Package 'SpaceMarkers'

May 2, 2026

Type Package

Title Spatial Interaction Markers

Version 2.3.1

BugReports https://github.com/DeshpandeLab/SpaceMarkers/issues

URL https://github.com/DeshpandeLab/SpaceMarkers

Description Spatial transcriptomic technologies have helped to resolve the connection between gene expression and the 2D orientation of tissues relative to each other. However, the limited single-cell resolution makes it difficult to highlight the most important molecular interactions in these tissues. SpaceMarkers, R/Bioconductor software, can help to find molecular interactions, by identifying genes associated with latent space interactions in spatial transcriptomics.

Depends R (>= 4.4.0)

biocViews SingleCell, GeneExpression, Software, Spatial, Transcriptomics

Imports SpatialExperiment, SingleCellExperiment, SummarizedExperiment, S4Vectors, matrixStats, matrixTests, rstatix, spatstat.explore, spatstat.geom, ape, hdf5r, nanoparquet, jsonlite, Matrix, qvalue, stats, utils, methods, ggplot2, reshape2, RColorBrewer, circlize, mixtools, dplyr, readbitmap, rlang, effsize, viridis

Suggests data.table, devtools, knitr, cowplot, rjson, rmarkdown, BiocStyle, BiocFileCache, testthat (>= 3.0.0), CoGAPS, ComplexHeatmap, zellkonverter, anndataR

Enhances BiocParallel

VignetteBuilder knitr

Collate 'AllClasses.R' 'AllGenerics.R' 'SpaceMarkers.R' 'SpaceMarkersExperiment-methods.R' 'data.R' 'findGenesOfInterest.R' 'getInteractingGenes.R' 'getSpatialParameters.R' 'hdutils.R' 'plotLRCircos.R' 'preprocessing.R' 'test-helper.R' 'utils.R'

Config/testthat/edition 3

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

License MIT + file LICENSE

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

git_branch devel

git_last_commit 145f3da

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Atul Deshpande [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5144-6924>),
Orian Stapleton [aut],
Ludmila Danilova [ctb],
Dmitrijs Lvovs [ctb] (ORCID: <https://orcid.org/0009-0003-2152-6853>)

Maintainer Atul Deshpande <adeshpande@jhu.edu>

Contents

`.apply_sme_filters` *Filter an SME by gene expression / barcode match and (re)compute spatial params*

Description

Filter an SME by gene expression / barcode match and (re)compute spatial params

Usage

```
.apply_sme_filters(
  sme,
  genes = NULL,
  min.gene.expr = 10,
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  resolution = "fullres"
)
```

Value

The filtered `SpaceMarkersExperiment` (genes/spots restricted, `spatial_params(sme)` populated when not already set).

.calc_IM_scores *Calculate interaction scores for a specific pattern pair*

Description

This function calculates interaction scores for a specific pattern pair using the `.classify_spots` function to determine the region of each spot.

Usage

```
.calc_IM_scores(
  data,
  pat_hotspots,
  influence_hotspots,
  patternpair,
  avoid_confounders = FALSE,
  ...
)
```

Arguments

`data`	A numeric matrix with genes as rows and barcodes as columns.
`pat_hotspots`	A data frame with pattern hotspots, containing columns for x, y, and barcode.
`influence_hotspots`	A data frame with influence hotspots, containing columns for x, y, and barcode.
`patternpair`	A character vector of length 2 specifying the pattern pair to analyze.
`avoid_confounders`	Logical (default=FALSE) indicating whether to avoid confounding effects due to colocalization.
`...`	Additional parameters to pass to lower level functions.

Value

A data frame with interaction scores for the specified pattern pair.

.calc_threshold *Compute the threshold for identifying outlier values or hotspots*

Description

This function computes the threshold for identifying outlier values or hotspots by fitting a normal mixture model to the data.

Usage

```
.calc_threshold(df, minval = 0.01, maxval = 0.99, method = c("abs", "pct"))
```

Arguments

df	A vector containing pattern values
minval	Minimum value for quantile threshold
maxval	Maximum value for quantile threshold
method	Method to use for threshold calculation. Options are "abs" for absolute (default) and "pct" for percentile.

Value

A list containing the computed thresholds

.directed_SpaceMarkers *Directed SpaceMarkers workflow*

Description

Internal function to run the directed SpaceMarkers analysis.

Usage

```
.directed_SpaceMarkers(
  features,
  data,
  genes = NULL,
  min.gene.expr = 10,
  resolution = c("fullres", "lowres", "hires"),
  version = NULL,
  h5filename = "filtered_feature_bc_matrix.h5",
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  ...
)
```

Arguments

features	A path to a csv features file. Ignored when x is provided.
data	A path to a 10X Visium directory. Ignored when x is provided.
genes	Optional character vector of genes to retain. If NULL, genes are filtered by min.gene.expr.
min.gene.expr	Minimum summed expression threshold for retaining genes when genes is NULL.
resolution	Resolution passed to load10XCoords(). One of "fullres", "lowres", or "hires".
version	Optional Spaceranger version passed to load10XCoords().
h5filename	Name of the 10X H5 expression file passed to load10XExpr().
spatialDir	Name of the spatial subdirectory passed to get_spatial_parameters().
pattern	Name of the JSON scale-factor file passed to get_spatial_parameters().
sigma	Optional numeric sigma passed to get_spatial_parameters().
threshold	Numeric threshold passed to get_spatial_parameters().
...	Additional arguments passed only to get_pairwise_interacting_genes() or calculate_gene_scores_directed().

Value

Interaction scores for the directed workflow.

.directed_SpaceMarkers_sme

Directed workflow on a SpaceMarkersExperiment

Description

Directed workflow on a SpaceMarkersExperiment

Usage

```
.directed_SpaceMarkers_sme(
  sme,
  genes = NULL,
  min.gene.expr = 10,
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  resolution = "fullres",
  lr_pairs = NULL,
  returnSME = TRUE,
  ...
)
```

Value

If returnSME = TRUE, the input SME with directed hotspots, gene scores, and LR scores attached; otherwise the legacy data frame produced by the underlying pipeline.

`.find_genes_of_interest` *.find_genes_of_interest Identify genes associated with pattern interaction. This function identifies genes exhibiting significantly higher values of testMat in the Interaction region of the two patterns compared to regions with exclusive influence from either pattern. It uses Kruskal-Wallis test followed by posthoc analysis using Dunn's Test to identify the genes.*

Description

.find_genes_of_interest Identify genes associated with pattern interaction. This function identifies genes exhibiting significantly higher values of testMat in the Interaction region of the two patterns compared to regions with exclusive influence from either pattern. It uses Kruskal-Wallis test followed by posthoc analysis using Dunn's Test to identify the genes.

Usage

```
.find_genes_of_interest(testMat, goodGenes, region, fdr.level=0.05,
       analysis=c("enrichment","overlap"),...)
```

Arguments

`testMat`	A matrix of counts with cells as columns and genes as rows
`goodGenes`	A vector of user specified genes expected to interact a priori. The default for this is NULL as the function can find these genes itself
`region`	A data frame of the reference pattern regions that overlap with the other patterns
`fdr.level`	False Discovery Rate. The default value is 0.05.
`analysis`	a character string that specifies the type of analysis to carry out, whether overlap or enrichment.
`...`	Additional arguments to be passed to lower level functions

Value

a list of genes exhibiting significantly higher values of testMat in the Interaction region of the two #' patterns compared to regions with exclusive influence from either pattern.

`.get_BTME_features` *.get_BTME_features Load features BayesTME object*

Description

.get_BTME_features Load features BayesTME object

Usage

```
.get_BTME_features(hf)
```

Value

Numeric matrix of BayesTME cell-type counts (rows = barcodes, cols = cell types).

.get_cogaps_features *.get_cogaps_features Load features CoGAPS object*

Description

.get_cogaps_features Load features CoGAPS object

Usage

```
.get_cogaps_features(obj)
```

Value

Numeric matrix of CoGAPS sample factors (rows = barcodes, cols = patterns).

.get_csv_features *.get_csv_features Load features from dataframe*

Description

.get_csv_features Load features from dataframe

Usage

```
.get_csv_features(obj)
```

Value

data.frame of features with barcodes as rownames.

.get_seurat_features *.get_seurat_features Load features Seurat object*

Description

.get_seurat_features Load features Seurat object

Usage

```
.get_seurat_features(obj)
```

Value

data.frame of `_Feature`-suffixed metadata columns extracted from `slot(obj, "meta.data")`.

.get_spe_features *.get_spe_features Extract spatial features from a SpatialExperiment object's colData*

Description

.get_spe_features Extract spatial features from a SpatialExperiment object's colData

Usage

```
.get_spe_features(obj)
```

Value

data.frame of numeric `colData` columns (rows = spots) with standard SE/SPE bookkeeping columns removed.

.infer_method *inferMethod Infer the method used to obtain spatial features*

Description

inferMethod Infer the method used to obtain spatial features

Usage

```
.infer_method(spObject, method)
```

Value

Character of length one: one of `"SpatialExperiment"`, `"BayesTME"`, `"CoGAPS"`, `"Seurat"`, or `"CSV"`.

.pick_image *.pick_image*

Description

The function picks the appropriate histology image file from the spatial directory based on the specified resolution.

Usage

```
.pick_image(sp_dir, res)
```

Arguments

`sp_dir`	path to the spatial directory
`res`	a character string specifying the resolution of the image

Value

a character string of the image file name

`.read_format` *readFormat Reads a format into an R object*

Description

readFormat Reads a format into an R object

Usage

```
.read_format(path)
```

Value

The deserialized R object: result of `readRDS` for .rds, an open `H5File` for .h5ad, a `data.frame` for .csv.

`.row_t_test` *Perform row-wise t-tests from scratch*

Description

This function iterates over the rows of a matrix and performs a t-test comparing two groups of columns. It calculates the t-statistic, p-value, and sample sizes without relying on `stats::t.test()` for the core logic.

Usage

```
.row_t_test(in.data, region, min_bins = 50, ...)
```

Arguments

`in.data`	A numeric matrix. Rows represent features, columns represent samples.
`region`	A factor or vector indicating the group membership for each column of `in.data`. Must have exactly two levels/unique values. Its length must equal `ncol(in.data)`.
`min_bins`	Minimum number of non-missing observations required in each group to perform the t-test.
`...`	Additional parameters to pass to the t-test function.

Value

A matrix with rows corresponding to the features and columns: - `statistic`: The calculated t-statistic. - `p.value`: The calculated two-sided p-value. - `n1`: Number of non-missing observations in group 1 for that row. - `n2`: Number of non-missing observations in group 2 for that row.

.undirected_SpaceMarkers *Undirected SpaceMarkers workflow*

Description

Internal function to run the undirected SpaceMarkers analysis.

Usage

```
.undirected_SpaceMarkers(
  features,
  data,
  cpus = 1,
  genes = NULL,
  min.gene.expr = 10,
  resolution = c("fullres", "lowres", "hires"),
  version = NULL,
  h5filename = "filtered_feature_bc_matrix.h5",
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  ...
)
```

Arguments

features	A path to a csv features file. Ignored when x is provided.
data	A path to a 10X Visium directory. Ignored when x is provided.
cpus	Number of workers used in the undirected workflow passed to get_pairwise_interacting_genes()
genes	Optional character vector of genes to retain. If NULL, genes are filtered by min.gene.expr.
min.gene.expr	Minimum summed expression threshold for retaining genes when genes is NULL.
resolution	Resolution passed to load10XCoords(). One of "fullres", "lowres", or "hires".
version	Optional Spaceranger version passed to load10XCoords().
h5filename	Name of the 10X H5 expression file passed to load10XExpr().
spatialDir	Name of the spatial subdirectory passed to get_spatial_parameters().
pattern	Name of the JSON scale-factor file passed to get_spatial_parameters().
sigma	Optional numeric sigma passed to get_spatial_parameters().
threshold	Numeric threshold passed to get_spatial_parameters().
...	Additional arguments passed only to get_pairwise_interacting_genes() or calculate_gene_scores_directed().

Value

Interaction scores for the undirected workflow.

.undirected_SpaceMarkers_sme *Undirected workflow on a SpaceMarkersExperiment*

Description

Undirected workflow on a SpaceMarkersExperiment

Usage

```
.undirected_SpaceMarkers_sme(
  sme,
  cpus = 1,
  genes = NULL,
  min.gene.expr = 10,
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  resolution = "fullres",
  returnSME = TRUE,
  ...
)
```

Value

If `returnSME = TRUE`, the input SME with hotspot, overlap, and interaction results attached; otherwise the legacy data frame produced by the underlying pipeline.

