# Package: scider (via r-universe)

November 25, 2026

**Type** Package

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

**Version** 1.12.0

**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

**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, Matrix, patchwork, withr

**Suggests** edgeR, RBioFormats, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 4.4.0)

**VignetteBuilder** knitr

**Config/pak/sysreqs** libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libmagick++-dev gsfonts libicu-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

**Repository** Bioconductor 3.24

**Date/Publication** 2026-11-23 22:51:32 UTC

**RemoteUrl** https://github.com/bioc/scider

**RemoteRef** RELEASE_3_24

**RemoteSha** 5804cf3daf226c3edc601400d5dc692260e7867e

# Contents

[,SpatialExperiment,ANY,ANY,ANY-method . . . 3
allocateCells . . . 4
cellsInRegion . . . 5
computeDensity . . . 5
computeDensityHex . . . 6
contour2sf . . . 7
coord_hash . . . 8
corDensity . . . 9
findNbrsGrid . . . 11
findNbrsSNN . . . 12
findNbrsSpatial . . . 14
findROI . . . 15
getClusters . . . 16
getContour . . . 18
getContourRegions . . . 19
getDE . . . 19
getHVG . . . 21
getMarkers . . . 22
getNiche . . . 23
getSubClusters . . . 24
globalMoran . . . 26
grid2df . . . 27
grid2sf . . . 28
gridDensity . . . 29
gridSPE . . . 30
localMoran . . . 31
mergeClusters . . . 32
mergeROI . . . 33
normalizeAssay . . . 34
plotCellCompo . . . 35
plotContour . . . 36
plotContourRegion . . . 37
plotCorHeatmap . . . 38
plotDensCor . . . 39

plotDensity . . . 41
plotDots . . . 42
plotDR . . . 43
plotGrid . . . 45
plotHVG . . . 47
plotImage . . . 48
plotLISA . . . 48
plotLISAscatter . . . 49
plotROI . . . 50
plotSpatial . . . 52
plotTopMarkers . . . 54
plotViolin . . . 56
postSelRegion . . . 57
readProseg . . . 58
readVisium . . . 59
readVisiumHD . . . 60
readXenium . . . 60
realignVisium . . . 61
realignVisiumHD . . . 62
relabelClusters . . . 63
runPCA . . . 64
runUMAP . . . 65
selectRegion . . . 66
spe2PB . . . 67
topMarkers . . . 68
trimEdge . . . 69
update_bound . . . 70

**Index** **71**

---

`[,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.

---

## 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

### Value

An integer vector with one hash per (a, b) pair.

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,
  min.density = NULL,
  trace = FALSE,
  method = c("permutation", "modified.ttest"),
  id = "cell_type",
  nperm = 999,
  permute.within = NULL,
  cpu_threads = 1
)
```

## 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. |
| min.density | A numeric value. The cut-off on the same density score as 'probs': grids with a score of at least 'min.density' are kept. Ignored if 'roi' is given. Default is NULL and overwrites probs. |
| trace | Logical. If TRUE, print the process of testing. Default to FALSE. |
| method | Character. The testing method, either "permutation" (default) or "modified.ttest". See Details. |
| id | Character. The column of colData(spe) holding the cell type labels that were used in gridDensity(). Only used by method "permutation". Default to "cell_type". |
| nperm | Integer. The number of random relabellings used by method "permutation". Default to 999. |

`permute.within` Character or NULL. The column of colData(spe) that splits cells into groups (e.g. tissue compartments) for method "permutation". Cell type labels are then shuffled only among cells of the same group. Default to NULL, which shuffles labels across all cells. See Details.

`cpu_threads` Integer. The number of threads used by method "permutation". The results do not depend on it. Default to 1.

## Details

Method "permutation" tests each pair of cell types against random labelling. Every cell keeps its location and the cell type labels are shuffled across all cells of the sample, after which the cell type densities are recomputed with the same kernel and bandwidth as in gridDensity(). The statistic is the Spearman correlation between the two densities across grids: within each ROI for the ROI level results, and pooled over all ROIs after centering the ranks at the ROI level for the overall results. The column 'cor.expected' gives the mean of the statistic under random labelling, which is the correlation expected from the shared dependence of both densities on local cellularity alone. 'p.Pos' and 'p.Neg' are one sided Monte Carlo p-values, (1 + b) / (1 + nperm), for the observed correlation being larger or smaller than expected under random labelling. The 'shift' (cor.coef - cor.expected) measures the strength of co-localization (positive) or segregation (negative) beyond random labelling. The overall results also give a per pair summary of the ROI level results: 'n.roi' ROIs with a defined correlation, 'n.above' and 'n.below' ROIs with the observed correlation above or below its expected value, and 'n.sig.Pos' and 'n.sig.Neg' ROIs with p.Pos or p.Neg below 0.05. With many grids, small shifts reach significance, so rank pairs by the shift and check the ROI summary for consistency. Use set.seed() for reproducible p-values; the results do not depend on 'cpu_threads'.

With 'permute.within', the null hypothesis is random labelling within groups: each group keeps its cell locations and cell type counts, and only the labels inside a group are shuffled. Structure already encoded by the groups, such as the split between tumour and stroma compartments, then becomes part of the expected value, and the test asks whether cell types are organized beyond it. For two cell types in the same group a positive shift means they share neighbourhoods more than two random cells of that group would; for cell types in different groups it means they sit closer to each other than random cells of their groups would. Groups should be coarse: a cell type that is the only type in every group it occupies is never moved, so pairs involving it are only tested through the other cell type.

Method "modified.ttest" is the original approach: a Pearson correlation within each ROI tested by the modified t-test of Dutilleul (1993) from SpatialPack, with the ROI level p-values combined by Fisher's method. Its effective degrees of freedom are estimated from the spatial correlogram within each ROI.

## Value

A list of two DataFrames, 'ROI' (results per ROI) and 'overall' (results combined across ROIs). When 'roi' is NULL the selected grids form a single ROI, labelled "1", and the overall results equal the ROI level results. For method "permutation" the columns are celltype1, celltype2, ROI, ngrid, cor.coef (observed Spearman correlation), cor.expected (mean under random labelling), shift, p.Pos and p.Neg; the overall DataFrame has n.roi, n.above, n.below, n.sig.Pos and n.sig.Neg (see Details) in place of ROI and ngrid. For method "modified.ttest" the columns are celltype1, celltype2, ROI, ngrid, cor.coef, t, df, p.Pos and p.Neg.

## 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")

set.seed(1)
result <- corDensity(spe, roi = coi, nperm = 199)
```

