Package ‘DESpace’

November 2, 2025

Type Package

Title DESpace: a framework to discover spatially variable genes and differential spatial patterns across conditions

Version 2.3.1

Description Intuitive framework for identifying spatially variable genes (SVGs) and differential spatial variable pattern (DSP) between conditions via edgeR, a popular method for performing differential expression analyses.
Based on pre-annotated spatial clusters as summarized spatial information, DESpace models gene expression using a negative binomial (NB), via edgeR, with spatial clusters as covariates. SVGs are then identified by testing the significance of spatial clusters.
For multi-sample, multi-condition datasets, we again fit a NB model via edgeR, incorporating spatial clusters, conditions and their interactions as covariates.
DSP genes-
representing differences in spatial gene expression patterns across experimental conditions-
are identified by testing the interaction between spatial clusters and conditions.

biocViews Spatial, SingleCell, RNASeq, Transcriptomics, GeneExpression, Sequencing, DifferentialExpression,StatisticalMethod, Visualization

License GPL-3

Depends R (>= 4.5.0)

Imports edgeR, limma, dplyr, stats, Matrix, SpatialExperiment, ggplot2, SummarizedExperiment, S4Vectors, BiocGenerics, data.table, assertthat, terra, sf, spatstat.explore, spatstat.geom, ggforce, ggnewscale, patchwork, BiocParallel, methods, scales, scuttle

Suggests knitr, rmarkdown, testthat, BiocStyle, muSpaData, ExperimentHub, spatialLIBD, purrr, reshape2, tidyverse, concaveman

VignetteBuilder knitr

RoxygenNote 7.3.2

ByteCompile true

Encoding UTF-8

URL https://github.com/peicai/DESpace,

https://peicai.github.io/DESpace/

BugReports https://github.com/peicai/DESpace/issues

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

git_branch devel

git_last_commit 3dbd101

git_last_commit_date 2025-10-31

Repository Bioconductor 3.23

Date/Publication 2025-11-02

Author Peiying Cai [aut, cre] (ORCID: <https://orcid.org/0009-0001-9229-2244>),
Simone Tiberi [aut] (ORCID: <https://orcid.org/0000-0002-3054-9964>)

Maintainer Peiying Cai <peiying.cai@uzh.ch>

Contents

DESpace . . . 2
dsp_test . . . 3
FeaturePlot . . . 5
individual_dsp . . . 8
individual_svg . . . 9
LIBD_subset . . . 11
results_individual_svg . . . 12
results_svg_test . . . 13
svg_test . . . 14
top_results . . . 16

Index **18**

DESpace *DESpace: A package for identifying spatially variable genes*

Description

An intuitive framework for identifying spatially variable genes (SVGs) and differential spatial pattern (DSP) genes via edgeR, one of the most common methods for performing differential expression analyses.

Details

Based on pre-annotated spatial clusters as summarized spatial information, DESpace models gene expression using a negative binomial (NB), via edgeR, with spatial clusters as covariates. SVGs are then identified by testing the significance of spatial clusters, whereas DSP genes are identified by testing the significance of the interaction terms between spatial clusters and conditions (e.g., treatment conditions or time phases).

Our approach assumes that the spatial structure can be summarized by spatial clusters, which should reproduce the key features of the tissue (e.g., white matter and layers in brain cortex). These spatial clusters are therefore taken as proxy for the actual spatial distribution; a significant test of these covariates indicates that space influences gene expression, hence identifying spatially variable genes.

Our model is flexible and robust, and is significantly faster than the most SV methods. Furthermore, to the best of our knowledge, it is the only SV approach that allows: - performing a SV test on each individual spatial cluster, hence identifying the key regions affected by spatial variability; - jointly fitting multiple samples, targeting genes with consistent spatial patterns across replicates.

For an overview of the functionality provided by the package, please see the vignette: `vignette("DESpace", package="DESpace")`

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

svg_test, individual_svg, top_results, FeaturePlot, dsp_test, individual_dsp

---

dsp_test *dsp_test*

Description

'dsp_test' identifies differential spatial pattern (DSP) genes between conditions from spatially-resolved transcriptomics data, provided spatial clusters are available.

