Package 'sosta'

November 12, 2026

Title A package for the analysis of anatomical tissue structures in spatial omics data

Version 1.4.0

Description sosta (Spatial Omics STructure Analysis) is a package for analyzing spatial omics data to explore tissue organization at the anatomical structure level. It reconstructs anatomically relevant structures based on molecular features or cell types. It further calculates a range of metrics at the structure level to quantitatively describe tissue architecture. The package is designed to integrate with other packages for the analysis of spatial omics data.

License GPL (>= 3) + file LICENSE

Encoding UTF-8

Depends R (>= 4.4.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/sgunz/sosta, https://sgunz.github.io/sosta/

BugReports https://github.com/sgunz/sosta/issues

biocViews Software, Spatial, Transcriptomics, Visualization

Imports terra, sf, smoothr, spatstat.explore, spatstat.geom, SpatialExperiment, SingleCellExperiment, dplyr, ggplot2, patchwork, SummarizedExperiment, stats, rlang, parallel, EBImage, spatstat.random, S4Vectors

Suggests knitr, rmarkdown, BiocStyle, ExperimentHub, lme4, lmerTest, ggfortify, tidyr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

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

git_branch RELEASE_3_23

git_last_commit e6bf7c2

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Samuel Gunz [aut, cre] (ORCID: <https://orcid.org/0000-0002-8909-0932>), Mark D. Robinson [aut, fnd]

Maintainer Samuel Gunz <samuel.gunz@uzh.ch>

Contents

sosta-package . . . 3
.df2ppp . . . 3
.intensityImage . . . 4
.intensityThreshold . . . 5
.SPE2df . . . 5
assingCellsToStructures . . . 6
binaryImageToSF . . . 7
cellTypeProportions . . . 8
createPointPatternTissue . . . 9
estimateReconstructionParametersSPE . . . 10
findIntensityThreshold . . . 11
getDimXY . . . 11
meanShapeMetrics . . . 12
minBoundaryDistances . . . 13
minCellTypeStructDist . . . 14
normalizeCoordinates . . . 15
reconstructShapeDensity . . . 16
reconstructShapeDensityImage . . . 17
reconstructShapeDensitySPE . . . 18
shapeIntensityImage . . . 19
shapeMetrics . . . 20
simulateTissueBlobs . . . 21
sostaSPE . . . 21
spatialCoords2SF . . . 22
SPE2ppp . . . 23
stCalculateCurvature . . . 23
stCalculateShapeCurl . . . 24
stFeatureAxes . . . 25
totalShapeMetrics . . . 26
xyCoordinates . . . 26

Index **28**

sosta-package — *sosta: A package for the analysis of anatomical tissue structures in spatial omics data*

Description

sosta (Spatial Omics STructure Analysis) is a package for analyzing spatial omics data to explore tissue organization at the anatomical structure level. It reconstructs anatomically relevant structures based on molecular features or cell types. It further calculates a range of metrics at the structure level to quantitatively describe tissue architecture. The package is designed to integrate with other packages for the analysis of spatial omics data.

Author(s)

Maintainer: Samuel Gunz <samuel.gunz@uzh.ch> (ORCID)

Authors:

- Mark D. Robinson <mark.robinson@mls.uzh.ch> [funder]

See Also

Useful links:

- https://github.com/sgunz/sosta
- https://sgunz.github.io/sosta/
- Report bugs at https://github.com/sgunz/sosta/issues

.df2ppp — *Function to convert* data.frame *to* ppp *object*

Description

Assumes that the data.frame is the output of .SPE2df(). Column order is important!

Usage

```
.df2ppp(df, xName, yName, marks = NULL)
```

Arguments

df	data.frame; with x, y coordinates, image, and categorical mark information.
xName	character; column name of x coordinate
yName	character; column name of y coordinate
marks	character; column name of the mark variable

Value

ppp; object of type ppp

See Also

.SPE2df, as.ppp

Examples

```
data(sostaSPE)
df <- .SPE2df(sostaSPE, marks = "cellType", imageCol = "imageName")
ppp <- .df2ppp(df, xName = "x", yName = "y", marks = "cellType")
```

.intensityImage *Function to estimate the intensity image of a point pattern*

Description

Function to estimate the intensity image of a point pattern

Usage

```
.intensityImage(ppp, markSelect = NULL, bndw = NULL, dim)
```

Arguments

ppp	point pattern object of class ppp
markSelect	character; name of mark that is to be selected for the reconstruction
bndw	bandwidth of kernel density estimator
dim	numeric; x dimension of the final reconstruction.