.wrap_directed_result *Wrap legacy directed result as SME (when file-path input used with returnSME=TRUE)*

Description

Wrap legacy directed result as SME (when file-path input used with returnSME=TRUE)

Usage

```
.wrap_directed_result(
  IMscores,
  features,
  data,
  resolution,
  version,
  h5filename
)
```

Value

A SpaceMarkersExperiment carrying the legacy directed result (IMScores, hotspots, LR scores) in its spacemarkers slot.

.wrap_undirected_result *Wrap legacy undirected result as SME (when file-path input used with returnSME=TRUE)*

Description

Wrap legacy undirected result as SME (when file-path input used with returnSME=TRUE)

Usage

```
.wrap_undirected_result(
  IMScores,
  features,
  data,
  resolution,
  version,
  h5filename
)
```

Value

A SpaceMarkersExperiment carrying the legacy undirected result (IMScores, hotspots, params) in its spacemarkers slot.

---

add_features *Add spatial features to a SpaceMarkersExperiment*

Description

Adds spatial features (patterns) to a SpaceMarkersExperiment object's colData after object initialization. When spots differ between the SME and the features, only the common spots are kept and a warning is emitted.

Usage

```
add_features(sme, features, ...)
```

Arguments

sme	A SpaceMarkersExperiment object.
features	A matrix or data.frame of features (rows = spots, cols = patterns), a file path, or an R object accepted by get_spatial_features.
...	Additional arguments passed to get_spatial_features() when features is a file path or object.

Value

The SpaceMarkersExperiment with features added to colData and pattern_names updated in the spacemarkers slot.

Examples

```
set.seed(1)
nb <- 20
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(10 * nb, 2), 10, nb,
        dimnames = list(paste0("G", 1:10), paste0("s", seq_len(nb))))),
    spatialCoords = matrix(runif(2 * nb), nb, 2,
        dimnames = list(paste0("s", seq_len(nb)), c("y", "x"))))
feat <- data.frame(Pattern_1 = runif(nb), Pattern_2 = runif(nb),
                   row.names = paste0("s", seq_len(nb)))
sme <- add_features(sme, feat)
spatial_patterns(sme)[1:3, ]
```

---

analysis_type *Access or set the analysis type on a SpaceMarkersExperiment*

Description

Read (analysis_type(x)) or write (analysis_type(x) <- value) the analysis-type tag stored in x@spacemarkers$analysis. Valid values are "undirected", "directed", or "both". The setter delegates validation to setValidity("SpaceMarkersExperiment", ...).

Usage

```
analysis_type(x, ...)

analysis_type(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
analysis_type(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
analysis_type(x) <- value
```

Arguments

x	A SpaceMarkersExperiment object.
...	Not used.
value	Character: "undirected", "directed", or "both".

Value

For the getter, a character string or NULL. For the setter, the modified SpaceMarkersExperiment.

A character string: "undirected", "directed", "both", or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
analysis_type(sme) <- "undirected"
analysis_type(sme)
```

---

as-SingleCellExperiment-SpaceMarkersExperiment

Coercion from SingleCellExperiment to SpaceMarkersExperiment

Description

AnnData-sourced objects (e.g. via `zellkonverter::readH5AD()` or `anndataR::as_SingleCellExperiment()`) typically arrive as `SingleCellExperiment` with spatial coordinates stored as a reducedDim, commonly named "spatial". This coerce method promotes that reducedDim into `spatialCoords()` and wraps the result as an SME, so a full AnnData -> SME path reads:

Details

```
sme <- as(zellkonverter::readH5AD(path), "SpaceMarkersExperiment")
```

If no suitable spatial reducedDim is found, the SME is built with empty `spatialCoords`; the caller can populate them afterwards.

Value

A `SpaceMarkersExperiment` carrying the same assays / colData / rowData as `from`, with the spatial reducedDim promoted to `spatialCoords()` when present.

---

as-SpatialExperiment-SpaceMarkersExperiment

Coercion from SpatialExperiment to SpaceMarkersExperiment

Description

Coercion from SpatialExperiment to SpaceMarkersExperiment

Value

A `SpaceMarkersExperiment` wrapping `from`, with an empty `spacemarkers` slot ready to be populated by the pipeline.

`calculate_gene_scores_directed` *Calculate interaction scores for all pattern pairs*

Description

This function calculates interaction scores for all pattern pairs using the `.calc_IM_scores` function. It can run in parallel if BiocParallel is available.

Usage

```
calculate_gene_scores_directed(
  data,
  pat_hotspots = NULL,
  influence_hotspots = NULL,
  pattern_pairs = NULL,
  ...
)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_gene_scores_directed(
  data,
  pat_hotspots = NULL,
  influence_hotspots = NULL,
  pattern_pairs = NULL,
  ...
)

## S4 method for signature 'ANY'
calculate_gene_scores_directed(
  data,
  pat_hotspots = NULL,
  influence_hotspots = NULL,
  pattern_pairs = NULL,
  ...
)
```

Arguments

`data`	A numeric matrix with genes as rows and barcodes as columns.
`pat_hotspots`	A data frame with pattern hotspots, containing columns for x, y, and barcode.
`influence_hotspots`	A data frame with influence hotspots, containing columns for x, y, and barcode.
`pattern_pairs`	A data frame with pattern pairs to calculate interaction scores for. If NULL, all combinations of patterns in `pat_hotspots` will be used. If provided, it should have two columns with pattern names. Each row should represent a pair of patterns for which interaction scores will be calculated.
`...`	Additional parameters to pass to lower level functions.

Value

A data frame with interaction scores for all pattern pairs.

Examples

```
set.seed(1)
nb <- 30
spHotspots <- data.frame(
    barcode = paste0("s", seq_len(nb)),
    x = runif(nb), y = runif(nb),
    Pattern_1 = ifelse(seq_len(nb) <= 10, "Pattern_1", NA),
    Pattern_2 = ifelse(seq_len(nb) > 10 & seq_len(nb) <= 20,
                       "Pattern_2", NA),
    Pattern_3 = ifelse(seq_len(nb) > 20, "Pattern_3", NA))
spInfl <- spHotspots
counts <- matrix(rpois(5 * nb, 3), nrow = 5,
    dimnames = list(paste0("G", 1:5), spHotspots$barcode))
pattern_pairs <- t(utils::combn(c("Pattern_1","Pattern_2","Pattern_3"), 2))
calculate_gene_scores_directed(counts, spHotspots, spInfl,
                               pattern_pairs = pattern_pairs)
```

`calculate_gene_set_score` *calculate_gene_set_score*

Description

Calculate the mean interaction score for a set of genes

Usage

```
calculate_gene_set_score(
  IMscores,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_gene_set_score(
  IMscores,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)

## S4 method for signature 'ANY'
calculate_gene_set_score(
  IMscores,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)
```

Arguments

IMscores	A matrix of interaction scores
gene_sets	A list of gene sets, where each set is a vector of gene names
weighted	Logical; if TRUE, gene scores are weighted by their occurrence in multiple gene sets
method	Character; specifies the aggregation method for gene set scores. Options are "geometric_mean" or "arithmetic_mean"

Details

This function computes mean interaction scores for given gene sets across cell interactions. It supports both geometric and arithmetic means, and can weight gene contributions based on their presence in multiple gene sets.

Value

A matrix of mean interaction scores for genes in each gene set, with attributes for log p-value sums and number of genes for later fisher combination

Examples

```
set.seed(1)
IMscores <- data.frame(
    gene = rep(paste0("G", 1:6), 2),
    effect_size = runif(12),
    p.value = runif(12),
    cell_interaction = rep(c("A_near_B", "B_near_A"), each = 6))
gene_sets <- list(set1 = c("G1","G2","G3"), set2 = c("G4","G5","G6"))
calculate_gene_set_score(IMscores, gene_sets = gene_sets,
                         method = "arithmetic_mean")
```

---

calculate_gene_set_specificity

calculate_gene_set_specificity

Description

This function computes specificity scores for given gene sets across cell types or spatial patterns. It uses fold-change scores and p-values to weight gene contributions, and supports both geometric and arithmetic means.

Usage

```
calculate_gene_set_specificity(
  data,
  spPatterns = NULL,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)
```

```
## S4 method for signature 'SpaceMarkersExperiment'
calculate_gene_set_specificity(
  data,
  spPatterns = NULL,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)

## S4 method for signature 'ANY'
calculate_gene_set_specificity(
  data,
  spPatterns = NULL,
  gene_sets = NULL,
  weighted = TRUE,
  method = c("geometric_mean", "arithmetic_mean")
)
```

Arguments

data	A numeric matrix or data frame of gene expression values (genes x samples).
spPatterns	A data frame or matrix containing spatial pattern information, with columns for cell types and optionally "x", "y", "barcode".
gene_sets	A named list of character vectors, where each vector contains gene names for a gene set.
weighted	Logical; if TRUE, gene scores are weighted by their occurrence in multiple gene sets.
method	Character; specifies the aggregation method for gene set scores. Options are "geometric_mean" or "arithmetic_mean".

Details

Calculate Gene Set Specificity Scores

- Genes not present in the data are excluded.
- Genes with all zero expression are removed.
- Fold-change scores and p-values are calculated using `.calculate_all_fc_scores`.
- Scores are normalized and weighted by p-value significance.
- For each gene set, scores are aggregated using the specified method and gene weights.

Value

A numeric matrix of gene set specificity scores (gene sets x cell types).

Examples

```
set.seed(1)
nb <- 30
counts <- matrix(rpois(6 * nb, 3), nrow = 6,
    dimnames = list(paste0("G", 1:6), paste0("s", seq_len(nb))))
spPatterns <- data.frame(
    barcode = paste0("s", seq_len(nb)),
```

```
    x = runif(nb), y = runif(nb),
    A = runif(nb), B = runif(nb))
gene_sets <- list(set1 = c("G1","G2"), set2 = c("G4","G5"))
calculate_gene_set_specificity(counts, spPatterns, gene_sets,
                               method = "arithmetic_mean")
```

`calculate_influence` *Compute the spatial influence of a spatial feature*

Description

This function computes the spatial influence of a specified pattern

Usage

```
calculate_influence(spPatterns, optParams = NULL, ...)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_influence(spPatterns, optParams = NULL, ...)

## S4 method for signature 'data.frame'
calculate_influence(spPatterns, optParams = NULL, ...)
```

Arguments

`spPatterns`	A data frame containing x, y coordinates and pattern name
`optParams`	A data frame with optimal parameters for the pattern
`...`	Additional parameters for the Smooth function

Value

A data frame with the spatial influence of the specified pattern

Examples

```
set.seed(1)
spPatterns <- data.frame(
    barcode = paste0("s", 1:50),
    x = runif(50), y = runif(50),
    Pattern_1 = runif(50), Pattern_2 = runif(50))
optParams <- matrix(c(0.1, 4, 0.1, 4), nrow = 2,
    dimnames = list(c("sigmaOpt", "threshOpt"),
                    c("Pattern_1", "Pattern_2")))
inf <- calculate_influence(spPatterns, optParams)
head(inf)
```

calculate_lr_scores *calculate_lr_scores*

Description

Calculate L-R pair scores using Fisher's method

Usage

```
calculate_lr_scores(
  ligand_scores,
  receptor_scores = NULL,
  lr_pairs = NULL,
  ligand_test = NULL,
  method = "geometric_mean",
  weighted = TRUE
)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_lr_scores(
  ligand_scores,
  receptor_scores = NULL,
  lr_pairs = NULL,
  ligand_test = NULL,
  method = "geometric_mean",
  weighted = TRUE
)

## S4 method for signature 'ANY'
calculate_lr_scores(
  ligand_scores,
  receptor_scores = NULL,
  lr_pairs = NULL,
  ligand_test = NULL,
  method = "geometric_mean",
  weighted = TRUE
)
```

Arguments

ligand_scores	Output from getGeneSetScore for ligands
receptor_scores	Output from getGeneSetScore for receptors
lr_pairs	Data frame with columns 'ligand' and 'receptor'
ligand_test	Character; specifies the type of test for ligand overexpression. Options are "greater" or "two.sided"
method	Character; specifies the aggregation method for L-R scores. Options are " geometric_mean" or "arithmetic_mean"
weighted	Logical; if TRUE, L-R scores are weighted by their occurrence in multiple L-R pairs

Details

This function computes L-R pair scores by combining ligand and receptor overexpression scores using either geometric or arithmetic mean. It can also weight L-R pairs based on their presence in multiple pairs to reduce bias from promiscuous ligands or receptors.

Value

Data frame with L-R scores and p-values

Examples

