Package 'Voyager'

April 11, 2023

Type Package

Title From geospatial to spatial omics

Version 1.0.10

Description SpatialFeatureExperiment (SFE) is a new S4 class for working with spatial single-cell genomics data. The voyager package implements basic exploratory spatial data analysis (ESDA) methods for SFE. This first version supports univariate global spatial ESDA methods such as Moran's I, permutation testing for Moran's I, and correlograms. The Voyager package also implements plotting functions to plot SFE data and ESDA results. Multivariate ESDA and univariate local metrics will be added in later versions.

Imports BiocParallel, bluster, ggnewscale, ggplot2 (>= 3.4.0), Matrix, methods, patchwork, rlang, S4Vectors, scales, scico, sf, SingleCellExperiment, SpatialExperiment, SpatialFeatureExperiment, spdep, stats, SummarizedExperiment

Suggests BiocSingular, BiocStyle, cowplot, dbscan, ExperimentHub, hexbin, knitr, rmarkdown, scater, scattermore, scran, SFEData, sparseMatrixStats, testthat (>= 3.0.0), vdiffr

License Artistic-2.0

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

Depends R (>= 4.2.0)

biocViews GeneExpression, Spatial, Transcriptomics, Visualization

VignetteBuilder knitr

URL `https://github.com/pachterlab/voyager`

BugReports `https://github.com/pachterlab/voyager/issues`

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

git_branch RELEASE_3_16

git_last_commit 317c847

git_last_commit_date 2023-03-07

Date/Publication 2023-04-10

Author Lambda Moses [aut, cre] (<`https://orcid.org/0000-0002-7092-9427`>),
Kayla Jackson [aut] (<`https://orcid.org/0000-0001-6483-0108`>),
Lior Pachter [aut, rev] (<`https://orcid.org/0000-0002-9164-6231`>)

Maintainer Lambda Moses <`dlu2@caltech.edu`>

R topics documented:

`calculateUnivariate` *Univariate spatial stiatistics*

Description

These functions compute univariate spatial statistics, both global and local, on matrices, data frames, and SFE objects. For SFE objects, the statistics can be computed for numeric columns of `colData`, `colGeometries`, and `annotGeometries`, and the results are stored within the SFE object. `calculateMoransI` and `runMoransI` are convenience wrappers for `calculateUnivariate` and `runUnivariate` respectively.

Usage

```
## S4 method for signature 'ANY'
calculateUnivariate(
  x,
  listw,
  type = c("moran", "geary", "moran.mc", "geary.mc", "moran.test", "geary.test",
   "globalG.test", "sp.correlogram", "moran.plot", "localmoran", "localmoran_perm",
   "localC", "localC_perm", "localG", "localG_perm", "LOSH", "LOSH.mc", "LOSH.cs",
    "gwss"),
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  returnDF = TRUE,
  p.adjust.method = "BH",
  ...
)

## S4 method for signature 'SpatialFeatureExperiment'
calculateUnivariate(
  x,
  type,
  features = NULL,
  colGraphName = 1L,
  sample_id = NULL,
  exprs_values = "logcounts",
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  returnDF = TRUE,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

## S4 method for signature 'ANY'
calculateMoransI(x, ..., BPPARAM = SerialParam(), zero.policy = NULL)

## S4 method for signature 'SpatialFeatureExperiment'
calculateMoransI(
  x,
  features = NULL,
  colGraphName = 1L,
  sample_id = NULL,
  exprs_values = "logcounts",
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  returnDF = TRUE,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
```

```
)

colDataUnivariate(
  x,
  type,
  features,
  colGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

colDataMoransI(
  x,
  features,
  colGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

colGeometryUnivariate(
  x,
  type,
  features,
  colGeometryName = 1L,
  colGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

colGeometryMoransI(
  x,
  features,
  colGeometryName = 1L,
  colGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

annotGeometryUnivariate(
  x,
  type,
  features,
  annotGeometryName = 1L,
  annotGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

annotGeometryMoransI(
  x,
  features,
  annotGeometryName = 1L,
  annotGraphName = 1L,
  sample_id = NULL,
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

runUnivariate(
  x,
  type,
  features = NULL,
  colGraphName = 1L,
  sample_id = NULL,
  exprs_values = "logcounts",
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)

runMoransI(
  x,
  features = NULL,
  colGraphName = 1L,
  sample_id = NULL,
  exprs_values = "logcounts",
  BPPARAM = SerialParam(),
  zero.policy = NULL,
  include_self = FALSE,
  p.adjust.method = "BH",
  ...
)
```

Arguments

`x`	A numeric matrix whose rows are features/genes, or a `SpatialFeatureExperiment` (SFE) object with such a matrix in an assay.
`listw`	Weighted neighborhood graph as a spdep `listw` object.
`type`	An integer specifying the index or string specifying the name of the *Geometry to query or replace. If missing, then the first item in the *Geometries will be returned or replaced.
`BPPARAM`	A `BiocParallelParam` object specifying whether and how computing the metric for numerous genes shall be parallelized.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`returnDF`	Logical, when the results are not added to a SFE object, whether the results should be formatted as a `DataFrame`.
`p.adjust.method`	Method to correct for multiple testing, passed to `p.adjustSP`. Methods allowed are in `p.adjust.methods`.
`...`	Other arguments passed to S4 method (for convenience wrappers like `calculateMoransI`) or method used to compute metrics as specified by the argument `type` (as in more general functions like `calculateUnivariate`). See documentation in the `spdep` package for the latter.
`features`	Genes (`calculate*` SFE method and `run*`) or numeric columns of `colData(x)` (`colData*`) or any `colGeometry` (`colGeometry*`) or `annotGeometry` (`annotGeometry*`) for which the univariate metric is to be computed. Default to `NULL`. When `NULL`, then the metric is computed for all genes with the values in the assay specified in the argument `exprs_values`. This can be parallelized with the argument `BPPARAM`. For genes, if the column "symbol" is present in `rowData` and the row names of the SFE object are Ensembl IDs, then the gene symbol can be used and converted to IDs behind the scene. However, if one symbol matches multiple IDs, a warning will be given and the first match will be used. Internally, the results are always stored by the Ensembl ID rather than symbol.
`colGraphName`	Name of the listw graph in the SFE object that corresponds to entities represented by columns of the gene count matrix. Use `colGraphNames` to look up names of the available graphs for cells/spots. Note that for multiple `sample_ids`, it is assumed that all of them have a graph of this same name.

