Package 'SVP'

November 12, 2026

Title Predicting cell states and their variability in single-cell or spatial omics data

Version 1.4.1

Description SVP uses the distance between cells and cells, features and features, cells and features in the space of MCA to build nearest neighbor graph, then uses random walk with restart algorithm to calculate the activity score of gene sets (such as cell marker genes, kegg pathway, go ontology, gene modules, transcription factor or miRNA target sets, reactome pathway, ...), which is then further weighted using the hypergeometric test results from the original expression matrix. To detect the spatially or single cell variable gene sets or (other features) and the spatial colocalization between the features accurately, SVP provides some global and local spatial autocorrelation method to identify the spatial variable features. SVP is developed based on SingleCellExperiment class, which can be interoperable with the existing computing ecosystem.

Depends R (>= 4.0)

Imports Rcpp, RcppParallel, methods, cli, dplyr, rlang, S4Vectors, SummarizedExperiment, SingleCellExperiment, SpatialExperiment, BiocGenerics, BiocParallel, fastmatch, pracma, stats, withr, Matrix, DelayedMatrixStats, deldir, utils, BiocNeighbors, ggplot2, ggstar, ggtree, ggfun

Suggests rmarkdown, prettydoc, broman, RSpectra, BiasedUrn, knitr, ks, igraph, testthat (>= 3.0.0), scuttle, magrittr, DropletUtils, tibble, tidyr, harmony, aplot, scales, ggsc, scatterpie, scran, scater, STexampleData, ape

License GPL-3

BugReports https://github.com/YuLab-SMU/SVP/issues

URL https://github.com/YuLab-SMU/SVP

Encoding UTF-8

Roxygen list(markdown = TRUE)

biocViews SingleCell, Software, Spatial, Transcriptomics, GeneTarget, GeneExpression, GeneSetEnrichment, Transcription, GO, KEGG

SystemRequirements GNU make

ByteCompile true

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo (>= 14.0), RcppParallel, RcppEigen, dqrng

Config/testthat/edition 3

LazyData false

Config/roxygen2/version 8.0.0

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

git_branch RELEASE_3_23

git_last_commit 6a11da4

git_last_commit_date 2026-08-18

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Shuangbin Xu [aut, cre] (ORCID: <https://orcid.org/0000-0003-3513-5362>), Guangchuang Yu [aut, ctb] (ORCID: <https://orcid.org/0000-0002-6485-8781>)

Maintainer Shuangbin Xu <xshuangbin@163.com>

Contents

SVP-package . . . 3
as_tbl_df . . . 4
cal_lisa_f1 . . . 5
CellCycle.Hs . . . 7
cluster.assign . . . 7
data_CancerSEA . . . 9
data_hpda_spe_cell_dec . . . 10
data_sceSubPbmc . . . 11
data_SenMayo . . . 11
extract_weight_adj . . . 12
fast_cor . . . 13
fscoreDfs . . . 14
gsvaExps . . . 16
LISAResult . . . 19
LISAsce . . . 20
mob_marker_genes . . . 21
mob_sce . . . 22
plot_heatmap_globalbv . . . 22
pred.cell.signature . . . 24
reexports . . . 25
runCORR . . . 26
runDetectMarker . . . 29
runDetectSVG . . . 30
runENCODE . . . 34

runGLOBALBV 35
runKldSVG 40
runLISA 44
runLOCALBV 48
runMCA 53
runSGSA 54
runWKDE 59
svDfs 62
SVP-accessors 63
SVPExperiment 64

Index **66**

`SVP-package` *SVP: Predicting cell states and their variability in single-cell or spatial omics data*

Description

SVP uses the distance between cells and cells, features and features, cells and features in the space of MCA to build nearest neighbor graph, then uses random walk with restart algorithm to calculate the activity score of gene sets (such as cell marker genes, kegg pathway, go ontology, gene modules, transcription factor or miRNA target sets, reactome pathway, ...), which is then further weighted using the hypergeometric test results from the original expression matrix. To detect the spatially or single cell variable gene sets or (other features) and the spatial colocalization between the features accurately, SVP provides some global and local spatial autocorrelation method to identify the spatial variable features. SVP is developed based on SingleCellExperiment class, which can be interoperable with the existing computing ecosystem.

Author(s)

Maintainer: Shuangbin Xu <xshuangbin@163.com> (ORCID)

Authors:

- Shuangbin Xu <xshuangbin@163.com> (ORCID)
- Guangchuang Yu <guangchuangyu@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://github.com/YuLab-SMU/SVP
- Report bugs at https://github.com/YuLab-SMU/SVP/issues

as_tbl_df *convert the square matrix to long tidy table*

Description

This function is designed to convert the output of runGLOBALBV, fast_cor or the matrix output of cor to long tidy table.

Usage

```
as_tbl_df(
  x,
  listn = NULL,
  diag = TRUE,
  rmrd = TRUE,
  flag.clust = FALSE,
  dist.method = "euclidean",
  hclust.method = "average"
)
```

Arguments

x	list or matrix object, which is the output of runGLOBALBV, fast_cor or the matrix output of cor.
listn	list object, which must have name, and the element must from the row names of x or x[[1]] (when x is a list) default is NULL.
diag	logical whether include the diagonal (only work when the cor matrix is square), default is TRUE.
rmrd	logical whether remove of redundancy when the correlation matrix is a square matrix, default is TRUE.
flag.clust	logical whether perform the hierarchical cluster analysis to obtain the label for visualization.
dist.method	the distance measure to be used, only work when flag.clust = TRUE. It must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
hclust.method	the agglomeration method to be used, only work with flag.clust=TRUE. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

Value

a long tidy table

Examples

```
library(ggplot2)
library(ggtree)
library(aplot)
example(fast_cor, echo=FALSE)
x <- as_tbl_df(res)
head(x)
xx <- as_tbl_df(res, flag.clust = TRUE,
                dist.method = 'euclidean', hclust.method = 'average')
p1 <- ggplot(xx, mapping = aes(x=x,y=y,color=r,size=abs(r))) +
      geom_point() + xlab(NULL) + ylab(NULL) +
      guides(y=guide_axis(position='right'))
p2 <- res$r |> dist() |> hclust(method = 'average') |>
      ggtree(layout='den', branch.length='none', ladderize=FALSE)
p3 <- res$r |> t() |> dist() |> hclust(method = 'average') |>
      ggtree(branch.length = 'none', ladderize = FALSE)
p4 <- p1 |> insert_left(p3, width=.12) |> insert_top(p2, height=.12)
aplot::plot_list(p1, p4)
x2 <- as_tbl_df(res2)
head(x2)
f1 <- ggplot(x2, aes(x=x, y=y, color=r, size=abs(r))) + geom_point() +
      xlab(NULL) + ylab(NULL) +
      guides(x=guide_axis(position='top', angle=45),
             y=guide_axis(position='right'))
f2 <- res2$r |> t() |> dist() |> hclust(method = 'average') |>
      ggtree(branch.length = 'none', ladderize=FALSE)
f3 <- f1 |> aplot::insert_left(f2, width=.12)
xx2 <- as_tbl_df(res2,
                flag.clust = TRUE,
                dist.method = 'euclidean',
                hclust.method = 'average'
      )
ff1 <- ggplot(xx2, mapping = aes(x=x,y=y, color=r,size=abs(r))) +
       geom_point() + xlab(NULL) + ylab(NULL) +
       guides(x=guide_axis(position='top', angle=45),
              y=guide_axis(position='right'))
ff3 <- ff1 |> aplot::insert_left(f2, width = .12)
aplot::plot_list(f3, ff3)
```

`cal_lisa_f1` *calculate the F1 value based on LISA result in the specified category.*

Description

this is to calculate the F1 value based on LISA result in some spatial domain. If a feature has a larger F1 value in a spatial domain, it means the feature is more concentrated in that spatial domain (specified category).

Usage

```
cal_lisa_f1(data, lisa.res, type = "High", group.by, rm.group.nm = NULL, ...)

## S4 method for signature 'SingleCellExperiment'
cal_lisa_f1(data, lisa.res, type = "High", group.by, rm.group.nm = NULL, ...)
```

Arguments

data	a SingleCellExperiment object
lisa.res	list the result of runLISA or runLOCALBV.
type	character the type of cluster.test column of result of runLISA or runLOCALBV, default is 'High'.
group.by	character a specified category column names (for example the cluster column name) of colData(data). Or a vector of length equal to ncol(data), specifying the group to which each cell is assigned. It is required.
rm.group.nm	character which want to remove some group type names from the names of the specified category group, default is NULL.
...	currently meaningless.

Value

a data.frame object containing the F1 value for each category in group.by.

Examples

```
data(hpda_spe_cell_dec)
lisa.res1 <- hpda_spe_cell_dec |>
   runLISA(
     features = rownames(hpda_spe_cell_dec),
     assay.type = 1
   )
res <- cal_lisa_f1(hpda_spe_cell_dec, lisa.res1, type='High', group.by = 'cluster_domain')
head(res)
# group.by, a vector of length equal to the ncol(data).
res2 <- cal_lisa_f1(hpda_spe_cell_dec,
                    lisa.res1,
                    type='High',
                    group.by = hpda_spe_cell_dec$cluster_domain
  )
identical(res, res2)
```

CellCycle.Hs *the Cell Cycle gene set*

Description

the S and G2M gene list are from the Seurat which refer to this article (doi:10.1126/science.aad050), the G1 gene list is from the G1_PHASE of Human Gene Set in MSigDB, but remove the duplicated records with S and G2M gene list.

Format

list

Value

a list object

Examples

```
data(CellCycle.Hs)
```

cluster.assign *clusting and assign the label for each feature(specify the gene sets).*

Description

clusting and assign the label for each feature(specify the gene sets).

Usage

```
cluster.assign(
  data,
  assay.type = "affi.score",
  assign = FALSE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
cluster.assign(
  data,
  assay.type = "affi.score",
  assign = FALSE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
```

```
  ...
)

## S4 method for signature 'SVPExperiment'
cluster.assign(
  data,
  assay.type = "affi.score",
  assign = FALSE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)
```

Arguments

`data`	A SVPExperiment, which has run `runSGSA` or `detect.svp`, or a SingleCellExperiment which was extracted from SVPExperiment using `gsvaExp` function.
`assay.type`	which expressed data to be pulled to run, default is `affi.score`.
`assign`	whether assign the max affinity of gene set or pathway to the each cell, default is FALSE.
`gsvaexp`	which gene set variation experiment will be pulled to run, this only work when `data` is a SVPExperiment, default is NULL.
`gsvaexp.assay.type`	which assay data in the specified `gsvaexp` will be used to run, default is NULL.
`...`	dot parameters

Details

when use `runSGSA` to calculated the gene set activity of cell, if `assign = TRUE` we will assign the max affinity of gene set or pathway to the each cell. If `assign = FALSE`, the max affinity of gene set or pathway will be kept.

Value

if input is a SVPExperiment, output will be also a SVPExperiment, and the result assay was stored in assay of the specified `gsvaexp`, which is a SingleCellExperiment. If input is a SingleCellExperiment (which is extracted from SVPExperiment using `gsvaExp()` function), output will be a SingleCellExperiment, the result can be extracted using `assay()` function.

See Also

to calculate the activity score of gene sets or pathway: `runSGSA`.

Examples

```
library(SpatialExperiment)
# This example data was extracted from the
# result of runSGSA with gsvaExp function.
data(hpda_spe_cell_dec)
```

```
assays(hpda_spe_cell_dec)
hpda_spe_cell_dec <- hpda_spe_cell_dec |>
   cluster.assign()
hpda_spe_cell_dec
```

`data_CancerSEA` *The Gene List of Cancer Single-cell State Atlas (CancerSEA)*

Description

CancerSEA is the first dedicated database that aims to comprehensively decode distinct functional states of cancer cells at single-cell resolution. CancerSEASymbol is a gene symbol list, and CancerSEAEnsemble is a Ensemble gene list, they are a list contained gene signature names collected in the database.

Format

list a gene symbol list with gene signature names collected in CancerSEA:

Angiogenesis Angiogenesis ensures that cancer cells receive continuous supplies of oxygen and other nutrients.

Apoptosis The inactivation of apoptosis in cancer cells lead to the persistence of such grossly abnormal cells in the tissues.

Cell Cycle Cell cycle,a critical process to ensure correct cell division,lies at the heart of cancer.

Differentiation The degree of cell differentiation can be used to measure the progress of cancer,and dedifferentiated cells can lead to the formation of cancer.

DNA damage DNA damage is an alteration in the chemical structure of DNA, and un-repaired DNA damages accumulate in replicating cells possibly contribute to progression to cancer.

DNA repair DNA repair plays a fundamental role in the maintenance of genomic integrity,it's deficits may lead to carcinogenesis.

EMT EMT has been indicated to be involved in the initiation of metastasis in cancer progression and in acquiring drug resistance.

Hypoxia Tumor-hypoxia contributes to cell mobility,metastasis and therapy resistance.

Inflammation Chronic inflammation can cause about 15% to 25% of human cancers.

Invasion Invasion is a critical carcinogenic event in which cancer cells escape from their primary sites and spread to blood or lymphatic vessels.

Metastasis Metastasis promotes the malignant transformation of cancer and causes most cancer deaths.

Proliferation Proliferation,as one of the cancer hallmarks,is responsible for tumor progression.

Quiescence Quiescent cancer cells are resistant to chemotherapy.

