Package 'spatialdataR'

November 21, 2026

Title Representation of Python's spatialdata in R

Depends R (>= 4.6)

Version 0.99.47

Description R interface to Python/scverse's 'spatialdata' framework for unified spatial omics data handling. Adheres to OME-NGFF standards, providing lazy, on-disk representations for multiscale images and labels (ZarrArray), as well as points and shapes (DuckDB-backed tables). Includes handling of coordinate transformation systems and spatial utilities for cropping, masking, and querying. Integrates tabular annotations as 'SingleCellExperiment's (via 'anndataR'), enabling interoperable spatial omics workflows across R and Python.

Imports anndataR (>= 1.1.3), BiocGenerics (>= 0.59.6), DBI, DelayedArray, dplyr, duckdb, duckspatial, graph, Matrix, methods, Rarr, RBGL, rlang, sf, S4Vectors, SingleCellExperiment, SummarizedExperiment, SparseArray, ZarrArray

Suggests BiocStyle, EBImage, knitr, Rgraphviz, testthat (>= 3.0.0)

biocViews DataImport, DataRepresentation, Infrastructure, ImmunoOncology, GeneExpression, Transcriptomics, SingleCell, Spatial

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

BugReports https://github.com/HelenaLC/spatialdataR/issues

URL https://helenalc.github.io/spatialdataR, https://github.com/HelenaLC/spatialdataR

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

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

git_branch devel

git_last_commit 00e7596

git_last_commit_date 2026-10-17

Repository Bioconductor 3.24

Date/Publication 2026-11-20

Author Helena L. Crowell [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4801-1767>),
Artür Manukyan [aut] (ORCID: <https://orcid.org/0000-0002-0441-9517>),
Hugo Gruson [aut] (ORCID: <https://orcid.org/0000-0002-4094-1476>),
Vince Carey [aut] (ORCID: <https://orcid.org/0000-0003-4046-0063>)

Maintainer Helena L. Crowell <helena@crowell.eu>

Contents

blobs . . . 2
centroids . . . 3
combine . . . 4
crop . . . 4
CTgraph . . . 6
CTutils . . . 7
extent . . . 9
mask . . . 10
misc . . . 11
path . . . 12
query . . . 13
readSpatialData . . . 13
reexports . . . 15
SpatialData-class . . . 15
SpatialDataArray . . . 17
SpatialDataAttrs . . . 20
SpatialDataFrame . . . 22
table-utils . . . 25
trans . . . 26

Index **29**

`blobs` *'SpatialData' .zarr toy datasets*

Description

data were retrieved on Nov. 11th, 2024, from here.

Value

zarr store.

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
(x <- readSpatialData(x))
```

centroids *Spatial element centroids*

Description

Spatial element centroids

Usage

```
## S4 method for signature 'ANY'
centroids(x, ...)

## S4 method for signature 'SpatialDataLabel'
centroids(x, as = c("data.frame", "matrix"))

## S4 method for signature 'SpatialDataShape'
centroids(x, as = c("data.frame", "matrix", "list"))

## S4 method for signature 'SpatialDataPoint'
centroids(x, as = c("data.frame", "list"))
```

Arguments

x	a SpatialData element (any but image).
...	ignored.
as	character string; how results should be returned.

Value

A table (data.frame or matrix) of spatial coordinates (if as="list", split by instance (shapes) or features (points)).

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

centroids(label(x))
centroids(shape(x))

head(centroids(point(x)))
xy <- centroids(point(x), "list")
plot(xy$gene_a, col=a <- "red")
points(xy$gene_b, col=b <- "blue")
legend("topright", legend=names(xy), col=c(a, b), pch=21)
```

combine *Combine two* SpatialData *objects*

Description

Combine two SpatialData objects

Usage

```
## S4 method for signature 'list,missing'
combine(x, y, ...)

## S4 method for signature 'SpatialData,SpatialData'
combine(x, y, ...)
```

Arguments

x, y	SpatialData objects to combine.
...	ignored.

Value

A SpatialData objects containing all elements from x and y with names made unique.

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x)

y <- combine(x, x)
imageNames(y)
region(table(y, 1))
region(table(y, 2))

y <- combine(list(Alpha=x, x, Omega=x))
shapeNames(y)
```

crop *spatial cropping*

Description

crop subsets SpatialData elements according to a rectangular bounding box or arbitrary polygonal shapes.

For SpatialData objects, crop propagates the operation across all layers that share the coordinate space j.

For SpatialDataFrames (points and shapes), cropping relies on sf::st_intersects (i.e., instances that intersect the query region in any way are kept). For circle shapes, radii are currently ignored (i.e., a circle is kept if its centroid intersects the query region).

For SpatialDataArrays (images and labels), only bounding box cropping is supported. The requested spatial bounding box is projected into pixel coordinates, and the underlying array is sliced accordingly. The wh metadata is updated to reflect the new spatial extent.

Usage

```
## S4 method for signature 'SpatialDataArray'
crop(x, y, j = 1, ...)