Value

list; list with the intensity image and the bandwidth and dimension parameters

.intensityThreshold *Function to estimate the intensity threshold for the reconstruction of spatial structures*

Description

Function to estimate the intensity threshold for the reconstruction of spatial structures

Usage

```
.intensityThreshold(densityImage, steps = 250, minRange = 0.15)
```

Arguments

densityImage	real-valued pixel image; output from the function .intensityImage
steps	numeric; value used to filter the density estimates, where only densities greater than the maximum value divided by threshold are considered. Default is 250.
minRange	numeric; value used to filter minimal threshold in percent of total range. Should range between (0,1].

Value

numeric; estimated threshold

.SPE2df *Function to convert* SpatialExperiment *object to a data frame*

Description

Function to convert SpatialExperiment object to a data frame

Usage

```
.SPE2df(spe, imageCol = NULL, marks = NULL, colNames = FALSE)
```

Arguments

spe	SpatialExperiment; a object of class SpatialExperiment
imageCol	character; name of a column in colData that corresponds to the image
marks	character; name of column in colData with categorical marks
colNames	logical; extract colnames from SpatialExperiment

Value

data.frame with x, y coordinates, image, and categorical mark information

Examples

```
data(sostaSPE)
.SPE2df(sostaSPE, marks = "cellType", imageCol = "imageName") |> head()
```

`assingCellsToStructures` *Function to assign points / coordinates to structures*

Description

This function assigns each spatial coordinate in a `SpatialExperiment` object (`spe`) to the first intersecting structure from a given set of spatial structures.

Usage

```
assingCellsToStructures(
  spe,
  allStructs,
  imageCol = NULL,
  uniqueId = "structID",
  nCores = 1
)
```

Arguments

`spe`	SpatialExperiment; An object of class `SpatialExperiment` containing spatial point data. Must contain `colnames` for correct assignment.
`allStructs`	sf; A simple feature collection (sf object) representing spatial structures. Must contain a column which contains a unique identifier for each structure. Default = `structID`.
`imageCol`	character; The column name in `spe` and `allStructs` that identifies the corresponding image.
`uniqueId`	character; The column name in the simple feature collection for which to compute the assignment.
`nCores`	integer; The number of cores to use for parallel processing (default is 1).

Value

A named list with structure assignments for each spatial point in `spe`. Points that do not overlap with any structure are assigned `NA`. Names correspond to `colnames` of the `SpatialExperiment` input object.

Examples

```
library("SpatialExperiment")
data("sostaSPE")
allStructs <- reconstructShapeDensitySPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", bndw = 3.5, thres = 0.045
)
# The function `assingCellsToStructures` needs colnames so we create them here
colnames(sostaSPE) <- paste0("cell_", c(1:dim(sostaSPE)[2]))

res <- assingCellsToStructures(
    spe = sostaSPE, allStructs = allStructs, imageCol = "imageName"
)
# Assign the structure assignment in the order of the columns in the `SpatialExperiment` object
colData(sostaSPE)$structAssign <- res[colnames(sostaSPE)]

if (require("ggplot2")) {
    cbind(
        colData(sostaSPE[, sostaSPE[["imageName"]] == "image1"]),
        spatialCoords(sostaSPE[, sostaSPE[["imageName"]] == "image1"])
    ) |>
        as.data.frame() |>
        ggplot(aes(x = x, y = y, color = structAssign)) +
        geom_point(size = 0.25) +
        coord_equal()
}
```

---

binaryImageToSF *Converts a binary matrix to an sf polygon*

Description

Converts a binary matrix to an sf polygon

Usage

```
binaryImageToSF(binaryMatrix, xmin, xmax, ymin, ymax)
```

Arguments

binaryMatrix	matrix; binary matrix
xmin	integer; minimum x coordinate of the coordinate system
xmax	integer; maximum x coordinate of the coordinate system
ymin	integer; minimum y coordinate of the coordinate system
ymax	integer; maximum y coordinate of the coordinate system

Value

sf object

Examples

```
matrixR <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
polyR <- binaryImageToSF(matrixR, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
plot(polyR)
```

---

cellTypeProportions *Calculate the proportion of each cell type within spatial structures*

Description

Calculate the proportion of each cell type within spatial structures

Usage

```
cellTypeProportions(spe, structColumn, cellTypeColumn, nCores = 1)
```

Arguments

spe	SpatialExperiment object
structColumn	character; name of the colData column specifying the structure assignments
cellTypeColumn	character; name of the colData column specifying cell types
nCores	integer; The number of cores to use for parallel processing (default is 1).

