Package ‘SpatialOmicsOverlay’

June 5, 2026

Title Spatial Overlay for Omic Data from Nanostring GeoMx Data

Description Tools for NanoString Technologies GeoMx Technology. Package to easily graph on top of an OME-TIFF image. Plotting annotations can range from tissue segment to gene expression.

Version 1.13.0

Encoding UTF-8

Depends R (>= 4.1.0)

Imports S4Vectors, Biobase, base64enc, EBImage, ggplot2, XML, scattermore, dplyr, pbapply, data.table, readxl, magick, grDevices, stringr, plotrix, GeomxTools, BiocFileCache, stats, utils, methods, ggtext, tools, RBioFormats

Suggests knitr, BiocStyle, rmarkdown, testthat (>= 3.0.0), stringi, qpdf, pheatmap, viridis, cowplot, vdiffr, sf

License MIT

Collate addImage.R addPlottingFactor.R coordinateGeneration.R imageManipulation.R omeExtraction.R plottingBasics.R readSpatialOverlay.R removeSamples.R SpatialPosition-class.R SpatialOverlay-class.R utils.R xmlParsing.R

biocViews GeneExpression, Transcription, CellBasedAssays, DataImport, Transcriptomics, Proteomics, ProprietaryPlatforms, RNASeq, Spatial, DataRepresentation, Visualization

VignetteEngine knitr::rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

Config/testthat/edition 3

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

git_branch devel

git_last_commit 840708e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Maddy Griswold [cre, aut],
Megan Vandenberg [ctb],
Stephanie Zimmerman [ctb]

Maintainer Maddy Griswold <`mgriswold@nanostring.com`>

Contents

`add4ChannelImage` *Add 4-channel image to SpatialOverlay from OME-TIFF. Allows for recoloring of image*

Description

Add 4-channel image to SpatialOverlay from OME-TIFF. Allows for recoloring of image

Usage

```
add4ChannelImage(overlay, ometiff = NULL, res = NULL, ...)
```

Arguments

`overlay`	SpatialOverlay object
`ometiff`	File path to OME-TIFF. NULL indicates pull info from overlay
`res`	resolution layer, 1 = largest & higher values = smaller. The images increase in resolution and memory. The largest image your environment can hold is recommended. NULL indicates pull info from overlay
`...`	Extra variables

Value

SpatialOverlay object with image

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- add4ChannelImage(overlay = muBrain,
                            ometiff = image, res = 8)

dim(EBImage::imageData(showImage(muBrain)))
```

`addImageFile` *Add image to SpatialOverlay from disk*

Description

Add image to SpatialOverlay from disk

Usage

```
addImageFile(overlay, imageFile = NULL, res = NULL)
```

Arguments

`overlay`	SpatialOverlay object
`imageFile`	path to image
`res`	what resolution is the image given? 1 = largest, higher number = smaller This value will affect the coordinates of the overlays. res = 2, resolution is 1/2 the size as the raw image res = 3, resolution is 1/4 the size as the raw image res = 4, resolution is 1/8 the size as the raw image resolution = 1/2^(res-1)

Value

SpatialOverlay object with image

`addImageOmeTiff` *Add image to SpatialOverlay from OME-TIFF*

Description

Add image to SpatialOverlay from OME-TIFF

Usage

```
addImageOmeTiff(overlay, ometiff = NULL, res = NULL, ...)
```

Arguments

`overlay`	SpatialOverlay object
`ometiff`	File path to OME-TIFF. NULL indicates pull info from overlay
`res`	resolution layer, 1 = largest & higher values = smaller. The images increase in resolution and memory. The largest image your environment can hold is recommended. NULL indicates pull info from overlay
`...`	Extra variables

Value

SpatialOverlay object with image

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- addImageOmeTiff(overlay = muBrain,
                           ometiff = image, res = 8)
showImage(muBrain)
```

`addPlottingFactor` *Add plotting factor to `SpatialOverlay` object*

Description

Add plotting factor to `SpatialOverlay` object

Usage

```
addPlottingFactor(overlay, annots, plottingFactor, ...)

## S4 method for signature 'NanoStringGeoMxSet'
addPlottingFactor(overlay, annots, plottingFactor, countMatrix = "exprs")

## S4 method for signature 'matrix'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'tbl_df'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'tbl'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'data.frame'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'character'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'numeric'
addPlottingFactor(overlay, annots, plottingFactor)

## S4 method for signature 'factor'
addPlottingFactor(overlay, annots, plottingFactor)
```