```
set.seed(1)
lr_pairs <- data.frame(
    ligand          = c("L1", "L2"),
    receptor        = c("R1", "R2"),
    ligand.symbol   = c("L1", "L2"),
    receptor.symbol = c("R1", "R2"),
    row.names       = c("L1_R1", "L2_R2"))
ls <- matrix(runif(4), 2, 2,
             dimnames = list(rownames(lr_pairs),
                             c("A_near_B", "B_near_A")))
rs <- matrix(runif(4), 2, 2,
             dimnames = list(rownames(lr_pairs), c("A", "B")))
calculate_lr_scores(ls, rs, lr_pairs = lr_pairs,
                    method = "arithmetic_mean", weighted = FALSE)
```

---

calculate_overlap_directed

calculate_overlap_directed

Description

Calculate the overlap scores between patterns in hotspots

Usage

```
calculate_overlap_directed(
  pat_hotspots,
  influence_hotspots = NULL,
  patternList = NULL,
  method = c("relative-abundance", "differential-abundance", "absolute")
)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_overlap_directed(
  pat_hotspots,
  influence_hotspots = NULL,
  patternList = NULL,
  method = c("relative-abundance", "differential-abundance", "absolute")
)

## S4 method for signature 'data.frame'
```

```
calculate_overlap_directed(
  pat_hotspots,
  influence_hotspots = NULL,
  patternList = NULL,
  method = c("relative-abundance", "differential-abundance", "absolute")
)
```

Arguments

pat_hotspots	A data frame with columns x, y, barcode and pattern names
influence_hotspots	A data frame with columns x, y, barcode and pattern names
patternList	A character vector of pattern names to calculate overlap scores for. If NULL, all patterns in pat_hotspots and influence_hotspots will be used.
method	The method to calculate overlapping abundance scores. Options are "relative-abundance", "differential-abundance" and "absolute"

Details

The function calculates the overlap scores between patterns hotspots using the specified method. The default method is "relative-abundance"

Value

A data frame with columns pattern, influence and overlapping abundance

Examples

```
hotspots <- data.frame(x = c(1,2,3,4,5),
                       y = c(1,2,3,4,5),
                       barcode = c("A","B","C","D","E"),
                       pattern1 = c(1,0,1,0,1),
                       pattern2 = c(1,1,0,0,1))
influence_hotspots <- data.frame(x = c(1,2,3,4,5),
                      y = c(1,2,3,4,5),
                     barcode = c("A","B","C","D","E"),
                     pattern1 = c(0,1,1,0,0),
                     pattern2 = c(0,1,0,1,1))
calculate_overlap_directed(pat_hotspots = hotspots, influence_hotspots = influence_hotspots)
```

---

calculate_overlap_undirected

calculate_overlap_undirected

Description

Calculate the overlap scores between patterns in hotspots

Usage

```
calculate_overlap_undirected(
  hotspots,
  patternList = NULL,
 method = c("Szymkiewicz-Simpson", "Jaccard", "Sorensen-Dice", "Ochiai", "absolute")
)

## S4 method for signature 'SpaceMarkersExperiment'
calculate_overlap_undirected(
  hotspots,
  patternList = NULL,
 method = c("Szymkiewicz-Simpson", "Jaccard", "Sorensen-Dice", "Ochiai", "absolute")
)

## S4 method for signature 'data.frame'
calculate_overlap_undirected(
  hotspots,
  patternList = NULL,
 method = c("Szymkiewicz-Simpson", "Jaccard", "Sorensen-Dice", "Ochiai", "absolute")
)
```

Arguments

hotspots	A data frame with columns x, y, barcode and pattern names
patternList	A character vector of pattern names to calculate overlap scores for
method	The method to calculate overlap scores. Options are "Szymkiewicz-Simpson", "Jaccard", "Sorensen-Dice", "Ochiai" and "absolute"

Details

The function calculates the overlap scores between patterns hotspots using the specified method. The default method is "Szymkiewicz-Simpson" overlap coefficient.

Value

A data frame with columns pattern1, pattern2 and overlapScore

Examples

```
hotspots <- data.frame(x = c(1,2,3,4,5),
                       y = c(1,2,3,4,5),
                       barcode = c("A","B","C","D","E"),
                       pattern1 = c("pattern1",NA,"pattern1",NA,"pattern1"),
                       pattern2 = c("pattern2","pattern2",NA,NA,"pattern2"))
calculate_overlap_undirected(hotspots)
calculate_overlap_undirected(hotspots, c("pattern1","pattern2"))
```

calculate_thresholds *Compute the thresholds for all columns in a data frame*

Description

This function computes the thresholds for all columns in a data frame. The data frame could be an spPatterns object or an spInfluence object.

Usage

```
calculate_thresholds(df, minvals = 0.01, maxvals = 0.99, ...)
```

Arguments

df	A data frame with pattern values (optionally with x, y, barcode columns)
minvals	Minimum value for quantile threshold
maxvals	Maximum value for quantile threshold
...	Additional parameters to pass to lower level functions

Value

A list containing the computed thresholds for each pattern

Examples

```
set.seed(1)
spPatterns <- data.frame(
    barcode = paste0("s", 1:50),
    x = runif(50), y = runif(50),
    Pattern_1 = rnorm(50), Pattern_2 = rnorm(50))
calculate_thresholds(spPatterns)
```

create_lr_dataframe *Combine LR / ligand / receptor score matrices into a long table*

Description

Take the three matrices produced by calculate_lr_scores + calculate_gene_set_score + calculate_gene_set_specificity and reshape them into a single long-format data frame keyed by interaction and source/target cell-type pair. Ligand and receptor gene names are pulled from lrpairs; multi-gene complexes are flattened with underscores.

Column names in ligand_scores that start with near_ are converted to to_ so they line up with the LR-score columns.

Missing LR scores are kept as NA (not coerced to 0). Callers choosing to plot zero-score interactions should filter or impute explicitly.

Usage

```
create_lr_dataframe(
  lrscores,
  ligand_scores,
  receptor_scores,
  lrpairs,
  complex_sep = ", "
)
```

Arguments

`lrscores`	Numeric matrix of LR scores. Rows are interaction IDs (matching `rownames(lrpairs)`); columns are `source_to_target` cell-type pairs (e.g. `"B_to_Plasma"`).
`ligand_scores`	Numeric matrix of ligand scores; same rows as `lrscores`; columns may be `near_` or `to_`-style and are normalised to `to_`.
`receptor_scores`	Numeric matrix of receptor scores. Rows match `lrscores`; columns are single cell-type names (e.g. `"Plasma"`).
`lrpairs`	Data frame with `ligand` and `receptor` columns and row names equal to interaction IDs.
`complex_sep`	Separator used inside `lrpairs$ligand` and `lrpairs$receptor` for multi-gene complexes (e.g. `", "`). Replaced with `"_"` in the output.

Value

A data frame with one row per interaction × source/target pair, containing:

`source_cell_type` Source cell type.

`ligand` Ligand gene(s), underscore-separated.

`receptor` Receptor gene(s), underscore-separated.

`target_cell_type` Target cell type.

`ligand_score` From `ligand_scores`.

`receptor_score` From `receptor_scores`.

`score` LR score from `lrscores`.

`interaction` Interaction ID (rowname of `lrpairs`).

`source_to_target` Convenience `paste0(source_cell_type, "_to_", target_cell_type)`.

Note

Cell-type names must not contain the substrings `"_to_"` or `"_and_"`; the function parses cell types out of column-name keys built from those separators.

Examples

```
lrpairs <- data.frame(
    ligand   = c("L1", "L2"),
    receptor = c("R1, R2", "R3"),
    row.names = c("L1_R1_R2", "L2_R3"))
lr <- matrix(c(0.4, 0.7, 0.1, NA),
             nrow = 2, dimnames = list(rownames(lrpairs),
```

```
                                        c("A_to_B", "B_to_A")))
lig <- matrix(c(0.5, 0.6, 0.2, 0.3),
              nrow = 2, dimnames = list(rownames(lrpairs),
                                        c("A_near_B", "B_near_A")))
rec <- matrix(c(0.7, 0.8, 0.9, 0.6),
              nrow = 2, dimnames = list(rownames(lrpairs),
                                        c("A", "B")))
create_lr_dataframe(lr, lig, rec, lrpairs)
```

---

curated_genes *Curated Genes for example purposes*

Description

A vector with genes selected based on previous runs of SpaceMarkers on the Visium 10x breast ductal carcinoma spatial transcriptomics dataset

Format

A vector with 114 pre-selected genes

Value

a vector of genes

---

directed_scores *Access directed interaction scores on a SpaceMarkersExperiment*

Description

Read (`directed_scores(x)`) or write (`directed_scores(x) <- value`) the data.frame of directed interaction scores stored in `x@spacemarkers$results$directed_scores`.

Usage

```
directed_scores(x, ...)

directed_scores(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
directed_scores(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
directed_scores(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A data.frame of directed interaction scores.

Value

For the getter, a data.frame of directed scores or NULL. For the setter, the modified SpaceMarkersExperiment.

A data.frame of directed interaction scores, or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
directed_scores(sme) <- data.frame(
    gene = paste0("G", 1:5),
    cell_interaction = "Pattern_1->Pattern_5",
    effect_size = runif(5))
head(directed_scores(sme))
```

---

find_all_hotspots *Find hotSpots for all spatial patterns*

Description

Convenience function to find hotspots for all spatial patterns

Usage

```
find_all_hotspots(
  spPatterns,
  params = NULL,
  outlier = "positive",
  nullSamples = 1000,
  includeSelf = TRUE,
  ...
)

## S4 method for signature 'SpaceMarkersExperiment'
find_all_hotspots(
  spPatterns,
  params = NULL,
  outlier = "positive",
  nullSamples = 1000,
  includeSelf = TRUE,
  ...
)

## S4 method for signature 'data.frame'
find_all_hotspots(
  spPatterns,
  params = NULL,
  outlier = "positive",
  nullSamples = 1000,
```

```
  includeSelf = TRUE,
  ...
)
```

Arguments

spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
params	a named vector of the optimal sigma and threshold for a given spatial pattern. The names are should be 'sigmaOpt' and 'threshOpt'. The default value is NULL.
outlier	a character string specifying whether to apply the outlier threshold to the kernel density distribution in a one-sided manner (specify 'positive' the default) or in a two sided manner (specify 'two.sided').
nullSamples	a numeric values specifying the number of spatial patterns to randomly sample for a null distribution.
includeSelf	a logic value specifying whether to consider the spatial influence the pattern has on surrounding regions only (set to FALSE), or whether to also consider the influence of the pattern itself (set to TRUE , the default).
...	Arguments passed to methods

Value

A data.frame with one column per pattern containing the pattern name where the spot is a hotspot and NA otherwise, plus the original barcode/x/y columns.

Examples

```
set.seed(1)
spPatterns <- data.frame(
    barcode = paste0("s", 1:50),
    x = runif(50), y = runif(50),
    Pattern_1 = runif(50), Pattern_2 = runif(50))
hs <- find_all_hotspots(spPatterns, nullSamples = 50)
head(hs)
```

---

find_hotspots_gmm *Find hotspots for all patterns or influences based on values*

Description

Convenience function to find hotspots for all spatial patterns or influence dataframes based on provided thresholds

Usage

```
find_hotspots_gmm(df, threshold = 0.1, ...)