Stemness Cancer cells with high stemness fuel the growth of cancer.

list

Value

a list object

Source

http://biocc.hrbmu.edu.cn/CancerSEA/goDownload

References

Yuan, H., Yan, M., Zhang, G., Liu, W., Deng, C., Liao, G., Xu, L., Luo, T., Yan, H., Long, Z., Shi, A., Zhao, T., Xiao, Y., & Li, X. (2019). CancerSEA: a cancer single-cell state atlas. Nucleic acids research, 47(D1), D900–D908. https://doi.org/10.1093/nar/gky939

Examples

```
data(CancerSEASymbol)
data(CancerSEAEnsemble)
```

---

data_hpda_spe_cell_dec

an example of result of runSGSA by extracting with gsvaExp

Description

The result of runSGSA with PDAC A sample from (doi:10.1038/s41587-019-0392-8)

Format

S4 class:SpatialExperiment

Value

a SpatialExperiment object

Examples

```
data(hpda_spe_cell_dec)
```

data_sceSubPbmc *a subset data of pbmck3 from SeuratData*

Description

a small SingleCellExperiment data set from pbmck3 which contains 1304 genes and 800 cells (extract randomly)

Format

S4 class:SingleCellExperiment

Value

a SingleCellExperiment object

Examples

```
data(sceSubPbmc)
```

data_SenMayo *A gene set identifies senescent cells and predicts senescence-associated pathways across tissues*

Description

SenMayoSymbol is a gene symbol list that can be used to identify senescent cells and predicts senescence-associated pathways across tissues

Format

list

Value

a list object

Source

https://static-content.springer.com/esm/art%3A10.1038%2Fs41467-022-32552-1/MediaObjects/41467_2022_32552_MOESM4_ESM.xlsx

References

Saul, D., Kosinsky, R.L., Atkinson, E.J. et al. A new gene set identifies senescent cells and predicts senescence-associated pathways across tissues. Nat Commun 13, 4827 (2022). https://doi.org/10.1038/s41467-022-32552-1

Examples

```
data(SenMayoSymbol)
```

`extract_weight_adj` *extract the cell adjacent matrix from spatial space or reduction space*

Description

the function provides `voronoi` or `knn` method to build the cell adjacent matrix.

Usage

```
extract_weight_adj(
  data,
  sample_id = "all",
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  cells = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
extract_weight_adj(
  data,
  sample_id = "all",
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  cells = NULL,
  ...
)
```

Arguments

`data`	a SingleCellExperiment object with contains `UMAP` or `TSNE`, or a SpatialExperiment object.
`sample_id`	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be `all` to compute metric for all samples; the matrix is computed separately for each sample. default is `"all"`.
`weight.method`	character the method to build the spatial neighbours weights, default is `voronoi` (Voronoi tessellation). Other method, which requires coord matrix as input and returns `nb`, `listw` or `Graph` object, also is available, such as `"knearneigh"`, `'dnearneigh'`, `"gabrielneigh"`, `"relativeneigh"`, which are from `spdep` package. default is `knn`, if it is `"none"`, meaning the distance weight of each spot is used to the weight.

reduction.used	character used as spatial coordinates to calculate the neighbours weights, default is NULL, the result of reduction can be specified, such as UMAP, TSNE, PCA. If it is specified, the weight neighbours matrix will be calculated using the result of specified reduction.
group.by	character a specified category column names (for example the cluster column name) of colData(data), if it was specified, the adjacency weighted matrix will be built based on the principle that spots or cells in the same category are adjacent, default is NULL.
cells	the cell name or index of data object, default is NULL.
...	additional parameters, when weight.method='knn', you can specified k=10.

Value

a dgCMatrix object

Examples

```
data(hpda_spe_cell_dec)
# knn method
wm <- extract_weight_adj(hpda_spe_cell_dec, weight.method='knn', k=7)
# voronoi method
wm <- extract_weight_adj(hpda_spe_cell_dec)
# specified group.by
wm <- extract_weight_adj(hpda_spe_cell_dec, group.by='cluster_domain')
```

---

fast_cor *Calculation of correlations and associated p-values*

Description

Calculation of correlations and associated p-values

Usage

```
fast_cor(
  x,
  y = NULL,
  combine = FALSE,
  method = c("pearson", "spearman", "bicorr"),
  alternative = c("two.sided", "less", "greater"),
  add.pvalue = FALSE,
  p.adjust.method = "BH"
)
```

Arguments

x	sparse Matrix which rows are the features and columns are the samples.
y	sparse Matrix which has the same column length of x, default is NULL.
combine	logical whether combine the correlation of x and y if y is provided, default is FALSE.
method	a character string indicating which correlation coefficient, One of "pearson" (default), "spearman" and "bicorr".
alternative	indicates the alternative hypothesis and must be one of the initial letter "two.sided", "less" and "greater". "greater" corresponds to positive association, "less" to negative association, default is "two.sided".
add.pvalue	logical whether calculate the pvalue of correlation using t test, default is FALSE.
p.adjust.method	a character string specifying the method to adjust pvalue.

Value

a list containing the matrix of correlation and matrix of pvalue (if add.pvalue is FALSE (default), the matrix of pvalue will be NULL).

Examples

```
set.seed(123)
x <- matrix(rnorm(500), ncol=10)
rownames(x) <- paste0('row', seq(nrow(x)))
colnames(x) <- paste0('col', seq(ncol(x)))
x <- Matrix::Matrix(x, sparse = TRUE)
x1 <- x[seq(10),]
x2 <- x[seq(11, 50),]
res <- fast_cor(x = x1, y = x2, combine = FALSE)
res$r |> dim()
res2 <- fast_cor(x = x1, y = x2, combine = TRUE)
res2$r |> dim()
```

---

fscoreDfs *features score matrix extract method*

Description

In some experiment, to calculated the contribution value of original features (such as genes) in the new features (gene sets), if the result is stored with the original object, which will simplify book-keeping in long workflows and ensure that samples remain synchronised.

Value

see Getter and setter

Getters

In the following examples, x is a SingleCellExperiment object.

`fscoreDf(x, type)`: Retrieves a DataFrame containing the new features (gene sets) (rows) for the specified `type`. `type` should either be a string specifying the name of the features scores matrix in `x` to retrieve, or a numeric scalar specifying the index of the desired matrix, defaulting to the first matrix is missing.

`fscoreDfNames(x)`: Retures a character vector containing the names of all features scores DataFrame Lists in `x`. This is guaranteed to be of the same length as the number of results.

`fscoreDfs(x)`: Returns a named List of matrices containing one or more DataFrame objects. Each object is guaranteed to have the same number of rows, in a 1:1 correspondence to those in `x`.

Single-object setter

`fscoreDf(x, type) <- value` will add or replace an features scores matrix in a SingleCellExperiment object `x`. The value of `type` determines how the result is added or replaced:

- If `type` is missing, `value` is assigned to the first result. If the result already exists, its name is preserved; otherwise it is given a default name `"unnamed.fscore1"`.
- If `type` is a numeric scalar, it must be within the range of existing results, and `value` will be assigned to the result at that index.
- If `type` is a string and a result exists with this name, `value` is assigned to to that result. Otherwise a new result with this name is append to the existing list of results.

Other setter

`fscoreDfs(x) <- value`: Replaces all features score matrices in `x` with those in `value`. The latter should be a list-like object containing any number of DataFrame objects with number of row equal to `nrow(x)`.

If `value` is named, those names will be used to name the features score matrices in `x`. Otherwise, unnamed results are assigned default names prefixed with `"unnamed.fscore"`.

If `value` is `NULL`, all features score matrices in `x` are removed.

`fscoreDfNames(x) <- value`: Replaces all names for features score matrices in `x` with a character vector `value`. This should be of length equal to the number of results currently in `x`.

Examples

```
# Using the class example
example(SVPExperiment, echo = FALSE)
dim(counts(svpe))
rownames(svpe) <- paste0("gene", seq(nrow(svpe)))
colnames(svpe) <- paste0("cell", seq(ncol(svpe)))
# Mocking up some GSVA Experiments
sce1 <- SingleCellExperiment(matrix(rpois(1000, 5), ncol=ncol(svpe)))
rownames(sce1) <- paste0("GO:",seq(nrow(sce1)))
colnames(sce1) <- colnames(svpe)
sce2 <- SingleCellExperiment(matrix(rpois(1000, 5), ncol=ncol(svpe)))
rownames(sce2) <- paste0("KEGG:", seq(nrow(sce2)))
```

```
colnames(sce2) <- colnames(svpe)

# Mocking up some relationship score between new feature and gene
fscore1 <- lapply(seq(nrow(sce1)), function(i) abs(rnorm(5, 0.5)) |>
             setNames(sample(rownames(svpe),5))) |>
             List() |>
             DataFrame() |> setNames("rwr_score")
rownames(fscore1) <- rownames(sce1)
fscore2 <- lapply(seq(nrow(sce2)), function(i) abs(rnorm(5, 0.8)) |>
             setNames(sample(rownames(svpe),5))) |>
             List() |>
             DataFrame() |> setNames("hyper_test")
# Setting the score
fscoreDfs(sce1) <- list()
fscoreDfs(sce2) <- list()
fscoreDf(sce1, "rwr_score") <- fscore1
fscoreDf(sce2, "hyper_test") <-fscore2

# Setting the GSVA Experiments
gsvaExp(svpe, "GO1") <- sce1
gsvaExp(svpe, "KEGG1") <- sce2

# Getting the GSVA Experiment data
fscoreDf(gsvaExp(svpe), "rwr_score")
fscoreDf(gsvaExp(svpe, 'KEGG1'), "hyper_test")
fscoreDf(gsvaExp(svpe, 'KEGG1'), 1)
fscoreDfNames(gsvaExp(svpe))
fscoreDfs(gsvaExp(svpe))

# Setting the names of features score DataFrame
fscoreDfNames(gsvaExp(svpe, withColData=FALSE)) <- "rwr.score"
fscoreDfNames(gsvaExp(svpe, withColData=FALSE))[1] <- "Test"
```

---

gsvaExps *Gene Set Variation Analysis Experiment methods*

Description

In some experiments, gene set variation analysis will generated different features (the names of KEGG pathway or the GO term). These data cannot be stored in the main `assays` of the SVPExperiment itself. However, it is still desirable to store these features *somewhere* in the SVPExperiment. This simplifies book-keeping in long workflows and ensure that samples remain synchronised.

To facilitate this, the SVPExperiment class allows for "gene set variation analysis experiments". Nested SingleCellExperiment-class objects are stored inside the SVPExperiment object `x`, in a manner that guarantees that the nested objects have the same columns in the same order as those in `x`. Methods are provided to enable convenient access to and manipulation of these gene set variation analysis Experiments. Each GSVA Experiment should contain experimental data and row metadata for a distinct set of features. (These methods refer to the `altExp` of `SingleCellExperiment`).

Value

see `Getter` and `setter`.

Getters

In the following examples, `x` is a SVPExperiment object.

`gsvaExp(x, e, withDimnames=TRUE, withColData=TRUE, withSpatialCoords = TRUE, withImgData=TRUE, withRedu`
: Retrieves a SingleCellExperiment containing gene set name features (rows) for all cells (columns) in `x`. `e` should either be a string specifying the name of the gene set variation Experiment in `x` to retrieve, or a numeric scalar specifying the index of the desired Experiment, defaulting to the first Experiment is missing.

  `withDimnames=TRUE`, the column names of the output object are set to `colnames(x)`. In addition, if `withColData=TRUE`, `colData(x)` is `cbind`ed to the front of the column data of the output object. `withSpatialCoords = TRUE`, the spatial coordinates of the output object are set to `spatialCoords(x)` if `x` has spatial coordinates. `withImgData=TRUE`, the image metadata of the output object are set to `imgData(x)` if `x` has image metadata. If `withReducedDim=TRUE`, the dimensionality reduction results of output object are set to `reducedDims(x)` if `x` has dimensionality reduction results

`gsvaExpNames(x)`**:** Returns a character vector containing the names of all gene set variation Experiments in `x`. This is guaranteed to be of the same length as the number of results, though the names may not be unique.

`gsvaExps(x, withDimnames=TRUE, withColData=TRUE, withSpatialCoords = TRUE, withImgData=TRUE, withReduce`
: Returns a named List of matrices containing one or more SingleCellExperiment objects. Each object is guaranteed to have the same number of columns, in a 1:1 correspondence to those in `x`.

  If `withDimnames=TRUE`, the column names of each output object are set to `colnames(x)`. In addition, if `withColData=TRUE`, `colData(x)` is `cbind`ed to the front of the column data of each output object. `withSpatialCoords = TRUE`, the spatial coordinates of the output object are set to `spatialCoords(x)` if `x` has spatial coordinates. `withImgData=TRUE`, the image metadata of the output object are set to `imgData(x)` if `x` has image metadata. If `withReducedDim=TRUE`, the dimensionality reduction results of output object are set to `reducedDims(x)` if `x` has dimensionality reduction results

Single-object setter

`gsvaExp(x, e, withDimnames=TRUE, withColData=FALSE, withSpatialCoords = FALSE, withImgData = FALSE, withReducedDim = FALSE) <- value` will add or replace an gene set variation Experiment in a SVPExperiment object `x`. The value of `e` determines how the result is added or replaced:

- If `e` is missing, `value` is assigned to the first result. If the result already exists, its name is preserved; otherwise it is given a default name `"unnamed.gsva1"`.
- If `e` is a numeric scalar, it must be within the range of existing results, and `value` will be assigned to the result at that index.
- If `e` is a string and a result exists with this name, `value` is assigned to to that result. Otherwise a new result with this name is append to the existing list of results.

value is expected to be a SingleCellExperiment object with number of columns equal to ncol(x). Alternatively, if value is NULL, the gene set variation Experiment at e is removed from the object.

If withDimnames=TRUE, the column names of value are checked against those of x. A warning is raised if these are not identical, with the only exception being when value=NULL. This is inspired by the argument of the same name in assay<-.

If withColData=TRUE, we assume that the left-most columns of colData(value) are identical to colData(x). If so, these columns are removed, effectively reversing the withColData=TRUE setting for the gsvaExp getter. Otherwise, a warning is raised.

If withSpatialCoords = TRUE, the spatial coordinates will be kept in the value if it has, and will add or replace it in a SVPExperiment object x.

If withImgData = TRUE, the image metadata will be kept in the value if it has, and will add or replace it in a SVPExperiment object x.

If withReducedDim = TRUE, the dimensionality reduction results will be kept in the value if it has, and will add or replace it in a SVPExperiment object x.