Usage

```
dsp_test(
  spe,
  design = NULL,
  cluster_col,
  sample_col,
  condition_col,
  min_counts = 20,
  min_non_zero_spots = 10,
  min_pct_cells = 0.5,
  filter_gene = FALSE,
  filter_cluster = TRUE,
```

```
  verbose = FALSE
)
```

Arguments

`spe`	SpatialExperiment or SingleCellExperiment.
`design`	Matrix or array. Numeric design matrix for a regression-like model created by ‘model.matrix‘ function.
`cluster_col`	Character. Column name of spatial clusters in `colData(spe)`.
`sample_col`	Character. Column name of sample ids in `colData(spe)`. Sample ids must be either a factor or character.
`condition_col`	Character. Column name of condition ids in `colData(spe)`.
`min_counts`	Numeric. Minimum number of counts per sample (across all spots) for a gene to be analyzed.
`min_non_zero_spots`	Numeric. Minimum number of non-zero spots per sample, for a gene to be analyzed.
`min_pct_cells`	Numeric. Minimum percentage of cells required for each cluster to be included in the analysis across the specified conditions. Default value is 0.5 (i.e., 0.5% of total cells per cluster per condition).
`filter_gene`	Logical. If TRUE, `dsp_test` filters genes by requiring them to be expressed in at least ’min_non_zero_spots’ cells and have at least ’min_counts’ counts per sample across all locations.
`filter_cluster`	Logical. When set to TRUE, `dsp_test` excludes clusters that are insufficiently represented in the dataset. Only clusters meeting the ’min_pct_cells’ threshold (i.e., containing at least the specified percentage of cells across all conditions) will be retained for analysis.
`verbose`	Logical. If TRUE, `svg_test` returns two more results: ’DGEGLM’ and ’DGELRT’ objects contain full statistics from ’edgeR::glmFit’ and ’edgeR::glmLRT’.

Value

A list of results:

- "gene_results": a dataframe contains main edgeR test results;

- "estimated_y": a DGEList object contains the estimated common dispersion, which can later be used to speed-up calculation when testing individual clusters.

- "glmFit" (only if `verbose = TRUE`): a DGEGLM object contains full statistics from "edgeR::glmFit".

- "glmLRT" (only if `verbose = TRUE`): a DGELRT object contains full statistics from "edgeR::glmLRT".

See Also

`svg_test`, `individual_svg`, `individual_dsp`, `FeaturePlot`, `top_results`

Examples

```
## Load the example multi-sample multi-group spe object
spe <- muSpaData::Wei22_example()
# Fit the model via \code{\link{dsp_test}} function.
set.seed(123)
results_dsp <- dsp_test(spe = spe,
                        cluster_col = "Banksy_smooth",
                        sample_col = "sample_id",
                        condition_col = "condition",
                        verbose = FALSE)

# dsp_test returns of an object:
# "gene_results": a dataframe contains main edgeR test results.

# We visualize differential results:
head(results_dsp, 3)
```

FeaturePlot *FeaturePlot*

Description

Plot spatial gene expression. This function is a modified version of the `featurePlot` function from 'BayesSpace' R package. In comparison to the original BayesSpace function, this function allows plotting multiple genes simultaneously and drawing an outline around a specified cluster. To draw outlines, the `reconstructShapeDensityImage` function from the 'sosta' R package has been adapted. Compared to the original 'sosta' function, this version allows the use of a SingleCellExperiment object, which cannot be used with 'spatialCoords()'.