Arguments

`overlay`	`SpatialOverlay` object
`annots`	factor vector with the plottingFactor. if names match sample names in overlay vector will be matched on those, otherwise assumed in the correct order
`plottingFactor`	name of the new plotting factor
`...`	if using `NanoStringGeoMxSet`, name of count matrix to pull counts from
`countMatrix`	name of count matrix to pull counts from

Value

`SpatialOverlay` object with new plotting factor

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

muBrainLW <- system.file("extdata", "muBrain_LabWorksheet.txt",
                         package = "SpatialOmicsOverlay")

muBrainLW <- readLabWorksheet(muBrainLW, slideName = "D5761 (3)")

muBrain <- addPlottingFactor(overlay = muBrain,
                             annots = muBrainLW,
```

```
                              plottingFactor = "segment")

muBrainGxT <- readRDS(unzip(system.file("extdata", "muBrain_GxT.zip",
                                   package = "SpatialOmicsOverlay")))

muBrain <- addPlottingFactor(overlay = muBrain,
                              annots = muBrainGxT,
                              plottingFactor = "Calm1",
                              countMatrix = "exprs")

muBrain <- addPlottingFactor(overlay = muBrain,
                              annots = seq_len(length(sampNames(muBrain))),
                              plottingFactor = "ROINum")

head(plotFactors(muBrain))

muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

muBrain <- addPlottingFactor(overlay = muBrain,
                              annots = as.factor(seq_len(length(sampNames(muBrain)))),
                              plottingFactor = "ROINum")

head(plotFactors(muBrain))
```

`annotMatching` *Match ROIs in annotation file to xml*

Description

Match ROIs in annotation file to xml

Usage

```
annotMatching(annots, ROInum, maskNum, maskText, segCol = NULL)
```

Arguments

`annots`	df of annotations
`ROInum`	ROI number from xml file
`maskNum`	number of masks for ROI, used for AOI matching in software <= v2.4
`maskText`	segment name, used for AOI matching in software v2.4+
`segCol`	column containing segment name, if NULL function will determine automatically

Value

df with ROI unique identifiers

`bookendStr` *Print long string in more managable fashion*

Description

Print first and last n characters of string in this format: "### ... ### (x total char)"

Usage

```
bookendStr(x, bookend = 8)
```

Arguments

`x`	long string
`bookend`	number of characters on either side to print

Value

reformatted string

Examples

```
start_string <- stringi::stri_rand_strings(n = 1, length = 250)
bookendStr(start_string, bookend = 6)
```

`changeColoringIntensity` *Update color intensities for changing to RGB image*

Description

Update color intensities for changing to RGB image

Usage

```
changeColoringIntensity(overlay, minInten = NULL, maxInten = NULL, dye)
```

Arguments

`overlay`	SpatialOverlay object
`minInten`	value to change MinIntensity to; NULL indicates no change
`maxInten`	value to change MaxIntensity to; NULL indicates no change
`dye`	which dye to change color, can be from Dye or DisplayName column from fluor(overlay)

Value

SpatialOverlay object with updated fluor data

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- add4ChannelImage(overlay = muBrain,
                            ometiff = image, res = 8)

fluor(muBrain)

muBrain <- changeColoringIntensity(overlay = muBrain,
                                   minInten = 500,
                                   maxInten = 10000,
                                   dye = "Cy5")

fluor(muBrain)
```

`changeImageColoring` *Update color scheme for changing to RGB image*

Description

Update color scheme for changing to RGB image

Usage

```
changeImageColoring(overlay, color, dye)
```

Arguments

`overlay`	SpatialOverlay object, with 4channel image
`color`	color to change dye to, hex or color name
`dye`	which dye to change color, can be from Dye or DisplayName column from fluor(overlay)

Value

SpatialOverlay object with updated fluor data

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

muBrain <- add4ChannelImage(overlay = muBrain,
                            ometiff = downloadMouseBrainImage(), res = 8)

fluor(muBrain)

muBrain <- changeImageColoring(overlay = muBrain, color = "magenta",
                               dye = "Cy5")
```

```
muBrain <- changeImageColoring(overlay = muBrain, color = "#42f5ef",
                               dye = "Alexa 488")

fluor(muBrain)
```

`checkValidRes` *Determine lowest resolution image in OME-TIFF*

Description

Determine lowest resolution image in OME-TIFF

Usage

