

# Package ‘SpatialFeatureExperiment’

October 14, 2024

**Type** Package

**Title** Integrating SpatialExperiment with Simple Features in sf

**Version** 1.6.1

**Description** A new S4 class integrating Simple Features with the R package sf to bring geospatial data analysis methods based on vector data to spatial transcriptomics. Also implements management of spatial neighborhood graphs and geometric operations. This package builds upon SpatialExperiment and SingleCellExperiment, hence methods for these parent classes can still be used.

**Imports** Biobase, BiocGenerics, BiocNeighbors, BiocParallel, data.table, DropletUtils, EBImage, grDevices, lifecycle, Matrix, methods, rjson, rlang, S4Vectors, sf, sfheaders, SingleCellExperiment, SpatialExperiment, spdep (>= 1.1-7), SummarizedExperiment, stats, terra, utils, zeallot

**License** Artistic-2.0

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Collate** 'AllGenerics.R' 'utils.R' 'SFE-class.R' 'annotGeometries.R' 'cbind.R' 'coerce.R' 'data.R' 'df2sf.R' 'dimGeometries.R' 'featureData.R' 'formatTxSpots.R' 'geometry\_operation.R' 'graph\_wrappers.R' 'image.R' 'int\_dimData.R' 'internal-Voyager.R' 'localResults.R' 'read.R' 'reexports.R' 'saveRDS.R' 'spatialGraphs.R' 'subset.R' 'transformation.R' 'updateObject.R' 'validity.R'

**Suggests** arrow, BiocStyle, knitr, RBioFormats, rhdf5, rmarkdown, sfarrow, SFEData (>= 1.5.3), testthat (>= 3.0.0), Voyager, xml2, scater, Seurat, SeuratObject, dplyr, tidyr

**Remotes** Voyager=github::pachterlab/voyager

**Config/testthat/edition** 3

**Depends** R (>= 4.2.0)

**VignetteBuilder** knitr

**biocViews** DataRepresentation, Transcriptomics, Spatial

**URL** <https://github.com/pachterlab/SpatialFeatureExperiment>

**BugReports** <https://github.com/pachterlab/SpatialFeatureExperiment/issues>

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

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** b36d63b

**git\_last\_commit\_date** 2024-08-22

**Repository** Bioconductor 3.19

**Date/Publication** 2024-10-13

**Author** Lambda Moses [aut, cre] (<<https://orcid.org/0000-0002-7092-9427>>),  
 Alik Huseynov [aut] (<<https://orcid.org/0000-0002-1438-4389>>),  
 Lior Pachter [aut, ths] (<<https://orcid.org/0000-0002-9164-6231>>)

**Maintainer** Lambda Moses <dl3764@columbia.edu>

## Contents

addVisiumSpotPoly . . . . .	3
affineImg . . . . .	4
aggBboxes . . . . .	5
annotGeometries . . . . .	6
annotOp . . . . .	8
annotPred . . . . .	9
annotSummary . . . . .	10
bbox,SpatialFeatureExperiment-method . . . . .	11
bbox_center . . . . .	12
BioFormatsImage . . . . .	13
BioFormatsImage-getters . . . . .	14
cbind,SpatialFeatureExperiment-method . . . . .	15
changeSampleIDs . . . . .	16
colFeatureData . . . . .	16
colGeometries . . . . .	17
crop . . . . .	19
cropImg . . . . .	21
df2sf . . . . .	22
dim,BioFormatsImage-method . . . . .	24
dimGeometries . . . . .	24
ext . . . . .	27
ExtImage . . . . .	28
findSpatialNeighbors,SpatialFeatureExperiment-method . . . . .	29
findVisiumGraph . . . . .	31
formatTxSpots . . . . .	32
formatTxTech . . . . .	35
gdalParquetAvailable . . . . .	37
getParams . . . . .	38
getPixelSize . . . . .	39

imageIDs . . . . .	39
Img<-,SpatialExperiment-method . . . . .	40
imgRaster . . . . .	41
imgSource . . . . .	42
internal-Voyager . . . . .	42
localResults . . . . .	43
mirrorImg . . . . .	47
read10xVisiumSFE . . . . .	48
readCosMX . . . . .	49
readSelectTx . . . . .	51
readVizgen . . . . .	53
readXenium . . . . .	55
reexports . . . . .	57
removeEmptySpace . . . . .	58
rotateImg . . . . .	59
rowGeometries . . . . .	60
sampleIDs . . . . .	62
saveRDS,SpatialFeatureExperiment-method . . . . .	62
scaleImg . . . . .	63
SFE-image . . . . .	64
SFE-transform . . . . .	67
show,SpatialFeatureExperiment-method . . . . .	68
SpatialFeatureExperiment . . . . .	69
SpatialFeatureExperiment-class . . . . .	72
SpatialFeatureExperiment-coercion . . . . .	72
SpatialFeatureExperiment-subset . . . . .	75
spatialGraphs . . . . .	76
SpatRasterImage . . . . .	79
st_any_pred . . . . .	80
toExtImage . . . . .	81
toSpatRasterImage . . . . .	82
translateImg . . . . .	83
transposeImg . . . . .	84
unit,SpatialFeatureExperiment-method . . . . .	85
updateObject . . . . .	85
visium_row_col . . . . .	86
<b>Index</b>	<b>87</b>

---

addVisiumSpotPoly	<i>Add Visium spot polygons to colGeometry</i>
-------------------	--

---

## Description

For adding the spot polygons to SFE objects converted from SPE.

**Usage**

```
addVisiumSpotPoly(x, spotDiameter)
```

**Arguments**

x	A SpatialFeatureExperiment object.
spotDiameter	Spot diameter for technologies with arrays of spots of fixed diameter per slide, such as Visium, ST, DBiT-seq, and slide-seq. The diameter must be in the same unit as the coordinates in the *Geometry arguments. Ignored for geometries that are not POINT or MULTIPOINT.

**Value**

A SFE object with a new colGeometry called spotPoly, which has polygons of the spots.

**Examples**

```
library(SpatialExperiment)
example(read10xVisium)
# There can't be suplicate barcodes
colnames(spe) <- make.unique(colnames(spe), sep = "-")
rownames(spatialCoords(spe)) <- colnames(spe)
sfe <- toSpatialFeatureExperiment(spe)
# A hypothetical spot diameter; check the scalefactors_json.json file for
# actual diameter in pixels in full resolution image.
sfe <- addVisiumSpotPoly(sfe, spotDiameter = 80)
```

---

 affineImg

*Affine transformation of images*


---

**Description**

This function performs affine transformation on images, with any matrix and translation vector.

**Usage**

```
## S4 method for signature 'SpatRasterImage'
affineImg(x, M, v, maxcell = 1e+07, ...)

## S4 method for signature 'BioFormatsImage'
affineImg(x, M, v, ...)

## S4 method for signature 'ExtImage'
affineImg(x, M, v, ...)
```

**Arguments**

x	An object of class *Image as implemented in this package.
M	A 2x2 numeric matrix for the linear transformation in the xy plane.
v	A numeric vector of length 2 for translation in the xy plane.
maxcell	Max number of pixels to load SpatRasterImage into memory. The default 1e7 is chosen because this is the approximate number of pixels in the medium resolution image at resolution = 4L in Xenium OME-TIFF to make different methods of this function consistent.
...	Ignored. It's there so different methods can all be passed to the same lapply in the method for SFE objects. Some methods have extra arguments.

**Value**

SpatRasterImage will be converted to ExtImage. Otherwise \*Image object of the same class. For BioFormatsImage, the transformation info is stored and will be applied when the image is loaded into memory as ExtImage.

**See Also**

Other image methods: [SFE-image](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

aggBboxes

*Aggregate bounding boxes*


---

**Description**

To find the bounding box of multiple bounding boxes.

**Usage**

```
aggBboxes(bboxes)
```

**Arguments**

bboxes	Either a matrix with 4 rows whose columns are the different bounding boxes, with row names "xmin", "xmax", "ymin", and "ymax" in any order, or a list of bounding boxes which are named numeric vectors.
--------	--

**Value**

A named numeric vector for the total bounding box.

**Examples**

```
bboxes <- list(c(xmin = 5, xmax = 10, ymin = 2, ymax = 20),
c(xmin = 8, xmax = 18, ymin = 0, ymax = 15))
bbox_all <- aggBboxes(bboxes)
```

---

annotGeometries      *Annotation geometry methods*

---

## Description

"Annotation geometry" refers to Simple Feature (sf) geometries NOT associated with rows (features, genes) or columns (cells or spots) of the gene count matrix in the `SpatialFeatureExperiment` object. So there can be any number of rows in the sf data frame specifying the geometry. Examples of such geometries are tissue boundaries, pathologist annotation of histological regions, and objects not characterized by columns of the gene count matrix (e.g. nuclei segmentation in a Visium dataset where the columns are Visium spots). This page documents getters and setters for the annotation geometries. Internally, annotation geometries are stored in `int_metadata`.

## Usage

```
## S4 method for signature 'SpatialFeatureExperiment'
annotGeometries(x)

## S4 replacement method for signature 'SpatialFeatureExperiment'
annotGeometries(x, translate = TRUE, ...) <- value

## S4 method for signature 'SpatialFeatureExperiment'
annotGeometryNames(x)

## S4 replacement method for signature 'SpatialFeatureExperiment,character'
annotGeometryNames(x) <- value

## S4 method for signature 'SpatialFeatureExperiment'
annotGeometry(x, type = 1L, sample_id = NULL)

## S4 replacement method for signature 'SpatialFeatureExperiment'
annotGeometry(x, type = 1L, sample_id = NULL, translate = TRUE, ...) <- value

tissueBoundary(x, sample_id = 1L)

tissueBoundary(x, sample_id = 1L, translate = TRUE, ...) <- value
```

## Arguments

<code>x</code>	A <code>SpatialFeatureExperiment</code> object.
<code>translate</code>	Logical. Only used if <code>removeEmptySpace</code> has been run of the SFE object. If that's the case, this argument indicates whether the new value to be assigned to the geometry is in the coordinates prior to removal of empty space so it should be translated to match the new coordinates after removing empty space. Default to TRUE.

...	spatialCoordsNames, spotDiameter, geometryType passed to <code>df2sf</code> . Defaults are the same as in <code>df2sf</code> . For <code>dimGeometries&lt;-</code> only: <code>geometryType</code> can be a character vector of the geometry type of each data frame in the list of the same length as the list if the data frames specify different types of geometries.
value	Value to set. For <code>annotGeometry</code> , must be a <code>sf</code> data frame, or an ordinary data frame that can be converted to a <code>sf</code> data frame (see <code>df2sf</code> ). For <code>annotGeometries</code> , must be a list of such <code>sf</code> or ordinary data frames. There must be a column <code>sample_id</code> to indicate the sample the geometries are for, and the <code>sample_id</code> must also appear in <code>colData</code> .
type	An integer specifying the index or string specifying the name of the <code>*Geometry</code> to query or replace. If missing, then the first item in the <code>*Geometries</code> will be returned or replaced.
sample_id	Sample ID to get or set geometries.

### Details

Wrapper for getter and setter of special geometry:

**tisseuBoundary** Boundary of the tissue of interest, including holes. This is usually of geometry type MULTIPOLYGON, though geometries in `annotGeometries` can have any type supported by `sf`.

### Value

Getters for multiple geometries return a named list. Getters for names return a character vector of the names. Getters for single geometries return an `sf` data frame. Setters return an `SFE` object.

### Examples

```
# Example dataset
library(SFEData)
sfe_small <- McKellarMuscleData(dataset = "small")

# Get all annotation geometries, returning a named list
annotGeometries(sfe_small)

# Set all annotation geometries, in a named list
toy <- readRDS(system.file("extdata/sfe_toy.rds",
  package = "SpatialFeatureExperiment"
))
ag <- readRDS(system.file("extdata/ag.rds",
  package = "SpatialFeatureExperiment"
))
annotGeometries(toy) <- list(hull = ag)

# Get names of annotation geometries
annotGeometryNames(sfe_small)

# Set names of annotation geometries
```

```

annotGeometryNames(toy) <- "foo"

# Get a specific annotation geometry by name
# sample_id is optional when there is only one sample present
nuclei <- annotGeometry(sfe_small, type = "nuclei", sample_id = "Vis5A")

# Get a specific annotation geometry by index
tb <- annotGeometry(sfe_small, type = 1L)

# Set a specific annotation geometry
annotGeometry(sfe_small, type = "nuclei2") <- nuclei

# Special convenience function for tissue boundaries
# Getter
tb <- tissueBoundary(sfe_small, sample_id = "Vis5A")
# Setter
tissueBoundary(sfe_small, sample_id = "Vis5A") <- tb

```

---

annotOp

*Binary operations for geometry of each cell/spot and annotation*


---

## Description

Just like [annotPred](#), but performs the operation rather than predicate. For example, this function would return the geometry of the intersections between each Visium spot and the tissue boundary for each sample, rather than whether each Visium spot intersects the tissue boundary. In case one cell/spot gets broken up into multiple geometries, the union of those geometries will be taken, so each cell/spot will only get one geometry.

## Usage

```

annotOp(
  sfe,
  colGeometryName = 1L,
  annotGeometryName = 1L,
  sample_id = "all",
  op = st_intersection
)

```

## Arguments

sfe	An SFE object.
colGeometryName	Name of column geometry for the predicate.
annotGeometryName	Name of annotation geometry for the predicate.
sample_id	Which sample(s) to operate on. Can be "all" to indicate all samples.
op	A binary operation function for the geometries. Defaults to <a href="#">st_intersection</a> .



**Value**

A sf data frame with geometry column containing the geometries and corresponding column names of sfe as row names. There is no guarantee that the returned geometries are valid or preserve the geometry class (e.g. when the intersection of polygons result into a line or a point).

**See Also**

annotPred

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData("small")
# Get the intersection of myofibers with each Visium spot
myofibers_on_spots <- annotOp(sfe, "spotPoly",
  annotGeometryName = "myofiber_simplified"
)
```

---

annotPred

*Binary predicates for geometry of each cell/spot and annotation*

---

**Description**

This function finds binary predicates for the geometry of each cell/spot (i.e. colGeometry) and an annotation geometry for each sample. For example, whether each Visium spot intersects with the tissue boundary in each sample.

**Usage**

```
annotPred(
  sfe,
  colGeometryName = 1L,
  annotGeometryName = 1L,
  sample_id = "all",
  pred = st_intersects
)

annotNPred(
  sfe,
  colGeometryName = 1L,
  annotGeometryName = 1L,
  sample_id = "all",
  pred = st_intersects
)
```

**Arguments**

sfe	An SFE object.
colGeometryName	Name of column geometry for the predicate.
annotGeometryName	Name of annotation geometry for the predicate.
sample_id	Which sample(s) to operate on. Can be "all" to indicate all samples.
pred	Predicate function to use, defaults to <code>st_intersects</code> .

**Value**

For `annotPred`, a logical vector of the same length as the number of columns in the sample(s) of interest, with barcodes (or corresponding column names of `sfe`) as names. For `annotNPred`, a numeric vector of the same length as the number of columns in the sample(s) of interest with barcodes as names, indicating the number of geometries in the `annotGeometry` of interest returns TRUE for the predicate for each each geometry in the `colGeometry` of interest.

**See Also**

`annotOp`

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData("small")
# Whether each spot is in tissue
in_tissue <- annotPred(sfe, "spotPoly", annotGeometryName = "tissueBoundary")
# How many nuclei are there in each Visium spot
n_nuclei <- annotNPred(sfe, "spotPoly", annotGeometryName = "nuclei")
```

---

annotSummary

*Summarize attributes of an annotGeometry for each cell/spot*

---

**Description**

In SFE objects, the annotation geometries don't have to correspond to the dimensions of the gene count matrix, so there generally is no one to one mapping between annotation geometries and cells/spots. However, it may be interesting to relate attributes of annotation geometries to cell/spots so the attributes can be related to gene expression. This function summarizes attributes of an `annotGeometry` for each cell/spot by a geometric predicate with a `colGeometry`.

**Usage**

```

annotSummary(
  sfe,
  colGeometryName = 1L,
  annotGeometryName = 1L,
  annotColNames = 1L,
  sample_id = "all",
  pred = st_intersects,
  summary_fun = mean
)

```

**Arguments**

sfe	An SFE object.
colGeometryName	Name of column geometry for the predicate.
annotGeometryName	Name of annotation geometry for the predicate.
annotColNames	Character, column names of the annotGeometry of interest, to indicate the columns to summarize. Columns that are absent from the annotGeometry are removed. The column cannot be "geometry" or "barcode".
sample_id	Which sample(s) to operate on. Can be "all" to indicate all samples.
pred	Predicate function to use, defaults to <code>st_intersects</code> .
summary_fun	Function for the summary, defaults to <code>mean</code> .

**Value**

A data frame whose row names are the relevant column names of `sfe`, and each column of which is the summary of each column specified in `annotColName`.

**Examples**

```

library(SFEData)
sfe <- McKellarMuscleData("small")
s <- annotSummary(sfe, "spotPoly", "myofiber_simplified",
  annotColNames = c("area", "convexity")
)

```

---

bbox,SpatialFeatureExperiment-method

*Find bounding box of SFE objects*


---

**Description**

Find bounding box of the union of all `colGeometries` and `annotGeometries` of each sample in the SFE object. This can be used to remove empty space so the tissue and geometries have one corner at the origin so all samples will be on comparable coordinates.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
bbox(sfe, sample_id = "all", include_images = FALSE, include_row = TRUE)
```

**Arguments**

sfe	A SpatialFeatureExperiment object.
sample_id	Sample(s) whose bounding box(es) to find. The bounding box would be for the union of all colGeometries and annotGeometries associated with each sample.
include_images	Logical, whether the bounding boxes should include image extents. Defaults to FALSE because often the image has a lot of empty space surrounding the tissue.
include_row	Logical, whether the bounding boxes should include rowGeometries, defaults to TRUE.

**Value**

For one sample, then a named vector with names xmin, ymin, xmax, and ymax specifying the bounding box. For multiple samples, then a matrix whose columns are samples and whose rows delineate the bounding box.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData("small")
bbox(sfe, sample_id = "Vis5A")
```

---

bbox_center	<i>Find center of bounding box</i>
-------------	------------------------------------

---

**Description**

Get x-y coordinates of the center of any bounding box

**Usage**

```
bbox_center(bbox)
```

**Arguments**

bbox	A numeric vector of length 4 with names xmin, xmax, ymin, ymax, in any order.
------	---

**Value**

A numeric vector of length 2.

**Examples**

```
bbox <- c(xmin = 0, xmax = 100, ymin = 0, ymax = 80)
bbox_center(bbox)
```

---

BioFormatsImage

*On disk representation of BioFormats images in SFE object*


---

**Description**

`'r lifecycle::badge("experimental")'` At present, the BioFormatsImage is designed for OME-TIFF from Xenium and has not been tested on other formats that can be read with BioFormats. The image is not loaded into memory, and when it is, the BioFormatsImage object is converted into ExtImage because the loaded image is of a class that inherits from Image. The ExtImage class is a thin wrapper inheriting from VirtualSpatialImage so it's compatible with SpatialExperiment from which SFE is derived. This class might drastically change as it matures, say to accommodate other formats supported by BioFormats and to store the transformation matrix rather than loading image into memory upon transform.

