

# Package ‘sosta’

February 26, 2025

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

**Version** 0.99.4

**Description** `sosta` (Spatial Omics SStructure Analysis) is a package for analyzing spatial omics data to explore tissue organization at the anatomical structure level. It reconstructs morphologically relevant structures based on molecular features or cell types. It further calculates a range of structural and shape metrics 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** 7.3.2

**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, dplyr, ggplot2, patchwork, SummarizedExperiment, stats, rlang, parallel

**Suggests** knitr, rmarkdown, ggspavis, BiocStyle, imcdatasets, ggfortify, tidyr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/sosta>

**git\_branch** devel

**git\_last\_commit** e5f5f7c

**git\_last\_commit\_date** 2025-02-08

**Repository** Bioconductor 3.21

**Date/Publication** 2025-02-25

**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 . . . . .	2
.intensityImage . . . . .	3
.intensityThreshold . . . . .	3
binaryImageToSF . . . . .	4
estimateReconstructionParametersSPE . . . . .	5
findIntensityThreshold . . . . .	6
getDimXY . . . . .	7
meanShapeMetrics . . . . .	7
normalizeCoordinates . . . . .	8
reconstructShapeDensity . . . . .	9
reconstructShapeDensityImage . . . . .	10
reconstructShapeDensitySPE . . . . .	11
shapeIntensityImage . . . . .	12
shapeMetrics . . . . .	13
SPE2ppp . . . . .	14
st_calculateCurvature . . . . .	14
st_calculateShapeCurl . . . . .	15
st_feature_axes . . . . .	16
totalShapeMetrics . . . . .	17
xyCoordinates . . . . .	17

**Index** **19**

---

sosta-package	<i>sosta: A package for the analysis of anatomical tissue structures in spatial omics data</i>
---------------	--

---

## 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 morphologically relevant structures based on molecular features or cell types. It further calculates a range of structural and shape metrics 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@m1s.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>

---

.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, mark_select = NULL, bndw = NULL, dim)
```

### Arguments

ppp	point pattern object of class ppp
mark_select	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(density_image, steps = 250)
```

**Arguments**

density\_image 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.

**Value**

numeric; estimated threshold

---

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**

```
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)
poly_R <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
plot(poly_R)
```

---

 estimateReconstructionParametersSPE

*Estimate reconstruction parameters from a set of images*


---

## Description

Estimate reconstruction parameters from a set of images

## Usage

```
estimateReconstructionParametersSPE(
  spe,
  marks,
  image_col,
  mark_select = NULL,
  nimages = NULL,
  fun = "bw.diggle",
  dim = 500,
  ncores = 1,
  plot_hist = TRUE
)
```

## Arguments

spe	SpatialExperiment; a object of class SpatialExperiment
marks	character; name of column in colData that will correspond to the ppp marks
image_col	character; name of a column in colData that corresponds to the image
mark_select	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
plot_hist	logical; if histogram of estimated densities and thresholds should be plotted. Default = TRUE

## Value

tibble; tibble with estimated intensities and thresholds

**Examples**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
spe_sel <- spe[, spe[["image_name"]] %in% c("E02", "E03", "E04")]
estimateReconstructionParametersSPE(spe_sel,
  marks = "cell_category",
  image_col = "image_name", mark_select = "islet", plot_hist = 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, mark_select = NULL, bndw = NULL, dim, steps = 250)
```

**Arguments**

ppp	point pattern object of class ppp
mark_select	character; name of mark that is to be selected for the reconstruction
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 <code>bw.diggle</code> 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**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
ppp <- SPE2ppp(spe, marks = "cell_category", image_col = "image_name", image_id = "E04")
findIntensityThreshold(ppp, mark_select = "islet", 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**

<code>ppp</code>	point pattern object of class ppp
<code>ydim</code>	dimension of y axis

**Value**

vector; vector with x and y dimension

**Examples**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
ppp <- SPE2ppp(spe,
  marks = "cell_category", image_col = "image_name",
  image_id = "E04"
)
getDimXY(ppp, 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**

<code>totalShapeMetricMatrix</code>	matrix of shape metrics
-------------------------------------	-------------------------

**Value**

matrix; matrix of mean shape metrics

**Examples**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
islet_poly <- reconstructShapeDensityImage(spe,
  marks = "cell_category",
  image_col = "image_name", image_id = "E04", mark_select = "islet", dim = 500
)
shape_metrics <- totalShapeMetrics(islet_poly)
meanShapeMetrics(shape_metrics)
```

---

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**

```
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)
coords <- xyCoordinates(matrix_R)
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,  
  mark_select = NULL,  
  bndw = NULL,  
  thres = NULL,  
  dim  
)
```

### Arguments

<code>ppp</code>	point pattern object of class ppp
<code>mark_select</code>	character; name of mark that is to be selected for the reconstruction
<code>bndw</code>	bandwidth of kernel density estimator
<code>thres</code>	intensity threshold for the reconstruction
<code>dim</code>	numeric; x dimension of the final reconstruction.