Other setters

In the following examples, x is a SVPExperiment object.

gsvaExps(x, withDimnames=TRUE, withColData=FALSE, withSpatialCoords = FALSE, withImgData = FALSE, withRe
: Replaces all gene set variation Experiments in x with those in value. The latter should be a list-like object containing any number of SingleCellExperiment objects with number of columns equal to ncol(x).

  If value is named, those names will be used to name the gene set variant Experiments in x. Otherwise, unnamed results are assigned default names prefixed with "unnamed.gsva".

  If value is NULL, all gene set variation Experiments in x are removed.

  If value is a Annotated object, any metadata will be retained in gsvaExps(x). If value is a Vector object, any mcols will also be retained.

  If withDimnames=TRUE, the column names of each entry of value are checked against those of x. A warning is raised if these are not identical.

  If withColData=TRUE, we assume that the left-most columns of the colData for each entry of value are identical to colData(x). If so, these columns are removed, effectively reversing the withColData=TRUE setting for the gsvaExps getter. Otherwise, a warning is raised. If withSpatialCoords = TRUE, withImgData = TRUE, and withReducedDim = TRUE refer to the gsvaExp(...) <- value.

gsvaExpNames(x) <- value**:** Replaces all names for gene set variant Experiments in x with a character vector value. This should be of length equal to the number of results currently in x.

Main Gene Set Variation Experiment naming

The Gene Set Variation Experiments are naturally associated with names (e during assignment). However, we can also name the main Experiment in a SVPExperiment x:

mainGsvaExpName(x) <- value**:** Set the name of the main Experiment to a non-NA string value. This can also be used to unset the name if value=NULL.

mainGsvaExpName(x)**:** Returns a string containing the name of the main Experiment. This may also be NULL if no name is specified.

Examples

```
# Using the class example
example(SVPExperiment, echo = FALSE)
dim(counts(svpe))

# Mocking up some GSVA Experiments
sce1 <- SingleCellExperiment(matrix(rpois(1000, 5), ncol=ncol(svpe)))
rownames(sce1) <- paste0("GO:",seq(nrow(sce1)))
colnames(sce1) <- colnames(svpe)
sce2 <- SingleCellExperiment(matrix(rpois(1000, 5), ncol=ncol(svpe)))
rownames(sce2) <- paste0("KEGG:", seq(nrow(sce2)))
colnames(sce2) <- colnames(svpe)

# Setting the GSVA Experiments
gsvaExp(svpe, "GO") <- sce1
gsvaExp(svpe, "KEGG") <- sce2

# Getting the GSVA Experiment data
gsvaExp(svpe, "GO")
gsvaExp(svpe, "KEGG")
gsvaExp(svpe, 2)
gsvaExpNames(svpe)
gsvaExps(svpe)

# Setting the names of GSVA Experiments
gsvaExpNames(svpe) <- c("GO1", "KEGG1")
svpe
gsvaExpNames(svpe)[1] <- "Test"
```

`LISAResult` *LISAResult*

Description

Extracting the result of `runLISA()`

Usage

```
LISAResult(x, type = NULL, features = NULL, ...)
```

Arguments

`x`	object SingleCellExperiment.
`type`	character, the name of `method` parameter of `runLISA()` combining `.SVP`,so it can be one of `localG.SVP` or `localmoran.SVP`, default is NULL.
`features`	character or index which have been specified in `features` of `code{runLISA()}` and `action='add'`, default is NULL.
`...`	additional parameter, meaningless now.

Value

a data.frame or SimpleList.

Examples

```
data(hpda_spe_cell_dec)
hpda_spe_cell_dec <- hpda_spe_cell_dec |>
                     runLISA(features = 'Cancer clone A',
                             assay.type = 'affi.score',
                             method = 'localG',
                             action = 'add'
                     )
hpda_spe_cell_dec <- hpda_spe_cell_dec |>
                     runLISA(features = 'Cancer clone A',
                             assay.type = 'affi.score',
                             method = 'localmoran',
                             action = 'add'
                     )
local.G <- LISAResult(hpda_spe_cell_dec,
                      type='localG.SVP', features='Cancer clone A'
           )
localmoran <- LISAResult(hpda_spe_cell_dec,
                         type = 'logcalmoran.SVP',
                         features = 'Cancer clone A'
           )
hpda_spe_cell_dec |> LISAResult() |> head()
```

LISAsce *convert LISA result to SVPExperiment.*

Description

convert the Gi for runLISA result or LocalLee for runLOCALBV result to a SVPExperiment.

Usage

```
LISAsce(data, lisa.res, gsvaexp.name = "LISA", ...)

## S4 method for signature 'SingleCellExperiment'
LISAsce(data, lisa.res, gsvaexp.name = "LISA", ...)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object.
lisa.res	list the result of runLISA or runLOCALBV.
gsvaexp.name	character the name of gsveExp for the LISA result, default is "LISA".
...	currently meaningless.

Value

a SVPExperiment object

See Also

runLISA and runLOCALBV

Examples

```
data(hpda_spe_cell_dec)
lisa.res12 <- hpda_spe_cell_dec |>
   runLISA(
     features = c(1, 2, 3),
     assay.type = 'affi.score',
     weight.method = "knn",
     k = 10,
     action = 'get',
   )
hpda_spe_cell_dec <- LISAsce(hpda_spe_cell_dec, lisa.res12)
hpda_spe_cell_dec
gsvaExp(hpda_spe_cell_dec, 'LISA')
localbv.res1 <- hpda_spe_cell_dec |> runLOCALBV(
          features1 = 'Cancer clone A',
          features2 = 'Cancer clone B',
          assay.type='affi.score'
        )
hpda_spe_cell_dec <- LISAsce(hpda_spe_cell_dec, localbv.res1, 'LOCALBV')
gsvaExp(hpda_spe_cell_dec, 'LOCALBV')
```

---

mob_marker_genes *the marker genes of mouse olfactory bulb*

Description

this is extracted from the single cell transcriptome of a mouse olfactory bulb from (doi:10.1016/j.celrep.2018.11.034)

Format

list

Value

a list object with name

Examples

```
data(mob_marker_genes)
```

mob_sce *the single cell gene profiler of a mouse olfactory bulb*

Description

The single cell transcriptome of WT sample of mouse olfactory bulb from (doi:10.1016/j.celrep.2018.11.034)

Format

S4 class:SingleCellExperiment

Value

a SingleCellExperiment object

Examples

```
data(mob_sce)
```

plot_heatmap_globalbv *plot_heatmap_globalbv*

Description

visualize the result of global bivariate spatial analysis with heatmap

Usage

```
plot_heatmap_globalbv(
  globalbv,
  moran.t = NULL,
  moran.l = NULL,
  lisa.t = NULL,
  lisa.l = NULL,
  max.point.size = 4.5,
  font.size = 2.5,
  limits.size = NULL,
  limits.colour = NULL,
  dist.method = "euclidean",
  hclust.method = "average",
  threshold = 0.05
)
```

Arguments

globalbv	the result of runGLOBALBV with action = 'get'.
moran.t	the result of global spatial variable features for one type features default is NULL. or runDetectSVG and then using svDf to extract the result.
moran.l	the result of global spatial variable features for another type features default is NULL.
lisa.t	the result of cal_lisa_f1 for another type features.
lisa.l	the result of cal_lisa_f1 for one type features
max.point.size	the max point size for main dotplot, default is 4.5.
font.size	the size of font when the triangle heatmap is displayed, default is 2.5.
limits.size	adjust the limit of point size for main dotplot via limits of scale_size_continuous, default is NULL.
limits.colour	adjust the limit of point colour for main dotplot via limits of scale_fill_gradient2, default is NULL.
dist.method	the distance measure to be used for the result of global bivariate spatial. which is to measure the dissimilarity between the features, default is 'euclidean'.
hclust.method	the agglomeration method to be used for the result of global bivariate spatial. which is also to measure the similarity between the features, default is 'averate'.
threshold	numeric the threshold to display the point with the significance level, default is 0.05.

Value

a ggplot2 or aplot object

Examples

```
data(hpda_spe_cell_dec)
gbv.res <- runGLOBALBV(
              hpda_spe_cell_dec, features1=rownames(hpda_spe_cell_dec),
              assay.type=1, add.pvalue=TRUE, permutation=NULL, alternative='greater'
           )

moran.res <- runDetectSVG(hpda_spe_cell_dec, assay.type=1) |> svDf()

lisa.res <- runLISA(hpda_spe_cell_dec, features=rownames(hpda_spe_cell_dec), assay.type=1)

lisa.f1 <- cal_lisa_f1(hpda_spe_cell_dec, lisa.res, group.by='cluster_domain')

plot_heatmap_globalbv(gbv.res, moran.t=moran.res, lisa.t=lisa.f1)
```

`pred.cell.signature` *predict the cell signature according the gene sets or pathway activity score.*

Description

predict the cell signature according the gene sets or pathway activity score.

Usage

```
pred.cell.signature(
  data,
  assay.type = "affi.score",
  threshold = NULL,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  pred.col.name = "pred.cell.sign",
  ...
)

## S4 method for signature 'SingleCellExperiment'
pred.cell.signature(
  data,
  assay.type = "affi.score",
  threshold = NULL,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  pred.col.name = "pred.cell.sign",
  ...
)

## S4 method for signature 'SVPExperiment'
pred.cell.signature(
  data,
  assay.type = "affi.score",
  threshold = NULL,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  pred.col.name = "pred.cell.sign",
  ...
)
```

Arguments

`data`	A SVPExperiment, which has run `runSGSA` or `detect.svp`, or a SingleCellExperiment which was extracted from SVPExperiment using `gsvaExp` function.
`assay.type`	which expressed data to be pulled to run, default is `affi.score`.

threshold	numeric when the gene set activity score of cell less than the threshold, the cell signature will be consider as 'unassigned', default is NULL, meaning will be calculated internally.
gsvaexp	which gene set variation experiment will be pulled to run, this only work when data is a SVPExperiment, default is NULL.
gsvaexp.assay.type	which assay data in the specified gsvaexp will be used to run, default is NULL.
pred.col.name	character the column name in colData of the result, default is pred.cell.sign.
...	dot parameters

Value

if input is a SVPExperiment, output will be also a SVPExperiment, and the result was stored at the pred.col.name column of colData in the specified gsvaexp, which is a SingleCellExperiment. If input is a SingleCellExperiment (which is extracted from SVPExperiment using gsvaExp() function), output will be a SingleCellExperiment, the result can be extracted using colData() function with specified column in default is pred.cell.sign.

See Also

to calculate the activity score of gene sets or pathway: runSGSA, to keep the max gene set or pathway activity score of cell: cluster.assign.

Examples

```
data(hpda_spe_cell_dec)
hpda_spe_cell_dec <- hpda_spe_cell_dec |>
  pred.cell.signature(assay.type = 1)
hpda_spe_cell_dec$pred.cell.sign |> table()
#\donttest{
  library(ggsc)
  library(ggplot2)
  hpda_spe_cell_dec |>
    sc_spatial(
      mapping = aes(x, y, colour = pred.cell.sign),
      geom = geom_bgpoint,
      pointsize = 2
    )
#}
```

---

reexports *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

SpatialExperiment imgData(), imgData<-(), spatialCoords(), spatialCoords<-(), spatialCoordsNames(), spatialCoordsNames<-()

Value

function

---

runCORR *runCORR*

Description

This function to perform the correlation of the features in main experiment or features of gsva experiment.

Usage

```
runCORR(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  method = c("spearman", "pearson", "bicorr"),
  alternative = c("greater", "two.sided", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runCORR(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  method = c("spearman", "pearson", "bicorr"),
  alternative = c("greater", "two.sided", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
```

```
  ...
)

## S4 method for signature 'SVPExperiment'
runCORR(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  method = c("spearman", "pearson", "bicorr"),
  alternative = c("greater", "two.sided", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
features1	the features name data object (only supporting character), default is NULL, see also features2 parameter.
features2	character, if features1 is not NULL, and features2 is NULL, only the features1 are analyzed, if features1 is NULL, and features2 is not NULL, the features2 are analyzed, if features2 is also NULL, all of features in the data object will be analyzed. If features2 and features1 are not NULL, the bivariate spatial autocorrelation analysis will be performed between the features1 and features2. default is NULL.
assay.type	which expressed data to be pulled to run, default is logcounts.
method	character should be one of the spearman, pearson and bicorr, default is 'spearman'.
alternative	indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less". You can specify just the initial letter. "greater" corresponds to positive association, "less" to negative association, default is "two.sided".
add.pvalue	logical whether calculate the pvalue, which is calculated with permutation test. So it might be slow, default is FALSE, which the pvalue of result will be NULL.
p.adjust.method	character the method to adjust the pvalue of the result, default is BH.
action	character, which should be one of 'only' and 'get', default is "get". If action='only', it will return a long tidy table contains the correlation for each feature pairs. If action='get', it will return a list containing the correlation matrix and pvalue matrix (if add.pvalue=TRUE).

verbose — logical whether print the help information, default is TRUE.

gsvaexp — character the one character from the name of gsvaExpNames(data), default is NULL. If data is SVPExperiment, and the parameter is specified simultaneously. the features (Usually genes) from the displayed class, and gsvaexp.features from name in rownames(gsvaExp(data, gsvaexp)) will be performed the analysis.

gsvaexp.assay.type — character the assay name in the assays(gsvaExp(data, gsvaexp)), default is NULL, which works with gsvaexp parameter.

gsvaexp.features — character the name from the rownames(gsvaExp(data, gsvaexp)). If gsvaexp is specified and data is SVPExperiment, it should be provided. Default is NULL.

across.gsvaexp — logical whether only calculate the relationship of features between the multiple gsvaExps not the internal features of gsvaExp. For example, 'a' and 'b' features are from the 'AB' gsvaExp, 'c' and 'd' features are from the 'CD' gsvaExp. When across.gsvaexp=TRUE and gsvaexp.features = c('a', 'b', 'c', 'd') and gsvaexp = c('AB', 'CD'), Only the relationship of a and c, a and d, b and c, and b and d will be calculated. default is TRUE.