**Usage**

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

BioFormatsImage(
  path,
  ext = NULL,
  isFull = TRUE,
  origin = c(0, 0),
  transformation = list()
)
```

**Arguments**

object	A BioFormatsImage object.
path	Path to an OME-TIFF image file.
ext	Numeric vector with names "xmin", "xmax", "ymin", "ymax" in microns indicating the spatial extent covered by the image. If NULL, then the extent will be inferred from the metadata, from physical pixel size and the number of pixels.
isFull	Logical, if the extent specified in ext is the full extent. If ext = NULL so it will be inferred from metadata then isFull = TRUE will be set internally.
origin	Origin of the whole image in the x-y plane, defaults to c(0, 0). This is shifted when the image is translated. This is not the same as xmin and xmax. For example, when the extent is only part of the whole image and the whole image itself can be spatially translated, the origin is needed to determine which part of the whole image this extent corresponds to.

`transformation` Named list specifying affine transformation. The list can have names "name" and named parameter of the transformation, e.g. `list(name = "mirror", direction = "vertical")`, "rotate" and `degrees = 90` (clockwise), and "scale" and `factor = 2`. The list can also have names "M" for a 2x2 linear transformation matrix in the xy plane and "v" for a translation vector of length 2 to specify general affine transformation.

### Details

Spatial extent is inferred from OME-TIFF metadata if not specified. Physical pixel size from the metadata is used to make the extent in micron space. If physical pixel size is absent from metadata, then the extent will be in pixel space, which might mean that the image will not align with the geometries because often the geometry coordinates are in microns, so a warning is issued in this case.

Affine transformations can be specified in the `transformation` argument, either by name or by directly specifying the matrix. The transformations specified by name will always preserve the center of the image. When named transformations are chained, name and parameter will be converted to matrix and translation vector the second time a transformation is specified. If the subsequent transformation happens to restore the image to its original place, then transformation specifications will be removed.

### Value

A BioFormatsImage object.

### See Also

`[isFull()]`, `[origin()]`

---

BioFormatsImage-getters

*Other BioFormatsImage getters*

---

### Description

`isFULL` indicates if the extent is the full extent of the image. `origin` gets the x-y coordinates of the origin of the image, i.e. the smallest possible x-y coordinate values within the full image.

### Usage

```
## S4 method for signature 'BioFormatsImage'
isFull(x)
```

```
## S4 method for signature 'BioFormatsImage'
origin(x)
```

```
## S4 method for signature 'BioFormatsImage'
transformation(x)
```

**Arguments**

x                    A [BioFormatsImage](#) object.

**Value**

For isFull: Logical scalar indicating whether the extent is the full extent. For origin: Numeric vector of length 2. For transformation, a list.

---

cbind,SpatialFeatureExperiment-method

*Concatenate SpatialFeatureExperiment objects*

---

**Description**

On top of the cbind method of SpatialExperiment, this method is needed to properly merge the spatialGraphs field in the different SFE objects. rowGeometries and annotGeometries also need to be combined properly.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'  
cbind(..., deparse.level = 1)
```

**Arguments**

...                    SFE objects to cbind.  
deparse.level    See [?rbind](#).

**Value**

A combined SFE object.

**Examples**

```
library(SFEData)  
sfe_small <- McKellarMuscleData(dataset = "small")  
sfe_small2 <- McKellarMuscleData(dataset = "small2")  
sfe2 <- cbind(sfe_small, sfe_small2)
```

---

changeSampleIDs	<i>Change sample IDs</i>
-----------------	--------------------------

---

**Description**

Change sample IDs in all fields of the SFE object where sample IDs are present, not just the colData.

**Usage**

```
changeSampleIDs(sfe, replacement)
```

**Arguments**

sfe	A SpatialFeatureExperiment object.
replacement	A named character vector whose names are the existing sample IDs to be changed and whose values are the corresponding replacements.

**Value**

An SFE object.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
sfe <- changeSampleIDs(sfe, c(Vis5A = "sample01"))
sampleIDs(sfe)
```

---

colFeatureData	<i>Get global spatial analysis results and metadata of colData, rowData, and geometries</i>
----------------	---

---

**Description**

Results of spatial analyses on columns in colData, rowData, and geometries are stored in their metadata, which can be accessed by the [metadata](#) function. The colFeatureData function allows the users to more directly access these results.

**Usage**

```
colFeatureData(sfe)

rowFeatureData(sfe)

geometryFeatureData(sfe, type, MARGIN = 2L)

reducedDimFeatureData(sfe, dimred)
```



**Arguments**

sfe	An SFE object.
type	Which geometry, can be name (character) or index (integer)
MARGIN	Integer, 1 means rowGeometry, 2 means colGeometry, and 3 means annotGeometry. Defaults to 2, colGeometry.
dimred	Name of a dimension reduction, can be seen in <a href="#">reducedDimNames</a> .

**Value**

A DataFrame.

**See Also**

`getParams`

**Examples**

```
library(SpatialFeatureExperiment)
library(SingleCellExperiment)
library(SFEData)
library(Voyager)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
# Moran's I for colData
sfe <- colDataMoransI(sfe, "nCounts")
colFeatureData(sfe)
```

---

colGeometries

*Column geometry getters and setters*

---

**Description**

`colGeometries` are geometries that correspond to columns of the gene count matrix, such as Visium spots or cells. Same as `dimGeometry(x, MARGIN = 2L, ...)`, with convenience wrappers for getters and setters of special geometries:

**spotPoly** Polygons of spots from technologies such as Visium, ST, and slide-seq, which do not correspond to cells. Centroids of the polygons are stored in `spatialCoords` of the underlying `SpatialExperiment` object.

**ROI** Polygons of regions of interest (ROIs) from technologies such as laser capture microdissection (LCM) and GeoMX DSP. These should correspond to columns of the gene count matrix.

**cellSeg** Cell segmentation polygons. If the columns of the gene count matrix are single cells, then this is stored in `colGeometries`. Otherwise, this is stored in [annotGeometries](#).

**nucSeg** Similar to `cellSeg`, but for nuclei rather than whole cell.

**Usage**

```

colGeometry(x, type = 1L, sample_id = 1L, withDimnames = TRUE)

colGeometry(
  x,
  type = 1L,
  sample_id = 1L,
  withDimnames = TRUE,
  translate = TRUE
) <- value

colGeometries(x, withDimnames = TRUE)

colGeometries(x, withDimnames = TRUE, translate = TRUE) <- value

colGeometryNames(x)

colGeometryNames(x) <- value

spotPoly(x, sample_id = 1L, withDimnames = TRUE)

spotPoly(x, sample_id = 1L, withDimnames = TRUE, translate = TRUE) <- value

centroids(x, sample_id = 1L, withDimnames = TRUE)

centroids(x, sample_id = 1L, withDimnames = TRUE, translate = TRUE) <- value

ROIpoly(x, sample_id = 1L, withDimnames = TRUE)

ROIpoly(x, sample_id = 1L, withDimnames = TRUE, translate = TRUE) <- value

cellSeg(x, sample_id = 1L, withDimnames = TRUE)

cellSeg(x, sample_id = 1L, withDimnames = TRUE, translate = TRUE) <- value

nucSeg(x, sample_id = 1L, withDimnames = TRUE)

nucSeg(x, sample_id = 1L, withDimnames = TRUE, translate = TRUE) <- value

```

**Arguments**

x	A SpatialFeatureExperiment object.
type	An integer specifying the index or string specifying the name of the *Geometry to query or replace. If missing, then the first item in the *Geometries will be returned or replaced.
sample_id	Sample ID to get or set geometries.
withDimnames	Logical. If TRUE, then the dimnames (colnames or rownames) of the gene count matrix should correspond to row names of the sf data frames of interest.

translate	Logical. Only used if <code>removeEmptySpace</code> has been run of the SFE object. If that's the case, this argument indicates whether the new value to be assigned to the geometry is in the coordinates prior to removal of empty space so it should be translated to match the new coordinates after removing empty space. Default to TRUE.
value	Value to set. For <code>dimGeometry</code> , must be a <code>sf</code> data frame with the same number of rows as size in the dimension of interest, or an ordinary data frame that can be converted to such a <code>sf</code> data frame (see <code>df2sf</code> ). For <code>dimGeometries</code> , must be a list of such <code>sf</code> or ordinary data frames.

### See Also

[`dimGeometries()`], [`rowGeometries()`]

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
cgs <- colGeometries(sfe)
spots <- spotPoly(sfe)
```

---

crop

*Crop an SFE object with a geometry*

---

### Description

Returns an SFE object whose specified `colGeometry` returns TRUE with a geometric predicate function (usually `intersects`) with another geometry of interest. This can be used to subset an SFE object with a tissue boundary or histological region polygon, or crop away empty spaces. After cropping, not only will the cells/spots be subsetted, but also all geometries will be cropped.

### Usage

```
crop(
  x,
  y = NULL,
  colGeometryName = 1L,
  sample_id = "all",
  pred = deprecated(),
  op = st_intersection,
  keep_whole = "none",
  cover = FALSE,
  xmin = deprecated(),
  xmax = deprecated(),
  ymin = deprecated(),
  ymax = deprecated()
)
```

**Arguments**

<code>x</code>	An SFE object.
<code>y</code>	An object of class <code>sf</code> , <code>sfg</code> , <code>sfc</code> with which to crop the SFE object, or a bounding box with the format of the output of <code>bbox</code> , <a href="#">SpatialFeatureExperiment-method</a> .
<code>colGeometryName</code>	Column geometry to used to indicate which cells/spots to keep.
<code>sample_id</code>	Samples to crop. Optional when only one sample is present. Can be multiple samples, or "all", which means all samples. For multiple samples, <code>sf</code> data frame <code>y</code> may have column <code>sample_id</code> indicating which geometry subsets which sample or matrix <code>y</code> may indicate sample specific bounding boxes in its column names. Only samples included in the indicated sample IDs are subsetted. If sample is not indicated in <code>y</code> , then the same geometry or bounding box is used to subset all samples specified in the <code>sample_id</code> argument.
<code>pred</code>	Deprecated. The binary predicate is now tied to the geometric operation specified in <code>op</code> .
<code>op</code>	A geometric operation function to crop the geometries in the SFE object. Only <a href="#">st_intersection</a> and <a href="#">st_difference</a> are allowed. If "intersection", then only things inside <code>y</code> is kept after cropping. If "difference", then only things outside <code>y</code> is kept.
<code>keep_whole</code>	Character vector, can be one or more of "col" and "annot" to keep whole items from <code>colGeometries</code> or <code>annotGeometries</code> , keeping geometries that partially intersect with <code>y</code> whole. This can greatly speed up code while not breaking geometries into multiple pieces. Can also be "none" so all geometries are actually cropped.
<code>cover</code>	Logical, whether the geometries in <code>x</code> must be entirely covered by <code>y</code> if <code>op = st_intersection</code> or whether <code>x</code> must be entirely outside <code>y</code> if <code>op = st_difference</code> . Only relevant when <code>keep_whole != "none"</code> .
<code>xmin</code>	Deprecated. Supply the bounding box to argument <code>y</code> instead.
<code>xmax</code>	Deprecated.
<code>ymin</code>	Deprecated.
<code>ymax</code>	Deprecated.

**Details**

3D geometries are allowed, but geometric operations can only be performed in `x` and `y` but not `z`.

**Value**

An SFE object. There is no guarantee that the geometries after cropping are still all valid or preserve the original geometry class.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData("small")
```

```
# Subset sfe to only keep spots on tissue
sfe_on_tissue <- crop(sfe, tissueBoundary(sfe),
  colGeometryName = "spotPoly",
  sample_id = "Vis5A"
)
```

---

cropImg

*Crop images*

---

## Description

Crop images of class \*Image in this package with a bounding box.

## Usage

```
## S4 method for signature 'SpatRasterImage'
cropImg(x, bbox, filename = "")
```

```
## S4 method for signature 'BioFormatsImage'
cropImg(x, bbox)
```

```
## S4 method for signature 'ExtImage'
cropImg(x, bbox)
```

## Arguments

x	An object of class *Image as implemented in this package.
bbox	Numeric vector with names "xmin", "xmax", "ymin", "ymax", in any order, to specify the bounding box.
filename	Output file name for transformed SpatRaster.

## Value

Image of the same class as input but cropped. For BioFormatsImage, the image is not loaded into memory; only the extent is changed.

## See Also

Other image methods: [SFE-image](#), [affineImg\(\)](#), [dim](#), [BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

## Description

While the `SpatialFeatureExperiment` constructor and `*Geometry` replacement methods can convert properly formatted ordinary data frames into `sf` objects which are used to store the geometries internally, the user might want to do the conversion, check if the geometry is valid, and inspect and fix any invalid geometries.

## Usage

```
df2sf(
  df,
  spatialCoordsNames = c("x", "y"),
  spotDiameter = NA,
  geometryType = c("POINT", "LINESTRING", "POLYGON", "MULTIPOINT", "MULTILINESTRING",
    "MULTIPOLYGON"),
  group_col = "group",
  id_col = "ID",
  subid_col = "subID",
  check = TRUE,
  BPPARAM = deprecated(),
  ...
)
```

## Arguments

<code>df</code>	An ordinary data frame, i.e. not <code>sf</code> . Or a matrix that can be converted to a data frame.
<code>spatialCoordsNames</code>	Column names in <code>df</code> that specify spatial coordinates.
<code>spotDiameter</code>	Spot diameter for technologies with arrays of spots of fixed diameter per slide, such as Visium, ST, DBiT-seq, and slide-seq. The diameter must be in the same unit as the coordinates in the <code>*Geometry</code> arguments. Ignored for geometries that are not <code>POINT</code> or <code>MULTIPOINT</code> .
<code>geometryType</code>	Type of geometry to convert the ordinary data frame to. If the geometry in <code>df</code> is de facto points, then this argument will be ignored and the returned <code>sf</code> will have geometry type <code>POINT</code> .
<code>group_col</code>	Column to indicate which coordinates for which <code>MULTI</code> geometry, such as to identify which <code>MULTIPOLYGON</code> or <code>MULTIPOINT</code> .
<code>id_col</code>	Column to indicate coordinates for which geometry, within a <code>MULTI</code> geometry if applicable, such as to identify which <code>POLYGON</code> or which polygon within a <code>MULTIPOLYGON</code> .
<code>subid_col</code>	Column to indicate coordinates for holes in polygons.

check	Logical, whether to check the input data frame for issues related to constructing the geometry of interest such as number of vertices per geometry. If FALSE, it will save a bit of time, which is useful when the input is already known to be good.
BPPARAM	Deprecated. The 'sfheaders' package is used in <code>df2sf</code> for much better performance.
...	Other arguments passed to 'sf::st_buffer', mainly to make polygon shapes, eg Visium spot 'endCapStyle = "ROUND"' and VisiumHD bin 'endCapStyle = "SQUARE"'

### Value

An sf object.

### Examples

```
# Points, use spotDiameter to convert to circle polygons
# This is done to Visium spots
pts_df <- readRDS(system.file("extdata/pts_df.rds",
  package = "SpatialFeatureExperiment"
))
sf_use <- df2sf(pts_df, geometryType = "POINT", spotDiameter = 0.1)
# Linestring
ls_df <- readRDS(system.file("extdata/ls_df.rds",
  package = "SpatialFeatureExperiment"
))
sf_use <- df2sf(ls_df, geometryType = "LINESTRING")
# Polygon
pol_df <- readRDS(system.file("extdata/pol_df.rds",
  package = "SpatialFeatureExperiment"
))
sf_use <- df2sf(pol_df,
  geometryType = "POLYGON",
  spatialCoordsNames = c("V1", "V2")
)
# Multipolygon
mpol_df <- readRDS(system.file("extdata/mpol_df.rds",
  package = "SpatialFeatureExperiment"
))
sf_use <- df2sf(mpol_df,
  geometryType = "MULTIPOLYGON",
  spatialCoordsNames = c("V1", "V2")
)
# Multiple sample_ids present
multipts_df <- readRDS(system.file("extdata/multipts_df.rds",
  package = "SpatialFeatureExperiment"
))
sf_use <- df2sf(multipts_df, geometryType = "MULTIPOINT")
```

---

dim, BioFormatsImage-method

*Find dimension of BioFormatsImage*

---

### Description

This is different from other classes. The metadata is read where the dimensions in pixels can be found. The image itself is not read into memory here.

### Usage

```
## S4 method for signature 'BioFormatsImage'
dim(x)
```

### Arguments

x                    A [BioFormatsImage](#) object.

### Value

An integer vector of length 5 showing the number of rows and columns in the full resolution image. The 5 dimensions are in the order of XYCZT: x, y, channel, z, and time. This is not changed by transformations. Use [ext](#) to see the extent after transformation.

### See Also

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

dimGeometries

*Dimension geometry methods*

---

### Description

"Dimension geometry" refers to Simple Feature (sf) geometries associated with rows (features, genes) or columns (cells or spots) of the gene count matrix in the `SpatialFeatureExperiment` object. For each dimension, the number of rows in the sf data frame specifying the geometries must match the size of the dimension of interest. For example, there must be the same number of rows in the sf data frame describing cells as there are cells in the gene count matrix. This page documents getters and setters for the dimension geometries. The getters and setters are implemented in a way similar to those of `reducedDims` in `SingleCellExperiment`.



**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
dimGeometries(x, MARGIN = 2, withDimnames = TRUE)

## S4 replacement method for signature 'SpatialFeatureExperiment'
dimGeometries(x, MARGIN, withDimnames = TRUE, translate = TRUE, ...) <- value

## S4 method for signature 'SpatialFeatureExperiment'
dimGeometryNames(x, MARGIN)

## S4 replacement method for signature 'SpatialFeatureExperiment,numeric,character'
dimGeometryNames(x, MARGIN) <- value

## S4 method for signature 'SpatialFeatureExperiment'
dimGeometry(x, type = 1L, MARGIN, sample_id = 1L, withDimnames = TRUE)

## S4 replacement method for signature 'SpatialFeatureExperiment'
dimGeometry(
  x,
  type = 1L,
  MARGIN,
  sample_id = 1L,
  withDimnames = TRUE,
  translate = TRUE,
  ...
) <- value
```

**Arguments**

x	A <code>SpatialFeatureExperiment</code> object.
MARGIN	As in <a href="#">apply</a> . 1 stands for rows and 2 stands for columns.
withDimnames	Logical. If TRUE, then the dimnames (colnames or rownames) of the gene count matrix should correspond to row names of the sf data frames of interest.
translate	Logical. Only used if <a href="#">removeEmptySpace</a> has been run of the SFE object. If that's the case, this argument indicates whether the new value to be assigned to the geometry is in the coordinates prior to removal of empty space so it should be translated to match the new coordinates after removing empty space. Default to TRUE.
...	<code>spatialCoordsNames</code> , <code>spotDiameter</code> , <code>geometryType</code> passed to <a href="#">df2sf</a> . Defaults are the same as in <a href="#">df2sf</a> . For <code>dimGeometries&lt;-</code> only: <code>geometryType</code> can be a character vector of the geometry type of each data frame in the list of the same length as the list if the data frames specify different types of geometries.
value	Value to set. For <code>dimGeometry</code> , must be a sf data frame with the same number of rows as size in the dimension of interest, or an ordinary data frame that can be converted to such a sf data frame (see <a href="#">df2sf</a> ). For <code>dimGeometries</code> , must be a list of such sf or ordinary data frames.

`type` An integer specifying the index or string specifying the name of the \*Geometry to query or replace. If missing, then the first item in the \*Geometries will be returned or replaced.

`sample_id` Sample ID to get or set geometries.

### Value

Getters for multiple geometries return a named list. Getters for names return a character vector of the names. Getters for single geometries return an sf data frame. Setters return an SFE object.