`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`exprs_values`	Integer scalar or string indicating which assay of x contains the expression values.
`include_self`	Logical, whether the spatial neighborhood graph should include edges from each location to itself. This is for Getis-Ord Gi* as in `localG` and `localG_perm`, not to be used for any other method.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`annotGeometryName`	Name of a `annotGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `annotGeometryNames` to look up names of the `sf` data frames associated with annotations.
`annotGraphName`	Name of the listw graph in the SFE object that corresponds to the `annotGeometry` of interest. Use `annotGraphNames` to look up names of available annotation graphs.

Details

Most univariate methods in the package `spdep` are supported here. These methods are global, meaning returning one result for all spatial locations in the dataset: `moran`, `geary`, `moran.mc`, `geary.mc`, `moran.test`, `geary.test`, `globalG.test`, `sp.correlogram`.

The following methods are local, meaning each location has its own results: `moran.plot`, `localmoran`, `localmoran_perm`, `localC`, `localC_perm`, `localG`, `localG_perm`, `LOSH`, `LOSH.mc`, `LOSH.cs`. The `GWmodel::gwss` method will be supported soon, but is not supported yet.

Global results for genes are stored in `rowData`. For `colGeometry` and `annotGeometry`, the results are added to an attribute of the data frame called `featureData`, which is a DataFrame analogous to `rowData` for the gene count matrix. New column names in `featureData` would follow the same rules as in `rowData`. For `colData`, the results can be accessed with the `colFeatureData` function.

Local results are stored in the field `localResults` field of the SFE object, which can be accessed with `localResults` or `localResult`. If the results have p-values, then -log10 p and Benjamin-Hochberg corrected -log10 p are added. Note that in the multiple testing correction, `p.adjustSP` is used.

Value

In `calculateUnivariate`, if `returnDF = TRUE`, then a `DataFrame`, otherwise a list each element of which is the results for each feature. For `run*`, a `SpatialFeatureExperiment` object with the results added. See Details for where the results are stored.

Examples

```
library(SpatialFeatureExperiment)
library(SingleCellExperiment)
library(SFEData)
```

```
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
features_use <- rownames(sfe)[1:5]

# Moran's I
moran_results <- calculateMoransI(sfe,
    features = features_use,
    colGraphName = "visium",
    exprs_values = "counts"
)

# This does not advocate for computing Moran's I on raw counts.
# Just an example for function usage.

sfe <- runMoransI(sfe,
    features = features_use, colGraphName = "visium",
    exprs_values = "counts"
)
# Look at the results
head(rowData(sfe))

# Local Moran's I
sfe <- runUnivariate(sfe,
    type = "localmoran", features = features_use,
    colGraphName = "visium", exprs_values = "counts"
)
head(localResult(sfe, "localmoran", features_use[1]))

# For colData
sfe <- colDataUnivariate(sfe,
    type = "localmoran", features = "nCounts",
    colGraphName = "visium"
)
head(localResult(sfe, "localmoran", "nCounts"))

# For annotGeometries
annotGraph(sfe, "myofiber_tri2nb") <-
    findSpatialNeighbors(sfe,
        type = "myofiber_simplified", MARGIN = 3L,
        method = "tri2nb", dist_type = "idw",
        zero.policy = TRUE
    )
sfe <- annotGeometryUnivariate(sfe,
    type = "localG", features = "area",
    annotGraphName = "myofiber_tri2nb",
    annotGeometryName = "myofiber_simplified",
    zero.policy = TRUE
)
head(localResult(sfe, "localG", "area",
    annotGeometryName = "myofiber_simplified"
))
```

clusterCorrelograms *Find clusters of correlogram patterns*

Description

Cluster the correlograms to find patterns in length scales of spatial autocorrelation. All the correlograms clustered must be computed with the same method and have the same number of lags.

Usage

```
clusterCorrelograms(
  sfe,
  features,
  BLUSPARAM,
  sample_id = NULL,
  method = "I",
  colGeometryName = NULL,
  annotGeometryName = NULL,
  show_symbol = TRUE
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object with correlograms computed for features of interest.
`features`	Features whose correlograms to cluster.
`BLUSPARAM`	A BlusterParam object specifying the algorithm to use.
`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`method`	"corr" for correlation, "I" for Moran's I, "C" for Geary's C
`colGeometryName`	Name of colGeometry from which to look for features.
`annotGeometryName`	Name of annotGeometry from which to look for features.
`show_symbol`	Logical, whether to show gene symbol instead when Ensembl ID is supplied.

Value

A `DataFrame` with 3 columns: `feature` for the features, `cluster` a factor for cluster membership of the features within each sample, and `sample_id` for the sample.

Examples

```
library(SpatialFeatureExperiment)
library(SFEData)
library(bluster)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
inds <- c(1, 3, 4, 5)
sfe <- runUnivariate(sfe,
    type = "sp.correlogram",
    features = rownames(sfe)[inds],
    exprs_values = "counts", order = 5
)
clust <- clusterCorrelograms(sfe,
    features = rownames(sfe)[inds],
    BLUSPARAM = KmeansParam(2)
)
```

`clusterMoranPlot` *Find clusters on the Moran plot*

Description

The Moran plot plots the value at each location on the x axis, and the average of the neighbors of each locations on the y axis. Sometimes clusters can be seen on the Moran plot, indicating different types of neighborhoods.

Usage

```
clusterMoranPlot(
  sfe,
  features,
  BLUSPARAM,
  sample_id = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  show_symbol = TRUE
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object with Moran plot computed for the feature of interest. If the Moran plot for that feature has not been computed for that feature in this sample_id, it will be calculated and stored in `rowData`. See `calculateUnivariate`.
`features`	Features whose Moran plot are to be cluster. Features whose Moran plots have not been computed will be skipped, with a warning.
`BLUSPARAM`	A BlusterParam object specifying the algorithm to use.

`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`colGeometryName`	Name of colGeometry from which to look for features.
`annotGeometryName`	Name of annotGeometry from which to look for features.
`show_symbol`	Logical, whether to show gene symbol instead when Ensembl ID is supplied.

Value

A `DataFrame` each column of which is a factor for cluster membership of each feature. The column names are the features.

Examples

```
library(SpatialFeatureExperiment)
library(SingleCellExperiment)
library(SFEData)
library(bluster)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
# Compute moran plot
sfe <- runUnivariate(sfe,
    type = "moran.plot", features = rownames(sfe)[1],
    exprs_values = "counts"
)
clusts <- clusterMoranPlot(sfe, rownames(sfe)[1],
    BLUSPARAM = KmeansParam(2)
)
```

`colFeatureData` *Get metadata of colData and rowData*

Description

Results of spatial analyses on columns in `colData` and `rowData` are stored in `int_metadata(sfe)`, or internal metadata. This function allows the users to access these results.

