Package 'scider'

May 9, 2026

Type Package

Title Spatial cell-type inter-correlation by density in R

Version 1.11.0

Author Mengbo Li, Ning Liu, Quoc Hoang Nguyen, Yunshun Chen

Maintainer Yunshun Chen <yuchen@wehi.edu.au>

Description scider is an user-friendly R package providing functions to model the global density of cells in a slide of spatial transcriptomics data. All functions in the package are built based on the SpatialExperiment object, allowing integration into various spatial transcriptomics-related packages from Bioconductor. After modelling density, the package allows for several downstream analysis, including colocalization analysis, boundary detection analysis and differential density analysis.

biocViews Spatial, Transcriptomics

License GPL-3 + file LICENSE

URL https://github.com/ChenLaboratory/scider, https://chenlaboratory.github.io/scider/

BugReports https://github.com/ChenLaboratory/scider/issues

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports SpatialExperiment, SummarizedExperiment, spatstat.geom, spatstat.explore, sf, lwgeom, SpatialPack, ggplot2, stats, pheatmap, plotly, shiny, igraph, janitor, knitr, methods, utils, isoband, S4Vectors, grDevices, dbscan, hexDensity, hexbin, uwot, SingleCellExperiment, BiocNeighbors, irlba, DropletUtils, arrow, RBioFormats, Matrix

Suggests edgeR, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.3)

VignetteBuilder knitr

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

git_branch devel

git_last_commit 7dcb837

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Contents

scider-package . . . 3
allocateCells . . . 4
cellsInRegion . . . 5
computeDensity . . . 5
computeDensityHex . . . 6
contour2sf . . . 7
coord_hash . . . 7
corDensity . . . 8
findNbrsGrid . . . 9
findNbrsSNN . . . 10
findNbrsSpatial . . . 11
findROI . . . 12
getClusters . . . 13
getContour . . . 14
getContourRegions . . . 15
getHVG . . . 16
getNiche . . . 16
globalMoran . . . 17
grid2df . . . 18
grid2sf . . . 19
gridDensity . . . 20
gridSPE . . . 21
localMoran . . . 22
mergeROI . . . 23
normalizeAssay . . . 24
plotCellCompo . . . 25
plotContour . . . 26
plotContourRegion . . . 27
plotCorHeatmap . . . 28
plotDensCor . . . 29
plotDensity . . . 30
plotDots . . . 31
plotDR . . . 32
plotGrid . . . 33
plotImage . . . 34
plotLISA . . . 35
plotLISAscatter . . . 36
plotROI . . . 37
plotSpatial . . . 38
plotViolin . . . 39
postSelRegion . . . 40
readProseg . . . 41

readVisium 41
readVisiumHD 42
readXenium 42
realignVisium 43
realignVisiumHD 43
runPCA 44
runUMAP 45
selectRegion 46
spe2PB 46
update_bound 48
xenium_bc_spe 48
[,SpatialExperiment,ANY,ANY,ANY-method 49

Index **50**

`scider-package` *scider: Spatial cell-type inter-correlation by density in R*

Description

scider is an user-friendly R package providing functions to model the global density of cells in a slide of spatial transcriptomics data. All functions in the package are built based on the SpatialExperiment object, allowing integration into various spatial transcriptomics-related packages from Bioconductor. After modelling density, the package allows for several downstream analysis, including colocalization analysis, boundary detection analysis and differential density analysis.

`scider` implements functions to analyse spatial transcriptomics data with cell type annotations by performing cell type correlation via density estimation and cell type co-localization via real number distance. Functions include density estimation, statistical modelling and visualizations.

Details

`scider` uses SpatialExperiment objects as the main infrastructure, which can easily be integrated with a wide variety of Bioconductor packages.

Author(s)

Maintainer: Yunshun Chen <yuchen@wehi.edu.au> (ORCID)

Authors:

- Mengbo Li <li.me@wehi.edu.au> (ORCID)
- Ning Liu <liu.n@wehi.edu.au> (ORCID)
- Quoc Hoang Nguyen <nguyen.q@wehi.edu.au> (ORCID)

Ning Liu <liu.n@wehi.edu.au>, Mengbo Li <li.me@wehi.edu.au>, Yunshun Chen <yuchen@wehi.edu.au>, Quoc Hoang Nguyen <nguyen.q@wehi.edu.au>

See Also

Useful links:

- https://github.com/ChenLaboratory/scider
- https://chenlaboratory.github.io/scider/
- Report bugs at https://github.com/ChenLaboratory/scider/issues

`allocateCells` *Annotate all cells with contour level of cell type-specific density.*

Description

Annotate all cells with contour level of cell type-specific density.

Usage

```
allocateCells(
  spe,
  to.roi = TRUE,
  roi = NULL,
  to.contour = TRUE,
  contour = NULL
)
```

Arguments

`spe`	A SpatialExperiment object.
`to.roi`	Logical. Whether to allocate cells to ROIs.
`roi`	Character. The name of the group or cell type on which the roi is computed. If NULL, then the cell allocation will be performed for all detected roi Default to NULL.
`to.contour`	Logical. Whether to allocate cells to contour levels.
`contour`	Character. The name of the group or cell type on which the contour level is computed. If NULL, then the cell allocation will be performed for all detected contours. Default to NULL.

Value

A SpatialExperiment object. An extra column is added to the colData.

Examples

```
data("xenium_bc_spe")
spe <- gridDensity(spe)
coi <- "Breast cancer"
spe <- findROI(spe, coi = coi)
spe <- getContour(spe, coi = coi)
spe <- allocateCells(spe, contour = coi)
```

cellsInRegion *Check which cells are in which regions*

Description

Check which cells are in which regions

Usage

```
cellsInRegion(spe, region, name_to, NA_level = "0", levels = NULL)
```

Arguments

spe	A SpatialExperiment object.
region	List or an sf object that represents a region or an ROI.
name_to	Colname in colData(spe) to store the annotation.
NA_level	Label for cells not falling in any of the regions. Default to 0.
levels	Factor levels.

Value

A SpatialExperiment object. The region information of each cell is stored in the colData.

computeDensity *Perform kernel density estimation on SpatialExperiment*

Description

Perform kernel density estimation on SpatialExperiment

Usage

```
computeDensity(
  xy,
  kernel = c("gaussian", "epanechnikov", "quartic", "disc"),
  bandwidth = NULL,
  weights = NULL,
  ngrid.x = NULL,
  xlim = NULL,
  ylim = NULL,
  diggle = FALSE,
  gridInfo = FALSE
)
```

Arguments

xy	A numeric matrix of spatial coordinates.
kernel	The smoothing kernel. Options are gaussian, epanechnikov, quartic or disc.
bandwidth	The smoothing bandwidth. By default performing automatic bandwidth selection using cross-validation using function spatstat.explore::bw.diggle.
weights	Optional weights to be attached to the points.
ngrid.x	Number of grids in the x-direction.
xlim	The range of the x-coordinates of the image.
ylim	The range of the y-coordinates of the image.
diggle	Logical. If TRUE, use the Jones-Diggle improved edge correction. See spatstat.explore::density.ppp() for details.
gridInfo	Logical. If TRUE, then the grid information is also returned.

Value

Output from spatstat.explore::density.ppp.

