help(GsvaExprData)</code> .
<code>geneSets</code>	The gene sets. Must be one of the classes supported by GsvaGeneSets . For a list of these classes, see its help page using <code>help(GsvaGeneSets)</code> .
<code>assay</code>	Character vector of length 1. The name of the assay to use in case <code>exprData</code> is a multi-assay container, otherwise ignored. By default, an assay called 'log-counts' will be used if present, otherwise the first assay is used.
<code>annotation</code>	<p>An object of class <code>GeneIdentifierType</code> from package <code>GSEABase</code> describing the gene identifiers used as the row names of the expression data set. See <code>GeneIdentifierType</code> for help on available gene identifier types and how to construct them. This information can be used to map gene identifiers occurring in the gene sets.</p> <p>If the default value <code>NULL</code> is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as <code>exprData</code> (by calling gsvaAnnotation on it). If still not successful, the <code>NullIdentifier()</code> will be used as the gene identifier type, gene identifier mapping will be disabled and gene identifiers used in expression data set and gene sets can only be matched directly.</p>
<code>minSize</code>	Numeric vector of length 1. Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 1.

maxSize	Numeric vector of length 1. Maximum size of the resulting gene sets after gene identifier mapping. By default, the maximum size is Inf.
ondisk	Character vector of length 1 denoting whether an on-disk backend should be used to reduce the memory footprint. The default value <code>ondisk="auto"</code> will attempt to load all the data in main memory when the input nonzero values fit in main memory, otherwise it will attempt working with an on-disk data structure that reduces the memory footprint. When <code>ondisk="yes"</code> it will attempt to work with an on-disk data structure, while when <code>ondisk="no"</code> it will attempt to load all the data in main memory.
verbose	Logical vector of length 1. It gives information about some decisions made by the software during parameter object construction when <code>verbose=TRUE</code> (default) and remains silent otherwise.
x	An object of class <code>zscoreParam</code> . Since the z-score method does not handle missing (NA) values, <code>anyNA()</code> always returns FALSE for this class, without checking the input expression data.
recursive	Not used with x being an object of class <code>zscoreParam</code> .

Details

Since the combined z-scores method does not take any method-specific parameters, this class does not add any slots to the common slots inherited from `GsvaMethodParam`.

The combined z-scores method takes a number of parameters shared with all methods implemented by package GSVA but does not take any method-specific parameters.

Value

A new `zscoreParam` object.

References

Lee, E. et al. Inferring pathway activity toward precise disease classification. *PLoS Comp Biol*, 4(11):e1000217, 2008. doi:[10.1371/journal.pcbi.1000217](https://doi.org/10.1371/journal.pcbi.1000217)

See Also

[GsvaExprData](#), [GsvaGeneSets](#), [GsvaMethodParam](#), [plageParam](#), [ssgseaParam](#), [gsvaParam](#)
[GeneIdentifierType](#)

Examples

```
suppressPackageStartupMessages({
  library(GSEABase)
  library(GSVA)
  library(GSVAdata)
})

data(geneprotExpCostaEtAl2021)
data(c2BroadSets)
```

```
## for simplicity, use only a subset of the sample data
se <- geneExpCostaEtAl2021[1:1000, ]
gsc <- c2BroadSets[1:100]
zp1 <- zscoreParam(se, gsc)
zp1
```

Index

- * **GSVA**
 - igsva, [39](#)
- * **Gene**
 - computeGeneSetsOverlap, [6](#)
 - filterGeneSets, [8](#)
- * **set**
 - computeGeneSetsOverlap, [6](#)
 - filterGeneSets, [8](#)
- * **shiny**
 - igsva, [39](#)
- anyNA, avgParam-method (avgParam-class), [3](#)
- anyNA, gsvaParam-method (gsvaParam-class), [28](#)
- anyNA, plageParam-method (plageParam-class), [40](#)
- anyNA, ssgseaParam-method (ssgseaParam-class), [49](#)
- anyNA, zscoreParam-method (zscoreParam-class), [53](#)
- arrow_with_s3, [46](#)
- avgParam, [4](#), [5](#), [28](#), [33](#), [41](#), [52](#)
- avgParam (avgParam-class), [3](#)
- avgParam-class, [3](#)
- batchtools::makeClusterFunctionsSlurm(), [24](#)
- BatchtoolsParam, [21](#), [22](#)
- BiocParallelParam, [14](#), [37](#), [49](#)
- computeGeneSetsOverlap, [6](#), [9](#)
- computeGeneSetsOverlap, GeneSetCollection, character-method (computeGeneSetsOverlap), [6](#)
- computeGeneSetsOverlap, list, character-method (computeGeneSetsOverlap), [6](#)
- deduplicateGeneSets, [7](#), [10](#), [43](#)
- DelayedMatrix, [14](#), [20](#), [46](#), [47](#)
- details, avgParam-method (GsvaMethodParam-class), [26](#)
- details, GsvaMethodParam-method (GsvaMethodParam-class), [26](#)
- details, gsvaParam-method (GsvaMethodParam-class), [26](#)
- details, ssgseaParam-method (GsvaMethodParam-class), [26](#)
- dgCMatrx, [14](#), [17](#), [20](#), [46](#)
- ExpressionSet, [17](#), [20](#), [33](#), [47](#), [52](#)
- filterGeneSets, [7](#), [8](#)
- filterGeneSets, GeneSetCollection-method (filterGeneSets), [8](#)
- filterGeneSets, list-method (filterGeneSets), [8](#)
- GeneIdentifierType, [5](#), [10](#), [17](#), [28](#), [33](#), [38](#), [41](#), [43](#), [52](#), [54](#)
- geneIds, [10](#)
- geneIdsToGeneSetCollection, [9](#)
- GeneSet, [10](#)
- GeneSetCollection, [10](#), [20](#), [36](#), [43](#)
- geneSets, [10](#)
- geneSets, GsvaExprData-method (geneSets), [10](#)
- geneSets, GsvaMethodParam-method (geneSets), [10](#)
- geneSets, SingleCellExperiment-method (geneSets), [10](#)
- geneSets, SpatialExperiment-method (geneSets), [10](#)
- geneSets, SummarizedExperiment-method (geneSets), [10](#)
- geneSets<- (gsvaParam-class), [28](#)
- geneSets<- , gsvaParam, GsvaGeneSets-method (gsvaParam-class), [28](#)
- geneSetSizes (geneSets), [10](#)
- geneSetSizes, GsvaExprData-method (geneSets), [10](#)