Usage

```
colFeatureData(sfe)

rowFeatureData(sfe)
```

Arguments

`sfe`	An SFE object.

Value

A DataFrame.

Examples

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

---

ditto_colors *Colorblind friendly palette from dittoSeq*

Description

Just to get the palette without having to install all those dependencies of dittoSeq.

Usage

```
ditto_colors
```

Format

A character vector of hex colors of the palette. There are 40 colors.

Source

The dittoSeq package.

ElbowPlot *Plot the elbow plot or scree plot for PCA*

Description

Apparently, there is no apparent way to plot the PC elbow plot other than extracting the variance explained attribute of the dimred slot, because even the OSCA book makes the elbow plot this way, which I find kind of cumbersome compared to Seurat. So I'm writing this function to make the elbow plot with SCE less cumbersome.

Usage

```
ElbowPlot(sce, ndims = 20, reduction = "PCA")
```

Arguments

`sce`	A `SingleCellExperiment` object, or anything that inherits from `SingleCellExperiment`.
`ndims`	Number of PCs to plot.
`reduction`	Name of the dimension reduction to use. It must have an attribute called "percentVar". Defaults to "PCA".

Value

A ggplot object. The y axis is percentage of variance explained.

Examples

```
library(SFEData)
library(scater)
sfe <- McKellarMuscleData("small")
sfe <- runPCA(sfe, ncomponents = 10, exprs_values = "counts")
ElbowPlot(sfe, ndims = 10)
```

`getDivergeRange` *Get beginning and end of palette to center a divergent palette*

Description

This function is no longer used internally as it's unnecessary for `scico` divergent palettes. But it can be useful when using divergent palettes outside `scico` where one must specify beginning and end but not midpoint, to override the default palette.

Usage

```
getDivergeRange(values, diverge_center = 0)
```

Arguments

`values`	Numeric vector to be colored.
`diverge_center`	Value to center on, defaults to 0.

Value

A numeric vector of length 2, the first element is for beginning, and the second for end. The values are between 0 and 1.

Examples

```
v <- rnorm(10)
getDivergeRange(v, diverge_center = 0)
```

moranPlot *Use ggplot to plot the moran.plot results*

Description

This function uses ggplot2 to plot the Moran plot. The plot would be more aesthetically pleasing than the base R version implemented in spdep. In addition, contours are plotted to show point density on the plot, and the points can be colored by a variable, such as clusters. The contours may also be filled and only influential points plotted. When filled, the viridis E option is used.

Usage

```
moranPlot(
  sfe,
  feature,
  graphName = 1L,
  sample_id = NULL,
  contour_color = "cyan",
  color_by = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  plot_singletons = TRUE,
  binned = FALSE,
  filled = FALSE,
  divergent = FALSE,
  diverge_center = NULL,
  show_symbol = TRUE,
  bins = 100,
  binwidth = NULL,
  hex = FALSE,
  plot_influential = TRUE,
  ...
)
```

Arguments

sfe	A SpatialFeatureExperiment object.
feature	Name of one variable to show on the plot. It will be converted to sentence case on the x axis and lower case in the y axis appended after "Spatially lagged". One feature at a time since the colors in color_by may be specific to this feature (e.g. from clusterMoranPlot).
graphName	Name of the colGraph or annotGraph, the spatial neighborhood graph used to compute the Moran plot. This is to determine which points are singletons to plot differently on this plot.
sample_id	One sample_id for the sample whose graph to plot.

`contour_color`	Color of the point density contours, which can be changed so the contours stand out from the points.
`color_by`	Variable to color the points by. It can be the name of a column in colData, a gene, or the name of a column in the colGeometry specified in colGeometryName. Or it can be a vector of the same length as the number of cells/spots in the sample_id of interest.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`annotGeometryName`	Name of a `annotGeometry` of the SFE object, to annotate the gene expression plot.
`plot_singletons`	Logical, whether to plot items that don't have spatial neighbors.
`binned`	Logical, whether to plot 2D histograms. This argument has precedence to `filled`.
`filled`	Logical, whether to plot filled contours for the non-influential points and only plot influential points as points.
`divergent`	Logical, whether a divergent palette should be used.
`diverge_center`	If `divergent = TRUE`, the center from which the palette should diverge. If `NULL`, then not centering.
`show_symbol`	Logical, whether to show human readable gene symbol on the plot instead of Ensembl IDs when the row names are Ensembl IDs. There must be a column in `rowData(sfe)` called "symbol" for this to work.
`bins`	Numeric vector giving number of bins in both vertical and horizontal directions. Set to 100 by default.
`binwidth`	Numeric vector giving bin width in both vertical and horizontal directions. Overrides `bins` if both set.
`hex`	Logical, whether to use hexagon rather than rectangular bins. Requires the `hexbin` package.
`plot_influential`	Logical, whether to plot influential points with different palette if `binned = TRUE`.
`...`	Other arguments to pass to `geom_density2d`.

Value

A ggplot object.

Examples

```
library(SpatialFeatureExperiment)
library(SingleCellExperiment)
library(SFEData)
library(bluster)
library(scater)
```

```
sfe <- McKellarMuscleData("full")
sfe <- sfe[, colData(sfe)$in_tissue]
sfe <- logNormCounts(sfe)
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
sfe <- runUnivariate(sfe, type = "moran.plot", features = "Myh1")
clust <- clusterMoranPlot(sfe, "Myh1", BLUSPARAM = KmeansParam(2))
moranPlot(sfe, "Myh1", graphName = "visium", color_by = clust[, 1])
```

plotCellBin2D *Plot cell density as 2D histogram*

Description

This function plots cell density in histological space as 2D histograms, especially helpful for larger smFISH-based datasets.

Usage

```
plotCellBin2D(sfe, bins = 200, binwidth = NULL, hex = FALSE)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`bins`	Numeric vector giving number of bins in both vertical and horizontal directions. Set to 100 by default.
`binwidth`	Numeric vector giving bin width in both vertical and horizontal directions. Overrides `bins` if both set.
`hex`	Logical, whether to use hexagon rather than rectangular bins. Requires the `hexbin` package.

Value

A ggplot object.

Examples

```
library(SFEData)
sfe <- HeNSCLCData()
plotCellBin2D(sfe)
```

`plotColDataBin2D` *Plot colData and rowData with 2D histograms*

Description

To avoid overplotting in large datasets. The 2D histogram is more informative of point density on the plot than the scatter plot where there are so many points plotted that they effectively form a solid block.

Usage

```
plotColDataBin2D(
  sfe,
  x,
  y,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  hex = FALSE,
  name_true = NULL,
  name_false = NULL
)

plotRowDataBin2D(
  sfe,
  x,
  y,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  hex = FALSE,
  name_true = NULL,
  name_false = NULL
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`x`	Name of the column in `colData` or `rowData` to plot on the x axis of the plot.
`y`	Name of the column in `colData` or `rowData` to plot on the y axis of the plot.
`subset`	Name of a logical column in `colData` or `rowData`, indicating cells or genes to plot with a different palette. Since the 2D histogram is effectively an opaque heatmap, don't use this argument unless the two groups are largely non-overlapping in the variables being plotted.
`bins`	Numeric vector giving number of bins in both vertical and horizontal directions. Set to 100 by default.

