Package 'SEraster'

November 12, 2026

Type Package

Title Rasterization Preprocessing Framework for Scalable Spatial Omics Data Analysis

Version 1.4.0

URL https://github.com/JEFworks-Lab/SEraster

BugReports https://github.com/JEFworks-Lab/SEraster/issues

Description SEraster is a rasterization preprocessing framework that aggregates cellular information into spatial pixels to reduce resource requirements for spatial omics data analysis. SEraster reduces the number of spatial points in spatial omics datasets for downstream analysis through a process of rasterization where single cells' gene expression or cell-type labels are aggregated into equally sized pixels based on a user-defined resolution. SEraster is built on an R/Bioconductor S4 class called SpatialExperiment. SEraster can be incorporated with other packages to conduct downstream analyses for spatial omics datasets, such as detecting spatially variable genes.

biocViews Software, Spatial, GeneExpression, Transcriptomics, SingleCell, Preprocessing

License GPL-3

Encoding UTF-8

LazyData FALSE

Suggests CooccurrenceAffinity, nnSVG, testthat (>= 3.0.0), knitr, rmarkdown, BiocManager, remotes

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

Depends R (>= 4.5.0)

Imports BiocParallel, ggplot2, Matrix, methods, rearrr, sf, SpatialExperiment, SummarizedExperiment

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

git_branch RELEASE_3_23

git_last_commit 66e8e1f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Gohta Aihara [aut, cre] (ORCID:
<https://orcid.org/0000-0002-2492-9610>),
Mayling Chen [aut] (ORCID: <https://orcid.org/0009-0009-0961-6665>),
Lyla Atta [aut] (ORCID: <https://orcid.org/0000-0002-6113-0082>),
Jean Fan [aut, rev] (ORCID: <https://orcid.org/0000-0002-0212-5451>)

Maintainer Gohta Aihara <gohta.aihara@gmail.com>

Contents

merfish_mousePOA 2
permutateByRotation 3
plotRaster 4
rasterizeCellType 6
rasterizeGeneExpression 8
rasterizeMatrix 10

Index **13**

`merfish_mousePOA` *Preprocessed MERFISH dataset of the mouse preoptic area for a bregma -0.29 slice from a female naive animal (Animal ID = 1, Animal Sex = "Female", Behavior = "Naive", Bregma = "-0.29").*

Description

Preprocessed MERFISH dataset of the mouse preoptic area for a bregma -0.29 slice from a female naive animal (Animal ID = 1, Animal Sex = "Female", Behavior = "Naive", Bregma = "-0.29").

Usage

```
data("merfish_mousePOA")
```

Format

`SpatialExperiment` object where `assay` slot contains genes-by-cells matrix with preprocessed gene expression (total RNA counts per cell divided by cell volume and scaled by 1000) as `dgCMatrix`, `spatialCoords` slot contains x,y coordinates of cells, and `colData` slot contains bregma, cell type, and neuron type meta data.

Value

SpatialExperiment object for the preprocessed MERFISH dataset of the mouse preoptic area for a bregma -0.29 slice from a female naive animal (Animal ID = 1, Animal Sex = "Female", Behavior = "Naive", Bregma = "-0.29").

Source

https://www.science.org/doi/10.1126/science.aau5324

---

permutateByRotation *permutateByRotation*

Description

Function to permutate a given input SpatialExperiment object(s) by rotating the x,y coordinates around the midrange point.

This function assumes that the input is provided as a SpatialExperiment object or a list of SpatialExperiment objects.

When the input is a list of SpatialExperiment objects, all SpatialExperiment objects will be rotated around a common midrange point computed based on combined x,y coordinates.

Usage

```
permutateByRotation(input, n_perm = 1, verbose = FALSE)
```

Arguments

input	SpatialExperiment or list: Input data represented as a SpatialExperiment or list of SpatialExperiment. Each SpatialExperiment is assumed to have an assay slot containing feature (genes) x observation (cells) matrix as dgCmatrix or matrix and a spatialCoords slot containing spatial x,y coordinates of observations as matrix array. Further, x,y coordinates are assumed to be stored in column 1 and 2 of spatialCoords, and column names of spatialCoords are assumed to be "x" and "y", respectively.
n_perm	integer: Number of permutations. Default = 1. This number is used to compute the angles at which the input is rotated at.
verbose	logical: Whether to display verbose output or warning. Default is FALSE.

