Package: spiSEE (via r-universe)

November 18, 2026

Title Interactive Exploration of Cell-Resolution Spatial Transcriptomics with iSEE

Version 0.99.2

Description Provides reusable iSEE panels for interactive exploration of cell-resolution spatial transcriptomics data stored in SpatialExperiment objects. Supports CosMx, Xenium and cell-segmented Visium HD data, with coordinated region selection across spatial visualisations, quality-control summaries and spatial analyses. Panels can be combined with standard iSEE panels to explore gene expression, spatial statistics, cellular neighbourhoods and spatial domains.

License MIT + file LICENSE

URL https://github.com/RighelliLab/spiSEE

BugReports https://github.com/RighelliLab/spiSEE/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.5.0), iSEE, SpatialExperiment

Imports methods, stats, utils, grDevices, grid, tools, BiocGenerics, S4Vectors, SummarizedExperiment, SingleCellExperiment, SpatialFeatureExperiment, SpaceTrooper (>= 1.3.1), Voyager, spatialFDA, spdep, BiocNeighbors, Matrix, igraph, scater, ggplot2, pheatmap, pals, plotly, htmlwidgets, base64enc, png, digest, cachem, DT, sf, shiny (>= 1.8.1), htmltools, withr

Suggests OSTA.data, SpatialExperimentIO, VisiumIO, BiocIO, BiocStyle, knitr, rmarkdown, scuttle, svglite, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Software, Spatial, Transcriptomics, SingleCell, Visualization, GUI, ShinyApps, Clustering, QualityControl, GeneExpression, DataImport

Config/testthat/edition 3

Config/pak/sysreqs libabsl-dev cmake libfftw3-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev libgmp3-dev make libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libxml2-dev libssl-dev perl libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-11-18 23:32:20 UTC

RemoteUrl https://github.com/BiocStaging/spiSEE

RemoteRef HEAD

RemoteSha 78171358e86cfd66d82ffb1e0087ae4f10183fc1

Contents

buildSpatialGraph . . . 2
computeNiches . . . 3
computeSpatialDomains . . . 5
FOVStatsPanel-class . . . 6
FOVZoomPanel-class . . . 8
loadCosmx . . . 9
loadVisiumHD . . . 10
loadXenium . . . 10
plotSpatialCells . . . 11
setPlatform . . . 12
spatialCacheDir . . . 13
SpatialMapPanel-class . . . 14
SpatialNichePanel-class . . . 15
SpatialPanelBase-class . . . 17
SpatialPlotPanel-class . . . 18
SpatialQCPanel-class . . . 19
SpatialStatsPanel-class . . . 21
spiSEE . . . 23
SVGPanel-class . . . 24

Index **27**

`buildSpatialGraph` *Build the spatial k-NN graph once and store it on the object*

Description

Computes a k-nearest-neighbour graph on `spatialCoords(spe)` and stores it as a column pairing (`colPair(spe, name)`). Because column pairings are subset together with the cells, any region selected later reuses this graph instead of rebuilding it.

Usage

```
buildSpatialGraph(
  spe,
  k = 20L,
  name = "spatial_knn",
  BNPARAM = BiocNeighbors::KmknnParam()
)
```

Arguments

spe	A SpatialExperiment.
k	Number of neighbours per cell. The default, 20, is that of computeNiches(), so the graph is reused there.
name	Name of the stored column pairing.
BNPARAM	A BiocNeighborParam; use BiocNeighbors::AnnoyParam() for very large datasets.

Value

spe with colPair(spe, name) set.

See Also

computeNiches(), computeSpatialDomains().

Examples

```
# two zones of cell types on a 20 x 20 grid
xy <- as.matrix(expand.grid(x = 1:20, y = 1:20))
type <- ifelse(xy[, "x"] <= 10,
    c("Tumor", "T.cells")[(seq_len(400) %% 5 == 0) + 1],
    c("Fibroblast", "T.cells")[(seq_len(400) %% 4 == 0) + 1])
counts <- matrix(seq_len(400 * 10) %% 7, 10, 400,
    dimnames = list(paste0("g", 1:10), paste0("c", 1:400)))
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = counts, logcounts = log1p(counts)),
    colData = S4Vectors::DataFrame(Level1 = type), spatialCoords = xy)
spe <- buildSpatialGraph(spe, k = 8)
SingleCellExperiment::colPairNames(spe)
```

---

computeNiches *Cellular niches from the spatial neighbour graph*

Description

For every cell, computes the cell-type composition of its k nearest spatial neighbours, then groups cells with similar surroundings by k-means. Adds to colData: niche (factor, named after the two most enriched cell types) and niche_entropy (Shannon entropy of the neighbourhood, bits). Stores the composition as reducedDim(spe, "NicheComposition") and the niche x cell-type log2 enrichment in metadata(spe)$spiSEE_niches.

