Package: GSVA (via r-universe)

November 6, 2026

Version 2.6.15

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

Depends R (>= 4.0.0)

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

Suggests RUnit, BiocStyle, knitr, rmarkdown, limma, RColorBrewer, org.Hs.eg.db, genefilter, edgeR, GSVAdata, sva, ExperimentHub, TENxPBMCData, TENxVisiumData, spatialLIBD, scrapper, bluster, igraph, ggspavis, patchwork, ggplot2, shiny, shinydashboard, data.table, plotly, future, promises, shinybusy, shinyjs

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 8.1.0

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

Repository Bioconductor 3.23

Date/Publication 2026-10-22 17:56:24 UTC

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

RemoteRef RELEASE_3_23

RemoteSha d2fdc554b5578be78a54aa59c50efd62bff46b5a

Contents

computeGeneSetsOverlap 2
deduplicateGeneSets 4
filterGeneSets 5
geneIdsToGeneSetCollection 6
geneSets 7
gsva 9
gsvaAnnotation 12
gsvaEnrichment 14
GsvaExprData-class 15
GsvaGeneSets-class 16
GsvaMethodParam-class 17
gsvaParam-class 18
gsvaRanks 23
guessGeneIdType 25
igsva 26
plageParam-class 28
readGMT 30
saveHDF5GSVAranks 32
spatCor 33
ssgseaParam-class 35
zscoreParam-class 38

Index **41**

`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

`gSets`	Gene sets given either as a `list` or a `GeneSetCollection` object.
`uniqGenes`	Vector of unique genes to be considered when calculating the overlaps.
`minSize`	Minimum size.
`maxSize`	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`	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: • `first` (the default): drops all gene sets whose names are duplicated according to the base R function and retains only the first occurence of a gene set name. • `drop`: removes *all* 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` *Filter gene sets*

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 `geneIds()`. Names must be unique; otherwise see `deduplicateGeneSets()` 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 *Retrieve or Determine Gene Sets*

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", 1:p, sep="") , paste("s", 1:n, sep="")))
```

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

es <- gsva(gsvapar)

geneSets(es)
```

---

gsva *Gene Set Variation Analysis*

Description

Estimates GSVA enrichment scores.

Usage

```
## 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	A parameter object of one of the following classes: • A gsvaParam object built using the constructor function gsvaParam. This object will trigger gsva() to use the GSVA algorithm by Hänzelmann et al. (2013). • A plageParam object built using the constructor function plageParam. This object will trigger gsva() to use the PLAGE algorithm by Tomfohr et al. (2005). • A zscoreParam object built using the constructor function zscoreParam. This object will trigger gsva() 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 gsva() to use the ssGSEA algorithm by Barbie et al. (2009).
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.
maxmem	A vector of length 1 either specifying a number in bytes, or a character string with either the word auto (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 maxmem="auto", indicating that the maximum memory will be the 90% of the total main memory, as calculated by Sys.meminfo(). 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.

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

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

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

See Also

plageParam, zscoreParam, ssgseaParam, gsvaParam, BiocParallelParam, dgCMatrix, ExpressionSet, SingleCellExperiment

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", 1:p, sep="") , paste("s", 1:n, sep="")))

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

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

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

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

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

## 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")
```

---

gsvaAnnotation *Store and Retrieve Annotation Metadata*

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 dgCMatrix 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 dgCMatrix 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, dgCMatrix

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

```
## S4 method for signature 'gsvaRanksParam'
gsvaEnrichment(
  param,
  column = 1,
  geneSet = 1,
  plot = c("auto", "base", "ggplot", "no"),
  ...
)
```

Arguments

`param`	A `gsvaRanksParam` object obtained with the method `gsvaRanks`.
`column`	The column for which we want to retrieve the enrichment data. This parameter is only available in the `gsvaEnrichment()` method.
`geneSet`	Either a positive integer number between 1 and the number of available gene sets in `param`, or a character string with the name of one of the gene sets available in `param`.
`plot`	A character string indicating whether an enrichment plot should be produced using either base R graphics (`plot="base"`) or the ggplot2 package (`plot="ggplot"`), or not (`plot="no"`). In the latter case, the enrichment data will be returned. By default `plot="auto"`, 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 `plot()` function when the previous parameter `plot="base"`.