```
checkValidRes(ometiff)
```

Arguments

`ometiff`	path to OME-TIFF

Value

value of lowest res image

Examples

```
image <- downloadMouseBrainImage()
checkValidRes(ometiff = image)
```

`createCoordFile` *Create coordinate file for entire scan*

Description

Create coordinate file for entire scan

Usage

```
createCoordFile(overlay, outline = TRUE)
```

Arguments

`overlay`	SpatialOverlay object
`outline`	returned coordinates only contain boundaries, will not work for segmented ROIs

Value

df of coordinates for every AOI in the scan

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

muBrain <- createCoordFile(muBrain, outline = FALSE)

head(coords(muBrain))
```

`createMask` *Create a binary mask from a base 64 string*

Description

Create a binary mask from a base 64 string

Usage

```
createMask(b64string, metadata, outline = TRUE)
```

Arguments

`b64string`	base 64 string
`metadata`	metadata of AOI including: Height, Width of AOI
`outline`	only the outline points should be returned

Value

binary mask image

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

samp <- which(sampNames(muBrain) == "DSP-1012999073013-A-F12")

ROIMask <- createMask(b64string = position(overlay(muBrain))[samp],
                      metadata = meta(overlay(muBrain))[samp,],
                      outline = TRUE)

pheatmap::pheatmap(ROIMask, cluster_rows = FALSE, cluster_cols = FALSE)
```

`cropSamples` *Crop to zoom in on given ROIs*

Description

Crop to zoom in on given ROIs

Usage

```
cropSamples(overlay, sampleIDs, buffer = 0.1, sampsOnly = TRUE)
```

Arguments

`overlay`	SpatialOverlay object
`sampleIDs`	sampleIDs of ROIs to keep in image
`buffer`	percent of new image size to add to each edge as a buffer
`sampsOnly`	should only ROIs with given sampleIDs be in image

Value

SpatialOverlay object

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- addImageOmeTiff(overlay = muBrain,
                           ometiff = image, res = 8)

samps <- sampNames(muBrain)

muBrainCrop <- suppressWarnings(cropSamples(overlay = muBrain,
                                            sampleIDs = samps,
                                            sampsOnly = TRUE))

plotSpatialOverlay(overlay = muBrainCrop, scaleBar = FALSE,
                   hiRes = TRUE, legend = FALSE)
```

`cropTissue` *Crop to remove black boundary around tissue.*

Description

Crop to remove black boundary around tissue.

Usage

```
cropTissue(overlay, buffer = 0.05)
```

Arguments

`overlay`	SpatialOverlay object
`buffer`	percent of new image size to add to each edge as a buffer

Value

SpatialOverlay object

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- addImageOmeTiff(overlay = muBrain,
                           ometiff = image, res = 8)

muBrainCrop <- cropTissue(overlay = muBrain)

plotSpatialOverlay(overlay = muBrainCrop, legend = FALSE,
                   hiRes = FALSE, scaleBar = FALSE)
```

`downloadMouseBrainImage` *Download Mouse Brain OME-TIFF from NanoString's Spatial Organ Atlas*

Description

Download Mouse Brain OME-TIFF from NanoString's Spatial Organ Atlas

Usage

```
downloadMouseBrainImage()
```

Details

https://nanostring.com/products/geomx-digital-spatial-profiler/spatial-organ-atlas/mouse-brain/

Value

mouse brain OME-TIFF

Examples

```
image <- downloadMouseBrainImage()
```

`flipX` *Flip x axis in image and overlay points*

Description

Flip x axis in image and overlay points

Usage

```
flipX(overlay)
```

Arguments

`overlay`	SpatialOverlay object

Value

SpatialOverlay object with x axis flipped

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- addImageOmeTiff(overlay = muBrain,
                           ometiff = image, res = 8)

showImage(flipX(muBrain))
```

`flipY` *Flip y axis in image and overlay points*

Description

Flip y axis in image and overlay points

Usage

```
flipY(overlay)
```

Arguments

`overlay`	SpatialOverlay object

Value

SpatialOverlay object with y axis flipped

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- addImageOmeTiff(overlay = muBrain,
                           ometiff = image, res = 8)

showImage(flipY(muBrain))
```

`fluorLegend` *Add legend of fluorescence targets that make up image*

Description

Creates legend that can be overlayed on image using `cowplot`.

Usage