Usage

```
FeaturePlot(
  spe,
  feature,
  coordinates = NULL,
  concave_hull = FALSE,
  sf_dim = 200,
  assay.type = "logcounts",
  annotation_cluster = FALSE,
  annotation_title = NULL,
  platform = "Visium",
  cluster_col = NULL,
  cluster = NULL,
  legend_cluster = FALSE,
  legend_exprs = FALSE,
  diverging = FALSE,
  low = NULL,
```

```
  high = NULL,
  mid = NULL,
  color = NULL,
  linewidth = 0.4,
  linecolor = NULL,
  label = FALSE,
  ncol = 3,
  title = FALSE,
  title_size = 10,
  point_size = 0.5
)
```

Arguments

`spe` SpatialExperiment or SingleCellExperiment. If `feature` is specified and is a string, it must exist as a row in the specified assay of `spe`.

`feature` Feature vector used to color each cell. May be the name of a gene/row in an assay of `spe`, or a vector of continuous values.

`coordinates` Column names for the spatial coordinates of cells stored in `colData(spe)`. If specified, these coordinates will be used. If not, the function defaults to using 'row' and 'col' in `colData(spe)` if they exist. Otherwise, it will use `spatialCoords(spe)` if 'spe' is a SpatialExperiment object and `spatialCoords(spe)` is not NULL.

`concave_hull` A logical value (TRUE or FALSE). For Visium or ST platforms, 'concave_hull' is automatically set to TRUE. If TRUE, the function uses `geom_mark_hull` to outline cluster boundaries (recommended for non-discontinuous clusters). If FALSE, the adapted `reconstructShapeDensityImage` is used for complex cluster shapes.

`sf_dim` A numeric value for the x-dimension of the reconstruction (default is 200). A lower value speeds up computation but reduces accuracy. Used only when 'concave_hull' is FALSE.

`assay.type` String indicating which assay in `spe` the expression vector should be taken from.

`annotation_cluster` A logical value. TRUE or FALSE. If TRUE, annotated spatial clusters are plotted alongside expression plots. If FALSE, clusters are not displayed.

`annotation_title` A character string for the title of the annotated spatial clusters. Applied only when 'annotation_cluster' is TRUE.

`platform` A character string specifying the spatial sequencing platform. If "Visium" or "ST", a hexagonal spot layout will be used. Otherwise, points will be plotted.

`cluster_col` Column name of spatial clusters in `colData(spe)`.

`cluster` Names of the spatial clusters used for drawing a boundary around a group of points that belong to the specify cluster. It can be NULL, "all"/"ALL", or a vector of cluster names.

`legend_cluster` A logical value. TRUE of FALSE, indicating whether to plot the legend for the shaped clusters (TRUE), or not (FALSE). Only used when 'cluster_col'

and 'cluster' are specified, and is supported only when 'concave_hull' is set to TRUE.

`legend_exprs` A logical value. TRUE of FALSE, indicating whether to plot the legend for the expression level (TRUE), or not (FALSE).

`diverging` A logical value. If TRUE, uses a diverging color gradient in `FeaturePlot` (e.g., for fold change). If FALSE, uses a sequential gradient (e.g., for expression).

`low`, `mid`, `high` Optional hex codes for low, mid, and high values of the color gradient used for continuous cell values.

`color` Optional hex code to set color of borders around cells. Set to `NA` to remove borders.

`linewidth` The width of the boundary line around the cluster. The default ('0.4') size of the boundary line is one.

`linecolor` The colors of the boundary lines around the cluster. If unspecified, the default color scheme is used.

`label` A logical. TRUE of FALSE. Adding a label and an arrow pointing to a group.

`ncol` The dimensions of the grid to create. By default, 1, if the length of feature equals to 1, and 3, otherwise.

`title` A logical. TRUE or FALSE. If true, the title name of each (subplot) is the gene name.

`title_size` Title font size.

`point_size` Point size.

Value

Returns a ggplot object.