Usage

```
computeNiches(
  spe,
  label_col,
  k = 20L,
  n_niches = 6L,
  include_self = FALSE,
  seed = 1L
)
```

Arguments

spe	A SpatialExperiment.
label_col	colData column with cell-type labels.
k	Neighbours per cell (default 20, as buildSpatialGraph() and the niche panel).
n_niches	Number of niches (k-means centres).
include_self	Count the cell itself in its neighbourhood.
seed	Random seed for k-means.

Value

spe with niche annotations added. Cells with an NA label get NA niche.

See Also

buildSpatialGraph(), SpatialNichePanel().

Examples

```
# two zones of cell types on a 20 x 20 grid
xy <- as.matrix(expand.grid(x = 1:20, y = 1:20))
type <- ifelse(xy[, "x"] <= 10,
    c("Tumor", "T.cells")[(seq_len(400) %% 5 == 0) + 1],
    c("Fibroblast", "T.cells")[(seq_len(400) %% 4 == 0) + 1])
counts <- matrix(seq_len(400 * 10) %% 7, 10, 400,
    dimnames = list(paste0("g", 1:10), paste0("c", 1:400)))
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = counts, logcounts = log1p(counts)),
    colData = S4Vectors::DataFrame(Level1 = type), spatialCoords = xy)
spe <- computeNiches(spe, label_col = "Level1", k = 8, n_niches = 2)
table(spe$niche, spe$Level1)
```

`computeSpatialDomains` *Spatial domains: clustering on neighbourhood-augmented expression*

Description

Clusters cells on their expression PCs concatenated with the average PCs of their spatial neighbours (weight `lambda`; a simplified version of the BANKSY idea). With `lambda = 0` this is plain expression clustering, which is also returned (`expr_cluster`) so the two can be compared side by side.

Usage

```
computeSpatialDomains(
  spe,
  k_spatial = 20L,
  lambda = 0.2,
  use_dimred = "PCA",
  n_pcs = 20L,
  k_graph = 20L,
  resolution = 0.8,
  seed = 1L
)
```

Arguments

`spe`	A `SpatialExperiment` with a `logcounts` assay (or a `reducedDim` named by `use_dimred`).
`k_spatial`	Spatial neighbours used for the neighbourhood average (default 20, as `buildSpatialGraph()`, whose graph is then reused).
`lambda`	Share (0-1) of the neighbourhood part.
`use_dimred`	Name of an existing `reducedDim` to use; if absent, PCA is computed on `logcounts` with `scater::calculatePCA()`.
`n_pcs`	Number of PCs when computing PCA.
`k_graph`	Neighbours for the feature-space clustering graph.
`resolution`	Louvain resolution (higher = more domains).
`seed`	Random seed.

Value

`spe` with `colData` columns `spatial_domain` and `expr_cluster`. The parameters are stored in `metadata(spe)$spiSEE_domains`; *Compute* in `SpatialNichePanel()` reuses `resolution` and `k_graph`.

See Also

`buildSpatialGraph()`, `SpatialNichePanel()`.

Examples

```
# two zones of cell types on a 20 x 20 grid
xy <- as.matrix(expand.grid(x = 1:20, y = 1:20))
type <- ifelse(xy[, "x"] <= 10,
    c("Tumor", "T.cells")[(seq_len(400) %% 5 == 0) + 1],
    c("Fibroblast", "T.cells")[(seq_len(400) %% 4 == 0) + 1])
counts <- matrix(seq_len(400 * 10) %% 7, 10, 400,
    dimnames = list(paste0("g", 1:10), paste0("c", 1:400)))
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = counts, logcounts = log1p(counts)),
    colData = S4Vectors::DataFrame(Level1 = type), spatialCoords = xy)
spe <- computeSpatialDomains(spe, k_spatial = 8, n_pcs = 3, k_graph = 10)
table(spe$spatial_domain)
```

FOVStatsPanel-class *Per-FOV QC summary (CosMx)*

Description

iSEE panel with one boxplot per field of view (FOV) for each of several QC metrics, on the FOVs selected on the SpatialMapPanel() (all FOVs when nothing is selected). Useful to spot FOVs with weak staining or few transcripts before filtering.

Usage

```
## S4 method for signature 'FOVStatsPanel'
.fullName(x)

## S4 method for signature 'FOVStatsPanel'
.panelColor(x)

## S4 method for signature 'FOVStatsPanel'
.defineOutput(x)

## S4 method for signature 'FOVStatsPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'FOVStatsPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'FOVStatsPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'FOVStatsPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

FOVStatsPanel(...)
```

Arguments

x	A panel instance (the class this page documents).
se	The SpatialExperiment being explored.
select_info	A list of selection information, passed by iSEE.
...	Slots, e.g. Selection.
all_memory, all_contents	iSEE's full panel-state list and the current selection contents of every panel, passed to .generateOutput() for export/snapshot but unused by this package's own panels (selection is handled client-side, see SpatialMapPanel).
output, pObjects, rObjects	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.
input, session	The current Shiny input and session objects.

