Package 'SpatialFeatureExperiment'

April 11, 2023

Type Package

Title Integrating SpatialExperiment with Simple Features in sf

Version 1.0.3

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 pakage builds upon SpatialExperiment and SingleCellExperiment, hence methods for these parent classes can still be used.

Imports BiocGenerics, BiocParallel, methods, rjson, S4Vectors, sf, SingleCellExperiment, SpatialExperiment, spdep (>= 1.1-7), SummarizedExperiment, stats, utils

License Artistic-2.0

Encoding UTF-8

RoxygenNote 7.2.2

Collate 'AllGenerics.R' 'utils.R' 'SFE-class.R' 'annotGeometries.R' 'cbind.R' 'coerce.R' 'data.R' 'df2sf.R' 'dimGeometries.R' 'geometry_operation.R' 'graph_wrappers.R' 'int_dimData.R' 'internal-Voyager.R' 'localResults.R' 'read.R' 'spatialGraphs.R' 'subset.R' 'validity.R'

Suggests BiocStyle, DropletUtils, knitr, Matrix, rmarkdown, SFEData, testthat (>= 3.0.0)

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_16

git_last_commit 24ae8e3

git_last_commit_date 2023-02-07

Date/Publication 2023-04-10

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

Maintainer Lambda Moses <dlu2@caltech.edu>

R topics documented:

`addVisiumSpotPoly` *Add Visium spot polygons to colGeometry*

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)
```

---

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,missing'
annotGeometry(x, type, sample_id = NULL)

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

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

## S4 replacement method for signature 'SpatialFeatureExperiment,missing'
annotGeometry(x, type, sample_id = NULL) <- value

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

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

tissueBoundary(x, sample_id = NULL)

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

Arguments

x	A SpatialFeatureExperiment object.
translate	Logical. Only used if removeEmptySpace 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 df2sf. Defaults are the same as in df2sf. For dimGeometries<- only: geometryType 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 annotGeometry, must be a sf data frame, or an ordinary data frame that can be converted to a sf data frame (see df2sf). For annotGeometries, must be a list of such sf or ordinary data frames. There must be a column sample_id to indicate the sample the geometries are for, and the sample_id must also appear in colData.

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.

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 = NULL,
  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 st_intersection.

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 of 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 = NULL,
  pred = st_intersects
)

annotNPred(
  sfe,
  colGeometryName = 1L,
  annotGeometryName = 1L,
  sample_id = NULL,
  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 `st_intersects`.

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 = NULL,
  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 st_intersects.
summary_fun	Function for the summary, defaults to mean.

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 = NULL)
```

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.

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")
```

---

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.

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 *Change sample IDs*

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)
```

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 = NULL,
  pred = st_intersects,
  op = st_intersection,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL
)
```

Arguments

x	An SFE object.
y	An object of class sf, sfg, or sfc with which to crop the SFE object. Optional if xmin, xmax, ymin, and ymax are specified for a bounding box.
colGeometryName	Column geometry to used to indicate which cells/spots to keep.
sample_id	Samples to crop. Optional when only one sample is present. Can be multiple samples, or "all", which means all samples. For multiple samples, y may have column sample_id indicating which geometry subsets which sample. Only samples included in the sample_id column are subsetted. If there is no sample_id column or y is not specified, then the same geometry or bounding box is used to subset all samples specified in the sample_id argument.
pred	A geometric binary predicate function to indicate which cells/spots to keep, defaults to st_intersects.
op	A geometric operation function to crop the geometries in the SFE object. Defaults to st_intersection.
xmin	Minimum x coordinate of bounding box. Ignored if y is specified.
xmax	Maximum x coordinate of bounding box.
ymin	Minimum y coordinate of bounding box.
ymax	Maximum y coordinate of bounding box.

Value

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

Note

In this version, this function does NOT crop the image.

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"
)
# Subset sfe to only keep what's within a bounding box
# All geometries will be cropped
# sample_id is optional when only one sample is present
sfe_cropped <- crop(sfe,
    colGeometryName = "spotPoly",
    xmin = 5500, xmax = 6500, ymin = 13500, ymax = 14500
)
```