See Also

`svg_test`, `individual_svg`, `top_results`, `dsp_test`, `individual_dsp`

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")

# load pre-computed results (obtained via `svg_test`)
data("results_svg_test", package = "DESpace")

# Visualize the gene expression of the top three genes
feature = results_svg_test$gene_results$gene_id[seq_len(3)]
FeaturePlot(LIBD_subset, feature, coordinates = c("array_row", "array_col"),
            ncol = 3, title = TRUE)
```

individual_dsp *individual_dsp*

Description

DESpace can also be used to reveal the specific areas of the tissue affected by DSP genes; i.e., spatial clusters that are particularly over/under abundant compared to the average signal across conditions. This function can be used to identify SVGs among conditions for each individual cluster.

Usage

```
individual_dsp(
  spe,
  cluster_col,
  sample_col,
  condition_col,
  min_counts = 20,
  min_non_zero_spots = 10,
  min_pct_cells = 0.5,
  filter_gene = TRUE,
  filter_cluster = TRUE
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment.
cluster_col	Character. Column name of spatial clusters in colData(spe).
sample_col	Character. Column name of sample ids in colData(spe). Sample ids must be either a factor or character.
condition_col	Character. Column name of condition ids in colData(spe).
min_counts	Numeric. Minimum number of counts per sample (across all spots) for a gene to be analyzed.
min_non_zero_spots	Numeric. Minimum number of non-zero spots per sample, for a gene to be analyzed.
min_pct_cells	Numeric. Minimum percentage of cells required for each cluster to be included in the analysis across the specified conditions. Default value is 0.5 (i.e., 0.5% of total cells per cluster per condition).
filter_gene	Logical. If TRUE, dsp_test filters genes: genes have to be expressed in at least 'min_non_zero_spots' spots, and a gene requires at least 'min_counts' counts per sample (across all locations).
filter_cluster	Logical. When set to TRUE, dsp_test excludes clusters that are insufficiently represented in the dataset. Only clusters meeting the 'min_pct_cells' threshold (i.e., containing at least the specified percentage of cells across all conditions) will be retained for analysis.

Value

A list of results, with one result per spatial cluster in each element. Specifically, each item in the list is a "gene_results" dataframe which contains main edgeR test results.

See Also

top_results, svg_test, dsp_test, FeaturePlot

Examples

```
# load the input data:
spe <- muSpaData::Wei22_example()
set.seed(123)
results_individual_dsp <- individual_dsp(spe,
                                         cluster_col = "Banksy_smooth",
                                         sample_col = "sample_id",
                                         condition_col = "condition")

# We visualize results for the cluster '3'
results <- results_individual_dsp[['3']]
head(results,3)
```

---

individual_svg *individual_svg*

Description

DESpace can also be used to reveal the specific areas of the tissue affected by SVGs; i.e., spatial clusters that are particularly over/under abundant compared to the average signal. This function can be used to identify SVGs for each individual cluster.

Usage

```
individual_svg(
  spe,
  cluster_col,
  sample_col = "sample_id",
  edgeR_y = NULL,
  min_counts = 20,
  min_non_zero_spots = 10,
  filter_gene = TRUE,
  replicates = FALSE,
  BPPARAM = NULL
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment.
cluster_col	Column name of spatial clusters in colData(spe).
sample_col	Column name of sample ids in colData(spe).
edgeR_y	Pre-estimated dispersion; if it's null, compute dispersion.
min_counts	Minimum number of counts per sample (across all spots) for a gene to be analyzed.
min_non_zero_spots	Minimum number of non-zero spots per sample, for a gene to be analyzed.
filter_gene	A logical. If TRUE, svg_test filters genes: genes have to be expressed in at least 'min_non_zero_spots' spots, and a gene requires at least 'min counts' counts per sample (across all locations).
replicates	Single sample or multi-sample test.
BPPARAM	An optional parameter passed internally to bplapply. We suggest using as many cores as the number of spatial clusters. If unspecified, the script does not run in parallel. Note that parallel coding performs better only when dispersion estimations are not provided beforehand. Moreover, parallelizing the script will increase the memory requirement; if memory is an issue, leave 'BPPARAM' unspecified and, hence, avoid parallelization.

Details

For every spatial cluster we test, edgeR would normally re-compute the dispersion estimates based on the specific design of the test. However, this calculation represents the majority of the overall computing time. Therefore, to speed-up calculations, we propose to use the dispersion estimates which were previously computed for the gene-level tests. This introduces a minor approximation which, in our benchmarks, does not lead to decreased accuracy. If you want to use pre-computed gene-level dispersion estimates, set edgeR_y to 'estimated_y'. Alternatively, if you want to re-compute dispersion estimates (significantly slower, but marginally more accurate option), leave edgeR_y empty.

Value

A list of results, with one result per spatial cluster in each element. Specifically, each item in the list is a "gene_results" dataframe which contains main edgeR test results.

See Also

top_results, svg_test, FeaturePlot

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")
LIBD_subset

# load pre-computed results (obtaines via `svg_test`)
```

```
data("results_svg_test", package = "DESpace")

# svg_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.

# We visualize differential results:
head(results_svg_test$gene_results, 3)

# Individual cluster test: identify SVGs for each individual cluster
# set parallel computing; we suggest using as many cores as the number of spatial clusters.
# Note that parallelizing the script will increase the memory requirement;
# if memory is an issue, leave 'BPPARAM' unspecified and, hence, avoid parallelization.
set.seed(123)
results_individual_svg <- individual_svg(LIBD_subset,
                                         edgeR_y = results_svg_test$estimated_y,
                                         cluster_col = "layer_guess_reordered")

# We visualize results for the cluster 'WM'
results_WM <- results_individual_svg[[7]]
head(results_WM,3)
```