... — additional parameters the parameters which are from the weight.method function.

Value

long tidy table or list see also the help information of action argument.

Author(s)

Shuangbin Xu

See Also

runCORR to explore the global bivariate relationship in the spatial space.

Examples

```
data(hpda_spe_cell_dec)
rownames(hpda_spe_cell_dec) |> head()
res1 <- runCORR(hpda_spe_cell_dec,
                features1 = "Ductal APOL1 high-hypoxic",
                features2 = c('Cancer clone A', "Cancer clone B"),
                assay.type = 'affi.score',
                action='only'
        )
res1
res2 <- runCORR(hpda_spe_cell_dec,
                features1 = c("Acinar cells",
                              "Ductal APOL1 high-hypoxic",
                              "Cancer clone A",
```

```
                              "Cancer clone B"),
                assay.type = 1,
                action = 'get'
        )
res2
```

---

runDetectMarker *Detecting the specific cell features with nearest distance of cells in MCA space*

Description

Detecting the specific cell features with nearest distance of cells in MCA space

Usage

```
runDetectMarker(
  data,
  group.by,
  aggregate.group = TRUE,
  reduction = "MCA",
  dims = 30,
  ntop = 200,
  present.prop.in.group = 0.1,
  present.prop.in.sample = 0.2,
  BPPARAM = SerialParam(),
  ...
)

## S4 method for signature 'SingleCellExperiment'
runDetectMarker(
  data,
  group.by,
  aggregate.group = TRUE,
  reduction = "MCA",
  dims = 30,
  ntop = 200,
  present.prop.in.group = 0.1,
  present.prop.in.sample = 0.2,
  BPPARAM = SerialParam(),
  ...
)
```

Arguments

data	SingleCellExperiment object
group.by	the column name of cell annotation. Or a vector of length equal to `ncol(data)`, specifying the group to which each cell is assigned. It is required.

aggregate.group
: logical whether calculate the center cluster of each group of cell according to the group.by, then find the nearest features of the center cluster, default TRUE. If FALSE, meaning the nearest features to each cell are detected firstly.

reduction
: character which reduction space, default is 'MCA'.

dims
: integer the number of components to defined the nearest distance.

ntop
: integer the top number of nearest or furthest (type = 'negative') features, default is 200.

present.prop.in.group
: numeric the appearance proportion of groups which have the marker default is .1, smaller value represent the marker will have higher specificity, but the number of marker for each group might also decrease, the minimum value is 1/length(unique(data[[group.by]])).

present.prop.in.sample
: numeric the appearance proportion of samples which have the marker in the corresponding group by specific group.by, default is 0.2.

BPPARAM
: A BiocParallelParam object specifying whether perform the analysis in parallel using BiocParallel default is SerialParam(), meaning no parallel. You can use BiocParallel::MulticoreParam(workers=4, progressbar=TRUE) to parallel it, the workers of MulticoreParam is the number of cores used, see also MulticoreParam. default is SerialParam().

...
: additional parameters.

Value

a list, which contains features and named with clusters of group.by.

Examples

```
# The example data (small.sce) is generated through simulation and has no actual meaning.
set.seed(123)
example(runMCA, echo = FALSE)
small.sce |> runDetectMarker(group.by = 'Cell_Cycle', ntop = 20,
              present.prop.in.sample = .2)
# group.by, a vector of length equal to ncol(small.sce)
small.sce |> runDetectMarker(
               group.by = small.sce$Cell_Cycle,
               ntop = 20,
               present.prop.in.sample = .2
             )
```

---

runDetectSVG *Detecting the spatially or single cell variable features with Moran's I or Geary's C*

Description

This function use Moran's I, Geary's C or global G test to detect the signal genes in a low-dimensional space (UMAP or TSNE for single cell omics data) or a physical space (for spatial omics data).

Usage

```
runDetectSVG(
  data,
  assay.type = "logcounts",
  method = c("moransi", "gearysc", "getisord"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  sample_id = "all",
  reduction.used = NULL,
  group.by = NULL,
  permutation = NULL,
  p.adjust.method = "BH",
  random.seed = 1024,
  verbose = TRUE,
  action = c("add", "only", "get"),
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runDetectSVG(
  data,
  assay.type = "logcounts",
  method = c("moransi", "gearysc", "getisord"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  sample_id = "all",
  reduction.used = NULL,
  group.by = NULL,
  permutation = NULL,
  p.adjust.method = "BH",
  random.seed = 1024,
  verbose = TRUE,
  action = c("add", "only", "get"),
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SVPExperiment'
runDetectSVG(
```

```
  data,
  assay.type = "logcounts",
  method = c("moransi", "gearysc", "getisord"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  sample_id = "all",
  reduction.used = NULL,
  group.by = NULL,
  permutation = NULL,
  p.adjust.method = "BH",
  random.seed = 1024,
  verbose = TRUE,
  action = c("add", "only", "get"),
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)
```

Arguments

`data`	a SingleCellExperiment object with contains `UMAP` or `TSNE`, or a SpatialExperiment object, or a SVPExperiment object with specified `gsvaexp` argument.
`assay.type`	which expressed data to be pulled to run, default is `logcounts`.
`method`	character the method of spatial autocorrelation using a spatial weights to detect spatial variable features, one of `'moransi'`, `"gearysc"` or `"getisord"`, default is `'moransi'`.
`weight`	object, which can be `nb`, `listw` or `Graph` object, default is NULL, meaning the spatial neighbours weights will be calculated using the `weight.method`. if the `data` contains multiple samples, and the `sample_id` is specified, it should be provided as a list object with names (using `sample_id`).
`weight.method`	character the method to build the spatial neighbours weights, default is `voronoi` (Voronoi tessellation). Other method, which requires coord matrix as input and returns `nb`, `listw` or `Graph` object, also is available, such as `"knearneigh"`, `'dnearneigh'`, `"gabrielneigh"`, `"relativeneigh"`, which are from `spdep` package. default is `knn`, if it is `"none"`, meaning the distance weight of each spot is used to the weight.
`sample_id`	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be `all` to compute metric for all samples; the metric is computed separately for each sample. default is `"all"`.
`reduction.used`	character used as spatial coordinates to detect SVG, default is `UMAP`, if `data` has `spatialCoords`, which will be used as spatial coordinates.
`group.by`	character a specified category column names (for example the cluster column name) of `colData(data)`. Or a vector of length equal to `ncol(data)`, specifying the group to which each cell is assigned. If it was specified, the adjacency weighted matrix will be built based on the principle that spots or cells in the same category are adjacent, default is NULL.

permutation	integer the number to permutation test for the calculation of Moran's I, default is NULL. We do not recommend using this parameter, as the permutation test is too slow.
p.adjust.method	character the method to adjust the pvalue of the result, default is BH.
random.seed	numeric random seed number to repeatability, default is 1024.
verbose	logical whether print the intermediate message when running the program, default is TRUE.
action	character control the type of output, if action='add', the result of identification will add the original object, if action = 'get', the result will return a SimpleList, if action = 'only', the result will return a DataFrame by merging the result of all sample, default is add.
gsvaexp	which gene set variation experiment will be pulled to run, this only work when data is a SVPExperiment, default is NULL.
gsvaexp.assay.type	which assay data in the specified gsvaexp will be used to run, default is NULL.
...	additional parameters

Value

a SVPExperiment or a SingleCellExperiment, see action parameter details.

Author(s)

Shuangbin Xu

References

1. P. A. P. Moran, The Interpretation of Statistical Maps, Journal of the Royal Statistical Society: Series B (Methodological), Volume 10, Issue 2, July 1948, Pages 243–251, https://doi.org/10.1111/j.2517-6161.1948.tb00012.x
2. R. C. Geary, The Contiguity Ratio and Statistical Mapping, Journal of the Royal Statistical Society Series D: The Statistician, Volume 5, Issue 3, November 1954, Pages 115–141, https://doi.org/10.2307/2986645
3. Cli AD, Ord JK (1981) Spatial processes: models & applications. Pion Limited, London
4. Bivand, R.S., Wong, D.W.S. Comparing implementations of global and local indicators of spatial association. TEST 27, 716–748 (2018). https://doi.org/10.1007/s11749-018-0599-x

See Also

runLISA to explore the hotspot for specified features in the spatial space.

Examples

```
# This example dataset is extracted from the
# result of runSGSA with gsvaExp(svpe).
data(hpda_spe_cell_dec)

# using Moran's I test
######################
hpda_spe_cell_dec <-
    hpda_spe_cell_dec |>
    runDetectSVG(
      assay.type = 'affi.score',
      method = 'moransi'
    )
# The result also is saved in the svDfs in the SVPExample object
# which can be extrated with svDf
svDfs(hpda_spe_cell_dec)

hpda_spe_cell_dec |> svDf("sv.moransi") |> data.frame() |> dplyr::arrange(rank)

# using Geary's C test
######################
hpda_spe_cell_dec <-
    hpda_spe_cell_dec |>
    runDetectSVG(assay.type ='affi.score', method = 'gearysc')

svDfs(hpda_spe_cell_dec)

hpda_spe_cell_dec |> svDf("sv.gearysc") |> data.frame() |> dplyr::arrange(rank)

# using Global G test (Getis-Ord)
#################################
hpda_spe_cell_dec <- hpda_spe_cell_dec |>
    runDetectSVG(assay.type = 1, method = 'getisord')

svDfs(hpda_spe_cell_dec)

hpda_spe_cell_dec |> svDf(3) |> data.frame() |> dplyr::arrange(rank)
```

---

runENCODE *One hot encode for the specified cell category.*

Description

This function convert the specified cell category to one hot encode

Usage

```
runENCODE(data, group.by, rm.group.nm = NULL, ...)

## S4 method for signature 'SingleCellExperiment'
runENCODE(data, group.by, rm.group.nm = NULL, ...)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
group.by	character a specified category column names (for example the cluster column name) of colData(data). Or a vector of length equal to 'ncol(data)', specifying the group to which each cell is assigned. It is required.
rm.group.nm	character which want to remove some group type names from the names of the specified category group, default is NULL.
...	currently meaningless.

Value

SVPExperiment object

Examples

```
data(sceSubPbmc)
sceSubPbmc
sceSubPbmc <- runENCODE(sceSubPbmc, group.by = 'seurat_annotations')
sceSubPbmc
gsvaExp(sceSubPbmc, 'seurat_annotations')
sceSubPbmc <- runENCODE(sceSubPbmc, group.by = 'seurat_annotations', rm.group.nm = c('Platelet'))
sceSubPbmc
gsvaExp(sceSubPbmc, 'seurat_annotations')
# The group.by also can be a vector of length equal to ncol(data).
sceSubPbmc <- runENCODE(
                sceSubPbmc,
                group.by = sceSubPbmc$seurat_annotations,
                rm.group.nm = c('Platelet')
              )
sceSubPbmc
identical(gsvaExp(sceSubPbmc, 'seurat_annotations'), gsvaExp(sceSubPbmc, "ENCODE"))
```

---

runGLOBALBV *Global Bivariate analysis for spatial autocorrelation*

Description

This function is to explore the global bivariate relationship in the spatial space. It efficiently reflects the extent to which bivariate associations are spatially grouped. Put differently, it can be utilized to quantify the bivariate spatial dependency. See also the references.

Usage

```
runGLOBALBV(
  data,
  features1 = NULL,
```

```
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("lee"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  alternative = c("two.sided", "greater", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  random.seed = 1024,
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runGLOBALBV(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("lee"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  alternative = c("two.sided", "greater", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  random.seed = 1024,
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)
```

```
## S4 method for signature 'SVPExperiment'
runGLOBALBV(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("lee"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  alternative = c("two.sided", "greater", "less"),
  add.pvalue = FALSE,
  p.adjust.method = "BH",
  random.seed = 1024,
  action = c("get", "only"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
features1	the features name data object (only supporting character), default is NULL, see also features2 parameter.
features2	character, if features1 is not NULL, and features2 is NULL, only the features1 are analyzed, if features1 is NULL, and features2 is is not NULL, the features2 are analyzed, if features2 is also NULL, all of features in the data object will be analyzed. If features2 and features1 are not NULL, the bivariate spatial autocorrelation analysis will be performed between the features1 and features2. default is NULL.
assay.type	which expressed data to be pulled to run, default is logcounts.
sample_id	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be all to compute metric for all samples; the metric is computed separately for each sample. default is "all".
method	character now only the 'lee', default is 'lee'.
weight	object, which can be nb, listw or Graph object, default is NULL, meaning the spatial neighbours weights will be calculated using the weight.method. if the data contains multiple samples, and the sample_id is specified, it should be provided as a list object with names (using sample_id).

weight.method character the method to build the spatial neighbours weights, default is voronoi (Voronoi tessellation). Other method, which requires coord matrix as input and returns nb, listw or Graph object, also is available, such as "knearneigh", 'dnearneigh', "gabrielneigh", "relativeneigh", which are from spdep package. default is knn, if it is "none", meaning the distance weight of each spot is used to the weight.

reduction.used character used as spatial coordinates to calculate the neighbours weights, default is NULL, the result of reduction can be specified, such as UMAP, TSNE, PCA. If it is specified, the weight neighbours matrix will be calculated using the result of specified reduction.

group.by character a specified category column names (for example the cluster column name) of colData(data). Or a vector of length equal to ncol(x), specifying the group to which each cell is assigned. If it was specified, the adjacency weighted matrix will be built based on the principle that spots or cells in the same category are adjacent, default is NULL.

permutation integer the permutation number to test, default is 100L, if permutation is smaller than 10 or NULL, which will use mantel test to calculate the pvalue.

alternative a character string specifying the alternative hypothesis, which only work with add.pvalue = TRUE, default is two.sided.

add.pvalue logical whether calculate the pvalue, which is calculated with permutation test. So it might be slow, default is FALSE, which the pvalue of result will be NULL.

p.adjust.method
character the method to adjust the pvalue of the result, default is BH.

random.seed numeric random seed number to repeatability, default is 1024.

action character, which should be one of 'only' and 'get', default is "only". This will return a long tidy table (when the sample number of data is one) or a SimpleList which contains long tidy table for each sample. When action="get", it will return a list contained global bivariate spatial autocorrelation and pvalue (when add.pvalue=TRUE), or a SimpleList which contains a list global bivariate spatial result for each sample (when the sample number of data is larger than one).

verbose logical whether print the help information, default is TRUE.

gsvaexp character the one character from the name of gsvaExpNames(data), default is NULL. If data is SVPExperiment, and the parameter is specified simultaneously. the features (Usually genes) from the displayed class, and gsvaexp.features from name in rownames(gsvaExp(data, gsvaexp)) will be performed the analysis.

gsvaexp.assay.type
character the assay name in the assays(gsvaExp(data, gsvaexp)), default is NULL, which works with gsvaexp parameter.

gsvaexp.features
character the name from the rownames(gsvaExp(data, gsvaexp)). If gsvaexp is specified and data is SVPExperiment, it should be provided. Default is NULL.