Details

By default the first six QC metrics of colData are shown, on their own scale. A table in metadata(spe)$fov_stats replaces them, for instance to show transformed values: a data.frame in long format with one row per cell and metric and the columns fov (FOV id, as in spe$fov), name (metric name, one facet each) and value (number).

Value

A FOVStatsPanel object.

See Also

SpatialQCPanel() for the distribution of one metric, FOVZoomPanel() for the cells of the selected FOVs.

Examples

```
FOVStatsPanel()

# optional: transformed values, one row per cell and metric
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = matrix(1, 2, 6)),
    colData = S4Vectors::DataFrame(fov = rep(1:2, 3), total = 1:6),
    spatialCoords = matrix(runif(12), 6, 2,
        dimnames = list(NULL, c("x", "y"))))
S4Vectors::metadata(spe)$fov_stats <- data.frame(fov = spe$fov,
    name = "log10 total", value = log10(spe$total))
```

`FOVZoomPanel-class` *Zoom on selected FOVs (CosMx)*

Description

iSEE panel drawing the cell polygons of the FOVs selected on the `SpatialMapPanel()`, coloured by a QC metric, with `SpaceTrooper::plotZoomFovsMap()`. Needs a `polygons` column.

Usage

```
## S4 method for signature 'FOVZoomPanel'
.fullName(x)

## S4 method for signature 'FOVZoomPanel'
.panelColor(x)

## S4 method for signature 'FOVZoomPanel'
.defineOutput(x)

## S4 method for signature 'FOVZoomPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'FOVZoomPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'FOVZoomPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'FOVZoomPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

FOVZoomPanel(ColorBy = "", ...)
```

Arguments

`x`	A panel instance (the class this page documents).
`se`	The `SpatialExperiment` being explored.
`select_info`	A list of selection information, passed by iSEE.
`...`	Other slots, e.g. `Selection`.
`all_memory`, `all_contents`	iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).
`output`, `pObjects`, `rObjects`	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.

input, session	The current Shiny input and session objects.
ColorBy	QC metric used to colour cell polygons. Empty = the first QC metric.

Value

A FOVZoomPanel object.

See Also

FOVStatsPanel(), SpatialQCPanel().

Examples

```
FOVZoomPanel(ColorBy = "QScore")
```

---

loadCosmx *Load a prepared CosMx dataset*

Description

Reads a CosMx SpatialExperiment saved as RDS and records the platform. Such an object is typically made with SpaceTrooper::readCosmxSPE() and the QC steps shown in the vignette *CosMx workflow: the mouse dentate gyrus*. Optional extras used by the app are described in spiSEE() (section *Optional metadata*).

Usage

```
loadCosmx(rds_path)
```

Arguments

rds_path	Path to the .RDS file.

Value

A SpatialExperiment.

See Also

loadXenium(), loadVisiumHD(), spiSEE().

Examples

```
dir <- system.file("extdata", "CosMx_DBKero_Tiny", package = "SpaceTrooper")
f <- tempfile(fileext = ".rds")
saveRDS(SpaceTrooper::readCosmxSPE(dir, sampleName = "DBKero"), f)
spe <- loadCosmx(f)
```

loadVisiumHD *Load 10x Visium HD cell-segmented outputs*

Description

Reads the segmented_outputs folder of a Visium HD run (Space Ranger cell segmentation) with VisiumIO::TENxVisiumHD(), as in the OSTA book chapter "Visium HD (segmented)". Gene symbols become unique row names, a logcounts assay is added if missing, and the platform is recorded. For the full workflow (QC, annotation, niches) see system.file("scripts", "visiumhd_osta_workflow.R", package = "spiSEE").

Usage

```
loadVisiumHD(segmented_outputs, images = "lowres")
```

Arguments

segmented_outputs	Path to the segmented_outputs directory.
images	Image resolution to load (passed to TENxVisiumHD).

Value

A SpatialExperiment.

See Also

loadCosmx(), loadXenium(), spiSEE().

Examples

```
d <- system.file("extdata", "segmented_outputs", package = "VisiumIO")
if (requireNamespace("VisiumIO", quietly = TRUE) &&
    requireNamespace("BiocIO", quietly = TRUE))
    spe <- loadVisiumHD(d)
```

loadXenium *Load a 10x Xenium output directory*

Description

Reads the bundle with SpatialExperimentIO::readXeniumSXE(), adds logcounts (log1p of counts) and, when cell_boundaries.parquet is present, cell polygons via SpaceTrooper. Missing polygons are not an error: the panels work on centroids.

Usage

```
loadXenium(xenium_dir)
```

Arguments

xenium_dir	Path to an unzipped Xenium output directory.

Value

A SpatialExperiment.

See Also

loadCosmx(), loadVisiumHD(), spiSEE().