Value

A data frame where rows correspond to unique structures and columns correspond to cell types, containing the proportion of each cell type within each structure.

Examples

```
library("SpatialExperiment")
data("sostaSPE")
allStructs <- reconstructShapeDensitySPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", bndw = 3.5, thres = 0.045
)
# The function `assingCellsToStructures` needs colnames so we create them here
colnames(sostaSPE) <- paste0("cell_", c(1:dim(sostaSPE)[2]))
```

```
# Assign the structure assignment in the order of the columns in the `SpatialExperiment` object
colData(sostaSPE)$structAssign <- assingCellsToStructures(
    spe = sostaSPE, allStructs = allStructs, imageCol = "imageName"
)[colnames(sostaSPE)]
cellTypeProportions(sostaSPE, "structAssign", "cellType")
```

`createPointPatternTissue` *Create a Point Pattern on a Simulated Tissue Image*

Description

This function generates a spatial point pattern with different types of points (`A`, `B`, `C`) distributed over the simulated tissue structure.

Usage

```
createPointPatternTissue(
  tissueImage,
  intA,
  intB,
  noiseA = 0.005,
  intCInA,
  intCInB
)
```

Arguments

`tissueImage`	Matrix; A binary matrix representing the simulated tissue.
`intA`	Numeric; Intensity of type "A" points (points per unit area) on tissue regions.
`intB`	Numeric; Intensity of type "B" points (points per unit area) on non-tissue regions.
`noiseA`	Numeric; Intensity of type "A" points (points per unit area) on non-tissue regions.
`intCInA`	Numeric; Intensity of type "C" points placed in extended regions around tissue.
`intCInB`	Numeric; Intensity of type "C" points placed within tissue.

Value

A `ppp` object representing the spatial point pattern.

Examples

```
tissueImage <- simulateTissueBlobs(128, 100, 7)
createPointPatternTissue(tissueImage, 0.01, 0.01, 0.005, 0.005, 0.005)
```

`estimateReconstructionParametersSPE` *Estimate reconstruction parameters from a set of images*

Description

Estimate reconstruction parameters from a set of images

Usage

```
estimateReconstructionParametersSPE(
  spe,
  marks,
  imageCol = NULL,
  markSelect = NULL,
  nImages = NULL,
  fun = "bw.diggle",
  dim = 500,
  nCores = 1,
  plotHist = TRUE
)
```

Arguments

`spe`	SpatialExperiment; a object of class `SpatialExperiment`
`marks`	character; name of column in `colData` that will correspond to the ppp marks
`imageCol`	character; name of a column in `colData` that corresponds to the image
`markSelect`	character; name of mark that is to be selected for the reconstruction
`nImages`	integer; number of images for the estimation. Will be randomly sampled
`fun`	character; function to estimate the kernel density. Default bw.diggle.
`dim`	numeric; x dimension of the final reconstruction. A lower resolution speed up computation but lead to less exact reconstruction. Default = 500
`nCores`	numeric; number of cores for parallel processing using `mclapply`. Default = 1
`plotHist`	logical; if histogram of estimated densities and thresholds should be plotted (only if `imageCol` is not NULL). Default = TRUE

Value

tibble; tibble with estimated intensities and thresholds

Examples

```
data("sostaSPE")
estimateReconstructionParametersSPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", plotHist = TRUE
)
```

findIntensityThreshold

Estimate the intensity threshold for the reconstruction of spatial structures

Description

Estimate the intensity threshold for the reconstruction of spatial structures

Usage

```
findIntensityThreshold(ppp, markSelect = NULL, bndw = NULL, dim, steps = 250)
```

Arguments

ppp	point pattern object of class ppp
markSelect	character; name of mark that is to be selected for the reconstruction
bndw	numeric; bandwith of the sigma parameter in the density estimation, if no value is given the bandwith is estimated using cross validation with the bw.diggle function.
dim	numeric; x dimension of the final reconstruction.
steps	numeric; value used to filter the density estimates, where only densities greater than the maximum value divided by threshold are considered. Default is 250.