---

`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 = NULL,
  k = 20,
  BNPARAM = BiocNeighbors::AnnoyParam(),
  type = c("jaccard", "rank", "number"),
  prune = NULL,
  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. Defaults to NULL, which uses all available dimensions of the reduced dim (e.g. all PCs produced by runPCA). |
| k | Integer scalar for number of nearest neighbors to find. Default to 20. |
| BNPARAM | BiocNeighborParam object specifying the nearest neighbor algorithm. Default is Annoy. |
| type | Type of weighting scheme for shared neighbors. Options are jaccard (default), rank, and number. type="rank" is defined in Xu and Su (2015). |
| prune | Edges with SNN weight at or below this value are removed before clustering. Defaults to NULL, which applies a jaccard cutoff of 0.1 for "jaccard", converted to the equivalent shared-neighbour count (2(k+1)*0.1/1.1) for "number". type="rank" is not pruned by default. Set to 0 to disable. |
| 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",
  unassigned = c("merge", "label", "discard"),
  min_size = NULL,
  start_from = 0,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| spe | A SpatialExperiment object. |
| nbrs_name | Name of neighbour list for clustering. If NULL, uses the most recent SNN list from findNbrsSNN, which must have been run first; otherwise getClusters errors. |
| 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 |
| unassigned | How to handle cells in clusters with at most min_size cells (singletons and tiny fragments, often left isolated by SNN pruning). One of: "merge" (default) graph-merges each cell into its most-connected retained cluster; "label" leaves all such cells "unassigned"; "discard" removes them from the spe (changing ncol(spe)). |
| min_size | Maximum size for a cluster to be treated as too small (see unassigned). Defaults to NULL, which uses either 5 or 0.01\ whichever is smaller. |
| start_from | Integer at which cluster numbering starts. 0 (default) numbers clusters 0..K-1; set to 1 for 1..K. |
| seed | Integer or NULL. If NULL (default), the clustering uses the current random number stream, so call set.seed() beforehand for reproducible clusters. Otherwise the clustering uses this seed and the caller's random number stream is left unchanged. |
| verbose | Logical. Whether to report how cells in tiny clusters were reassigned/labelled/discarded (see unassigned). Defaults to FALSE. |
| ... | Other clustering arguments for cluster_leiden or cluster_louvain |

## Details

Cluster cells with igraph using an SNN neighbour list built by findNbrsSNN, which must be run before getClusters. Any neighbour list in spe@metadata$nbrs$cell can be selected via nbrs_name.

## 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")
set.seed(1)
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.

---

`getDE` *Differential expression between two clusters or two groups of clusters.*

---

## Description

Pseudo-bulks cells by cluster via spe2PB and performs a **two-sided** quasi-likelihood test comparing the average of one set of clusters (`cluster`) against another (`cluster_2`), analogous to glmQLFTest. The contrast is the difference of the two group means, tested by fitting the null design with glmQLFit (see getMarkers for why the null design is used). The result table is post-processed like topTags.

## Usage

```
getDE(
  spe,
  cluster,
  cluster_2,
  cluster_name = "cluster",
  dispersion = 0.05,
  method = c("quasi", "lrt"),
  n = Inf,
  adjust.method = "BH",
  sort.by = "PValue",
  p.value = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| spe | A SpatialExperiment object. |
| cluster | Character vector. One or more cluster labels forming the first group. Positive logFC means up-regulated in this group. |
| cluster_2 | Character vector. One or more cluster labels forming the second (reference) group. Must not overlap cluster. |
| cluster_name | Character. Column name in colData(spe) containing cluster labels. Default "cluster". |
| dispersion | Numeric. Fixed NB dispersion (BCV^2) used for the GLM fits. Default 0.05 (BCV $\approx$ 0.224). |
| method | Character. Testing pipeline: "quasi" (default, glmQLFit quasi-deviance test) or "lrt" (glmFit + glmLRT). Both use the preset dispersion. |
| n | Integer. Maximum number of genes to return. Default Inf (all genes). |
| adjust.method | Character. Multiple-testing adjustment method passed to p.adjust. Default "BH". |
| sort.by | Character. How to order the table: "PValue" (default), "logFC", "stat", or "none". |
| p.value | Numeric. Cutoff on the adjusted p-value; genes above it are dropped. Default 1 (no filtering). |
| ... | Additional arguments passed to glmQLFit for the null-model fit (used only when method = "quasi"). |

## Value

A data frame with columns logFC (log2 fold change of cluster vs cluster_2), logCPM, Prop.1 and Prop.2 (proportion of cells expressing the gene in group 1 and group 2 respectively), stat (signed quasi statistic), PValue (two-sided), and the adjusted p-value column.

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
spe <- getClusters(spe, resolution = 0.5)
# cluster 1 vs cluster 2
de <- getDE(spe, cluster = 1, cluster_2 = 2)
# clusters 1 & 3 vs clusters 2 & 4
de2 <- getDE(spe, cluster = c(1, 3), cluster_2 = c(2, 4))
```