## S4 method for signature 'SpaceMarkersExperiment'
find_hotspots_gmm(
  df,
  threshold = 0.1,
  ...,
  type = c("pattern", "influence"),
  minvals = NULL,
  maxvals = NULL
)

## S4 method for signature 'data.frame'
find_hotspots_gmm(df, threshold = 0.1, ...)
```

Arguments

df	A data frame with pattern values (optionally with x, y, barcode columns)
threshold	a scalar or vector of thresholds for each column in the data frame. Either user provided or the output of @calculate_thresholds
...	Additional parameters to pass to lower level functions
type	Character; one of "pattern" or "influence". Selects which SME slot to operate on.
minvals, maxvals	Numeric thresholds passed to calculate_thresholds when the default threshold = 0.1 is used.

Value

a data frame with the same dimensions as the input data frame.

Examples

```
set.seed(1)
spPatterns <- data.frame(
    barcode = paste0("s", 1:50),
    x = runif(50), y = runif(50),
    Pattern_1 = rnorm(50), Pattern_2 = rnorm(50))
thr <- calculate_thresholds(spPatterns)
hs  <- find_hotspots_gmm(spPatterns, threshold = thr)
head(hs)
```

---

find_pattern_hotspots *Identify hotspots of spatial pattern influence*

Description

This function calculates 'hotspots' which are regions of high spatial influence based on an outlier threshold from a null distribution.

Usage

```
find_pattern_hotspots(
  spPatterns,
  params = NULL,
  patternName = "Pattern_1",
  outlier = "positive",
  nullSamples = 1000,
  includeSelf = TRUE,
  ...
)
```

Arguments

spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
params	a named vector of the optimal sigma and threshold for a given spatial pattern. The names are should be 'sigmaOpt' and 'threshOpt'. The default value is NULL.
patternName	a character string that specifies the pattern of interest
outlier	a character string specifying whether to apply the outlier threshold to the kernel density distribution in a one-sided manner (specify 'positive' the default) or in a two sided manner (specify 'two.sided').
nullSamples	a numeric values specifying the number of spatial patterns to randomly sample for a null distribution.
includeSelf	a logic value specifying whether to consider the spatial influence the pattern has on surrounding regions only (set to FALSE), or whether to also consider the influence of the pattern itself (set to TRUE , the default).
...	Arguments passed to methods

Value

a character vector with the spatial feature name if the spatial influence exceeded the threshold for that spot/cell, and NA otherwise

See Also

Other getIntGenes: `get_interacting_genes()`, `get_pairwise_interacting_genes()`

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
                             package="SpaceMarkers",mustWork = TRUE))
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
#Obtaining CoGAPS Patterns i.e Spatial Features
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
                                     package="SpaceMarkers",mustWork = TRUE))
```

```
spFeatures <- slot(cogaps_result,"sampleFactors")
#Obtaining Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
#Match Dimensions
barcodes <- intersect(rownames(spFeatures),spCoords$barcode)
spCoords <- spCoords[barcodes,]
spFeatures <- spFeatures[barcodes,]
spPatterns <- cbind(spCoords,spFeatures[barcodes,])
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1","Pattern_5")]
data("optParams")
hotspots <- find_pattern_hotspots(
spPatterns = spPatterns,
patternName = "Pattern_1",
params = optParams[,"Pattern_1"],
outlier = "positive",nullSamples = 1000,includeSelf = TRUE)
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
```

`get_im_scores` *get_im_scores*

Description

Get the interaction scores for SpaceMarkers

Usage

```
get_im_scores(SpaceMarkers)

## S4 method for signature 'SpaceMarkersExperiment'
get_im_scores(SpaceMarkers)

## S4 method for signature 'list'
get_im_scores(SpaceMarkers)
```

Arguments

`SpaceMarkers`	A list returned by `get_pairwise_interacting_genes` (or a `SpaceMarkersExperiment` with interactions populated).

Value

A data frame with columns Gene and SpaceMarkersMetric

Examples

```
# Minimal mock of the list shape get_pairwise_interacting_genes returns:
sm_result <- list(
    list(patterns = c("A","B"),
         interacting_genes = list(data.frame(
             Gene = paste0("G", 1:5),
             SpaceMarkersMetric = sort(runif(5), decreasing = TRUE)))),
    list(patterns = c("A","C"),
         interacting_genes = list(data.frame(
             Gene = paste0("G", 1:5),
             SpaceMarkersMetric = sort(runif(5), decreasing = TRUE)))))
get_im_scores(sm_result)
```

---

get_interacting_genes *Calculate Interaction Regions and Associated Genes*

Description

This function calculates statistically significant genes using a non-parametric Kruskal-Wallis test for genes in any one region of influence and a post hoc Dunn's test is used for analysis of genes between regions.

Usage

```
get_interacting_genes(
  data,
  spPatterns,
  refPattern = "Pattern_1",
  mode = c("DE", "residual"),
  optParams = NULL,
  reconstruction = NULL,
  hotspots = NULL,
  analysis = c("enrichment", "overlap"),
  minOverlap = 50,
  ...
)
```

Arguments

data	original spatial data matrix.
spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
refPattern	a character string that specifies the pattern whose "interaction" with every other pattern we want to study. The default value is "Pattern_1".
mode	SpaceMarkers mode of operation. Possible values are "DE" (the default) or "residual".
optParams	a matrix with dimensions 2 X N, where N is the number of spatial patterns with optimal parameters. The first row contains the kernel width 'sigmaOpt' for each pattern, and the second row is the threshOpt (outlier threshold) for each pattern. Users can also input their preferred param values. The default value is NULL.

reconstruction	reconstruction of the data matrix from latent spaces. Required for "residual" mode.
hotspots	a vector that specifies the patterns to compare to the 'refPattern'. The default is NULL which indicates that all patterns would be compared to the 'refPattern'.
analysis	a character string that specifies the type of downstream analysis to be performed. Possible values are "enrichment" (default) and "overlap". In enrichment mode, all genes are returned, ranked by the SpaceMarkers metric. In overlap mode, only the genes which are significantly overexpressed in the interaction region are returned.
minOverlap	a number that specifies the minimum overlap between genes in two patterns to be considered for the statistical tests. The default is 50.
...	Arguments passed to methods

Value

a list of data frames with information about the interacting genes of the refPattern and each latent feature pattern matrix (interacting_genes object). There is also a data frame with all of the regions of influence for any two of patterns (the hotspots object).

See Also

Other getIntGenes: find_pattern_hotspots(), get_pairwise_interacting_genes()

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package="SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",h5filename = basename(counts_url))
#Obtaining CoGAPS Patterns
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
package="SpaceMarkers",mustWork = TRUE))
features <- intersect(rownames(counts_matrix),rownames(
    slot(cogaps_result,"featureLoadings")))
barcodes <- intersect(colnames(counts_matrix),rownames(
    slot(cogaps_result,"sampleFactors")))
counts_matrix <- counts_matrix[features,barcodes]
cogaps_matrix <- slot(cogaps_result,"featureLoadings")[features,]%*%
    t(slot(cogaps_result,"sampleFactors")[barcodes,])
#Obtaining Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
spCoords <- spCoords[barcodes,]
```

```
spPatterns <- cbind(spCoords,slot(cogaps_result,
"sampleFactors")[barcodes,])
data("curated_genes")
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1","Pattern_5")]
counts_matrix <- counts_matrix[curated_genes,]
cogaps_matrix <- cogaps_matrix[curated_genes, ]
data("optParams")
SpaceMarkersMode <- "DE"
ref_Pattern <- "Pattern_1"
SpaceMarkers_test <- get_interacting_genes(
    data=counts_matrix,reconstruction=NULL,
    optParams = optParams,
    spPatterns = spPatterns,
    refPattern = "Pattern_1",
    mode="DE",analysis="overlap")
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

`get_pairwise_interacting_genes` *get_pairwise_interacting_genes*

Description

Performs pairwise analysis to find genes associated with spatial interaction between pairs of spatially varying patterns.

Usage

```
get_pairwise_interacting_genes(
  data,
  spPatterns = NULL,
  mode = c("DE", "residual"),
  optParams = NULL,
  reconstruction = NULL,
  hotspots = NULL,
  minOverlap = 50,
  analysis = c("enrichment", "overlap"),
  pattern_pairs = NULL,
  ...,
  workers = NULL
)

## S4 method for signature 'SpaceMarkersExperiment'
get_pairwise_interacting_genes(
  data,
  spPatterns = NULL,
  mode = c("DE", "residual"),
  optParams = NULL,
```

```
  reconstruction = NULL,
  hotspots = NULL,
  minOverlap = 50,
  analysis = c("enrichment", "overlap"),
  pattern_pairs = NULL,
  ...,
  workers = NULL
)

## S4 method for signature 'ANY'
get_pairwise_interacting_genes(
  data,
  spPatterns,
  mode = c("DE", "residual"),
  optParams = NULL,
  reconstruction = NULL,
  hotspots = NULL,
  minOverlap = 50,
  analysis = c("enrichment", "overlap"),
  pattern_pairs = NULL,
  patternList = NULL,
  ...,
  workers = NULL
)
```

Arguments

data	original spatial data matrix.
spPatterns	A data frame that contains the spatial coordinates and metrics for spatial features (cell types/cell processes). The column names must include 'x' and 'y' as well as the spatially varying features.
mode	SpaceMarkers mode of operation. Possible values are "DE" (the default) or "residual".
optParams	a matrix with dimensions 2 X N, where N is the number of spatial patterns with optimal parameters. The first row contains the kernel width 'sigmaOpt' for each pattern, and the second row is the threshOpt (outlier threshold) for each pattern. Users can also input their preferred param values. The default value is NULL.
reconstruction	reconstruction of the data matrix from latent spaces. Required for "residual" mode.
hotspots	a vector that specifies the patterns to compare to the 'refPattern'. The default is NULL which indicates that all patterns would be compared to the 'refPattern'.
minOverlap	a number that specifies the minimum overlap between genes in two patterns to be considered for the statistical tests. The default is 50.
analysis	a character string that specifies the type of downstream analysis to be performed. Possible values are "enrichment" (default) and "overlap". In enrichment mode, all genes are returned, ranked by the SpaceMarkers metric. In overlap mode, only the genes which are significantly overexpressed in the interaction region are returned.
pattern_pairs	A matrix of pattern pairs to be analyzed. Default is
...	Arguments passed to methods

workers	(optional) Number of workers to be used for parallel processing.
patternList	(optional) Character vector of pattern column names in spPatterns to restrict the analysis to. Defaults to all pattern columns.

Details

==========

Value

a list of data frames for each pattern with 1) names of the patterns (patterns object) 2) data frame with the hotspots of influence for the two patterns (the hotspots object). 3) data frame with the genes associated with the interaction between the two patterns (interacting genes object, empty if insufficient interaction).

See Also

Other getIntGenes: find_pattern_hotspots(), get_interacting_genes()

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package="SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
sp_url <- urls[["visium_url"]][2]
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",
h5filename = basename(counts_url))
#Obtaining CoGAPS Patterns
cogaps_result <- readRDS(system.file("extdata","CoGAPS_result.rds",
package="SpaceMarkers",mustWork = TRUE))
features <- intersect(rownames(counts_matrix),rownames(
    slot(cogaps_result,"featureLoadings")))
barcodes <- intersect(colnames(counts_matrix),rownames(
    slot(cogaps_result,"sampleFactors")))
counts_matrix <- counts_matrix[features,barcodes]
cogaps_matrix <- slot(cogaps_result,"featureLoadings")[features,]%*%
    t(slot(cogaps_result,"sampleFactors")[barcodes,])
#Obtaining Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
rownames(spCoords) <- spCoords$barcode
spCoords <- spCoords[barcodes,]
spPatterns <- cbind(spCoords,
slot(cogaps_result,"sampleFactors")[barcodes,])
data("curated_genes")
spPatterns<-spPatterns[c("barcode","y","x","Pattern_1",
"Pattern_3","Pattern_5")]
```

```
counts_matrix <- counts_matrix[curated_genes,]
cogaps_matrix <- cogaps_matrix[curated_genes, ]
optParams <- matrix(c(6, 2, 6, 2, 6, 2), nrow = 2)
rownames(optParams) <- c("sigmaOpt","threshOpt")
colnames(optParams) <- c("Pattern_1","Pattern_3","Pattern_5")
SpaceMarkersMode <- "DE"
pattern_pairs <- matrix(c("Pattern_1", "Pattern_1",
          "Pattern_3", "Pattern_5"), nrow=2)
SpaceMarkers <- get_pairwise_interacting_genes(
    data=counts_matrix,reconstruction=NULL,
    optParams = optParams,
    spPatterns = spPatterns,
    mode="DE",analysis="enrichment", pattern_pairs=pattern_pairs)
#Remove present Directories if any
unlink(basename(sp_url))
unlink("spatial", recursive = TRUE)
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

`get_spatial_features` *Load spatial features*

Description

This function loads spatial features from a file containing spatial features

Usage

```
get_spatial_features(filePath, method = NULL, featureNames = ".")
```

Arguments

`filePath`	A string path to the location of the file containing the spatial features.
`method`	A string specifying the type of object to obtain spatial feature from. Default NULL, where the method is inferred based on object type. Other methods are: "CoGAPS", "Seurat", or "BayesTME".
`featureNames`	An array of strings specifying the column names corresponding to the feature names or a regex string. In the case of Seurat, all metadata columns with "_Feature" suffix are selected.

Value

a matrix of spatial features with barcodes associated with individual coordinates

Examples

```
library(SpaceMarkers)
#CoGAPS data filePath
filePath <- system.file("extdata","CoGAPS_result.rds",
package = "SpaceMarkers",mustWork = TRUE)
spFeatures <- get_spatial_features(filePath, method = "CoGAPS")
head(spFeatures)
```

`get_spatial_parameters` *Read optimal parameters for spatial kernel density from user input or .json file*

Description

This function obtains the width of a spatial kernel density (sigma) from either the user input or from a scale factors .json file. The outlier threshold around the set of spots (threshold) for each pattern is specified by the user (default is 4).