---

computeDensityHex *Perform kernel density estimation on SpatialExperiment*

Description

Perform kernel density estimation on SpatialExperiment

Usage

```
computeDensityHex(
  xy,
  kernel = c("gaussian"),
  bandwidth = NULL,
  weights = NULL,
  ngrid.x = NULL,
  xlim = NULL,
  ylim = NULL,
  diggle = FALSE,
  gridInfo = FALSE
)
```

Arguments

xy	A numeric matrix of spatial coordinates.
kernel	The smoothing kernel. Options are gaussian, epanechnikov, quartic or disc. ONLY GAUSSIAN IS IMPLEMENTED
bandwidth	The smoothing bandwidth. By default performing automatic bandwidth selection using cross-validation using function spatstat.explore::bw.diggle.
weights	Optional weights to be attached to the points.
ngrid.x	Number of grids in the x-direction.

xlim	The range of the x-coordinates of the image.
ylim	The range of the y-coordinates of the image.
diggle	Logical. If TRUE, use the Jones-Diggle improved edge correction. See spatstat.explore::density.ppp() for details.
gridInfo	Logical. If TRUE, then the grid information is also returned.

Value

Output from spatstat.explore::density.ppp.

---

contour2sf *Draw a contour region on some density level*

Description

Draw a contour region on some density level

Usage

```
contour2sf(spe, contour, cutoff)
```

Arguments

spe	A SpatialExperiment object.
contour	Name in metadata.
cutoff	A numeric scalar specifying the density cutoff.

Value

An sf object of the contour region of the specified level.

---

coord_hash *Hash two 15-bytes signed integers into one 32-bytes integer.*

Description

Hash two 15-bytes signed integers into one 32-bytes integer.

Usage

```
coord_hash(a, b)
```

Arguments

a, b	Integer vectors of same lengths. Can be negative.

Details

Should work for a,b in range (-2^14, 2^14-1) which is good enough for our purpose.

Integer in R is 32-bytes but reserve 1 byte for NA

corDensity *Test for density correlation between two cell types.*

Description

Test for density correlation between two cell types.

Usage

```
corDensity(spe, coi = NULL, roi = NULL, probs = 0.85, trace = FALSE)
```

Arguments

spe	A SpatialExperiment object.
coi	Character vector for cell types of interest for density correlation analysis. Default is NULL, which is to consider all cell types previously calculated in the gridDensity() step.
roi	Character. The name of the group or cell type on which the roi is computed. Default is NULL for no subsetting cell types by ROI
probs	A numeric scalar. The threshold of proportion that used to filter grids by density when ROIs have not been identified previously. Ignored if 'roi' is present in the 'metadata' component of spe. Default to 0.85.
trace	Logical. If TRUE, print the process of testing. Default to FALSE.

Value

A DataFrame containing the testing results.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts", "B cells", "T cells")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi, method = "walktrap")

result <- corDensity(spe, roi = coi)
```

findNbrsGrid *Construct a neighbour list from grid coordinates.*

Description

Construct a neighbour list from grid coordinates.

Usage

```
findNbrsGrid(
  spe,
  n = 1,
  radius = NULL,
  diagonal = FALSE,
  dist_func = c("idw", "exp", "binary", "none"),
  dist_type = c("euclidean", "manhattan"),
  standardisation = c("row", "none"),
  scale = 1,
  nbrs_name = NULL,
  cpu_threads = 6
)
```

Arguments

spe	A SpatialExperiment object.
n	Integer. Search for neighbours within (...). Either the number of neighbors or radius
radius	Numeric. Search for neighbours within the radius.
diagonal	Whether to consider diagonal connection if using square grid
dist_func	Options for distance-based weight. "idw" for inverse distance, "exp" for exponential decay, "binary" for constant weight, and "raw" for raw distance.
dist_type	Options of using euclidean or manhattan for distance calculation
standardisation	Options for weight standardisation. "none" for nothing, and "row" for dividing weights by number of neighbours.
scale	Numeric scaler for weight scaling.
nbrs_name	Name of the neighbour list to be stored. Default to be "grid".
cpu_threads	Number of cpu threads for parallel computation.

Details

If n is used, distance is scaled to unit distance

Value

A SpatialExperiment object with neighbour list stored in `spe@metadata$nbrs$grid[[nbrs_name]]`

Examples

```
data("xenium_bc_spe")
spe <- gridDensity(spe)
spe <- findNbrsGrid(spe,n=3)
```

`findNbrsSNN` *Construct a SNN neighbour list from assay.*

Description

Construct a SNN neighbour list from assay.

Usage

```
findNbrsSNN(
  spe,
  assay = NULL,
  dimred = "PCA",
  n_dimred = 10,
  k = 20,
  BNPARAM = BiocNeighbors::AnnoyParam(),
  type = c("rank", "number", "jaccard"),
  nbrs_name = NULL,
  cpu_threads = 6
)
```

Arguments

`spe`	A SpatialExperiment object.
`assay`	Name of assay for clustering. Incompatible with dimred.
`dimred`	Name of the dimensionality reduction (e.g. PCA) for clustering. Incompatible with assay
`n_dimred`	Integer scalar or vector specifying the dimensions to use if dimred is specified.
`k`	Integer scalar for number of nearest neighbors to find.
`BNPARAM`	BiocNeighborParam object specifying the nearest neighbor algorithm. Default is Annoy.
`type`	Type of weighting scheme for shared neighbors. Options are rank, number, and jaccard. type="rank" is defined in Xu and Su (2015).
`nbrs_name`	Name of the neighbour list to be stored in spe. Default to be assay/dimred + "_snn".
`cpu_threads`	Number of cpu threads for parallel computation.

Details

Construct a SNN neighbour list using either the spe's assay or reduced dimension and store it in spe@metadata$nbrs$cell

neighbour list contain

Value

A spe with the clusters stored in reducedDims.

A SpatialExperiment object

Examples

```
data("xenium_bc_spe")
spe <- runPCA(spe)
spe <- findNbrsSNN(spe,dimred="PCA")
```

`findNbrsSpatial` *Construct a distance-based neighbour list from cell coordinates.*

Description

Construct a distance-based neighbour list from cell coordinates.

Usage

```
findNbrsSpatial(
  spe,
  k = NULL,
  radius = NULL,
  dist_func = c("idw", "exp", "binary", "none"),
  standardisation = c("none", "row"),
  scale = 1,
  nbrs_name = NULL,
  cpu_threads = 6
)
```

Arguments

`spe`	A SpatialExperiment object.
`k`	Integer scalar for number of nearest neighbours to find. Can be used with radius. See details.
`radius`	Numeric for maximum distance to search for neighbours. Can be with k. See details
`dist_func`	Options for distance-based weight. "idw" for inverse distance, "exp" for exponential decay, "binary" for constant weight, and "none" for raw euclidean distance.
`standardisation`	Options for weight standardisation. "none" for nothing, and "row" for dividing weights by number of neighbours.
`scale`	Numeric scaler for weight scaling.
`nbrs_name`	Name of the neighbour list to be stored. Default to be "spatial".
`cpu_threads`	Number of cpu threads for parallel computation.

Details

if only k is provided, neighbours are found using findKNN. If only radius is provided, neighbours are found using findNeighbors. If both are provided, then knn is done first then neighbours are filtered to only those within radius.

Value

A SpatialExperiment object with neighbour list stored in spe@metadata$nbrs$cell[[nbrs_name]]

Examples