### See Also

[colGeometries()], [rowGeometries()]

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")

# Get all column geometries as a named list
# Use MARGIN = 1 or rowGeometry/ies for rowGeometries
cgs <- dimGeometries(sfe, MARGIN = 2)
# Or equivalently
cgs <- colGeometries(sfe)

# Set all column geometries with a named list
dimGeometries(sfe, MARGIN = 2) <- cgs
# Or equivalently
colGeometries(sfe) <- cgs

# Get names of column geometries
cgns <- dimGeometryNames(sfe, MARGIN = 2)
cgns <- colGeometryNames(sfe)

# Set column geometry names
dimGeometryNames(sfe, MARGIN = 2) <- cgns
colGeometryNames(sfe) <- cgns

# Get a specific column geometry by name
spots <- dimGeometry(sfe, "spotPoly", MARGIN = 2)
spots <- colGeometry(sfe, "spotPoly")
# Or equivalently, the wrapper specifically for Visium spot polygons,
# for the name "spotPoly"
spots <- spotPoly(sfe)
# Other colGeometry wrappers for specific names:
# ROI Poly (for LCM and GeoMX DSP), cellSeg and nucSeg (for MERFISH; would
# query annotGeometries for Visium)
# rowGeometry wrappers for specific names: txSpots (MERFISH transcript spots)
# By index
spots <- colGeometry(sfe, 1L)

# Multiple samples, only get geometries for one sample
```

```

sfe2 <- McKellarMuscleData("small2")
sfe_combined <- cbind(sfe, sfe2)
spots1 <- colGeometry(sfe, "spotPoly", sample_id = "Vis5A")
spots2 <- spotPoly(sfe_combined, sample_id = "sample02")
# Get geometries for multiple samples
spots3 <- spotPoly(sfe_combined, sample_id = c("Vis5A", "sample02"))
# All samples
spots3 <- spotPoly(sfe_combined, sample_id = "all")

# Set specific column geometry by name
colGeometry(sfe, "foobar") <- spots
# Or use wrapper
spotPoly(sfe) <- spots
# Specify sample_id
colGeometry(sfe_combined, "foobar", sample_id = "Vis5A") <- spots1
# Only entries for the specified sample are set.
foobar <- colGeometry(sfe_combined, "foobar", sample_id = "sample02")

```

---

ext

*Get and set extent of image objects*


---

## Description

Unlike in `SpatialExperiment`, images in SFE have extents which are used to align them to the geometries and in geometric operations on SFE objects. These functions get or set the extent for S4 image classes inheriting from `VirtualSpatialImage` implemented in the SFE package.

## Usage

```

## S4 method for signature 'BioFormatsImage'
ext(x)

## S4 method for signature 'ExtImage'
ext(x)

## S4 method for signature 'SpatRasterImage'
ext(x)

## S4 replacement method for signature 'BioFormatsImage,numeric'
ext(x) <- value

## S4 replacement method for signature 'ExtImage,numeric'
ext(x) <- value

## S4 replacement method for signature 'SpatRasterImage,numeric'
ext(x) <- value

```

**Arguments**

x	A *Image object.
value	A numeric vector with names "xmin", "xmax", "ymin", "ymax" specifying the extent to use.

**Value**

Getters return a numeric vector specifying the extent. Setters return a \*Image object of the same class as the input.

**Note**

For `SpatRasterImage`, the image may be may not be loaded into memory. You can check if the image is loaded into memory with `terra::inMemory(imgRaster(x))`, and check the original file path with `imgSource`. If the image is not loaded into memory, then the original file must be present at the path indicated by `imgSource` in order for any code using the image to work, which includes this function `ext`.

For `BioFormatsImage`, internally only the pre-transform extent is stored. The `ext` getter will apply the transformation on the fly. The setter sets the pre-transformation extent.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

ExtImage

*Use the EBImage Image class in SFE objects*

---

**Description**

This is a thin wrapper around the [Image](#) class in the `EBImage` package so it inherits from `VirtualSpatialImage` to be compatible with `SpatialExperiment` from which `SFE` inherits. An `ext` field is added to specify the spatial extent of the image in microns to facilitate geometric operations on the `SFE` object (including the images) and plotting with `Voyager`.

**Usage**

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

ExtImage(img, ext = NULL)
```

**Arguments**

object	An ExtImage object.
img	An Image object or anything that inherits from Image such as AnnotatedImage in RBioFormats.
ext	Numeric vector with names "xmin", "xmax", "ymin", "ymax" in microns indicating the spatial extent covered by the image. If NULL, then the extent will be inferred from the metadata, from physical pixel size and the number of pixels.

**Value**

An ExtImage object.

---

findSpatialNeighbors, SpatialFeatureExperiment-method  
*Find spatial neighborhood graph*

---

**Description**

This function wraps all spatial neighborhood graphs implemented in the package spdep for the SpatialFeatureExperiment (SFE) class, to find spatial neighborhood graphs for the entities represented by columns or rows of the gene count matrix in the SFE object or spatial entities in the annotGeometries field of the SFE object. Results are stored as listw objects in the spatialGraphs field of the SFE object, as listw is used in many methods that facilitate the spatial neighborhood graph in the spdep, spatialreg, and adespatial. The edge weights of the graph in the listw object are by default style W (see [nb2listw](#)) and the unweighted neighbor list is in the neighbours field of the listw object.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
findSpatialNeighbors(
  x,
  sample_id = "all",
  type = "spatialCoords",
  MARGIN = 2,
  method = c("tri2nb", "knearneigh", "dnearneigh", "gabrielneigh", "relativeneigh",
    "soi.graph", "poly2nb"),
  dist_type = c("none", "idw", "exp", "dpd"),
  glist = NULL,
  style = c("raw", "W", "B", "C", "U", "minmax", "S"),
  nn_method = c("bioc", "spdep"),
  alpha = 1,
  dmax = NULL,
  BPPARAM = SerialParam(),
  BNPARAM = KmknnParam(),
  zero.policy = TRUE,
```

```
    ...
  )
```

### Arguments

x	A <a href="#">SpatialFeatureExperiment</a> object.
sample_id	Which sample(s) in the SFE object to use for the graph. Can also be "all", which means this function will compute the graph for all samples independently.
type	Name of the geometry associated with the MARGIN of interest for which to compute the graph.
MARGIN	Just like in <a href="#">apply</a> , where 1 stands for row, 2 stands for column. Here, in addition, 3 stands for annotation, to query the <a href="#">annotGeometries</a> , such as nuclei segmentation in a Visium data
method	Name of function in the package <code>spdep</code> to use to find the spatial neighborhood graph.
dist_type	Type of distance-based weight. "none" means not using distance-based weights; the edge weights of the spatial neighborhood graph will be entirely determined by the <code>style</code> argument. "idw" means inverse distance weighting. "exp" means exponential decay. "dpd" means double-power distance weights. See <a href="#">nb2listwdist</a> for details.
glist	list of general weights corresponding to neighbours
style	style can take values "W", "B", "C", "U", "minmax" and "S"
nn_method	Method to find k nearest neighbors and distance based neighbors. Can be either "bioc" or "spdep". For "bioc", methods from <code>BiocNeighbors</code> are used. For "spdep", methods from the <code>spdep</code> package are used. The "bioc" option is more scalable to larger datasets and supports multithreading.
alpha	Only relevant when <code>dist_type = "dpd"</code> .
dmax	Only relevant when <code>dist_type = "dpd"</code> .
BPPARAM	A <a href="#">BiocParallelParam</a> object for multithreading. Only used for k nearest neighbor and distance based neighbor with <code>nn_method = "bioc"</code> .
BNPARAM	A <a href="#">BiocNeighborParam</a> object specifying the algorithm to find k nearest neighbors and distance based neighbors with <code>nn_method = "bioc"</code> . For distance based neighbors, only <a href="#">KmknnParam</a> and <a href="#">VptreeParam</a> are applicable.
zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
...	Extra arguments passed to the <code>spdep</code> function stated in the <code>method</code> argument, such as <code>k</code> , <code>use_kd_tree</code> , <code>d1</code> , <code>d2</code> , <code>nnmult</code> , <code>sym</code> , and <code>quadsegs</code> . Note that any arguments about using longitude and latitude, which are irrelevant, are ignored.

### Value

For one sample, then a `listw` object representing the graph, with an attribute "method" recording the function used to build the graph, its arguments, and information about the geometry for which the graph was built. The attribute is used to reconstruct the graphs when the SFE object is subsetted

since some nodes in the graph will no longer be present. If `sample_id = "all"` or has length  $> 1$ , then a named list of `listw` objects, whose names are the `sample_ids`. To add the list for multiple samples to a SFE object, specify the name argument in the `spatialGraphs` replacement method, so graph of the same name will be added to the SFE object for each sample.

### Note

`style = "raw"` is only applicable when `dist_type` is not "none". If `dist_type = "none"` and `style = "raw"`, then style will default to "W". Using distance based weights does not supplant finding a spatial neighborhood graph. The spatial neighborhood graph is first found and then its edges weighted based on distance in this function.

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
# sample_id is optional when only one sample is present
g <- findSpatialNeighbors(sfe, sample_id = "Vis5A")
attr(g, "method")
# Returns named list for multiple samples
sfe2 <- McKellarMuscleData(dataset = "small2")
sfe_combined <- cbind(sfe, sfe2)
gs <- findSpatialNeighbors(sfe, sample_id = "all")
```

---

findVisiumGraph

*Find spatial neighborhood graphs for Visium spots*

---

### Description

Visium spots are arranged in a hexagonal grid. This function uses the known locations of the Visium barcodes to construct a neighborhood graph, so adjacent spots are connected by edges. Since the known rows and columns of the spots are used, the unit the spot centroid coordinates are in does not matter.

### Usage

```
findVisiumGraph(x, sample_id = "all", style = "W", zero.policy = NULL)
```

### Arguments

<code>x</code>	A <code>SpatialFeatureExperiment</code> object with Visium data. Column names of the gene count matrix must be Visium barcodes, which may have a numeric suffix to distinguish between samples (e.g. "AAACAACGAATAGTTC-1").
<code>sample_id</code>	Which sample(s) in the SFE object to use for the graph. Can also be "all", which means this function will compute the graph for all samples independently.
<code>style</code>	style can take values "W", "B", "C", "U", "minmax" and "S"
<code>zero.policy</code>	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

**Value**

For one sample, then a `listw` object representing the graph, with an attribute "method" recording the function used to build the graph, its arguments, and information about the geometry for which the graph was built. The attribute is used to reconstruct the graphs when the SFE object is subsetting since some nodes in the graph will no longer be present. If `sample_id = "all"` or has length  $> 1$ , then a named list of `listw` objects, whose names are the `sample_ids`. To add the list for multiple samples to a SFE object, specify the name argument in the `spatialGraphs` replacement method, so graph of the same name will be added to the SFE object for each sample.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
g <- findVisiumGraph(sfe)
# For multiple samples, returns named list
sfe2 <- McKellarMuscleData(dataset = "small2")
sfe_combined <- cbind(sfe, sfe2)
gs <- findVisiumGraph(sfe, sample_id = "all")
```

---

formatTxSpots

*Read and process transcript spots geometry for SFE*


---

**Description**

The function 'formatTxSpots' reads the transcript spot coordinates of smFISH-based data and formats the data. The data is not added to an SFE object. If the file specified in 'file\_out' already exists, then this file will be read instead of the original file in the 'file' argument, so the processing is not run multiple times. The function 'addTxSpots' adds the data read and processed in 'formatTxSpots' to the SFE object, and reads all transcript spot data. To only read a subset of transcript spot data, first use 'formatTxSpots' to write the re-formatted data to disk. Then read the specific subset and add them separately to the SFE object with the setter functions.

**Usage**

```
formatTxSpots(
  file,
  dest = c("rowGeometry", "colGeometry"),
  spatialCoordsNames = c("global_x", "global_y", "global_z"),
  gene_col = "gene",
  cell_col = "cell_id",
  z = "all",
  phred_col = "qv",
  min_phred = 20,
  split_col = NULL,
  not_in_cell_id = c("-1", "UNASSIGNED"),
  z_option = c("3d", "split"),
  flip = FALSE,
```



```

    file_out = NULL,
    BPPARAM = SerialParam(),
    return = TRUE
)

addTxSpots(
  sfe,
  file,
  sample_id = 1L,
  spatialCoordsNames = c("global_x", "global_y", "global_z"),
  gene_col = "gene",
  z = "all",
  phred_col = "qv",
  min_phred = 20,
  split_col = NULL,
  z_option = c("3d", "split"),
  flip = FALSE,
  file_out = NULL,
  BPPARAM = SerialParam()
)

```

### Arguments

file	File with the transcript spot coordinates. Should be one row per spot when read into R and should have columns for coordinates on each axis, gene the transcript is assigned to, and optionally cell the transcript is assigned to. Must be csv, tsv, or parquet.
dest	Where in the SFE object to store the spot geometries. This affects how the data is processed. Options: <b>rowGeometry</b> All spots for each gene will be a ‘MULTIPOINT’ geometry, regardless of whether they are in cells or which cells they are assigned to. <b>colGeometry</b> The spots for each gene assigned to a cell of interest will be a ‘MULTIPOINT’ geometry; since the gene count matrix is sparse, the geometries are NOT returned to memory.
spatialCoordsNames	Column names for the x, y, and optionally z coordinates of the spots. The defaults are for Vizgen.
gene_col	Column name for genes.
cell_col	Column name for cell IDs, ignored if ‘dest = "rowGeometry"’. Can have length > 1 when multiple columns are needed to uniquely identify cells, in which case the contents of the columns will be concatenated, such as in CosMX data where cell ID is only unique within the same FOV. Default "cell_id" is for Vizgen MERFISH. Should be ‘c("cell_ID", "fov")’ for CosMX.
z	Index of z plane to read. Can be "all" to read all z-planes into MULTIPOINT geometries with XYZ coordinates. If z values are not integer, then spots with all z values will be read.
phred_col	Column name for Phred scores of the spots.

min_phred	Minimum Phred score to keep spot. By default 20, the conventional threshold indicating "acceptable", meaning that there's 1 chance that the spot was decoded in error.
split_col	Categorical column to split the geometries, such as cell compartment the spots are assigned to as in the "CellComp" column in CosMX output.
not_in_cell_id	Value of cell ID indicating that the spot is not assigned to any cell, such as "-1" in Vizgen MERFISH and "0" in CosMX. When there're multiple columns for 'cell_col', the first column is used to identify spots that are not in cells.
z_option	What to do with z coordinates. "3d" is to construct 3D geometries. "split" is to create a separate 2D geometry for each z-plane so geometric operations are fully supported but some data wrangling is required to perform 3D analyses. When the z coordinates are not integers, 3D geometries will always be constructed since there are no z-planes to speak of. This argument does not apply when 'spatialCoordsNames' has length 2.
flip	Logical, whether to flip the geometry to match image. Here the y coordinates are simply set to -y, so the original bounding box is not preserved. This is consistent with readVizgen and readXenium.
file_out	Name of file to save the geometry or raster to disk. Especially when the geometries are so large that it's unwieldy to load everything into memory. If this file (or directory for multiple files) already exists, then the existing file(s) will be read, skipping the processing. When writing the file, extensions supplied are ignored and extensions are determined based on 'dest'.
BPPARAM	<a href="#">BiocParallelParam</a> object to specify multithreading to convert raw char in some parquet files to R objects. Not used otherwise.
return	Logical, whether to return the geometries in memory. This does not depend on whether the geometries are written to file. Always 'FALSE' when 'dest = "colGeometry"'.
sfe	A 'SpatialFeatureExperiment' object.
sample_id	Which sample in the SFE object the transcript spots should be added to.

### Value

A sf data frame for vector geometries if 'file\_out' is not set. 'SpatRaster' for raster. If there are multiple files written, such as when splitting by cell compartment or when 'dest = "colGeometry"', then a directory with the same name as 'file\_out' will be created (but without the extension) and the files are written to that directory with informative names. 'parquet' files that can be read with 'st\_read' is written for vector geometries. When 'return = FALSE', the file name or directory (when there're multiple files) is returned.

The 'sf' data frame, or path to file where geometries are written if 'return = FALSE'.

### Note

When 'dest = "colGeometry"', the geometries are always written to disk and not returned in memory, because this is essentially the gene count matrix, which is sparse. This kind of reformatting is implemented so users can read in MULTIPOINT geometries with transcript spots for each gene assigned to each cell for spatial point process analyses, where not all genes are loaded at once.

**Examples**

```

# Default arguments are for MERFISH
fp <- tempdir()
dir_use <- SFEData::VizgenOutput(file_path = file.path(fp, "vizgen_test"))
g <- formatTxSpots(file.path(dir_use, "detected_transcripts.csv"))
unlink(dir_use, recursive = TRUE)

# For CosMX, note the colnames, also dest = "colGeometry"
# Results are written to the tx_spots directory
dir_use <- SFEData::CosMXOutput(file_path = file.path(fp, "cosmx_test"))
cg <- formatTxSpots(file.path(dir_use, "Run5642_S3_Quarter_tx_file.csv"),
  dest = "colGeometry", z = "all",
  cell_col = c("cell_ID", "fov"),
  gene_col = "target", not_in_cell_id = "0",
  spatialCoordsNames = c("x_global_px", "y_global_px", "z"),
  file_out = file.path(dir_use, "tx_spots"))
# Cleanup
unlink(dir_use, recursive = TRUE)

```

---

formatTxTech

*Read and process transcript spots for specific commercial technologies*


---

**Description**

To preset parameters such as spatialCoordsNames, gene\_col, cell\_col, and phred\_col that are standard for the output of the technology.

**Usage**

```

formatTxTech(
  data_dir,
  tech = c("Vizgen", "Xenium", "CosMX"),
  dest = c("rowGeometry", "colGeometry"),
  z = "all",
  min_phred = 20,
  split_cell_comps = FALSE,
  z_option = c("3d", "split"),
  flip = FALSE,
  file_out = NULL,
  BPPARAM = SerialParam(),
  return = TRUE
)

addTxTech(
  sfe,
  data_dir,
  sample_id = 1L,

```

```

tech = c("Vizgen", "Xenium", "CosMX"),
z = "all",
min_phred = 20,
split_cell_comps = FALSE,
z_option = c("3d", "split"),
flip = FALSE,
file_out = NULL,
BPPARAM = SerialParam()
)

```

### Arguments

data_dir	Top level output directory.
tech	Which technology whose output to read, must be one of "Vizgen", "Xenium", or "CosMX" though more technologies may be added later.
dest	Where in the SFE object to store the spot geometries. This affects how the data is processed. Options: <b>rowGeometry</b> All spots for each gene will be a 'MULTIPOINT' geometry, regardless of whether they are in cells or which cells they are assigned to. <b>colGeometry</b> The spots for each gene assigned to a cell of interest will be a 'MULTIPOINT' geometry; since the gene count matrix is sparse, the geometries are NOT returned to memory.
z	Which z-planes to read. Always "all" for Xenium where the z coordinates are not discrete.
min_phred	Minimum Phred score to keep spot. By default 20, the conventional threshold indicating "acceptable", meaning that there's 1 chance that the spot was decoded in error.
split_cell_comps	Only relevant to CosMX whose transcript spot file assigns the spots to cell components. Setting this argument to TRUE
z_option	What to do with z coordinates. "3d" is to construct 3D geometries. "split" is to create a separate 2D geometry for each z-plane so geometric operations are fully supported but some data wrangling is required to perform 3D analyses. When the z coordinates are not integers, 3D geometries will always be constructed since there are no z-planes to speak of. This argument does not apply when 'spatialCoordsNames' has length 2.
flip	Logical, whether to flip the geometry to match image. Here the y coordinates are simply set to -y, so the original bounding box is not preserved. This is consistent with readVizgen and readXenium.
file_out	Name of file to save the geometry or raster to disk. Especially when the geometries are so large that it's unwieldy to load everything into memory. If this file (or directory for multiple files) already exists, then the existing file(s) will be read, skipping the processing. When writing the file, extensions supplied are ignored and extensions are determined based on 'dest'.
BPPARAM	<a href="#">BiocParallelParam</a> object to specify multithreading to convert raw char in some parquet files to R objects. Not used otherwise.

return	Logical, whether to return the geometries in memory. This does not depend on whether the geometries are written to file. Always 'FALSE' when 'dest = "colGeometry"'.
sf	A 'SpatialFeatureExperiment' object.
sample_id	Which sample in the SFE object the transcript spots should be added to.

**Value**

The 'sf' data frame, or path to file where geometries are written if 'return = FALSE'.

**Examples**