Value

numeric; estimated intensity threshold

Examples

```
data(sostaSPE)
ppp <- SPE2ppp(sostaSPE, marks = "cellType", imageCol = "imageName", imageId = "image1")
findIntensityThreshold(ppp, markSelect = "A", dim = 250)
```

getDimXY

Function to get the dimension based on dim of y axis

Description

Function to get the dimension based on dim of y axis

Usage

```
getDimXY(ppp, ydim)
```

Arguments

ppp	point pattern object of class ppp
ydim	dimension of y axis

Value

vector; vector with x and y dimension

Examples

```
data(sostaSPE)
pp <- SPE2ppp(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    imageId = "image1"
)
getDimXY(pp, 500)
```

---

meanShapeMetrics *Calculate mean shape metrics of a set of polygons*

Description

Calculate mean shape metrics of a set of polygons

Usage

```
meanShapeMetrics(totalShapeMetricMatrix)
```

Arguments

totalShapeMetricMatrix	matrix of shape metrics

Value

matrix; matrix of mean shape metrics

Examples

```
data(sostaSPE)
struct <- reconstructShapeDensityImage(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    imageId = "image1", markSelect = "A", dim = 500
)
shapeMetrics <- totalShapeMetrics(struct)
meanShapeMetrics(shapeMetrics)
```

`minBoundaryDistances` *Compute minimum boundary distances for each cell within its corresponding image structures*

Description

Compute minimum boundary distances for each cell within its corresponding image structures

Usage

```
minBoundaryDistances(
  spe,
  imageCol,
  structColumn = NULL,
  allStructs,
  nCores = 1
)
```

Arguments

`spe`	SpatialExperiment object
`imageCol`	character; name of the `colData` column specifying the image name
`structColumn`	character; name of the `colData` column specifying structure assignments. Default = NULL.
`allStructs`	sf object; contains spatial structures with corresponding image names
`nCores`	integer; The number of cores to use for parallel processing (default is 1).

Value

A named list containing the minimum distances between cells and structure boundaries, values within structures have negative values. Names correspond to `colnames` of the `SpatialExperiment` input object.

Examples

```
library("SpatialExperiment")
data("sostaSPE")

allStructs <- reconstructShapeDensitySPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", bndw = 3.5, thres = 0.045
)
# The function `assingCellsToStructures` needs colnames so we create them here
colnames(sostaSPE) <- paste0("cell_", c(1:dim(sostaSPE)[2]))
# Assign the structure assignment in the order of the columns in the `SpatialExperiment` object
colData(sostaSPE)$structAssign <- assingCellsToStructures(
    spe = sostaSPE, allStructs = allStructs, imageCol = "imageName"
```

```
)[colnames(sostaSPE)]

res <- minBoundaryDistances(
    spe = sostaSPE, imageCol = "imageName", structColumn = "structAssign",
    allStructs = allStructs
)

colData(sostaSPE)$minDist <- res[colnames(sostaSPE)]

if (require("ggplot2")) {
    cbind(colData(sostaSPE), spatialCoords(sostaSPE)) |>
        as.data.frame() |>
        ggplot(aes(x = x, y = y, color = minDist)) +
        geom_point(size = 0.25) +
        scale_colour_gradient2() +
        geom_sf(data = allStructs, fill = NA, inherit.aes = FALSE) +
        facet_wrap(~imageName)
}
```

`minCellTypeStructDist` *Compute minimum distances from each cell types to structure boundaries per structure*

Description

Compute minimum distances from each cell types to structure boundaries per structure

Usage

```
minCellTypeStructDist(
  spe,
  allStructs,
  structID = "structID",
  cellTypeColumn,
  imageCol,
  nCores = 1
)
```

Arguments

`spe`	SpatialExperiment object
`allStructs`	sf object; contains spatial structures with corresponding image names
`structID`	character; name of the column in `allStructs` containing structure IDs (default: "structID")
`cellTypeColumn`	character; name of the `colData` column specifying cell types
`imageCol`	character; name of the `colData` column specifying the image name
`nCores`	integer; Number of cores for parallel processing (default = 1)

Value

A data frame where rows are structure IDs and cols are cell types with minimum distances

Examples

```
data("sostaSPE")
allStructs <- reconstructShapeDensitySPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", bndw = 3.5, thres = 0.045
)
minCellTypeStructDist(sostaSPE, allStructs, cellTypeColumn = "cellType", imageCol = "imageName")
```