```
data("xenium_bc_spe")
spe <- findNbrsSpatial(spe,k=20,radius=100)
```

`findROI` *Find ROIs based on cell type-specific densities via graph-based method.*

Description

Find ROIs based on cell type-specific densities via graph-based method.

Usage

```
findROI(
  spe,
  coi = NULL,
  probs = 0.85,
  min.density = NULL,
  ngrid.min = 20,
  method = c("greedy", "walktrap", "connected", "hdbscan", "eigen", "dbscan"),
  diag.nodes = FALSE,
  sequential.roi.name = TRUE,
  zoom.in = FALSE,
  zoom.in.size = 500L,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
coi	A character vector of cell types of interest (COIs). Default to all cell types.
probs	A numeric scalar. The threshold of proportion that used to filter grids by density. Default to 0.85.
min.density	A numeric value. The cut-off value used to filter grids by density. Default is NULL and overwrites probs.
ngrid.min	An integer. The minimum number of grids required for defining a ROI. Default to 20.
method	The community dectection method to be used, possible options are greedy, walktrap, connected, hdbscan, eigen or dbscan. Default to greedy, can be abbreviated.

diag.nodes	Logical. Set this to TRUE to allow diagonal grid points to be adjacent nodes.
sequential.roi.name	Logical. Set this to FALSE if you want the original ROI name before filtering are retained.
zoom.in	Logical. For very large ROIs, whether to zoom in and try to get more refined ROIs.
zoom.in.size	A numeric scaler. Smallest size of an ROI to be able to zoom in. Default is 500L.
...	Other parameters that passed to walktrap.community when method = "walktrap".

Value

A SpatialExperiment object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi, method = "walktrap")
```

---

getClusters *Cluster cells in spe using graph methods.*

Description

Cluster cells in spe using graph methods.

Usage

```
getClusters(
  spe,
  nbrs_name = NULL,
  method = c("leiden", "louvain"),
  resolution = 1,
  cluster_name = "cluster",
  seed = 1,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
nbrs_name	Name of neighbour list for clustering. If NULL, will use the newest one in spe@metadata$nbrs$cell or create one if none are available.

method	Clustering methods. Options are leiden and louvain.
resolution	Higher resolution for more clusters and lower for fewer clusters. See cluster_leiden and cluster_louvain
cluster_name	Name to store the clusters in spe's colData
seed	seed for clustering
...	Other clustering arguments for cluster_leiden or cluster_louvain

Details

Cluster cells with igraph using SNN calculated by findNbrsSNN. Any neighbour list in spe@metadata$nbrs$cell can also be used

Value

A spe with the clusters stored in reducedDims.

A SpatialExperiment object

Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe,dimred="PCA")
spe <- getClusters(spe, resolution=0.5)
```

---

getContour *Get contour from density*

Description

Get contour from density

Usage

```
getContour(
  spe,
  coi = NULL,
  equal.cell = TRUE,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  id = NULL
)
```

Arguments

spe	A SpatialExperiment object.
coi	A character vector of cell types of interest (COIs). All cell types are chosen if NULL or overall.
equal.cell	Logical. Whether to use produce contour levels so that there are roughly the same number of cells of the COI at each level. Default to TRUE.
bins	An integer. Number of contour levels.
binwidth	A numeric scale of the smoothing bandwidth.
breaks	A numeric scale referring to the breaks in ggplot2:::contour_breaks.
id	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to cell_type by default or in_tissue if spe is Visium. Only needed when equal.cell = TRUE.

Value

A SpatialExperiment object. An sf object of the contour region of the specified level is stored in the metadata of the SpatialExperiment object.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

coi <- "Breast cancer"

spe <- getContour(spe, coi = coi)
```

---

getContourRegions *Calculate areas between every two density levels*

Description

Calculate areas between every two density levels

Usage

```
getContourRegions(spe, contour_name)
```

Arguments

spe	A SpatialExperiment object.
contour_name	Name of contour in spe@metadata

Value

A list of sf objects, each representing the region between two contour density levels.

getHVG *Get top highly variable genes.*

Description

Get top highly variable genes.

Usage

```
getHVG(spe, n = 1000, min.total.count = 100, min.prop = 0.01)
```

Arguments

spe	A SpatialExperiment object.
n	Integer. The number of HVGs.
min.total.count	Numeric. Genes with total counts less than min.total.count across all cells are not considered for HVGs.
min.prop	Numeric. Genes that have non-zero counts in less than the specified proportion (min.prop) of all cells are excluded from HVG selection.

Details

getHVG adopts a fast approach of NB dispersion estimation for all the genes across all cells. A lowess curve is fit to represent mean-dispersion trend. Top HVGs are selected based on the ratio of gene-wise dispersion and their trended dispersion.

Value

A SpatialExperiment object with HVG information stored in rowData(spe)$hvg as a logical vector.

Examples

```
data("xenium_bc_spe")
spe <- getHVG(spe, n=100)
```

getNiche *Build a niche assay based on the profile of neighbouring cells*

Description

Build a niche assay based on the profile of neighbouring cells

Usage

```
getNiche(
  spe,
  at = c("cell", "grid"),
  nbrs_name = NULL,
  group.by,
  use_weight = FALSE
)
```

Arguments

spe	A SpatialExperiment object
at	Option of cell or grid neighbourhood
nbrs_name	Name of the neighbour list in spe@metadata$grid[[at]]
group.by	Character vector to group neighbours cell by. Should be in either colData(spe) or spe@metadata$grid_density, depending on "at". Multiple groups can be used. See details
use_weight	Whether to scale each nbr based on its weight

Details

For numerical group, result will be sum of nbrs for each cell. For categorical group (factor/string), result will be counts of nbrs belonging in category

Value

A matrix where rows are cells/grid points and cols are groups based on group.by

Examples

```
data("xenium_bc_spe")

spe <- findNbrsSpatial(spe, k=30)
niche <- getNiche(spe, at="cell", group.by="cell_type")
```

---

globalMoran *Calculate global Moran for 1 to 2 variables.*

Description

Calculate global Moran for 1 to 2 variables.

Usage

```
globalMoran(
  spe,
  data1,
  data2 = data1[],
  at = c("grid", "cell"),
  nbrs_name = NULL,
  permutations = 999,
  seed = 123456789,
  cpu_threads = 6
)
```

Arguments

spe	A SpatialExperiment object.
data1	Numeric vector 1. Must be same length as nrow(spe@metadata$grid_density) or spatialCoords(spe), depending on 'at'.
data2	Numeric vector 2 for bivariate local Moran. Must be same length as data1.
at	Option of grid or cell for where to look for neighbour list
nbrs_name	Name of the neighbour list in `spe@metadata$grid[[at]]` for Moran's I
permutations	Number of permutations for p-value.
seed	Integer. For random permutations.
cpu_threads	(optional) The number of cpu threads used for parallel LISA computation

Value

List with global lisa, p.value, and vector of permuted lisa.