## S4 method for signature 'SpatialDataFrame'
crop(x, y, j = 1, ...)

## S4 method for signature 'SpatialData'
crop(x, y, j = 1, drop = TRUE, ...)
```

Arguments

x	SpatialData object or element.
y	query specification; bounding box: length-4 numeric list with names 'xmin/xmax/ymin/ymax', or an st_bbox; polygon: numeric matrix with 2 columns (= xy-coordinates), or an st_polygon (sfg) or sfc/sf object.
j	character string specifying a coordinate system.
...	optional arguments passed to and from other methods.
drop	logical scalar; should elements without coordinate system j be dropped from the object (TRUE) or left untouched (FALSE)?

Value

same as input

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")
sd <- readSpatialData(zs, tables=FALSE)

# bounding box crop of a SpatialData object
y <- list(xmin=10, xmax=50, ymin=10, ymax=50)
crop(sd, y, j="global")

# cropping individual elements
a <- sf::st_bbox(c(xmin=10, xmax=50, ymin=10, ymax=50))
b <- matrix(c(10,10, 25,50, 40,10, 10,10), ncol=2, byrow=TRUE)
p <- crop(point(sd), a)
q <- crop(point(sd), b)

plot(p$geometry, col="blue")
plot(q$geometry, col="red", add=TRUE)
plot(sf::st_as_sfc(a), add=TRUE)
lines(b, type="l")
```

CTgraph *Coord. trans. graph*

Description

Coord. trans. graph

Usage

```
## S4 method for signature 'SpatialData'
CTgraph(x)

## S4 method for signature 'SpatialDataElement'
CTgraph(x)

## S4 method for signature 'ANY'
CTgraph(x)

## S4 method for signature 'SpatialData'
CTpath(x, i, j)

## S4 method for signature 'SpatialDataElement'
CTpath(x, j)

## S4 method for signature 'ANY'
CTpath(x)

CTplot(g, cex = 0.5, fac = 2, max = 10)
```

Arguments

x	SpatialData, an element, or SpatialDataAttrs.
i	character string; name of source node.
j	character string; name of target coordinate space.
g	base R graph; extracted with CTgraph.
cex	scalar numeric; controls fontsize of node labels.
fac, max	scalar numeric; node labels with nchar>max are split and hyphenated at position floor(nchar/fac)

Value

- CTgraph: graph::graphAM object with nodes for each element and coordinate space, and edges for each transformation (if specified)
- CTpath: list of transformations from i to j; length > 1 if type is "sequential", length-1 otherwise; each element specifies type and data of the transformation
- CTplot: visualizes the element-coordinate space graph with Rgraphviz

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

# object-wide
g <- CTgraph(x)
CTplot(g)

# one element
y <- label(x)
g <- CTgraph(y)
CTplot(g)

# retrieve transformation(s)
# from element to target space
CTpath(x, "blobs_labels", "sequence")
```

CTutils *Coord. trans. utilities*

Description

Coord. trans. utilities

Usage

```
## S4 method for signature 'SpatialDataAttrs'
axes(x, y = NULL, ...)

## S4 method for signature 'SpatialDataAttrs'
CTlist(x, ...)

## S4 method for signature 'SpatialDataAttrs'
CTdata(x, i = 1, ...)

## S4 method for signature 'SpatialDataAttrs'
CTtype(x, ...)

## S4 method for signature 'SpatialDataAttrs'
CTname(x, ...)

## S4 method for signature 'SpatialDataAttrs'
CTunit(x, ...)

## S4 method for signature 'SpatialDataElement'
axes(x, y = NULL, ...)

## S4 method for signature 'SpatialDataElement'
CTlist(x, ...)

## S4 method for signature 'SpatialDataElement'
```

```
CTtype(x, ...)

## S4 method for signature 'SpatialDataElement'
CTunit(x, ...)

## S4 method for signature 'SpatialDataElement'
CTname(x, ...)

## S4 method for signature 'SpatialDataElement'
CTdata(x, i = 1, ...)

## S4 method for signature 'SpatialData'
CTname(x, ...)

## S4 method for signature 'SpatialDataElement'
rmvCT(x, i)

## S4 method for signature 'SpatialDataAttrs'
rmvCT(x, i)

## S4 method for signature 'SpatialDataElement'
addCT(x, name, type = "identity", data = NULL)

## S4 method for signature 'SpatialDataAttrs'
addCT(x, name, type = "identity", data = NULL)
```

Arguments

x	SpatialData, an element, or SpatialDataAttrs.
y	NULL (default) returns a list where each element is an axis: a list with name/type/unit (e.g., x/space/micrometer); y="name/type/unit" extracts specific data over all axiis.
...	option arguments passed to and from other methods.
i	for CTpath, source node label; else, string or scalar integer giving the name or index of a coordinate space.
name	character(1); name of coordinate space
type	character(1); type of transformation
data	transformation data; size and shape depend on transformation and element type (e.g., numeric(1) for rotation, numeric(2) for scaling in 2D)

Value

- CTname: character string; transformation name (e.g., "global")
- CTtype: character string; transformation type (e.g., "affine")
- CTdata: list; transformation data (e.g., scalar numeric for rotation)
- CTlist: list; list of transformation specifications per OME-NGFF spec
- CTunit: list of lists; for each transformation, the unit of each axis
- add/rmvCT: SpatialDataElement or SpatialDataAttrs with transformation(s) added/removed
- axes: list; each element is a character string (name), or list with axis name and type (e.g., "space" or "channel")

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

# view available target coordinate systems
CTname(z <- meta(label(x)))

# add
addCT(z, "scale", "scale", c(12, 34)) # overwrite
CTname(addCT(z, "new", "translation", c(12, 34)))

# rmv
CTname(rmvCT(z, 2))        # by index
CTname(rmvCT(z, "scale"))  # by name
CTname(rmvCT(z, "global")) # identity is protected
```

`extent` *Spatial element extent*

Description

Spatial element extent

Usage

```
## S4 method for signature 'SpatialData'
extent(x, i = 1)

## S4 method for signature 'SpatialDataArray'
extent(x, i = 1)

## S4 method for signature 'SpatialDataFrame'
extent(x, i = 1)
```

Arguments

`x`	a `SpatialData` element (any but table).
`i`	scalar integer or string; target coordinate space.

Value

Length-2 list with numeric x and y ranges.