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. GSVA: Gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics*, 14:7, 2013. doi:10.1186/14712105147

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", 1:p, sep="") , paste("s", 1:n, sep="")))

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

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

## calculate GSVA ranks
gsvarankspar <- gsvaRanks(gsvapar)
gsvarankspar

## by default the enrichment data for the first column and the first
## gene set are retrieved
gsvaEnrichment(gsvarankspar)
```

GsvaExprData-class GsvaExprData *class*

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.

See Also

matrix, dgCMatrix, SVT_SparseMatrix, DelayedMatrix, ExpressionSet, SummarizedExperiment, SingleCellExperiment, SpatialExperiment, HDF5Array

---

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")
```

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.

Details

GSVA implements four single-sample gene set analysis methods: PLAGE, combined z-scores, 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.

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 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.

See Also

GsvaExprData, GsvaGeneSets, zscoreParam, 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", 1:p, sep="") , paste("s", 1:n, sep="")))

gsvapar <- gsvaParam(y, gsets)

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,
```

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

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

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 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, an assay called 'log-counts' will be used if present, otherwise 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. If the default value NULL is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as exprData (by calling gsvaAnnotation on it). If still not successful, the NullIdentifier() 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.
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.
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.

- `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 `exprData` is stored in a sparse matrix (e.g., a `dgCMatrix` or a `SingleCellExperiment` object storing the expression data in a `dgCMatrix`). In such a case, when `sparse=TRUE` (default), 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 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.

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.

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 gsvaParam.

recursive — Not used with x being an object of class gsvaParam.

object — For the replacement method, an object of class gsvaRanksParam.

value — For the replacement method, an object of the classes supported by GsvaGeneSets.

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

`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 `dgCMatrix` or a container object, such as a `SingleCellExperiment`, storing the expression data in a `dgCMatrix`). 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

See Also

GsvaExprData, GsvaGeneSets, GsvaMethodParam, plageParam, zscoreParam, ssgseaParam

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

---

gsvaRanks *GSVA ranks and scores*

Description

Calculate GSVA scores in two steps: (1) calculate GSVA ranks; and (2) calculate GSVA scores using the previously calculated ranks.

Usage

```
## S4 method for signature 'gsvaParam'
gsvaRanks(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
```

```
  maxmem = "auto"
)

## S4 method for signature 'gsvaRanksParam'
gsvaScores(
  param,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  maxmem = "auto"
)
```

Arguments

param	A parameter object of the gsvaRanksParam class.
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.
maxmem	A vector of length 1 either specifying a number in bytes, or a character string with either the word auto (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 maxmem="auto", indicating that the maximum memory will be the 90% of the total main memory, as calculated by Sys.meminfo(). 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.

Value

In the case of the gsvaRanks() method, an object of class gsvaRanksParam.

In the case of the gsvaScores() method, a gene-set by sample matrix of GSVA enrichment scores stored in a container object of the same type as the input ranks data container. If the input was a base matrix or a dgCMatrix object, then the output will be a base matrix object with the gene sets employed in the calculations stored in an attribute called geneSets. 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 attributed called geneSets. If the input was an object of one of the classes described in GsvaExprData, such as a SingleCellExperiment, 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

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

See Also

gsvaParam, gsvaRanksParam, gsva, BiocParallelParam, dgCMatrix, ExpressionSet, SingleCellExperiment

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", 1:p, sep="") , paste("s", 1:n, sep="")))

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

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

## calculate GSVA ranks
gsvarankspar <- gsvaRanks(gsvapar)
gsvarankspar
## calculate GSVA scores
gsva_es <- gsvaScores(gsvarankspar)
gsva_es

## 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 GSVA ranks again
geneSets(gsvarankspar) <- geneSets2
gsvaScores(gsvarankspar)
```

---

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(GSVA)

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

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, an assay called 'log-counts' will be used if present, otherwise 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. If the default value NULL is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as exprData (by calling gsvaAnnotation on it). If still not successful, the NullIdentifier() 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.
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.
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 plageParam. Since the PLAGE method does not handle missing (NA) values, anyNA() always returns FALSE for this class, without checking the input expression data.
recursive	Not used with x being an object of class plageParam.

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 GSVA but does not take any method-specific parameters. These parameters are described in detail below.

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

See Also

GsvaExprData, GsvaGeneSets, GsvaMethodParam, zscoreParam, 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 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. Depending on the value of argument valueType, the gene ID type of the resulting list or of all GeneSet objects in the resulting GeneSetCollection will be set to this value.
collectionType	Only used when valueType == "GeneSetCollection". See getGmt for more information.
valueType	A character vector of length 1 specifying the desired type of return value. It must be one of: • GeneSetCollection (the default): a GeneSetCollection object as defined and described by package GSEABase.

- `list`: 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.

`deduplUse` 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:

- `first` (the default): drops all gene sets whose names are duplicated according to the base R function and retains only the first occurence of a gene set name.
- `drop`: removes *all* 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.