LIBD_subset *Subset from the human DLPFC 10x Genomics Visium dataset of the* spatialLIBD *package*

Description

Subset from the human DLPFC 10x Genomics Visium dataset of the spatialLIBD package

Arguments

LIBD_subset contains a SpatialExperiment-class object, representing a subset of the sample 151673 from the full real data of the spatialLIBD package. Below the code used to subset the original dataset.

Value

A spatial experiment object

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

svg_test, individual_svg

Examples

```
# Connect to ExperimentHub
# ehub <- ExperimentHub::ExperimentHub()
# Download the example spe data
# spe_all <- spatialLIBD::fetch_data(type = "spe", eh = ehub)
# Select one sample only:
# LIBD_subset <- spe_all[, colData(spe_all)$sample_id == '151673']
# Select small set of random genes for faster runtime
# set.seed(123)
# sel_genes <- sample(dim(LIBD_subset)[1],500)
# LIBD_subset <- LIBD_subset[sel_genes,]
# keep_col <- c("array_row","array_col","layer_guess_reordered")
# library(SingleCellExperiment)
# LIBD_subset <- SpatialExperiment(assay = list(counts = assay(LIBD_subset),
#                                               logcounts = logcounts(LIBD_subset)),
#                                  colData = colData(LIBD_subset)[keep_col])
# save(LIBD_subset, file = "./DESpace/data/LIBD_subset.RData")
```

---

results_individual_svg

Results from `individual_svg` *function*

Description

Results from `individual_svg` function

Arguments

`results_individual_svg`
: contains a `list` object, with the results obtained applying `individual_svg` function to an external dataset from the spatialLIBD package. Below the code used to obtain 'results_individual_svg'.

Value

A List of 7 elements - one element for each spatial cluster

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

`svg_test`, `individual_svg`

Examples

```
# load the input data:
# data("LIBD_subset", package = "DESpace")
# LIBD_subset
# load pre-computed results (obtained via `svg_test`)
# data("results_svg_test", package = "DESpace")
# results_svg_test

# Function `individual_svg()` can be used to identify SVGs for each individual cluster.
# Parameter `spatial_cluster` indicates the column names of `colData(spe)`
# containing spatial clusters.
# set.seed(123)
# results_individual_svg <- individual_svg(LIBD_subset,
#                                          edgeR_y = results_svg_test$estimated_y,
#                                          spatial_cluster = "layer_guess_reordered")
# save(results_individual_svg, file = "./DESpace/data/results_individual_svg.RData")
```

`results_svg_test` *Results from* `svg_test` *function*

Description

Results from `svg_test` function

Arguments

`results_svg_test`
: contains a `list` object, with the results obtained applying `svg_test` function to an external dataset from the spatialLIBD package. Below the code used to obtain 'results_svg_test'.