Examples

```
data("xenium_bc_spe")

## At grid.
spe <- gridDensity(spe, coi = "Breast cancer")
dat <- spe@metadata$grid_density$density_breast_cancer
spe <- findNbrsGrid(spe)
res <- globalMoran(spe,data1 = dat,at="grid")
res$lisa
## At cell.
dat <- as.numeric(spe$cell_type=="Breast cancer")
spe <- findNbrsSpatial(spe,k=10)
res <- globalMoran(spe,data1 = dat,at="cell")
res$lisa
```

---

grid2df *Convert x,y nodes to data.frame of polygons*

Description

Convert x,y nodes to data.frame of polygons

Usage

```
grid2df(
  spe,
  x = spe@metadata$grid_density$node_x,
  y = spe@metadata$grid_density$node_y,
  reverseY = FALSE,
  ...
)
```

Arguments

spe	A SpatialExperiment object with grid density calculated
x	vector of x nodes of the polygons
y	vector of y nodes of the polygons
reverseY	Reverse y coordinates. Can be numeric to specify the value to subtract y coordinates from (reverseY - y coords).
...	other elements to be stored as columns of the data.frame. Each one must be a vector same length as x.

Details

Basically grid2sf() but returns a data.frame for plotting with geom_polygon(), which allows for scale_*_transform(), unlike geom_sf().

Column names are kept similar to sf::st_coordinates

Value

data.frame with X, Y, and L2. Points with the same L2 belong to the same polygons

---

grid2sf *Convert x,y nodes to sf polygons*

Description

Convert x,y nodes to sf polygons

Usage

```
grid2sf(
  spe,
  x = spe@metadata$grid_density$node_x,
  y = spe@metadata$grid_density$node_y,
  reverseY = FALSE
)
```

Arguments

spe	A SpatialExperiment object with grid density calculated
x	vector of x nodes of the polygons
y	vector of y nodes of the polygons
reverseY	Reverse y coordinates. Can be numeric to specify the value to subtract y coordinates from (reverseY - y coords).

Details

Default is to generate sf polygons for all grid. For plotting with geom_sf, use sf::st_as_sfc(grid2sf(spe)) to convert list into Geometry Set.

Value

List of sf polygons

gridDensity *Perform kernel density estimation on SpatialExperiment for cell types of interest*

Description

Perform kernel density estimation on SpatialExperiment for cell types of interest

Usage

```
gridDensity(
  spe,
  id = if (isVisium != "none") NULL else "cell_type",
  coi = NULL,
  feature = NULL,
  assay = "counts",
  kernel = "gaussian",
  bandwidth = NULL,
  ngrid.x = NULL,
  grid.length.x = NULL,
  diggle = FALSE,
  grid.type = c("hex", "square"),
  isVisium = c("none", "visium", "visiumHD")
)
```

Arguments

spe	A SpatialExperiment object.
id	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to cell_type by default. Set to NULL for overall density.
coi	A character vector of cell types of interest (COIs). Default to all cell types.
feature	Feature(s) to calculate density with. Must be in rownames(spe).
assay	Name of assay to use for finding feature(s).
kernel	The smoothing kernel. Options are "gaussian", "epanechnikov", "quartic" or "disc". For hexagonal grid, only Gaussian is implemented
bandwidth	The smoothing bandwidth. By default performing automatic bandwidth selection using cross-validation using function spatstat.explore::bw.diggle.
ngrid.x	Number of grids in the x-direction. Ignored when 'grid.length.x' is specified. Default to NULL.
grid.length.x	Grid length in the x-direction. If both 'ngrid.x' and 'grid.length.x' are NULL, then 'grid.length.x' is set to 100 (micron) by default.
diggle	Logical. If TRUE, use the Jones-Diggle improved edge correction. See spatstat.explore::density.ppp() for details.
grid.type	Type of grid can be either hexagon or square.
isVisium	Options of 'none','visium', and 'visiumHD'. If TRUE, converts coordinates from pixel to um and fit the density grid to the same Visium spots arrangement. visium will use hexagonal while visiumHD will use rectangular grid.

Value

A SpatialExperiment object. Grid density estimates for all cell type of interest are stored in spe@metadata$grid_density. Grid information is stored in spe@metadata$grid_info

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)
```

`gridSPE` *Summarize a SpatialExperiment object at grid-level*

Description

Summarize a SpatialExperiment object at grid-level

Usage

```
gridSPE(spe, cell.count = FALSE, id = "cell_type", split.count.by = id)
```

Arguments

`spe`	A SpatialExperiment object.
`cell.count`	Logical. Whether to obtain the number of cells within each group identified by the 'id' column in colData(spe). Default to FALSE.
`id`	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to 'cell_type' by default.
`split.count.by`	A character. The name of the column of colData(spe). When it is not NULL, a grid-level count matrix is calculated for each member specified in that column of colData(spe) and stored in the assays(spe). Set to 'cell_type' by default.

Value

A SpatialExperiment object.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

spe_grid <- gridSPE(spe)
```

`localMoran` *Calculate local Moran for 1 to 2 variables.*

Description

Calculate local Moran for 1 to 2 variables.

Usage

```
localMoran(
  spe,
  data1,
  data2 = data1[],
  at = c("grid", "cell"),
  nbrs_name = NULL,
  hhonly = FALSE,
  significance_cutoff = 0.05,
  permutations = 999,
  seed = 123456789,
  cpu_threads = 6
)
```

Arguments

`spe`	A SpatialExperiment object.
`data1`	Numeric vector 1. Must be same length as number of grid points or cells, depending on 'at'.
`data2`	Numeric vector 2 for bivariate local Moran. Must be same length as data1.
`at`	Option of grid or cell for where to look for neighbour list
`nbrs_name`	Name of the neighbour list in `spe@metadata$grid[[at]]` for Moran's I. If NULL, will use the newest neighbour list.
`hhonly`	Only high-high clusters, which is more interpretable. Other clusters (e.g. "Low-Low","High-Low",...) will be assigned as undefined.
`significance_cutoff`	Cutoff for p-value to filter non-significant clusters
`permutations`	Number of permutations for p-value.
`seed`	Integer. For random permutations.
`cpu_threads`	The number of cpu threads used for parallel computation.

Value

List of lisa_value, clusters, and pseudo p-value.

Examples

```
data("xenium_bc_spe")

## At grid.
spe <- gridDensity(spe, coi = "Breast cancer")
dat <- spe@metadata$grid_density$density_breast_cancer
spe <- findNbrsGrid(spe)
res <- localMoran(spe,data1=dat, at = "grid")

## At cell.
dat <- as.numeric(spe$cell_type=="Breast cancer")
spe <- findNbrsSpatial(spe,k=20)
res <- localMoran(spe,data1=dat,at="cell")
```

---

mergeROI *Manually merge ROIs*

Description

Manually merge ROIs

Usage

```
mergeROI(
  spe,
  roi = NULL,
  merge.list = NULL,
  remove.ids = NULL,
  id = "component",
  rename = FALSE
)
```

Arguments

spe	A SpatialExperiment object.
roi	Character. The name of the group or cell type on which the roi is computed. All cell types are chosen if NULL or 'overall'.
merge.list	A (named) list of vectors of ROI ids to be merged. Each vector in the list should be of length greater than or equal to 2. If no name is specified, the merged ROI will be named by concatenating ROIs being merged.
remove.ids	Optional. A vector of ROI ids to be removed.
id	Character. The name of the column in `spe@metadata$roi` that stores the ROIs to be merged. Default is "component".
rename	Logical. If TRUE, names of merge.list are ignored. ROIs will be given a new name. For the unmerged ROIs, their new names are not necessarily the same as those before merging.