`...` Further arguments passed on to `readLines()`

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)
```

---

saveHDF5GSVAranks *Save/load GSVA rank values to disk using HDF5 format*

Description

The functions saveHDF5GSVAranks and loadHDF5GSVAranks can be used to save and load the GSVA rank values to/from disk, respectively. The saveHDF5GSVAranks function takes a gsvaRanksParam object and saves the rank values along with the relevant metadata to a specified directory. The loadHDF5GSVAranks function reads the saved data from the specified directory and reconstructs the gsvaRanksParam object with the rank values and their corresponding metadata.

Usage

```
saveHDF5GSVAranks(x, dir, ...)

loadHDF5GSVAranks(dir, ...)
```

Arguments

x	A gsvaRanksParam object to save to disk.
dir	The path to the directory where to save or load the GSVA ranks data.
...	Additional arguments to be passed to the underlying HDF5 saving/loading functions saveHDF5SummarizedExperiment and loadHDF5SummarizedExperiment, respectively.

Value

For saveHDF5GSVAranks, the path to the directory where the data has been saved is returned invisibly. For loadHDF5GSVAranks, a gsvaRanksParam object is returned containing the loaded GSVA rank values and their corresponding metadata.

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", 1:p, sep="") , paste("s", 1:n, sep="")))

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

## calculate GSVA ranks
gsvarankspar <- gsvaRanks(gsvapar)

## calculate GSVA scores
es <- gsvaScores(gsvarankspar)

## save the GSVA ranks to disk
dir <- tempfile()
saveHDF5GSVAranks(gsvarankspar, dir)

## load the GSVA ranks from disk
loaded_gsvarankspar <- loadHDF5GSVAranks(dir)

## check that the loaded ranks provide the
## same scores as the original ranks
loaded_es <- gsvaScores(loaded_gsvarankspar)
identical(es, loaded_es)
```

`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

spe	An object of SpatialExperiment class.
assay	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.
na.rm	A logical indicating whether missing values should be removed.
alternative	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 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, an assay called 'log-counts' will be used if present, otherwise 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.

  If the default value NULL is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as exprData (by calling gsvaAnnotation on it). If still not successful, the NullIdentifier() 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.

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.

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 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 `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 GSVA, `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

See Also

GsvaExprData, GsvaGeneSets, GsvaMethodParam, plageParam, zscoreParam, 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

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, an assay called 'log-counts' will be used if present, otherwise 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. If the default value NULL is provided, an attempt will be made to extract the gene identifier type from the expression data set provided as exprData (by calling gsvaAnnotation on it). If still not successful, the NullIdentifier() 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.
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.
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 zscoreParam. Since the z-score method does not handle missing (NA) values, anyNA() always returns FALSE for this class, without checking the input expression data.
recursive	Not used with x being an object of class zscoreParam.

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

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, 26
∗ **Gene**
computeGeneSetsOverlap, 2
filterGeneSets, 5
∗ **set**
computeGeneSetsOverlap, 2
filterGeneSets, 5
∗ **shiny**
igsva, 26

anyNA,gsvaParam-method
*(*gsvaParam-class*)*, 18
anyNA,plageParam-method
*(*plageParam-class*)*, 28
anyNA,ssgseaParam-method
*(*ssgseaParam-class*)*, 35
anyNA,zscoreParam-method
*(*zscoreParam-class*)*, 38

BiocParallelParam, *11*, *24*, *34*

computeGeneSetsOverlap, 2, *6*
computeGeneSetsOverlap,GeneSetCollection,character-method
*(*computeGeneSetsOverlap*)*, 2
computeGeneSetsOverlap,list,character-method
*(*computeGeneSetsOverlap*)*, 2

deduplicateGeneSets, 4, *7*, *31*
DelayedMatrix, *10*, *16*
dgCMatrix, *10, 11*, *13*, *16*, *24*

ExpressionSet, *11*, *13*, *16*, *23, 24*, *38*

filterGeneSets, *3*, 5
filterGeneSets,GeneSetCollection-method
*(*filterGeneSets*)*, 5
filterGeneSets,list-method
*(*filterGeneSets*)*, 5