Value

Large List of 2 elements:

- "gene_results": a dataframe contains main edgeR test results;

- "estimated_y": a DGEList object contains the estimated common dispersion,

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

`svg_test`

Examples

```
# load the input data:
# data("LIBD_subset", package = "DESpace")
# LIBD_subset
#
# Fit the model via `svg_test` function.
# Parameter `spe` specifies the input `SpatialExperiment` or `SingleCellExperiment` object,
# while `cluster_col` defines the column names of `colData(spe)` containing spatial clusters.
# To obtain all statistics, set `verbose` to `TRUE`.
#
# set.seed(123)
# results_svg_test <- svg_test(spe = LIBD_subset,
#                                          cluster_col = "layer_guess_reordered",
#                                          verbose = FALSE)
#
# save(results_svg_test, file = "./DESpace/data/results_svg_test.RData")
```

---

svg_test *svg_test*

Description

'svg_test' identifies spatially variable genes (SVGs) from spatially-resolved transcriptomics data, provided spatial clusters are available.

Usage

```
svg_test(
  spe,
  cluster_col,
  sample_col = NULL,
  replicates = FALSE,
  min_counts = 20,
  min_non_zero_spots = 10,
  filter_gene = TRUE,
  verbose = FALSE
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment.
cluster_col	Column name of spatial clusters in colData(spe).
sample_col	Column name of sample ids in colData(spe).
replicates	A logical, indicating whether biological replicates are provided (TRUE) or not (FALSE). If biological replicates are provided, svg_test performs a joint test across all replicates, searching for SVGs with consistent spatial patterns across samples.

`min_counts` Minimum number of counts per sample (across all spots) for a gene to be analyzed.

`min_non_zero_spots` Minimum number of non-zero spots per sample, for a gene to be analyzed.

`filter_gene` A logical. If TRUE, `svg_test` filters genes: genes have to be expressed in at least 'min_non_zero_spots' spots, and a gene requires at least 'min counts' counts per sample (across all locations).

`verbose` A logical. If TRUE, `svg_test` returns two more results: 'DGEGLM' and 'DGELRT' objects contain full statistics from 'edgeR::glmFit' and 'edgeR::glmLRT'.

Details

If 'sample_col' is not specified and 'replicates == FALSE', `svg_test` assumed that data comes from an individual sample, and performs SV testing on it.

If 'sample_col' is provided and 'replicates == FALSE', `svg_test` tests each sample individually and returns a list of results for each sample.

If 'sample_col' is provided and 'replicates == TRUE', `svg_test` performs a joint multi-sample test.

Value

A list of results:

- "gene_results": a dataframe contains main edgeR test results;

- "estimated_y": a DGEList object contains the estimated common dispersion, which can later be used to speed-up calculation when testing individual clusters.

- "glmFit" (only if `verbose = TRUE`): a DGEGLM object contains full statistics from "edgeR::glmFit".

- "glmLRT" (only if `verbose = TRUE`): a DGELRT object contains full statistics from "edgeR::glmLRT".

See Also

`top_results`, `individual_svg`, `FeaturePlot`, `dsp_test`, `individual_dsp`

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")
LIBD_subset

# Fit the model via \code{\link{svg_test}} function.
set.seed(123)
results_svg_test <- svg_test(spe = LIBD_subset,
                             cluster_col = "layer_guess_reordered",
                             verbose = FALSE)

# svg_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.
```

```
# We visualize differential results:
head(results_svg_test$gene_results, 3)
```

`top_results` *top_results*

Description

Filter significant results. `top_results` returns the significant results obtained via `svg_test` and `individual_svg`. It can also be used to merge gene- and cluster-level results into a single object.