`binwidth`	Numeric vector giving bin width in both vertical and horizontal directions. Overrides `bins` if both set.
`hex`	Logical, whether to use hexagon rather than rectangular bins. Requires the `hexbin` package.
`name_true`	Character, name to show on the legend for cells or genes indicated `TRUE` in the `subset` argument.
`name_false`	Character, name to show on the legend for cells or genes indicated `FALSE` in the `subset` argument.

Value

A ggplot object

Examples

```
library(SFEData)
sfe <- McKellarMuscleData()
sfe <- sfe[, sfe$in_tissue]
plotColDataBin2D(sfe, "nCounts", "nGenes")
```

`plotColDataFreqpoly` *Plot frequency polygons for colData and rowData columns*

Description

This function is recommended instead of `plotColDataHistogram` when coloring by multiple categories and log transforming the y axis, which causes problems in stacked histograms.

Usage

```
plotColDataFreqpoly(
  sfe,
  feature,
  color_by = NULL,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  linewidth = 1.2,
  scales = "free",
  ncol = 1,
  position = "identity"
)

plotRowDataFreqpoly(
  sfe,
  feature,
  color_by = NULL,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  linewidth = 1.2,
  scales = "free",
  ncol = 1,
  position = "identity"
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`feature`	Names of columns in `colData` or `rowData` to plot. When multiple features are specified, they will be plotted in separate facets.
`color_by`	Name of a categorical column in `colData` or `rowData` to color the polygons.
`subset`	Name of a logical column to only plot a subset of the data.
`bins`	Number of bins. Overridden by `binwidth`. Defaults to 30.
`binwidth`	The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in `bins`, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
`linewidth`	Line width of the polygons, defaults to a thicker 1.2.
`scales`	Should scales be fixed (`"fixed"`, the default), free (`"free"`), or free in one dimension (`"free_x"`, `"free_y"`)?
`ncol`	Number of columns in the facetting.
`position`	Position adjustment, either as a string naming the adjustment (e.g. `"jitter"` to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

See Also

plotColDataHistogram

Examples

```
library(SFEData)
sfe <- McKellarMuscleData()
plotColDataFreqpoly(sfe, c("nCounts", "nGenes"), color_by = "in_tissue",
                    bins = 50)
plotColDataFreqpoly(sfe, "nCounts", subset = "in_tissue")
sfe2 <- sfe[, sfe$in_tissue]
plotColDataFreqpoly(sfe2, c("nCounts", "nGenes"), bins = 50)
```

`plotColDataHistogram` *Plot histograms for colData and rowData columns*

Description

Plot histograms for colData and rowData columns

Usage

```
plotColDataHistogram(
  sfe,
  feature,
  fill_by = NULL,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  scales = "free",
  ncol = 1,
  position = "identity"
)

plotRowDataHistogram(
  sfe,
  feature,
  fill_by = NULL,
  subset = NULL,
  bins = 100,
  binwidth = NULL,
  scales = "free",
  ncol = 1,
  position = "identity"
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`feature`	Names of columns in `colData` or `rowData` to plot. When multiple features are specified, they will be plotted in separate facets.
`fill_by`	Name of a categorical column in `colData` or `rowData` to fill the histogram.
`subset`	Name of a logical column to only plot a subset of the data.
`bins`	Numeric vector giving number of bins in both vertical and horizontal directions. Set to 100 by default.
`binwidth`	Numeric vector giving bin width in both vertical and horizontal directions. Overrides `bins` if both set.
`scales`	Should scales be fixed (`"fixed"`, the default), free (`"free"`), or free in one dimension (`"free_x"`, `"free_y"`)?

`ncol`	Number of columns in the facetting.
`position`	Position adjustment, either as a string naming the adjustment (e.g. `"jitter"` to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

Value

A ggplot object

See Also

plotColDataFreqpoly

Examples

```
library(SFEData)
sfe <- McKellarMuscleData()
plotColDataHistogram(sfe, c("nCounts", "nGenes"), fill_by = "in_tissue",
                     bins = 50, position = "stack")
plotColDataHistogram(sfe, "nCounts", subset = "in_tissue")
sfe2 <- sfe[, sfe$in_tissue]
plotColDataHistogram(sfe2, c("nCounts", "nGenes"), bins = 50)
```

`plotColGraph` *Plot spatial graphs*

Description

A ggplot version of `spdep::plot.nb`, reducing boilerplate for SFE objects.

Usage

```
plotColGraph(
  sfe,
  colGraphName = 1L,
  colGeometryName = NULL,
  sample_id = NULL,
  weights = FALSE,
  segment_size = 0.5,
  geometry_size = 0.5,
  ncol = NULL
)

plotAnnotGraph(
  sfe,
  annotGraphName = 1L,
  annotGeometryName = 1L,
  sample_id = NULL,
  weights = FALSE,
  segment_size = 0.5,
  geometry_size = 0.5,
  ncol = NULL
)
```

Arguments

sfe	A SpatialFeatureExperiment object.
colGraphName	Name of graph associated with columns of the gene count matrix to be plotted.
colGeometryName	Name of a colGeometry sf data frame whose numeric columns of interest are to be used to compute the metric. Use colGeometryNames to look up names of the sf data frames associated with cells/spots.
sample_id	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
weights	Whether to plot weights. If TRUE, then transparency (alpha) of the segments will represent edge weights.
segment_size	Thickness of the segments that represent graph edges.
geometry_size	Point size (for POINT geometries) or line thickness (for LINESTRING and POLYGON) to plot the geometry in the background.
ncol	Number of columns if plotting multiple features. Defaults to NULL, which means using the same logic as facet_wrap, which is used by patchwork's wrap_plots by default.
annotGraphName	Name of the annotation graph to plot.
annotGeometryName	Name of the annotGeometry, which is associated with the graph specified with annotGraphName, for spatial coordinates of the graph nodes and for context.

Value

A ggplot2 object.