---

normalizeCoordinates *Function to normalize coordinates between zero and one while keep scaling*

Description

Function to normalize coordinates between zero and one while keep scaling

Usage

```
normalizeCoordinates(coords)
```

Arguments

coords	matrix; matrix with coordinates

Value

matrix; coordinates scaled between 0 and 1

Examples

```
matrixR <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
coords <- xyCoordinates(matrixR)
normalizeCoordinates(coords)
```

`reconstructShapeDensity` *Reconstruct polygon from point pattern density*

Description

This function estimates the density of a spatial point pattern (`ppp`), thresholds the density to create a binary image, and then converts it to a valid `sf` object (polygons).

Usage

```
reconstructShapeDensity(
  ppp,
  markSelect = NULL,
  bndw = NULL,
  thres = NULL,
  complement = FALSE,
  dim
)
```

Arguments

`ppp`	point pattern object of class `ppp`
`markSelect`	character; name of mark that is to be selected for the reconstruction
`bndw`	bandwidth of kernel density estimator
`thres`	intensity threshold for the reconstruction
`complement`	logical; reconstruct everything but the mark of interest, default = `FALSE`
`dim`	numeric; x dimension of the final reconstruction.

Value

sf object of class `POLYGON`

Examples

```
data("sostaSPE")
ppp <- SPE2ppp(sostaSPE, marks = "cellType", imageCol = "imageName", imageId = "image1")
thres <- findIntensityThreshold(ppp, markSelect = "A", dim = 500)
struct <- reconstructShapeDensity(ppp, markSelect = "A", thres = thres, dim = 500)
plot(struct)
```

`reconstructShapeDensityImage` *Reconstruct structure from spe object with given image id*

Description

Reconstruct structure from spe object with given image id

Usage

```
reconstructShapeDensityImage(
  spe,
  marks,
  imageCol = NULL,
  imageId = NULL,
  markSelect,
  dim = 500,
  bndw = NULL,
  thres = NULL,
  complement = FALSE
)
```

Arguments

`spe`	SpatialExperiment; a object of class `SpatialExperiment`
`marks`	character; name of column in `colData` that will correspond to the ppp marks
`imageCol`	character; name of a column in `colData` that corresponds to the image
`imageId`	character; image id, must be present in imageCol
`markSelect`	character; name of mark that is to be selected for the reconstruction
`dim`	numeric; x dimension of the final reconstruction. A lower resolution speed up computation but lead to less exact reconstruction. Default = 500
`bndw`	numeric; smoothing bandwidth in the density estimation, corresponds to the `sigma` parameter in the `density.ppp` function, if no value is given the bandwidth is estimated using cross validation with the `bw.diggle` function.
`thres`	numeric; intensity threshold for the reconstruction; if NULL the threshold is set as the mean between the mode of the pixel intensity distributions
`complement`	logical; reconstruct everything but the mark of interest, default = `FALSE`

Value

sf object of class `POLYGON`

Examples

```
data("sostaSPE")
struct <- reconstructShapeDensityImage(sostaSPE,
    marks = "cellType", imageCol = "imageName", imageId = "image1",
    markSelect = "A", dim = 500
)
plot(struct)
```

`reconstructShapeDensitySPE` *Reconstruct structure from spatial experiment object per image id*

Description

Reconstruct structure from spatial experiment object per image id

Usage

```
reconstructShapeDensitySPE(
  spe,
  marks,
  imageCol = NULL,
  markSelect,
  dim = 500,
  bndw = NULL,
  thres = NULL,
  complement = FALSE,
  nCores = 1
)
```

Arguments

`spe`	SpatialExperiment; a object of class `SpatialExperiment`
`marks`	character; name of column in `colData` that will correspond to the ppp marks
`imageCol`	character; name of a column in `colData` that corresponds to the image
`markSelect`	character; name of mark that is to be selected for the reconstruction
`dim`	numeric; x dimension of the final reconstruction. A lower resolution speed up computation but lead to less exact reconstruction. Default = 500
`bndw`	numeric; bandwidth of the sigma parameter in the density estimation, if no value is given the bandwidth is estimated using cross validation with the `bw.diggle` function for each image individually.
`thres`	numeric; intensity threshold for the reconstruction; if NULL the threshold is set as the mean between the mode of the pixel intensity distributions estimated for each image individual
`complement`	logical; reconstruct everything but the mark of interest, default = `FALSE`
`nCores`	numeric; number of cores for parallel processing using `mclapply`. Default = 1

Value

simple feature collection

Examples

```
data("sostaSPE")
allStructs <- reconstructShapeDensitySPE(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    markSelect = "A", bndw = 3.5, thres = 0.005
)
allStructs
```

---

shapeIntensityImage *Intensity plot*

Description

This function plots the intensity of a point pattern image and displays a histogram of the intensity values. Note that intensities less than largest intensity value divided by 250 are not displayed in the histogram.