Value

If the input was given as SpatialExperiment, the output is returned as a list of n_perm SpatialExperiment objects. Each SpatialExperiment object has an updated spatialCoords slot containing the spatial x,y coordinates rotated at a corresponding angle. assay and colData slots are inherited. Further, names() of the output indicates the angles at which the input is rotated at. If the input was given as list of SpatialExperiment, the output is returned as a new list of length(input) * n_perm SpatialExperiment objects. Each SpatialExperiment object has an updated spatialCoords

slot containing the spatial x,y coordinates rotated at a corresponding angle. `assay` and `colData` slots are inherited. Further, `names()` of the output indicates the dataset names from `names(input)` and the angles at which the input is rotated at.

Examples

```
data("merfish_mousePOA")

# create a list of 3 permutated datasets rotated at 0 (original), 120, and 240 degrees
# this output can directly be fed into rasterizeGeneExpression or rasterizeCellType
# functions to rasterize all 3 permutations at once with the same pixel coordinates
spe_list <- permutateByRotation(merfish_mousePOA, n_perm = 3)

# create a list of 5 permutated datasets rotated at 0 (original), 72, 144, 216, 288 degrees
spe_list <- permutateByRotation(merfish_mousePOA, n_perm = 5)
```

`plotRaster` *plotRaster*

Description

Function based on `ggplot2::geom_tile` to visualize a rasterized spatial omics dataset represented as a `SpatialExperiment` object.

Usage

```
plotRaster(
  input,
  assay_name = NULL,
  feature_name = "sum",
  factor_levels = NULL,
  showLegend = TRUE,
  plotTitle = NULL,
  showAxis = FALSE,
  ...
)
```

Arguments

`input`	`SpatialExperiment`: Input data represented as a `SpatialExperiment`. The given `SpatialExperiment` is assumed to have an `assay` slot containing a features-by-observations matrix as `dgCmatrix` or `matrix` and a `colData` slot containing `sfc_POLYGON` geometry of pixels. The features-by-observations matrix is assumed to have either genes or cell types as features and pixels as observations.
`assay_name`	`character`: Name of the assay slot of the input that you want to visualize. If no argument is given, the first assay of the input would be visualized. This argument is useful when you have multiple assays stored in the input, and you want to visualize a specific assay. Default is NULL.

feature_name	character: Name of the feature in the input that you want to visualize. This argument is useful when you want to specify a feature you want to visualize. You can also use "sum" to visualize sum of all feature values per observation or "mean" to visualize mean of all feature values per observation. Default is "sum".
factor_levels	character or numeric or factor: An optional vector to convert and plot the input data as factor. This argument is useful if you want to plot categorical/ordinal variables, such as binarized occurrence of a specific cell type. factor_levels is fed into levels argument of the factor function in base R. Default is NULL.
showLegend	logical: Boolean to show the legend. Default is TRUE.
plotTitle	character: An optional argument to add a title to the resulting plot. Default is NULL.
showAxis	logical: Boolean to show axis title, texts, and ticks. Default is FALSE.
...	Additional parameters to pass to ggplot2::scale_fill_viridis_c if no argument is provided to factor_levels or ggplot2::scale_fill_viridis_d if a vector is provided to factor_levels. If you wish to use other color maps, we recommend overriding the resulting ggplot object.

Value

The output is returned as a ggplot object, where the input is visualized as ggplot2::geom_sf. Each pixel is plotted based on sfc_POLYGON geometry stored in the colData slot. Coloring of pixel represent the corresponding values of summarized (sum or mean) or specific feature (e.g. rasterized gene expression) per observation (pixel).

Examples

```
data("merfish_mousePOA")

# rasterize gene expression
out <- rasterizeGeneExpression(merfish_mousePOA, assay_name = "volnorm", fun = "mean")

# plot total rasterized gene expression per pixel (there is only one assay_name
# in out and default for feature_name argument is "sum"; therefore, these arguments
# are not specified)
plotRaster(out, name = "total rasterized gexp")

# plot rasterized expression of a specific gene/feature per pixel
plotRaster(out, feature_name = "Esr1", name = "Esr1")

# rasterize cell-type labels with user-defined resolution and hexagonal pixels
out <- rasterizeCellType(merfish_mousePOA, col_name = "celltype", resolution = 50,
square = FALSE, fun = "sum")

# plot total cell counts per pixel (there is only one assay_name in out and default
# for feature_name argument is "sum"; therefore, these arguments are not specified)
# here, let's use additional parameters for ggplot2::scale_fill_viridis_c so
# that it would have a different color scheme from gene expression plots
plotRaster(out, name = "total cell counts", option = "inferno")
```

```
# plot specific cell type's cell counts per pixel
plotRaster(out, feature_name = "Inhibitory", name = "Inhibitory neuron counts", option = "inferno")
```

`rasterizeCellType` *rasterizeCellType*

Description

Function to rasterize cell type labels in spatially-resolved omics data represented as SpatialExperiment class.