```
library(SFData)
fp <- tempdir()
dir_use <- XeniumOutput("v2", file_path = file.path(fp, "xenium_test"))
fn_tx <- formatTxTech(dir_use, tech = "Xenium", flip = TRUE, return = FALSE,
                      file_out = file.path(dir_use, "tx_spots.parquet"))
```

---

gdalParquetAvailable *Check if Parquet GDAL driver is available*

---

**Description**

The GeoParquet files for geometries are typically written and read with the sfarrow package, but to add only a select few genes to the SFE object say for visualization purposes, the Parquet GDAL driver is required in order to use GDAL's SQL to query the GeoParquet file to only load the few genes requested. The transcript spots from a large dataset can take up a lot of memory if all loaded.

**Usage**

```
gdalParquetAvailable()
```

**Details**

The Parquet driver has been supported since GDAL 3.5.0. The arrow C++ library must be installed in order to make the Parquet driver available. When arrow is installed, newer versions of GDAL installed from Homebrew (Mac) should have the Parquet driver. For Linux, the binary from apt-get's default repo is 3.4.1 (as of April 2024). To use the Parquet driver, GDAL may need to be installed from source. See script from the [geospatial rocker](#). A Voyager docker container with the Parquet driver will soon be provided.

**Value**

Logical, indicating whether the Parquet driver is present.

**Examples**

```
gdalParquetAvailable()
```

---

<code>getParams</code>	<i>Get parameters used in spatial methods</i>
------------------------	---

---

### Description

The `getParams` function allows users to access the parameters used to compute the results that may be stored in `colFeatureData`.

### Usage

```
getParams(
  sfe,
  name,
  local = FALSE,
  colData = FALSE,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  reducedDimName = NULL
)
```

### Arguments

<code>sfe</code>	A <code>SpatialFeatureExperiment</code> object.
<code>name</code>	Name used to store the results.
<code>local</code>	Logical, whether the results of interest come from a local spatial method.
<code>colData</code>	Logical, whether the results were computed for a column of <code>colData(sfe)</code> .
<code>colGeometryName</code>	To get results for a <code>colGeometry</code> .
<code>annotGeometryName</code>	To get results for an <code>annotGeometry</code> ; <code>colGeometry</code> has precedence so this argument is ignored if <code>colGeometryName</code> is specified.
<code>reducedDimName</code>	Name of a dimension reduction, can be seen in <code>reducedDimNames</code> . <code>colGeometryName</code> and <code>annotGeometryName</code> have precedence over <code>reducedDimName</code> .

### Value

A named list showing the parameters

### Examples

```
library(SFEData)
library(scater)
library(Voyager)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
sfe <- colDataMoransI(sfe, "nCounts")
getParams(sfe, "moran", colData = TRUE)
```

---

getPixelSize	<i>Get physical size of pixels</i>
--------------	------------------------------------

---

**Description**

This function gets physical size of pixels in each resolution of a OME-TIFF pyramid in [BioFormatsImage](#).

**Usage**

```
getPixelSize(file, resolution = 1L)
```

**Arguments**

file	Path to an OME-TIFF file.
resolution	Which resolution to query; 1 means the highest resolution. The pixels will be larger for the lower resolutions.

**Value**

Numeric vector of length 2 of pixel size in x and y. Usually they're the same.

**Examples**

```
library(SFEData)
fp <- tempdir()
dir_use <- XeniumOutput("v1", file_path = file.path(fp, "xenium_test"))
# RBioFormats null pointer error
try(getPixelSize(file.path(dir_use, "morphology_focus.ome.tif")))
getPixelSize(file.path(dir_use, "morphology_focus.ome.tif"))
unlink(dir_use, recursive = TRUE)
```

---

imageIDs	<i>Show all image_ids in the SFE object</i>
----------	---

---

**Description**

The title is self-explanatory. Some functions require image\_id to get or set images.

**Usage**

```
imageIDs(sfe)
```

**Arguments**

sfe	A SpatialFeatureExperiment object.
-----	------------------------------------

**Value**

A character vector of image\_ids.

**Examples**

```
fp <- system.file(file.path("extdata", "sample01"),
  package = "SpatialFeatureExperiment")
sfe <- read10xVisiumSFE(fp, type = "sparse")
imageIDs(sfe)
```

---

Img<-,SpatialExperiment-method

*Image setter*

---

**Description**

Modify or replace images stored in a SpatialExperiment object. This is different from [addImg](#) which adds the image from files and can't replace existing images, which is there to be consistent with SpatialExperiment. This setter here can replace existing images with another object that inherits from VirtualSpatialImage, including [SpatRasterImage](#), [BioFormatsImage](#), and [ExtImage](#).

**Usage**

```
## S4 replacement method for signature 'SpatialExperiment'
Img(x, sample_id = 1L, image_id, scale_fct = 1) <- value
```

**Arguments**

x	A SpatialExperiment object, which includes SFE.
sample_id	Which sample the image is associated with. Use <a href="#">sampleIDs</a> to get sample IDs present in the SFE object.
image_id	Image ID, such as "lowres" and "hires" for Visium data and "DAPI" and "PolyT" for Vizgen MERFISH data.
scale_fct	Scale factor to convert pixels in lower resolution to those in the full resolution. Only relevant to image classes implemented in SpatialExperiment but not SpatialFeatureExperiment because the spatial extent of images in SFE takes precedence.
value	New version of image to add, must inherit from VirtualSpatialImage.

**Value**

SFE object with the new image added.



**Examples**

```

library(EBImage)
library(SFEData)
library(RBioFormats)
fp <- tempdir()
fn <- XeniumOutput("v2", file_path = file.path(fp, "xenium_test"))
# Weirdly the first time I get the null pointer error
try(sfe <- readXenium(fn))
sfe <- readXenium(fn)
img <- getImg(sfe) |> toExtImage(resolution = 1L)
img <- img[, ,1] > 500
Img(sfe, image_id = "mask") <- img
imageIDs(sfe)
unlink(fn, recursive = TRUE)

```

---

imgRaster

*Get the image from \*Image class*


---

**Description**

In SFE, S4 classes inheriting from `VirtualSpatialImage` have been implemented to make these image classes compatible with `SpatialExperiment`. The `imgRaster` methods in SFE are meant to extract the original image from the `*Image` classes, such as `SpatRaster` from `SpatRasterImage`, and `Image` from `ExtImage` and `BioFormatsImage`. For `BioFormatsImage`, the image of the specified resolution will be read into memory as `AnnotatedImage`, which inherits from `EBImage::Image`.

**Arguments**

<code>x</code>	An object of class <code>*Image</code> as implemented in this package.
<code>resolution</code>	Resolution to read in from OME-TIFF, defaults to 4, which is a medium resolution in Xenium.

**Value**

`SpatRaster` from `SpatRasterImage`, and `Image` from `ExtImage` and `BioFormatsImage`. For `BioFormatsImage`, the image of the specified resolution will be read into memory as `AnnotatedImage` and `ExtImage`, which both inherit from `EBImage::Image`.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

imgSource	<i>Source of images that are on disk</i>
-----------	--

---

### Description

Get the file path of images that are on disk and not read into memory. Only applies to SpatRasterImage and BioFormatsImage.

### Usage

```
## S4 method for signature 'SpatRasterImage'
imgSource(x)
```

```
## S4 method for signature 'BioFormatsImage'
imgSource(x)
```

```
## S4 method for signature 'ExtImage'
imgSource(x)
```

### Arguments

x                    An object of class \*Image as implemented in this package.

### Value

String, file path to the original image on disk. For SpatRasterImage, if the image is loaded into memory, then NULL.

### See Also

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

internal-Voyager	<i>Internal functions also used in Voyager</i>
------------------	--

---

### Description

Not meant for the user, but exporting to be used internally in Voyager. But one day I may clean these up and remove the internal note for people building on top of SFE.

**Usage**

```
.value2df(value, use_geometry, feature = NULL)

.check_features(x, features, colGeometryName = NULL, swap_rownames = NULL)

.warn_symbol_duplicate(x, symbols, swap_rownames = "symbol")

.symbol2id(x, features, swap_rownames)

.check_sample_id(x, sample_id, one = TRUE, mustWork = TRUE)

.rm_empty_geometries(g, MARGIN)
```

**Value**

Internal

---

localResults	<i>Get and set results from local spatial statistics</i>
--------------	--

---

**Description**

Local spatial statics like local Moran's I, local Geary's C, Getis-Ord  $G_i^*$ , and geographically weighted summary statistics return values at each spatial location. Just like dimension reductions, these results are clearly associated with the broader SFE object, so they should have a place within the object. However, a separate field is needed because these analyses are conceptually distinct from dimension reduction. Also, each feature (e.g. gene) can have its own results with values at each location. The localResults field in the SFE object stores these results that has a value for each spatial location.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
localResults(
  x,
  sample_id = "all",
  name = "all",
  features = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  withDimnames = TRUE,
  swap_rownames = NULL,
  ...
)

## S4 replacement method for signature 'SpatialFeatureExperiment'
localResults(
```

```
x,  
sample_id = "all",  
name = "all",  
features = NULL,  
colGeometryName = NULL,  
annotGeometryName = NULL,  
withDimnames = TRUE,  
swap_rownames = NULL,  
...  
) <- value  
  
## S4 method for signature 'SpatialFeatureExperiment'  
localResultNames(x)  
  
## S4 replacement method for signature 'SpatialFeatureExperiment,character'  
localResultNames(x) <- value  
  
## S4 method for signature 'SpatialFeatureExperiment'  
localResultFeatures(  
  x,  
  type = 1L,  
  colGeometryName = NULL,  
  annotGeometryName = NULL,  
  swap_rownames = NULL  
)  
  
## S4 method for signature 'SpatialFeatureExperiment'  
localResultAttrs(  
  x,  
  type = 1L,  
  feature,  
  colGeometryName = NULL,  
  annotGeometryName = NULL,  
  swap_rownames = NULL  
)  
  
## S4 method for signature 'SpatialFeatureExperiment'  
localResult(  
  x,  
  type = 1L,  
  feature,  
  colGeometryName = NULL,  
  annotGeometryName = NULL,  
  sample_id = 1L,  
  withDimnames = TRUE,  
  simplify = TRUE,  
  swap_rownames = NULL  
)
```

```
## S4 replacement method for signature 'SpatialFeatureExperiment'
localResult(
  x,
  type = 1L,
  feature,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  sample_id = 1L,
  withDimnames = TRUE
) <- value
```

### Arguments

x	A SpatialFeatureExperiment object.
sample_id	Sample ID to get or set geometries.
name	Name of the spatial method used, such as "localmoran".
features	Features whose local results to get or set, for localResults getter and setter for multiple features at a time.
colGeometryName	Which colGeometry to get or set local results.
annotGeometryName	Which annotGeometry to get or set local results.
withDimnames	Logical. If TRUE, then the dimnames (colnames or rownames) of the gene count matrix should correspond to row names of the sf data frames of interest.
swap_rownames	Name of a column in rowData to identify features instead of the row names of the SFE object. For example, if the row names of the SFE object are Ensembl IDs and gene symbols are in the "symbol" column in rowData, then putting "symbol" for this argument will use the gene symbols to identify which gene's local results to get or set.
...	Ignored
value	Values to set, should be either a matrix or a data frame.
type	Name or index of the spatial method used, such as "localmoran".
feature	Feature whose local results to get or set, for localResult getter and setter for one feature at a time.
simplify	Basically whether to return the content of the list rather than a list when the list only has one element, such as results for one type and one feature.

### Value

localResults returns a named list each element of which is a set of local results of interest. localResult returns a matrix or a data frame, whichever the original is when it's set. localResultNames returns a character vector. Setters return an SFE object with the desired field set. For genes and colData columns, the local results are stored in the localResults field in int\_colData, whereas for colGeometries and annotGeometries, the local results are stored as columns in the same sf

data frames. `localResultFeatures` returns a character vector of names of features for which local results are available. `localResultAttrs` returns a character vector of the column names of the local results of one type for one feature. It returns NULL if the results are a vector.

## Examples

```
# Toy example
sfe <- readRDS(system.file("extdata/sfe_toy.rds",
  package = "SpatialFeatureExperiment"
))
# localResults functions are written for organizing results from local
# spatial statistics (see the Voyager package). But for the examples here,
# random toy matrices are used. The real results are often matrices, with a
# matrix for each feature.
library(S4Vectors)
set.seed(29)
toy_res1 <- matrix(rnorm(10),
  nrow = 5, ncol = 2,
  dimnames = list(colnames(sfe), c("meow", "purr")))
)
toy_res1b <- matrix(rgamma(10, shape = 2),
  nrow = 5, ncol = 2,
  dimnames = list(colnames(sfe), c("meow", "purr")))
)
toy_df1 <- DataFrame(gene1 = I(toy_res1), gene2 = I(toy_res1b))

toy_res2 <- matrix(rpois(10, lambda = 2),
  nrow = 5, ncol = 2,
  dimnames = list(colnames(sfe), c("sassy", "tortitude")))
)
toy_df2 <- DataFrame(gene1 = I(toy_res2))
# Set all local results
localResults(sfe) <- list(localmoran = toy_df1, Gistar = toy_df2)
# Get all local results
lrs <- localResults(sfe)

# Set results of the same type for multiple genes
localResults(sfe, name = "localmoran") <- toy_df1
# Can also use a list
localResults(sfe, name = "localmoran") <- as.list(toy_df1)
# Get results of the same type for multiple genes
lrs <- localResults(sfe, name = "localmoran", features = c("gene1", "gene2"))

# Set results for one type and one gene
localResult(sfe, "localmoran", feature = "gene1") <- toy_res1
# Get results for one type and one gene
lr <- localResult(sfe, "localmoran", feature = "gene1")

# Set results for a feature in colGeometries
cg_toy <- readRDS(system.file("extdata/cg_toy.rds",
  package = "SpatialFeatureExperiment"
))
colGeometry(sfe, "cg") <- cg_toy
```

```

localResult(sfe, "localmoran",
            feature = "gene1",
            colGeometryName = "cg"
) <- toy_res1
# Get results for a feature in colGeometries
lr <- localResult(sfe, "localmoran", "gene1", colGeometryName = "cg")

```

---

mirrorImg

*Mirror/flip images*


---

## Description

Flip images along the middle horizontal or vertical axis.

## Usage

```

## S4 method for signature 'SpatRasterImage'
mirrorImg(
  x,
  direction = c("vertical", "horizontal"),
  filename = "",
  maxcell = NULL,
  ...
)

## S4 method for signature 'BioFormatsImage'
mirrorImg(x, direction = c("vertical", "horizontal"), ...)

## S4 method for signature 'ExtImage'
mirrorImg(x, direction = c("vertical", "horizontal"), ...)

```

## Arguments

x	SpatRaster or SpatVector
direction	character. Should (partially) match "vertical" to flip by rows, or "horizontal" to flip by columns
filename	character. Output filename
maxcell	Max number of pixels to load SpatRasterImage into memory. The default 1e7 is chosen because this is the approximate number of pixels in the medium resolution image at resolution = 4L in Xenium OME-TIFF to make different methods of this function consistent.
...	additional arguments for writing files as in <a href="#">writeRaster</a>

## Value

\*Image object of the same class.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method,ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

read10xVisiumSFE      *Read 10X Visium data as SpatialFeatureExperiment*

---

**Description**

Read Space Ranger output as a `SpatialFeatureExperiment` object, where spots are represented with polygons in the `colGeometry` called "spotPoly". Other geometries can be added later after the dataset is read. If `data = "filtered"`, then spatial neighborhood graphs of the spots are also computed and stored in the `colGraph` called "visium" in all samples for downstream spatial analyses.

**Usage**

```
read10xVisiumSFE(
  samples = "",
  dirs = file.path(samples, "outs"),
  sample_id = paste0("sample", sprintf("%02d", seq_along(samples))),
  type = c("HDF5", "sparse"),
  data = c("filtered", "raw"),
  images = c("lowres", "hires"),
  unit = c("full_res_image_pixel", "micron"),
  style = "W",
  zero.policy = NULL,
  load = FALSE
)
```

**Arguments**

<code>samples</code>	a character vector specifying one or more directories, each corresponding to a 10x Genomics Visium sample (see Details); if provided, names will be used as sample identifiers
<code>dirs</code>	Directory for each sample that contains the <code>spatial</code> and <code>raw/filtered_features_bc_matrix</code> directories. By default, the <code>outs</code> directory under the directory specified in the <code>samples</code> argument, as in Space Ranger output. Change the <code>dirs</code> argument if you have moved or renamed the output directory.
<code>sample_id</code>	character string specifying unique sample identifiers, one for each directory specified via <code>samples</code> ; ignored if <code>!is.null(names(samples))</code>
<code>type</code>	Either "HDF5", and the matrix will be represented as <code>TENxMatrix</code> , or "sparse", and the matrix will be read as a <code>dgCMatrix</code> .
<code>data</code>	character string specifying whether to read in filtered (spots mapped to tissue) or raw data (all spots).



images	character vector specifying which images to include. Valid values are "lowres", "hires", "fullres", "detected", "aligned"
unit	Whether to use pixels in full resolution image or microns as the unit. If using microns, then spacing between spots in pixels will be used to convert the coordinates into microns, as the spacing is known to be 100 microns. This is used to plot scale bar.
style	style can take values "W", "B", "C", "U", "minmax" and "S"
zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
load	Not used, kept for backward compatibility.

**Value**

A SpatialFeatureExperiment object. The images might need to be manually transposed and/or mirrored to match the spots in this version of this package.

**Note**

The `as(<dgTMatrix>, "dgCMatrix")` is deprecated warning comes from the DropletUtils package which is used by SpatialExperiment to read 10X outputs. This will be fixed when SpatialExperiment switches to TENxIO.

It is assumed that the images have not been cropped. Otherwise the images might not align with the spots.

**Examples**

```
dir <- system.file("extdata", package = "SpatialFeatureExperiment")

sample_ids <- c("sample01", "sample02")
samples <- file.path(dir, sample_ids)

list.files(samples[1])
list.files(file.path(samples[1], "spatial"))
(sfe <- read10xVisiumSFE(samples, sample_id = sample_ids,
  type = "sparse", data = "filtered",
  load = FALSE
))
```

---

readCosMX

*Read CosMX data into SFE*


---

**Description**

This function reads the standard CosMX output into an SFE object, as in "Basic Data Files" on the Nanostring website.

**Usage**

```
readCosMX(
  data_dir,
  z = "all",
  sample_id = "sample01",
  add_molecules = FALSE,
  split_cell_comps = FALSE,
  BPPARAM = SerialParam(),
  file_out = file.path(data_dir, "tx_spots.parquet"),
  z_option = c("3d", "split")
)
```

**Arguments**

<code>data_dir</code>	Top level output directory.
<code>z</code>	Integer z index or "all" to indicate which z-planes to read for the transcript spots.
<code>sample_id</code>	A character sample identifier, which matches the <code>sample_id</code> in <code>imgData</code> . The <code>sample_id</code> will also be stored in a new column in <code>colData</code> , if not already present. Default = <code>sample01</code> .
<code>add_molecules</code>	Logical, whether to add transcripts coordinates to an object.
<code>split_cell_comps</code>	Logical, whether to split transcript spot geometries by cell compartment. Only relevant when <code>'add_molecules = TRUE'</code> .
<code>BPPARAM</code>	A <a href="#">BiocParallelParam</a> object specifying parallel processing backend and number of threads to use for parallelizable tasks: <ol style="list-style-type: none"> <li>1. To load cell segmentation from HDF5 files from different fields of view (FOVs) with multiple cores. A progress bar can be configured in the <a href="#">BiocParallelParam</a> object. When there are numerous FOVs, reading in the geometries can be time consuming, so we recommend using a server and larger number of threads. This argument is not used if <code>use_cellpose = TRUE</code> and the parquet file is present.</li> <li>2. To get the largest piece and see if it's larger than <code>min_area</code> when there are multiple pieces in the cell segmentation for one cell.</li> </ol>
<code>file_out</code>	Name of file to save the geometry or raster to disk. Especially when the geometries are so large that it's unwieldy to load everything into memory. If this file (or directory for multiple files) already exists, then the existing file(s) will be read, skipping the processing. When writing the file, extensions supplied are ignored and extensions are determined based on <code>'dest'</code> .
<code>z_option</code>	What to do with z coordinates. "3d" is to construct 3D geometries. "split" is to create a separate 2D geometry for each z-plane so geometric operations are fully supported but some data wrangling is required to perform 3D analyses. When the z coordinates are not integers, 3D geometries will always be constructed since there are no z-planes to speak of. This argument does not apply when <code>'spatialCoordsNames'</code> has length 2.