```
fluorLegend(overlay, nrow = 4, textSize = 10, boxColor = "grey", alpha = 0.25)
```

Arguments

`overlay`	SpatialOverlay
`nrow`	number of rows in the legend. Most studies have 4 which is where the values came from: 1 = horizontal legend, 4 = vertical legend, 2 = box legend
`textSize`	font size
`boxColor`	color of box behind legend
`alpha`	alpha value of box behind legend

Value

gp of fluorescence legend

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))

# image <- downloadMouseBrainImage()

# muBrain <- addImageOmeTiff(overlay = muBrain,
#                            ometiff = image, res = 8)

gp <- plotSpatialOverlay(overlay = muBrain,
                         hiRes = FALSE, scaleBar = FALSE)

legend <- fluorLegend(muBrain, nrow = 2, textSize = 3, boxColor = "red")

cowplot::ggdraw() +
    cowplot::draw_plot(gp) +
    cowplot::draw_plot(legend, scale = 0.12, x = -0.3, y = -0.25)
```

`moveCoords` *Move coordinates if they don't match image*

Description

If generated coordinates do not match the image use this function to move coordinates. Coordinates are only changed 1 pixel at a time.

Usage

```
moveCoords(overlay, direction = "right")
```

Arguments

`overlay`	SpatialOverlay object
`direction`	which direction should coordinates move: left, right, up, down

Value

SpatialOverlay object

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                    package = "SpatialOmicsOverlay")))
head(coords(muBrain), 3)
head(coords(moveCoords(muBrain, direction = "up")), 3)
```

`parseOverlayAttrs` *Parse the xml file for AOI attributes in GeoMx images*

Description

Parse the xml file for AOI attributes in GeoMx images

Usage

```
parseOverlayAttrs(omexml, annots, labworksheet, ...)
```

Arguments

`omexml`	xml file from OME-TIFF, can provide path to OME-TIFF and xml will automatically be extracted
`annots`	df of annotations
`labworksheet`	annots are from lab worksheet file
`...`	segCol in annotMatching, if auto detection doesn't work.

Value

SpatialPosition of AOIs containing metadata and base64encoded positions

Examples

```
image <- downloadMouseBrainImage()

xml <- xmlExtraction(ometiff = image)

muBrainLW <- system.file("extdata", "muBrain_LabWorksheet.txt",
                         package = "SpatialOmicsOverlay")

muBrainLW <- readLabWorksheet(muBrainLW, slideName = "D5761 (3)")

overlay <- parseOverlayAttrs(omexml = xml,
                             annots = muBrainLW,
                             labworksheet = TRUE)
```

`parseScanMetadata` *Parse the xml file for the scan metadata of GeoMx images*

Description

Parse the xml file for the scan metadata of GeoMx images

Usage

```
parseScanMetadata(omexml)
```

Arguments

`omexml`	xml file from OME-TIFF, can provide path to OME-TIFF and xml will automatically be extracted

Value

metadata for entire scan

Examples

```
image <- downloadMouseBrainImage()

xml <- xmlExtraction(ometiff = image)

scan_metadata <- parseScanMetadata(omexml = xml)
```

`plotSpatialOverlay` *overlay plots*

Description

overlay plots

Usage

```
plotSpatialOverlay(
  overlay,
  colorBy = "sampleID",
  hiRes = TRUE,
  alpha = 1,
  legend = TRUE,
  scaleBar = TRUE,
  image = TRUE,
  fluorLegend = FALSE,
  ...,
  corner = "bottomright",
  scaleBarWidth = 0.2,
  scaleBarMicrons = NULL,
  scaleBarColor = NULL,
  scaleBarFontSize = 6,
  scaleBarLineSize = 1.5,
  textDistance = 2
)
```

Arguments

`overlay`	SpatialOverlay object
`colorBy`	annotation to color by
`hiRes`	generated figures are either high resolution or print quickly. Note: hiRes and outline ggplots use fill, lowRes uses color

`alpha`	opacity of overlays
`legend`	should legend be plotted
`scaleBar`	should scale bar be plotted
`image`	should image be plotted, image must be added to SpatialOverlay object
`fluorLegend`	should viz marker on the image be added to legend
`...`	additional parameters for scale bar line & text, will affect both
`corner`	where in the figure should the scale bar be printed. Options: "bottomright" "topright" "bottomleft" "topleft" "bottomcenter" "topcenter"
`scaleBarWidth`	percent of total figure the scale bar should take up
`scaleBarMicrons`	specific microns to set scale bar at, overrides scaleBarWidth if set
`scaleBarColor`	scale bar & text color
`scaleBarFontSize`	font size
`scaleBarLineSize`	width of line
`textDistance`	text's distance from scale bar.

Value

gp

Note

hiRes and outline ggplots use fill, lowRes uses color

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

plotSpatialOverlay(overlay = muBrain, legend = FALSE,
                   hiRes = FALSE, scaleBar = FALSE)
```