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

# object-wide
extent(x)
```

```
# element-wise
extent(image(x))
extent(point(x))
extent(shape(x))

# with transformation(s)
extent(label(x), "scale")
extent(label(x), "translation")
```

---

mask *Aggregate data across layers*

Description

Masking operations serve to aggregate data across layers, e.g., counting points in shapes, averaging image channels by labels, etc. For added flexibility, these may be carried out directly between elements, or using an input `SpatialData` object and specifying element names.

Usage

```
## S4 method for signature 'SpatialData'
mask(
  x,
  i,
  j,
  k,
  how = NULL,
  name = function(i, j) sprintf("%s_by_%s", i, j),
  ...
)
```

Arguments

x	SpatialData object.
i, j	character string; names of elements to mask, specifically, `i` will be masked by `j`, adding a `table` for `j` in `x`.
k	string or scalar integer; specifies target coordinate space (defaults to first common coordinate space between `i` and `j`)
how	character string; statistic to use for masking.
name	function use to generate the new `table`'s name.
...	optional arguments passed to and from other methods.

Value

Input `SpatialData` object `x` with an additional table.

Examples

```
library(SingleCellExperiment)
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

# count points in shapes
y <- mask(x, "blobs_points", "blobs_circles")
tail(tables(y), 1)

# average image channels by labels
y <- mask(x, "blobs_image", "blobs_labels")
tail(tables(y), 1)

# TODO: shape,shape example
```

`misc` *Miscellaneous 'SpatialData' methods*

Description

Miscellaneous methods (e.g., `show`) for the `SpatialData` class and its elements.

Usage

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

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

Arguments

`object`	`SpatialData` object or one of its elements, i.e., a `SpatialDataImage/Label/Point/Shape`.

Value

`NULL`

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")
(sd <- readSpatialData(zs))

# show element
image(sd)
label(sd)
point(sd)
shape(sd)

# show .zattrs
meta(label(sd))
meta(image(sd, 2))
```

path *Retrieve* SpatialData *on-disk paths*

Description

Retrieve SpatialData on-disk paths

Usage

```
## S4 method for signature 'SpatialDataArray'
path(object, ...)

## S4 method for signature 'SpatialDataFrame'
path(object, ...)

## S4 method for signature 'SingleCellExperiment'
path(object, ...)

## S4 method for signature 'SpatialData'
path(object, simplify = TRUE, ...)
```

Arguments

object	SpatialData object or one of its elements.
...	ignored.
simplify	logical scalar; whether to flatten paths into a tibble.

Value

for single elements, a character string; for SpatialData objects, if simplify=TRUE (default), a tibble where rows=elements and columns=layers/elements/paths. if simplify=FALSE, a depth-3 list where levels=layers/elements/paths.

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")
sd <- readSpatialData(zs)

# element-wise
path(shape(sd))

# object-wide
path(sd)
path(sd, FALSE)$labels
```

query *queries*

Description

query provides a interface for table-based subsetting of SpatialData objects. It filters a specified table using dplyr::filter logic and propagates the result to all associated spatial elements (i.e., only instances present in the filtered table are kept).

For spatial cropping, see crop.

Usage

```
## S4 method for signature 'SpatialData'
query(x, ..., i = 1)
```

Arguments

x	SpatialData object.
...	logic passed to dplyr::filter.
i	index or name of table to query.

Value

SpatialData object

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")
sd <- readSpatialData(zs)

# filter by 'region' and propagate to shapes/points
t <- table(sd)
query(sd, i=1, region == region(t))
```

readSpatialData *Reading 'SpatialData'*

Description

Reading 'SpatialData'

Usage

```
readImage(x, ...)

readLabel(x, ...)

readPoint(x, ...)

readShape(x, ...)

readTable(x)

readSpatialData(
  x,
  images = TRUE,
  labels = TRUE,
  points = TRUE,
  shapes = TRUE,
  tables = TRUE
)
```

Arguments

x	For `readImage/Label/Point/Shape/Table`, path to a `SpatialData` element. For `readSpatialData`, path to a `SpatialData`-.zarr store.
...	option arguments passed to and from other methods.
images, labels, points, shapes, tables	Control which elements should be read for each layer. The default, NULL, reads all elements; alternatively, may be FALSE to skip a layer, or a integer vector specifying which elements to read.

Value

- For `readSpatialData`, a `SpatialData`.,
- For element readers, a `SpatialDataImage/Label/Point/Shape` or `SingleCellExperiment`.

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")

# read complete Zarr store
(sd <- readSpatialData(zs))

# helper that gets path to last element in layer 'l'
fn <- \(.) tail(list.files(file.path(zs, .), full.names=TRUE), 1)

# read individual elements
(i <- readImage(fn("images")))
channels(i)

(p <- readPoint(fn("points")))
as.data.frame(head(p))
```

```
(s <- readShape(fn("shapes")))
data(s)
```

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

dplyr `filter()`, `mutate()`, `pull()`, `select()`

`SpatialData-class` *The 'SpatialData' class*

Description

`SpatialData` provides an R interface to Python's `spatialdata`, which enables the representation of diverse spatial omics datasets using the OME-NGFF (Next Generation File Format) standard. In R,

- images and labels are `ZarrArrays` (`Rarr` package).
- points and shapes are managed using `duckspatial` tables.
- tables are `SingleCellExperiments` (read with `anndataR`).

Usage

```
SpatialData(
  images = list(),
  labels = list(),
  points = list(),
  shapes = list(),
  tables = list()
)

## S4 method for signature 'SpatialData'
x$name

## S4 replacement method for signature 'SpatialData'
x$name <- value

## S4 method for signature 'SpatialData,numeric,ANY'
x[[i, j, ...]]

## S4 method for signature 'SpatialData,character,ANY'
x[[i, j, ...]]

## S4 method for signature 'SpatialDataElement'
data(x, k = 1, ...)
```

```
## S4 method for signature 'SpatialDataElement'
meta(x)