Examples

```
library(SpatialFeatureExperiment)
library(SFEData)
library(sf)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
plotColGraph(sfe, colGraphName = "visium", colGeometryName = "spotPoly")
# Make the myofiber segmentations a valid POLYGON geometry
ag <- annotGeometry(sfe, "myofiber_simplified")
ag <- st_buffer(ag, 0)
ag <- ag[!st_is_empty(ag), ]
annotGeometry(sfe, "myofiber_simplified") <- ag
annotGraph(sfe, "myofibers") <-
    findSpatialNeighbors(sfe,
        type = "myofiber_simplified", MARGIN = 3,
        method = "tri2nb", dist_type = "idw"
    )
plotAnnotGraph(sfe,
    annotGraphName = "myofibers",
    annotGeometryName = "myofiber_simplified",
    weights = TRUE
)
```

---

plotCorrelogram *Plot correlogram*

Description

Use ggplot2 to plot correlograms computed by runUnivariate, pulling results from rowData. Correlograms of multiple genes with error bars can be plotted, and they can be colored by any numeric or categorical column in rowData or a vector with the same length as nrow of the SFE object. The coloring is useful when the correlograms are clustered to show types of length scales or patterns of decay of spatial autocorrelation. For method = "I", the error bars are twice the standard deviation of the estimated Moran's I value.

Usage

```
plotCorrelogram(
  sfe,
  features,
  sample_id = NULL,
  method = "I",
  color_by = NULL,
  facet_by = c("sample_id", "features"),
  ncol = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  plot_signif = TRUE,
  p_adj_method = "BH",
  divergent = FALSE,
  diverge_center = NULL,
  show_symbol = TRUE
)
```

Arguments

sfe	A SpatialFeatureExperiment object.
features	Features to plot, must be in rownames of the gene count matrix, colnames of colData or a colGeometry.
sample_id	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.

method	"corr" for correlation, "I" for Moran's I, "C" for Geary's C
color_by	Name of a column in rowData(sfe) or in the featureData of colData (see colFeatureData), colGeometry, or annotGeometry by which to color the correlogram of each feature. Alternatively, a vector of the same length as features.
facet_by	Whether to facet by sample_id (default) or features. If facetting by sample_id, then different features will be plotted in the same facet for comparison. If facetting by features, then different samples will be compared for each feature. Ignored if only one sample is specified.
ncol	Number of columns if facetting.
colGeometryName	Name of a colGeometry sf data frame whose numeric columns of interest are to be used to compute the metric. Use colGeometryNames to look up names of the sf data frames associated with cells/spots.
annotGeometryName	Name of a annotGeometry of the SFE object, to annotate the gene expression plot.
plot_signif	Logical, whether to plot significance symbols: p < 0.001: ***, p < 0.01: **, p < 0.05 *, p < 0.1: ., otherwise no symbol. The p-values are two sided, based on the assumption that the estimated Moran's I is normally distributed with mean from a randomized version of the data. The mean and variance come from moran.test for Moran's I and geary.test for Geary's C. Take the results with a grain of salt if the data is not normally distributed.
p_adj_method	Multiple testing correction method as in p.adjust, to correct for multiple testing (number of lags times number of features) in the Moran's I estimates if plot_signif = TRUE.
divergent	Logical, whether a divergent palette should be used.
diverge_center	If divergent = TRUE, the center from which the palette should diverge. If NULL, then not centering.
show_symbol	Logical, whether to show human readable gene symbol on the plot instead of Ensembl IDs when the row names are Ensembl IDs. There must be a column in rowData(sfe) called "symbol" for this to work.

Value

A ggplot object.

Examples

```
library(SpatialFeatureExperiment)
library(SFEData)
library(bluster)
library(scater)
sfe <- McKellarMuscleData("small")
sfe <- logNormCounts(sfe)
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
inds <- c(1, 3, 4, 5)
features <- rownames(sfe)[inds]
```

```
sfe <- runUnivariate(sfe,
    type = "sp.correlogram", features = features,
    exprs_values = "counts", order = 5
)
clust <- clusterCorrelograms(sfe,
    features = features,
    BLUSPARAM = KmeansParam(2)
)
# Color by features
plotCorrelogram(sfe, features)
# Color by something else
plotCorrelogram(sfe, features, color_by = clust$cluster)
# Facet by features
plotCorrelogram(sfe, features, facet_by = "features")
```

---

plotDimLoadings *Plot top PC loadings of genes*

Description

Just like Seurat's VizDimLoadings function. I haven't found an equivalent for SCE but find it useful. But I'm not trying to reproduce that Seurat function exactly. For instance, I don't like it when Seurat imposes a ggplot theme, and I don't like the cowplot theme. Maybe I should rewrite it in base R but for now I'm using Tidyverse.

Usage

```
plotDimLoadings(
  sce,
  dims = 1:4,
  nfeatures = 10,
  show_symbol = TRUE,
  symbol_col = "symbol",
  reduction = "PCA",
  balanced = TRUE,
  ncol = 2
)
```

Arguments

`sce`	A `SingleCellExperiment` object, or anything that inherits from `SingleCellExperiment`.
`dims`	Numeric vector specifying which PCs to plot.
`nfeatures`	Number of genes to plot.
`show_symbol`	Logical; if the row names of the matrix are Ensembl accessions, indicate whether to show more human readable gene symbols in the plot instead. Ignored if the column specified in `symbol_col` is absent from rowData.

`symbol_col`	If the row names of the gene expression matrix are Ensembl accessions to avoid ambiguity in analysis. If not found in `rowData`, then rownames of the gene count matrix will be used.
`reduction`	Name of the dimension reduction to use. It must have an attribute called "percentVar". Defaults to "PCA".
`balanced`	Return an equal number of genes with + and - scores. If FALSE, returns the top genes ranked by the scores absolute values.
`ncol`	Number of columns in the facetted plot.

Value

A ggplot object. Loadings for different PCs are plotted in different facets so one ggplot object is returned.

Examples

```
library(SFEData)
library(scater)
sfe <- McKellarMuscleData("small")
sfe <- runPCA(sfe, ncomponents = 10, exprs_values = "counts")
plotDimLoadings(sfe, dims = 1:2)
```

`plotLocalResult` *Plot local results*

Description

Plot results of local spatial analyses in space, such as local Getis-Ord Gi* values.