Value

A SpatialExperiment object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi, method = "walktrap")

spe <- mergeROI(spe, roi = coi, list("1-2" = 1:2))
```

`normalizeAssay` *Perform log normalization for counts*

Description

Perform log normalization for counts

Usage

```
normalizeAssay(
  spe,
  transformation = c("log"),
  scale.factor = 10000,
  assay = "counts",
  name = "logcounts"
)
```

Arguments

`spe`	A SpatialExperiment object.
`transformation`	Choice of transformation. "Log" for log1p
`scale.factor`	Factor to multiply the count of each cell by. A single value or a numeric vector of length equal to the number of cells.
`assay`	Name of assay in spe to perform the transformation on
`name`	Name of the transformed assay

Value

A SpatialExperiment object

Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
```

plotCellCompo *Plot cell type composition in each density level of cell of interest.*

Description

Plot cell type composition in each density level of cell of interest.

Usage

```
plotCellCompo(
  spe,
  contour = NULL,
  id = "cell_type",
  roi = NULL,
  self.included = TRUE
)
```

Arguments

spe	A SpatialExperiment object.
contour	A character vector of cell type(s) on which the contour density level is calculated. If NULL, it looks for 'overall_contour' in colData(spe). Default to NULL.
id	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to 'cell_type' by default.
roi	Character. The name of the group or cell type on which the roi is computed. Default is NULL for no plotting by ROI
self.included	Logical. Whether to include all the cell types in the plot. Default to TRUE. If FALSE, the cell types specified in 'contour' will not be included in the plot.

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")
coi <- "Breast cancer"
spe <- gridDensity(spe, coi = coi)
spe <- findROI(spe, coi = coi)
spe <- getContour(spe, coi = coi)
spe <- allocateCells(spe, contour = coi)
plotCellCompo(spe, contour = "Breast cancer")
plotCellCompo(spe, contour = "Breast cancer", roi = coi)
```

plotContour *Plot contour lines.*

Description

Plot contour lines.

Usage

```
plotContour(
  spe,
  coi = NULL,
  overlay = c("cell", "density", "none"),
  id = "cell_type",
  sub.level = NULL,
  line.type = 1,
  line.width = 0.5,
  line.alpha = 1,
  reverseY = NULL,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
coi	A character vector of cell types of interest (COIs). All cell types are chosen if NULL or 'overall'.
overlay	Character vector. Options are 'cell' (plot overlay on cells), 'density' (overlay on density), or 'none'. Default to 'cell'.
id	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to 'cell_type' by default.
sub.level	Character vector. Subset on specific level.
line.type	shape of contour. See 'ggplot2::geom_path()'.
line.width	size of contour.
line.alpha	alpha of contour between 0 and 1.
reverseY	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
...	Aesthetic mappings to pass to plotSpatial, plotDensity, or plotImage, depending on the overlay.

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

coi <- "Breast cancer"

spe <- getContour(spe, coi = coi)

plotContour(spe, coi = coi, line.width = 0.3, pt.alpha = 0.2)
```

---

plotContourRegion *Visualising an sf object (for internal use only at the moment)*

Description

Visualising an sf object (for internal use only at the moment)

Usage

```
plotContourRegion(
  spe,
  coi,
  id = "cell_type",
  overlay = c("density", "cell"),
  sub.level
)
```

Arguments

spe	A SpatialExperiment object.
coi	A character vector of length 1 of the cell type of interest.
id	A character. The name of the column of colData(spe) containing the cell type identifiers. Set to cell_type by default.
overlay	Character vector. Either plot overlay on density or cells.
sub.level	Numeric vector of length 1 or 2, identifies which density level to plot. When length is 1, plot the density region above this level. When length is 2, plot the density region between the two levels.

Value

A ggplot object.

plotCorHeatmap *Plot model statistics using heatmap.*

Description

Plot model statistics using heatmap.

Usage

```
plotCorHeatmap(
  model.result,
  stats = c("cor.coef", "t", "p.Pos", "p.Neg"),
  roi = "all",
  cell.type = "all",
  silent = FALSE
)
```

Arguments

model.result	A dataFrame object.
stats	Character value. Choose either coefficient or t. Coefficient by default.
roi	Character value. By default is all. The specific ROIs to be plotted.
cell.type	Character value. By default is all. The cell types to be plotted.
silent	Do not draw the plot (useful when using the gtable output).

Value

A pheatmap object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts", "B cells", "T cells")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi)

model_result <- corDensity(spe, roi = coi)

plotCorHeatmap(model_result$ROI)
```

plotDensCor *Plot density correlation between two cell types*

Description

Plot density correlation between two cell types

Usage

```
plotDensCor(
  spe,
  celltype1 = NULL,
  celltype2 = NULL,
  roi = NULL,
  probs = 0.85,
  fit = c("spline", "linear"),
  df = 3,
  pt.shape = 21,
  pt.size = 1.5,
  pt.alpha = 1,
  line.type = 1,
  line.width = 1,
  line.alpha = 1
)
```

Arguments

spe	A SpatialExperiment object.
celltype1	Cell type 1 to compare.
celltype2	Cell type 2 to compare.
roi	Character. The name of the group or cell type on which the roi is computed. Default is NULL for no facetting by ROI
probs	A numeric scalar. The threshold of proportion that used to filter grids by density when ROIs have not been identified previously. Ignored if 'roi' is present in the 'metadata' component of spe. Default to 0.85.
fit	Character. Options are "spline" and "linear".
df	Integer. Degrees of freedom of the spline fit. Default to 3 (i.e., a cubic spline fit).
pt.shape	shape of points.
pt.size	size of points.
pt.alpha	alpha of points between 0 and 1.
line.type	shape of line.
line.width	size of line.
line.alpha	alpha of line between 0 and 1.

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi, method = "walktrap")

plotDensCor(spe, celltype1 = "Breast cancer", celltype2 = "Fibroblasts", roi = coi)
```

`plotDensity` *Plot grid-based density.*

Description

Plot grid-based density.

Usage

```
plotDensity(spe, coi = NULL, probs = 0.5, reverseY = NULL, ...)
```

Arguments

`spe`	A SpatialExperiment object.
`coi`	A character vector of cell types of interest (COIs) to be plotted. Default to all cell types.
`probs`	Numeric value between 0 and 1, used for filtering uninformative grid, default is 0.5.
`reverseY`	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
`...`	Parameters pass to plotGrid

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

plotDensity(spe, coi = "Breast cancer")

plotDensity(spe, coi = "Fibroblasts")
```

plotDots *Dot plot of gene expression by groups*

Description

Visualizing average expression and percentage of cell that expression a certain gene(s). Similar to Seurat's DotPlot.

Usage

```
plotDots(
  spe,
  feature,
  assay = "counts",
  group.by = "cell_type",
  detection.limit = 0,
  expression.limit = c(-Inf, Inf),
  scale = TRUE,
  cols = NULL,
  dot.scale = 6,
  flip.axes = FALSE
)
```

Arguments

spe	A SpatialExperiment object.
feature	vector of feature names.
assay	Name of assay to use for plotting feature.
group.by	values to group plot by. Must be in colData of spe.
detection.limit	threshold for minimum expression value for percentage expression calculation (dot size)
expression.limit	Upper and lower bound for average expression. Values beyond this range are snapped to this limit. If only one value is provided, it it taken as the upper bound.
scale	Whether to scale the average expression data of each feature using scale.
cols	Custom color palette.
dot.scale	scale the radius of the plot. See scale_radius
flip.axes	Whether to flip the axes.