`readLabWorksheet` *Read lab worksheet into dataframe of annotations*

Description

Read lab worksheet into dataframe of annotations

Usage

```
readLabWorksheet(lw, slideName, roiCol = NULL, slideCol = NULL)
```

Arguments

`lw`	lab worksheet file path
`slideName`	name of slide
`roiCol`	column containing ROI information, if NULL function will determine automatically
`slideCol`	column containing slide name, if NULL function will determine automatically

Value

df of ROI annotations

Examples

```
muBrainLW <- system.file("extdata", "muBrain_LabWorksheet.txt",
                         package = "SpatialOmicsOverlay")

muBrainLW <- readLabWorksheet(muBrainLW, slideName = "D5761 (3)")
```

`readSpatialOverlay` *Read in* `SpatialOverlay` *from tiff file and annotations*

Description

Create an instance of class `SpatialOverlay` by reading data from OME-TIFF and annotation sheet.

Usage

```
readSpatialOverlay(
  ometiff,
  annots,
  slideName,
  image = FALSE,
  res = NULL,
  saveFile = FALSE,
  outline = TRUE,
  ...,
  segCol = NULL
)
```

Arguments

`ometiff`	path to OME-TIFF
`annots`	path to annotation file: can be labWorksheet, DA excel file, or delimted file
`slideName`	name of slide
`image`	should image be extracted from OME-TIFF
`res`	resolution of image 1 = largest, higher number = smaller This value will affect the coordinates of the overlays. res = 2, resolution is 1/2 the size as the raw image res = 3, resolution is 1/4 the size as the raw image res = 4, resolution is 1/8 the size as the raw image resolution = 1/2^(res-1)

`saveFile`	should xml & image be saved, file is saved in working directory with same name as OME-TIFF
`outline`	returned coordinates only contain outlinearies, will not work for segmented ROIs
`...`	additional parameters for ‘readLabWorksheet‘ like ‘roiCol‘ and ‘slideCol‘
`segCol`	additional parameter for ‘annotMatching‘ if default search doesn’t work. For default search, set to NULL

Value

`SpatialOverlay` of slide

See Also

`SpatialOverlay-class`

Examples

```
muBrain_GxT <- readRDS(unzip(system.file("extdata", "muBrain_GxT.zip",
                          package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- readSpatialOverlay(ometiff = image, annots = muBrain_GxT[,1:5],
                              slideName = "D5761 (3)", image = TRUE, res = 8,
                              saveFile = FALSE, outline = FALSE)
```

`recolor` *recolor images after changing colors and/or color intensities*

Description

recolor images after changing colors and/or color intensities

Usage

```
recolor(overlay)
```

Arguments

`overlay`	SpatialOverlay object

Value

SpatialOverlay object with RGB image

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

image <- downloadMouseBrainImage()

muBrain <- add4ChannelImage(overlay = muBrain,
                            ometiff = image, res = 8)

muBrain <- changeImageColoring(overlay = muBrain, color = "magenta",
                               dye = "Cy5")
showImage(recolor(muBrain))
```

`removeSample` *Remove sample(s) from SpatialOverlay*

Description

Remove sample(s) from SpatialOverlay

Usage

```
removeSample(overlay, remove)
```

Arguments

`overlay`	SpatialOverlay object
`remove`	sampNames of overlay to remove

Value

SpatialOverlay object without samples in remove

Examples