---

df2sf *From ordinary data frame to sf to construct SFE object*

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"),
  BPPARAM = SerialParam()
)
```

Arguments

`df`	An ordinary data frame, i.e. not `sf`. Or a matrix that can be converted to a data frame.
`spatialCoordsNames`	Column names in `df` that specify spatial coordinates.
`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.

geometryType	Type of geometry to convert the ordinary data frame to. If the geometry in df is de facto points, then this argument will be ignored and the returned sf will have geometry type POINT. For any geometry type where one geometry is specified by multiple coordinates, the data frame df must have a column "ID" specifying which coordinate belongs to which geometry. For MULTI* geometries, there must be a "group" column specifying which coordinates for which MULTI geometry.
BPPARAM	An optional BiocParallelParam instance, passed to df2sf to parallelize the conversion of data frames with coordinates to sf geometries.

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")
```

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,missing'
dimGeometry(x, type, MARGIN, sample_id = NULL, withDimnames = TRUE)

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

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

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

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

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

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

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

colGeometries(x, withDimnames = TRUE)

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

colGeometryNames(x)

colGeometryNames(x) <- value

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

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

```
) <- value

rowGeometries(x, withDimnames = TRUE)

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

rowGeometryNames(x)

rowGeometryNames(x) <- value

spotPoly(x, sample_id = NULL, withDimnames = TRUE)

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

centroids(x, sample_id = NULL, withDimnames = TRUE)

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

ROIPoly(x, sample_id = NULL, withDimnames = TRUE)

ROIPoly(x, sample_id = NULL, withDimnames = TRUE, translate = TRUE) <- value

cellSeg(x, sample_id = NULL, withDimnames = TRUE)

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

nucSeg(x, sample_id = NULL, withDimnames = TRUE)

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

txSpots(x, withDimnames = TRUE)

txSpots(x, withDimnames = TRUE, translate = TRUE) <- value
```

Arguments

x	A SpatialFeatureExperiment object.
MARGIN	As in apply. 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 removeEmptySpace 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 df2sf. Defaults are the same as in df2sf. For dimGeometries<- only: geometryType 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 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 df2sf). For dimGeometries, 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.

Details

These are convenience wrappers for getters and setters of special geometries:

colGeometry/ies dimGeometry/ies with MARGIN = 2, for geometries associated with columns of the gene count matrix (cells/Visium spots/samples).

rowGeometry/ies dimGeometry/ies with MARGIN = 1, for geometries associated with rows of the gene count matrix (genes/features).

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.

ROIPoly 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.

txSpots POINT or MULTIPOINT geometries of transcript spots of single molecular resolution technologies, stored in rowGeometries.

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

```
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:
# ROIPoly (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")
```

---

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 = NULL,
  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"),
  alpha = 1,
  dmax = NULL,
  zero.policy = NULL,
  ...
)
```

Arguments

x	A SpatialFeatureExperiment 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 apply, where 1 stands for row, 2 stands for column. Here, in addition, 3 stands for annotation, to query the annotGeometries, such as nuclei segmentation in a Visium data
method	Name of function in the package spdep 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 style argument. "idw" means inverse distance weighting. "exp" means exponential decay. "dpd" means double-power distance weights. See nb2listwdist for details.
glist	list of general weights corresponding to neighbours

style	style can take values “W”, “B”, “C”, “U”, “minmax” and “S”
alpha	Only relevant when dist_type = "dpd".
dmax	Only relevant when dist_type = "dpd".
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 spdep function stated in the method argument, such as k, use_kd_tree, d1, d2, nnmult, sym, and quadsegs. 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 = NULL, style = "W", zero.policy = NULL)
```

Arguments

x	A SpatialFeatureExperiment 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").
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.
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

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.