Examples

```
data("xenium_bc_spe")
plotDots(spe,feature = rownames(spe)[1:5])
```

plotDR *Plot reduced dimensions.*

Description

plotDR is the main function for plotting reduced dimension. Others are wrapper functions for convenience.

Usage

```
plotDR(
  spe,
  dimred = NULL,
  dims = c(1, 2),
  group.by = NULL,
  feature = NULL,
  assay = "counts",
  cols = NULL,
  pt.shape = 16,
  pt.size = 1,
  pt.alpha = 0.6,
  label = NULL,
  xlab = NULL,
  ylab = NULL,
  cols.scale = NULL
)

plotUMAP(spe, dimred = "UMAP", ...)

plotPCA(spe, dimred = "PCA", ...)
```

Arguments

spe	A SpatialExperiment object.
dimred	Name of the reduced dimension in reducedDims
dims	Numeric vector length 2 for the dimensions to be plotted. Default to first two dimensions
group.by	values to group points by. Must be in colData of spe. If NULL, will try with 'cols' if available.
feature	Feature to group points by. Must be in rownames(spe).
assay	Name of assay to use for plotting feature.
cols	Colour palette. Can be a vector of colours or a function that accepts an integer n and return n colours.
pt.shape	shape of points.
pt.size	size of points.
pt.alpha	alpha of points between 0 and 1.
label	label for the legend
xlab	label for the x-axis

ylab	label for the y-axis
cols.scale	vector of position for color if colors should not be evenly positioned. See scale_color_gradientn. Only applicable for continuous values.
...	Additional arguments pass to plotDR

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")
spe = runUMAP(spe)
plotDR(spe, group.by = "cell_type")
```

---

plotGrid *Plot grid from metadata.*

Description

Plot grid from metadata.

Usage

```
plotGrid(
  spe,
  group.by = NULL,
  feature = NULL,
  assay = "counts",
  type = c("raw", "log", "cpm", "logcpm"),
  cols = NULL,
  pol.border = FALSE,
  pol.alpha = 1,
  probs = 0,
  cutoff = NULL,
  label = NULL,
  cols.scale = NULL,
  reverseY = NULL,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
group.by	values to group polygons by. Must be in spe@metadata$grid_density, or colData(spe) if gridLevelAnalysis is TRUE. If NULL, will try with cols if available.
feature	Feature to group polygons by. Must be in rownames(spe).
assay	Name of assay to use for plotting feature.

type	Transformation to apply for the group/feature. Options are "raw" , "log", "cpm", "logcpm", or a function that accepts and returns a vector of the same length.
cols	Colour palette. Can be a vector of colours or a function that accepts an integer n and return n colours.
pol.border	Boolean. Whether to draw border for each polygon.
pol.alpha	alpha of points between 0 and 1.
probs	Numeric value between 0 and 1, used for filtering uninformative grid. Only applicable for continuous values.
cutoff	Numeric. Either a vector of length 2 for the lower & upper bounds of data to be included, or length 1 for the lower bound. Override probs if specified. Only applicable for continuous values.
label	label for the legend
cols.scale	vector of position for color if colors should not be evenly positioned. See scale_fill_gradientn. Only applicable for continuous values.
reverseY	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
...	Parameters pass to plotImage

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

plotGrid(spe, group.by = "density_overall")
```

---

plotImage *Plot background image of spe*

Description

Plot background image of spe

Usage

```
plotImage(
  spe,
  image = TRUE,
  image_id = NULL,
  sample_id = NULL,
  reverseY = NULL,
  crop = TRUE,
  image.alpha = 1
)
```

Arguments

spe	A SpatialExperiment object.
image	Logical. Whether to plot image (if present).
image_id, sample_id	sample and image identifiers for scaling factor. See scaleFactors for default behavior.
reverseY	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
crop	Whether to crop the plot to the spots.
image.alpha	alpha of points between 0 and 1.

Value

a ggplot object if there is a valid image, else NULL.

---

plotLISA *Plotting LISA (e.g. moran)*

Description

Plotting LISA (e.g. moran)

Usage

```
plotLISA(
  spe,
  lisa,
  overlay = c("grid", "point"),
  type = c("cluster", "logpvalue"),
  reverseY = NULL,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
lisa	Output from localMoran
overlay	Option of grid or point. Depend on whether localMoran is calculated at grid or point.
type	Option of cluster or logpvalue for plotting lisa's cluster or p-value, respectively.
reverseY	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
...	Parameters pass to plotGrid or plotSpatial, depending on overlay.

Value

a ggplot object

Examples

```
data("xenium_bc_spe")
spe <- gridDensity(spe, coi = "Breast cancer")
dat <- spe@metadata$grid_density$density_breast_cancer
spe <- findNbrsGrid(spe)
res <- localMoran(spe,data1=dat, at = "grid")
plotLISA(spe,lisa = res)
```

`plotLISAscatter` *Scatterplot for local moran's I*

Description

plot result obtained from localMoran()

Usage

```
plotLISAscatter(
  lisa,
  quadrant.count = TRUE,
  text.size = 11/.pt,
  xlab = "Data",
  ylab = "Spatial Lag"
)
```

Arguments

`lisa`	A list obtained from localMoran
`quadrant.count`	Whether to count values at each quadrant (Low-Low, Low-High, High-High, High-Low)
`text.size`	Numeric for text size of quadrant.count
`xlab`	label for the x-axis
`ylab`	label for the y-axis

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")
dat <- spe$total_counts
spe <- findNbrsSpatial(spe,k=50)
res <- localMoran(spe,data1=dat,at="cell")
plotLISAscatter(res)
```

plotROI *Plot ROIs on spatial.*

Description

Plot ROIs on spatial.

Usage

```
plotROI(
  spe,
  roi = NULL,
  id = "cell_type",
  label = TRUE,
  show.legend = FALSE,
  reverseY = NULL,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
roi	Character. The name of the group or cell type on which the roi is computed. All cell types are chosen if NULL or 'overall'.
id	Character. The name of the column of colData(spe) containing the cell type identifiers. Set to cell_type by default.
label	Logical. Show ROI label or not.
show.legend	Logical. Show legend or not.
reverseY	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
...	Parameters pass to plotSpatial

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts")

spe <- gridDensity(spe, coi = coi)

spe <- findROI(spe, coi = coi, method = "walktrap", steps = 5)

plotROI(spe, roi = coi, pt.size = 0.3, pt.alpha = 0.2)
```

`plotSpatial` *Plot cells based on spatial coordinates.*

Description

Plot cells based on spatial coordinates.

Usage