```
muBrain <- readRDS(unzip(system.file("extdata", "muBrainSubset_SpatialOverlay.zip",
                                     package = "SpatialOmicsOverlay")))

muBrainLW <- system.file("extdata", "muBrain_LabWorksheet.txt",
                         package = "SpatialOmicsOverlay")

muBrainLW <- readLabWorksheet(muBrainLW, slideName = "D5761 (3)")

samps <- muBrainLW$Sample_ID[muBrainLW$segment != "Full ROI"]

muBrainSub <- removeSample(overlay = muBrain, remove = samps)

muBrain
muBrainSub
```

`SpatialOverlay-class` *Class to Contain NanoString Spatial Overlay Images and Data*

Description

The `SpatialOverlay` class organizes the pertinant information from the OME-TIFFs allowing for plotting on top of or beside the image

Usage

```
SpatialOverlay(slideName,
               scanMetadata,
               overlayData,
               coords = NULL,
               plottingFactors = NULL,
               workflow = list(outline=FALSE,
                               labWorksheet=TRUE,
                               scaled=FALSE),
               image = list(filePath = NULL,
                            imagePointer = NULL,
                            resolution = NULL))
```

Arguments

`slideName`	The name of the slide in the `SpatialOverlay` object
`scanMetadata`	A `list` containing the scan metadata: panel(s) used, Physical sizes (x,y) for scale bar, fluorescence data, and segmentation info
`overlayData`	An `SpatialPosition` containing individiual sample info: SampleID, Height, Width, X&Y coordinates in overall scan, segmentation, and base64 encoded position
`coords`	An optional `data.frame` with coordinates derived from the encoded position.
`plottingFactors`	An optional `data.frame` with annotations to plot on. These can be added using `addPlottingFactor`
`workflow`	A `list` containing meta-data on the processing workflow including `"outline"`, `"labWorksheet"`, & `"scaled"`. These booleans are kept automatically in `readSpatialOverlay` and are used for logic gates in downstream processing
`image`	A `list` containing location of and pointer to the image: `"filePath"`, `"imagePointer"`, & `"resolution"`

Value

An S4 class containing image data from a NanoString GeoMx experiment

Accessing

`SpatialOverlay` objects have the following accessor methods:

sampNames(object) extracts the sample names of each ROI in the slide.

slideName(object) extracts the slide name.

overlay(object) extracts the `SpatialPosition` information for each ROI.

scanMeta(object) extracts the scan metadata.

scanMeta(object) `coords(object)`: extracts the coordinates for the entire scan.

plotFactors(object) extracts avaliable plotting factors.

labWork(object) extracts the boolean if a lab worksheet was used.

outline(object) extracts the boolena if only the outline points were generated.

seg(object) extracts if there are any segmented ROIs in the slide.

scaleBarRatio(object) extracts the scale bar ratio from `scanMeta` for the X axis

fluor(object) extracts fluorenscence information for the scan.

showImage(object) prints image.

res(object) extracts resolution of image.

workflow(object) extracts workflow data.

scaled(object) extracts if coordinates have been scaled.

imageInfo(object) extracts image data.

See Also

`readSpatialOverlay`

`SpatialPosition-class` *Class to Contain NanoString Spatial Overlay ROI information*

Description

The `SpatialPosition` class organizes the pertinant ROI specific information from the OME-TIFFs

Usage

```
SpatialPosition(position)
```

Arguments

`position`	`data.frame` containing information from the OME-TIFF: `"ROILabel"` order of ROIs `"Sample_ID"` unique identifier `"Height"` total height of ROI `"Width"` total width of ROI `"X"` top left corner (x coordinate) of ROI in total scan `"Y"` top left corner (y coordinate) of ROI in total scan `"Segmentation"` part of segmented ROI `"Position"` base64 encoding of coordinates

Value

An S4 class containing image data from a NanoString GeoMx experiment

Accessing

`SpatialPosition` objects have the following accessor methods:

spatialPos(object) returns SpatialPosition object

meta(object) extracts the metadata for each ROI, does not include the base64 encoding.

position(object) extracts the base64 encoding for each ROI, CAUTION: very long strings.

See Also

`SpatialOverlay-class`

`xmlExtraction` *Extract xml from OME-TIFF*

Description

Extract xml from OME-TIFF

Usage

```
xmlExtraction(ometiff, saveFile = FALSE, outdir = NULL)
```

Arguments

`ometiff`	path to OME-TIFF
`saveFile`	should xml be saved, file is saved in working directory with same name as OME-TIFF
`outdir`	output directory for saved xml. If NULL, saved in same directory as OME-TIFF

Value

list of xml data

Examples

```
image <- downloadMouseBrainImage()
xml <- xmlExtraction(ometiff = image)
```

Index