Examples

```
d <- system.file("extdata", "Xenium_small", package = "SpatialExperimentIO")
if (requireNamespace("SpatialExperimentIO", quietly = TRUE))
    spe <- loadXenium(d)
```

---

plotSpatialCells *Static plot of cells on the tissue*

Description

Draws the cells of a SpatialExperiment as their polygons (or as points if there are none), coloured by annotations, QC metrics or genes, with the colours and legends of the app's spatial plots. Several values give one panel each, on a shared colour scale.

Usage

```
plotSpatialCells(
  spe,
  colour_by,
  polygons = TRUE,
  palette = NULL,
  ncol = NULL,
  borders = FALSE,
  point_size = 0.3
)
```

Arguments

spe	A SpatialExperiment.
colour_by	One or more colData columns or genes (row names); genes are coloured by logcounts (or counts). All numeric or all categorical.
polygons	Draw the cell polygons of a polygons column (as created by SpaceTrooper) if present; FALSE draws points.
palette	Named colours for categorical values (default: distinct colours, see the app).
ncol	Number of columns of panels.
borders	Draw cell borders (polygons only).
point_size	Point size when drawing points.

Value

A ggplot.

See Also

SpatialPlotPanel(), the same plot in the app.

Examples

```
dir <- system.file("extdata", "CosMx_DBKero_Tiny", package = "SpaceTrooper")
if (nzchar(dir)) {
    spe <- SpaceTrooper::readCosmxSPE(dir, sampleName = "DBKero")
    spe <- SpaceTrooper::spatialPerCellQC(spe)
    spe <- SpaceTrooper::readAndAddPolygonsToSPE(spe)
    plotSpatialCells(spe, "Area_um")
}
```

---

setPlatform *Set or detect the spatial platform of a dataset*

Description

spiSEE adapts its panels to the platform: CosMx data are selected by field of view (FOV), Xenium and Visium HD data by individual cells. The platform is stored in metadata(spe)$spiSEE_platform; when it is absent it is guessed from characteristic colData columns.

Usage

```
setPlatform(spe, platform = c("cosmx", "xenium", "visiumhd", "generic"))

getPlatform(spe)
```

Arguments

spe	A SpatialExperiment.
platform	One of "cosmx", "xenium", "visiumhd", "generic".

Value

setPlatform() returns spe with the platform recorded; getPlatform() returns the platform key (character).

See Also

spiSEE(), whose platform argument calls setPlatform().

Examples

```
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = matrix(1, 2, 3)),
    spatialCoords = matrix(runif(6), 3, 2,
        dimnames = list(NULL, c("x", "y"))))
getPlatform(spe)
spe <- setPlatform(spe, "xenium")
getPlatform(spe)
```

---

spatialCacheDir *Location of the spiSEE image cache*

Description

Plot images and map backgrounds are cached on disk by spiSEE() (argument cache). spatialCacheDir() gives the default folder for the images (a plots subfolder of the package's user cache, next to the map backgrounds); clearSpatialCache() deletes the cached images in a folder and, for the default folder, the map backgrounds too.

Usage

```
spatialCacheDir()

clearSpatialCache(dir = spatialCacheDir())
```

Arguments

dir	Cache folder. Only cache files (.rds) in it are deleted.

Value

spatialCacheDir(): the folder path. clearSpatialCache(): dir, invisibly.

See Also

spiSEE(), whose cache argument chooses the folder.

Examples

```
spatialCacheDir()
```

SpatialMapPanel-class *Overview map for building a region selection*

Description

Shows the whole tissue and lets the user pick FOVs (CosMx) or cells (Xenium, Visium HD) by clicking or dragging a rectangle. Selection happens in the browser; pressing *Validate* applies it to every spiSEE panel in the app (all classes extending SpatialPanelBase).

Usage

```
## S4 method for signature 'SpatialMapPanel'
.fullName(x)

## S4 method for signature 'SpatialMapPanel'
.panelColor(x)

## S4 method for signature 'SpatialMapPanel'
.defineOutput(x)

## S4 method for signature 'SpatialMapPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SpatialMapPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SpatialMapPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SpatialMapPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SpatialMapPanel(...)
```

Arguments

x	A panel instance (the class this page documents).
se	The SpatialExperiment being explored.
select_info	A list of selection information, passed by iSEE.

`...` Slots passed to `new()`, e.g. `PanelWidth`, or `CommittedIDs`: an initial selection (comma-separated ids), as set by the `selection` argument of `spiSEE()`.

`all_memory`, `all_contents`
iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).

`output`, `pObjects`, `rObjects`
The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.

`input`, `session` The current Shiny `input` and `session` objects.

Value

A `SpatialMapPanel` object.

See Also

`spiSEE()`; SpatialPanelBase for the panels that follow the selection.