`across.gsvaexp` logical whether only calculate the relationship of features between the multiple `gsvaExps` not the internal features of gsvaExp. For example, `'a'` and `'b'` features are from the `'AB'` `gsvaExp`, `'c'` and `'d'` features are from the `'CD'` `gsvaExp`. When `across.gsvaexp=TRUE` and `gsvaexp.features = c('a', 'b', 'c', 'd')` and `gsvaexp = c('AB', 'CD')`, Only the relationship of a and c, a and d, b and c, and b and d will be calculated. default is TRUE.

`...` additional parameters the parameters which are from the weight.method function.

Value

SimpleList or long tidy table see also the help information of `action` argument.

Author(s)

Shuangbin Xu

References

1. Lee, SI. Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I . J Geograph Syst 3, 369–385 (2001). https://doi.org/10.1007/s101090100064
2. Lee, SI. A Generalized Significance Testing Method for Global Measures of Spatial Association: An Extension of the Mantel Test. Environment and Planning A: Economy and Space, 36(9), 1687-1703. https://doi.org/10.1068/a34143.

See Also

`runDetectSVG` and `runKldSVG` to identify the spatial variable features. `runLISA` to explore the spatial hotspots.

Examples

```
data(hpda_spe_cell_dec)
rownames(hpda_spe_cell_dec) |> head()
res1 <- runGLOBALBV(hpda_spe_cell_dec,
                    features1 = "Ductal APOL1 high-hypoxic",
                    features2 = c('Cancer clone A', "Cancer clone B"),
                    assay.type = 'affi.score',
                    action='only'
        )
res1
res2 <- runGLOBALBV(hpda_spe_cell_dec,
                    features1 = c("Acinar cells",
                                  "Ductal APOL1 high-hypoxic",
                                  "Cancer clone A",
                                  "Cancer clone B"),
                    assay.type = 1,
                    action = 'get'
        )
res2
# when add.pvalue = TRUE and permutation <= 10 or NULL, the pvalue will be
```

```
# calculated using mantel test.
res3 <- runGLOBALBV(hpda_spe_cell_dec, features1 = rownames(hpda_spe_cell_dec),
                    assay.type = 1, action='get', add.pvalue=TRUE, permutation=NULL)
res3 |> as_tbl_df(diag=FALSE)
```

---

runKldSVG *Detecting the spatially or single cell variable features with Kullback–Leibler divergence of 2D weighted kernel density estimation*

Description

To resolve the sparsity of single cell or spatial omics data, we use kernel function smoothing cell density weighted by the gene expression in a low-dimensional space or physical space. This method had reported that it can better represent the gene expression, it can also recover the signal from cells that are more likely to express a gene based on their neighbouring cells (first reference). Next, we use kullback-leibler divergence to detect the signal genes in a low-dimensional space (UMAP or TSNE for single cell omics data) or a physical space (for spatial omics data). See details to learn more.

Usage

```
runKldSVG(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  sample_id = "all",
  grid.n = 100,
  permutation = 100,
  p.adjust.method = "BH",
  verbose = TRUE,
  action = c("add", "only", "get"),
  random.seed = 1024,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runKldSVG(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  sample_id = "all",
  grid.n = 100,
  permutation = 100,
  p.adjust.method = "BH",
  verbose = TRUE,
  action = c("add", "only", "get"),
```

```
  random.seed = 1024,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SVPExperiment'
runKldSVG(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  sample_id = "all",
  grid.n = 100,
  permutation = 100,
  p.adjust.method = "BH",
  verbose = TRUE,
  action = c("add", "only", "get"),
  random.seed = 1024,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
assay.type	which expressed data to be pulled to run, default is logcounts.
reduction.used	character used as spatial coordinates to calculate the neighbours weights, default is NULL, the result of reduction can be specified, such as UMAP, TSNE, PCA. If it is specified, the weight neighbours matrix will be calculated using the result of specified reduction.
sample_id	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be all to compute metric for all samples; the metric is computed separately for each sample. default is "all".
grid.n	numeric number of grid points in the two directions to estimate 2D weighted kernel density, default is 100.
permutation	numeric the number of permutation for each single feature to detect the significantly spatially or single cell variable features, default is 100.
p.adjust.method	character the method to adjust the pvalue of the result, default is BH.
verbose	logical whether print the intermediate message when running the program, default is TRUE.
action	character control the type of output, if action='add', the result of identification will add the original object, if action = 'get', the result will return a SimpleList, if action = 'only', the result will return a DataFrame by merging the result of all sample, default is add.

random.seed	numeric random seed number to repeatability, default is 1024.
gsvaexp	which gene set variation experiment will be pulled to run, this only work when data is a SVPExperiment, default is NULL.
gsvaexp.assay.type	which assay data in the specified gsvaexp will be used to run, default is NULL.
...	additional parameters

Details

if input is a SVPExperiment, output will be also a SVPExperiment, the spatially variable gene sets result is stored in svDfs of the specified gsvaexp, which is a SingleCellExperiment. If input is a SingleCellExperiment (which is extracted from SVPExperiment using gsvaExp() function), output will be also a SingleCellExperiment, the spatial variable gene sets result can be extracted using svDf function. The result of svDf will return a matrix which has sp.kld, boot.sp.kld.mean, boot.sp.kld.sd, pvalue, padj and rank.

- sp.kld which is logarithms of Kullback–Leibler divergence, larger value meaning the greater the difference from the background distribution without spatial variability.
- boot.sp.kld.mean which is mean of logarithms of Kullback–Leibler divergence based on the permutation of each features.
- boot.sp.kld.sd which is standard deviation of logarithms of Kullback–Leibler divergence based on the permutation of each features.
- pvalue the pvalue is calculated using the real sp.kld and the permutation boot.sp.kld.mean and boot.sp.kld.sd based on the normal distribution.
- padj the adjusted pvalue based on the specified p.adjust.method, default is BH.
- rank the order of significant spatial variable features based on padj and sp.kld.

The kernel density estimation for each features in each cells is done in the following way (first reference article):

$f_h(x) = 1/n \sum_{i=1}^{n} W_i * K_h(x - X_i)$

Where W_i is the value of feature (such as gene expression or gene set score). X_i is the embeddings (two dimension coordinates of UMAP or TSNE or the physical space for spatial omics data) of the cell i. h is a smoothing parameter corresponding to the bandwidth matrix, default is the implementation of ks package. $K(x)$ is a gaussian kernel function. x is the a reference point in the embedding space defined by the grid size used for the computation to weight the distances of nearby cells. $K_h(x^-X_i)$works as a weight for W_i to smooth the feature value based on neighbouring cells at a UMAP or TSNE or physical space.

The Kullback-Leibler divergence for each features is calculated in the following way:

$D_{KL}(G) = \sum_{x \in X} P(x) * \log(P(x)/Q(x))$

Where $P(x)$ is the kernel density value of a feature at the space X. and $Q(x)$ is the kernel density value of no spatially variability reference feature at the space X. The smaller kullback-leibler divergence ($D_{KL}(G)$) show that the distribution of features is more like the no spatially variability reference feature at th space X. So we randomly shuffle the position of each feature and calculate Kullback-Leibler divergence, next we use the normal distribution to calculate the pvalue with

the actual Kullback-Leibler divergence, and the average value and standard deviation value of random Kullback-Leibler divergence, since the random Kullback-Leibler divergence for each feature is normally distributed in the following:

$X \sim \mathcal{N}(\mu, \sigma^2)$

where μ is the average value of random Kullback-Leibler divergence, and σ is standard deviation.

Value

a SVPExperiment or a SingleCellExperiment, see details.

Author(s)

Shuangbin Xu

References

1. Jose Alquicira-Hernandez, Joseph E Powell, Nebulosa recovers single-cell gene expression signals by kernel density estimation. Bioinformatics, 37, 2485–2487(2021), https://doi.org/10.1093/bioinformatics/btab
2. Vandenbon, A., Diez, D. A clustering-independent method for finding differentially expressed genes in single-cell transcriptome data. Nat Commun, 11, 4318 (2020). https://doi.org/10.1038/s41467-020-17900-3
3. https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

See Also

runSGSA to calculate the activity score of gene sets, runLISA to explore the hotspot for specified features in the spatial space.

Examples

```
# This example dataset is extracted from the
# result of runSGSA with gsvaExp(svpe).
data(hpda_spe_cell_dec)

hpda_spe_cell_dec <-
    hpda_spe_cell_dec |>
    runKldSVG(
      assay.type = 'affi.score'
    )

# The result can be extracted svDf()
hpda_spe_cell_dec |> svDf() |> data.frame() |> dplyr::arrange(rank)
# the Acinar cells, Cancer clone A, Cancer clone B etc have
# significant spatial variable.
# Then we can use pred.feature.mode to predict the activity
# mode in spatial domain.
```

runLISA *Local indicators of spatial association analysis*

Description

This function use the local indicators of spatial association (LISA) to identify the hotspot in the spatial space. In other word, it allow users to explore local variations in spatial dependence by measuring each area's relative contribution to the corresponding global measure.

Usage

```
runLISA(
  data,
  features,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("localG", "localmoran"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  cells = NULL,
  action = c("get", "add", "only"),
  alternative = "two.sided",
  p.adjust.method = "BH",
  flag.method = c("mean", "median"),
  BPPARAM = SerialParam(),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runLISA(
  data,
  features,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("localG", "localmoran"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  cells = NULL,
  action = c("get", "add", "only"),
```

```
  alternative = "two.sided",
  p.adjust.method = "BH",
  flag.method = c("mean", "median"),
  BPPARAM = SerialParam(),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  ...
)

## S4 method for signature 'SVPExperiment'
runLISA(
  data,
  features,
  assay.type = "logcounts",
  sample_id = "all",
  method = c("localG", "localmoran"),
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  reduction.used = NULL,
  group.by = NULL,
  cells = NULL,
  action = c("get", "add", "only"),
  alternative = "two.sided",
  p.adjust.method = "BH",
  flag.method = c("mean", "median"),
  BPPARAM = SerialParam(),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
features	the feature name or index of data object, which are required. If gsvaexp is provided and data is SingleCellExperiment, it should be the features from rownames(gsvaExp(data, gsvaexp)).
assay.type	which expressed data to be pulled to run, default is logcounts.
sample_id	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be all to compute metric for all samples; the metric is computed separately for each sample. default is "all".
method	character the method for the local spatial statistic, one of 'localG', "localmoran", default is 'localG'.

weight — object, which can be nb, listw or Graph object, default is NULL, meaning the spatial neighbours weights will be calculated using the weight.method. if the data contains multiple samples, and the sample_id is specified, it should be provided as a list object with names (using sample_id).

weight.method — character the method to build the spatial neighbours weights, default is voronoi (Voronoi tessellation). Other method, which requires coord matrix as input and returns nb, listw or Graph object, also is available, such as "knearneigh", 'dnearneigh', "gabrielneigh", "relativeneigh", which are from spdep package. default is knn, if it is "none", meaning the distance weight of each spot is used to the weight.

reduction.used — character used as spatial coordinates to calculate the neighbours weights, default is NULL, the result of reduction can be specified, such as UMAP, TSNE, PCA. If it is specified, the weight neighbours matrix will be calculated using the result of specified reduction.

group.by — character a specified category column names (for example the cluster column name) of colData(data). Or a vector of length equal to ncol(x), specifying the group to which each cell is assigned. If it was specified, the adjacency weighted matrix will be built based on the principle that spots or cells in the same category are adjacent, default is NULL.

cells — the cell name or index of data object, default is NULL.

action — character, which control the type of return result, default is get, which will return a SimpleList.

alternative — a character string specifying the alternative hypothesis, default is two.sided.

p.adjust.method — a character string specifying the method to adjust p-value, default is BH.

flag.method — a character string specifying the method to calculate the threshold for the cluster type, default is "mean". Other option is "median".

BPPARAM — A BiocParallelParam object specifying whether perform the analysis in parallel using BiocParallel default is SerialParam(), meaning no parallel. You can use BiocParallel::MulticoreParam(workers=4, progressbar=TRUE) to parallel it, the workers of MulticoreParam is the number of cores used, see also MulticoreParam. default is SerialParam().

verbose — logical whether print the help information, default is TRUE.

gsvaexp — which gene set variation experiment will be pulled to run, this only work when data is a SVPExperiment, default is NULL.

gsvaexp.assay.type — which assay data in the specified gsvaexp will be used to run, default is NULL.

gsvaexp.features — character which is from the rownames(gsvaExp(data, gsvaexp)). If gsvaexp is specified and data is SVPExperiment, it should be provided. Default is NULL.

... — additional parameters the parameters which are from the weight.method function.

Value

if `action = 'get'` (in default), the SimpleList object (like list object) will be return, if `action = 'only'`, the data.frame will be return. if `action = 'add'`, the result of LISA is stored in the `localResults` column of `int_colData` (internal column metadata), which can be extracted using `LISAResult`

Author(s)

Shuangbin Xu

References

1. Anselin, L. (1995), Local Indicators of Spatial Association—LISA. Geographical Analysis, 27: 93-115. https://doi.org/10.1111/j.1538-4632.1995.tb00338.x
2. Bivand, R.S., Wong, D.W.S. (2018), Comparing implementations of global and local indicators of spatial association. TEST 27, 716–748. https://doi.org/10.1007/s11749-018-0599-x

See Also

`runDetectSVG` and `runKldSVG` to identify the spatial variable features.