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)$hvg as a logical vector. When a mean-dispersion trend is fitted (i.e. more eligible genes than requested HVGs), the per-gene trend is also stored in metadata(spe)$hvg_trend for use with plotHVG.

## Examples

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

`getMarkers` *Find up-regulated marker genes for all clusters (quasi-NB GLM).*

## Description

Pseudo-bulks cells by cluster via spe2PB and, for each cluster, tests its genes against the average of all other clusters (1-vs-rest) using a quasi-likelihood approach. Because the full design is saturated (one pseudo-bulk per cluster, 0 residual df), each contrast is tested by fitting the corresponding null design (one fewer column, 1 residual df) with glmQLFit and using the quasi-deviance chi-square statistic. The test is **one-sided**: p-values reflect up-regulation in the cluster, so each table ranks that cluster's positive markers.

## Usage

```
getMarkers(
  spe,
  cluster_name = "cluster",
  dispersion = 0.05,
  method = c("quasi", "lrt"),
  adjust.method = "BH",
  sort.by = "PValue",
  ...
)
```

## Arguments

| | |
|---|---|
| `spe` | A SpatialExperiment object. |
| `cluster_name` | Character. Column name in `colData(spe)` containing cluster labels. Default `"cluster"`. |
| `dispersion` | Numeric. Fixed NB dispersion (BCV^2) used for the GLM fits. Required because the saturated full design (one pseudo-bulk per cluster) leaves no residual df for dispersion estimation. Default 0.05 (BCV $\approx$ 0.224). |
| `method` | Character. Testing pipeline: "quasi" (default) fits each null design with glmQLFit and uses the quasi-deviance chi-square statistic; "lrt" uses the standard glmFit + glmLRT likelihood-ratio test. Both use the preset `dispersion`. |
| `adjust.method` | Character. Multiple-testing adjustment method passed to p.adjust (e.g. "BH", "BY", "holm", "bonferroni", "none"). Default "BH". |
| `sort.by` | Character. How to order each table: "PValue" (default), "logFC" (by absolute log fold change), "stat", or "none". |
| `...` | Additional arguments passed to glmQLFit for the null-model fits (used only when `method = "quasi"`). |

## Details

Each per-cluster table holds **all** genes, sorted by `sort.by` with an adjusted p-value column added via `adjust.method`. Use topMarkers to extract and combine the top genes per cluster. For a direct comparison between two clusters (or two groups of clusters), see getDE.

## Value

A named `list` of data frames, one per cluster, each containing all genes. Columns:

**logFC** log2 fold change of the cluster vs the average of the rest.

**logCPM** average log2 counts-per-million across pseudo-bulk samples.

**Prop** proportion of cells in the cluster with a non-zero count for the gene.

**stat** signed quasi statistic (signed square root of the quasi chi-square statistic); positive means up-regulated in the cluster.

**PValue** one-sided (up-regulation) p-value.

**FDR** adjusted p-value (column named per `adjust.method`; "FDR" for BH/BY, "FWER" for family-wise methods). Absent when `adjust.method = "none"`.

## Examples

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

---

# 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$nbrs[[at]]`. If NULL, uses the most recent spatial list from findNbrsSpatial at cell level, or the most recent grid list from findNbrsGrid at grid level. |

| | |
|---|---|
| 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")
```

---

# getSubClusters *Sub-cluster a single cluster.*

---

## Description

Take one cluster from an existing clustering and partition it further by community detection on the sub-graph of its cells, similar to `Seurat::FindSubCluster`. New sub-clusters are labelled `<cluster>_1`, `<cluster>_2`, ... (largest first).

## Usage

```
getSubClusters(
  spe,
  cluster,
  resolution = 1,
  nbrs_name = NULL,
  method = c("leiden", "louvain"),
  cluster_name = "cluster",
  new_name = cluster_name,
  relabel = FALSE,
  min_size = NULL,
  start_from = NULL,
  sep = "_",
  seed = NULL,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| spe | A SpatialExperiment object. |
| cluster | The cluster label to sub-cluster (e.g. 2 or "2"). |
| resolution | Resolution for the sub-clustering. Higher gives more sub-clusters. See cluster_leiden and cluster_louvain. |
| nbrs_name | Name of neighbour list to use. If NULL, uses the most recent SNN list from findNbrsSNN, which must have been run first. |
| method | Clustering method. Options are leiden and louvain. |
| cluster_name | Name of the cluster column in colData to sub-cluster. |
| new_name | Name of the column to store the result. Defaults to overwriting cluster_name. |
| relabel | Logical. If TRUE, renumber all clusters and sub-clusters by size (largest first) after sub-clustering, replacing the <cluster>_<n> labels with plain numbers. "unassigned" is kept as the last level. Defaults to FALSE. |
| min_size | Sub-clusters with at most this many cells are merged into the larger sub-clusters (graph connection first, then nearest centroid), as in getClusters. Defaults to NULL, which uses either 5 or 0.01\ the cells, whichever is smaller. |
| start_from | Integer at which numbering starts, for both the sub-cluster suffixes and the relabel renumbering. Defaults to NULL, which infers the base (0- or 1-based) from the existing labels. |
| sep | Separator between a cluster and its sub-cluster number. Default "_". |
| seed | Integer or NULL. If NULL (default), the clustering uses the current random number stream, so call set.seed() beforehand for reproducible sub-clusters. Otherwise the clustering uses this seed and the caller's random number stream is left unchanged. |
| verbose | Logical. Whether to report how the cluster was split (number of sub-clusters and any small ones merged in). Defaults to FALSE. |
| ... | Other clustering arguments for cluster_leiden or cluster_louvain. |