Examples

```
SpatialMapPanel()
```

SpatialNichePanel-class

Spatial niches and domains panel

Description

iSEE panel showing cellular niches (from the cell-type composition of each cell's spatial neighbours) and spatial domains (clustering on neighbourhood-augmented expression), on any platform. It follows the selection of the `SpatialMapPanel()`. Results precomputed offline with `computeNiches()` / `computeSpatialDomains()` are shown instantly when the panel's parameters match them; *Compute* re-runs on the selection.

Usage

```
## S4 method for signature 'SpatialNichePanel'
.refineParameters(x, se)

## S4 method for signature 'SpatialNichePanel'
.fullName(x)

## S4 method for signature 'SpatialNichePanel'
.panelColor(x)

## S4 method for signature 'SpatialNichePanel'
```

```
.defineOutput(x)

## S4 method for signature 'SpatialNichePanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SpatialNichePanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SpatialNichePanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SpatialNichePanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SpatialNichePanel(
  PlotType = "niche_map",
  LabelCol = NA_character_,
  K = NA,
  NNiches = NA,
  Lambda = NA,
  UsePolygons = NA,
  ...
)
```

Arguments

`x`	A panel instance (the class this page documents).
`se`	The `SpatialExperiment` being explored.
`select_info`	A list of selection information, passed by iSEE.
`...`	Other slots: `Selection`, and `ComputeTrigger` (see `SpatialStatsPanel()`).
`all_memory`, `all_contents`	iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).
`output`, `pObjects`, `rObjects`	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.
`input`, `session`	The current Shiny `input` and `session` objects.
`PlotType`	One of the values of the "Show" menu, e.g. `"niche_map"`.
`LabelCol`	`colData` column with cell-type labels (auto-detected if `NA`).
`K`, `NNiches`, `Lambda`	Neighbours per cell, number of niches, spatial weight for domains. `NA` (the default) takes the values used for the niches and domains stored in the object, so they are shown without recomputing, or 20, 6 and 0.2 if none are stored.
`UsePolygons`	Draw real cell-shape polygons instead of a centroid scatter on the "on tissue" plot types (needs a `polygons` column, as created by SpaceTrooper). `NA` (the

default) follows the dataset: on at load time if a `polygons` column exists, off otherwise.

Value

A `SpatialNichePanel` object.

See Also

`computeNiches()` and `computeSpatialDomains()`, which run the same analyses outside the app.

Examples

```
SpatialNichePanel(K = 30, NNiches = 8)
```

SpatialPanelBase-class

Base class for spiSEE panels

Description

Virtual class extending iSEE::Panel. It adds a `Selection` slot that the `SpatialMapPanel()` fills when the user presses *Validate*.

Usage

```
## S4 method for signature 'SpatialPanelBase'
.createObservers(x, se, input, session, pObjects, rObjects)
```

Arguments

`x`	A panel extending `SpatialPanelBase`.
`se`	The `SpatialExperiment` being explored.
`input, session, pObjects, rObjects`	Shiny and iSEE state, passed by iSEE.

Value

`.createObservers()` is called for its side effects (it gives a panel added while the app runs the selection of the map) and returns `NULL`.

Slots

`Selection` Comma-separated region identifiers (FOV ids or cell ids, depending on the platform). Empty string = whole dataset.

See Also

`SpatialMapPanel()`, which sets the selection.

Examples

```
# every spiSEE panel except the map follows the map selection
is(SpatialPlotPanel(Selection = "1,2"), "SpatialPanelBase")
SpatialPlotPanel(Selection = "1,2")[["Selection"]]
```

`SpatialPlotPanel-class` *Spatial plot of cells coloured by annotation, QC metric or gene*

Description

iSEE panel showing the cells of the region selected on the `SpatialMapPanel()`, as centroids or as their segmentation polygons, coloured by a cell annotation, a QC metric or the expression of a gene (`logcounts`). It redraws as soon as the selection or a parameter changes.

Usage

```
## S4 method for signature 'SpatialPlotPanel'
.refineParameters(x, se)

## S4 method for signature 'SpatialPlotPanel'
.fullName(x)

## S4 method for signature 'SpatialPlotPanel'
.panelColor(x)

## S4 method for signature 'SpatialPlotPanel'
.defineOutput(x)

## S4 method for signature 'SpatialPlotPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SpatialPlotPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SpatialPlotPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SpatialPlotPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SpatialPlotPanel(ColorBy = "", UsePolygons = NA, PointSize = 0.5, ...)
```

Arguments

`x`	A panel instance (the class this page documents).
`se`	The `SpatialExperiment` being explored.

`select_info` A list of selection information, passed by iSEE.

`...` Other slots, e.g. `Selection`.

`all_memory`, `all_contents`
iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).

`output`, `pObjects`, `rObjects`
The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.

`input`, `session` The current Shiny `input` and `session` objects.

`ColorBy` A `colData` column or a gene name. Empty string = first cell-type column, else first QC metric.

`UsePolygons` Draw segmentation polygons instead of centroids (needs a `polygons` column, as created by SpaceTrooper). `NA` (the default) follows the dataset: on if a `polygons` column exists and the dataset has at most 20,000 cells (more polygons make the browser slow), off otherwise. It can still be ticked in the app on a small selection.