Examples

```
library(SpatialExperiment)
# This example data was extracted from the
# result of runSGSA with gsvaExp() function.
data(hpda_spe_cell_dec)
# using global spatial autocorrelation test to identify the spatial
# variable features.
svres <- runDetectSVG(hpda_spe_cell_dec, assay.type = 'affi.score',
           method = 'moransi', action = 'only')
svres |> dplyr::arrange(rank) |> head()
# In this example, we found the `Cancer clone A` and `Cancer clone B`
# have significant spatial autocorrelation. Next, we use the `runLISA()`
# to explore the spatial hotspots for the features.
lisa.res12 <- hpda_spe_cell_dec |>
   runLISA(
     features = c(1, 2, 3),
     assay.type = 'affi.score',
     weight.method = "knn",
     k = 10,
     action = 'get',
   )
lisa.res12
lisa.res12[['Acinar cells']] |> head()
lisa.res12[["Cancer clone A"]] |> head()
# add the Gi of LISA result to input object.
hpda_spe_cell_dec <- LISAsce(hpda_spe_cell_dec, lisa.res12)
hpda_spe_cell_dec
gsvaExp(hpda_spe_cell_dec, 'LISA')
# Then using ggsc to visualize the result
```

```
#\donttest{
  library(ggplot2)
  library(ggsc)
  p1 <- plot_lisa_feature(hpda_spe_cell_dec, lisa.res12, assay.type=1)
  p2 <- gsvaExp(hpda_spe_cell_dec, 'LISA') |>
  plot_lisa_feature(lisa.res12, assay.type='Gi')
  p1 / p2
#}
```

---

runLOCALBV *Local Bivariate analysis with spatial autocorrelation*

Description

This function is to explore the local bivariate relationship in the spatial space. Like runGLOBALBV, It efficiently reflects the extent to which bivariate associations are spatially grouped in local. Put differently, it can be utilized to quantify the bivariate spatial dependency in local. See also the references.

Usage

```
runLOCALBV(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  bv.method = c("locallee", "localmoran_bv"),
  bv.alternative = "two.sided",
  bv.p.adjust.method = "BH",
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  lisa.method = c("localG", "localmoran"),
  lisa.alternative = "greater",
  lisa.p.adjust.method = "BH",
  lisa.flag.method = c("mean", "median"),
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  random.seed = 1024,
  BPPARAM = SerialParam(),
  action = c("get", "only", "add"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
```

```
)

## S4 method for signature 'SingleCellExperiment'
runLOCALBV(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  bv.method = c("locallee", "localmoran"),
  bv.alternative = "two.sided",
  bv.p.adjust.method = "BH",
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  lisa.method = c("localG", "localmoran"),
  lisa.alternative = "greater",
  lisa.p.adjust.method = "BH",
  lisa.flag.method = c("mean", "median"),
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  random.seed = 1024,
  BPPARAM = SerialParam(),
  action = c("get", "only", "add"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)

## S4 method for signature 'SVPExperiment'
runLOCALBV(
  data,
  features1 = NULL,
  features2 = NULL,
  assay.type = "logcounts",
  sample_id = "all",
  bv.method = c("locallee", "localmoran_bv"),
  bv.alternative = "two.sided",
  bv.p.adjust.method = "BH",
  weight = NULL,
  weight.method = c("voronoi", "knn", "none"),
  lisa.method = c("localG", "localmoran"),
  lisa.alternative = "greater",
  lisa.p.adjust.method = "BH",
  lisa.flag.method = c("mean", "median"),
```

```
  reduction.used = NULL,
  group.by = NULL,
  permutation = 100,
  random.seed = 1024,
  BPPARAM = SerialParam(),
  action = c("get", "only", "add"),
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  gsvaexp.features = NULL,
  across.gsvaexp = TRUE,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
features1	the features name data object (only supporting character), see also features2 parameter.
features2	character, if features1 is not NULL, and features2 is NULL, only the features1 are analyzed, if features1 is NULL, and features2 is is not NULL, the features2 are analyzed, if features2 is also NULL, all of features in the data object will be analyzed. If features2 and features1 are not NULL, the bivariate spatial autocorrelation analysis will be performed between the features1 and features2. default is NULL.
assay.type	which expressed data to be pulled to run, default is logcounts.
sample_id	character the sample(s) in the SpatialExperiment object whose cells/spots to use. Can be all to compute metric for all samples; the metric is computed separately for each sample. default is "all".
bv.method	character one of the 'locallee' and 'localmoran_bv', default is 'locallee'.
bv.alternative	a character string specifying the alternative hypothesis, default is tow.sided. This only work when bv.method = 'localmoran_bv'.
bv.p.adjust.method	character the method to adjust p-value for local bivariate analysis, default is BH.
weight	object, which can be nb, listw or Graph object, default is NULL, meaning the spatial neighbours weights will be calculated using the weight.method. if the data contains multiple samples, and the sample_id is specified, it should be provided as a list object with names (using sample_id).
weight.method	character the method to build the spatial neighbours weights, default is voronoi (Voronoi tessellation). Other method, which requires coord matrix as input and returns nb, listw or Graph object, also is available, such as "knearneigh", 'dnearneigh', "gabrielneigh", "relativeneigh", which are from spdep package. default is knn, if it is "none", meaning the distance weight of each spot is used to the weight.

lisa.method character one of the 'localG' and 'localmoran', this is to perform the LISA analysis using the result of bv.method, which can identify the spatial domain of the bivariate spatial analysis result, default is 'localG'.

lisa.alternative a character string specifying the alternative hypothesis, which works with lisa.method, default is greater.

lisa.p.adjust.method a character string specifying the method to adjust p-value of LISA, default is BH.

lisa.flag.method a character string specifying the method to calculate the threshold for the cluster type, default is "mean". Other option is "median".

reduction.used character used as spatial coordinates to calculate the neighbours weights, default is NULL, the result of reduction can be specified, such as UMAP, TSNE, PCA. If it is specified, the weight neighbours matrix will be calculated using the result of specified reduction.

group.by character a specified category column names (for example the cluster column name) of colData(data). Or a vector of length equal to 'ncol(x)', specifying the group to which each cell is assigned. If it was specified, the adjacency weighted matrix will be built based on the principle that spots or cells in the same category are adjacent, default is NULL.

permutation integer the permutation number to test, which only work with bv.method='localmoran_bv', default is 100L.

random.seed numeric random seed number to repeatability, default is 1024.

BPPARAM A BiocParallelParam object specifying whether perform the analysis in parallel using BiocParallel default is SerialParam(), meaning no parallel. You can use BiocParallel::MulticoreParam(workers=4, progressbar=TRUE) to parallel it, the workers of MulticoreParam is the number of cores used, see also MulticoreParam. default is SerialParam().

action character, which control the type of return result, default is get, which will return a SimpleList.

verbose logical whether print the help information, default is TRUE.

gsvaexp character the one character from the name of gsvaExpNames(data), default is NULL. If data is SVPExperiment, and the parameter is specified simultaneously. the features (Usually genes) from the displayed class, and gsvaexp.features from name in rownames(gsvaExp(data, gsvaexp)) will be performed the analysis.

gsvaexp.assay.type character the assay name in the assays(gsvaExp(data, gsvaexp)), default is NULL, which works with gsvaexp parameter.

gsvaexp.features character the name from the rownames(gsvaExp(data, gsvaexp)). If gsvaexp is specified and data is SVPExperiment, it should be provided. Default is NULL.

across.gsvaexp logical whether only calculate the relationship of features between the multiple gsvaExps not the internal features of gsvaExp. For example, 'a' and 'b'

features are from the 'AB' gsvaExp, 'c' and 'd' features are from the 'CD' gsvaExp. When across.gsvaexp=TRUE and gsvaexp.features = c('a', 'b', 'c', 'd') and gsvaexp = c('AB', 'CD'), Only the relationship of a and c, a and d, b and c, and b and d will be calculated. default is TRUE.

... additional parameters the parameters which are from the weight.method function.

Value

if action = 'get' (in default), the SimpleList object (like list object) will be return, if action = 'only', the data.frame will be return. if action = 'add', the result of LISA is stored in the localResults column of int_colData (internal column metadata). You can use localResults() function of SpatialFeatureExperiment package to extract it.

Author(s)

Shuangbin Xu

References

Lee, SI. Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I . J Geograph Syst 3, 369–385 (2001). https://doi.org/10.1007/s101090100064

See Also

runDetectSVG and runKldSVG to identify the spatial variable features, runGLOBALBV to analysis the global bivariate spatial analysis, runLISA to identify the spatial domain of specified features.

Examples

```
data(hpda_spe_cell_dec)
res1 <- hpda_spe_cell_dec |> runLOCALBV(
          features1 = 'Cancer clone A',
          features2 = 'Cancer clone B',
          assay.type='affi.score'
        )
res1
res1[['Cancer clone A_VS_Cancer clone B']] |> head()
# add the LocalLee and Gi of LOCALBV result to input object.
hpda_spe_cell_dec <- LISAsce(hpda_spe_cell_dec, res1, 'LOCALBV')
hpda_spe_cell_dec
gsvaExp(hpda_spe_cell_dec, 'LOCALBV')
# Then using ggsc to visualize the result
#\donttest{
  library(ggplot2)
  library(ggsc)
  gsvaExp(hpda_spe_cell_dec, 'LOCALBV') |>
  plot_lisa_feature(res1, assay.type='LocalLee') + ggtitle(NULL)
#}
```

runMCA *Run Multiple Correspondence Analysis*

Description

Perform a Multiple Correspondence Analysis (MCA) on cells, based on the expression data in a SingleCellExperiment object. It is modified based on the RunMCA of CelliD with the source codes of C++.

Usage

```
runMCA(
  data,
  assay.type = "logcounts",
  reduction.name = "MCA",
  ncomponents = 30,
  subset.row = NULL,
  subset.col = NULL,
  group.by.vars = NULL,
  consider.spcoord = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runMCA(
  data,
  assay.type = "logcounts",
  reduction.name = "MCA",
  ncomponents = 50,
  subset.row = NULL,
  subset.col = NULL,
  group.by.vars = NULL,
  consider.spcoord = FALSE,
  ...
)
```

Arguments

data	a SingleCellExperiment object
assay.type	which expressed data to be pulled to run, default is logcounts.
reduction.name	name of the reduction result, default is MCA.
ncomponents	number of components to compute and store, default is 30.
subset.row	Vector specifying the subset of features to be used for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector, default is NULL, meaning all features to be used for dimensionality reduction.

subset.col	Vector specifying the subset of cells to be used for dimensionality reduction. This can be a character vector of column names, an integer vector of column indices or a logical vector, default is NULL, meaning all cells to be used for dimensionality reduction.
group.by.vars	character the name(s) of covariates that harmony will remove its effect on the data, default is NULL.
consider.spcoord	whether consider the spatial coords as the features of data to run MCA, default is FALSE (TRUE is experimental).
...	additional parameters, see also RunHarmony.

Value

a SingleCellExperiment and the reduction result of MCA can be extracted using reducedDim() function.

Examples

```
library(scuttle)
library(SingleCellExperiment)
small.sce <- mockSCE()
small.sce <- logNormCounts(small.sce)
# To improve computational efficiency, you can use RhpcBLASctl to control the number
# of threads on BLAS. From example
# RhpcBLASctl::blas_set_num_threads(threads = 48)
small.sce <- runMCA(small.sce, assay.type = 'logcounts',
                    reduction.name = 'MCA', ncomponents = 20)
# The MCA result can be extracted using reducedDim of SingleCellExperiment
mca.res <- reducedDim(small.sce, 'MCA')
mca.res |> str()
```

---

runSGSA *Calculate the activity of gene sets in spatial or single-cell data with restart walk with restart and hyper test weighted.*

Description

First, we calculated the distance between cells and between genes, between cells and genes in space of MCA. Because the closer gene is to a cell, the more specific to such the cell it can be considered in MCA space (first reference). We extract the top nearest genes for each cells, to obtain the cells and cells association, genes and gens association, we also extract the top nearest cells or genes respectively, then combine all the association into the same network to obtain the adjacency matrix of all cells and genes. Another method is that we build the network using the combined MCA space of cells and genes directly. Next, we build a starting seed matrix (which each column measures the initial probability distribution of each gene set in graph nodes) for random walk with restart using the gene set and all nodes of the graph. Finally, we employ the restart walk with restart algorithm to compute the affinity score for each gene set or pathway, which is then further weighted using the hypergeometric test result from the original expression matrix controlled by hyper.test.weighted parameter.

Usage

```
runSGSA(
  data,
  gset.idx.list,
  gsvaExp.name = "gset1.rwr",
  symbol.from.gson = FALSE,
  min.sz = 5,
  max.sz = Inf,
  gene.occurrence.rate = 0.2,
  assay.type = "logcounts",
  knn.used.reduction.dims = 30,
  knn.combined.cell.feature = FALSE,
  knn.graph.weighted = TRUE,
  knn.k.use = round(0.06 * nrow(data)),
  rwr.restart = 0.75,
  rwr.normalize.adj.method = c("laplacian", "row", "column", "none"),
  rwr.normalize.affinity = FALSE,
  rwr.prop.normalize = FALSE,
  rwr.threads = NULL,
  hyper.test.weighted = c("Hypergeometric", "Wallenius", "none"),
  hyper.test.by.expr = TRUE,
  prop.score = FALSE,
  add.weighted.metric = FALSE,
  add.cor.features = FALSE,
  cells = NULL,
  features = NULL,
  verbose = TRUE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runSGSA(
  data,
  gset.idx.list,
  gsvaExp.name = "gset1.rwr",
  symbol.from.gson = FALSE,
  min.sz = 5,
  max.sz = Inf,
  gene.occurrence.rate = 0.2,
  assay.type = "logcounts",
  knn.used.reduction.dims = 30,
  knn.combined.cell.feature = FALSE,
  knn.graph.weighted = TRUE,
  knn.k.use = round(0.06 * nrow(data)),
  rwr.restart = 0.75,
  rwr.normalize.adj.method = c("laplacian", "row", "column", "none"),
  rwr.normalize.affinity = FALSE,
  rwr.prop.normalize = FALSE,
```