Usage

```
plotLocalResult(
  sfe,
  type,
  features,
  attribute = NULL,
  sample_id = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  ncol = NULL,
  ncol_sample = NULL,
  annot_aes = list(),
  annot_fixed = list(),
  aes_use = c("fill", "color", "shape", "linetype"),
  divergent = FALSE,
  diverge_center = NULL,
  annot_divergent = FALSE,
  annot_diverge_center = NULL,
  size = 0,
  shape = 16,
  linetype = 1,
  alpha = 1,
  color = NA,
  fill = "gray80",
  show_symbol = TRUE,
  scattermore = FALSE,
  pointsize = 0,
  ...
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`type`	Which local spatial results. Use `localResultNames` to see which types of results have already been calculated.
`features`	Character vector of vectors. To see which features have the results of a given type, see `localResultFeatures`.
`attribute`	Which field in the local results of the type and features. If the result of each feature is a vector, the this argument is ignored. But if the result is a data frame or a matrix, then this is the column name of the result, such as "Ii" for local Moran's I. For each local spatial analysis method, there's a default attribute. See Details. Use `localResultAttrs`.
`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`annotGeometryName`	Name of a `annotGeometry` of the SFE object, to annotate the gene expression plot.
`ncol`	Number of columns if plotting multiple features. Defaults to `NULL`, which means using the same logic as `facet_wrap`, which is used by `patchwork`'s `wrap_plots` by default.
`ncol_sample`	If plotting multiple samples as facets, how many columns of such facets. This is distinct from `ncols`, which is for multiple features. When plotting multiple features for multiple samples, then the result is a multi-panel plot each panel of which is a plot for each feature facetted by samples.
`annot_aes`	A named list of plotting parameters for the annotation sf data frame. The names are which geom (as in ggplot2, such as color and fill), and the values are column names in the annotation sf data frame. Tidyeval is NOT supported.
`annot_fixed`	Similar to `annot_aes`, but for fixed aesthetic settings, such as `color = "gray"`. The defaults are the same as the relevant defaults for this function.

`aes_use`	Aesthetic to use for discrete variables. For continuous variables, it's always "fill" for polygons and point shapes 21-25. For discrete variables, it can be fill, color, shape, or linetype, whenever applicable. The specified value will be changed to the applicable equivalent. For example, if the geometry is point but "linetype" is specified, then "shaped" will be used instead.
`divergent`	Logical, whether a divergent palette should be used.
`diverge_center`	If `divergent = TRUE`, the center from which the palette should diverge. If `NULL`, then not centering.
`annot_divergent`	Just as `divergent`, but for the annotGeometry in case it's different.
`annot_diverge_center`	Just as `diverge_center`, but for the annotGeometry in case it's different.
`size`	Fixed size of points or width of lines, including outlines of polygons. For polygons, this defaults to 0, meaning no outlines. For points and lines, this defaults to 0.5. Ignored if `size_by` is specified.
`shape`	Fixed shape of points, ignored if `shape_by` is specified and applicable.
`linetype`	Fixed line type, ignored if `linetype_by` is specified and applicable.
`alpha`	Transparency.
`color`	Fixed color for `colGeometry` if `color_by` is not specified or not applicable, or for `annotGeometry` if `annot_color_by` is not specified or not applicable.
`fill`	Similar to `color`, but for fill.
`show_symbol`	Logical, whether to show human readable gene symbol on the plot instead of Ensembl IDs when the row names are Ensembl IDs. There must be a column in `rowData(sfe)` called "symbol" for this to work.
`scattermore`	Logical, whether to use the `scattermore` package to greatly speed up plotting numerous points. Only used for POINT `colGeometries`. If the geometry is not POINT, then the centroids are used. Recommended for plotting hundreds of thousands or more cells where the cell polygons can't be seen when plotted due to the large number of cells and small plot size such as when plotting multiple panels for multiple features.
`pointsize`	Radius of rasterized point in `scattermore`. Default to 0 for single pixels (fastest).
`...`	Other arguments passed to `wrap_plots`.

Details

Many local spatial analyses return a data frame or matrix as the results, whose columns can be the statistic of interest at each location, its variance, expected value from permutation, p-value, and etc. The `attribute` argument specifies which column to use when there are multiple columns. Below are the defaults for each local method supported by this package what what they mean:

`localmoran` **and** `localmoran_perm` `Ii`, local Moran's I statistic at each location.

`localC_perm` `localC`, the local Geary C statistic at each location.

`localG` **and** `localG_perm` `localG`, the local Getis-Ord Gi or Gi* statistic. If `include_self = TRUE` when `calculateUnivariate` or `runUnivariate` was called, then it would be Gi*. Otherwise it's Gi.

`LOSH` **and** `LOSH.mc` `Hi`, local spatial heteroscedasticity

`moran.plot` `wx`, the average of the value of each neighbor of each location. Moran plot is best plotted as a scatter plot of `wx` vs `x`. See `moranPlot`.

Other local methods not listed above return vectors as results. For instance, `localC` returns a vector by default, which is the local Geary's C statistic.

Value

A `ggplot2` object if plotting one feature. A `patchwork` object if plotting multiple features.

Note

While this function shares internals with `plotSpatialFeature`, there are some important differences. In `plotSpatialFeature`, the `annotGeometry` is indeed only used for annotation and the protagonist is the `colGeometry`, since it's easy to directly use `ggplot2` to plot the data in `annotGeometry` `sf` data frames while overlaying `annotGeometry` and `colGeometry` involves more complicated code. In contrast, in this function, local results for `annotGeometry` can be plotted separately without anything related to `colGeometry`. Note that when `annotGeometry` local results are plotted without `colGeometry`, the `annot_*` arguments are ignored. Use the other arguments for aesthetics as if it's for `colGeometry`.

Examples

```
library(SpatialFeatureExperiment)
library(SFEData)
library(scater)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
feature_use <- rownames(sfe)[1]
sfe <- logNormCounts(sfe)
sfe <- runUnivariate(sfe, "localmoran", feature_use)
# Which types of results are available?
localResultNames(sfe)
# Which features for localmoran?
localResultFeatures(sfe, "localmoran")
# Which columns does the localmoran results have?
localResultAttrs(sfe, "localmoran", feature_use)
plotLocalResult(sfe, "localmoran", feature_use, "Ii",
    colGeometryName = "spotPoly"
)

# For annotGeometry
# Make sure it's type POLYGON
annotGeometry(sfe, "myofiber_simplified") <-
    sf::st_buffer(annotGeometry(sfe, "myofiber_simplified"), 0)
annotGraph(sfe, "poly2nb_myo") <-
    findSpatialNeighbors(sfe,
        type = "myofiber_simplified", MARGIN = 3,
        method = "poly2nb", zero.policy = TRUE
    )
```