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")
```

---

localResults *Organize results from local spatial statistics*

Description

Local spatial statics like local Moran’s I, local Geary’s C, Getis-Ord Gi*, 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,missing,missing'
localResults(
  x,
  sample_id = NULL,
  name,
  features = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  withDimnames = TRUE,
  ...
)

## S4 replacement method for signature 'SpatialFeatureExperiment,missing,missing'
localResults(
  x,
  sample_id = NULL,
  name,
  features = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  withDimnames = TRUE,
  ...
) <- value

## S4 method for signature 'SpatialFeatureExperiment,ANY,character'
localResults(
  x,
  sample_id = NULL,
  name,
  features = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  withDimnames = TRUE,
  ...
)

## S4 replacement method for signature 'SpatialFeatureExperiment,ANY,character'
localResults(
  x,
  sample_id = NULL,
  name,
  features = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  withDimnames = TRUE,
  ...
) <- 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
)

## S4 method for signature 'SpatialFeatureExperiment'
localResultAttrs(
  x,
  type = 1L,
  feature,
  colGeometryName = NULL,
  annotGeometryName = NULL
)

## S4 method for signature 'SpatialFeatureExperiment,missing'
localResult(
  x,
  type,
  feature,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  sample_id = NULL,
  withDimnames = TRUE,
  simplify = TRUE
)

## S4 method for signature 'SpatialFeatureExperiment,numeric'
localResult(
  x,
  type,
  feature,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  sample_id = NULL,
  withDimnames = TRUE,
  simplify = TRUE
)
```

```
## S4 method for signature 'SpatialFeatureExperiment,character'
localResult(
  x,
  type,
  feature,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  sample_id = NULL,
  withDimnames = TRUE,
  simplify = TRUE
)

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

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

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

Arguments

x	A SpatialFeatureExperiment object.

sample_id	Sample ID to get or set geometries.
name	Name of the graphs to add to each sample_id; used in the spatialGraphs replacement method as it must be character while type can be either an integer index or a name.
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.
...	Ignored
value	Values to set, should be either a matrix or a data frame.
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.
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")
```

`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 = "",
  sample_id = paste0("sample", sprintf("%02d", seq_along(samples))),
  type = c("HDF5", "sparse"),
  data = c("filtered", "raw"),
  images = "lowres",
  load = TRUE,
  style = "W",
  zero.policy = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

samples	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
sample_id	character string specifying unique sample identifiers, one for each directory specified via samples; ignored if !is.null(names(samples))
type	Either "HDF5", and the matrix will be represented as TENxMatrix, or "sparse", and the matrix will be read as a dgCMatrix.
data	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"
load	logical; should the image(s) be loaded into memory as a grob? If FALSE, will store the path/URL instead.
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
BPPARAM	An optional BiocParallelParam instance, passed to df2sf to parallelize the conversion of data frames with coordinates to sf geometries.

Value

A SpatialFeatureExperiment object

Examples

```
library(SpatialExperiment) # Just for the example directory
dir <- system.file(
    file.path("extdata", "10xVisium"),
    package = "SpatialExperiment"
)
```

```
sample_ids <- c("section1", "section2")
samples <- file.path(dir, sample_ids, "outs")

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

`removeEmptySpace` *Remove empty space*

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.

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)
```

sampleIDs *Get all unique sample IDs*

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)
```

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 = "full_res_image_pixels",
  BPPARAM = SerialParam(),
  ...
)
```

Arguments

`assays`	A `list` or `SimpleList` of matrix-like elements, or a matrix-like object (e.g. an ordinary matrix, a data frame, a DataFrame object from the **S4Vectors** package, a sparseMatrix derivative from the **Matrix** package, a DelayedMatrix object from the **DelayedArray** 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 DataFrame describing the samples. Row names, if present, become the column names of the RangedSummarizedExperiment.
`rowData`	A DataFrame object describing the rows. Row names, if present, become the row names of the SummarizedExperiment object. The number of rows of the DataFrame must equal the number of rows of the matrices in `assays`.

sample_id A character sample identifier, which matches the sample_id in imgData. The sample_id will also be stored in a new column in colData, 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 spotDiamter 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. I'm thinking about using some custom engineering CRS's which can convert units and invert the y axis for Cartesian vs. image orientations. Units are also helpful when plotting scale bars. Ignored for now, until I find a better way to deal with it.

`BPPARAM` An optional `BiocParallelParam` instance, passed to `df2sf` to parallelize the conversion of data frames with coordinates to `sf` geometries.

`...` Additional arguments passed to the `SpatialExperiment` and `SingleCellExperiment` 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. For now coercion only works for SpatialExperiment. I'll deal with Seurat and SingleCellExperiment later.