`PointSize` Centroid size.

Value

A `SpatialPlotPanel` object.

See Also

`plotSpatialCells()` for the same plot outside the app.

Examples

```
SpatialPlotPanel(ColorBy = "total")
```

SpatialQCPanel-class *QC metric histogram for the selected region*

Description

iSEE panel with the histogram of one QC metric (a numeric `colData` column) over the cells of the region selected on the `SpatialMapPanel()`. When SpaceTrooper's medcouple outlier detection has been run (`SpaceTrooper::computeSpatialOutlier()`, which adds a column `<Metric>_outlier_mc`), the outlier fences are drawn and the number of flagged cells is given.

Usage

```
## S4 method for signature 'SpatialQCPanel'
.fullName(x)

## S4 method for signature 'SpatialQCPanel'
.panelColor(x)

## S4 method for signature 'SpatialQCPanel'
.defineOutput(x)

## S4 method for signature 'SpatialQCPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SpatialQCPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SpatialQCPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SpatialQCPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SpatialQCPanel(Metric = "", UseFences = TRUE, ...)
```

Arguments

`x`	A panel instance (the class this page documents).
`se`	The `SpatialExperiment` being explored.
`select_info`	A list of selection information, passed by iSEE.
`...`	Other slots, e.g. `Selection`.
`all_memory`, `all_contents`	iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).
`output`, `pObjects`, `rObjects`	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.
`input`, `session`	The current Shiny `input` and `session` objects.
`Metric`	A numeric `colData` column. Empty = first known QC metric of the platform.
`UseFences`	Show medcouple outlier fences when a column `<Metric>_outlier_mc` exists (SpaceTrooper).

Value

A `SpatialQCPanel` object.

See Also

FOVStatsPanel() for metrics per FOV, SpatialPlotPanel() to see a metric on the tissue.

Examples

```
SpatialQCPanel(Metric = "total")
```

SpatialStatsPanel-class

Spatial statistics on the selected region

Description

Neighbour graph, global and local Moran's I, local cluster significance, join-count statistics and L-functions, computed on the region selected on the SpatialMapPanel() when the user presses *Compute*.

Usage

```
## S4 method for signature 'SpatialStatsPanel'
.fullName(x)

## S4 method for signature 'SpatialStatsPanel'
.panelColor(x)

## S4 method for signature 'SpatialStatsPanel'
.defineOutput(x)

## S4 method for signature 'SpatialStatsPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SpatialStatsPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SpatialStatsPanel'
.refineParameters(x, se)

## S4 method for signature 'SpatialStatsPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SpatialStatsPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SpatialStatsPanel(
  PlotType = "graph_plot",
  Genes = character(0),
  LabelCol = NA_character_,
```

```
  K = 6L,
  Alpha = 0.05,
  ...
)
```

Arguments

x	A panel instance (the class this page documents).
se	The SpatialExperiment being explored.
select_info	A list of selection information, passed by iSEE.
...	Other slots: Selection, and ComputeTrigger, which is set when *Compute* is pressed. Give any non-empty value (e.g. ComputeTrigger = "launch") to compute the panel when the app opens.
all_memory, all_contents	iSEE's full panel-state list and the current selection contents of every panel, passed to .generateOutput() for export/snapshot but unused by this package's own panels (selection is handled client-side, see SpatialMapPanel).
output, pObjects, rObjects	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.
input, session	The current Shiny input and session objects.
PlotType	One of "graph_plot", "localmoran_plot", "moran_plot", "locclust_plot", "joincount_plot", "joincount_heatmap_plot", "lest_plot", "lcross_plot".
Genes	Genes for the Moran-type statistics.
LabelCol	Cell-type column (NA = auto-detect).
K	Number of spatial neighbours per cell.
Alpha	Significance level for the local cluster plot: a cell is called significant when its adjusted p-value (local Moran's I) is below Alpha. Shown on the plot.

Details

The neighbour graph is a k-nearest-neighbour graph on the cell centroids of the region. Moran's I and the local clusters come from Voyager, the join counts from spdep. The L-function (lest_plot) is computed for the most frequent cell type and the L-cross function (lcross_plot) for the two most frequent ones (types with at least 10 cells), with spatialFDA, up to a radius of a quarter of the shorter side of the region.

Changing a parameter does not recompute anything: results are computed on *Compute* and cached by what they compute (statistic, parameters, region and dataset), so the same analysis opens at once the next time.

Value

A SpatialStatsPanel object.

See Also

SVGPanel() to rank genes by Moran's I, SpatialNichePanel() for niches and domains.

Examples

```
SpatialStatsPanel(PlotType = "joincount_plot", K = 10)
```

---

spiSEE *Launch the spiSEE app*

Description

Opens an iSEE app whose panels adapt to the platform of spe (CosMx, Xenium, Visium HD, or any SpatialExperiment). A selection made on the overview map applies to every other panel.