## Details

The sub-graph of the target cluster's cells is induced from the SNN graph and clustered on its own. Cells that are weakly connected within the cluster may form small or singleton sub-clusters; lower the resolution if this is not wanted.

## Value

A SpatialExperiment with the sub-clusters stored in colData.

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
set.seed(1)
```

```
spe <- getClusters(spe, resolution = 0.5)
# Split cluster 2 into 2_1, 2_2, ...
spe <- getSubClusters(spe, cluster = 2, resolution = 0.5)
```

---

`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$grid_density) or spatialCoords(spe), depending on 'at'. |
| `data2` | Numeric vector 2 for bivariate local Moran. Must be same length as data1. |
| `at` | Option of grid or cell for where to look for neighbour list |
| `nbrs_name` | Name of the neighbour list in `spe@metadata$nbrs[[at]]` for Moran's I. If NULL, uses the most recent spatial list from findNbrsSpatial at cell level, or the most recent grid list from findNbrsGrid at grid level. |
| `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$nbrs[[at]]` for Moran's I. If NULL, uses the most recent spatial list from findNbrsSpatial at cell level, or the most recent grid list from findNbrsGrid at grid level. |
| `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 with the local Moran values (lisa), clusters (cluster), pseudo p-values (p.value), spatial lags (spatiallag) and the sum of all neighbour weights (W). `sum(lisa)/W` is the global Moran's I, as returned by globalMoran.

## 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")
```

---

`mergeClusters` *Manually merge clusters.*

---

## Description

Combine user-specified clusters into a single cluster, optionally relabelling all clusters afterwards.

## Usage

```
mergeClusters(
  spe,
  merge,
  cluster_name = "cluster",
  new_name = cluster_name,
  merge_label = NULL,
  relabel = FALSE,
  start_from = NULL,
  sep = "&"
)
```

## Arguments

| | |
|---|---|
| `spe` | A SpatialExperiment object. |
| `merge` | A vector of cluster labels to merge into one (e.g. c(2, 5, 7)), or a list of such vectors to perform several merges in one call. |
| `cluster_name` | Name of the cluster column in colData to merge. |
| `new_name` | Name of the column to store the result. Defaults to overwriting cluster_name. |
| `merge_label` | Optional label for the merged cluster (single merge only). Defaults to the merged cluster labels joined by sep, e.g. "2&5&7". |
| `relabel` | Logical. If TRUE, renumber all clusters by size (largest first) after merging, replacing the merged label with a number. "unassigned" is always kept as the last level. Defaults to FALSE (keep the merged label). |

| | |
|---|---|
| start_from | Integer at which renumbering starts when relabel=TRUE. Defaults to NULL, which infers the base (0- or 1-based) from the existing labels, matching what getClusters produced. Ignored when relabel=FALSE. |
| sep | Separator for the auto-generated merged label. Default "&". |

## Details

Clusters listed in merge that are not present in cluster_name are ignored; a merge group with fewer than two present clusters is skipped with a warning. Any "unassigned" cells are left untouched (and kept as the last factor level) unless explicitly included in merge.

## Value

A SpatialExperiment with the merged clusters stored in colData.

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
spe <- getClusters(spe, resolution = 0.5)
# Merge clusters 2, 5 and 7 into one labelled "2&5&7"
spe <- mergeClusters(spe, merge = c(2, 5, 7))
# Merge and then renumber everything 1..K by size
spe <- mergeClusters(spe, merge = c(2, 5, 7), relabel = TRUE)
```

---

# 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 = 100,
  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", "cor.expected", "shift", "t", "p.Pos", "p.Neg"),
  roi = "all",
  cell.type = "all",
  silent = FALSE,
  level = c("overall", "ROI"),
  ...
)
```

### Arguments

| | |
|---|---|
| model.result | The output of corDensity(), a list with components 'ROI' and 'overall', or one of these DataFrames. |
| stats | Character value. The statistic to plot, one of "cor.coef" (default), "cor.expected", "shift", "t", "p.Pos" or "p.Neg". "cor.expected" and "shift" are only reported by corDensity(method = "permutation") and "t" only by corDensity(method = "modified.ttest"). |
| 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). |

| | |
|---|---|
| level | Character value, either "overall" or "ROI". Which component to plot when 'model.result' is the list returned by corDensity(). Defaults to "overall", or to "ROI" when specific ROIs are selected with 'roi'. Ignored when 'model.result' is a DataFrame. |
| ... | Other parameters passed to pheatmap(). |