Usage

```
shapeIntensityImage(
  spe,
  marks,
  imageCol = NULL,
  imageId = NULL,
  markSelect,
  bndw = NULL,
  dim = 500
)
```

Arguments

spe	SpatialExperiment; a object of class SpatialExperiment
marks	character; name of column in colData that will correspond to the ppp marks
imageCol	character; name of a column in colData that corresponds to the image
imageId	character; image id, must be present in imageCol
markSelect	character; name of mark that is to be selected for the reconstruction
bndw	numeric; smoothing bandwidth in the density estimation, corresponds to the sigma parameter in the density.ppp function, if no value is given the bandwidth is estimated using cross validation with the bw.diggle function.
dim	numeric; x dimension of the final reconstruction. A lower resolution speeds up computation but lead to less exact reconstruction. Default = 500

Value

ggplot object with intensity image and histogram

Examples

```
data("sostaSPE")
shapeIntensityImage(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    imageId = "image1", markSelect = "A"
)
```

`shapeMetrics` *Calculate a set of shape metrics of a single polygon*

Description

Calculate a set of shape metrics of a single polygon

Usage

```
shapeMetrics(sfPoly)
```

Arguments

`sfPoly`	POLYGON of class sfc

Details

For multiple polyogns or a MULTIPOLYGON object use the function `totalShapeMetrics`

Value

list; list of shape metrics

Examples

```
matrix_R <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
polyR <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
shapeMetrics(polyR)
```

simulateTissueBlobs *Simulate Tissue Blobs*

Description

This function generates a simulated tissue-like structure using a Gaussian blur technique.

Usage

```
simulateTissueBlobs(size, seedNumber, clumpSize)
```

Arguments

size	Integer; The size (width and height) of the simulated tissue image.
seedNumber	Integer; The number of random seed points used to generate tissue blobs.
clumpSize	Numeric; The standard deviation (sigma) of the Gaussian blur applied to generate tissue clumps.

Value

A binary matrix representing the simulated tissue structure.

Examples

```
tissueImage <- simulateTissueBlobs(128, 100, 7)
image(tissueImage)
```

sostaSPE *Example SpatialExperiment Object with Simulated Tissue Images and Point Patterns*

Description

This dataset contains a simulated SpatialExperiment object (sostaSPE) representing three tissue images, each with a corresponding spatial point pattern. The point patterns contain different cell types (A, B, and C), distributed according to simulated tissue structures.

Usage

```
sostaSPE
```

Format

A `SpatialExperiment` object with the following structure:

x Numeric; x-coordinate of each point (cell location).

y Numeric; y-coordinate of each point (cell location).

cell_type Factor; Cell type assigned to each point (A, B, or C).

image_name Factor; Identifier for the tissue image (`image1`, `image2`, or `image3`).

Details

The dataset was generated as follows:

- Three tissue images were simulated using `simulateTissueBlobs()`.
- Spatial point patterns were created for each tissue using `createPointPatternTissue()`.
- The point pattern data was converted into a `SpatialExperiment` object with spatial coordinates.

---

spatialCoords2SF *Function to convert spatialCoords to an sf object*

Description

Function to convert spatialCoords to an sf object

Usage

```
spatialCoords2SF(spe)
```

Arguments

`spe`	SpatialExperiment; a object of class `SpatialExperiment`

Value

sf; Simple feature collection of geometry type POINT

Examples

```
data(sostaSPE)
speSel <- sostaSPE[, sostaSPE[["imageName"]] == "image1"]
spatialCoords2SF(speSel)
```

SPE2ppp *Function to convert spatial coordinates of a* SpatialExperiment *object to a* ppp *object*

Description

Function to convert spatial coordinates of a SpatialExperiment object to a ppp object