Usage

```
get_spatial_parameters(
  spatialPatterns,
  visiumDir = ".",
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  resolution = c("fullres", "lowres", "hires"),
  ...
)
```

Arguments

`spatialPatterns`	A data frame that contains the spatial coordinates for each cell type. The column names must include 'x' and 'y' as well as a set of numbered columns named 'Pattern_1.....N'.
`visiumDir`	A string path specifying the location of the 10xVisium directory
`spatialDir`	A string path specifying the location of the spatial folder containing the .json file of the spot characteristics
`pattern`	A string specifying the name of the .json file
`sigma`	A numeric value specifying the width of the kernel density estimate to be used for smoothing
`threshold`	A numeric value specifying how many standard deviations above the mean of a null distribution to use an outlier threshold for identifying 'hotspots'
`resolution`	A string specifying image resolution to be used for spot diameter. Can take values of "fullres" (default), "lowres" or "hires".
`...`	Arguments passed to methods

Value

a numeric matrix of sigmaOpts - the optimal width of the gaussian distribution, and the threshOpt - outlier threshold around the set of spots for each pattern

Examples

```
library(SpaceMarkers)
# Create test data
cells <- c()
test_num <- 500
for(i in 1:test_num){
    cells[length(cells)+1] <- paste0("cell_",i)
}
spPatterns <- data.frame(barcode = cells,
y = runif(test_num, min=0, max=test_num),
x = runif(test_num, min=0, max=test_num),
Pattern_1 = runif(test_num, min=0, max=1),
Pattern_2 = runif(test_num, min=0, max=1) )
# Call the get_spatial_parameters function with the test data
optParams <- get_spatial_parameters(spPatterns, sigma = 10)
```

---

get_spatial_params_morans_i

Calculate the optimal parameters from spatial kernel density for cell-cell interactions

---

Description

This function uses Morans.I to calculate the optimal width of the kernel density (sigmaOpt) as well as the outlier threshold around the set of spots (threshOpt) for a null distribution.

Usage

```
get_spatial_params_morans_i(spatialPatterns, ...)
```

Arguments

spatialPatterns	A data frame that contains the spatial coordinates for each cell type. The column names must include 'x' and 'y' as well as a set of numbered columns named 'Pattern_1.....N'.
...	Arguments passed to methods

Value

a numeric matrix of sigmaOpts - the optimal width of the gaussian distribution, and the threshOpt - outlier threshold around the set of spots for each pattern

Examples

```
library(SpaceMarkers)
# Create test data
cells <- c()
test_num <- 500
for(i in 1:test_num){
    cells[length(cells)+1] <- paste0("cell_",i)
}
```

```
spPatterns <- data.frame(barcode = cells,
y = runif(test_num, min=0, max=test_num),
x = runif(test_num, min=0, max=test_num),
Pattern_1 = runif(test_num, min=0, max=1),
Pattern_2 = runif(test_num, min=0, max=1) )
# Call the get_spatial_params_morans_i function with the test data
optParams <- get_spatial_params_morans_i(spPatterns)
```

`hotspots` *Access hotspot assignments on a SpaceMarkersExperiment*

Description

Read (`hotspots(x, type)`) or write (`hotspots(x, type) <- value`) the per-type hotspot data.frame stored in `metadata(x)$hotspots[[type]]`. `type` selects one of `"undirected"`, `"pattern"`, or `"influence"`.

Usage

```
hotspots(x, ...)

hotspots(x, type = "undirected") <- value

## S4 method for signature 'SpaceMarkersExperiment'
hotspots(x, type = c("undirected", "pattern", "influence"))

## S4 replacement method for signature 'SpaceMarkersExperiment'
hotspots(x, type = c("undirected", "pattern", "influence")) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`type`	Character: one of "undirected", "pattern", or "influence".
`value`	A data.frame of hotspot assignments.

Value

For the getter, a data.frame or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

A data.frame of hotspot assignments, or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
colnames(sme) <- paste0("s", seq_len(20))
hs <- data.frame(barcode = colnames(sme), x = 1:20, y = 1:20,
                 Pattern_1 = c(rep("hot", 10), rep(NA, 10)))
```

```
hotspots(sme, type = "undirected") <- hs
head(hotspots(sme, "undirected"))
```

`influence_map` *Access the per-spot influence map on a SpaceMarkersExperiment*

Description

Read (`influence_map(x)`) or write (`influence_map(x) <- value`) the per-spot influence values stored in `metadata(x)$influence`.

Usage

```
influence_map(x, ...)

influence_map(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
influence_map(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
influence_map(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A data.frame of per-spot influence values.

Value

For the getter, a data.frame of per-spot influence values or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

A data.frame of per-spot influence values, or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
colnames(sme) <- paste0("s", seq_len(20))
influence_map(sme) <- data.frame(barcode = colnames(sme),
    x = 1:20, y = 1:20, Pattern_1 = runif(20))
head(influence_map(sme))
```

`interactions` *Access pairwise interaction results on a SpaceMarkersExperiment*

Description

Read (`interactions(x, pair)`) or write (`interactions(x) <- value`) the per-pattern-pair interaction results stored in `metadata(x)$interactions`. When `pair` is supplied to the getter, only that pair's results are returned; otherwise the full named list is returned.

Usage

```
interactions(x, ...)

interactions(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
interactions(x, pair = NULL)

## S4 replacement method for signature 'SpaceMarkersExperiment'
interactions(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A named list of per-pair interaction results.
`pair`	Optional character string specifying a pattern pair name (e.g., "Pattern_1_Pattern_5"). If NULL, returns all pairs.

Value

For the getter, a named list of per-pair results (or a single pair's results when `pair` is specified). For the setter, the modified `SpaceMarkersExperiment`.

A named list of per-pair interaction results, or a single pair's results if `pair` is specified.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
interactions(sme) <- list(
    Pattern_1_Pattern_2 = list(interacting_genes = list()))
names(interactions(sme))
```

load10X *Load 10X Visium data as a SpaceMarkersExperiment*

Description

Convenience function that loads expression, coordinates, and optionally spatial features from a 10X Visium directory and assembles them into a SpaceMarkersExperiment object.

Usage

```
load10X(
  visiumDir,
  features = NULL,
  h5filename = "filtered_feature_bc_matrix.h5",
  resolution = c("fullres", "lowres", "hires"),
  version = NULL,
  ...
)
```

Arguments

visiumDir	A string path to the 10X Visium directory.
features	Optional: a file path or object for spatial features, passed to get_spatial_features.
h5filename	Name of the H5 expression file. Default "filtered_feature_bc_matrix.h5".
resolution	Resolution for coordinates. One of "fullres", "lowres", "hires".
version	Optional Spaceranger version.
...	Additional arguments passed to get_spatial_features.

Value

A SpaceMarkersExperiment object.

Examples

```
# Requires a 10x Visium directory on disk:
# sme <- load10X("path/to/Visium_outs",
#                features = "deconv_features.csv",
#                resolution = "lowres")
```

load10XCoords *Load 10x Visium Spatial Coordinates*

Description

This function loads spatial coordinates for each cell from a 10X Visium spatial folder.

Usage

```
load10XCoords(
  visiumDir,
  resolution = c("fullres", "lowres", "hires"),
  version = NULL
)
```

Arguments

visiumDir	A string path to the location of the folder containing the spatial coordinates. The folder in your visiumDir must be named 'spatial' and must contain files 'scalefactors_json.json' and 'tissue_positions_list.csv.'
resolution	A string specifying which values to look for in the .json object. Can be either fullres (default), lowres or hires.
version	A string specifying the version of the spaceranger data.

Value

a data frame of the spatial coordinates (x and y) for each spot/cell

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package = "SpaceMarkers",mustWork = TRUE))
sp_url <- urls[["visium_url"]][2]
# Spatial Coordinates
download.file(sp_url, basename(sp_url), mode = "wb")
untar(basename(sp_url))
spCoords <- load10XCoords(visiumDir = ".", version = "1.0")
unlink("spatial", recursive = TRUE)
unlink("Visium_Human_Breast_Cancer_spatial.tar.gz")
```

load10XExpr *Load 10X Visium Expression Data*

Description

This loads log-transformed 10X Visium expression data from standard 10X Visium folder.

Usage

```
load10XExpr(visiumDir = NULL, h5filename = "filtered_feature_bc_matrix.h5")
```

Arguments

visiumDir	A string path to the h5 file with expression information.
h5filename	A string of the name of the h5 file in the directory.

Value

A matrix of class dgeMatrix or Matrix that contains the expression info for each sample (cells) across multiple features (genes)

Examples

```
library(SpaceMarkers)
#Visium data links
urls <- read.csv(system.file("extdata","visium_data.txt",
package = "SpaceMarkers",mustWork = TRUE))
counts_url <- urls[["visium_url"]][1]
#Remove present Directories if any
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
download.file(counts_url,basename(counts_url), mode = "wb")
counts_matrix<-load10XExpr(visiumDir=".",h5filename = basename(counts_url))
files <- list.files(".")[grepl(basename(counts_url),list.files("."))]
unlink(files)
```

load_anndata *Load an AnnData file as a SpaceMarkersExperiment*

Description

Convenience wrapper that reads an `.h5ad` file and coerces the resulting `SingleCellExperiment` directly into a `SpaceMarkersExperiment`. If the AnnData stores spatial coordinates under `obsm["spatial"]`, they are promoted to `spatialCoords()`.

Two readers are supported:

- **anndataR** — pure-R, no Python/conda dependency, faster on first call. Recommended when available.

- **zellkonverter** — Bioconductor-standard, uses a **basilisk**-managed Python environment under the hood.

Both are `Suggests` dependencies. With `reader = "auto"` (default), anndataR is preferred when installed; otherwise zellkonverter is used. Pass `reader = "zellkonverter"` or `reader = "anndataR"` to force a specific backend.

Usage

```
load_anndata(file, reader = c("auto", "anndataR", "zellkonverter"), ...)
```

Arguments

`file`	Path to an `.h5ad` file.
`reader`	One of `"auto"`, `"anndataR"`, `"zellkonverter"`.
`...`	Additional arguments forwarded to the chosen reader's read function.

Value

A `SpaceMarkersExperiment` object.

Examples

```
# Requires anndataR or zellkonverter installed and a real .h5ad file:
# sme <- load_anndata("path/to/visium.h5ad")
```

---

lrdf *Curated Ligand-receptor interaction genes A list of vectors with genes associated with ligand-receptor interactions from CellChat database*

Description

Curated Ligand-receptor interaction genes A list of vectors with genes associated with ligand-receptor interactions from CellChat database

Format

A data frame with LR interaction genes

Value

A list of vectors of LR genes

`lr_scores` *Access ligand-receptor scores on a SpaceMarkersExperiment*

Description

Read (`lr_scores(x)`) or write (`lr_scores(x) <- value`) the matrix of ligand-receptor pair scores stored in `x@spacemarkers$results$lr_scores`.

Usage

```
lr_scores(x, ...)

lr_scores(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
lr_scores(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
lr_scores(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A matrix of ligand-receptor pair scores.

Value

For the getter, a matrix of LR pair scores or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

A matrix of ligand-receptor pair scores, or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
lr_scores(sme) <- matrix(runif(4), 2, 2,
    dimnames = list(c("LR1", "LR2"), c("P1_P2", "P2_P1")))
lr_scores(sme)
```

optParams *Optimal paramters of 5 patterns from CoGAPS.*

Description

A dataset with the optimal width of the gaussian distribution (sigmaOpt) and the outlier threshold around the set of spots (thresOpt) for each pattern obtained from CoGAPS. CoGAPS was ran on spatial transcriptomic data from a breast cancer sample.

Format

A data frame with 2 rows and 5 columns:

Pattern_1 immune cell pattern paramters

Pattern_2 Disp.1 parameters

Pattern_3 intraductal carcinoma (DCIS) parameters

Pattern_4 Disp.2 parameters

Pattern_5 invasive carcinoma lesion pattern paramters

Value

A matrix of optimal parameters for patterns identified by CoGAPS

overlap_map *Per-spot overlap classification map for a pattern pair*

Description

The spot-level counterpart to `overlap_scores`. Returns a factor with one label per spot, summarizing how each spot relates to a pattern pair given the hotspot information stored on the SME.

Usage

```
overlap_map(
  x,
  interaction_patterns,
  direction = c("forward", "reverse"),
  directed = NULL,
  interaction_label = NULL
)
```

Arguments

x	A SpaceMarkersExperiment.
interaction_patterns	Length-2 character vector of pattern names c(pat1, pat2). For directed mode these are interpreted as c(source, target) under direction = "forward".
direction	One of "forward" or "reverse" — controls which direction's classification is returned in directed mode. Ignored for undirected.
directed	Logical or NULL. When NULL (default), picks directed mode iff pattern and influence hotspots are both stored on the SME.
interaction_label	Optional override for the middle label in undirected mode (default "interacting"). Ignored in directed mode where labels are derived from interaction_patterns.

Details

Undirected (when hotspots(x, "undirected") is present and directed = FALSE or auto-detected). Three levels: <pat1>, "interacting" (override via interaction_label), <pat2>.

Directed (when hotspots(x, "pattern") AND hotspots(x, "influence") are both present). Because a directed analysis runs a separate t-test per direction (pat1 near pat2 vs pat1, and pat2 near pat1 vs pat2), the helper returns one direction at a time — the three-level factor that directly matches that direction's t-test:

- direction = "forward" (default; source = pat1, target = pat2). Levels: <pat1> (the "control" in the t-test), "<pat1> near <pat2>" (the interaction region - the "treatment"), "<pat2> influence" (context: where pat2's influence extends beyond pat1's hotspot).
- direction = "reverse" (source = pat2, target = pat1). Levels are the mirror image: <pat2>, "<pat2> near <pat1>", "<pat1> influence".

Downstream callers (including plot_spatial(..., source = "interaction")) use this helper so the plot and any user-side analysis stay consistent.

Value

A factor with one entry per spot, NA for spots outside any hotspot. Names are the spot barcodes.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
colnames(sme) <- paste0("s", seq_len(20))
hs <- data.frame(barcode = colnames(sme), x = 1:20, y = 1:20,
                 Pattern_1 = c(rep("hot", 10), rep(NA, 10)),
                 Pattern_2 = c(rep(NA, 5), rep("hot", 10), rep(NA, 5)))
hotspots(sme, type = "undirected") <- hs
table(overlap_map(sme, c("Pattern_1", "Pattern_2")))
```

overlap_scores *Access pattern overlap scores on a SpaceMarkersExperiment*

Description

Read (overlap_scores(x)) or write (overlap_scores(x) <- value) the data.frame of pattern overlap scores stored in x@spacemarkers$results$overlap_scores.