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

## S4 method for signature 'SpatialData'
rownames(x)

## S4 method for signature 'SpatialData'
colnames(x)

## S4 method for signature 'SpatialData,character'
layer(x, i)

## S4 method for signature 'SpatialData,ANY'
layer(x, i)

## S4 method for signature 'SpatialData,character'
element(x, i)

## S4 method for signature 'SpatialData,numeric'
element(x, i)

## S4 method for signature 'SpatialData,missing'
element(x, i)

## S4 method for signature 'SpatialData,ANY'
element(x, i)

## S4 replacement method for signature 'SpatialData,character'
element(x, i) <- value

## S4 method for signature 'SpatialData'
images(x)

## S4 method for signature 'SpatialData'
labels(object)

## S4 method for signature 'SpatialData'
points(x)

## S4 method for signature 'SpatialData'
shapes(x)

## S4 method for signature 'SpatialData'
tables(x)

## S4 replacement method for signature 'SpatialData,character,ANY,ANY'
x[[i]] <- value
```

```
## S4 replacement method for signature 'SpatialData,ANY,ANY,ANY'
x[[i]] <- value
```

Arguments

images	list of SpatialDataImages
labels	list of SpatialDataLabels
points	list of SpatialDataPoints
shapes	list of SpatialDataShapes
tables	list of SingleCellExperiments
x, object	SpatialData object.
name	character string for extraction (see ?base::`$`).
value	(list of) element(s) with layer-compliant object(s), or NULL/list() to remove an element/layer completely; for element<-, a single SpatialDataElement of the same class as element(x, i).
i, j	character string, scalar or vector of indices specifying the element to extract from a given layer.
...	optional arguments passed to and from other methods.
k	scalar index specifying which scale to use; Inf to use lowest available resolution.
drop	ignored.

Value

SpatialData

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
(x <- readSpatialData(x))

# subsetting
# layers are taken in order of appearance
# (images, labels, points, shapes, tables)
x[-4] # drop layer
x[4, -2] # drop element
x["shapes", c(1, 3)] # subset layer
x[c(1, 2), list(1, c(1, 2))] # multiple
```

SpatialDataArray SpatialDataArray

Description

The SpatialDataImage and -Label classes represent elements from a SpatialData's images/ and labels/ layers, respectively. In both cases, these are represented as a ZarrArray (data slot), and associated with .zattrs represented as SpatialDataAttrs (meta slot); a list of metadata stores other arbitrary info.

Currently defined methods (here, x is a SpatialDataArray):

- data/meta(x) access underlying data/.zattrs
- data_type(x) gets the underlying data type (e.g., float64)
- channels(x) gets channel names (applies to images only)
- dim(x) returns the dimensions of data(x)
- length(x) returns the length of data(x)

Usage

```
SpatialDataImage(
  data = list(),
  meta = SpatialDataAttrs(type = "image"),
  metadata = list(),
  ...
)

SpatialDataLabel(
  data = list(),
  meta = SpatialDataAttrs(type = "label"),
  metadata = list(),
  ...
)

## S4 method for signature 'SpatialDataArray'
dim(x)

## S4 method for signature 'SpatialDataArray'
length(x)

## S4 method for signature 'SpatialDataArray'
data_type(x)

## S4 method for signature 'DelayedArray'
data_type(x)

## S4 method for signature 'SpatialDataAttrs'
channels(x, ...)

## S4 method for signature 'SpatialDataImage'
channels(x, ...)

## S4 method for signature 'SpatialDataElement'
channels(x, ...)

## S4 method for signature 'SpatialDataImage,ANY,ANY,ANY'
```

```
x[i, j, k, ..., drop = FALSE]

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

Arguments

data	list of ZarrArrays
meta	SpatialDataAttrs
metadata	optional list of arbitrary additional content.
...	option arguments passed to and from other methods.
x	SpatialDataArray
i, j, k	indices specifying elements/slices to extract.
drop	ignored.

Value

SpatialDataArray

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")

# get path to 'i'th element in layer 'l'
fn <- \(l, i=1) list.dirs(file.path(zs, l), recursive=FALSE)[i]

# label
(x <- readLabel(fn("labels")))
x[1:10, 1:10]
meta(x)

# image
readImage(fn("images"))

# multi-scale
(x <- readImage(fn("images", 2)))

channels(x)
dim(data(x, 1))   # highest res.
dim(data(x, Inf)) # lowest res.

# RGB visual
rgb <- apply(
  data(x, 1), c(2, 3),
  \(.) rgb(.[1], .[2], .[3]))
plot(
  row(rgb), col(rgb), col=rgb,
  pch=15, asp=1, ylim=c(ncol(rgb), 0))
```

SpatialDataAttrs *The 'SpatialDataAttrs' class*

Description

The 'SpatialDataAttrs' class

Usage

```
SpatialDataAttrs(
  x,
  type = c("image", "label", "point", "shape"),
  trans = NULL,
  ver = NULL,
  dim = 2,
  nch = 3
)

## S4 method for signature 'SpatialDataAttrs'
x$name

## S4 method for signature 'SpatialDataPoint'
feature_key(x)

## S4 method for signature 'SpatialDataAttrs'
feature_key(x)

## S4 replacement method for signature 'SpatialDataAttrs,character'
feature_key(x) <- value

## S4 method for signature 'SingleCellExperiment'
region_key(x)

## S4 method for signature 'SingleCellExperiment'
region(x)

## S4 method for signature 'SingleCellExperiment'
regions(x)

## S4 replacement method for signature 'SingleCellExperiment,character'
regions(x) <- value

## S4 replacement method for signature 'SingleCellExperiment,NULL'
regions(x) <- value

## S4 method for signature 'list'
instance_key(x)

## S4 method for signature 'SingleCellExperiment'
instance_key(x)
```

```
## S4 method for signature 'SpatialDataFrame'
instance_key(x)

## S4 method for signature 'SpatialDataLabel'
instance_key(x)

## S4 replacement method for signature 'SpatialDataAttrs,character'
instance_key(x) <- value

## S4 replacement method for signature 'SingleCellExperiment,character'
instance_key(x) <- value

## S4 method for signature 'SpatialDataLabel'
instances(x)

## S4 method for signature 'SpatialDataPoint'
instances(x)

## S4 method for signature 'SpatialDataShape'
instances(x)

## S4 method for signature 'SingleCellExperiment'
instances(x)

## S4 replacement method for signature 'SingleCellExperiment'
instances(x) <- value
```