```
sfe <- annotGeometryUnivariate(sfe, "localmoran",
    features = "area",
    annotGraphName = "poly2nb_myo",
    annotGeometryName = "myofiber_simplified",
    zero.policy = TRUE
)
plotLocalResult(sfe, "localmoran", "area", "Ii",
    annotGeometryName = "myofiber_simplified",
    size = 0.3, color = "cyan"
)
plotLocalResult(sfe, "localmoran", "area", "Z.Ii",
    annotGeometryName = "myofiber_simplified"
)
# don't use annot_* arguments when annotGeometry is plotted without colGeometry
```

plotMoranMC *Plot Moran/Geary monte carlo results*

Description

Plot the simulations as a density plot or histogram compared to the observed Moran's I or Geary's C, with ggplot2 so it looks nicer. Unlike the plotting function in spdep, this function can also plot the same feature in different samples as facets or plot different features or samples together for comparison.

Usage

```
plotMoranMC(
  sfe,
  features,
  sample_id = NULL,
  facet_by = c("sample_id", "features"),
  ncol = NULL,
  colGeometryName = NULL,
  annotGeometryName = NULL,
  ptype = c("density", "histogram", "freqpoly"),
  show_symbol = TRUE,
  ...
)
```

Arguments

sfe	A SpatialFeatureExperiment object.
features	Features to plot, must be in rownames of the gene count matrix, colnames of colData or a colGeometry.
sample_id	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.

`facet_by`	Whether to facet by sample_id (default) or features. If facetting by sample_id, then different features will be plotted in the same facet for comparison. If facetting by features, then different samples will be compared for each feature. Ignored if only one sample is specified.
`ncol`	Number of columns if facetting.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`annotGeometryName`	Name of a `annotGeometry` of the SFE object, to annotate the gene expression plot.
`ptype`	Plot type, one of "density", "histogram", or "freqpoly".
`show_symbol`	Logical, whether to show human readable gene symbol on the plot instead of Ensembl IDs when the row names are Ensembl IDs. There must be a column in `rowData(sfe)` called "symbol" for this to work.
`...`	Other arguments passed to `geom_density`, `geom_histogram`, or `geom_freqpoly`, depending on `ptype`.

Value

A `ggplot2` object.

Examples

```
library(SpatialFeatureExperiment)
library(SFEData)
sfe <- McKellarMuscleData("small")
colGraph(sfe, "visium") <- findVisiumGraph(sfe)
sfe <- colDataUnivariate(sfe, type = "moran.mc", "nCounts", nsim = 100)
plotMoranMC(sfe, "nCounts")
```

`plotSpatialFeature` *Plot gene expression in space*

Description

Unlike `Seurat` and `ggspavis`, plotting functions in this package uses `geom_sf` whenever applicable.

Usage

```
plotSpatialFeature(
  sfe,
  features,
  colGeometryName = 1L,
  sample_id = NULL,
  ncol = NULL,
  ncol_sample = NULL,
  annotGeometryName = NULL,
  annot_aes = list(),
  annot_fixed = list(),
  exprs_values = "logcounts",
  aes_use = c("fill", "color", "shape", "linetype"),
  divergent = FALSE,
  diverge_center = NA,
  annot_divergent = FALSE,
  annot_diverge_center = NA,
  size = 0,
  shape = 16,
  linetype = 1,
  alpha = 1,
  color = NA,
  fill = "gray80",
  show_symbol = TRUE,
  scattermore = FALSE,
  pointsize = 0,
  ...
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`features`	Features to plot, must be in rownames of the gene count matrix, colnames of colData or a colGeometry.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`ncol`	Number of columns if plotting multiple features. Defaults to `NULL`, which means using the same logic as `facet_wrap`, which is used by `patchwork`'s `wrap_plots` by default.
`ncol_sample`	If plotting multiple samples as facets, how many columns of such facets. This is distinct from `ncols`, which is for multiple features. When plotting multiple features for multiple samples, then the result is a multi-panel plot each panel of which is a plot for each feature facetted by samples.
`annotGeometryName`	Name of a `annotGeometry` of the SFE object, to annotate the gene expression plot.
`annot_aes`	A named list of plotting parameters for the annotation sf data frame. The names are which geom (as in ggplot2, such as color and fill), and the values are column names in the annotation sf data frame. Tidyeval is NOT supported.

`annot_fixed`	Similar to `annot_aes`, but for fixed aesthetic settings, such as `color = "gray"`. The defaults are the same as the relevant defaults for this function.
`exprs_values`	Integer scalar or string indicating which assay of x contains the expression values.
`aes_use`	Aesthetic to use for discrete variables. For continuous variables, it's always "fill" for polygons and point shapes 21-25. For discrete variables, it can be fill, color, shape, or linetype, whenever applicable. The specified value will be changed to the applicable equivalent. For example, if the geometry is point but "linetype" is specified, then "shaped" will be used instead.
`divergent`	Logical, whether a divergent palette should be used.
`diverge_center`	If `divergent = TRUE`, the center from which the palette should diverge. If `NULL`, then not centering.
`annot_divergent`	Just as `divergent`, but for the annotGeometry in case it's different.
`annot_diverge_center`	Just as `diverge_center`, but for the annotGeometry in case it's different.
`size`	Fixed size of points or width of lines, including outlines of polygons. For polygons, this defaults to 0, meaning no outlines. For points and lines, this defaults to 0.5. Ignored if `size_by` is specified.
`shape`	Fixed shape of points, ignored if `shape_by` is specified and applicable.
`linetype`	Fixed line type, ignored if `linetype_by` is specified and applicable.
`alpha`	Transparency.
`color`	Fixed color for `colGeometry` if `color_by` is not specified or not applicable, or for `annotGeometry` if `annot_color_by` is not specified or not applicable.
`fill`	Similar to `color`, but for fill.
`show_symbol`	Logical, whether to show human readable gene symbol on the plot instead of Ensembl IDs when the row names are Ensembl IDs. There must be a column in `rowData(sfe)` called "symbol" for this to work.
`scattermore`	Logical, whether to use the `scattermore` package to greatly speed up plotting numerous points. Only used for POINT `colGeometries`. If the geometry is not POINT, then the centroids are used. Recommended for plotting hundreds of thousands or more cells where the cell polygons can't be seen when plotted due to the large number of cells and small plot size such as when plotting multiple panels for multiple features.
`pointsize`	Radius of rasterized point in `scattermore`. Default to 0 for single pixels (fastest).
`...`	Other arguments passed to `wrap_plots`.

Details

In the documentation of this function, a "feature" can be a gene (or whatever entity that corresponds to rows of the gene count matrix), a column in `colData`, or a column in the `colGeometry` `sf` data frame specified in the `colGeometryName` argument.