- geneSetSizes, GsvaMethodParam-method
(geneSets), 10
- getGmt, 43
- gsva, 37
- gsva (gsva, avgParam-method), 12
- gsva(), 24, 39
- gsva, avgParam-method, 12
- gsva, gsvaParam-method
(gsva, avgParam-method), 12
- gsva, plageParam-method
(gsva, avgParam-method), 12
- gsva, ssgseaParam-method
(gsva, avgParam-method), 12
- gsva, zscoreParam-method
(gsva, avgParam-method), 12
- gsvaAnnotation, 3, 15, 29, 40, 50, 53
- gsvaAnnotation, ExpressionSet-method
(gsvaAnnotation), 15
- gsvaAnnotation, GeneSetCollection-method
(gsvaAnnotation), 15
- gsvaAnnotation, GsvaExprData-method
(gsvaAnnotation), 15
- gsvaAnnotation, list-method
(gsvaAnnotation), 15
- gsvaAnnotation, SingleCellExperiment-method
(gsvaAnnotation), 15
- gsvaAnnotation, SpatialExperiment-method
(gsvaAnnotation), 15
- gsvaAnnotation, SummarizedExperiment-method
(gsvaAnnotation), 15
- gsvaAnnotation<- (gsvaAnnotation), 15
- gsvaAnnotation<- , ExpressionSet, character-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , ExpressionSet, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , GsvaExprData, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , list, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , SingleCellExperiment, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , SpatialExperiment, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaAnnotation<- , SummarizedExperiment, GeneIdentifierType-method
(gsvaAnnotation), 15
- gsvaBatchtoolsSlurmParam (gsvaMap), 21
- gsvaColRanks, 14, 18, 22, 36, 44, 45
- gsvaColRanks (gsvaRowNorm), 34
- gsvaColScores, 14, 44–46
- gsvaColScores (gsvaRowNorm), 34
- gsvaEnrichment, 17, 37
- GsvaExprData, 3, 5, 11, 17, 18, 22, 28, 29, 33, 36, 40, 41, 45, 50, 52–54
- GsvaExprData (GsvaExprData-class), 19
- GsvaExprData-class, 19
- GsvaGeneSets, 3, 5, 28, 29, 31, 33, 36, 40, 41, 50, 52–54
- GsvaGeneSets-class, 20
- gsvaMap, 21
- GsvaMethodParam, 5, 11, 33, 41, 52, 54
- GsvaMethodParam-class, 26
- gsvaParam, 5, 13, 14, 22, 28, 31, 32, 35–37, 41, 52, 54
- gsvaParam (gsvaParam-class), 28
- gsvaParam-class, 28
- gsvaRanksParam-class (gsvaParam-class), 28
- gsvaReduce (gsvaMap), 21
- gsvaRowNorm, 14, 22, 34, 36, 44, 45
- guessGeneIdType, 10, 38, 42
- HDF5Array, 20
- igsva, 39
- list, 20
- loadHDF5GSVA (saveHDF5GSVA), 44
- loadParquetGSVA (saveHDF5GSVA), 44
- matrix, 20, 33, 52
- MulticoreParam, 14, 35
- plageParam, 5, 13, 14, 28, 33, 41, 52, 54
- plageParam (plageParam-class), 40
- plageParam-class, 40
- readGMT, 7, 42
- readLines, 43
- readRDS, 46
- saveHDF5GSVA, 22, 23, 36, 44
- saveHDF5GSVA, SummarizedExperiment, 46
- saveParquetGSVA, 22, 23, 36
- saveParquetGSVA, SingleCellExperiment, 44
- saveHDF5GSVA, SingleCellExperiment, 14, 20, 33, 52
- spatCor, 48
- spatCor, SpatialExperiment-method
(spatCor), 48

SpatialExperiment, [14](#), [20](#), [33](#)
ssgseaParam, [5](#), [13](#), [14](#), [28](#), [33](#), [41](#), [51](#), [52](#), [54](#)
ssgseaParam (ssgseaParam-class), [49](#)
ssgseaParam-class, [49](#)
SummarizedExperiment, [14](#), [17](#), [20](#), [33](#),
[35–37](#), [46](#), [52](#)
SVT_SparseMatrix, [14](#), [20](#), [46](#)
Sys.meminfo(), [13](#), [35](#)

zscoreParam, [5](#), [13](#), [14](#), [28](#), [33](#), [41](#), [52](#), [54](#)
zscoreParam (zscoreParam-class), [53](#)
zscoreParam-class, [53](#)