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 = SerialParam()
)
```

Arguments

x	A SpatialExperiment object to be coerced to a SpatialFeatureExperiment object.
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 spotDiamter 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.

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.

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.

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
: Unit the coordinates are in. I'm thinking about using some custom engineering CRS's which can convert units and invert the y axis for Cartesian vs. image orientations. Units are also helpful when plotting scale bars. Ignored for now, until I find a better way to deal with it.

BPPARAM
: Passed to df2sf, to parallelize the conversion of centroid spatial coordinates in the SPE object to sf point geometry.

Value

An SFE object

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)
```

SpatialFeatureExperiment-subset

Subsetting SpatialFeatureExperiment objects

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

x	A SpatialFeatureExperiment object.
i	Row indices for subsetting.
j	column indices for subsetting.
...	Passed to the SingleCellExperiment method of [.
drop	Logical. If FALSE, 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 SummarizedExperiment

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,missing,missing,missing'
spatialGraphs(x, MARGIN, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,numeric,missing,missing'
```

```
spatialGraphs(x, MARGIN, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,missing,character,missing'
spatialGraphs(x, MARGIN, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,missing,missing'
colGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,missing,missing'
rowGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,missing,missing'
annotGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,numeric,character,missing'
spatialGraphs(x, MARGIN, sample_id = NULL, name)

## S4 method for signature
## 'SpatialFeatureExperiment,numeric,character,character'
spatialGraphs(x, MARGIN, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,missing'
colGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,character'
colGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,missing'
rowGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,character'
rowGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,missing'
annotGraphs(x, sample_id = NULL, name)

## S4 method for signature 'SpatialFeatureExperiment,character,character'
annotGraphs(x, sample_id = NULL, name)

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

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

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

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

## S4 replacement method for signature 'SpatialFeatureExperiment,missing,missing'
colGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,missing,missing'
rowGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,missing,missing'
annotGraphs(x, sample_id = NULL, name) <- value

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

## S4 replacement method for signature 'SpatialFeatureExperiment,character,missing'
colGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,character,missing'
rowGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,character,missing'
annotGraphs(x, sample_id = NULL, name) <- value

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

## S4 replacement method for signature 'SpatialFeatureExperiment,character,character'
colGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,character,character'
rowGraphs(x, sample_id = NULL, name) <- value

## S4 replacement method for signature 'SpatialFeatureExperiment,character,character'
annotGraphs(x, sample_id = NULL, name) <- value

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

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

colGraphNames(x, sample_id = NULL)

rowGraphNames(x, sample_id = NULL)

annotGraphNames(x, sample_id = NULL)
```

```
colGraphNames(x, sample_id = NULL) <- value

rowGraphNames(x, sample_id = NULL) <- value

annotGraphNames(x, sample_id = NULL) <- value

## S4 method for signature 'SpatialFeatureExperiment,missing,numeric'
spatialGraph(x, type, MARGIN, sample_id = NULL)

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

## S4 method for signature 'SpatialFeatureExperiment,character,numeric'
spatialGraph(x, type, MARGIN, sample_id = NULL)

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

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

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

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

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

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

colGraph(x, type = 1L, sample_id = NULL) <- value

rowGraph(x, type = 1L, sample_id = NULL) <- value

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

Arguments

x	A SpatialFeatureExperiment object.
MARGIN	As in apply. 1 stands for rows and 2 stands for columns. In addition, 3 stands for spatial neighborhood graphs that correspond to annotGeometries.
sample_id	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 sample_id argument in SpatialExperiment.
name	Name of the graphs to add to each sample_id; used in the spatialGraphs replacement method as it must be character while type can be either an integer

index or a name.

value — A `listw` object (`*Graph`), or a named list of list of `listw` objects (`*Graphs`) where the names of the top level list are `sample_id`s when adding graphs for all samples in the margin of interest, or a list of `listw` objects when adding graphs for one sample in one margin.

type — An integer specifying the index or string specifying the name of the *Graph to query or replace. If missing, then the first item in the *Graph 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")
```

`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 `sgbp`, 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)
```

---

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