This function assumes that the input is provided as a `SpatialExperiment` object or a `list` of `SpatialExperiment` objects.

Usage

```
rasterizeCellType(
  input,
  col_name,
  resolution = 100,
  square = TRUE,
  fun = "sum",
  n_threads = 1,
  BPPARAM = NULL,
  verbose = FALSE
)
```

Arguments

`input`	`SpatialExperiment` or `list`: Input data represented as a `SpatialExperiment` or `list` of `SpatialExperiment`. Each `SpatialExperiment` is assumed to have a `colData` slot containing cell type labels for observations as a data frame column and a `spatialCoords` slot containing spatial x,y coordinates of observations as matrix array. Further, x,y coordinates are assumed to be stored in column 1 and 2 of `spatialCoords`.
`col_name`	`character`: Column name of the `colData` object containing cell type labels for observations. If the input is a `list`, col_name is assumed to be present in all elements (`SpatialExperiment`) of the input.
`resolution`	`integer` or `double`: Resolution refers to the side length of each pixel for square pixels and the distance between opposite edges of each pixel for hexagonal pixels. The unit of this parameter is assumed to be the same as the unit of spatial coordinates of the input data.
`square`	`logical`: If TRUE (default), rasterize into square pixels. If FALSE, rasterize into hexagonal pixels.

fun	character: If "mean", pixel value for each pixel would be the proportion of each cell type based on the one-hot-encoded cell type labels for all cells within the pixel. If "sum", pixel value for each pixel would be the number of cells of each cell type based on the one-hot-encoded cell type labels for all cells within the pixel.
n_threads	integer: Number of threads for parallelization. Default = 1. Inputting this argument when the BPPARAM argument is missing would set parallel exeuction back-end to be BiocParallel::MulticoreParam(workers = n_threads). We recommend setting this argument to be the number of cores available (parallel::detectCores(logical = FALSE)). If BPPARAM argument is not missing, the BPPARAM argument would override n_threads argument.
BPPARAM	BiocParallelParam: Optional additional argument for parallelization. This argument is provided for advanced users of BiocParallel for further flexibility for setting up parallel-execution back-end. Default is NULL. If provided, this is assumed to be an instance of BiocParallelParam.
verbose	logical: Whether to display verbose output or warning. Default is FALSE

Value

If the input was given as SpatialExperiment, the output is returned as a new SpatialExperiment object with assay slot containing the feature (cell types) x observations (pixels) matrix (dgCmatrix), spatialCoords slot containing spatial x,y coordinates of pixel centroids, and colData slot containing meta data for pixels (number of cells that were aggregated in each pixel, cell IDs of cells that were aggregated in each pixel, pixel type based on the square argument, pixel resolution based on the resolution argument, pixel geometry as sfc_POLYGON). If the input was provided as list of SpatialExperiment, the output is returned as a new list of SpatialExperiment containing information described above for corresponding SpatialExperiment. Further, names(input) is inherited in the output.

Examples

```
library(SpatialExperiment)

data("merfish_mousePOA")

# check assay names for this particular SpatialExperiment object (you can see
# that cell-type labels are stored in the "celltype" column)
head(colData(merfish_mousePOA))

# rasterize a single SpatialExperiment object
# make sure to specify the col_name argument
out <- rasterizeCellType(merfish_mousePOA, col_name = "celltype", fun = "sum")

# rasterize a single SpatialExperiment object with user-defined resolution and hexagonal pixels
out <- rasterizeCellType(merfish_mousePOA, col_name = "celltype", resolution = 200,
square = FALSE, fun = "sum")

# rasterize a list of SpatialExperiment objects (in this case, permutated datasets
# with 3 different rotations)
spe_list <- permutateByRotation(merfish_mousePOA, n_perm = 3)
```

```
out_list <- rasterizeCellType(spe_list, col_name = "celltype", resolution = 100,
square = TRUE, fun = "sum")
```

`rasterizeGeneExpression` *rasterizeGeneExpression*

Description

Function to rasterize feature x observation matrix in spatially-resolved omics data represented as SpatialExperiment class.