**Value**

An SFE object. Cell polygons are written to 'cell\_boundaries\_sf.parquet' in 'data\_dir'. If reading transcript spots ('add\_molecules = TRUE'), then the reformatted transcript spots are saved to file specified in the 'file\_out' argument, which is by default 'tx\_spots.parquet' in the same directory as the rest of the data.

**Examples**

```
fp <- tempdir()
dir_use <- SFEData::CosMXOutput(file_path = file.path(fp, "cosmx_test"))
sfe <- readCosMX(dir_use, z = "all", add_molecules = TRUE)
# Clean up
unlink(dir_use, recursive = TRUE)
```

---

readSelectTx	<i>Read transcript spots of select genes</i>
--------------	--

---

**Description**

I speculate that in practice, the most common use of the transcript spots is visualization, and only a few genes can be visualized at a time or the spots will overcrowd. Then it doesn't make sense to load the transcript spots of all genes into memory as they can take up a lot of memory. The function readSelectTx reads transcript spots of select genes into R, and the function addSelectTx adds them to rowGeometries of the SFE object.

**Usage**

```
readSelectTx(file, gene_select, z = "all", z_option = c("3d", "split"))

addSelectTx(
  sfe,
  file,
  gene_select,
  sample_id = 1L,
  z = "all",
  z_option = c("3d", "split"),
  swap_rownames = NULL
)
```

**Arguments**

file	File path of a GeoParquet file (e.g. already reformatted with the <a href="#">formatTxSpots</a> or <a href="#">addTxSpots</a> function, should have already flipped to match image if necessary).
gene_select	Character vector of a subset of genes. If NULL, then all genes that have transcript spots are added. Only relevant when reading data from formatted files on disk. If specified, then return = TRUE.

z	Index of z plane to read. Can be "all" to read all z-planes into MULTIPOINT geometries with XYZ coordinates. If z values are not integer, then spots with all z values will be read.
z_option	What to do with z coordinates. "3d" is to construct 3D geometries. "split" is to create a separate 2D geometry for each z-plane so geometric operations are fully supported but some data wrangling is required to perform 3D analyses. When the z coordinates are not integers, 3D geometries will always be constructed since there are no z-planes to speak of. This argument does not apply when 'spatialCoordsNames' has length 2.
sfe	A 'SpatialFeatureExperiment' object.
sample_id	Which sample in the SFE object the transcript spots should be added to.
swap_rownames	Name of a column in rowData(sfe) to use as gene identifiers in place of the actual row names. In some cases this may be needed to match each transcript spot MULTIPOINT geometry to rows of sfe.

### Value

When there are multiple parquet files to be read, a list of sf data frames with MULTIPOINT geometry for genes selected. When there is only one file, then one sf data frame. For addSelectTx, an SFE object with the transcript spots of the selected genes added.

### Note

The GDAL Parquet driver is required for this function, though not for other functions that work with GeoParquet files. GDAL Parquet driver has been supported since GDAL 3.5.0, but is not part of the default installation. The z and z\_option arguments are there since the file names contain z-plane information when relevant. See the [GDAL documentation page for the Parquet driver](#).

### Examples

```
library(SFEData)
if (gdalParquetAvailable()) {
  fp <- tempdir()
  dir_use <- XeniumOutput("v2", file_path = file.path(fp, "xenium_test"))
  fn_tx <- formatTxTech(dir_use, tech = "Xenium", flip = TRUE, return = FALSE,
                       file_out = file.path(dir_use, "tx_spots.parquet"))
  gene_select <- c("ACE2", "BMX")
  df <- readSelectTx(fn_tx, gene_select)

  sfe <- readXenium(dir_use)
  sfe <- addSelectTx(sfe, fn_tx, head(rownames(sfe), 5), swap_rownames = "Symbol")
  unlink(dir_use, recursive = TRUE)
}
```

readVizgen

*Read Vizgen MERFISH output as SpatialFeatureExperiment*

## Description

This function reads the standard Vizgen MERFISH output into an SFE object. The coordinates are in microns. Cell centroids are read into `colGeometry` "centroids", and cell segmentations are read into `colGeometry` "cellSeg". The image(s) (polyT, DAPI, and cell boundaries) are also read as `SpatRaster` objects so they are not loaded into memory unless necessary. Because the image's origin is the top left while the geometry's origin is bottom left, either the image or the geometry needs to be flipped. Because the image accompanying MERFISH datasets are usually very large, the coordinates will be flipped so the flipping operation won't load the entire image into memory. Large datasets with hundreds of thousands of cells can take a while to read if reading transcript spots as it takes a while to convert the spots to MULTIPOINT geometries.

## Usage

```
readVizgen(
  data_dir,
  z = "all",
  sample_id = "sample01",
  min_area = 15,
  image = c("DAPI", "PolyT", "Cellbound"),
  flip = c("geometry", "image", "none"),
  max_flip = "50 MB",
  filter_counts = FALSE,
  add_molecules = FALSE,
  use_bboxes = FALSE,
  use_cellpose = TRUE,
  BPPARAM = SerialParam(),
  file_out = file.path(data_dir, "detected_transcripts.parquet"),
  z_option = c("3d", "split")
)
```

## Arguments

<code>data_dir</code>	Top level output directory.
<code>z</code>	Integer, z index to read, or "all", indicating z-planes of the images and transcript spots to read. While cell segmentation seems to have multiple z-planes, the segmentation in all z-planes are the same so in effect the cell segmentation is only in 2D.
<code>sample_id</code>	A character sample identifier, which matches the <code>sample_id</code> in <code>imgData</code> . The <code>sample_id</code> will also be stored in a new column in <code>colData</code> , if not already present. Default = <code>sample01</code> .
<code>min_area</code>	Minimum cell area in square microns. Anything smaller will be considered artifact or debris and removed.

image	Which image(s) to load, can be "DAPI", "PolyT", "Cellbound" or any combination of them.
flip	To flip the image, geometry coordinates, or none. Because the image has the origin at the top left while the geometry has origin at the bottom left, one of them needs to be flipped for them to match. If one of them is already flipped, then use "none". The image will not be flipped if it's GeoTIFF.
max_flip	Maximum size of the image allowed to flip the image. Because the image will be loaded into memory to be flipped. If the image is larger than this size then the coordinates will be flipped instead.
filter_counts	Logical, whether to keep cells with counts > 0.
add_molecules	Logical, whether to add transcripts coordinates to an object.
use_bboxes	If no segmentation output is present, use cell_metadata to make bounding boxes instead.
use_cellpose	Whether to read the parquet files from CellPose cell segmentation. If FALSE, cell segmentation will be read from the HDF5 files. Note that reading HDF5 files for numerous FOVs is very slow.
BPPARAM	A <a href="#">BiocParallelParam</a> object specifying parallel processing backend and number of threads to use for parallelizable tasks: <ol style="list-style-type: none"> <li>1. To load cell segmentation from HDF5 files from different fields of view (FOVs) with multiple cores. A progress bar can be configured in the <a href="#">BiocParallelParam</a> object. When there are numerous FOVs, reading in the geometries can be time consuming, so we recommend using a server and larger number of threads. This argument is not used if use_cellpose = TRUE and the parquet file is present.</li> <li>2. To get the largest piece and see if it's larger than min_area when there are multiple pieces in the cell segmentation for one cell.</li> </ol>
file_out	Name of file to save the geometry or raster to disk. Especially when the geometries are so large that it's unwieldy to load everything into memory. If this file (or directory for multiple files) already exists, then the existing file(s) will be read, skipping the processing. When writing the file, extensions supplied are ignored and extensions are determined based on 'dest'.
z_option	What to do with z coordinates. "3d" is to construct 3D geometries. "split" is to create a separate 2D geometry for each z-plane so geometric operations are fully supported but some data wrangling is required to perform 3D analyses. When the z coordinates are not integers, 3D geometries will always be constructed since there are no z-planes to speak of. This argument does not apply when 'spatialCoordsNames' has length 2.

**Value**

A SpatialFeatureExperiment object.

**Note**

Since the transcript spots file is often very large, we recommend only using add\_molecules = TRUE on servers with a lot of memory. If reading all z-planes, conversion of transcript spot geometry to

parquet file might fail due to arrow data length limit. In a future version, when the transcript spot geometry is large, it will be written to multiple separate parquet files which are then concatenated with DuckDB. Also, in a future version, the transcript spot processing function might be rewritten in C++ to stream the original CSV file so it's not entirely loaded into memory.

### Examples

```
fp <- tempdir()
dir_use <- SFEData::VizgenOutput(file_path = file.path(fp, "vizgen_test"))
sfe <- readVizgen(dir_use, z = 3L, image = "PolyT",
flip = "geometry")

## Filtering of counts, and addition of molecule coordinates..
sfe <- readVizgen(dir_use, z = 3L, image = "PolyT", filter_counts = TRUE,
add_molecules = TRUE, flip = "geometry")

unlink(dir_use, recursive = TRUE)
```

---

readXenium

*Read 10X Xenium output as SpatialFeatureExperiment*


---

### Description

This function reads the standard 10X Xenium output into an SFE object.

### Usage

```
readXenium(
  data_dir,
  sample_id = "sample01",
  image = c("morphology_focus", "morphology_mip"),
  segmentations = c("cell", "nucleus"),
  row.names = c("id", "symbol"),
  flip = c("geometry", "image", "none"),
  max_flip = "50 MB",
  filter_counts = FALSE,
  add_molecules = FALSE,
  min_phred = 20,
  BPPARAM = SerialParam(),
  file_out = file.path(data_dir, "tx_spots.parquet")
)
```

### Arguments

data_dir	Top level output directory.
sample_id	A character sample identifier, which matches the sample_id in <code>imgData</code> . The sample_id will also be stored in a new column in <code>colData</code> , if not already present. Default = sample01.

image	Which image(s) to load, can be "morphology_mip", "morphology_focus" or both. Note that in Xenium Onboarding Analysis (XOA) v2, there is no longer "morphology_mip" and "morphology_focus" is a directory with 4 images corresponding to 4 channels: DAPI, "Cadherin", 18S, and Vimentin. So this argument is ignored for XOA v2.
segmentations	Which segmentation outputs to read, can be "cell", "nucleus", or both.
row.names	String specifying whether to use Ensembl IDs ("id") or gene symbols ("symbol") as row names. If using symbols, the Ensembl ID will be appended to disambiguate in case the same symbol corresponds to multiple Ensembl IDs. Always "symbol" if 'add_molecules = TRUE' because only gene symbols are used in the transcript spot files.
flip	To flip the image, geometry coordinates, or none. Because the image has the origin at the top left while the geometry has origin at the bottom left, one of them needs to be flipped for them to match. If one of them is already flipped, then use "none". The image will not be flipped if it's GeoTIFF.
max_flip	Maximum size of the image allowed to flip the image. Because the image will be loaded into memory to be flipped. If the image is larger than this size then the coordinates will be flipped instead.
filter_counts	Logical, whether to keep cells with counts > 0.
add_molecules	Logical, whether to add transcripts coordinates to an object.
min_phred	Minimum Phred score to keep spot. By default 20, the conventional threshold indicating "acceptable", meaning that there's 1 chance that the spot was decoded in error.
BPPARAM	A <a href="#">BiocParallelParam</a> object specifying parallel processing backend and number of threads to use for parallelizable tasks: <ol style="list-style-type: none"> <li>1. To load cell segmentation from HDF5 files from different fields of view (FOVs) with multiple cores. A progress bar can be configured in the <a href="#">BiocParallelParam</a> object. When there are numerous FOVs, reading in the geometries can be time consuming, so we recommend using a server and larger number of threads. This argument is not used if use_cellpose = TRUE and the parquet file is present.</li> <li>2. To get the largest piece and see if it's larger than min_area when there are multiple pieces in the cell segmentation for one cell.</li> </ol>
file_out	Name of file to save the geometry or raster to disk. Especially when the geometries are so large that it's unwieldy to load everything into memory. If this file (or directory for multiple files) already exists, then the existing file(s) will be read, skipping the processing. When writing the file, extensions supplied are ignored and extensions are determined based on 'dest'.

### Value

An SFE object. If reading segmentations, the cell or nuclei segmentation will be saved to 'cell\_boundaries\_sf.parquet' and 'nucleus\_boundaries\_sf.parquet' respectively in 'data.dir' so next time the boundaries can be read much more quickly. If reading transcript spots ('add\_molecules = TRUE'), then the reformatted transcript spots are saved to file specified in the 'file\_out' argument, which is by default



'tx\_spots.parquet' in the same directory as the rest of the data. If images are present, then the images will be of the `BioFormatsImage` class and not loaded into memory until necessary in later operations.

### Note

Sometimes when reading images, you will see this error the first time: 'java.lang.NullPointerException: Cannot invoke "loci.formats.DimensionSwapper.setMetadataFiltered(boolean)" because "RBioFormats.reader" is null'. Rerun the code and it should work the second time.

### Examples

```
library(SFEData)
library(RBioFormats)
fp <- tempdir()
dir_use <- XeniumOutput("v2", file_path = file.path(fp, "xenium_test"))
# RBioFormats issue
try(sfe <- readXenium(dir_use, add_molecules = TRUE))
sfe <- readXenium(dir_use, add_molecules = TRUE)
unlink(dir_use, recursive = TRUE)
```

---

reexports

*Functions re-exported from other packages*

---

### Description

These are some commonly used getters and setters of classes that SFE inherits so you don't have to separately attach those packages to use these functions.

### Usage

```
colData(x, ...)

rowData(x, use.names = TRUE, ...)

colData(x, ...) <- value

spatialCoords(x, ...)

spatialCoords(x) <- value

spatialCoordsNames(x)

getImg(x, ...)

imgData(x)

rmvImg(x, ...)
```

```
counts(object, ...)
```

```
logcounts(object, ...)
```

```
reducedDim(x, type, ...)
```

### Arguments

x	A SummarizedExperiment object.
...	For assay, arguments in ... are forwarded to assays. For rbind, cbind, ... contains SummarizedExperiment objects to be combined. For other accessors, ignored.
use.names	For rowData: Like <code>mcols(x)</code> , by default <code>rowData(x)</code> propagates the rownames of x to the returned <code>DataFrame</code> object (note that for a SummarizedExperiment object, the rownames are also the names i.e. <code>rownames(x)</code> is always the same as <code>names(x)</code> ). Setting <code>use.names=FALSE</code> suppresses this propagation i.e. it returns a <code>DataFrame</code> object with no rownames. Use this when <code>rowData(x)</code> fails, which can happen when the rownames contain NAs (because the rownames of a SummarizedExperiment object can contain NAs, but the rownames of a <code>DataFrame</code> object cannot). For combineRows and combineCols: See Combining section below.
value	An object of a class specified in the S4 method signature or as outlined in ‘Details’.
object	A SingleCellExperiment object, which includes SFE.
type	Name or numeric index to indicate which reducedDim to get, such as "PCA". By default the first item in reducedDims.

---

removeEmptySpace	<i>Remove empty space</i>
------------------	---------------------------

---

### Description

For each sample independently, all geometries and `spatialCoords` are translated so the origin is at the minimum coordinates of the bounding box of all geometries of the sample. This way coordinates of different samples will be more comparable. This removes empty space in the images if present.

### Usage

```
removeEmptySpace(sfe, sample_id = "all")
```

### Arguments

sfe	An SFE object.
sample_id	Sample to remove empty space.

**Value**

An SFE object with empty space removed.

**Note**

Unlike other functions in this package, this function operates on all samples by default.

**Examples**

```
library(SFEData)
library(SingleCellExperiment)
sfe <- McKellarMuscleData("full")
# Only keep spots on tissue
sfe <- sfe[, colData(sfe)$in_tissue]
# Move the coordinates of the tissue
sfe <- removeEmptySpace(sfe)
```

---

 rotateImg

*Rotate image*


---

**Description**

As in SpatialExperiment, rotation here must be a multiple of 90 degrees.

**Usage**

```
## S4 method for signature 'SpatRasterImage'
rotateImg(x, degrees, maxcell = 1e+07, ...)

## S4 method for signature 'BioFormatsImage'
rotateImg(x, degrees, ...)

## S4 method for signature 'ExtImage'
rotateImg(x, degrees, ...)
```

**Arguments**

x	An object of class *Image as implemented in this package.
degrees	How many degrees to rotate. Positive number means clockwise and negative number means counterclockwise.
maxcell	Max number of pixels to load SpatRasterImage into memory. The default 1e7 is chosen because this is the approximate number of pixels in the medium resolution image at resolution = 4L in Xenium OME-TIFF to make different methods of this function consistent.
...	Ignored. It's there so different methods can all be passed to the same lapply in the method for SFE objects. Some methods have extra arguments.

**Value**

SpatRasterImage will be loaded into memory and converted to ExtImage. Otherwise \*Image object of the same class.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

rowGeometries

*Row geometry getters and setters*


---

**Description**

rowGeometries are geometries that corresponding to rows of the gene count matrix, such as sm-FISH transcript spots. The txSpots() function is a convenience wrapper for transcript spots, although this entirely depends on the rowGeometry being named txSpots.

**Usage**

```
rowGeometry(x, type = 1L, sample_id = 1L, withDimnames = TRUE)
```

```
rowGeometry(
  x,
  type = 1L,
  sample_id = 1L,
  withDimnames = TRUE,
  partial = FALSE,
  translate = TRUE
) <- value
```

```
rowGeometries(x, sample_id = "all", withDimnames = TRUE)
```

```
rowGeometries(
  x,
  sample_id = "all",
  withDimnames = TRUE,
  partial = FALSE,
  translate = TRUE
) <- value
```

```
rowGeometryNames(x)
```

```
rowGeometryNames(x) <- value
```

```
txSpots(x, sample_id = 1L, withDimnames = TRUE)
```

```
txSpots(
  x,
  sample_id = 1L,
  withDimnames = TRUE,
  partial = FALSE,
  translate = TRUE
) <- value
```

### Arguments

x	A SpatialFeatureExperiment object.
type	An integer specifying the index or string specifying the name of the *Geometry to query or replace. If missing, then the first item in the *Geometries will be returned or replaced.
sample_id	Sample ID to get or set geometries.
withDimnames	Logical. If TRUE, then the dimnames (colnames or rownames) of the gene count matrix should correspond to row names of the sf data frames of interest.
partial	In setters, if a rowGeometry of the same name exists, whether to only replace the rows present in value.
translate	Logical. Only used if <a href="#">removeEmptySpace</a> has been run of the SFE object. If that's the case, this argument indicates whether the new value to be assigned to the geometry is in the coordinates prior to removal of empty space so it should be translated to match the new coordinates after removing empty space. Default to TRUE.
value	Value to set. For dimGeometry, must be a sf data frame with the same number of rows as size in the dimension of interest, or an ordinary data frame that can be converted to such a sf data frame (see <a href="#">df2sf</a> ). For dimGeometries, must be a list of such sf or ordinary data frames.

### Details

When there are multiple samples in the SFE object, rowGeometries for each sample has the sample\_id appended to the name of the geometry. For example, if the name is txSpots and the sample ID is sample01, then the actual name of the rowGeometry is txSpots\_sample01. In the getter, one can still specify rowGeometry(sfe, "txSpots", sample\_id = "sample01").