Arguments

x	element or list extracted from a OME-NGFF compliant .zattrs file.
type	character string; either "image", "label", "point" or "shape"
trans	list of coordinate transformations; defaults to identity only.
ver	character string; specifies the version of the SpatialData element to comply with.
dim	scalar integer in 2-4 when type is either "image" or "label", and 2-3 when "shape" or "point"; number of dimensions: 2 = XY, 3 adds Z, 4 adds T (time) for image and label; when type="image", C (channel) will be added (for any dim).
nch	scalar integer; how many channels should there be? (ignored if type="label", type="shape", or type="point").
name	character string for extraction (see ?base::'$').
value	character string (for one region and _keys), or vector (for many regions, instances and regions).

Details

When x is a spatial element, the following applies: SpatialDataFrame: feature/instance_key, SingleCellExperiment: region, region/instance_key.

When missing x, SpatialDataAttrs will generate a valid object with default axes (image: cyx, label: yx, point/shape: xy) and transformations (identify) according to the specified type.

Value

SpatialDataAttrs object

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x)

# tables
region(table(x))
region_key(table(x))

# points
instance_key(point(x))
fk <- feature_key(point(x))
base::table(point(x)[[fk]])

# transformations
(z <- meta(label(x)))
CTname(z)
CTtype(z)
CTdata(z, "scale")

# constructor
SpatialDataAttrs(type="point")
SpatialDataAttrs(type="shape")
SpatialDataAttrs(type="image", nch=7)
SpatialDataAttrs(type="label", dim=3)
```

SpatialDataFrame SpatialDataFrame

Description

The SpatialDataPoint and -Shape classes represent elements from a SpatialData's points/ and shapes/ layers, respectively. In both cases, these are represented as a duckspatial_df (data slot), and associated with .zattrs represented as SpatialDataAttrs (meta slot); a list of metadata stores other arbitrary info.

Currently defined methods (here, x is an SpatialDataFrame):

- data/meta(x) access underlying data/.zattrs
- geom_type(x) get the shape's type (e.g., POLYGON)
- names(x) returns the underlying table's column names
- dim(x) returns the dimensions of data(x)
- `$`,`[[` directly access columns of data(x)
- filter,select to subset rows/columns à la dplyr
- as.data.frame to coerce x to a data.frame

Usage

```
SpatialDataPoint(
  data = NULL,
  meta = SpatialDataAttrs(type = "point"),
  metadata = list(),
  ik = NULL,
  fk = NULL,
  ...
)

SpatialDataShape(
  data = NULL,
  meta = SpatialDataAttrs(type = "shape"),
  metadata = list(),
  ...
)

## S4 method for signature 'SpatialDataFrame'
length(x)

## S4 method for signature 'SpatialDataFrame'
dim(x)

## S4 method for signature 'SpatialDataFrame'
names(x)

## S4 method for signature 'SpatialDataFrame'
as.data.frame(x)

## S4 method for signature 'SpatialDataShape'
geom_type(x)

## S3 method for class 'SpatialDataFrame'
pull(.data, ...)

## S3 method for class 'SpatialDataFrame'
select(.data, ...)

## S3 method for class 'SpatialDataFrame'
mutate(.data, ...)

## S3 method for class 'SpatialDataFrame'
filter(.data, ...)

## S4 method for signature 'SpatialDataFrame,ANY,ANY'
x[[i, j, ...]]

## S4 method for signature 'SpatialDataPoint'
x$name

## S3 method for class 'SpatialDataShape'
.DollarNames(x, pattern = "")
```

```
## S4 method for signature 'SpatialDataShape'
x$name

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

Arguments

data	duckspatial_df for on-disk representation, or a data.frame to be converted.
meta	SpatialDataAttrs
metadata	optional list of arbitrary content describing the overall object.
ik, fk	character string specifying "instance_/feature_key" of the spatialdata_attrs; used to match observations/features.
...	optional arguments passed to and from other methods.
x, .data	SpatialDataFrame
i, j	indices for subsetting (see ?base::Extract).
name	character string for extraction (see ?base::`$`).
drop, pattern	ignored.

Value

SpatialDataFrame

Examples

```
zs <- file.path("extdata", "blobs.zarr")
zs <- system.file(zs, package="spatialdataR")

# points
pa <- list.dirs(
  file.path(zs, "points"),
  recursive=FALSE, full.names=TRUE)
(x <- readPoint(pa))

y <- filter(x,
  genes == "gene_b",
  instance_id == 7)
head(as.data.frame(y))

# shapes
pa <- list.dirs(
  file.path(zs, "shapes"),
  recursive=FALSE, full.names=TRUE)

# circles
(x <- readShape(pa[1]))
length(x)
x$radius

# polygons
(y <- readShape(pa[2]))
df <- as.data.frame(y)
```

```
plot(df, col=seq(nrow(df)))

# multi-polygons
(z <- readShape(pa[3]))
df <- as.data.frame(z)
plot(df, col=seq(nrow(df)))
```