Usage

```
spiSEE(
  spe,
  mode = c("analysis", "qc"),
  platform = NULL,
  selection = "",
  sample = NULL,
  initial = NULL,
  cache = getOption("spiSEE.cache", TRUE),
  cache_size = getOption("spiSEE.cache_size", 1024^3),
  title = "spiSEE",
  ...
)
```

Arguments

spe	A SpatialExperiment (e.g. from loadCosmx(), loadXenium() or loadVisiumHD()).
mode	"analysis" or "qc".
platform	Optional platform key to override detection (see setPlatform()).
selection	Initial selection: FOV or cell ids, as a character vector or one comma-separated string. Empty = whole dataset. Unknown ids are dropped with a warning.
sample	With several samples (spe$sample_id), the one to show: the app shows one sample at a time (one map, one coordinate system). Open one app per sample to compare them.
initial	Optional list of panels; replaces the default layout.
cache	Where plot images are cached: TRUE (default) on disk in spatialCacheDir(), kept across restarts; a folder path (used only for the images: do not share it with other files); or FALSE for memory only (lost when R stops).
cache_size	Maximum size of the disk cache in bytes (default 1 GB); the least recently used images are removed first.
title	Title of the app (browser tab and header).
...	Passed to iSEE::iSEE().

Details

- `mode = "qc"`: map, spatial plot, two QC histograms (+ FOV zoom and per-FOV summary for CosMx). Meant for unfiltered data.
- `mode = "analysis"`: map, two spatial plots (annotation and gene), spatial statistics, spatially variable genes, niches and domains.

Value

A Shiny app object (prints/launches in an interactive session).

Optional metadata

Some entries of `metadata(spe)` are used when present:

- `gene_ranks`: gene names, in the order of the gene menus and of the genes scanned by `SVGPanel()` (default: by mean expression).
- `fov_stats`: per-FOV values for `FOVStatsPanel()` (see there).
- `technology`: as set by SpaceTrooper's readers; gives the unit of the scale bars of the spatial plots (none without it).
- `fov_positions`, `fov_dim`: FOV layout from `SpaceTrooper::readCosmxSPE()`, drawn on the map (CosMx); without them the tiles follow the extent of the cells of each FOV.
- Niches and domains stored by `computeNiches()` and `computeSpatialDomains()`: shown by `SpatialNichePanel()` without recomputing.

See Also

`SpatialMapPanel()` and the other panels; `spatialCacheDir()`.

Examples

```
spe <- SpatialExperiment::SpatialExperiment(
    assays = list(counts = matrix(rpois(200, 3), 10, 20,
        dimnames = list(paste0("g", 1:10), paste0("c", 1:20)))),
    spatialCoords = matrix(runif(40), 20, 2,
        dimnames = list(NULL, c("x", "y"))))
app <- spiSEE(spe, mode = "qc", cache = FALSE)
if (interactive()) shiny::runApp(app)
```

SVGPanel-class *Ranking of spatially variable genes (Moran's I)*

Description

iSEE panel with a table of genes ranked by Moran's I on the region selected on the `SpatialMapPanel()`, computed on *Compute* (Voyager, on a k-nearest-neighbour graph of the cell centroids). Clicking a row shows the gene in a linked `SpatialPlotPanel()`.

Usage

```
## S4 method for signature 'SVGPanel'
.fullName(x)

## S4 method for signature 'SVGPanel'
.panelColor(x)

## S4 method for signature 'SVGPanel'
.defineOutput(x)

## S4 method for signature 'SVGPanel'
.defineInterface(x, se, select_info)

## S4 method for signature 'SVGPanel'
.generateOutput(x, se, ..., all_memory, all_contents)

## S4 method for signature 'SVGPanel'
.refineParameters(x, se)

## S4 method for signature 'SVGPanel'
.renderOutput(x, se, output, pObjects, rObjects)

## S4 method for signature 'SVGPanel'
.createObservers(x, se, input, session, pObjects, rObjects)

SVGPanel(TopN = 200L, K = 6L, LinkedPanel = "", ...)
```

Arguments

`x`	A panel instance (the class this page documents).
`se`	The `SpatialExperiment` being explored.
`select_info`	A list of selection information, passed by iSEE.
`...`	Other slots: `Selection`, and `ComputeTrigger` (see `SpatialStatsPanel()`).
`all_memory`, `all_contents`	iSEE's full panel-state list and the current selection contents of every panel, passed to `.generateOutput()` for export/snapshot but unused by this package's own panels (selection is handled client-side, see `SpatialMapPanel`).
`output`, `pObjects`, `rObjects`	The app's Shiny output list and the persistent/reactive state environments iSEE passes to every panel method.
`input`, `session`	The current Shiny `input` and `session` objects.
`TopN`	Number of most expressed genes to scan (or the first `TopN` of `metadata(se)$gene_ranks`).
`K`	Number of spatial neighbours per cell.
`LinkedPanel`	Name of a `SpatialPlotPanel()` (e.g. `"SpatialPlotPanel2"`) that shows the gene clicked in the table.