Appending the sample\_id is unnecessary when there is only one sample, but sample\_id will be appended when to SFE objects are combined with cbind. It is necessary to distinguish between different samples because they can have overlapping coordinate values.

### See Also

[dimGeometries()], [colGeometries()]

### Examples

```
library(SFEData)
library(RBioFormats)
```

```

fp <- tempdir()
dir_use <- XeniumOutput("v2", file_path = file.path(fp, "xenium_test"))
# RBioFormats issue
try(sfe <- readXenium(dir_use, add_molecules = TRUE))
sfe <- readXenium(dir_use, add_molecules = TRUE)
rowGeometries(sfe)
rowGeometryNames(sfe)
tx <- rowGeometry(sfe, "txSpots")
txSpots(sfe)
unlink(dir_use, recursive = TRUE)

```

---

sampleIDs	<i>Get all unique sample IDs</i>
-----------	----------------------------------

---

### Description

The title is self-explanatory.

### Usage

```
sampleIDs(sfe)
```

### Arguments

sfe                    A SpatialFeatureExperiment object.

### Value

A character vector of all unique entries of the sample\_id column in colData(x).

### Examples

```

library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
sampleIDs(sfe)

```

---

saveRDS,SpatialFeatureExperiment-method	<i>Save SpatialFeatureExperiment as RDS file</i>
---	--

---

### Description

Saving SFE objects as RDS files is complicated by the SpatRaster class of the images. If present, the images need to be wrapped with the [wrap](#) function in terra before serializing the SFE object. Otherwise the images will be invalid pointers when the RDS is reloaded. If the image does not fit in memory and its file source is unknown, then it will be written to a temporary file, which is reloaded when the RDS file is loaded. When an SFE object with images is read from an RDS file, the images will not be unwrapped until necessary.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
saveRDS(
  object,
  file = "",
  ascii = FALSE,
  version = NULL,
  compress = TRUE,
  refhook = NULL
)
```

**Arguments**

object	A SpatialFeatureExperiment object.
file	a <a href="#">connection</a> or the name of the file where the R object is saved to or read from.
ascii	a logical. If TRUE or NA, an ASCII representation is written; otherwise (default), a binary one is used. See the comments in the help for <a href="#">save</a> .
version	the workspace format version to use. NULL specifies the current default version (3). The only other supported value is 2, the default from R 1.4.0 to R 3.5.0.
compress	a logical specifying whether saving to a named file is to use "gzip" compression, or one of "gzip", "bzip2" or "xz" to indicate the type of compression to be used. Ignored if file is a connection.
refhook	a hook function for handling reference objects.

**Value**

Invisibly NULL.

**Examples**

```
outdir <- system.file("extdata", package = "SpatialFeatureExperiment")
samples <- file.path(outdir, paste0("sample0", 1:2))
sfe <- read10xVisiumSFE(samples, type = "sparse", data = "filtered")
saveRDS(sfe, "foo.rds")
# Clean up
file.remove("foo.rds")
```

---

scaleImg

*Scale image*

---

**Description**

This function scales the image about its center. After scaling, the center of the image is not shifted.

**Usage**

```
## S4 method for signature 'AlignedSpatialImage'
scaleImg(x, factor, ...)
```

**Arguments**

x                    An object of class \*Image as implemented in this package.

factor                Numeric, scaling factor.

...                   Ignored. It's there so different methods can all be passed to the same lapply in the method for SFE objects. Some methods have extra arguments.

**Value**

A \*Image object of the same class that has been scaled. Behind the scene, it's only the extent that has been changed and the images are not changed. The center of the image is unchanged.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

---

SFE-image

*Methods for handling image-related data*

---

**Description**

Generics of these functions are defined in SpatialExperiment, except for transposeImg. These SFE methods cater to the new image-related classes in SFE. The SPE method for getImg, rmvImg, and imgRaster don't need to be modified for SFE and are hence not implemented here, but are simply re-exported.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
addImg(
  x,
  imageSource,
  sample_id = 1L,
  image_id,
  extent = NULL,
  scale_fct = 1,
  file = deprecated()
)

## S4 method for signature 'SpatialFeatureExperiment'
transposeImg(
```



```

    x,
    sample_id = 1L,
    image_id = NULL,
    maxcell = 1e+07,
    filename = ""
)

## S4 method for signature 'SpatialFeatureExperiment'
mirrorImg(
  x,
  sample_id = 1L,
  image_id = NULL,
  direction = "vertical",
  maxcell = 1e+07,
  filename = ""
)

## S4 method for signature 'SpatialFeatureExperiment'
rotateImg(x, sample_id = 1L, image_id = NULL, degrees, maxcell = 1e+07)

## S4 method for signature 'SpatialFeatureExperiment'
translateImg(x, sample_id = 1L, image_id = NULL, v)

## S4 method for signature 'SpatialFeatureExperiment'
scaleImg(x, sample_id = 1L, image_id = NULL, factor)

## S4 method for signature 'SpatialFeatureExperiment'
affineImg(x, sample_id = 1L, image_id = NULL, M, v)

```

## Arguments

x	A SFE object.
imageSource	a character string specifying an image file name (.png, .jpg or .tif) or URL to source the image from
sample_id	Which sample the image is associated with. Use <a href="#">sampleIDs</a> to get sample IDs present in the SFE object.
image_id	Image ID, such as "lowres" and "hires" for Visium data and "DAPI" and "PolyT" for Vizgen MERFISH data.
extent	A numeric vector of length 4 with names of the set xmin, ymin, xmax, and ymax, specifying the extent of the image.
scale_fct	Scale factor – multiply pixel coordinates in full resolution image by this scale factor should yield pixel coordinates in a different resolution. extent takes precedence over scale_fct.
file	File from which to read the image.
maxcell	Max number of pixels to load SpatRasterImage into memory. The default 1e7 is chosen because this is the approximate number of pixels in the medium

	resolution image at resolution = 4L in Xenium OME-TIFF to make different methods of this function consistent.
filename	character. Output filename
direction	character. Should (partially) match "vertical" to flip by rows, or "horizontal" to flip by columns
degrees	How many degrees to rotate. Positive number means clockwise and negative number means counterclockwise.
v	A numeric vector of length 2 specifying the vector in the xy plane to translate the SFE object.
factor	Numeric, scaling factor.
M	A 2x2 numeric matrix for the linear transformation in the xy plane.

### Details

Method of [transposeImg](#), [mirrorImg](#), and [rotateImg](#) perform the method on all images within the SFE object that are specified with `sample_id` and `image_id`. For images that are not loaded into memory, `rotateImg` will load `SpatRasterImage` into memory and all image operations except `translate` will load `BioFormatsImage` into memory.

### Note

If the image is already a GeoTIFF file that already has an extent, then the extent associated with the file will be honored and the `extent` and `scale_fct` arguments are ignored. Transposing the image is just like transposing a matrix. It's flipped about the line going from the top left to the bottom right.

### See Also

Other image methods: [affineImg\(\)](#), [cropImg\(\)](#), [dim](#), [BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#), [transposeImg\(\)](#)

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData("small")
img_path <- system.file(file.path("extdata", "sample01", "outs", "spatial",
  "tissue_lowres_image.png"), package = "SpatialFeatureExperiment")
sfe <- addImg(sfe, img_path, sample_id = "Vis5A", image_id = "lowres", scale_fct =
  0.023)
img <- getImg(sfe)
# SpatRasterImage method
img_t <- transposeImg(img)
# SFE method
sfe <- transposeImg(sfe, sample_id = "Vis5A", image_id = "lowres")
```

SFE-transform

*Affine transform of SFE object in histological space***Description**

These functions perform affine transformations on SFE objects, including all geometries and images. The transformation is performed on each sample as a whole. This differs from functions such as `mirrorImg` in that `mirrorImg` and `rotateImg` transform the image with the center at the center of the image itself. In contrast, the center of transformation here is the center of the bounding box of the entire sample, including images.

**Usage**

```
transpose(sfe, sample_id = "all", maxcell = NULL, filename = "")
```

```
mirror(
  sfe,
  sample_id = "all",
  direction = c("vertical", "horizontal"),
  maxcell = NULL,
  filename = ""
)
```

```
rotate(sfe, sample_id = "all", degrees, maxcell = 1e+07)
```

```
translate(sfe, sample_id = "all", v)
```

```
scale(sfe, sample_id = "all", factor)
```

```
affine(sfe, sample_id = "all", M, v, maxcell = 1e+07)
```

**Arguments**

<code>sfe</code>	An SFE object.
<code>sample_id</code>	Sample(s) to transform.
<code>maxcell</code>	Rotating <code>SpatRasterImage</code> will convert it into <code>ExtImage</code> , loading the image into memory. This argument specifies the maximum number of pixels in the image loaded into memory. The image will be down sampled to approximately this number of pixels.
<code>filename</code>	character. Output filename
<code>direction</code>	character. Should (partially) match "vertical" to flip by rows, or "horizontal" to flip by columns
<code>degrees</code>	How many degrees to rotate. Positive number means clockwise and negative number means counterclockwise.
<code>v</code>	Vector to spatially translate the SFE object.

factor            Numeric, scaling factor.  
 M                A 2x2 numeric matrix for the linear transformation in the xy plane.

### Details

For images that are not loaded into memory, rotateImg will load SpatRasterImage into memory and all image operations except translate will load BioFormatsImage into memory.

### Value

An SFE object with the sample(s) transformed.

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData("small")
sfe2 <- transpose(sfe)
sfe3 <- mirror(sfe)
```

---

show,SpatialFeatureExperiment-method

*Print method for SpatialFeatureExperiment*

---

### Description

Printing summaries of colGeometries, rowGeometries, and annotGeometries in addition to what's shown for SpatialExperiment. Geometry names and types are printed.

### Usage

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

### Arguments

object            A SpatialFeatureExperiment object.

### Value

None (invisible NULL).

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
sfe # The show method is implicitly called
```

---

 SpatialFeatureExperiment

*Constructor of SpatialFeatureExperiment object*


---

## Description

Create a SpatialFeatureExperiment object.

## Usage

```
SpatialFeatureExperiment(
  assays,
  colData = DataFrame(),
  rowData = NULL,
  sample_id = "sample01",
  spatialCoordsNames = c("x", "y"),
  spatialCoords = NULL,
  colGeometries = NULL,
  rowGeometries = NULL,
  annotGeometries = NULL,
  spotDiameter = NA_real_,
  annotGeometryType = "POLYGON",
  spatialGraphs = NULL,
  unit = c("full_res_image_pixel", "micron"),
  BPPARAM = deprecated(),
  ...
)
```

## Arguments

assays	A list or SimpleList of matrix-like elements, or a matrix-like object (e.g. an ordinary matrix, a data frame, a <a href="#">DataFrame</a> object from the <b>S4Vectors</b> package, a <a href="#">sparseMatrix</a> derivative from the <b>Matrix</b> package, a <a href="#">DelayedMatrix</a> object from the <b>DelayedArray</b> package, etc...). All elements of the list must have the same dimensions, and dimension names (if present) must be consistent across elements and with the row names of rowRanges and colData.
colData	An optional <a href="#">DataFrame</a> describing the samples. Row names, if present, become the column names of the RangedSummarizedExperiment.
rowData	A <a href="#">DataFrame</a> object describing the rows. Row names, if present, become the row names of the SummarizedExperiment object. The number of rows of the <a href="#">DataFrame</a> must equal the number of rows of the matrices in assays.
sample_id	A character sample identifier, which matches the sample_id in <a href="#">imgData</a> . The sample_id will also be stored in a new column in <a href="#">colData</a> , if not already present. Default = sample01.

**spatialCoordsNames**

A character vector of column names if `*Geometries` arguments have ordinary data frames, to identify the columns in the ordinary data frames that specify the spatial coordinates. If `colGeometries` is not specified, then this argument will behave as in [SpatialExperiment](#), but `colGeometries` will be given precedence if provided.

**spatialCoords**

A numeric matrix containing columns of spatial coordinates, as in [SpatialExperiment](#). The coordinates are centroids of the entities represented by the columns of the gene count matrix. If `colGeometries` is also specified, then it will be given priority and a warning is issued. Otherwise, the `sf` representation of the centroids will be stored in the `colGeometry` called `centroids`.

**colGeometries**

Geometry of the entities that correspond to the columns of the gene count matrix, such as cells and Visium spots. It must be a named list of one of the following:

**An `sf` data frame** The geometry column specifies the geometry of the entities.

**An ordinary data frame specifying centroids** Column names for the coordinates are specified in the `spatialCoordsNames` argument. For Visium and ST, in addition to the centroid coordinate data frame, the spot diameter in the same unit as the coordinates can be specified in the `spotDiameter` argument.

**An ordinary data frame specifying polygons** Also use `spatialCoordsNames`. There should an additional column "ID" to specify which vertices belong to which polygon. The coordinates should not be in list columns. Rather, the data frame should look like it is passed to `ggplot2::geom_polygon`. If there are holes, then there must also be a column "subID" that differentiates between the outer polygon and the holes.

In all cases, the data frame should specify the same number of geometries as the number of columns in the gene count matrix. If the column "barcode" is present, then it will be matched to column names of the gene count matrix. Otherwise, the geometries are assumed to be in the same order as columns in the gene count matrix. If the geometries are specified in an ordinary data frame, then it will be converted into `sf` internally. Named list of data frames because each entity can have multiple geometries, such as whole cell and nuclei segmentations. The geometries are assumed to be POINTs for centroids and POLYGONs for segmentations. If polygons are specified in an ordinary data frame, then anything with fewer than 3 vertices will be removed. For anything other than POINTs, attributes of the geometry will be ignored.

**rowGeometries**

Geometry associated with genes or features, which correspond to rows of the gene count matrix.

**annotGeometries**

Geometry of entities that do not correspond to columns or rows of the gene count matrix, such as tissue boundary and pathologist annotations of histological regions, and nuclei segmentation in a Visium dataset. Also a named list as in `colGeometries`. The ordinary data frame may specify POINTs, POLYGONs, or LINESTRINGs, or their MULTI versions. Each data frame can only specify one type of geometry. For MULTI versions, there must be a column "group" to identify each MULTI geometry.

spotDiameter	Spot diameter for technologies with arrays of spots of fixed diameter per slide, such as Visium, ST, DBiT-seq, and slide-seq. The diameter must be in the same unit as the coordinates in the *Geometry arguments. Ignored for geometries that are not POINT or MULTIPOINT.
annotGeometryType	Character vector specifying geometry type of each element of the list if annotGeometry is specified. Each element of the vector must be one of POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, and MULTIPOLYGON. Must be either length 1 (same for all elements of the list) or the same length as the list. Ignored if the corresponding element is an sf object.
spatialGraphs	A named list of listw objects (see spdep) for spatial neighborhood graphs.
unit	Unit the coordinates are in, either microns or pixels in full resolution image.
BPPARAM	Deprecated. The 'sfheaders' package is used in df2sf for much better performance.
...	Additional arguments passed to the <a href="#">SpatialExperiment</a> and <a href="#">SingleCellExperiment</a> constructors.

### Value

A SFE object. If neither colGeometries nor spotDiameter is specified, then a colGeometry called "centroids" will be made, which is essentially the spatial coordinates as sf POINTs. If spotDiameter is specified, but not colGeometries, then the spatial coordinates will be buffered by half the diameter to get spots with the desired diameter, and the resulting colGeometry will be called "spotPoly", for which there's a convenience getter and setter, [spotPoly](#).

### Examples

```
library(Matrix)
data("visium_row_col")
coords1 <- visium_row_col[visium_row_col$col < 6 & visium_row_col$row < 6, ]
coords1$row <- coords1$row * sqrt(3)
cg <- df2sf(coords1[, c("col", "row")], c("col", "row"), spotDiameter = 0.7)

set.seed(29)
col_inds <- sample(seq_len(13), 13)
row_inds <- sample(seq_len(5), 13, replace = TRUE)
values <- sample(seq_len(5), 13, replace = TRUE)
mat <- sparseMatrix(i = row_inds, j = col_inds, x = values)
colnames(mat) <- coords1$barcode
rownames(mat) <- sample(LETTERS, 5)
rownames(cg) <- colnames(mat)

sfe <- SpatialFeatureExperiment(list(counts = mat),
  colData = coords1,
  spatialCoordsNames = c("col", "row"),
  spotDiameter = 0.7
)
sfe2 <- SpatialFeatureExperiment(list(counts = mat),
  colGeometries = list(foo = cg)
)
)
```

---

SpatialFeatureExperiment-class

*The SpatialFeatureExperiment class*

---

### Description

This class inherits from the [SpatialExperiment](#) (SPE) class, which in turn inherits from [SingleCellExperiment](#) (SCE). `SpatialFeatureExperiment` stores geometries of spots or cells in `sf` objects which form columns of a `DataFrame` which is in turn a column of the `int_colData` `DataFrame` of the underlying SCE object, just like `reducedDim` in SCE. Geometries of the tissue outline, pathologist annotations, and objects (e.g. nuclei segmentation in a Visium dataset) are stored in `sf` objects in a named list called `annotGeometries` in `int_metadata`.

---

SpatialFeatureExperiment-coercion

*SpatialFeatureExperiment coercion methods*

---

### Description

The `SpatialFeatureExperiment` class inherits from `SpatialExperiment`, which in turn inherits from `SingleCellExperiment`. A `SpatialExperiment` object with geometries in `colGeometries` in the `int_colData`, `rowGeometries` in the `int_elementMetadata`, or `annotGeometries` in the `int_metadata` can be directly converted to `SpatialFeatureExperiment` with `as(spe, "SpatialFeatureExperiment")`. A `SpatialExperiment` object without the geometries can also be converted; the coordinates in the `spatialCoords` field will be used to make POINT geometries named "centroids" to add to `colGeometries`. The geometries can also be supplied separately when using `toSpatialFeatureExperiment`. Images are converted to `SpatRaster`.

### Usage

```
## S4 method for signature 'SpatialExperiment'
toSpatialFeatureExperiment(
  x,
  colGeometries = NULL,
  rowGeometries = NULL,
  annotGeometries = NULL,
  spatialCoordsNames = c("x", "y"),
  annotGeometryType = "POLYGON",
  spatialGraphs = NULL,
  spotDiameter = NA,
  unit = NULL,
  BPPARAM = deprecated()
)

## S4 method for signature 'SingleCellExperiment'
```



```

toSpatialFeatureExperiment(
  x,
  sample_id = "sample01",
  spatialCoordsNames = c("x", "y"),
  spatialCoords = NULL,
  colGeometries = NULL,
  rowGeometries = NULL,
  annotGeometries = NULL,
  annotGeometryType = "POLYGON",
  spatialGraphs = NULL,
  spotDiameter = NA,
  scaleFactors = 1,
  imageSources = NULL,
  image_id = NULL,
  loadImage = TRUE,
  imgData = NULL,
  unit = NULL,
  BPPARAM = deprecated()
)