## 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, nperm = 199)

plotCorHeatmap(model_result)

plotCorHeatmap(model_result, level = "ROI", stats = "shift")
```

---

# 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,
  range = 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) |
| range | Upper and lower bound for average expression. Values beyond this range are snapped to this limit. If only one value is provided, it is taken as the upper bound. |
| scale | Whether to scale the average expression data of each feature using scale. |
| cols | Custom colour palette (a vector of colours defining the fill gradient). Default NULL uses a light-grey (low) to dark-red (high) gradient. |
| dot.scale | scale the radius of the plot. See scale_radius |
| flip.axes | Whether to flip the axes. |

## Value

A ggplot object.

## 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",
  type = c("log", "raw", "cpm", "logcpm"),
  cols = NULL,
  highlight = NULL,
  cols.highlight = NULL,
  pt.shape = 16,
  pt.size = 0.7,
  pt.size.highlight = 1,
  pt.alpha = 0.6,
  label = NULL,
  xlab = NULL,
  ylab = NULL,
  cols.scale = NULL,
  ncol = NULL,
  per.scale = TRUE,
  range = NULL,
  transform = "identity"
)

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(s) to colour points by; must be in rownames(spe). If a vector of more than one feature is supplied, one reduced-dimension panel is drawn per feature, faceted (see `ncol`), with a shared colour scale.

`assay` Name of assay to use for plotting feature. Default "counts".

`type` Transformation applied to feature expression: "log" (default, log2(1+x)), "raw" (no transform), "cpm", or "logcpm" (log2 CPM). For multiple features the legend title defaults to the matching unit ("log2 Cts", "Counts", "CPM", "log2-CPM"); for a single feature the unit is the legend title and the gene name becomes the plot title. Override the legend with `label`.

`cols` Colour palette. Can be a vector of colours or a function that accepts an integer n and return n colours.

`highlight` Optional cells to emphasise, given as either a vector of group.by levels (characters or cluster numbers), or a logical vector of length ncol(spe) selecting cells directly. Highlighted cells are drawn last (on top) at pt.size.highlight; all other cells are light grey at pt.size.

`cols.highlight` Colour(s) for the highlighted cells. Defaults to NULL, which keeps each level's usual group.by palette colour. A single colour (e.g. "red") colours all highlighted cells the same; a vector matching the number of 'highlight' entries gives one colour per level (matched by position).

`pt.shape` shape of points.

`pt.size` size of points.

`pt.size.highlight` size of highlighted points (see highlight).

`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.

`ncol` Number of columns when plotting multiple features. Passed to facet_wrap (shared scale) or wrap_plots (per-panel scales). Default NULL lets the layout be chosen automatically.

`per.scale` Logical. For multiple features, whether each panel gets its own colour scale (TRUE, default; like `Seurat::FeaturePlot`, via the 'patchwork' package) or a single shared colour scale across panels (FALSE).

`range` Numeric length-2 (lower, upper) cap for continuous colour values

- a single feature, or multiple features with a shared scale; values outside are clamped. A single value is taken as the upper bound. Default NULL (no capping). Ignored when `per.scale = TRUE` (each panel auto-scales).

| | |
|---|---|
| transform | Name of a transformation for the continuous colour scale (e.g. "log10", "log1p", "pseudo_log", "sqrt"), passed to scale_color_gradientn; the colour spectrum is spaced by the transform while the legend stays in original units. Default "identity" (no transformation). Use "log10" for nicely log-spaced legend breaks (needs positive values); "log1p"/"pseudo_log" tolerate zeros but keep linear breaks. Only affects continuous (feature or numeric group.by) colouring. |
| ... | 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")
```

plotHVG *Plot the mean-dispersion trend from getHVG().*

## Description

Visualises each eligible gene's dispersion against its mean expression, overlaid with the fitted lowess trend, and highlights the selected highly variable genes. Requires `getHVG` to have been run with a fitted trend (i.e. more eligible genes than requested HVGs).

## Usage

```
plotHVG(
  spe,
  pt.size = 0.6,
  pt.alpha = 0.6,
  cols = c("grey70", "firebrick"),
  line.col = "blue"
)
```

## Arguments

| | |
|---|---|
| `spe` | A SpatialExperiment processed by `getHVG`. |
| `pt.size` | Point size. Default 0.6. |
| `pt.alpha` | Point alpha (0-1). Default 0.6. |
| `cols` | Length-2 vector of colours for non-HVG and HVG points. Default `c("grey70", "firebrick")`. |
| `line.col` | Colour of the fitted trend line. Default "blue". |

## Value

A ggplot object (dispersion vs mean expression, both on log10 axes).

## Examples

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

## 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 |

## Details

Each point is a cell or grid point, with its standardised value on the x-axis and its spatial lag on the y-axis. The title gives the global Moran's I, as returned by globalMoran. The line is the least-squares fit of the spatial lag on the value; its slope equals Moran's I only when the weights are row-standardised.