Value

A SVGPanel object.

See Also

SpatialStatsPanel() for Moran's I of chosen genes.

Examples

```
SVGPanel(TopN = 100, LinkedPanel = "SpatialPlotPanel1")
```

Index

.createObservers,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.createObservers,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.createObservers,SVGPanel-method
(SVGPanel-class), 24
.createObservers,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.createObservers,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.createObservers,SpatialPanelBase-method
(SpatialPanelBase-class), 17
.createObservers,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.createObservers,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.createObservers,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.defineInterface,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.defineInterface,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.defineInterface,SVGPanel-method
(SVGPanel-class), 24
.defineInterface,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.defineInterface,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.defineInterface,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.defineInterface,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.defineInterface,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.defineOutput,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.defineOutput,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.defineOutput,SVGPanel-method
(SVGPanel-class), 24
.defineOutput,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.defineOutput,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.defineOutput,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.defineOutput,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.defineOutput,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.fullName,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.fullName,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.fullName,SVGPanel-method
(SVGPanel-class), 24
.fullName,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.fullName,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.fullName,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.fullName,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.fullName,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.generateOutput,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.generateOutput,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.generateOutput,SVGPanel-method
(SVGPanel-class), 24
.generateOutput,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.generateOutput,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.generateOutput,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18

.generateOutput,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.generateOutput,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.panelColor,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.panelColor,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.panelColor,SVGPanel-method
(SVGPanel-class), 24
.panelColor,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.panelColor,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.panelColor,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.panelColor,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.panelColor,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.refineParameters,SVGPanel-method
(SVGPanel-class), 24
.refineParameters,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.refineParameters,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.refineParameters,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21
.renderOutput,FOVStatsPanel-method
(FOVStatsPanel-class), 6
.renderOutput,FOVZoomPanel-method
(FOVZoomPanel-class), 8
.renderOutput,SVGPanel-method
(SVGPanel-class), 24
.renderOutput,SpatialMapPanel-method
(SpatialMapPanel-class), 14
.renderOutput,SpatialNichePanel-method
(SpatialNichePanel-class), 15
.renderOutput,SpatialPlotPanel-method
(SpatialPlotPanel-class), 18
.renderOutput,SpatialQCPanel-method
(SpatialQCPanel-class), 19
.renderOutput,SpatialStatsPanel-method
(SpatialStatsPanel-class), 21

buildSpatialGraph, 2
buildSpatialGraph(), *4*, *5*

clearSpatialCache *(*spatialCacheDir*)*, 13
computeNiches, 3
computeNiches(), *3*, *15*, *17*, *24*
computeSpatialDomains, 5
computeSpatialDomains(), *3*, *15*, *17*, *24*

FOVStatsPanel *(*FOVStatsPanel-class*)*, 6
FOVStatsPanel(), *9*, *21*, *24*
FOVStatsPanel-class, 6
FOVZoomPanel *(*FOVZoomPanel-class*)*, 8
FOVZoomPanel(), *7*
FOVZoomPanel-class, 8

getPlatform *(*setPlatform*)*, 12

iSEE::iSEE(), *23*
iSEE::Panel, *17*

loadCosmx, 9
loadCosmx(), *10*, *11*, *23*
loadVisiumHD, 10
loadVisiumHD(), *9*, *11*, *23*
loadXenium, 10
loadXenium(), *9*, *10*, *23*

plotSpatialCells, 11
plotSpatialCells(), *19*

setPlatform, 12
setPlatform(), *23*
spatialCacheDir, 13
spatialCacheDir(), *23*, *24*
SpatialMapPanel
(SpatialMapPanel-class), 14
SpatialMapPanel(), *6*, *8*, *15*, *17–19*, *21*, *24*
SpatialMapPanel-class, 14
SpatialNichePanel
(SpatialNichePanel-class), 15
SpatialNichePanel(), *4*, *5*, *22*, *24*
SpatialNichePanel-class, 15
SpatialPanelBase, *14*, *15*
SpatialPanelBase-class, 17
SpatialPlotPanel
(SpatialPlotPanel-class), 18
SpatialPlotPanel(), *12*, *21*, *24*, *25*
SpatialPlotPanel-class, 18
SpatialQCPanel *(*SpatialQCPanel-class*)*,
19
SpatialQCPanel(), *7*, *9*
SpatialQCPanel-class, 19

SpatialStatsPanel
(SpatialStatsPanel-class), 21
SpatialStatsPanel(), *16*, *25, 26*
SpatialStatsPanel-class, 21
spiSEE, 23
spiSEE(), *9–11*, *13–15*
SVGPanel (SVGPanel-class), 24
SVGPanel(), *22*, *24*
SVGPanel-class, 24