## S4 method for signature 'Seurat'
toSpatialFeatureExperiment(
  x,
  add_molecules = TRUE,
  flip = c("geometry", "image", "none"),
  image_scalefactors = c("lowres", "hires"),
  unit = NULL,
  BPPARAM = SerialParam()
)

```

## Arguments

- |               |   |
|---------------|---|
| x             | A SpatialExperiment or Seurat object to be coerced to a SpatialFeatureExperiment object.  |
| colGeometries | <p>Geometry of the entities that correspond to the columns of the gene count matrix, such as cells and Visium spots. It must be a named list of one of the following:</p> <p><b>An sf data frame</b> The geometry column specifies the geometry of the entities.</p> <p><b>An ordinary data frame specifying centroids</b> Column names for the coordinates are specified in the spatialCoordsNames argument. For Visium and ST, in addition to the centroid coordinate data frame, the spot diameter in the same unit as the coordinates can be specified in the spotDiameter argument.</p> <p><b>An ordinary data frame specifying polygons</b> Also use spatialCoordsNames. There should an additional column "ID" to specify which vertices belong to which polygon. The coordinates should not be in list columns. Rather, the data frame should look like it is passed to <code>ggplot2::geom_polygon</code>. If there are holes, then there must also be a column "subID" that differentiates between the outer polygon and the holes.</p> |

In all cases, the data frame should specify the same number of geometries as the number of columns in the gene count matrix. If the column "barcode" is present, then it will be matched to column names of the gene count matrix. Otherwise, the geometries are assumed to be in the same order as columns in the gene count matrix. If the geometries are specified in an ordinary data frame, then it will be converted into *sf* internally. Named list of data frames because each entity can have multiple geometries, such as whole cell and nuclei segmentations. The geometries are assumed to be POINTs for centroids and POLYGONs for segmentations. If polygons are specified in an ordinary data frame, then anything with fewer than 3 vertices will be removed. For anything other than POINTs, attributes of the geometry will be ignored.

rowGeometries	Geometry associated with genes or features, which correspond to rows of the gene count matrix.
annotGeometries	Geometry of entities that do not correspond to columns or rows of the gene count matrix, such as tissue boundary and pathologist annotations of histological regions, and nuclei segmentation in a Visium dataset. Also a named list as in colGeometries. The ordinary data frame may specify POINTs, POLYGONs, or LINESTRINGs, or their MULTI versions. Each data frame can only specify one type of geometry. For MULTI versions, there must be a column "group" to identify each MULTI geometry.
spatialCoordsNames	A character vector of column names if *Geometries arguments have ordinary data frames, to identify the columns in the ordinary data frames that specify the spatial coordinates. If colGeometries is not specified, then this argument will behave as in <a href="#">SpatialExperiment</a> , but colGeometries will be given precedence if provided.
annotGeometryType	Character vector specifying geometry type of each element of the list if annotGeometry is specified. Each element of the vector must be one of POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, and MULTIPOLYGON. Must be either length 1 (same for all elements of the list) or the same length as the list. Ignored if the corresponding element is an <i>sf</i> object.
spatialGraphs	A named list of listw objects (see spdep) for spatial neighborhood graphs.
spotDiameter	Spot diameter for technologies with arrays of spots of fixed diameter per slide, such as Visium, ST, DBiT-seq, and slide-seq. The diameter must be in the same unit as the coordinates in the *Geometry arguments. Ignored for geometries that are not POINT or MULTIPOINT.
unit	# Default unit is "micron". However for Visium one can choose between "micron" or "full_res_image_pixel".
BPPARAM	Deprecated when coercing from <a href="#">SpatialExperiment</a> , but is used when coercing from Seurat object.
sample_id	A character sample identifier, which matches the sample_id in <a href="#">imgData</a> . The sample_id will also be stored in a new column in <a href="#">colData</a> , if not already present. Default = sample01.

spatialCoords	A numeric matrix containing columns of spatial coordinates, as in <a href="#">SpatialExperiment</a> . The coordinates are centroids of the entities represented by the columns of the gene count matrix. If colGeometries is also specified, then it will be given priority and a warning is issued. Otherwise, the sf representation of the centroids will be stored in the colGeometry called centroids.
scaleFactors	Optional scale factors associated with the image(s). This can be provided as a numeric value, numeric vector, list, or file path to a JSON file for the 10x Genomics Visium platform. For 10x Genomics Visium, the correct scale factor will automatically be selected depending on the resolution of the image from imageSources. Default = 1.
imageSources	Optional file path(s) or URL(s) for one or more image sources.
image_id	Optional character vector (same length as imageSources) containing unique image identifiers.
loadImage	Logical indicating whether to load image into memory. Default = FALSE.
imgData	Optional <a href="#">DataFrame</a> containing the image data. Alternatively, this can be built from the arguments imageSources and image_id (see Details).
add_molecules	Logical, whether to add transcripts coordinates to an object.
flip	To flip the image, geometry coordinates, or none. Because the image has the origin at the top left while the geometry has origin at the bottom left, one of them needs to be flipped for them to match. If one of them is already flipped, then use "none". The image will not be flipped if it's GeoTIFF.
image_scalefactors	# A character, choose between "lowres" or "hires". Only for 10X Visium, image scaling factors are from 'scalefactors_json.json' file.

**Value**

A SpatialFeatureExperiment object

**Examples**

```
library(SpatialExperiment)
example(read10xVisium)
# There can't be duplicate barcodes
colnames(spe) <- make.unique(colnames(spe), sep = "--")
rownames(spatialCoords(spe)) <- colnames(spe)
sfe <- toSpatialFeatureExperiment(spe)
# For coercing Seurat to SFE see this -> ./vignettes/seurat_sfe_coerce.Rmd
```

**Description**

The method for SFE reconstructs the spatial graphs when the SFE object is subsetted as the `listw` objects encodes the nodes with indices which are no longer valid after subsetting as some nodes are no longer present.

**Usage**

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

**Arguments**

<code>x</code>	A <code>SpatialFeatureExperiment</code> object.
<code>i</code>	Row indices for subsetting.
<code>j</code>	column indices for subsetting.
<code>...</code>	Passed to the <code>SingleCellExperiment</code> method of <code>[</code> .
<code>drop</code>	Logical. If <code>FALSE</code> , then a warning will be issued that the node indices in the graphs are no longer valid so the row and col graphs affected by subsetting are dropped. At present, this only works with the wrapper functions in this package that take in SFE objects and records the info required to reconstruct the graphs. While this argument is ignored for <code>SummarizedExperiment</code>

**Value**

A subsetted `SpatialFeatureExperiment` object.

**Examples**

```
# Just like subsetting matrices and SingleCellExperiment
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
sfe_subset <- sfe[seq_len(10), seq_len(10), drop = TRUE]
# Gives warning as graph reconstruction fails

sfe_subset <- sfe[seq_len(10), seq_len(10)]
```

---

spatialGraphs

*Spatial graph methods*


---

**Description**

Spatial neighborhood graphs as `spdep`'s `listw` objects are stored in the `int_metadata` of the SFE object. The `listw` class is used because `spdep` has many useful methods that rely on the neighborhood graph as `listw`.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
spatialGraphs(x, MARGIN = NULL, sample_id = "all", name = "all")

colGraphs(x, sample_id = "all", name = "all")

rowGraphs(x, sample_id = "all", name = "all")

annotGraphs(x, sample_id = "all", name = "all")

## S4 replacement method for signature 'SpatialFeatureExperiment'
spatialGraphs(x, MARGIN = NULL, sample_id = "all", name = "all") <- value

colGraphs(x, sample_id = "all", name = "all") <- value

rowGraphs(x, sample_id = "all", name = "all") <- value

annotGraphs(x, sample_id = "all", name = "all") <- value

## S4 method for signature 'SpatialFeatureExperiment,numeric'
spatialGraphNames(x, MARGIN, sample_id = 1L)

## S4 replacement method for signature 'SpatialFeatureExperiment,numeric,ANY,character'
spatialGraphNames(x, MARGIN, sample_id = 1L) <- value

colGraphNames(x, sample_id = 1L)

rowGraphNames(x, sample_id = 1L)

annotGraphNames(x, sample_id = 1L)

colGraphNames(x, sample_id = 1L) <- value

rowGraphNames(x, sample_id = 1L) <- value

annotGraphNames(x, sample_id = 1L) <- value

## S4 method for signature 'SpatialFeatureExperiment'
spatialGraph(x, type = 1L, MARGIN, sample_id = 1L)

colGraph(x, type = 1L, sample_id = 1L)

rowGraph(x, type = 1L, sample_id = 1L)

annotGraph(x, type = 1L, sample_id = 1L)

## S4 replacement method for signature 'SpatialFeatureExperiment'
spatialGraph(x, type = 1L, MARGIN, sample_id = NULL) <- value
```

```
colGraph(x, type = 1L, sample_id = 1L) <- value
rowGraph(x, type = 1L, sample_id = 1L) <- value
annotGraph(x, type = 1L, sample_id = 1L) <- value
```

### Arguments

<code>x</code>	A <code>SpatialFeatureExperiment</code> object.
<code>MARGIN</code>	As in <code>apply</code> . 1 stands for rows and 2 stands for columns. In addition, 3 stands for spatial neighborhood graphs that correspond to <code>annotGeometries</code> .
<code>sample_id</code>	Name of the sample the graph is associated with. This is useful when multiple pieces of tissues are in the same SFE object (say for a joint dimension reduction and clustering) and the spatial neighborhood is only meaningful within the same piece of tissue. See the <code>sample_id</code> argument in <code>SpatialExperiment</code> .
<code>name</code>	Name of the graphs to add to each <code>sample_id</code> ; used in the <code>spatialGraphs</code> replacement method as it must be character while <code>type</code> can be either an integer index or a name.
<code>value</code>	A <code>listw</code> object ( <code>*Graph</code> ), or a named list of list of <code>listw</code> objects ( <code>*Graphs</code> ) where the names of the top level list are <code>sample_ids</code> when adding graphs for all samples in the margin of interest, or a list of <code>listw</code> objects when adding graphs for one sample in one margin.
<code>type</code>	An integer specifying the index or string specifying the name of the <code>*Graph</code> to query or replace. If missing, then the first item in the <code>*Graph</code> will be returned or replaced.

### Value

Getters for multiple graphs return a named list. Getters for names return a character vector of the names. Getters for single graphs return a `listw` object. Setters return an SFE object.

### Examples

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
g1 <- findVisiumGraph(sfe)
g2 <- findSpatialNeighbors(sfe)

# Set all graphs of a margin by a named list
spatialGraphs(sfe, MARGIN = 2L, sample_id = "Vis5A") <-
  list(tri2nb = g2, visium = g1)
# Or equivalently
colGraphs(sfe, sample_id = "Vis5A") <- list(tri2nb = g2, visium = g1)

# Get all graphs of a margin, returning a named list
gs <- spatialGraphs(sfe, MARGIN = 2L)
# Or equivalently
gs <- colGraphs(sfe)
```

```

# Set graph of the same name and same margin for multiple samples
# Each sample has a separate graph
sfe2 <- McKellarMuscleData("small2")
sfe_combined <- cbind(sfe, sfe2)
colGraphs(sfe_combined, name = "visium", sample_id = "all") <-
  findVisiumGraph(sfe_combined, sample_id = "all")

# Get graph names
spatialGraphNames(sfe, MARGIN = 2L, sample_id = "Vis5A")
# Or equivalently (sample_id optional as only one sample is present)
colGraphNames(sfe)

# Set graph names
spatialGraphNames(sfe, MARGIN = 2L) <- c("foo", "bar")
colGraphNames(sfe) <- c("tri2nb", "visium")

# MARGIN = 1 means rowGraphs; MARGIN = 3 means annotation graphs (annotGraphs)
# for both getters and setters

# Set single graph by
# Spatial graph for myofibers
g_myofiber <- findSpatialNeighbors(sfe,
  type = "myofiber_simplified",
  MARGIN = 3L
)
spatialGraph(sfe, type = "myofiber", MARGIN = 3L) <- g_myofiber
# Or equivalently
annotGraph(sfe, "myofiber") <- g_myofiber

# Get a specific graph by name
g <- spatialGraph(sfe, "myofiber", MARGIN = 3L)
g2 <- spatialGraph(sfe, "visium", MARGIN = 2L)
# Or equivalently
g <- annotGraph(sfe, "myofiber")
g2 <- colGraph(sfe, "visium")

```

---

SpatRasterImage

*SpatRaster representation of images in SFE objects*


---

## Description

SpatialFeatureExperiment and the Voyager package work with images differently from SpatialExperiment. In SFE and Voyager's, plotting functions for SFE objects, the images can be read with `rast` and represented as `SpatRaster`, so the image is not entirely loaded into memory unless necessary. Plotting will not load a large image into memory; rather the image will be downsampled and the downsampled version is plotted. A `SpatRasterImage` object (as of Bioc 3.19 or SFE version 1.6 and above) is a `SpatRaster` object but also inheriting from `VirtualSpatialImage` as required by `SpatialExperiment`.

**Usage**

```
SpatRasterImage(img)

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

**Arguments**

img                    A [SpatRaster](#) or PackedSpatRaster object.  
 object                A SpatRasterImage object.

**Value**

A SpatRasterImage object.

**Examples**

```
# Example code
```

---

st\_any\_pred                    *Simple geometry predicates*

---

**Description**

Unlike functions in `sf` like `st_intersects`, this function simply returns a logical vector indicating whether each geometry in `x` intersects (or returns TRUE from other predicates) anything in `y`, preferably when `y` only contains a small number of geometries or is one single MULTI geometry. This is useful when cropping or subsetting an SFE object with a geometry, such as tissue boundary or histological region polygons or a bounding box.

**Usage**

```
st_any_pred(x, y, pred)

st_any_intersects(x, y)

st_n_pred(x, y, pred)

st_n_intersects(x, y)
```

**Arguments**

x                    An object of class `sf`, `sfc`, or `sfg`.  
 y                    Another object of class `sf`, `sfc`, or `sfg`.  
 pred                A geometric binary predicate function, such as [st\\_intersects](#). It should return an object of class `sge`, for sparse predicates.



**Value**

For `st_any_*`, a logical vector indicating whether each geometry in `x` intersects (or other predicates such as is covered by) anything in `y`. Simplified from the `sgbp` results which indicate which item in `y` each item in `x` intersects, which might not always be relevant. For `st_n_*`, an integer vector indicating the number of geometries in `y` returns TRUE for each geometry in `x`.

**Examples**

```
library(sf)
pts <- st_sfc(
  st_point(c(.5, .5)), st_point(c(1.5, 1.5)),
  st_point(c(2.5, 2.5))
)
pol <- st_polygon(list(rbind(c(0, 0), c(2, 0), c(2, 2), c(0, 2), c(0, 0))))
st_any_pred(pts, pol, pred = st_disjoint)
st_any_intersects(pts, pol)
st_n_pred(pts, pol, pred = st_disjoint)
st_n_intersects(pts, pol)
```

---

toExtImage

*Convert images to ExtImage*


---

**Description**

The `ExtImage` class is a thin wrapper around the `Image` class in `ExtImage` so it inherits from `VirtualSpatialImage` as required by `SpatialExperiment` and has extent as used in `Voyager`'s plotting functions. This function converts `SpatRasterImage` (thin wrapper around the class in `terra`) and `BioFormatsImage` into `ExtImage` for image operations as implemented in the `ExtImage` package.

**Usage**

```
## S4 method for signature 'BioFormatsImage'
toExtImage(x, resolution = 4L, channel = NULL)

## S4 method for signature 'SpatRasterImage'
toExtImage(x, maxcell = 1e+07, channel = NULL)
```

**Arguments**

<code>x</code>	Either a <code>BioFormatsImage</code> or <code>SpatRasterImage</code> object.
<code>resolution</code>	Integer, which resolution in the <code>BioFormatsImage</code> to read and convert. Defaults to 4, which is a lower resolution. Ignored if only 1 resolution is present.
<code>channel</code>	Integer vector to indicate channel(s) to read. If <code>NULL</code> , then all channels will be read.
<code>maxcell</code>	Maximum number of pixels when <code>SpatRasterImage</code> is read into memory.

**Value**

A ExtImage object. The image is loaded into memory.

**See Also**

toSpatRasterImage

---

toSpatRasterImage	<i>Convert images to SpatRasterImage</i>
-------------------	--

---

**Description**

The resolution specified from the OME-TIFF file will be read into memory and written to disk as a GeoTIFF file that has the extent. The output file will have the same file name as the input file except without the ome in the extension.

**Usage**

```
## S4 method for signature 'ExtImage'
toSpatRasterImage(
  x,
  save_geotiff = TRUE,
  file_out = "img.tiff",
  overwrite = FALSE
)

## S4 method for signature 'BioFormatsImage'
toSpatRasterImage(
  x,
  save_geotiff = TRUE,
  resolution = 4L,
  channel = NULL,
  overwrite = FALSE
)
```

**Arguments**

x	Either a BioFormatsImage or EBImage object.
save_geotiff	Logical, whether to save the image to GeoTIFF file.
file_out	File to save the non-OME TIFF file for SpatRaster.
overwrite	Logical, whether to overwrite existing file of the same name.
resolution	Integer, which resolution in the BioFormatsImage to read and convert. Defaults to 4, which is a lower resolution. Ignored if only 1 resolution is present.
channel	Integer vector to indicate channel(s) to read. If NULL, then all channels will be read.

**Value**

A `SpatRasterImage` object

**See Also**

`toExtImage`

---

<code>translateImg</code>	<i>Translate/shift image in space</i>
---------------------------	---------------------------------------

---

**Description**

This function shifts the spatial extent of the image in the x-y plane.

**Usage**

```
## S4 method for signature 'SpatRasterImage'  
translateImg(x, v, ...)
```

```
## S4 method for signature 'BioFormatsImage'  
translateImg(x, v, ...)
```

```
## S4 method for signature 'ExtImage'  
translateImg(x, v, ...)
```

**Arguments**

<code>x</code>	An object of class <code>*Image</code> as implemented in this package.
<code>v</code>	Numeric vector of length 2 to shift the image in the x-y plane.
<code>...</code>	Ignored. It's there so different methods can all be passed to the same <code>lapply</code> in the method for SFE objects. Some methods have extra arguments.

**Value**

A `*Image` object of the same class that has been shifted in space.

**See Also**

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [transposeImg\(\)](#)

---

 transposeImg

*Transpose images*


---

### Description

Swap rows and columns of images. In effect, this will flip the image around the diagonal running from top left to bottom right.

### Usage

```
## S4 method for signature 'SpatRasterImage'
transposeImg(x, filename = "", maxcell = NULL, ...)

## S4 method for signature 'BioFormatsImage'
transposeImg(x, ...)

## S4 method for signature 'ExtImage'
transposeImg(x, ...)
```

### Arguments

x	An object of class *Image as implemented in this package.
filename	Output file name for transformed SpatRaster.
maxcell	Max number of pixels to load SpatRasterImage into memory. The default 1e7 is chosen because this is the approximate number of pixels in the medium resolution image at resolution = 4L in Xenium OME-TIFF to make different methods of this function consistent.
...	Ignored. It's there so different methods can all be passed to the same lapply in the method for SFE objects. Some methods have extra arguments.

### Value

For SpatRasterImage and ExtImage, object of the same class. For BioFormatsImage, the image of the specified resolution is read into memory and then the ExtImage method is called, returning ExtImage. For the extent: xmin and xmax are switched with ymin and ymax.

### See Also

Other image methods: [SFE-image](#), [affineImg\(\)](#), [cropImg\(\)](#), [dim,BioFormatsImage-method](#), [ext\(\)](#), [imgRaster\(\)](#), [imgSource\(\)](#), [mirrorImg\(\)](#), [rotateImg\(\)](#), [scaleImg\(\)](#), [translateImg\(\)](#)

---

```
unit, SpatialFeatureExperiment-method
  Get unit of a SpatialFeatureExperiment
```

---

**Description**

Length units can be microns or pixels in full resolution image in SFE objects.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
unit(x)
```

**Arguments**

x                    A SpatialFeatureExperiment object.

**Value**

A string for the name of the unit. At present it's merely a string and `units` is not used.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData(dataset = "small")
SpatialFeatureExperiment::unit(sfe)
```

---

```
updateObject            Update a SpatialFeatureExperiment object
```

---

**Description**

Update a [SpatialFeatureExperiment](#) to the latest version of object structure. This is usually called by internal functions.

**Usage**

```
## S4 method for signature 'SpatialFeatureExperiment'
updateObject(object, ..., verbose = FALSE)

SFEVersion(object)
```

**Arguments**

object	An old <a href="#">SpatialFeatureExperiment</a> object.
...	Additional arguments that are ignored.
verbose	Logical scalar indicating whether a message should be emitted as the object is updated.

**Details**

Version 1.1.4 adds package version to the SFE object. We are considering an overhaul of the `spatialGraphs` slot in a future version using the `sfdep` package to decouple the adjacency graph from the edge weights.