---

table-utils SpatialData *annotations*

Description

SpatialData annotations

Usage

```
## S4 method for signature 'SingleCellExperiment'
meta(x)

## S4 method for signature 'SpatialData,ANY'
hasTable(x, i)

## S4 method for signature 'SpatialData,character'
hasTable(x, i, name = FALSE)

## S4 method for signature 'SpatialData,ANY'
getTable(x, i, j, assay = 1, drop = TRUE)

## S4 method for signature 'SpatialData,character'
getTable(x, i, j, assay = 1, drop = TRUE)

## S4 method for signature 'SpatialData,ANY'
setTable(x, i, ..., name = NULL, rk = "rk", ik = "ik")

## S4 method for signature 'SpatialData,character'
setTable(x, i, y, name = NULL, rk = "region", ik = "instance_id")
```

Arguments

x	SpatialData object.
i	character string; name of the element for which to get/set a table.
name	logical; should the table name be returned instead of TRUE/FALSE?
j	character string; colData column, or row name to retrieve assay data.
assay	character string or scalar integer; specifies which assay to use when j is a row name.
drop	logical; should observations (columns) that don't belong to i be filtered out?
...	option arguments passed to and from other methods.
rk, ik	character string; region and instance key (the latter will be ignored if an instance key is already specified within element i).
y	SingleCellExperiment containing annotations for i.

Value

- hasTable: logical scalar (or character string, if name=TRUE); whether or not a table annotating i exists in x
- getTable: SingleCellExperiment; the table annotating i with optional filtering of matching observations
- valTable: vector of values (according to j) from the table annotating i

Examples

```
library(SingleCellExperiment)
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x)

# check if element has a 'table'
hasTable(x, "blobs_points")
hasTable(x, "blobs_labels")

# retrieve 'table' for element 'i'
sce <- getTable(x, i="blobs_labels")
head(colData(sce))
meta(sce)

# get values from 'table'
getTable(x,
  i="blobs_labels",
  j="channel_0_sum")

# add 'table' annotating an element 'i'

# labels
y <- x; tables(y) <- list()
mtx <- matrix(0, 1, length(instances(label(y))))
sce <- SingleCellExperiment(list(counts=mtx))
y <- setTable(y, i <- "blobs_labels", sce)
getTable(y, i)

# shapes
i <- "blobs_circles"
mtx <- matrix(0, 1, nrow(shape(x, i)))
sce <- SingleCellExperiment(list(counts=mtx))
y <- setTable(x, i, sce)
getTable(y, i)
```

---

trans *Transformations*

Description

Transformations

Usage

```
## S4 method for signature 'SpatialDataElement'
transform(x, i = 1, ...)

## S4 method for signature 'SpatialDataElement'
sequence(x, t, ..., rev = FALSE)

## S4 method for signature 'SpatialDataArray'
mirror(x, t = c("v", "h"), k = 1, ...)

## S4 method for signature 'SpatialDataArray'
flip(x, k = 1, ...)

## S4 method for signature 'SpatialDataArray'
flop(x, k = 1, ...)

## S4 method for signature 'SpatialDataArray'
rotate(x, t, k = 1, ..., rev = FALSE)

## S4 method for signature 'SpatialDataArray'
scale(x, t, ...)

## S4 method for signature 'SpatialDataArray,numeric'
translation(x, t, ...)

## S4 method for signature 'SpatialDataFrame'
rotate(x, t, ...)

## S4 method for signature 'SpatialDataFrame'
scale(x, t, ...)

## S4 method for signature 'SpatialDataFrame,numeric'
translation(x, t, ...)

## S4 method for signature 'SpatialDataFrame,matrix'
affine(x, t, ...)
```

Arguments

x	SpatialData element.
i	scalar integer or string; target coordinate space.
...	option arguments passed to and from other methods.
t	transformation data; exceptions: for mirror, controls whether to perform **v**ertical or **h**orizontal reflection; no data is needed for flip (**v**) and flop (**h**).
rev	flag; should transformation(s) be reversed?
k	scalar index specifying which scale to use; Inf to use lowest available resolution; only applies to SpatialDataArrays (images, labels).

Value

SpatialData element with transformation(s) applied.

Examples

```
x <- file.path("extdata", "blobs.zarr")
x <- system.file(x, package="spatialdataR")
x <- readSpatialData(x, tables=FALSE)

# image
y <- x
image(y) <- scale(image(y), c(1, 1, 1/3))
dim(image(x))
dim(image(y))

# point
y <- x
point(y, "rot") <- rotate(point(y), 20)
point(y, "wide") <- scale(point(y), c(1.2, 1))

xy0 <- centroids(point(y))
xy1 <- centroids(point(y, "rot"))
xy2 <- centroids(point(y, "wide"))

plot(xy0[, c(1, 2)], asp=1)
points(xy1[, c(1, 2)], col=2)
points(xy2[, c(1, 2)], col=4)

# shape
y <- x
shape(y, "rot") <- rotate(shape(y), 5)
shape(y, "wide") <- scale(shape(y), c(1.2, 1))
shape(y, "left") <- translation(shape(y), c(-5, 0))
y["shapes", c("rot", "wide", "left")]
```

Index

∗ **internal**
    reexports, 15
.DollarNames.SpatialDataShape
        (SpatialDataFrame), 22
.SpatialData *(SpatialData-class)*, 15
[,SpatialData,ANY,ANY,ANY-method
        (SpatialData-class), 15
[,SpatialDataFrame,ANY,ANY,ANY-method
        (SpatialDataFrame), 22
[,SpatialDataImage,ANY,ANY,ANY-method
        (SpatialDataArray), 17
[,SpatialDataLabel,ANY,ANY,ANY-method
        (SpatialDataArray), 17
[[,SpatialData,character,ANY-method
        (SpatialData-class), 15
[[,SpatialData,numeric,ANY-method
        (SpatialData-class), 15
[[,SpatialDataFrame,ANY,ANY-method
        (SpatialDataFrame), 22
[[<-,SpatialData,ANY,ANY,ANY-method
        (SpatialData-class), 15
[[<-,SpatialData,character,ANY,ANY-method
        (SpatialData-class), 15
[[<-,SpatialData,character,ANY-method
        (SpatialData-class), 15
[[<-,SpatialData,numeric,ANY-method
        (SpatialData-class), 15
$,SpatialData-method
        (SpatialData-class), 15
$,SpatialDataAttrs-method
        (SpatialDataAttrs), 20
$,SpatialDataPoint-method
        (SpatialDataFrame), 22
$,SpatialDataShape-method
        (SpatialDataFrame), 22
$<-,SpatialData-method
        (SpatialData-class), 15