Usage

```
SPE2ppp(spe, marks = NULL, imageCol = NULL, imageId = NULL)
```

Arguments

spe	SpatialExperiment; a object of class SpatialExperiment
marks	character; name of column in colData that will correspond to the ppp marks
imageCol	character; name of a column in colData that corresponds to the image
imageId	character; image id, must be present in imageCol

Value

ppp; object of type ppp

Examples

```
data(sostaSPE)
SPE2ppp(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    imageId = "image1"
)
```

stCalculateCurvature *Calculate curvature of sf object*

Description

Calculate curvature of sf object

Usage

```
stCalculateCurvature(sfPoly, smoothness = 5)
```

Arguments

sfPoly	POLYGON of class sf
smoothness	list; curvature measures

Value

list; list of curvatures values

References

https://stackoverflow.com/questions/62250151/calculate-curvature-of-a-closed-object-in-r

Examples

```
matrixR <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
polyR <- binaryImageToSF(matrixR, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
stCalculateCurvature(polyR)
```

---

stCalculateShapeCurl *Calculate curl of a polygon*

Description

Calculate curl of a polygon

Usage

```
stCalculateShapeCurl(sfPoly)
```

Arguments

sfPoly	POLYGON of class sf

Value

numeric; the curl of the polygon

Examples

```
matrixR <- matrix(c(
    1, 1, 1, 1, 1, 0,
    1, 1, 0, 0, 1, 1,
    1, 1, 0, 0, 1, 1,
    1, 1, 1, 1, 1, 0,
    1, 1, 0, 1, 1, 0,
    1, 1, 0, 0, 1, 1,
    1, 1, 0, 0, 1, 1
), nrow = 7, byrow = TRUE)
polyR <- binaryImageToSF(matrixR, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
stCalculateShapeCurl(polyR)
```

---

stFeatureAxes *Calculate the length of feature axes of a single sf polygon*

Description

Calculate the length of feature axes of a single sf polygon

Usage

```
stFeatureAxes(sfPoly)
```

Arguments

sfPoly	POLYGON of class sf

Value

list; list containing the major and minor axis lengths

Examples

```
matrixR <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
polyR <- binaryImageToSF(matrixR, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
stFeatureAxes(polyR)
```

totalShapeMetrics *Calculate a set of shape metrics of a set of polygons*

Description

Calculate a set of shape metrics of a set of polygons

Usage

```
totalShapeMetrics(sfInput)
```

Arguments

sfInput	MULTIPOLYGON of class sf

Details

Calculate a set of shape metrics of a set of polygons. The function calculates all metrics that are implemented in the function shapeMetrics

Value

matrix; matrix of shape metrics

Examples

```
data(sostaSPE)
struct <- reconstructShapeDensityImage(sostaSPE,
    marks = "cellType", imageCol = "imageName",
    imageId = "image1", markSelect = "A", dim = 500
)
totalShapeMetrics(struct)
```

xyCoordinates *Function to extract x y coordinates from binary image*

Description

Function to extract x y coordinates from binary image

Usage

```
xyCoordinates(inputMatrix)
```

Arguments

inputMatrix	a binary matrix

Value

matrix; matrix with x,y coordinates of the cell of the input matrix

Examples

```
matrixR <- matrix(c(
    0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0
), nrow = 9, byrow = TRUE)
xyCoordinates(matrixR)
```

Index

∗ **datasets**
    sostaSPE, 21
∗ **internal**
    sosta-package, 3
.SPE2df, *4*, 5
.df2ppp, 3
.intensityImage, 4
.intensityThreshold, 5

as.ppp, *4*
assingCellsToStructures, 6

binaryImageToSF, 7

cellTypeProportions, 8
createPointPatternTissue, 9

estimateReconstructionParametersSPE, 10

findIntensityThreshold, 11

getDimXY, 11

meanShapeMetrics, 12
minBoundaryDistances, 13
minCellTypeStructDist, 14

normalizeCoordinates, 15

reconstructShapeDensity, 16
reconstructShapeDensityImage, 17
reconstructShapeDensitySPE, 18

shapeIntensityImage, 19
shapeMetrics, 20, *26*
simulateTissueBlobs, 21
sosta *(*sosta-package*)*, 3
sosta-package, 3
sostaSPE, 21
spatialCoords2SF, 22
SPE2ppp, 23
stCalculateCurvature, 23
stCalculateShapeCurl, 24
stFeatureAxes, 25

totalShapeMetrics, *20*, 26

xyCoordinates, 26