Usage

```
top_results(
  gene_results = NULL,
  cluster_results,
  cluster = NULL,
  select = "both",
  high_low = NULL
)
```

Arguments

`gene_results`	Results returned from `svg_test`.
`cluster_results`	Results returned from `individual_svg`.
`cluster`	A character indicating the cluster(s) whose results have to be returned. Results from all clusters are returned by default ("NULL").
`select`	A character indicating what results should be returned ("FDR", "logFC", or "both"). Only used if "cluster_results" are provided. By default ("both"), both FDR and logFC are returned.
`high_low`	A character indicating whether to filter results or not. Only used if "cluster_results" are provided, and one cluster is specified in "cluster" parameter. By default (NULL), all results are returned in a single data.frame. If "high" or "HIGH", we only return SVGs with average abundace in "cluster" higher than in the rest of the tissue (i.e., logFC > 0). If "low" or "LOW", we only return SVGs with average abundace in "cluster" lower than in the rest of the tissue (i.e., logFC < 0). If "both" or "BOTH", then both "high" and "low" results are returned, but in two separate data.frames.

Value

A `data.frame` object or a list of `data.frame` with results.

- When only "cluster_results" is provided, results are reported as a `data.frame` with columns for gene names (gene_id), spatial clusters affected by SV (Cluster), cluster-specific likelihood ratio test statistics (LR), cluster-specific average (across spots) log-2 counts per million (logCPM),

cluster-specific log2-fold changes (logFC), cluster-specific raw p-values (PValue), and Benjamini-Hochberg adjusted p-values (FDR) for each spatial cluster.

- When "gene_results" and "cluster_results" are given, results are reported as a data.frame that merges gene- and cluster-level results.

- If "cluster" is specified, the function returns a subset data.frame for the given cluster, which contains cluster name, gene name, LR, logCPM, logFC, PValue and FDR, ordered by FDR for the specified cluster.

- If "high_low" is set, the function returns a list of data.frame that contains subsets of results for genes with higher and/or lower expression in the given cluster compared to the rest of the tissue.

See Also

svg_test, individual_svg, FeaturePlot, dsp_test, individual_dsp

Examples

```
# load pre-computed results (obtained via `svg_test`)
data("results_svg_test", package = "DESpace")

# svg_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.

# We visualize differential results:
head(results_svg_test$gene_results, 3)

# load pre-computed results (obtained via `individual_svg`)
data("results_individual_svg", package = "DESpace")
# Function `individual_svg()` can be used to identify SVGs for each individual cluster.
# `individual_svg()` returns a list containing the results of individual clusters.
# For each cluster, results are reported as a data.frame,
# where columns For each cluster, results are reported as a data.frame,
# where columns contain gene names (`genes`), likelihood ratio (`LR`),
# log2-fold changes (`logFC`) and adjusted p-value (`FDR`).
#
# Combine gene-and cluster-level results
merge_res = top_results(results_svg_test$gene_results,
                        results_individual_svg)
head(merge_res,3)
# 'select = "FDR"' can be used to visualize adjusted p-values for each spatial cluster.
merge_res = top_results(results_svg_test$gene_results,
                        results_individual_svg, select = "FDR")
head(merge_res,3)
# Specify the cluster of interest and check top genes detected by svg_test.
results_WM_both = top_results(cluster_results = results_individual_svg,
                              cluster = "WM", high_low = "both")
head(results_WM_both$high_genes, 3)
head(results_WM_both$low_genes, 3)
```

Index

∗ **internal**
    DESpace, 2
∗ **spatial plotting functions**
    FeaturePlot, 5

data.frame, *16, 17*
DESpace, 2
DESpace-package (DESpace), 2
dsp_test, *3*, 3, *4*, *7–9*, *15*, *17*

FeaturePlot, *3, 4*, 5, *7*, *9, 10*, *15*, *17*
featurePlot, *5*

geom_mark_hull, *6*

individual_dsp, *3, 4*, *7*, 8, *15*, *17*
individual_svg, *3, 4*, *7*, 9, *11, 12*, *15–17*

LIBD_subset, 11
list, *12, 13*

reconstructShapeDensityImage, *5, 6*
results_individual_svg, 12
results_svg_test, 13

svg_test, *3, 4*, *7*, *9–14*, 14, *15–17*

top_results, *3, 4*, *7*, *9, 10*, *15, 16*, 16