Usage

```
overlap_scores(x, ...)

overlap_scores(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
overlap_scores(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
overlap_scores(x) <- value
```

Arguments

x	A SpaceMarkersExperiment object.
...	Not used.
value	A data.frame of pattern overlap scores.

Value

For the getter, a data.frame of pattern overlap scores or NULL. For the setter, the modified SpaceMarkersExperiment.

A data.frame of pattern overlap scores, or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
overlap_scores(sme) <- data.frame(pattern1 = "Pattern_1",
    pattern2 = "Pattern_2", overlapScore = 0.5)
overlap_scores(sme)
```

params *Access analysis hyperparameters on a SpaceMarkersExperiment*

Description

Read-only accessor that returns the full list stored in `x@spacemarkers$params`, including `spatial_params`, `pattern_names`, `min_gene_expr`, `mode`, `analysis`, `min_overlap`, `directed`, `outlier`, `null_samples`, `genes`, and any other hyperparameters set during analysis.

Usage

```
params(x, ...)

## S4 method for signature 'SpaceMarkersExperiment'
params(x)
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.

Value

The full list of hyperparameters, or `NULL` if none has been set.

The full list of hyperparameters stored during analysis, including `spatial_params`, `pattern_names`, `min_gene_expr`, `mode`, `analysis`, `min_overlap`, `directed`, `outlier`, `null_samples`, `genes`, etc. Returns NULL if no parameters have been set.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
params(sme)
```

plot_cell_interaction_circos *Plot Ligand-Receptor Interactions between Cell Types*

Description

Visualizes ligand-receptor interactions as a circular plot, allowing for selection of cell types, customization of segment and link appearance, and different modes of representing interactions.

Usage

```
plot_cell_interaction_circos(
  lr_interactions_df,
  selected_cell_types = NULL,
  cell_order = NULL,
  gap_degree_after_sector = 5,
  track_height_molecules = 0.1,
  molecule_label_cex = 0.6,
  cell_label_cex = 0.9,
  link_transparency = 0.5,
  link_connection_rou = 0.7,
  link_arrowhead_type = "big.arrow",
  link_arrowhead_width = NULL,
  link_arrowhead_length = NULL,
  score_color_palette_fun = NULL,
  split_segments_for_links = TRUE,
  link_buffer_fraction = 0.1,
  scale_link_width_by_score = FALSE,
  score_transform_for_width = function(s) s,
  use_individual_molecule_colors = TRUE,
  default_ligand_color = "lightgreen",
  default_receptor_color = "lightblue",
  individual_ligand_palette_generator = NULL,
  individual_receptor_palette_generator = NULL,
  molecule_segment_border_col = NA,
  inter_molecule_segment_gap = 0.1
)
```

Arguments

`lr_interactions_df`
: A data.frame with columns: `ligand`, `receptor`, `source_cell_type`, `target_cell_type`, `score`.

`selected_cell_types`
: Optional character vector. If provided, only these cell types and interactions *between* them will be shown.

`cell_order`
: Optional character vector for specific cell type ordering. If NULL, alphabetical order is used for the (selected) cell types.

`gap_degree_after_sector`
: Numeric, gap in degrees after each sector. Default is 5.

`track_height_molecules`
: Numeric, height of the track for molecule segments. Default is 0.1.

`molecule_label_cex`
: Numeric, cex for molecule labels. Default is 0.6.

`cell_label_cex`
: Numeric, cex for cell type labels. Default is 0.9.

`link_transparency`
: Numeric, alpha for links (0 to 1). Default is 0.5.

`link_connection_rou`
: Numeric (0-1) or vector of two for `rou` in `circos.link`. Default is 0.7.

`link_arrowhead_type`
: Character. Type of arrowhead (e.g., "triangle", "big.arrow"). Default is "big.arrow".

`link_arrowhead_width`
: Numeric. Width of the arrowhead. Default uses circlize default.

`link_arrowhead_length`
: Numeric. Length of the arrowhead. Default uses circlize default.

`score_color_palette_fun`
: A function (e.g., from `circlize::colorRamp2`) to map scores to colors.

`split_segments_for_links`
: Logical. If TRUE, molecule segments are sized by interaction count/score and links connect to unique sub-segments. Default is TRUE.

`link_buffer_fraction`
: Numeric (0 to <0.5). Buffer around each link within its sub-segment. Applies if `split_segments_for_links` is TRUE. Default is 0.1.

`scale_link_width_by_score`
: Logical. If TRUE (and `split_segments_for_links` is TRUE), link/sub-segment width is proportional to score. Default is FALSE.

`score_transform_for_width`
: Function to transform scores for width scaling. Default is `function(s) s`.

`use_individual_molecule_colors`
: Logical. If TRUE, each unique ligand/receptor name gets a distinct color. Default is TRUE.

`default_ligand_color`
: Character, color for all ligand segments if `use_individual_molecule_colors` is FALSE. Default is "lightgreen".

`default_receptor_color`
: Character, color for all receptor segments if `use_individual_molecule_colors` is FALSE. Default is "lightblue".

`individual_ligand_palette_generator`
: Function (takes n, returns n colors) for unique ligands. Default generates a green palette.

`individual_receptor_palette_generator`
: Function (takes n, returns n colors) for unique receptors. Default generates a blue palette.

`molecule_segment_border_col`
: Color for the border of L/R segments. Default `NA` (no border).

`inter_molecule_segment_gap`
: Numeric, gap between L/R segments on the same track. Default is 0.1.

Value

Invisibly returns NULL. The function is called for its side effect of creating a plot.

Examples

```
if (requireNamespace("circlize", quietly = TRUE) &&
    requireNamespace("RColorBrewer", quietly = TRUE)) {
  lr_data <- data.frame(
    ligand = c("LGF1", "LGF1", "LGF2", "LGF3", "LGF4", "LGF1", "LGF5", "LGF6"),
    receptor = c("REC1", "REC2A", "REC1", "REC3B", "REC1", "REC4", "REC4", "REC2A"),
    source_cell_type = c("CellA", "CellA", "CellB", "CellC",
                         "CellA", "CellD", "CellD", "CellB"),
    target_cell_type = c("CellB", "CellC", "CellB", "CellA",
```

```
                          "CellD", "CellA", "CellD", "CellC"),
    score = c(4, 0.9, 0.7, 0.95, 0.6, 0.1, 0.88, 2.5),
    stringsAsFactors = FALSE
  )
  # Basic plot with defaults
  # plot_cell_interaction_circos(lr_data)

  # Plot with selected cell types and custom order
  # plot_cell_interaction_circos(lr_data,
  #                              selected_cell_types = c("CellA", "CellB", "CellD"),
  #                              cell_order = c("CellD", "CellA", "CellB"))
 }
```

`plot_im_scores` *plot_im_scores*

Description

Plot the top SpaceMarkers IMScores

Usage

```
plot_im_scores(
  df,
  interaction,
  cutOff = 0,
  nGenes = 20,
  geneText = 12,
  metricText = 12,
  increments = 1,
  out = NULL
)
```

Arguments

`df`	A data frame with columns Gene and SpaceMarkersMetric
`interaction`	The interaction to plot
`cutOff`	The cut off value for the plot
`nGenes`	The number of genes to plot
`geneText`	The font size for the gene text
`metricText`	The font size for the metric text
`increments`	The increments for the y-axis
`out`	The output path for the plot

Value

A `ggplot` object showing the top genes ranked by SpaceMarkersMetric for `interaction`.

Examples

```
df <- data.frame(Gene = paste0("G", 1:20),
                 Pattern_1_Pattern_2 = sort(runif(20), decreasing = TRUE))
plot_im_scores(df, interaction = "Pattern_1_Pattern_2",
               nGenes = 10)
```

`plot_overlap_scores` *plot_overlap_scores*

Description

Plot the overlap scores between patterns in hotspots

Usage

```
plot_overlap_scores(
  df,
  title = "Spatial Overlap Scores",
  out = NULL,
  fontsize = 15
)
```

Arguments

`df`	A data frame with columns pattern1, pattern2 and overlapScore
`title`	The title of the plot
`out`	The output path for the plot
`fontsize`	The font size of the plot

Value

A ggplot object

Examples

```
df <- data.frame(pattern1 = c("pattern1","pattern1","pattern2","pattern2"),
                 pattern2 = c("pattern1","pattern2","pattern1","pattern2"),
                 overlapScore = c(0.5,0.7,0.3,0.9))
plot_overlap_scores(df)
plot_overlap_scores(df, "Overlap Scores", "overlapScores.png", 15)
```

plot_source_to_target_circos *Plot Ligand-Receptor Interactions from a Single Source to Target Cell Types*

Description

Visualizes ligand-receptor interactions focusing on a single source cell type and its outgoing interactions to a specified set of target cell types. Only ligands from the source and relevant receptors on the targets are shown.

Usage

```
plot_source_to_target_circos(
  lr_interactions_df,
  source_cell_name,
  target_cell_names,
  cell_order = NULL,
  gap_degree_after_sector = 5,
  track_height_molecules = 0.1,
  molecule_label_cex = 0.6,
  cell_label_cex = 0.9,
  link_transparency = 0.5,
  link_connection_rou = 0.7,
  link_arrowhead_type = "triangle",
  link_arrowhead_width = NULL,
  link_arrowhead_length = NULL,
  score_color_palette_fun = NULL,
  split_segments_for_links = TRUE,
  link_buffer_fraction = 0.1,
  scale_link_width_by_score = FALSE,
  score_transform_for_width = function(s) s,
  use_individual_molecule_colors = TRUE,
  default_ligand_color = "lightgreen",
  default_receptor_color = "lightblue",
  individual_ligand_palette_generator = NULL,
  individual_receptor_palette_generator = NULL,
  molecule_segment_border_col = NA,
  inter_molecule_segment_gap = 0.1
)
```

Arguments

`lr_interactions_df`
: A data.frame with columns: `ligand`, `receptor`, `source_cell_type`, `target_cell_type`, `score`.

`source_cell_name`
: Character, name of the source cell type.

`target_cell_names`
: Character vector, names of target cell types to show.

`cell_order` Optional character vector for specific cell type ordering. If NULL, source cell is first, then targets alphabetically.

`gap_degree_after_sector`
Numeric, gap in degrees after each sector. Default is 5.

`track_height_molecules`
Numeric, height of the track for molecule segments. Default is 0.1.

`molecule_label_cex`
Numeric, cex for molecule labels. Default is 0.6.

`cell_label_cex` Numeric, cex for cell type labels. Default is 0.9.

`link_transparency`
Numeric, alpha for links (0 to 1). Default is 0.5.

`link_connection_rou`
Numeric (0-1) or vector of two for `rou` in `circos.link`. Default is 0.7.

`link_arrowhead_type`
Character. Type of arrowhead (e.g., "triangle", "big.arrow"). Default is "big.arrow".

`link_arrowhead_width`
Numeric. Width of the arrowhead. Default uses circlize default.

`link_arrowhead_length`
Numeric. Length of the arrowhead. Default uses circlize default.

`score_color_palette_fun`
A function (e.g., from `circlize::colorRamp2`) to map scores to colors.

`split_segments_for_links`
Logical. If TRUE, molecule segments are sized by interaction count/score and links connect to unique sub-segments. Default is TRUE.

`link_buffer_fraction`
Numeric (0 to <0.5). Buffer around each link within its sub-segment. Applies if `split_segments_for_links` is TRUE. Default is 0.1.

`scale_link_width_by_score`
Logical. If TRUE (and `split_segments_for_links` is TRUE), link/sub-segment width is proportional to score. Default is FALSE.

`score_transform_for_width`
Function to transform scores for width scaling. Default is `function(s) s`.

`use_individual_molecule_colors`
Logical. If TRUE, each unique ligand/receptor name gets a distinct color. Default is TRUE.

`default_ligand_color`
Character, color for all ligand segments if `use_individual_molecule_colors` is FALSE. Default is "lightgreen".

`default_receptor_color`
Character, color for all receptor segments if `use_individual_molecule_colors` is FALSE. Default is "lightblue".

`individual_ligand_palette_generator`
Function (takes n, returns n colors) for unique ligands. Default generates a green palette.

`individual_receptor_palette_generator`
Function (takes n, returns n colors) for unique receptors. Default generates a blue palette.

`molecule_segment_border_col`
Color for the border of L/R segments. Default `NA` (no border).

`inter_molecule_segment_gap`
Numeric, gap between L/R segments on the same track. Default is 0.1.

Value

Invisibly returns NULL. The function is called for its side effect of creating a plot.

Examples