```
  rwr.threads = NULL,
  hyper.test.weighted = c("Hypergeometric", "Wallenius", "none"),
  hyper.test.by.expr = TRUE,
  prop.score = FALSE,
  add.weighted.metric = FALSE,
  add.cor.features = FALSE,
  cells = NULL,
  features = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	a SingleCellExperiment object normalized and have the result of UMAP or TSNE. Or a SVPExperiment object.
gset.idx.list	gene set list contains the names, or GSON object or a gmt file, and the online gmt file is also supported.
gsvaExp.name	a character the name of gsvaExp of result SVP object.
symbol.from.gson	logical whether extract the SYMBOL ID as gset.idx.list, only work when gset.idx.list is a GSON object.
min.sz	integer the minimum gene set number, default is 5, the number of gene sets smaller than min.sz will be ignored.
max.sz	integer the maximum gene set number, default is Inf, the number of gene sets larger than max.sz will be ignored.
gene.occurrence.rate	the occurrence proportion of the gene set in the input object, default is 0.2.
assay.type	which expressed data to be pulled to build KNN Graph, default is logcounts.
knn.used.reduction.dims	the top components of the reduction with MCA to be used to build KNN Graph, default is 30.
knn.combined.cell.feature	whether combined the embeddings of cells and features to find the nearest neighbor and build graph, default is FALSE, meaning the nearest neighbor will be found in cells to cells, features to features, cells to features respectively to build graph.
knn.graph.weighted	logical whether consider the distance of nodes in the Nearest Neighbors, default is TRUE.
knn.k.use	numeric the number of the Nearest Neighbors nodes, default is 0.06 * number of gene in data.
rwr.restart	the restart probability used for restart walk with restart, should be between 0 and 1, default is 0.75.
rwr.normalize.adj.method	character the method to normalize the adjacency matrix of the input graph, default is laplacian.

`rwr.normalize.affinity`
: logical whether normalize the activity (affinity) result score using quantile normalization, default is FALSE.

`rwr.prop.normalize`
: logical whether divide the specific activity score by total activity score for a sample, default is FALSE. if `gset.idx.list` is celltype gene set, we recommend using TRUE.

`rwr.threads`
: the threads to run Random Walk With Restart (RWR), default is NULL, which will initialize with the default number of threads, you can also set this using `RcppParallel::setThreadOptions(numThreads=10)`.

`hyper.test.weighted`
: character which method to weight the activity score of cell, should is one of "Hypergeometric", "Wallenius", "none", default is "Hypergeometric".

`hyper.test.by.expr`
: logical whether using the expression matrix to find the nearest genes of cells, default is TRUE, if it is FALSE, meaning using the result of reduction to find the nearest genes of cells to perform the `hyper.test.weighted`.

`prop.score`
: logical whether to normalize each feature for each sample, default is FALSE.

`add.weighted.metric`
: logical whether return the weight activity score of cell using the corresponding `hyper.test.weighted`, default is FALSE.

`add.cor.features`
: logical whether calculate the correlation between the new features and original features (genes), default is FALSE. If it is TRUE the correlation result will be kept in fscoreDf which can be extracted using `fscoreDf()` function.

`cells`
: Vector specifying the subset of cells to be used for the calculation of the active score or identification of SV features. This can be a character vector of cell names, an integer vector of column indices or a logical vector, default is NULL, meaning all cells to be used for the calculation of the activity score or identification of SV features.

`features`
: Vector specifying the subset of features to be used for the calculation of the activity score or identification of SV features. This can be a character vector of features names, an integer vector of row indices or a logical vector, default is NULL, meaning all features to be used for the calculation of the activity score or identification of SV features.

`verbose`
: logical whether print the intermediate message when running the program, default is TRUE.

`...`
: additional parameters

Details

if input is a SVPExperiment, output will be also a SVPExperiment, the activity score of gene sets was stored in `assay` slot of the specified `gsvaexp`, and the spatially variable gene sets result is stored in `svDfs` of the specified `gsvaexp`, which is a SingleCellExperiment. If input is a SingleCellExperiment (which is extracted from SVPExperiment using `gsvaExp()` function), output will be also a

SingleCellExperiment, the activity score of gene sets result can be extracted using `assay()` function. The spatially variable gene sets result can be extracted using `svDf()` function. The affinity score is calculated in the following way (refer to the second article):

$P_{t+1} = (1 - r) * M * P_t + r * P_0$

where P_0 is the initial probability distribution for each gene set, M is the transition matrix that is the column normalization of adjacency matrix of graph, r is the global restart probability, P_{t+1} and P_t represent the probability distribution in each iteration. After several iterations, the difference between P_{t+1} and P_t becomes negligible, the stationary probability distribution is reached, indicating proximity measures from every graph node. Iterations are stopped when the difference between P_{t+1} and P_t falls below 1e-10.

Value

a SVPExperiment or a SingleCellExperiment, see details.

Author(s)

Shuangbin Xu

References

1. Cortal, A., Martignetti, L., Six, E. et al. Gene signature extraction and cell identity recognition at the single-cell level with Cell-ID. Nat Biotechnol 39, 1095–1102 (2021). https://doi.org/10.1038/s41587-021-00896-6
2. Alberto Valdeolivas, Laurent Tichit, Claire Navarro, Sophie Perrin, et al. Random walk with restart on multiplex and heterogeneous biological networks, Bioinformatics, 35, 3, 497–505(2019), https://doi.org/10.1093/bioinformatics/bty637

See Also

`runDetectSVG` and `runKldSVG` to identify the spatial variable features. `runGLOBALBV` to explore the spatial co-distribution between the spatial variable features

Examples

```
data(sceSubPbmc)
library(SingleCellExperiment) |> suppressPackageStartupMessages()
library(scuttle) |> suppressPackageStartupMessages()
sceSubPbmc <- scuttle::logNormCounts(sceSubPbmc)
# the using runMCA to perform MCA (Multiple Correspondence Analysis)
# this is refer to the CelliD, but we using the Eigen to speed up.
# You can view the help information of runMCA using ?runMCA.
sceSubPbmc <- runMCA(sceSubPbmc, assay.type = 'logcounts')

# Next, we can calculate the activity score of gene sets provided.
# Here, we use the Cell Cycle gene set from the Seurat
# You can use other gene set, such as KEGG pathway, GO, Hallmark of MSigDB
# or TFs gene sets etc.
#
# supporting the list with names or gson object or the gmt file
```

```
# online gmt file is also be supported
# such as
# https://data.broadinstitute.org/gsea-msigdb/msigdb/release/2023.2.Hs/h.all.v2023.2.Hs.symbols.gmt

data(CellCycle.Hs)
sceSubPbmc <- runSGSA(sceSubPbmc, gset.idx.list = CellCycle.Hs, gsvaExp.name = 'CellCycle')
# Then a SVPE class which inherits SingleCellExperiment, is return.
sceSubPbmc

# You can obtaion the score matrix by following the commond
sceSubPbmc |> gsvaExp('CellCycle')
sceSubPbmc |> gsvaExp("CellCycle") |> assay() |> t() |> head()

# Then you can use the ggsc or other package to visulize
# and you can try to use the findMarkers of scran or other packages to identify
# the different gene sets.
#\donttest{
  library(ggplot2)
  library(ggsc)
  sceSubPbmc <- sceSubPbmc |>
                scater::runPCA(assay.type = 'logcounts', ntop = 600) |>
                scater::runUMAP(dimred = 'PCA')
  # withReducedDim = TRUE, the original reducetion results from original gene features
  # will be add the colData in the sce.cellcycle.
  sce.cellcycle <- sceSubPbmc |> gsvaExp('CellCycle', withReducedDim=TRUE)
  sce.cellcycle
  sce.cellcycle |> sc_violin(
                      features = rownames(sce.cellcycle),
                      mapping = aes(x=seurat_annotations, fill = seurat_annotations)
                    ) +
                    scale_x_discrete(guide=guide_axis(angle=-45))
  sce.cellcycle |> sc_feature(features= "S", reduction='UMAP')
  library(scran)
  cellcycle.test.res <- sce.cellcycle |> findMarkers(
                      group = sce.cellcycle$seurat_annotations,
                      test.type = 'wilcox',
                      assay.type = 'affi.score',
                      add.summary = TRUE
                   )
  cellcycle.test.res$B
#}
```

---

runWKDE *Calculating the 2D Weighted Kernel Density Estimation*

Description

Calculating the 2D Weighted Kernel Density Estimation

Usage

```
runWKDE(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  grid.n = 100,
  adjust = 1,
  bandwidths = NULL,
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
runWKDE(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  grid.n = 100,
  adjust = 1,
  bandwidths = NULL,
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)

## S4 method for signature 'SVPExperiment'
runWKDE(
  data,
  assay.type = "logcounts",
  reduction.used = NULL,
  grid.n = 100,
  adjust = 1,
  bandwidths = NULL,
  verbose = TRUE,
  gsvaexp = NULL,
  gsvaexp.assay.type = NULL,
  ...
)
```

Arguments

data	a SingleCellExperiment object with contains UMAP or TSNE, or a SpatialExperiment object, or a SVPExperiment object with specified gsvaexp argument.
assay.type	which expressed data to be pulled to run, default is logcounts.

reduction.used	character used as spatial coordinates to detect SVG, default is NULL, if data has spatialCoords, which will be used as spatial coordinates, if this is provided the coordinate of specified reduced result will be used.
grid.n	integer number of grid points in the two directions to estimate 2D weighted kernel density, default is 100.
adjust	numeric to adjust the bandwidths, default is 1.0.
bandwidths	vector a two length numeric vector represents the bandwidths for x and y directions, default is normal reference bandwidth hpi, see also bandwidth.nrd.
verbose	logical whether print the intermediate message when running the program, default is TRUE.
gsvaexp	which gene set variation experiment will be pulled to run, this only work when data is a SVPExperiment, default is NULL.
gsvaexp.assay.type	which assay data in the specified gsvaexp will be used to run, default is NULL.
...	additional parameters

Value

a SVPExperiment or SingleCellExperiment

Author(s)

Shuangbin Xu

Examples

```
library(SpatialExperiment)
data(hpda_spe_cell_dec)
hpda_spe_cell_dec <- hpda_spe_cell_dec |> runWKDE(assay.type = 'affi.score')
# The result is saved in the assays (affi.score.density name) of SVPExperiment
# which can be extracted using assay and visualized using ggsc or
# other packages
assays(hpda_spe_cell_dec)
#\donttest{
    library(ggsc)

    f1 <- sc_spatial(hpda_spe_cell_dec, features="Cancer clone A",
              mapping=aes(x=x,y=y),
              slot = 'affi.score.density',
              pointsize=10
    ) +
    scale_bg_color_manual(values=c('black'))
    f1

    f2 <- sc_spatial(hpda_spe_cell_dec, features="Cancer clone B",
              mapping=aes(x=x,y=y),
              pointsize=10,
              slot = 'affi.score.density'
    ) +
```

```
    scale_bg_color_manual(values=c('black'))
    f2
#}
```

---

svDfs *spatial or single cell variable features matrix extract method*

Description

the identification result of the spatial variable or single cell variable (SV) features is important to the downstream analysis.

Value

see `Getters` and `setter`.

Getters

In the following examples, `x` is a SingleCellExperiment object.

`svDf(x, type)`: Retrieves a DataFrame containing the new features (gene sets) (rows) for the specified `type`. `type` should either be a string specifying the name of the features scores matrix in `x` to retrieve, or a numeric scalar specifying the index of the desired matrix, defaulting to the first matrix is missing.

`svDfNames(x)`: Retures a character vector containing the names of all features SV DataFrame Lists in `x`. This is guaranteed to be of the same length as the number of results.

`svDfs(x)`: Returns a named List of matrices containing one or more DataFrame objects. Each object is guaranteed to have the same number of rows, in a 1:1 correspondence to those in `x`.

Single-object setter

`svDf(x, type) <- value` will add or replace an SV matrix in a SingleCellExperiment object `x`. The value of `type` determines how the result is added or replaced:

- If `type` is missing, `value` is assigned to the first result. If the result already exists, its name is preserved; otherwise it is given a default name `"unnamed.sv1"`.
- If `type` is a numeric scalar, it must be within the range of existing results, and `value` will be assigned to the result at that index.
- If `type` is a string and a result exists with this name, `value` is assigned to to that result. Otherwise a new result with this name is append to the existing list of results.

Other setter

`svDfs(x) <- value`: Replaces all features sv result matrices in `x` with those in `value`. The latter should be a list-like object containing any number of DataFrame objects with number of row equal to `nrow(x)`.

If `value` is named, those names will be used to name the SV matrices in `x`. Otherwise, unnamed results are assigned default names prefixed with `"unnamed.sv"`.

If `value` is `NULL`, all SV matrices in `x` are removed.

`svDfNames(x) <- value`: Replaces all names for SV matrices in `x` with a character vector `value`. This should be of length equal to the number of results currently in `x`.