```
plotSpatial(
  spe,
  group.by = NULL,
  feature = NULL,
  assay = "counts",
  type = c("raw", "log", "cpm", "logcpm"),
  cols = NULL,
  pt.shape = 16,
  pt.size = 0.3,
  pt.alpha = 0.5,
  label = NULL,
  cols.scale = NULL,
  reverseY = NULL,
  ...
)
```

Arguments

`spe`	A SpatialExperiment object.
`group.by`	values to group points by. Must be in colData of spe. If NULL, will try with 'cols' if available.
`feature`	Feature to group polygons by. Must be in rownames(spe).
`assay`	Name of assay to use for plotting feature.
`type`	Transformation to apply for the group/feature. Options are "raw" , "log", "cpm", "logcpm", or a function that accepts and returns a vector of the same length.
`cols`	Colour palette. Can be a vector of colours or a function that accepts an integer n and return n colours.
`pt.shape`	shape of points.
`pt.size`	size of points.
`pt.alpha`	alpha of points between 0 and 1.
`label`	label for the legend
`cols.scale`	vector of position for color if colors should not be evenly positioned. See scale_color_gradientn. Only applicable for continuous values.
`reverseY`	Logical. Whether to reverse Y coordinates. Default is TRUE if the spe contains an image (even if not plotted) and FALSE if otherwise.
`...`	Parameters pass to plotImage

Value

A ggplot object.

Examples

```
data("xenium_bc_spe")

plotSpatial(spe, group.by = "cell_type", pt.size = 0.5, pt.alpha = 0.6)
```

`plotViolin` *Violin plot using genes or cell data*

Description

Violin plot using genes or cell data

Usage

```
plotViolin(
  spe,
  feature,
  assay = "counts",
  group.by = NULL,
  type = c("raw", "log", "cpm", "logcpm"),
  point = FALSE,
  cols = NULL,
  ncol = NULL,
  pt.size = 0.3,
  pt.alpha = 0.3,
  pt.shape = ".",
  ylab = "Expression",
  xlab = NULL
)
```

Arguments

`spe`	A SpatialExperiment object.
`feature`	can be a vector of gene names in rownames(spe), or column names in colData(spe) if those columns are numeric.
`assay`	Name of assay to use for plotting feature.
`group.by`	values to group plot by. Must be in colData of spe and must be either factor or character.
`type`	Transformation to apply for the group/feature. Options are "raw" , "log", "cpm", "logcpm", or a function that accepts and returns a vector of the same length.
`point`	Whether to plot points.
`cols`	Colour palette for violins. Can be a vector of colours or a function that accepts an integer n and return n colours.
`ncol`	Number of column if group.by is used.

pt.size	Size of points.
pt.alpha	Alpha of points between 0 and 1.
pt.shape	Shape of points.
ylab	Label for the y-axis.
xlab	Label for the x-axis

Examples

```
data("xenium_bc_spe")
plotViolin(spe,c("cell_area","nucleus_area"),group.by="cell_type",ylab="Area")
```

---

postSelRegion *Merge sel_region from the selectRegion function to SpatialExperiment.*

Description

Merge sel_region from the selectRegion function to SpatialExperiment.

Usage

```
postSelRegion(spe, sel_region)
```

Arguments

spe	A SpatialExperiment object.
sel_region	A dataframe object. Can be generated from function selectRegion.

Value

A SpatialExperiment object.

Examples

```
data("xenium_bc_spe")

coi <- c("Breast cancer", "Fibroblasts", "B cells", "T cells")

spe <- gridDensity(spe, coi = coi)

sel_region <- data.frame(
    "node" = seq(10),
    "node_x" = seq(10),
    "node_y" = seq(10)
)

spe1 <- postSelRegion(spe, sel_region)
```

readProseg *Read Proseg V2 output into spe*

Description

Read Proseg V2 output into spe

Usage

```
readProseg(
  dir,
  sample_id = "sample01",
  count = "expected-counts\\.(csv|mtx)\\.gz",
  coord = "cell-metadata.csv.gz",
  gene = "gene-metadata.csv.gz",
  coordNames = c("centroid_x", "centroid_y", "centroid_z")
)
```

Arguments

dir	directory containing the Proseg files
sample_id	Name of the sample.
count	Name of the file with the count assay.
coord	Name of the file with the tissue coordinates
gene	Name of the file with the gene metadata. This
coordNames	Name of the coordinates for the spe

Details

This does not work on zarr file output of proseg V3

readVisium *Read Visium output into spe*

Description

Read Visium output into spe

Usage

```
readVisium(
  dir,
  sample_id = "sample01",
  count = NULL,
  coord = NULL,
  image = NULL,
  scale_factors = NULL
)
```

Arguments

dir	directory containing the Visium files
sample_id	Name of the sample.
count	Name of the h5 file with the count assay.
coord	Name of the csv file with the tissue coordinates
image	Names of the image files.
scale_factors	Names of the scale factors file

---

readVisiumHD *Read VisiumHD output into spe*

Description

Read VisiumHD output into spe

Usage

```
readVisiumHD(dir, bin = c("016um", "008um", "002um"), ...)
```

Arguments

dir	directory containing the VisiumHD files
bin	Which bin size to use. Options of "016um","008um", and "002um"
...	Parameters for readVisium

---

readXenium *Read Xenium output into spe*

Description

Read Xenium output into spe

Usage

```
readXenium(
  dir,
  sample_id = "sample01",
  count = file.path(dir, "cell_feature_matrix.h5"),
  coord = file.path(dir, "cells.parquet"),
  image = file.path(dir, "morphology.ome.tif"),
  image_reso = 6,
  image_layer = NULL
)
```

Arguments

dir	directory containing the Xenium files
sample_id	Name of the sample.
count	Name of the h5 file with the count assay.
coord	Name of the parquet file with the tissue coordinates
image	Names of the ome.tif image files.
image_reso	resolution of the image to use. From 1-8 (lower = better resolution). See https://kb.10xgenomics.com/hc/articles/11636252598925. Default to 6
image_layer	Which layer of the tiff image to use. Default is the middle-most layer

---

realignVisium *Scale and straighten out Visium coordinates*

Description

Scale and straighten out Visium coordinates

Usage

```
realignVisium(spe, distPoint = 100)
```

Arguments

spe	A SpatialExperiment object.
distPoint	Numeric. Desired point to point distance.

Details

This function rescale the distance between points to 100um (or other value) to matchthe real distance. In addition, Visium spots have a slight tilt to them which this function will also fix

---

realignVisiumHD *Scale and straighten out VisiumHD coordinates*

Description

Scale and straighten out VisiumHD coordinates

Usage

```
realignVisiumHD(spe, distPoint = NULL)
```

Arguments

spe	A SpatialExperiment object.
distPoint	Numeric. Desired point to point distance. If NULL, will try to determine the bin level of spe and use that.

Details

This function rescale the distance between points to 8um (or other value) to matchthe real distance. In addition, Visium spots have a slight tilt to them which this function will also fix

---

`runPCA` *Fast PCA using irlba.*

Description

Fast PCA using irlba.

Usage

```
runPCA(
  spe,
  n_pcs = 50,
  assay = "logcounts",
  centre = TRUE,
  scale = TRUE,
  name = "PCA",
  genes = "hvg",
  ...
)
```

Arguments

`spe`	A SpatialExperiment object.
`n_pcs`	Number of principal components to calculate
`assay`	Name of assay used for PCA. See details for defaults.
`centre`	Logical. Whether to centre the assay before PCA.
`scale`	Logical. Whether to scale the variance to 1 before PCA.
`name`	Name to store the PCA in the spe's reducedDims
`genes`	Subset of features for PCA. Can be a column in rowData or a vector of gene names, indices, or booleans. Default to hvg if getHVG was run.
`...`	Other parameters to be passed to irlba.

Details

By default, runPCA uses logcounts assay (from normalizeAssay). If that's unavailable, it falls back to counts assay

Value

A SpatialExperiment with the PCA stored in reducedDims.

Examples