## 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("log", "raw", "cpm", "logcpm"),
  cols = NULL,
  highlight = NULL,
  cols.highlight = NULL,
  pt.shape = 16,
  pt.size = 0.3,
  pt.size.highlight = 1,
  pt.alpha = 1,
  label = NULL,
  cols.scale = NULL,
  reverseY = NULL,
  image = FALSE,
  ncol = NULL,
  per.scale = TRUE,
  range = NULL,
  transform = "identity",
  ...
)
```

## 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(s) to colour points by; must be in rownames(spe). If a vector of more than one feature is supplied, one panel is drawn per feature (see per.scale, ncol), as in Seurat::FeaturePlot. |
| 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. For feature plots the legend title defaults to the matching unit ("Counts", "log2 Cts", "CPM", "log2-CPM"); for a single feature the gene name becomes the plot title. Override the legend with label. |

`cols` Colour palette. Can be a vector of colours or a function that accepts an integer n and return n colours.

`highlight` Optional cells to emphasise, given as either a vector of group.by levels (characters or cluster numbers), or a logical vector of length ncol(spe) selecting cells directly. Highlighted cells are drawn last (on top) at pt.size.highlight; all other cells are light grey at pt.size.

`cols.highlight` Colour(s) for the highlighted cells. Defaults to NULL, which keeps each level's usual group.by palette colour. A single colour (e.g. "red") colours all highlighted cells the same; a vector matching the number of 'highlight' entries gives one colour per level (matched by position).

`pt.shape` shape of points.

`pt.size` size of points.

`pt.size.highlight` size of highlighted points (see highlight).

`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.

`image` Logical. Whether to draw the background tissue image (forwarded to plotImage). Default FALSE.

`ncol` Number of columns when plotting multiple features. Passed to facet_wrap (shared scale) or wrap_plots (per-panel scales). Default NULL lets the layout be chosen automatically.

`per.scale` Logical. For multiple features, whether each panel gets its own colour scale (TRUE, default; like `Seurat::FeaturePlot`, via the 'patchwork' package; the image is drawn in every panel) or a single shared colour scale across panels (FALSE).

`range` Numeric length-2 (lower, upper) cap for continuous colour values

- a single feature, or multiple features with a shared scale; values outside are clamped. A single value is taken as the upper bound. Default NULL (no capping). Ignored when `per.scale = TRUE` (each panel auto-scales).

`transform` Name of a transformation for the continuous colour scale (e.g. "log10", "log1p", "pseudo_log", "sqrt"), passed to scale_color_gradientn; the colour spectrum is spaced by the transform while the legend stays in original units. Default "identity" (no transformation). Use "log10" for nicely log-spaced legend breaks (needs positive values); "log1p"/"pseudo_log" tolerate zeros but keep linear breaks. Only affects continuous (feature or numeric group.by) colouring. Use "log10" for log-spaced legend breaks (requires positive values); "log1p"/"pseudo_log" tolerate zeros but keep linear breaks.

`...` 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)
```

---

`plotTopMarkers` *Dot plot or heatmap of the top marker genes per cluster.*

---

## Description

Takes the output of getMarkers and visualises the top `n` marker genes of each cluster, either as a dot plot or a genes-by-clusters expression heatmap. Genes are block-ordered by cluster (in the order of `names(markers)`); a gene that is a top marker for more than one cluster is shown once, under the first cluster it appears in.

## Usage

```
plotTopMarkers(
  markers,
  spe,
  n = 5,
  min.prop = 0,
  type = c("dotplot", "heatmap"),
  profile = c("pseudobulk", "cell"),
  cluster_name = "cluster",
  assay = "counts",
  scale = TRUE,
  range = NULL,
  cols = NULL,
  dot.scale = 6,
  ...
)
```

## Arguments

| | |
|---|---|
| `markers` | The named list returned by getMarkers (one data frame per cluster, gene names as row names). |
| `spe` | The SpatialExperiment object the markers were computed from. |
| `n` | Integer. Number of top markers to take from each cluster. Default 5. |

| | |
|---|---|
| min.prop | Numeric in [0, 1]. Drop candidate markers whose detection rate (proportion of cells expressing) in their cluster is below this value before taking the top n, so low-prevalence genes are skipped and back-filled from further down the ranking. Default 0 (no filtering). |
| type | Character. "dotplot" (default) or "heatmap". |
| profile | Character. "pseudobulk" (default) or "cell"; see Details. |
| cluster_name | Character. Column in colData(spe) holding the cluster labels used in markers. Default "cluster". |
| assay | Name of the assay to use for expression. Only used when profile = "cell" (pseudo-bulk always uses raw counts). Default "counts". |
| scale | Logical. Scale expression per gene (z-score across clusters for the colour). Default TRUE. |
| range | Numeric. Lower and upper caps for the colour values; values outside are clamped (winsorised). A single value is taken as the upper bound. Default NULL picks c(-1.5, 2.2) when scale = TRUE (suited to z-scores) and c(-Inf, Inf) (no capping) otherwise. |
| cols | Custom colour palette. For a cell-level dot plot a gradient passed to plotDots; otherwise the full colour vector for the gradient (dot plot) or pheatmap (heatmap). |
| dot.scale | Numeric. Scales the dot radius. See scale_radius. Default 6. |
| ... | Further arguments passed to plotDots (cell-level dot plot) or pheatmap (heatmap). |

## Details

Two expression profiles are available via profile:

- **"pseudobulk"** (default): per-cluster log-CPM from spe2PB + cpm - the same space the markers were tested in. Dot colour is (scaled) log-CPM; dot size is the proportion of cells expressing the gene.
- **"cell"**: Seurat-style cell-level summaries via plotDots (dot plot) or cell-level mean expression (heatmap).

## Value

A ggplot object when type = "dotplot", or a pheatmap object when type = "heatmap".