```
if (requireNamespace("circlize", quietly = TRUE) &&
    requireNamespace("RColorBrewer", quietly = TRUE)) {
  lr_data <- data.frame(
    ligand = c("LGF1", "LGF1", "LGF2", "LGF3", "LGF4", "LGF1", "LGF5", "LGF6", "LGF7"),
   receptor = c("REC1", "REC2A", "REC1", "REC3B", "REC1", "REC4", "REC4", "REC2A", "REC5"),
    source_cell_type = c("CellA", "CellA", "CellB", "CellC",
                         "CellA", "CellD", "CellD", "CellB", "CellA"),
    target_cell_type = c("CellB", "CellC", "CellB", "CellA",
                         "CellD", "CellA", "CellD", "CellC", "CellE"),
    score = c(4,0.9,0.7,0.95,0.6,0.1,0.88,2.5,3.0),
    stringsAsFactors = FALSE
  )
  # Plot interactions from CellA to CellB and CellC
  # plot_source_to_target_circos(lr_data,
  #                           source_cell_name = "CellA",
  #                           target_cell_names = c("CellB", "CellC"))

  # Custom order and score-based link widths
  # score_pal <- circlize::colorRamp2(c(0, 4), c("white", "blue"))
  # plot_source_to_target_circos(lr_data,
  #                           source_cell_name = "CellD",
  #                          target_cell_names = c("CellA", "CellD"), # Include autocrine
  #                           cell_order = c("CellD", "CellA"),
  #                           scale_link_width_by_score = TRUE,
  #                           score_color_palette_fun = score_pal)
 }
```

`plot_spatial` *plot_spatial*

Description

SME-native spatial plot. Extracts coordinates from the `SpaceMarkersExperiment`, looks up `feature_col` in colData, rownames (gene expression), or hotspot metadata, and overlays on the tissue image if available. If no image is found, plots on a blank background.

Usage

```
plot_spatial(
  sme,
  feature_col = NULL,
  hotspot_type = c("undirected", "pattern", "influence"),
 source = c("auto", "colData", "assay", "hotspots", "influence_map", "interaction"),
  interaction_patterns = NULL,
  interaction_label = NULL,
  direction = c("forward", "reverse"),
  image = NULL,
```

```
  resolution = c("lowres", "hires", "fullres"),
  colors = NULL,
  point_size = 2.5,
  stroke = 0.05,
  alpha = 0.5,
  title = feature_col,
  bg_color = NULL,
  crop = TRUE,
  text_size = 15
)
```

Arguments

sme	A SpaceMarkersExperiment object.
feature_col	Character. Name of the feature to plot. Optional when source = "interaction" (derived from interaction_patterns). Otherwise searched according to source: colData(sme), rownames(sme) (gene expression), hotspots metadata for the specified hotspot_type, or the stored influence map.
hotspot_type	One of "undirected", "pattern", "influence". Selects which hotspot slot plot_spatial() reads when source = "hotspots" (or when source = "auto" falls through to hotspots).
source	One of "auto", "colData", "assay", "hotspots", "influence_map", or "interaction". Selects where feature_col is resolved. • "auto" (default): colData → assay → hotspots fall-through. • "colData" / "assay" / "hotspots" / "influence_map": force the lookup to a single source. • "interaction": synthesize a categorical overlay via overlap_map — feature_col is ignored; supply interaction_patterns instead.
interaction_patterns	Length-2 character vector c(pat1, pat2), required when source = "interaction". Passed to overlap_map.
interaction_label	Optional override for the middle label of the undirected interaction overlay (default "interacting"); ignored when directed mode is auto-picked.
direction	For source = "interaction" in directed mode, one of "forward" (default; source = pat1, target = pat2) or "reverse" (source = pat2, target = pat1). Ignored in undirected mode.
image	Optional: a raster or matrix image to overlay. If NULL, tries imgRaster(sme), then the stored visiumDir from load10X(), then falls back to no image.
resolution	Image resolution used when loading from visiumDir. Default "lowres".
point_size, stroke, alpha, title, bg_color, text_size, colors	ggplot aesthetics (see plot_spatial_data_over_image).
crop	Logical; crop the view to the spot bounding box. Default TRUE.

Value

A ggplot object.

Examples

```
set.seed(1)
nb <- 40
coord_mat <- matrix(runif(2 * nb), nb, 2,
    dimnames = list(paste0("s", seq_len(nb)), c("y", "x")))
sme <- SpaceMarkersExperiment(
    assays       = list(logcounts = matrix(rpois(5 * nb, 3), nrow = 5,
        dimnames = list(paste0("G", 1:5), rownames(coord_mat)))),
    colData      = data.frame(Pattern_1 = runif(nb),
                              row.names = rownames(coord_mat)),
    spatialCoords = coord_mat)
plot_spatial(sme, feature_col = "Pattern_1", source = "colData")
```

---

plot_spatial_data_over_image

plotSpatialDataOverImage

Description

This function plots spatial data over the complementary histology image of varing resolutions

Usage

```
plot_spatial_data_over_image(
  visiumDir,
  df,
  feature_col,
  barcode_col = "barcode",
  resolution = c("lowres", "hires", "fullres"),
  version = NULL,
  colors = NULL,
  point_size = 2.5,
  stroke = 0.05,
  alpha = 0.5,
  title = "Spatial Heatmap",
  bg_color = NULL,
  crop = TRUE,
  text_size = 15
)
```

Arguments

visiumDir	directory with a spatial folder containing scalefactors_json.json, images (lowres,or hires), and coordinates (tissue_positons_(list).csv or probe.csv)
df	a dataframe with the features of interest. For example, can behotspots (character), and/or influence (numeric))
feature_col	feature to plot over spots, Default: NULL
barcode_col	barcode column name to match with coordinates, Default: 'barcode'
resolution	Image resoultion to scale coordinates too, Default: c("lowres", "hires", "fullres")

version	Visium version. Automatically infers from load10XCoords if NULL,Default: NULL
colors	colors to be displayed over spots. If set to NULL, it automatically colors the spots red for character values and uses viridis for numeric values. Default: NULL
point_size	size of spots displayed on the plot, Default: 2.5
stroke	thickness of spot outline, Default: 0.05
alpha	Transparency of the spots, Default: 0.5
title	Title displayed on the plot, Default: 'Spatial Heatmap'
bg_color	background color of ggplot box, Default: NULL
crop	crop spatial plot to a zoomed in window, Default: TRUE
text_size	size of text on the plot, Default: 15

Value

a ggplot object

Examples

```
# Requires a 10x Visium directory on disk and a per-spot data.frame
# with at least `barcode` plus the column referenced by `feature_col`.
# Sketch:
# df <- data.frame(barcode = c("s1","s2"), Pattern_1 = c(0.2, 0.8))
# plot_spatial_data_over_image(visiumDir = "path/to/visium",
#                              df = df, feature_col = "Pattern_1")
```

---

plot_target_from_sources_circos

Plot Ligand-Receptor Interactions from Multiple Source to a Single Target Cell Type

---

Description

Visualizes ligand-receptor interactions focusing on a single target cell type and its incoming interactions from a specified set of source cell types. Only relevant ligands from the source cells and receptors on the target cell are shown.

Usage

```
plot_target_from_sources_circos(
  lr_interactions_df,
  source_cell_names,
  target_cell_name,
  cell_order = NULL,
  gap_degree_after_sector = 5,
  track_height_molecules = 0.1,
  molecule_label_cex = 0.6,
  cell_label_cex = 0.9,
```

```
  link_transparency = 0.5,
  link_connection_rou = 0.7,
  link_arrowhead_type = "triangle",
  link_arrowhead_width = 0.1,
  link_arrowhead_length = 0.1,
  score_color_palette_fun = NULL,
  split_segments_for_links = TRUE,
  link_buffer_fraction = 0.1,
  scale_link_width_by_score = FALSE,
  score_transform_for_width = function(s) s,
  use_individual_molecule_colors = TRUE,
  default_ligand_color = "lightgreen",
  default_receptor_color = "lightblue",
  individual_ligand_palette_generator = NULL,
  individual_receptor_palette_generator = NULL,
  molecule_segment_border_col = NA,
  inter_molecule_segment_gap = 0.1
)
```

Arguments

`lr_interactions_df`
: A data.frame with columns: `ligand`, `receptor`, `source_cell_type`, `target_cell_type`, `score`.

`source_cell_names`
: Character vector, names of source cell types to show.

`target_cell_name`
: Character, name of the single target cell type.

`cell_order`
: Optional character vector for specific cell type ordering. If NULL, target cell is first, then sources alphabetically.

`gap_degree_after_sector`
: Numeric, gap in degrees after each sector. Default is 5.

`track_height_molecules`
: Numeric, height of the track for molecule segments. Default is 0.1.

`molecule_label_cex`
: Numeric, cex for molecule labels. Default is 0.6.

`cell_label_cex`
: Numeric, cex for cell type labels. Default is 0.9.

`link_transparency`
: Numeric, alpha for links (0 to 1). Default is 0.5.

`link_connection_rou`
: Numeric (0-1) or vector of two for `rou` in `circos.link`. Default is 0.7.

`link_arrowhead_type`
: Character. Type of arrowhead (e.g., "triangle", "big.arrow"). Default is "big.arrow".

`link_arrowhead_width`
: Numeric. Width of the arrowhead. Default uses circlize default.

`link_arrowhead_length`
: Numeric. Length of the arrowhead. Default uses circlize default.

`score_color_palette_fun`
: A function (e.g., from `circlize::colorRamp2`) to map scores to colors.

split_segments_for_links
: Logical. If TRUE, molecule segments are sized by interaction count/score and links connect to unique sub-segments. Default is TRUE.

link_buffer_fraction
: Numeric (0 to <0.5). Buffer around each link within its sub-segment. Applies if split_segments_for_links is TRUE. Default is 0.1.

scale_link_width_by_score
: Logical. If TRUE (and split_segments_for_links is TRUE), link/sub-segment width is proportional to score. Default is FALSE.

score_transform_for_width
: Function to transform scores for width scaling. Default is function(s) s.

use_individual_molecule_colors
: Logical. If TRUE, each unique ligand/receptor name gets a distinct color. Default is TRUE.

default_ligand_color
: Character, color for all ligand segments if use_individual_molecule_colors is FALSE. Default is "lightgreen".

default_receptor_color
: Character, color for all receptor segments if use_individual_molecule_colors is FALSE. Default is "lightblue".

individual_ligand_palette_generator
: Function (takes n, returns n colors) for unique ligands. Default generates a green palette.

individual_receptor_palette_generator
: Function (takes n, returns n colors) for unique receptors. Default generates a blue palette.

molecule_segment_border_col
: Color for the border of L/R segments. Default NA (no border).

inter_molecule_segment_gap
: Numeric, gap between L/R segments on the same track. Default is 0.1.

Value

Invisibly returns NULL. The function is called for its side effect of creating a plot.

Examples