GeneIdentifierType, *7*, *13*, *18*, *23*, *26*, *29*, *31*, *38*, *40*
geneIds, *7*
geneIdsToGeneSetCollection, 6
GeneSet, *7*
GeneSetCollection, *7*, *16*, *31*
geneSets, 7
geneSets,GsvaExprData-method
*(*geneSets*)*, 7
geneSets,GsvaMethodParam-method
*(*geneSets*)*, 7
geneSets,SingleCellExperiment-method
*(*geneSets*)*, 7
geneSets,SpatialExperiment-method
*(*geneSets*)*, 7
geneSets,SummarizedExperiment-method
*(*geneSets*)*, 7
geneSets<- *(*gsvaParam-class*)*, 18
geneSets<-,gsvaRanksParam,GsvaGeneSets-method
*(*gsvaParam-class*)*, 18
geneSetSizes *(*geneSets*)*, 7
geneSetSizes,GsvaExprData-method
*(*geneSets*)*, 7
geneSetSizes,GsvaMethodParam-method
*(*geneSets*)*, 7
getGmt, *31*
gsva, 9, *24*
gsva(), *27*
gsva,gsvaParam-method *(*gsva*)*, 9
gsva,plageParam-method *(*gsva*)*, 9
gsva,ssgseaParam-method *(*gsva*)*, 9
gsva,zscoreParam-method *(*gsva*)*, 9
gsvaAnnotation, 12, *19*, *28*, *36*, *39*
gsvaAnnotation,ExpressionSet-method
*(*gsvaAnnotation*)*, 12
gsvaAnnotation,GeneSetCollection-method
*(*gsvaAnnotation*)*, 12
gsvaAnnotation,GsvaExprData-method
*(*gsvaAnnotation*)*, 12
gsvaAnnotation,list-method
*(*gsvaAnnotation*)*, 12

gsvaAnnotation,SingleCellExperiment-method (gsvaAnnotation), 12
gsvaAnnotation,SpatialExperiment-method (gsvaAnnotation), 12
gsvaAnnotation,SummarizedExperiment-method (gsvaAnnotation), 12
gsvaAnnotation<- (gsvaAnnotation), 12
gsvaAnnotation<-,ExpressionSet,character-method (gsvaAnnotation), 12
gsvaAnnotation<-,ExpressionSet,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaAnnotation<-,GsvaExprData,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaAnnotation<-,list,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaAnnotation<-,SingleCellExperiment,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaAnnotation<-,SpatialExperiment,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaAnnotation<-,SummarizedExperiment,GeneIdentifierType-method (gsvaAnnotation), 12
gsvaEnrichment, 14
gsvaEnrichment,gsvaRanksParam-method (gsvaEnrichment), 14
GsvaExprData, *8*, *13*, *18, 19*, *23, 24*, *28, 29*, *35*, *38–40*
GsvaExprData (GsvaExprData-class), 15
GsvaExprData-class, 15
GsvaGeneSets, *18, 19*, *21*, *23*, *28, 29*, *35*, *38–40*
GsvaGeneSets-class, 16
GsvaMethodParam, *8*, *23*, *29*, *38*, *40*
GsvaMethodParam-class, 17
gsvaParam, *10, 11*, *18*, *21*, *24*, *29*, *38*, *40*
gsvaParam (gsvaParam-class), 18
gsvaParam-class, 18
gsvaRanks, *14*, 23
gsvaRanks,gsvaParam-method (gsvaRanks), 23
gsvaRanksParam, *14*, *21*, *24*, *32*
gsvaRanksParam-class (gsvaParam-class), 18
gsvaScores (gsvaRanks), 23
gsvaScores,gsvaRanksParam-method (gsvaRanks), 23
guessGeneIdType, *6, 7*, 25, *30*

HDF5Array, *16*

igsva, 26

list, *16*
loadHDF5GSVAranks (saveHDF5GSVAranks), 32
loadHDF5SummarizedExperiment, *32*

matrix, *16*, *23*, *38*

plageParam, *10, 11*, *18*, *23*, *29*, *38*, *40*
plageParam (plageParam-class), 28
plageParam-class, 28

readGMT, *4*, 30
readLines, *31*

saveHDF5GSVAranks, 32
saveHDF5SummarizedExperiment, *32*
SingleCellExperiment, *10, 11*, *16*, *23, 24*, *38*
spatCor, 33
spatCor,SpatialExperiment-method (spatCor), 33
SpatialExperiment, *10*, *16*
ssgseaParam, *10, 11*, *18*, *23*, *29*, *37*, *40*
ssgseaParam (ssgseaParam-class), 35
ssgseaParam-class, 35
SummarizedExperiment, *10*, *13*, *16*, *23*, *38*
SVT_SparseMatrix, *10*, *16*
Sys.meminfo(), *10*, *24*

zscoreParam, *10, 11*, *18*, *23*, *29*, *38–40*
zscoreParam (zscoreParam-class), 38
zscoreParam-class, 38