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
spe <- getClusters(spe, resolution = 0.5)
markers <- getMarkers(spe)
plotTopMarkers(markers, spe, n = 5)
plotTopMarkers(markers, spe, n = 5, type = "heatmap")
```

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 |

## Value

A ggplot object.

## 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

## Value

A SpatialExperiment object containing the expected counts, cell metadata and cell coordinates.

`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,
  feature_type = "Gene Expression",
  pixel_to_micron = TRUE
)
```

## Arguments

| | |
|---|---|
| `dir` | directory containing the Visium files |
| `sample_id` | Name of the sample. |
| `count` | Name of the h5 file with the count assay. |
| `coord` | Path to the tissue coordinates file (csv or parquet), or a data.frame of coordinates (rownames = barcodes) with columns 'pxl_col_in_fullres' and 'pxl_row_in_fullres'. |
| `image` | Names of the image files. |
| `scale_factors` | Names of the scale factors file |
| `feature_type` | Feature type to retain. Defaults to "Gene Expression" to exclude non-gene features. Set to NULL to keep all features. |
| `pixel_to_micron` | Logical. If TRUE (default), convert the spot coordinates from full-resolution pixels to microns using 'spot_diameter_fullres' from the scale factors file, and store the conversion factor in metadata(spe)$um_per_pixel. Set to FALSE to keep the coordinates in pixels (previous behaviour). |

## Value

A SpatialExperiment object containing the counts, spot metadata and spatial coordinates.

`readVisiumHD` *Read VisiumHD output into spe*

## Description

Read VisiumHD output into spe

## Usage

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

## Arguments

| | |
|---|---|
| `dir` | directory containing the VisiumHD files |
| `bin` | Which output to read. Bin sizes "016um", "008um", "002um" read the corresponding 'binned_outputs/square_*' folder. "segmented" reads the cell-segmentation results in 'segmented_outputs': the count matrix 'filtered_feature_cell_matrix.h5', with per-cell coordinates taken from the centroids of 'cell_segmentations.geojson'. |
| `...` | Parameters for readVisium |

## Details

For "segmented", cells have no array_row/array_col grid, so the result is cell-level (like Xenium) and is not compatible with the Visium spot-grid options of gridDensity() / trimEdge().

## Value

A SpatialExperiment object containing the counts and the coordinates of the bins or segmented cells.

`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,
  feature_type = "Gene Expression"
)
```

### 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. Reading the image requires the suggested package RBioFormats; without it, the image is skipped. |
| 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 |
| feature_type | Feature type to retain. Defaults to "Gene Expression" to exclude control codewords. Set to NULL to keep all features. |

### Value

A SpatialExperiment object containing the counts, cell metadata and cell centroid coordinates.

---

## 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

## Value

A SpatialExperiment object with spatial coordinates rescaled to microns and aligned to the spot array.

---

`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

## Value

A SpatialExperiment object with spatial coordinates rescaled to microns and aligned to the bin grid.

`relabelClusters` *Renumber all clusters by size.*

## Description

Renumbers the clusters in a colData column `1..K` by size (largest first), with "unassigned" kept as the last level. Useful for tidying up the compound labels left by `getSubClusters(relabel = FALSE)` / `mergeClusters(relabel = FALSE)` after a multi-step workflow.

## Usage

```
relabelClusters(
  spe,
  cluster_name = "cluster",
  new_name = cluster_name,
  start_from = NULL
)
```

## Arguments

| | |
|---|---|
| `spe` | A SpatialExperiment object. |
| `cluster_name` | Name of the cluster column in colData to renumber. |
| `new_name` | Name of the column to store the result. Defaults to overwriting cluster_name. |
| `start_from` | Integer at which numbering starts. Defaults to NULL, which infers the base (0- or 1-based) from the existing labels. |

## Details

Renumbering is size-based, so run it **before** cell-type annotation - relabelling after a cluster-to-cell-type mapping would scramble that mapping.

## Value

A SpatialExperiment with the renumbered clusters in colData.

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
spe <- getClusters(spe, resolution = 0.5)
spe <- getSubClusters(spe, cluster = 2, resolution = 0.5)
# Tidy the 2_1/2_2/... labels into plain 1..K numbers by size
spe <- relabelClusters(spe)
```

runPCA *Fast PCA using irlba.*

## Description

Fast PCA using irlba.

## Usage

```
runPCA(
  spe,
  n_pcs = 50,
  assay = "logcounts",
  centre = TRUE,
  scale = TRUE,
  clip = NULL,
  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. |
| clip | Maximum absolute z-score after scaling. Values beyond this are clipped. Prevents rare-marker genes (near-zero SD) from creating extreme outliers that fragment the UMAP. Defaults to 10. Set to Inf to disable. |
| 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,
  spread = 1,
  assay = NULL,
  dimred = "PCA",
  n_dimred = NULL,
  name = "UMAP",
  ...
)
```

## Arguments

| | |
|---|---|
| spe | A SpatialExperiment object. |
| n_neighbors, n_components, metric, min_dist, spread | 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)
```

---

`topMarkers` *Combine the top markers of each cluster into one data frame.*

---

## Description

Convenience wrapper around getMarkers output: takes the top `n` genes from each cluster's (already ranked) table and stacks them into a single data frame, with a leading `cluster` column and a `gene` column (the same gene can be a top marker for more than one cluster, so row names cannot stay unique). Analogous to the single table returned by `Seurat::FindAllMarkers`.

## Usage

```
topMarkers(markers, n = 10, p.value = 1, min.prop = 0)
```

## Arguments

| | |
|---|---|
| `markers` | The named list returned by getMarkers. |
| `n` | Integer. Number of top genes to take from each cluster. Default 10. |
| `p.value` | Numeric. Keep only genes whose adjusted p-value (or raw `PValue` if no adjusted column is present) is at or below this cutoff, applied before taking the top `n`. Default 1 (no filtering). |
| `min.prop` | Numeric in [0, 1]. Drop genes whose detection rate (the `Prop` column - proportion of cells expressing in the cluster) is below this value before taking the top `n`, so the slots are back-filled from further down the ranking. Default 0 (no filtering). |

## Value

A data frame with columns `cluster`, `gene`, and the per-cluster statistics from getMarkers, ordered by cluster then by each cluster's ranking.

## Examples