addCT *(CTutils)*, 7
addCT,SpatialDataAttrs-method
        (CTutils), 7
addCT,SpatialDataElement-method
        (CTutils), 7
affine *(trans)*, 26
affine,SpatialDataFrame,matrix-method
        (trans), 26
as.data.frame,SpatialDataFrame-method
        (SpatialDataFrame), 22
axes *(CTutils)*, 7
axes,SpatialDataAttrs-method *(CTutils)*,
        7
axes,SpatialDataElement-method
        (CTutils), 7

blobs, 2

centroids, 3
centroids,ANY-method *(centroids)*, 3
centroids,SpatialDataLabel-method
        (centroids), 3
centroids,SpatialDataPoint-method
        (centroids), 3
centroids,SpatialDataShape-method
        (centroids), 3
channels *(SpatialDataArray)*, 17
channels,SpatialDataAttrs-method
        (SpatialDataArray), 17
channels,SpatialDataElement-method
        (SpatialDataArray), 17
channels,SpatialDataImage-method
        (SpatialDataArray), 17
colnames,SpatialData-method
        (SpatialData-class), 15
combine, 4
combine,list,missing-method *(combine)*, 4
combine,SpatialData,SpatialData-method
        (combine), 4
crop, 4, *13*
crop,SpatialData-method *(crop)*, 4
crop,SpatialDataArray-method *(crop)*, 4
crop,SpatialDataFrame-method *(crop)*, 4
CTdata *(CTutils)*, 7
CTdata,SpatialDataAttrs-method
        (CTutils), 7
CTdata,SpatialDataElement-method
        (CTutils), 7
CTgraph, 6
CTgraph,ANY-method *(CTgraph)*, 6

CTgraph,SpatialData-method (CTgraph), 6
CTgraph,SpatialDataElement-method (CTgraph), 6
CTlist (CTutils), 7
CTlist,SpatialDataAttrs-method (CTutils), 7
CTlist,SpatialDataElement-method (CTutils), 7
CTname (CTutils), 7
CTname,SpatialData-method (CTutils), 7
CTname,SpatialDataAttrs-method (CTutils), 7
CTname,SpatialDataElement-method (CTutils), 7
CTpath (CTgraph), 6
CTpath,ANY-method (CTgraph), 6
CTpath,SpatialData-method (CTgraph), 6
CTpath,SpatialDataElement-method (CTgraph), 6
CTplot (CTgraph), 6
CTtype (CTutils), 7
CTtype,SpatialDataAttrs-method (CTutils), 7
CTtype,SpatialDataElement-method (CTutils), 7
CTunit (CTutils), 7
CTunit,SpatialDataAttrs-method (CTutils), 7
CTunit,SpatialDataElement-method (CTutils), 7
CTutils, 7

data (SpatialData-class), 15
data,SpatialDataElement-method (SpatialData-class), 15
data_type (SpatialDataArray), 17
data_type,DelayedArray-method (SpatialDataArray), 17
data_type,SpatialDataArray-method (SpatialDataArray), 17
dim,SpatialDataArray-method (SpatialDataArray), 17
dim,SpatialDataFrame-method (SpatialDataFrame), 22

element (SpatialData-class), 15
element,SpatialData,ANY-method (SpatialData-class), 15
element,SpatialData,character-method (SpatialData-class), 15
element,SpatialData,missing-method (SpatialData-class), 15
element,SpatialData,numeric-method (SpatialData-class), 15
element<- (SpatialData-class), 15
element<-,SpatialData,character-method (SpatialData-class), 15
extent, 9
extent,SpatialData-method (extent), 9
extent,SpatialDataArray-method (extent), 9
extent,SpatialDataFrame-method (extent), 9

feature_key (SpatialDataAttrs), 20
feature_key,SpatialDataAttrs-method (SpatialDataAttrs), 20
feature_key,SpatialDataPoint-method (SpatialDataAttrs), 20
feature_key<- (SpatialDataAttrs), 20
feature_key<-,SpatialDataAttrs,character-method (SpatialDataAttrs), 20
filter (reexports), 15
filter(), *15*
filter.SpatialDataFrame (SpatialDataFrame), 22
flip (trans), 26
flip,SpatialDataArray-method (trans), 26
flop (trans), 26
flop,SpatialDataArray-method (trans), 26

geom_type (SpatialDataFrame), 22
geom_type,SpatialDataShape-method (SpatialDataFrame), 22
getTable (table-utils), 25
getTable,SpatialData,ANY-method (table-utils), 25
getTable,SpatialData,character-method (table-utils), 25

hasTable (table-utils), 25
hasTable,SpatialData,ANY-method (table-utils), 25
hasTable,SpatialData,character-method (table-utils), 25

image (SpatialData-class), 15
image<- (SpatialData-class), 15
imageNames (SpatialData-class), 15
imageNames<- (SpatialData-class), 15
images (SpatialData-class), 15
images,SpatialData-method (SpatialData-class), 15
images<- (SpatialData-class), 15
instance_key (SpatialDataAttrs), 20