**Value**

An updated version of object.

**See Also**

[objectVersion](#), which is used to determine if the object is up-to-date.

**Examples**

```
library(SFEData)
sfe <- McKellarMuscleData("small")
# First version of SFE object doesn't log SFE package version, so should be NULL
SFEVersion(sfe)
sfe <- updateObject(sfe)
# See current version
SFEVersion(sfe)
```

---

visium\_row\_col

*Row and columns of Visium barcodes on the slide*

---

**Description**

From Space Ranger 1.3.1.

**Usage**

```
visium_row_col
```

**Format**

A data frame with 4992 rows with columns barcode, col, and row.

**Source**

Space Ranger 1.3.1

# Index

- \* **Geometric operations**
  - addVisiumSpotPoly, 3
  - annotOp, 8
  - annotPred, 9
  - annotSummary, 10
  - bbox, SpatialFeatureExperiment-method, 11
  - crop, 19
  - removeEmptySpace, 58
  - SFE-transform, 67
  - st\_any\_pred, 80
- \* **Getters and setters**
  - annotGeometries, 6
  - colFeatureData, 16
  - colGeometries, 17
  - dimGeometries, 24
  - getParams, 38
  - localResults, 43
  - rowGeometries, 60
  - spatialGraphs, 76
- \* **Image affine transformation**
  - affineImg, 4
  - mirrorImg, 47
  - rotateImg, 59
  - scaleImg, 63
  - translateImg, 83
  - transposeImg, 84
- \* **Image classes**
  - BioFormatsImage, 13
  - BioFormatsImage-getters, 14
  - ExtImage, 28
  - SpatRasterImage, 79
  - toExtImage, 81
  - toSpatRasterImage, 82
- \* **Image methods**
  - cropImg, 21
  - dim, BioFormatsImage-method, 24
  - ext, 27
  - Img<-, SpatialExperiment-method, 40
  - imgRaster, 41
  - imgSource, 42
  - SFE-image, 64
- \* **Non-spatial operations**
  - cbind, SpatialFeatureExperiment-method, 15
  - SpatialFeatureExperiment-subset, 75
- \* **Read data into SFE**
  - read10xVisiumSFE, 48
  - readCosMX, 49
  - readVizgen, 53
  - readXenium, 55
- \* **Spatial neighborhood graph**
  - findSpatialNeighbors, SpatialFeatureExperiment-method, 29
  - findVisiumGraph, 31
- \* **SpatialFeatureExperiment class**
  - show, SpatialFeatureExperiment-method, 68
  - SpatialFeatureExperiment, 69
  - SpatialFeatureExperiment-class, 72
  - SpatialFeatureExperiment-coercion, 72
  - unit, SpatialFeatureExperiment-method, 85
  - updateObject, 85
- \* **Transcript spots**
  - formatTxSpots, 32
  - formatTxTech, 35
  - readSelectTx, 51
- \* **Utilities**
  - aggBboxes, 5
  - bbox\_center, 12
  - changeSampleIDs, 16
  - df2sf, 22
  - gdalParquetAvailable, 37
  - getPixelSize, 39
  - imageIDs, 39

- sampleIDs, [62](#)
- saveRDS, SpatialFeatureExperiment-method, [62](#)
- visium\_row\_col, [86](#)
- \* **datasets**
  - visium\_row\_col, [86](#)
- \* **image methods**
  - affineImg, [4](#)
  - cropImg, [21](#)
  - dim, BioFormatsImage-method, [24](#)
  - ext, [27](#)
  - imgRaster, [41](#)
  - imgSource, [42](#)
  - mirrorImg, [47](#)
  - rotateImg, [59](#)
  - scaleImg, [63](#)
  - SFE-image, [64](#)
  - translateImg, [83](#)
  - transposeImg, [84](#)
- \* **internal**
  - internal-Voyager, [42](#)
  - .check\_features (internal-Voyager), [42](#)
  - .check\_sample\_id (internal-Voyager), [42](#)
  - .rm\_empty\_geometries (internal-Voyager), [42](#)
  - .symbol2id (internal-Voyager), [42](#)
  - .value2df (internal-Voyager), [42](#)
  - .warn\_symbol\_duplicate (internal-Voyager), [42](#)
- [, SpatialFeatureExperiment, ANY, ANY, ANY-method (SpatialFeatureExperiment-subset), [75](#)
- addImg, [40](#)
- addImg, SpatialFeatureExperiment-method (SFE-image), [64](#)
- addSelectTx (readSelectTx), [51](#)
- addTxSpots, [51](#)
- addTxSpots (formatTxSpots), [32](#)
- addTxTech (formatTxTech), [35](#)
- addVisiumSpotPoly, [3](#)
- affine (SFE-transform), [67](#)
- affineImg, [4](#), [21](#), [24](#), [28](#), [41](#), [42](#), [48](#), [60](#), [64](#), [66](#), [83](#), [84](#)
- affineImg, BioFormatsImage-method (affineImg), [4](#)
- affineImg, ExtImage-method (affineImg), [4](#)
- affineImg, SpatialFeatureExperiment-method (SFE-image), [64](#)
- affineImg, SpatRasterImage-method (affineImg), [4](#)
- aggBboxes, [5](#)
- annotGeometries, [6](#), [17](#), [30](#)
- annotGeometries, SpatialFeatureExperiment-method (annotGeometries), [6](#)
- annotGeometries<- (annotGeometries), [6](#)
- annotGeometries<- , SpatialFeatureExperiment-method (annotGeometries), [6](#)
- annotGeometry (annotGeometries), [6](#)
- annotGeometry, SpatialFeatureExperiment-method (annotGeometries), [6](#)
- annotGeometry<- (annotGeometries), [6](#)
- annotGeometry<- , SpatialFeatureExperiment-method (annotGeometries), [6](#)
- annotGeometryNames (annotGeometries), [6](#)
- annotGeometryNames, SpatialFeatureExperiment-method (annotGeometries), [6](#)
- annotGeometryNames<- (annotGeometries), [6](#)
- annotGeometryNames<- , SpatialFeatureExperiment, character-method (annotGeometries), [6](#)
- annotGraph (spatialGraphs), [76](#)
- annotGraph<- (spatialGraphs), [76](#)
- annotGraphNames (spatialGraphs), [76](#)
- annotGraphNames<- (spatialGraphs), [76](#)
- annotGraphs (spatialGraphs), [76](#)
- annotGraphs<- (spatialGraphs), [76](#)
- annotNPred (annotPred), [9](#)
- annotOp, [8](#)
- annotPred, [8](#), [9](#)
- annotSummary, [10](#)
- apply, [25](#), [30](#), [78](#)
- bbox (bbox, SpatialFeatureExperiment-method), [11](#)
- bbox, SpatialFeatureExperiment-method, [11](#)
- bbox\_center, [12](#)
- BiocNeighborParam, [30](#)
- BiocParallelParam, [30](#), [34](#), [36](#), [50](#), [54](#), [56](#)
- BioFormatsImage, [13](#), [15](#), [24](#), [39](#), [40](#)
- BioFormatsImage-class (BioFormatsImage), [13](#)
- BioFormatsImage-getters, [14](#)
- cbind, SpatialFeatureExperiment-method, [15](#)



- cellSeg (colGeometries), 17
- cellSeg<- (colGeometries), 17
- centroids (colGeometries), 17
- centroids<- (colGeometries), 17
- changeSampleIDs, 16
- colData, 50, 53, 55, 69, 74
- colData (reexports), 57
- colData<- (reexports), 57
- colFeatureData, 16, 38
- colGeometries, 17
- colGeometries<- (colGeometries), 17
- colGeometry, 53
- colGeometry (colGeometries), 17
- colGeometry<- (colGeometries), 17
- colGeometryNames (colGeometries), 17
- colGeometryNames<- (colGeometries), 17
- colGraph (spatialGraphs), 76
- colGraph<- (spatialGraphs), 76
- colGraphNames (spatialGraphs), 76
- colGraphNames<- (spatialGraphs), 76
- colGraphs (spatialGraphs), 76
- colGraphs<- (spatialGraphs), 76
- connection, 63
- counts (reexports), 57
- crop, 19
- cropImg, 5, 21, 24, 28, 41, 42, 48, 60, 64, 66, 83, 84
- cropImg, BioFormatsImage-method (cropImg), 21
- cropImg, ExtImage-method (cropImg), 21
- cropImg, SpatRasterImage-method (cropImg), 21
  
- DataFrame, 58, 69, 75
- DelayedMatrix, 69
- df2sf, 7, 19, 22, 23, 25, 61, 71
- dim, BioFormatsImage-method, 24
- dimGeometries, 24
- dimGeometries, SpatialFeatureExperiment-method (dimGeometries), 24
- dimGeometries<- (dimGeometries), 24
- dimGeometries<- , SpatialFeatureExperiment-method (dimGeometries), 24
- dimGeometry (dimGeometries), 24
- dimGeometry, SpatialFeatureExperiment-method (dimGeometries), 24
- dimGeometry<- (dimGeometries), 24
- dimGeometry<- , SpatialFeatureExperiment-method (dimGeometries), 24
- dimGeometryNames (dimGeometries), 24
- dimGeometryNames, SpatialFeatureExperiment-method (dimGeometries), 24
- dimGeometryNames<- (dimGeometries), 24
- dimGeometryNames<- , SpatialFeatureExperiment, numeric, character (dimGeometries), 24
  
- ext, 5, 21, 24, 27, 41, 42, 48, 60, 64, 66, 83, 84
- ext, BioFormatsImage-method (ext), 27
- ext, ExtImage-method (ext), 27
- ext, SpatRasterImage-method (ext), 27
- ext<- , BioFormatsImage, numeric-method (ext), 27
- ext<- , ExtImage, numeric-method (ext), 27
- ext<- , SpatRasterImage, numeric-method (ext), 27
- ExtImage, 13, 28, 40
- ExtImage-class (ExtImage), 28
  
- findSpatialNeighbors (findSpatialNeighbors, SpatialFeatureExperiment-method), 29
- findSpatialNeighbors, SpatialFeatureExperiment-method, 29
- findVisiumGraph, 31
- formatTxSpots, 32, 51
- formatTxTech, 35
  
- gdalParquetAvailable, 37
- geometryFeatureData (colFeatureData), 16
- getImg (reexports), 57
- getParams, 38
- getPixelSize, 39
  
- Image, 13, 28
- imageIDs, 39
- Img<- , SpatialExperiment-method, 40
- Img<- (Img<- , SpatialExperiment-method), 40
- imgData, 50, 53, 55, 69, 74
- imgData (reexports), 57
- imgRaster, 5, 21, 24, 28, 41, 42, 48, 60, 64, 66, 83, 84
- imgRaster, BioFormatsImage-method (imgRaster), 41
- imgRaster, ExtImage-method (imgRaster), 41
- imgRaster, SpatRasterImage-method (imgRaster), 41

- imgSource, [5](#), [21](#), [24](#), [28](#), [41](#), [42](#), [48](#), [60](#), [64](#), [66](#), [83](#), [84](#)
- imgSource, BioFormatsImage-method (imgSource), [42](#)
- imgSource, ExtImage-method (imgSource), [42](#)
- imgSource, SpatRasterImage-method (imgSource), [42](#)
- internal-Voyager, [42](#)
- isFull (BioFormatsImage-getters), [14](#)
- isFull, BioFormatsImage-method (BioFormatsImage-getters), [14](#)
  
- KmknnParam, [30](#)
  
- localResult (localResults), [43](#)
- localResult, SpatialFeatureExperiment-method (localResults), [43](#)
- localResult<- (localResults), [43](#)
- localResult<-, SpatialFeatureExperiment-method (localResults), [43](#)
- localResultAttrs (localResults), [43](#)
- localResultAttrs, SpatialFeatureExperiment-method (localResults), [43](#)
- localResultFeatures (localResults), [43](#)
- localResultFeatures, SpatialFeatureExperiment-method (localResults), [43](#)
- localResultNames (localResults), [43](#)
- localResultNames, SpatialFeatureExperiment-method (localResults), [43](#)
- localResultNames<- (localResults), [43](#)
- localResultNames<-, SpatialFeatureExperiment-method (localResults), [43](#)
- localResults, [43](#)
- localResults, SpatialFeatureExperiment-method (localResults), [43](#)
- localResults<- (localResults), [43](#)
- localResults<-, SpatialFeatureExperiment-method (localResults), [43](#)
- logcounts (reexports), [57](#)
  
- mcols, [58](#)
- metadata, [16](#)
- mirror (SFE-transform), [67](#)
- mirrorImg, [5](#), [21](#), [24](#), [28](#), [41](#), [42](#), [47](#), [60](#), [64](#), [66](#), [67](#), [83](#), [84](#)
- mirrorImg, BioFormatsImage-method (mirrorImg), [47](#)
- mirrorImg, ExtImage-method (mirrorImg), [47](#)
- mirrorImg, SpatialFeatureExperiment-method (SFE-image), [64](#)
- mirrorImg, SpatRasterImage-method (mirrorImg), [47](#)
  
- nb2listw, [29](#)
- nb2listwdist, [30](#)
- nucSeg (colGeometries), [17](#)
- nucSeg<- (colGeometries), [17](#)
  
- objectVersion, [86](#)
- origin (BioFormatsImage-getters), [14](#)
- origin, BioFormatsImage-method (BioFormatsImage-getters), [14](#)
  
- rast, [79](#)
- rbind, [15](#)
- read10xVisiumSFE, [48](#)
- readCosMX, [49](#)
- readSelectTx, [51](#)
- readVizgen, [53](#)
- readXenium, [55](#)
- reducedDim (reexports), [57](#)
- reducedDimFeatureData (colFeatureData), [16](#)
- reducedDimNames, [17](#), [38](#)
- reexports, [57](#)
- removeEmptySpace, [6](#), [19](#), [25](#), [58](#), [61](#)
- rmvImg (reexports), [57](#)
- ROIIPoly (colGeometries), [17](#)
- ROIIPoly<- (colGeometries), [17](#)
- rotate (SFE-transform), [67](#)
- rotateImg, [5](#), [21](#), [24](#), [28](#), [41](#), [42](#), [48](#), [59](#), [64](#), [66](#), [83](#), [84](#)
- rotateImg, BioFormatsImage-method (rotateImg), [59](#)
- rotateImg, ExtImage-method (rotateImg), [59](#)
- rotateImg, SpatialFeatureExperiment-method (SFE-image), [64](#)
- rotateImg, SpatRasterImage-method (rotateImg), [59](#)
- rowData (reexports), [57](#)
- rowFeatureData (colFeatureData), [16](#)
- rowGeometries, [60](#)
- rowGeometries<- (rowGeometries), [60](#)
- rowGeometry (rowGeometries), [60](#)

- rowGeometry<- (rowGeometries), 60
- rowGeometryNames (rowGeometries), 60
- rowGeometryNames<- (rowGeometries), 60
- rowGraph (spatialGraphs), 76
- rowGraph<- (spatialGraphs), 76
- rowGraphNames (spatialGraphs), 76
- rowGraphNames<- (spatialGraphs), 76
- rowGraphs (spatialGraphs), 76
- rowGraphs<- (spatialGraphs), 76
  
- sampleIDs, 40, 62, 65
- save, 63
- saveRDS, SpatialFeatureExperiment-method, 62
- scale (SFE-transform), 67
- scaleImg, 5, 21, 24, 28, 41, 42, 48, 60, 63, 66, 83, 84
- scaleImg, AlignedSpatialImage-method (scaleImg), 63
- scaleImg, SpatialFeatureExperiment-method (SFE-image), 64
- SFE-image, 64
- SFE-transform, 67
- SFEVersion (updateObject), 85
- show, BioFormatsImage-method (BioFormatsImage), 13
- show, ExtImage-method (ExtImage), 28
- show, SpatialFeatureExperiment-method, 68
- show, SpatRasterImage-method (SpatRasterImage), 79
- SingleCellExperiment, 71, 72
- sparseMatrix, 69
- spatialCoords (reexports), 57
- spatialCoords<- (reexports), 57
- spatialCoordsNames (reexports), 57
- SpatialExperiment, 70–72, 74, 75, 78
- SpatialFeatureExperiment, 30, 69, 85, 86
- SpatialFeatureExperiment-class, 72
- SpatialFeatureExperiment-coercion, 72
- SpatialFeatureExperiment-subset, 75
- spatialGraph (spatialGraphs), 76
- spatialGraph, SpatialFeatureExperiment-method (spatialGraphs), 76
- spatialGraph<- (spatialGraphs), 76
- spatialGraph<-, SpatialFeatureExperiment-method (spatialGraphs), 76
- spatialGraphNames (spatialGraphs), 76
- spatialGraphNames, SpatialFeatureExperiment, numeric-method (spatialGraphs), 76
- spatialGraphNames<- (spatialGraphs), 76
- spatialGraphNames<-, SpatialFeatureExperiment, numeric, ANY, c (spatialGraphs), 76
- spatialGraphs, 31, 32, 76
- spatialGraphs, SpatialFeatureExperiment-method (spatialGraphs), 76
- spatialGraphs<- (spatialGraphs), 76
- spatialGraphs<-, SpatialFeatureExperiment-method (spatialGraphs), 76
- SpatRaster, 53, 80
- SpatRasterImage, 40, 79
- SpatRasterImage-class (SpatRasterImage), 79
- spotPoly, 71
- spotPoly (colGeometries), 17
- spotPoly<- (colGeometries), 17
- st\_any\_intersects (st\_any\_pred), 80
- st\_any\_pred, 80
- st\_difference, 20
- st\_intersection, 8, 20
- st\_intersects, 10, 11, 80
- st\_n\_intersects (st\_any\_pred), 80
- st\_n\_pred (st\_any\_pred), 80
  
- tissueBoundary (annotGeometries), 6
- tissueBoundary<- (annotGeometries), 6
- toExtImage, 81
- toExtImage, BioFormatsImage-method (toExtImage), 81
- toExtImage, SpatRasterImage-method (toExtImage), 81
- toSpatialFeatureExperiment (SpatialFeatureExperiment-coercion), 72
- toSpatialFeatureExperiment, Seurat-method (SpatialFeatureExperiment-coercion), 72
- toSpatialFeatureExperiment, SingleCellExperiment-method (SpatialFeatureExperiment-coercion), 72
- toSpatialFeatureExperiment, SpatialExperiment-method (SpatialFeatureExperiment-coercion), 72
- toSpatRasterImage, 82
- toSpatRasterImage, BioFormatsImage-method (toSpatRasterImage), 82

toSpatRasterImage,ExtImage-method  
    (toSpatRasterImage), 82

transformation  
    (BioFormatsImage-getters), 14

transformation,BioFormatsImage-method  
    (BioFormatsImage-getters), 14

translate (SFE-transform), 67

translateImg, 5, 21, 24, 28, 41, 42, 48, 60,  
    64, 66, 83, 84

translateImg,BioFormatsImage-method  
    (translateImg), 83

translateImg,ExtImage-method  
    (translateImg), 83

translateImg,SpatialFeatureExperiment-method  
    (SFE-image), 64

translateImg,SpatRasterImage-method  
    (translateImg), 83

transpose (SFE-transform), 67

transposeImg, 5, 21, 24, 28, 41, 42, 48, 60,  
    64, 66, 83, 84

transposeImg,BioFormatsImage-method  
    (transposeImg), 84

transposeImg,ExtImage-method  
    (transposeImg), 84

transposeImg,SpatialFeatureExperiment-method  
    (SFE-image), 64

transposeImg,SpatRasterImage-method  
    (transposeImg), 84

txSpots (rowGeometries), 60

txSpots<- (rowGeometries), 60

unit  
    (unit,SpatialFeatureExperiment-method),  
    85

unit,SpatialFeatureExperiment-method,  
    85

updateObject, 85

updateObject,SpatialFeatureExperiment-method  
    (updateObject), 85

visium\_row\_col, 86

VptreeParam, 30

wrap, 62

writeRaster, 47