```
data("xenium_bc_spe")
spe <- normalizeAssay(spe)
spe <- runPCA(spe)
spe <- findNbrsSNN(spe, dimred = "PCA")
spe <- getClusters(spe, resolution = 0.5)
markers <- getMarkers(spe)
top <- topMarkers(markers, n = 10)
```

---

# trimEdge *Trim spots/bins from the edges of a Visium (or VisiumHD) slide*

---

## Description

Removes whole lines of spots from the outer edges of an aligned Visium or VisiumHD array, using the regular `array_row`/`array_col` grid. Useful for quickly shaving off edge artefacts (folds, tears, capture effects).

## Usage

```
trimEdge(spe, trim = c(0, 0, 0, 0))
```

## Arguments

| | |
|---|---|
| `spe` | A SpatialExperiment with `array_row` and `array_col` in its `colData`. |
| `trim` | Integer vector of length 4 giving the number of spot lines to remove from each edge, in the order `c(bottom, left, top, right)` (the base R `par` `mar` order). Default `c(0, 0, 0, 0)` (no trimming). |

## Details

Edges are defined in the **stored-coordinate** frame, which is independent of any plotting choice: `left` = smallest x, `right` = largest x, `top` = smallest y, `bottom` = largest y. This matches the orientation of scider's default *image* plot (which is Y-reversed). Note that a plot drawn *without* the Y-reversal (e.g. an object read without an image, or `reverseY = FALSE`) is vertically mirrored, so `top` then appears at the bottom of that plot.

Trimming removes the `n` smallest/largest `array_col` values (for `left`/`right`) and `array_row` values (for `top`/ `bottom`). Because of the Visium hexagonal offset, one `array_col` value spans only alternate rows, so removing a single value trims roughly half a visual column at a time; use `2` for a full column line. `array_row` values are clean horizontal lines. VisiumHD's square grid has no such half-offset.

This function requires `array_row`/`array_col` in `colData(spe)` and therefore only applies to Visium/VisiumHD.

## Value

The SpatialExperiment with the edge spots removed.

## Examples

```
## Not run:
spe <- readVisium("path/to/visium/outs")
# 1 line off the bottom, 2 off the left, 3 off the top, 4 off the right
spe <- trimEdge(spe, trim = c(1, 2, 3, 4))

## End(Not run)
```

---

`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

# Index


[,SpatialExperiment,ANY,ANY,ANY-method, 3

allocateCells, 4

BiocNeighborParam, *13*

cellsInRegion, 5
cluster_leiden, *17*, *25*
cluster_louvain, *17*, *25*
colData, *17*, *25*, *32*, *63*
computeDensity, 5
computeDensityHex, 6
contour2sf, 7
coord_hash, 8
corDensity, 9
cpm, *55*

facet_wrap, *44*, *53*
findKNN, *14*
findNbrsGrid, 11, *23*, *26*, *31*
findNbrsSNN, 12, *17*, *25*
findNbrsSpatial, 14, *23*, *26*, *31*
findNeighbors, *14*
findROI, 15

getClusters, 16, *25*
getContour, 18
getContourRegions, 19
getDE, 19, *22*
getHVG, 21, *47*, *64*
getMarkers, *19*, 22, *54*, *68*
getNiche, 23
getSubClusters, 24
glmFit, *20*, *22*
glmLRT, *20*, *22*
glmQLFit, *19*, *20*, *22*
glmQLFTest, *19*
globalMoran, 26, *31*, *50*
grid2df, 27
grid2sf, 28

gridDensity, 29
gridSPE, 30

irlba, *64*

localMoran, 31, *49*, *50*

mergeClusters, 32
mergeROI, 33

normalizeAssay, 34, *64*

p.adjust, *20*, *22*
par, *69*
pheatmap, *55*
plotCellCompo, 35
plotContour, 36
plotContourRegion, 37
plotCorHeatmap, 38
plotDensCor, 39
plotDensity, *37*, 41
plotDots, 42, *55*
plotDR, 43
plotGrid, *41*, 45, *49*
plotHVG, 47
plotImage, *37*, *46*, 48, *53*
plotLISA, 48
plotLISAscatter, 49
plotPCA *(*plotDR*)*, 43
plotROI, 50
plotSpatial, *37*, *49*, *51*, 52
plotTopMarkers, 54
plotUMAP *(*plotDR*)*, 43
plotViolin, 56
postSelRegion, 57

readProseg, 58
readVisium, 59
readVisiumHD, 60
readXenium, 60
realignVisium, 61

realignVisiumHD, 62
reducedDims, *13*, *17*, *43*, *64–66*
relabelClusters, 63
runPCA, 64, *65*
runUMAP, 65

scale, *42*
scale_color_gradientn, *44, 45*, *53*
scale_fill_gradientn, *46*
scale_radius, *42*, *55*
scaleFactors, *48*
selectRegion, 66
spe2PB, *19*, *22*, *55*, 67

topMarkers, *22*, 68
topTags, *19*
trimEdge, 69

umap, *65*
update_bound, 70

wrap_plots, *44*, *53*