instance_key,list-method
(SpatialDataAttrs), 20
instance_key,SingleCellExperiment-method
(SpatialDataAttrs), 20
instance_key,SpatialDataFrame-method
(SpatialDataAttrs), 20
instance_key,SpatialDataLabel-method
(SpatialDataAttrs), 20
instance_key<- (SpatialDataAttrs), 20
instance_key<-,SingleCellExperiment,character-method
(SpatialDataAttrs), 20
instance_key<-,SpatialDataAttrs,character-method
(SpatialDataAttrs), 20
instances (SpatialDataAttrs), 20
instances,SingleCellExperiment-method
(SpatialDataAttrs), 20
instances,SpatialDataLabel-method
(SpatialDataAttrs), 20
instances,SpatialDataPoint-method
(SpatialDataAttrs), 20
instances,SpatialDataShape-method
(SpatialDataAttrs), 20
instances<- (SpatialDataAttrs), 20
instances<-,SingleCellExperiment-method
(SpatialDataAttrs), 20

label (SpatialData-class), 15
label<- (SpatialData-class), 15
labelNames (SpatialData-class), 15
labelNames<- (SpatialData-class), 15
labels (SpatialData-class), 15
labels,SpatialData-method
(SpatialData-class), 15
labels<- (SpatialData-class), 15
layer (SpatialData-class), 15
layer,SpatialData,ANY-method
(SpatialData-class), 15
layer,SpatialData,character-method
(SpatialData-class), 15
length,SpatialDataArray-method
(SpatialDataArray), 17
length,SpatialDataFrame-method
(SpatialDataFrame), 22

mask, 10
mask,SpatialData-method (mask), 10
meta (SpatialData-class), 15
meta,SingleCellExperiment-method
(table-utils), 25
meta,SpatialDataElement-method
(SpatialData-class), 15
mirror (trans), 26
mirror,SpatialDataArray-method (trans),
26
misc, 11
mutate (reexports), 15
mutate(), *15*
mutate.SpatialDataFrame
(SpatialDataFrame), 22

names,SpatialDataFrame-method
(SpatialDataFrame), 22

path, 12
path,SingleCellExperiment-method
(path), 12
path,SpatialData-method (path), 12
path,SpatialDataArray-method (path), 12
path,SpatialDataFrame-method (path), 12
point (SpatialData-class), 15
point<- (SpatialData-class), 15
pointNames (SpatialData-class), 15
pointNames<- (SpatialData-class), 15
points (SpatialData-class), 15
points,SpatialData-method
(SpatialData-class), 15
points<- (SpatialData-class), 15
pull (reexports), 15
pull(), *15*
pull.SpatialDataFrame
(SpatialDataFrame), 22

query, 13
query,SpatialData-method (query), 13

readImage (readSpatialData), 13
readLabel (readSpatialData), 13
readPoint (readSpatialData), 13
readShape (readSpatialData), 13
readSpatialData, 13
readTable (readSpatialData), 13
reexports, 15
region (SpatialDataAttrs), 20
region,SingleCellExperiment-method
(SpatialDataAttrs), 20
region<- (SpatialDataAttrs), 20
region_key (SpatialDataAttrs), 20
region_key,SingleCellExperiment-method
(SpatialDataAttrs), 20
region_key<- (SpatialDataAttrs), 20
regions (SpatialDataAttrs), 20
regions,SingleCellExperiment-method
(SpatialDataAttrs), 20
regions<- (SpatialDataAttrs), 20
regions<-,SingleCellExperiment,character-method
(SpatialDataAttrs), 20

regions<-,SingleCellExperiment,NULL-method (SpatialDataAttrs), 20
rmvCT (CTutils), 7
rmvCT,SpatialDataAttrs-method (CTutils), 7
rmvCT,SpatialDataElement-method (CTutils), 7
rotate (trans), 26
rotate,SpatialDataArray-method (trans), 26
rotate,SpatialDataFrame-method (trans), 26
rownames,SpatialData-method (SpatialData-class), 15

scale (trans), 26
scale,SpatialDataArray-method (trans), 26
scale,SpatialDataFrame-method (trans), 26
select (reexports), 15
select(), *15*
select.SpatialDataFrame (SpatialDataFrame), 22
sequence (trans), 26
sequence,SpatialDataElement-method (trans), 26
setTable (table-utils), 25
setTable,SpatialData,ANY-method (table-utils), 25
setTable,SpatialData,character-method (table-utils), 25
shape (SpatialData-class), 15
shape<- (SpatialData-class), 15
shapeNames (SpatialData-class), 15
shapeNames<- (SpatialData-class), 15
shapes (SpatialData-class), 15
shapes,SpatialData-method (SpatialData-class), 15
shapes<- (SpatialData-class), 15
show,SpatialData-method (misc), 11
show,SpatialDataArray-method (misc), 11
SpatialData, *10–12*, *25*
SpatialData (SpatialData-class), 15
SpatialData-class, 15
SpatialDataArray, 17
SpatialDataAttrs, *18, 19*, 20, *22*, *24*
SpatialDataFrame, 22
SpatialDataImage, *17*
SpatialDataImage (SpatialDataArray), 17
SpatialDataLabel, *17*
SpatialDataLabel (SpatialDataArray), 17
SpatialDataPoint, *17*
SpatialDataPoint (SpatialDataFrame), 22
SpatialDataShape, *17*
SpatialDataShape (SpatialDataFrame), 22

table (SpatialData-class), 15
table-utils, 25
table<- (SpatialData-class), 15
tableNames (SpatialData-class), 15
tableNames<- (SpatialData-class), 15
tables (SpatialData-class), 15
tables,SpatialData-method (SpatialData-class), 15
tables<- (SpatialData-class), 15
trans, 26
transform (trans), 26
transform,SpatialDataElement-method (trans), 26
translation (trans), 26
translation,SpatialDataArray,numeric-method (trans), 26
translation,SpatialDataFrame,numeric-method (trans), 26