Examples

```
# Using the SingleCellExperiment class example
library(SingleCellExperiment) |> suppressPackageStartupMessages()
example(SingleCellExperiment, echo = FALSE)
dim(counts(sce))
rownames(sce) <- paste0("gene", seq(nrow(sce)))
colnames(sce) <- paste0("cell", seq(ncol(sce)))
# Mocking up some result of spatially variable gene or high variable gene
da1 <- data.frame(kld = abs(rnorm(nrow(sce), .4)),
                  pvalue = abs(rnorm(nrow(sce), .001))) |>
       as.matrix()
rownames(da1) <- rownames(sce)
da2 <- data.frame(moransi = abs(rnorm(nrow(sce), .4)),
                  pvalue = abs(rnorm(nrow(sce), .001))) |>
       as.matrix()
rownames(da2) <- rownames(sce)
svDfs(sce) <- list()
svDf(sce, "kld") <- da1
svDf(sce, "moransi") <- da2
svDfs(sce)
svDfNames(sce)
svDf(sce, "kld") |> head()
svDf(sce, "moransi") |> head()
svDf(sce, 2) |> head()
```

SVP-accessors *Some accessor functions to get the internal slots of SVPExperiment*

Description

Some accessor functions to get the internal slots of SVPExperiment

Usage

```
## S4 method for signature 'SVPExperiment'
spatialCoords(x)
```

```
## S4 method for signature 'SVPExperiment'
spatialCoordsNames(x)

## S4 method for signature 'SVPExperiment'
imgData(x)

## S4 replacement method for signature 'SVPExperiment,DataFrame'
imgData(x) <- value

## S4 replacement method for signature 'SVPExperiment,NULL'
imgData(x) <- value

## S4 replacement method for signature 'SVPExperiment,matrix_Or_NULL'
spatialCoords(x) <- value

## S4 replacement method for signature 'SVPExperiment,character'
spatialCoordsNames(x) <- value

## S4 method for signature 'SVPExperiment'
show(object)
```

Arguments

x	a SVPExperiment class.
value	matrix for `spatialCoords(object) <- value` character for `spatialCoordsNames(object) <- value`.
object	a SVPExperiment class.

Value

matrix or character or print the information of object or a SVPExperiment object.

Examples

```
library(SpatialExperiment) |> suppressPackageStartupMessages()
library(DropletUtils) |> suppressPackageStartupMessages()
example(read10xVisium, echo = FALSE)
svpe <- as(spe, 'SVPExperiment')
svpe
spatialCoords(svpe) |> head()
```

SVPExperiment *The SVPExperiment class*

Description

The SVPExperiment class

Usage

```
SVPExperiment(..., gsvaExps = list())
```

Arguments

...	passed to the SingleCellExperiment constructor to fill the slots of the base class.
gsvaExps	list containing SingleCellExperiment object, each of which should have the same number of columns as the output SVPExperiment object.

Value

a SVPExperiment object

Author(s)

Shuangbin Xu

Examples

```
library(SingleCellExperiment) |> suppressPackageStartupMessages()
ncells <- 100
u <- matrix(rpois(20000, 5), ncol=ncells)
v <- log2(u + 1)
pca <- matrix(runif(ncells*5), ncells)
tsne <- matrix(rnorm(ncells*2), ncells)

svpe <- SVPExperiment(assays=list(counts=u, logcounts=v),
    reducedDims=SimpleList(PCA=pca, tSNE=tsne))
svpe

## coercion from SingleCellExperiment
sce <- SingleCellExperiment(assays=list(counts=u, logcounts=v),
reducedDims=SimpleList(PCA=pca, tSNE=tsne))
svpe <- as(sce, 'SVPExperiment')
svpe
```

Index

∗ **data**
CellCycle.Hs, 7
data_CancerSEA, 9
data_hpda_spe_cell_dec, 10
data_sceSubPbmc, 11
data_SenMayo, 11
mob_marker_genes, 21
mob_sce, 22
∗ **internal**
reexports, 25
SVP-package, 3
[,SCEByColumn,ANY,ANY,ANY-method *(gsvaExps)*, 16
[<-,SCEByColumn,ANY,ANY,ANY-method *(gsvaExps)*, 16

Annotated, *18*
as_tbl_df, 4

bandwidth.nrd, *61*

c,SCEByColumn-method *(gsvaExps)*, 16
cal_lisa_f1, 5
cal_lisa_f1,SingleCellExperiment *(cal_lisa_f1)*, 5
cal_lisa_f1,SingleCellExperiment-method *(cal_lisa_f1)*, 5
CancerSEAEnsemble *(data_CancerSEA)*, 9
CancerSEASymbol *(data_CancerSEA)*, 9
CellCycle.Hs, 7
cluster.assign, 7, *25*
cluster.assign,SingleCellExperiment *(cluster.assign)*, 7
cluster.assign,SingleCellExperiment-method *(cluster.assign)*, 7
cluster.assign,SVPExperiment *(cluster.assign)*, 7
cluster.assign,SVPExperiment-method *(cluster.assign)*, 7
coerce,SingleCellExperiment,SVPExperiment-method *(SVPExperiment)*, 64
colData, *17*

data_CacerSEA *(data_CancerSEA)*, 9
data_CancerSEA, 9
data_CellCycle.Hs *(CellCycle.Hs)*, 7
data_hpda_spe_cell_dec, 10
data_sceSubPbmc, 11
data_SenMayo, 11
DataFrame, *15*, *33*, *41*, *62*, *63*

extract_weight_adj, 12
extract_weight_adj,SingleCellExperiment *(extract_weight_adj)*, 12
extract_weight_adj,SingleCellExperiment-method *(extract_weight_adj)*, 12

fast_cor, 13
fscoreDf *(fscoreDfs)*, 14
fscoreDf,SingleCellExperiment,character-method *(fscoreDfs)*, 14
fscoreDf,SingleCellExperiment,missing-method *(fscoreDfs)*, 14
fscoreDf,SingleCellExperiment,numeric-method *(fscoreDfs)*, 14
fscoreDf<- *(fscoreDfs)*, 14
fscoreDf<-,SingleCellExperiment,character-method *(fscoreDfs)*, 14
fscoreDf<-,SingleCellExperiment,missing-method *(fscoreDfs)*, 14
fscoreDf<-,SingleCellExperiment,numeric-method *(fscoreDfs)*, 14
fscoreDfNames *(fscoreDfs)*, 14
fscoreDfNames,SingleCellExperiment-method *(fscoreDfs)*, 14
fscoreDfNames<- *(fscoreDfs)*, 14
fscoreDfNames<-,SingleCellExperiment,character-method *(fscoreDfs)*, 14
fscoreDfs, 14

fscoreDfs,SingleCellExperiment-method (fscoreDfs), 14
fscoreDfs<- (fscoreDfs), 14
fscoreDfs<-,SingleCellExperiment-method (fscoreDfs), 14

gsvaExp (gsvaExps), 16
gsvaExp,SVPExperiment,character-method (gsvaExps), 16
gsvaExp,SVPExperiment,missing-method (gsvaExps), 16
gsvaExp,SVPExperiment,numeric-method (gsvaExps), 16
gsvaExp<- (gsvaExps), 16
gsvaExp<-,SVPExperiment,character-method (gsvaExps), 16
gsvaExp<-,SVPExperiment,missing-method (gsvaExps), 16
gsvaExp<-,SVPExperiment,numeric-method (gsvaExps), 16
gsvaExpNames (gsvaExps), 16
gsvaExpNames,SVPExperiment-method (gsvaExps), 16
gsvaExpNames<- (gsvaExps), 16
gsvaExpNames<-,SVPExperiment,character-method (gsvaExps), 16
gsvaExps, 16
gsvaExps,SVPExperiment-method (gsvaExps), 16
gsvaExps<- (gsvaExps), 16
gsvaExps<-,SVPExperiment-method (gsvaExps), 16

hpda_spe_cell_dec (data_hpda_spe_cell_dec), 10
hpi, *61*

imgData (reexports), 25
imgData(), *25*
imgData,SVPExperiment-method (SVP-accessors), 63
imgData<- (reexports), 25
imgData<-,SVPExperiment,DataFrame-method (SVP-accessors), 63
imgData<-,SVPExperiment,NULL-method (SVP-accessors), 63

length,SCEByColumn-method (gsvaExps), 16
LISAResult, 19, *47*
LISAsce, 20
LISAsce,SingleCellExperiment (LISAsce), 20
LISAsce,SingleCellExperiment-method (LISAsce), 20
List, *15*, *17*, *62*

mainGsvaExpName (gsvaExps), 16
mainGsvaExpName,SVPExperiment-method (gsvaExps), 16
mainGsvaExpName<- (gsvaExps), 16
mainGsvaExpName<-,SVPExperiment,character_OR_NULL-method (gsvaExps), 16
mcols, *18*
metadata, *18*
mob_marker_genes, 21
mob_sce, 22
MulticoreParam, *30*, *46*, *51*

names,SCEByColumn-method (gsvaExps), 16
names<-,SCEByColumn-method (gsvaExps), 16

plot_heatmap_globalbv, 22
pred.cell.signature, 24
pred.cell.signature,SingleCellExperiment (pred.cell.signature), 24
pred.cell.signature,SingleCellExperiment-method (pred.cell.signature), 24
pred.cell.signature,SVPExperiment (pred.cell.signature), 24
pred.cell.signature,SVPExperiment-method (pred.cell.signature), 24

reexports, 25
runCORR, 26, *28*
runCORR,SingleCellExperiment (runCORR), 26
runCORR,SingleCellExperiment-method (runCORR), 26
runCORR,SVPExperiment (runCORR), 26
runCORR,SVPExperiment-method (runCORR), 26
runDetectMarker, 29
runDetectMarker,SingleCellExperiment (runDetectMarker), 29
runDetectMarker,SingleCellExperiment-method (runDetectMarker), 29
runDetectSVG, 30, *39*, *47*, *52*, *58*

runDetectSVG,SingleCellExperiment (runDetectSVG), 30
runDetectSVG,SingleCellExperiment-method (runDetectSVG), 30
runDetectSVG,SVPExperiment (runDetectSVG), 30
runDetectSVG,SVPExperiment-method (runDetectSVG), 30
runENCODE, 34
runENCODE,SingleCellExperiment (runENCODE), 34
runENCODE,SingleCellExperiment-method (runENCODE), 34
runGLOBALBV, 35, *52*, *58*
runGLOBALBV,SingleCellExperiment (runGLOBALBV), 35
runGLOBALBV,SingleCellExperiment-method (runGLOBALBV), 35
runGLOBALBV,SVPExperiment (runGLOBALBV), 35
runGLOBALBV,SVPExperiment-method (runGLOBALBV), 35
runKldSVG, *39*, 40, *47*, *52*, *58*
runKldSVG,SingleCellExperiment (runKldSVG), 40
runKldSVG,SingleCellExperiment-method (runKldSVG), 40
runKldSVG,SVPExperiment (runKldSVG), 40
runKldSVG,SVPExperiment-method (runKldSVG), 40
runLISA, *21*, *33*, *39*, *43*, 44, *52*
runLISA,SingleCellExperiment (runLISA), 44
runLISA,SingleCellExperiment-method (runLISA), 44
runLISA,SVPExperiment (runLISA), 44
runLISA,SVPExperiment-method (runLISA), 44
runLOCALBV, *21*, 48
runLOCALBV,SingleCellExperiment (runLOCALBV), 48
runLOCALBV,SingleCellExperiment-method (runLOCALBV), 48
runLOCALBV,SVPExperiment (runLOCALBV), 48
runLOCALBV,SVPExperiment-method (runLOCALBV), 48
runMCA, 53
runMCA,SingleCellExperiment (runMCA), 53
runMCA,SingleCellExperiment-method (runMCA), 53
runSGSA, *8*, *25*, *43*, 54
runSGSA,SingleCellExperiment (runSGSA), 54
runSGSA,SingleCellExperiment-method (runSGSA), 54
runWKDE, 59
runWKDE,SingleCellExperiment (runWKDE), 59
runWKDE,SingleCellExperiment-method (runWKDE), 59
runWKDE,SVPExperiment (runWKDE), 59
runWKDE,SVPExperiment-method (runWKDE), 59

sceSubPbmc (data_sceSubPbmc), 11
SenMayoSymbol (data_SenMayo), 11
show,SVPExperiment-method (SVP-accessors), 63
SimpleList, *33*, *41*, *46*, *51*
SingleCellExperiment, *6*, *8*, *11*, *12*, *15–17*, *19*, *20*, *22*, *24*, *25*, *27*, *32*, *33*, *35*, *37*, *41–43*, *45*, *50*, *54*, *56–58*, *60–62*, *65*
spatialCoords (reexports), 25
spatialCoords(), *25*
spatialCoords,SVPExperiment-method (SVP-accessors), 63
spatialCoords<- (reexports), 25
spatialCoords<-,SVPExperiment (SVP-accessors), 63
spatialCoords<-,SVPExperiment,matrix_Or_NULL-method (SVP-accessors), 63
spatialCoordsNames (reexports), 25
spatialCoordsNames(), *25*
spatialCoordsNames,SVPExperiment-method (SVP-accessors), 63
spatialCoordsNames<- (reexports), 25
spatialCoordsNames<-,SVPExperiment,character-method (SVP-accessors), 63
SpatialExperiment, *10*, *12*, *20*, *27*, *32*, *35*, *37*, *41*, *45*, *50*, *60*
svDf (svDfs), 62
svDf,SingleCellExperiment,character-method (svDfs), 62
svDf,SingleCellExperiment,missing-method (svDfs), 62

svDf,SingleCellExperiment,numeric-method
(svDfs), 62
svDf<- (svDfs), 62
svDf<-,SingleCellExperiment,character-method
(svDfs), 62
svDf<-,SingleCellExperiment,missing-method
(svDfs), 62
svDf<-,SingleCellExperiment,numeric-method
(svDfs), 62
svDfNames (svDfs), 62
svDfNames,SingleCellExperiment-method
(svDfs), 62
svDfNames<- (svDfs), 62
svDfNames<-,SingleCellExperiment,character-method
(svDfs), 62
svDfs, 62
svDfs,SingleCellExperiment-method
(svDfs), 62
svDfs<- (svDfs), 62
svDfs<-,SingleCellExperiment-method
(svDfs), 62
SVP (SVP-package), 3
SVP-accessors, 63
SVP-package, 3
SVPExperiment, *8*, *16–18*, *20*, *24, 25*, *27, 28*, *32, 33*, *35*, *37, 38*, *41–43*, *45, 46*, *50, 51*, *56–58*, *60, 61*, *64*, 64, *65*
SVPExperiment-class (SVPExperiment), 64

Vector, *18*