### Value

sf object of class POLYGON

### Examples

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)  
ppp <- SPE2ppp(spe, marks = "cell_category", image_col = "image_name", image_id = "E04")  
thres <- findIntensityThreshold(ppp, mark_select = "islet", dim = 500)  
islet_poly <- reconstructShapeDensity(ppp, mark_select = "islet", thres = thres, dim = 500)  
plot(islet_poly)
```

---

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

---

**Description**

Reconstruct structure from spe object with given image id

**Usage**

```
reconstructShapeDensityImage(  
  spe,  
  marks,  
  image_col,  
  image_id,  
  mark_select,  
  dim = 500,  
  bndw = NULL,  
  thres = NULL  
)
```

**Arguments**

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

**Value**

sf object of class POLYGON

**Examples**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)  
islet_poly <- reconstructShapeDensityImage(spe,  
  marks = "cell_category",  
  image_col = "image_name", image_id = "E04", mark_select = "islet", dim = 500  
)  
plot(islet_poly)
```

---

reconstructShapeDensitySPE

*Reconstruct structure from spatial experiment object per image id*


---

## Description

Reconstruct structure from spatial experiment object per image id

## Usage

```
reconstructShapeDensitySPE(
  spe,
  marks,
  image_col,
  mark_select,
  dim = 500,
  bndw = NULL,
  thres,
  ncores = 1
)
```

## Arguments

spe	SpatialExperiment; a object of class SpatialExperiment
marks	character; name of column in colData that will correspond to the ppp marks
image_col	character; name of a column in colData that corresponds to the image
mark_select	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.
thres	numeric; intensity threshold for the reconstruction
ncores	numeric; number of cores for parallel processing using mclapply. Default = 1

## Value

simple feature collection

## Examples

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
spe_sel <- spe[, spe[["image_name"]] %in% c("E02", "E03", "E04")]
all_islets <- reconstructShapeDensitySPE(spe_sel,
  marks = "cell_category",
```

```

    image_col = "image_name", mark_select = "islet", bndw = sigma, thres = 0.0025
  )
  all_islets

```

---

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,
  image_col,
  image_id,
  mark_select,
  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
image_col	character; name of a column in colData that corresponds to the image
image_id	character; image id, must be present in image_col
mark_select	character; name of mark that is to be selected for the reconstruction
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.
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**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
shapeIntensityImage(spe,
  marks = "cell_category", image_col = "image_name",
  image_id = "E04", mark_select = "islet"
)
```

---

shapeMetrics

*Calculate a set of shape metrics of a polygon*


---

**Description**

Calculate a set of shape metrics of a polygon

**Usage**

```
shapeMetrics(sfPoly)
```

**Arguments**

```
sfPoly          POLYGON of class sfc
```

**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)
poly_R <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
shapeMetrics(poly_R)
```

---

SPE2ppp	<i>Function to convert spatial coordinates of a SpatialExperiment object to a ppp object</i>
---------	--

---

**Description**

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

**Usage**

```
SPE2ppp(spe, marks, image_col = NULL, image_id = NULL)
```

**Arguments**

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

**Value**

ppp; object of type ppp

**Examples**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
SPE2ppp(spe, marks = "cell_category", image_col = "image_name", image_id = "E04")
```

---

st\_calculateCurvature *Title*

---

**Description**

Title

**Usage**

```
st_calculateCurvature(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**

```
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)
poly_R <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
st_calculateCurvature(poly_R)
```

---

st\_calculateShapeCurl *Calculate curl of a polygon*

---

**Description**

Calculate curl of a polygon

**Usage**

```
st_calculateShapeCurl(sfPoly)
```

**Arguments**

sfPoly            POLYGON of class sf

**Value**

numeric; the curl of the polygon

**Examples**

```

matrix_R <- 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)
poly_R <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
st_calculateShapeCurl(poly_R)

```

---

st_feature_axes	<i>Calculate the length of feature axes of an sf polygon</i>
-----------------	--

---

**Description**

Calculate the length of feature axes of an sf polygon

**Usage**

```
st_feature_axes(sfPoly)
```

**Arguments**

sfPoly            POLYGON of class sf

**Value**

list; list containing the major and minor axis lengths

**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)
poly_R <- binaryImageToSF(matrix_R, xmin = 0, xmax = 1, ymin = 0, ymax = 1)
st_feature_axes(poly_R)

```



---

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**

```
spe <- imcdatasets::Damond_2019_Pancreas("spe", full_dataset = FALSE)
islet_poly <- reconstructShapeDensityImage(spe,
  marks = "cell_category",
  image_col = "image_name", image_id = "E04", mark_select = "islet", dim = 500
)
totalShapeMetrics(islet_poly)
```

---

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**

```
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)
xyCoordinates(matrix_R)
```

# Index

## \* **internal**

- sosta-package, [2](#)
- .intensityImage, [3](#)
- .intensityThreshold, [3](#)
  
- binaryImageToSF, [4](#)
  
- estimateReconstructionParametersSPE, [5](#)
  
- findIntensityThreshold, [6](#)
  
- getDimXY, [7](#)
  
- meanShapeMetrics, [7](#)
  
- normalizeCoordinates, [8](#)
  
- reconstructShapeDensity, [9](#)
- reconstructShapeDensityImage, [10](#)
- reconstructShapeDensitySPE, [11](#)
  
- shapeIntensityImage, [12](#)
- shapeMetrics, [13](#)
- sosta (sosta-package), [2](#)
- sosta-package, [2](#)
- SPE2ppp, [14](#)
- st\_calculateCurvature, [14](#)
- st\_calculateShapeCurl, [15](#)
- st\_feature\_axes, [16](#)
  
- totalShapeMetrics, [17](#)
  
- xyCoordinates, [17](#)