```
data("xenium_bc_spe")

spe <- runPCA(spe)
```

runUMAP *UMAP using uwot. Parameters are set to be similar to Seurat's*

Description

UMAP using uwot. Parameters are set to be similar to Seurat's

Usage

```
runUMAP(
  spe,
  n_neighbors = 30,
  n_components = 2,
  metric = "cosine",
  min_dist = 0.3,
  assay = NULL,
  dimred = "PCA",
  n_dimred = NULL,
  name = "UMAP",
  ...
)
```

Arguments

spe	A SpatialExperiment object.
n_neighbors, n_components, metric, min_dist	See umap
assay	Name of assay for UMAP. Incompatible with dimred.
dimred	Name of the dimensionality reduction (e.g. PCA) for UMAP. Incompatible with assay
n_dimred	Integer scalar or vector specifying the dimensions to use if dimred is specified.
name	Name to store the UMAP in the spe's reducedDims.
...	Other parameters to be passed to umap.

Details

By default, runUMAP uses PCA (from runPCA). If that's unavailable, it falls back to logcounts, then counts assay.

Value

A SpatialExperiment with the UMAP stored in reducedDims.

Examples

```
data("xenium_bc_spe")
spe <- runPCA(spe)
spe <- runUMAP(spe,dimred="PCA",n_dimred=10)
```

selectRegion *Select region of interest from plot*

Description

Select region of interest from plot

Usage

```
selectRegion(data, x.col = "x", y.col = "y", save.region = FALSE)
```

Arguments

data	A data.frame object.
x.col	Column name of the x coordinates.
y.col	Column name of the y coordinates.
save.region	Whether to also export the region defined by box/lasso select as an sf polygon.

Value

A data.frame object in the global environment. If save.region is TRUE, output is a list with a data.frame of selected points and a sf polygon instead.

Examples

```
data("xenium_bc_spe")

spe_b <- spe[, SummarizedExperiment::colData(spe)$cell_type == "B cells"]

dat <- as.data.frame(SpatialExperiment::spatialCoords(spe_b))

# selectRegion(dat, x.col = "x_centroid", y.col = "y_centroid")
```

spe2PB *Given a 'SpatialExperiment' data object, create pseudo-bulk samples using the colData information and return a DGEList object*

Description

Given a 'SpatialExperiment' data object, create pseudo-bulk samples using the colData information and return a DGEList object

Usage

```
spe2PB(
  spe,
  by.group = TRUE,
  group.id = "cell_type",
  keep.groups = NULL,
  roi = NULL,
  roi.only = TRUE,
  contour = NULL
)
```

Arguments

spe	A SpatialExperiment object.
by.group	Logical. Whether to perform pseudo-bulking by group. TRUE by default.
group.id	Character. The column name of the colData(spe) that contains the group information. Default to 'cell_type'.
keep.groups	Vector. Values from group.id to include in pseudo- bulking. Default is NULL, where all cells are included in pseudo-bulking.
roi	Character. The name of the group or cell type on which the roi is computed. If NULL, then no pseudo-bulking will be performed based on roi. Default to NULL.
roi.only	Logical. Whether to filter out pseudo-bulk samples formed by cells not in any ROIs. TRUE by default.
contour	Character. The name of the group or cell type on which the contour level is computed. If NULL, then no pseudo-bulking will be performed based on contour level. Default to NULL.

Value

An edgeR::DGEList object where each library (column) is a pseudo-bulk sample.

Examples

```
data("xenium_bc_spe")

spe <- gridDensity(spe)

coi <- "Breast cancer"

spe <- findROI(spe, coi = coi)

spe <- allocateCells(spe, to.contour=FALSE)

y <- spe2PB(spe, roi = coi)
```

update_bound *Update the x,y limits of a plot*

Description

Update the x,y limits of a plot

Usage

```
update_bound(p, x = NULL, y = NULL)
```

Arguments

p	A ggplot() object
x, y	Vectors of new x and y limits.

Value

a ggplot object

xenium_bc_spe *Description of the scider example datasets*

Description

scider-package has 1 datasets:

- xenium_bc_spe Example test spatial transcriptomics data in SpatialExperiment format. This test data is randomly subsetting from a publicly available 10X Xenium breast cancer data. Source data: https://www.10xgenomics.com/resources/datasets/xenium-ffpe-human-breast-with-custom-add-on-panel-1-standard

Usage

```
data("xenium_bc_spe")
```

Format

A SpatialExperiment object

Value

A SpatialExperiment object

Examples

```
data(xenium_bc_spe)
```

`[,SpatialExperiment,ANY,ANY,ANY-method`
Subset for grid level analysis

Description

Overwrite the default SpatialExperiment subsetting method to ensure 'grid_density' is also subsetted if 'gridLevelAnalysis' is TRUE (1 polygon 1 spot)

Usage

```
## S4 method for signature 'SpatialExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A SpatialExperiment object.
`i`	row indices for subsetting.
`j`	col indices for subsetting.
`...`	further arguments to be passed to or from other methods.
`drop`	passed on to [indexing operator.

Value

A SpatialExperiment object.

Index

∗ **internal**
    scider-package, 3
    xenium_bc_spe, 48
[,SpatialExperiment,ANY,ANY,ANY-method, 49

allocateCells, 4

BiocNeighborParam, *10*

cellsInRegion, 5
cluster_leiden, *14*
cluster_louvain, *14*
colData, *14*
computeDensity, 5
computeDensityHex, 6
contour2sf, 7
coord_hash, 7
corDensity, 8

findKNN, *12*
findNbrsGrid, 9
findNbrsSNN, 10, *14*
findNbrsSpatial, 11
findNeighbors, *12*
findROI, 12

getClusters, 13
getContour, 14
getContourRegions, 15
getHVG, 16, *44*
getNiche, 16
globalMoran, 17
grid2df, 18
grid2sf, 19
gridDensity, 20
gridSPE, 21

irlba, *44*

localMoran, 22, *35, 36*

mergeROI, 23

normalizeAssay, 24, *44*

plotCellCompo, 25
plotContour, 26
plotContourRegion, 27
plotCorHeatmap, 28
plotDensCor, 29
plotDensity, *26*, 30
plotDots, 31
plotDR, 32
plotGrid, *30*, 33, *35*
plotImage, *26*, *34*, 34
plotLISA, 35
plotLISAscatter, 36
plotPCA *(*plotDR*)*, 32
plotROI, 37
plotSpatial, *26*, *35*, *37*, 38
plotUMAP *(*plotDR*)*, 32
plotViolin, 39
postSelRegion, 40

readProseg, 41
readVisium, 41
readVisiumHD, 42
readXenium, 42
realignVisium, 43
realignVisiumHD, 43
reducedDims, *11*, *14*, *32*, *44, 45*
runPCA, 44, *45*
runUMAP, 45

scale, *31*
scale_color_gradientn, *33*, *38*
scale_fill_gradientn, *34*
scale_radius, *31*
scaleFactors, *35*
scider *(*scider-package*)*, 3
scider-package, 3
selectRegion, 46
spe *(*xenium_bc_spe*)*, 48
spe2PB, 46

umap, *45*
update_bound, 48

xenium_bc_spe, 48