This function assumes that the input is provided as a `SpatialExperiment` object or a `list` of `SpatialExperiment` objects.

Usage

```
rasterizeGeneExpression(
  input,
  assay_name = NULL,
  resolution = 100,
  square = TRUE,
  fun = "mean",
  n_threads = 1,
  BPPARAM = NULL,
  verbose = FALSE
)
```

Arguments

`input`	`SpatialExperiment` or `list`: Input data represented as a `SpatialExperiment` or `list` of `SpatialExperiment`. Each `SpatialExperiment` is assumed to have an `assay` slot containing feature (genes) x observation (cells) matrix as `dgCmatrix` or `matrix` and a `spatialCoords` slot containing spatial x,y coordinates of observations as matrix array. Further, x,y coordinates are assumed to be stored in column 1 and 2 of `spatialCoords`.
`assay_name`	`character`: Name of the assay slot of the input that you want to apply rasterization. If no argument is given, the first assay of the input would be rasterized. This argument is useful when you have both raw and normalized assays stored in the input, and you want to apply rasterization to the normalized assay. If the input is a `list`, assay_name is assumed to be present in all elements (`SpatialExperiment`) of the input.
`resolution`	`integer` or `double`: Resolution refers to the side length of each pixel for square pixels and the distance between opposite edges of each pixel for hexagonal pixels. The unit of this parameter is assumed to be the same as the unit of spatial coordinates of the input data.

square — logical: If TRUE (default), rasterize into square pixels. If FALSE, rasterize into hexagonal pixels.

fun — character: If "mean", pixel value for each pixel would be mean of gene expression for all cells within the pixel. If "sum", pixel value for each pixel would be sum of gene expression for all cells within the pixel.

n_threads — integer: Number of threads for parallelization. Default = 1. Inputting this argument when the BPPARAM argument is missing would set parallel exeuction back-end to be BiocParallel::MulticoreParam(workers = n_threads). We recommend setting this argument to be the number of cores available (parallel::detectCores(logical = FALSE)). If BPPARAM argument is not missing, the BPPARAM argument would override n_threads argument.

BPPARAM — BiocParallelParam: Optional additional argument for parallelization. This argument is provided for advanced users of BiocParallel for further flexibility for setting up parallel-execution back-end. Default is NULL. If provided, this is assumed to be an instance of BiocParallelParam.

verbose — logical: Whether to display verbose output or warning. Default is FALSE

Value

If the input was given as SpatialExperiment, the output is returned as a new SpatialExperiment object with assay slot containing the feature (genes) x observations (pixels) matrix (dgCMatrix or matrix depending on the input, see documentation for rasterizeMatrix), spatialCoords slot containing spatial x,y coordinates of pixel centroids, and colData slot containing meta data for pixels (number of cells that were aggregated in each pixel, cell IDs of cells that were aggregated in each pixel, pixel type based on the square argument, pixel resolution based on the resolution argument, pixel geometry as sfc_POLYGON). If the input was provided as list of SpatialExperiment, the output is returned as a new list of SpatialExperiment containing information described above for corresponding SpatialExperiment. Further, names(input) is inherited in the output.

Examples

```
library(SpatialExperiment)

data("merfish_mousePOA")

# check assay names for this particular SpatialExperiment object (should be "volnorm")
assayNames(merfish_mousePOA)

# rasterize a single SpatialExperiment object
# make sure to specify the assay_name argument when the input SpatialExperiment
# object has multiple assay names (assay_name is used here as an example)
out <- rasterizeGeneExpression(merfish_mousePOA, assay_name = "volnorm", fun = "mean")

# rasterize a single SpatialExperiment object with user-defined resolution and hexagonal pixels
out <- rasterizeGeneExpression(merfish_mousePOA, assay_name = "volnorm", resolution = 200,
square = FALSE, fun = "mean")

# rasterize a list of SpatialExperiment objects (in this case, permutated datasets
# with 3 different rotations)
```

```
spe_list <- permutateByRotation(merfish_mousePOA, n_perm = 3)
out_list <- rasterizeGeneExpression(spe_list, assay_name = "volnorm", resolution = 100,
square = TRUE, fun = "mean")
```

---

rasterizeMatrix *rasterizeMatrix*

Description

Function to rasterize a given input matrix (both dense or sparse) based on a given position matrix.