```
if (requireNamespace("circlize", quietly = TRUE) &&
    requireNamespace("RColorBrewer", quietly = TRUE)) {
  # Use the same test data from previous examples
  test_lr_data <- data.frame(
    ligand = c("LGF1","LGF1","LGF2","LGF3","LGF4","LGF1","LGF5","LGF6","LGF5"),
    receptor = c("REC1","REC2","REC1","REC1","REC5","REC4","REC4","REC6","REC2"),
   source_cell_type = c("CellA","CellA","CellB","CellC","CellA","CellD","CellD","CellE","CellD"),
   target_cell_type = c("CellB","CellC","CellA","CellD","CellD","CellA","CellD","CellF","CellA"),
    score = c(4.0,2.5,3.0,2.2,1.5,1.0,3.5,0.5,2.8),
    stringsAsFactors = FALSE
  )
  # Plot interactions from CellB and CellD converging on CellA
  # plot_target_from_sources_circos(test_lr_data,
  #                               source_cell_names = c("CellB", "CellD"),
  #                               target_cell_name = "CellA")
```

```
  # Another example: Interactions from CellA and CellC targeting CellD
  # plot_target_from_sources_circos(test_lr_data,
  #                               source_cell_names = c("CellA", "CellC"),
  #                               target_cell_name = "CellD",
  #                               scale_link_width_by_score = TRUE)
}
```

`reexports` *SummarizedExperiment / SpatialExperiment accessors re-exported*

Description

These generics are defined in **SummarizedExperiment** and **SpatialExperiment** and re-exported here so that `library(SpaceMarkers)` alone is enough to call them on a `SpaceMarkersExperiment` without also attaching their source packages.

Value

The values returned by the underlying generics from **SummarizedExperiment** (`assay`, `assays`, `colData`, `rowData`) and **SpatialExperiment** (`spatialCoords`, `spatialCoords<-`, `imgRaster`).

Examples

```
set.seed(1)
sme <- SpaceMarkersExperiment(
    assays        = list(logcounts = matrix(rpois(40, 2), 4, 10,
        dimnames = list(paste0("G", 1:4), paste0("s", 1:10)))),
    spatialCoords = matrix(runif(20), 10, 2,
        dimnames = list(paste0("s", 1:10), c("y", "x"))))
dim(assay(sme))
colnames(colData(sme))
head(spatialCoords(sme))
```

`save_anndata` *Save a SpaceMarkersExperiment as an AnnData (.h5ad) file*

Description

Writes a `SpaceMarkersExperiment` out to an `.h5ad` file, preserving the full SpaceMarkers analysis state (hotspots, overlap scores, interactions, influence map, kernel parameters) under `uns["spacemarkers"]`. Reloading with `load_anndata()` restores that state.

The reader selection logic is identical to `load_anndata`: **anndataR** is preferred when installed, **zellkonverter** is the fallback.

Usage

```
save_anndata(sme, file, reader = c("auto", "anndataR", "zellkonverter"), ...)
```

Arguments

sme	A SpaceMarkersExperiment object.
file	Path to the .h5ad file to write.
reader	One of "auto", "anndataR", "zellkonverter".
...	Additional arguments forwarded to the chosen reader's write function.

Value

The path file, returned invisibly.

Examples

```
# Requires anndataR or zellkonverter installed:
# sme <- SpaceMarkersExperiment(
#     assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
#         dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
#     spatialCoords = matrix(runif(40), 20, 2,
#         dimnames = list(NULL, c("y", "x"))))
# save_anndata(sme, tempfile(fileext = ".h5ad"))
```

SpaceMarkers *Main dispatcher for SpaceMarkers*

Description

Main dispatcher for SpaceMarkers. Accepts either a SpaceMarkersExperiment object or file paths to features and a Visium directory, and returns interaction scores.

Usage

```
SpaceMarkers(
  x = NULL,
  features = NULL,
  data = NULL,
  directed = FALSE,
  genes = NULL,
  min.gene.expr = 10,
  resolution = c("fullres", "lowres", "hires"),
  version = NULL,
  h5filename = "filtered_feature_bc_matrix.h5",
  spatialDir = "spatial",
  pattern = "scalefactors_json.json",
  sigma = NULL,
  threshold = 4,
  cpus = 1,
  lr_pairs = NULL,
  returnSME = TRUE,
  ...
)
```

Arguments

x	An optional SpaceMarkersExperiment object. When provided, features and data are ignored.
features	A path to a csv features file. Ignored when x is provided.
data	A path to a 10X Visium directory. Ignored when x is provided.
directed	Logical; run directed analysis (TRUE) or undirected analysis (FALSE).
genes	Optional character vector of genes to retain. If NULL, genes are filtered by min.gene.expr.
min.gene.expr	Minimum summed expression threshold for retaining genes when genes is NULL.
resolution	Resolution passed to load10XCoords(). One of "fullres", "lowres", or "hires".
version	Optional Spaceranger version passed to load10XCoords().
h5filename	Name of the 10X H5 expression file passed to load10XExpr().
spatialDir	Name of the spatial subdirectory passed to get_spatial_parameters().
pattern	Name of the JSON scale-factor file passed to get_spatial_parameters().
sigma	Optional numeric sigma passed to get_spatial_parameters().
threshold	Numeric threshold passed to get_spatial_parameters().
cpus	Number of workers used in the undirected workflow passed to get_pairwise_interacting_genes()
lr_pairs	A data.frame with ligand.symbol, receptor.symbol, and pair columns, or NULL. Used by the directed SME workflow.
returnSME	Logical; if TRUE (default), return a SpaceMarkersExperiment with results stored in the object. If FALSE, return the legacy data frame of scores.
...	Additional arguments passed only to get_pairwise_interacting_genes() or calculate_gene_scores_directed().

Value

If returnSME = TRUE, a SpaceMarkersExperiment with analysis results. If returnSME = FALSE, a matrix or data frame of interaction scores (legacy format).

Examples

```
# End-to-end on an SME (small synthetic example):
set.seed(1)
nb <- 30
bc <- paste0("s", seq_len(nb))
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(20 * nb, 3), 20, nb,
        dimnames = list(paste0("G", 1:20), bc))),
    spatialCoords = matrix(runif(2 * nb), nb, 2,
        dimnames = list(bc, c("y", "x"))))
spatial_patterns(sme) <- data.frame(Pattern_1 = runif(nb),
                                    Pattern_2 = runif(nb),
                                    row.names = bc)
spatial_params(sme) <- matrix(c(0.1, 4, 0.1, 4), nrow = 2,
    dimnames = list(c("sigmaOpt","threshOpt"), c("Pattern_1","Pattern_2")))
result <- SpaceMarkers(sme, directed = FALSE)
```

SpaceMarkersExperiment *Create a SpaceMarkersExperiment object*

Description

Constructor for the SpaceMarkersExperiment class. Accepts either a SpatialExperiment object to coerce or raw components (assays, coordinates, etc.) to build from scratch. When an SPE is provided, features can optionally be added in the same call via add_features.

Usage

```
SpaceMarkersExperiment(
  assays,
  colData = NULL,
  spatialCoords = NULL,
  rowData = NULL,
  spaceMarkers = S4Vectors::SimpleList(),
  features = NULL,
  ...
)
```

Arguments

assays	A SpatialExperiment object to coerce, or a list of assay matrices (e.g., log-counts) when building from scratch.
colData	A DataFrame of per-spot metadata. Ignored when assays is a SpatialExperiment.
spatialCoords	A numeric matrix of spatial coordinates (spots x 2). Ignored when assays is a SpatialExperiment.
rowData	Optional DataFrame of per-gene metadata. Ignored when assays is a SpatialExperiment.
spaceMarkers	A SimpleList or list for SpaceMarkers results. Defaults to an empty SimpleList.
features	Optional: a file path, matrix/data.frame, or R object (e.g., CoGAPS result) of spatial features to add via add_features.
...	Additional arguments passed to SpatialExperiment() when building from scratch, or to get_spatial_features when features is provided.

Value

A SpaceMarkersExperiment object.

Examples

```
set.seed(1)
nb <- 20
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(10 * nb, 2), 10, nb,
        dimnames = list(paste0("G", 1:10), paste0("s", seq_len(nb))))),
    spatialCoords = matrix(runif(2 * nb), nb, 2,
```

```
        dimnames = list(paste0("s", seq_len(nb)), c("y", "x"))))
sme
```

SpaceMarkersExperiment-accessors
Accessor generics for SpaceMarkersExperiment

Description

Generics for accessing SpaceMarkersExperiment results.

Read (`spatial_patterns(x)`) or write (`spatial_patterns(x) <- value`) the per-spot spatial-pattern values stored in the columns of `colData(x)` named by `x@spacemarkers$params$pattern_names`. The setter also updates `pattern_names` to the column names of `value`.

Usage

```
spatial_patterns(x, ...)

spatial_patterns(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
spatial_patterns(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
spatial_patterns(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A data.frame or DataFrame of spatial pattern values. Row names must match `colnames(x)`.

Value

For the getter, a `DataFrame` of spatial patterns or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

A `DataFrame` of spatial pattern values per spot, or NULL if pattern names have not been set.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
colnames(sme) <- paste0("s", seq_len(20))
spatial_patterns(sme) <- S4Vectors::DataFrame(
    Pattern_1 = runif(20), Pattern_2 = runif(20),
    row.names = colnames(sme))
head(spatial_patterns(sme))
```

SpaceMarkersExperiment-class

SpaceMarkersExperiment class

Description

An S4 class extending SpatialExperiment to hold SpaceMarkers analysis results alongside spatial transcriptomics data.

Usage

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

Arguments

object	A SpaceMarkersExperiment.

Slots

spacemarkers A SimpleList containing primary analysis results:

- **params** A list of all hyperparameters used in analysis, including: pattern_names (character vector), spatial_params (2 x N matrix of sigmaOpt/threshOpt per pattern), min_gene_expr (numeric), mode (character), analysis_method (character), min_overlap (numeric), directed (logical), genes (character or NULL), etc. Access via params(sme) for the full list, or spatial_params(sme) for just the kernel parameter matrix.
- **results** A list with undirected_scores (data.frame), directed_scores (data.frame), lr_scores (matrix of ligand-receptor pair scores), and overlap_scores (data.frame of pattern overlap scores).
- **analysis** A character string: "undirected", "directed", or "both".

Detailed intermediate results are stored in metadata():

- **hotspots** A list with elements undirected, pattern, and influence (each a data.frame).
- **interactions** A named list of per-pair results from get_pairwise_interacting_genes().
- **influence** A data.frame of per-spot influence values.
- **ligand_scores** Output of calculate_gene_set_score().
- **receptor_scores** Output of calculate_gene_set_specificity().

SpaceMarkersExperiment-imports

Internal imports for SpaceMarkersExperiment methods

Description

Documentation stub whose only job is to declare package imports that roxygen2 turns into importFrom entries in NAMESPACE.

Value

No return value; used only for its roxygen side effects.

spatial_params *Access or set the spatial-kernel parameters on a SpaceMarkersExperiment*

Description

Read (`spatial_params(x)`) or write (`spatial_params(x) <- value`) the 2 x N kernel-parameter matrix (rows `sigmaOpt`, `threshOpt`; one column per pattern) stored in `x@spacemarkers$params$spatial_params`.

Usage

```
spatial_params(x, ...)

spatial_params(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
spatial_params(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
spatial_params(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A matrix of spatial parameters (2 x N, rows sigmaOpt and threshOpt).

Value

For the getter, a 2 x N matrix of spatial parameters or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

The optimal kernel parameters matrix (2 x N with rows sigmaOpt, threshOpt), or NULL if not computed.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
spatial_params(sme) <- matrix(c(1, 2, 1, 2), 2, 2,
    dimnames = list(c("sigmaOpt", "threshOpt"),
                    c("Pattern_1", "Pattern_2")))
spatial_params(sme)
```

undirected_scores *Access undirected interaction scores on a SpaceMarkersExperiment*

Description

Read (`undirected_scores(x)`) or write (`undirected_scores(x) <- value`) the data.frame of undirected interaction scores stored in `x@spacemarkers$results$undirected_scores`.

Usage

```
undirected_scores(x, ...)

undirected_scores(x) <- value

## S4 method for signature 'SpaceMarkersExperiment'
undirected_scores(x)

## S4 replacement method for signature 'SpaceMarkersExperiment'
undirected_scores(x) <- value
```

Arguments

`x`	A `SpaceMarkersExperiment` object.
`...`	Not used.
`value`	A data.frame of undirected interaction scores.

Value

For the getter, a data.frame of undirected scores or `NULL`. For the setter, the modified `SpaceMarkersExperiment`.

A data.frame of undirected interaction scores (genes x pattern pairs), or NULL.

Examples

```
sme <- SpaceMarkersExperiment(
    assays = list(logcounts = matrix(rpois(200, 2), 10, 20,
        dimnames = list(paste0("G", 1:10), paste0("s", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("y", "x"))))
undirected_scores(sme) <- data.frame(
    Gene = paste0("G", 1:5), Pattern_1_Pattern_2 = runif(5))
head(undirected_scores(sme))
```

Index