For continuous variables, the Blues palette from colorbrewer is used if `divergent = FALSE`, and the roma palette from the `scico` package if `divergent = TRUE`. For discrete variables, the `dittoSeq` palette is used. The defaults are colorblind friendly. For annotation, the PuRd colorbrewer palette is used for continuous variables and the other end of the `dittoSeq` palette is used for discrete variables.

Value

A `ggplot2` object if plotting one feature. A `patchwork` object if plotting multiple features.

Examples

```
library(SFEData)
library(sf)
sfe <- McKellarMuscleData("small")
# features can be genes or colData or colGeometry columns
plotSpatialFeature(sfe, c("nCounts", rownames(sfe)[1]),
    exprs_values = "counts",
    colGeometryName = "spotPoly",
    annotGeometryName = "tissueBoundary"
)
# Change fixed aesthetics
plotSpatialFeature(sfe, "nCounts",
    colGeometryName = "spotPoly",
    annotGeometryName = "tissueBoundary",
    annot_fixed = list(color = "blue", size = 0.3, fill = NA),
    alpha = 0.7
)
# Make the myofiber segmentations a valid POLYGON geometry
ag <- annotGeometry(sfe, "myofiber_simplified")
ag <- st_buffer(ag, 0)
ag <- ag[!st_is_empty(ag), ]
annotGeometry(sfe, "myofiber_simplified") <- ag
# Also plot an annotGeometry variable
plotSpatialFeature(sfe, "nCounts",
    colGeometryName = "spotPoly",
    annotGeometryName = "myofiber_simplified",
    annot_aes = list(fill = "area")
)
```

`spatialReducedDim` *Plot dimension reduction components in space*

Description

Such as plotting the value of projection of gene expression of each cell to a principal component in space. At present, this function does not work for the 3D array of geographically weighted PCA (GWPCA), but a future version will deal with GWPCA results.

Usage

```
spatialReducedDim(
  sfe,
  dimred,
  ncomponents,
  colGeometryName = 1L,
  sample_id = NULL,
  ncol = NULL,
  ncol_sample = NULL,
  annotGeometryName = NULL,
  annot_aes = list(),
  annot_fixed = list(),
  exprs_values = "logcounts",
  aes_use = c("fill", "color", "shape", "linetype"),
  divergent = FALSE,
  diverge_center = NULL,
  annot_divergent = FALSE,
  annot_diverge_center = NULL,
  size = 0,
  shape = 16,
  linetype = 1,
  alpha = 1,
  color = NA,
  fill = "gray80",
  scattermore = FALSE,
  pointsize = 0,
  ...
)
```

Arguments

`sfe`	A `SpatialFeatureExperiment` object.
`dimred`	A string or integer scalar indicating the reduced dimension result in `reducedDims(sfe)` to plot.
`ncomponents`	A numeric scalar indicating the number of dimensions to plot, starting from the first dimension. Alternatively, a numeric vector specifying the dimensions to be plotted.
`colGeometryName`	Name of a `colGeometry` `sf` data frame whose numeric columns of interest are to be used to compute the metric. Use `colGeometryNames` to look up names of the `sf` data frames associated with cells/spots.
`sample_id`	Sample(s) in the SFE object whose cells/spots to use. Can be "all" to compute metric for all samples; the metric is computed separately for each sample.
`ncol`	Number of columns if plotting multiple features. Defaults to `NULL`, which means using the same logic as `facet_wrap`, which is used by `patchwork`'s `wrap_plots` by default.

`ncol_sample`	If plotting multiple samples as facets, how many columns of such facets. This is distinct from `ncols`, which is for multiple features. When plotting multiple features for multiple samples, then the result is a multi-panel plot each panel of which is a plot for each feature facetted by samples.
`annotGeometryName`	Name of a `annotGeometry` of the SFE object, to annotate the gene expression plot.
`annot_aes`	A named list of plotting parameters for the annotation sf data frame. The names are which geom (as in ggplot2, such as color and fill), and the values are column names in the annotation sf data frame. Tidyeval is NOT supported.
`annot_fixed`	Similar to `annot_aes`, but for fixed aesthetic settings, such as `color = "gray"`. The defaults are the same as the relevant defaults for this function.
`exprs_values`	Integer scalar or string indicating which assay of x contains the expression values.
`aes_use`	Aesthetic to use for discrete variables. For continuous variables, it's always "fill" for polygons and point shapes 21-25. For discrete variables, it can be fill, color, shape, or linetype, whenever applicable. The specified value will be changed to the applicable equivalent. For example, if the geometry is point but "linetype" is specified, then "shaped" will be used instead.
`divergent`	Logical, whether a divergent palette should be used.
`diverge_center`	If `divergent = TRUE`, the center from which the palette should diverge. If `NULL`, then not centering.
`annot_divergent`	Just as `divergent`, but for the annotGeometry in case it's different.
`annot_diverge_center`	Just as `diverge_center`, but for the annotGeometry in case it's different.
`size`	Fixed size of points or width of lines, including outlines of polygons. For polygons, this defaults to 0, meaning no outlines. For points and lines, this defaults to 0.5. Ignored if `size_by` is specified.
`shape`	Fixed shape of points, ignored if `shape_by` is specified and applicable.
`linetype`	Fixed line type, ignored if `linetype_by` is specified and applicable.
`alpha`	Transparency.
`color`	Fixed color for `colGeometry` if `color_by` is not specified or not applicable, or for `annotGeometry` if `annot_color_by` is not specified or not applicable.
`fill`	Similar to `color`, but for fill.
`scattermore`	Logical, whether to use the `scattermore` package to greatly speed up plotting numerous points. Only used for POINT `colGeometries`. If the geometry is not POINT, then the centroids are used. Recommended for plotting hundreds of thousands or more cells where the cell polygons can't be seen when plotted due to the large number of cells and small plot size such as when plotting multiple panels for multiple features.
`pointsize`	Radius of rasterized point in `scattermore`. Default to 0 for single pixels (fastest).
`...`	Other arguments passed to `wrap_plots`.

Value

Same as in plotSpatialFeature. A ggplot2 object if plotting one component. A patchwork object if plotting multiple components.

See Also

scater::plotReducedDim

Examples

```
library(SFEData)
library(scater)
sfe <- McKellarMuscleData("small")
sfe <- logNormCounts(sfe)
sfe <- runPCA(sfe, ncomponents = 2)
spatialReducedDim(sfe, "PCA", 2, "spotPoly",
    annotGeometryName = "tissueBoundary",
    divergent = TRUE, diverge_center = 0
)
# Basically PC1 separates spots not on tissue from those on tissue.
```

Index