This function assumes that inputs are provided as a `dgCmatrix` or `matrix` for data and `matrix` for position.

Usage

```
rasterizeMatrix(
  data,
  pos,
  bbox,
  resolution = 100,
  square = TRUE,
  fun = "mean",
  n_threads = 1,
  BPPARAM = NULL,
  verbose = TRUE
)
```

Arguments

`data`	`dgCmatrix` or `matrix`: Feature x observation matrix represented as a `dgCmatrix` or `matrix` object. Features can be genes or cell types. In the case of features being cell types, this matrix is assumed to be a sparse model matrix with rows as cell types and columns as cell IDs.
`pos`	`matrix`: Spatial x,y coordinates of observations stored as a matrix array. Further, x,y coordinates are assumed to be stored in column 1 and 2 of `spatialCoords`.
`bbox`	`bbox` or `numeric`: Bounding box for rasterization defined by a bbox class object (as created by sf::st_bbox) or a numeric vector of length four, with xmin, ymin, xmax and ymax values. Values in a numeric vector are assumed to be in the order of xmin, ymin, xmax, and ymax.
`resolution`	`integer` or `double`: Resolution refers to the side length of each pixel for square pixels and the distance between opposite edges of each pixel for hexagonal pixels. The unit of this parameter is assumed to be the same as the unit of spatial coordinates of the input data.
`square`	`logical`: If TRUE (default), rasterize into square pixels. If FALSE, rasterize into hexagonal pixels.

fun	character: If "mean", pixel value for each pixel would be the average of gene expression for all cells within the pixel. If "sum", pixel value for each pixel would be the sum of gene expression for all cells within the pixel.
n_threads	integer: Number of threads for parallelization. Default = 1. Inputting this argument when the BPPARAM argument is missing would set parallel exeuction back-end to be BiocParallel::MulticoreParam(workers = n_threads). We recommend setting this argument to be the number of cores available (parallel::detectCores(logical = FALSE)). If BPPARAM argument is not missing, the BPPARAM argument would override n_threads argument.
BPPARAM	BiocParallelParam: Optional additional argument for parallelization. This argument is provided for advanced users of BiocParallel for further flexibility for setting up parallel-execution back-end. Default is NULL. If provided, this is assumed to be an instance of BiocParallelParam.
verbose	logical: Whether to display verbose output or warning. Default is FALSE

Value

The output is returned as a list containing rasterized feature x observation matrix as dgCmatrix if data was given as dgCmatrix and as matrix if data was given as matrix, spatial x,y coordinates of pixel centroids as matrix, and data.frame containing meta data for pixels (number of cells that were aggregated in each pixel, cell IDs of cells that were aggregated in each pixel, pixel type based on the square argument, pixel resolution based on the resolution argument, pixel geometry as sfc_POLYGON).

Examples

```
library(SpatialExperiment)
library(sf)

data("merfish_mousePOA")
# extract features-by-cells matrix, spatial coordinates from the SpatialExperiment object
data <- assay(merfish_mousePOA)
pos <- spatialCoords(merfish_mousePOA)
# compute bounding box
resolution <- 100
bbox <- st_bbox(c(
  xmin = floor(min(pos[,1])-resolution/2),
  xmax = ceiling(max(pos[,1])+resolution/2),
  ymin = floor(min(pos[,2])-resolution/2),
  ymax = ceiling(max(pos[,2])+resolution/2)
))
# rasterize with mean as the aggregation function
out_mean <- rasterizeMatrix(data, pos, bbox, resolution = resolution, fun = "mean")
# rasterize with sum as the aggregation function
out_sum <- rasterizeMatrix(data, pos, bbox, resolution = resolution, fun = "sum")
# rasterize with user-defined resolution and hexagonal pixels
# in this case, you need to update the bbox as well
resolution <- 200
bbox <- st_bbox(c(
  xmin = floor(min(pos[,1])-resolution/2),
```

```
  xmax = ceiling(max(pos[,1])+resolution/2),
  ymin = floor(min(pos[,2])-resolution/2),
  ymax = ceiling(max(pos[,2])+resolution/2)
))
out_hex <- rasterizeMatrix(data, pos, bbox, resolution = resolution, square = FALSE, fun = "mean")
```

Index

∗ **datasets**
merfish_mousePOA, 2

merfish_mousePOA, 2

permutateByRotation, 3
plotRaster, 4

rasterizeCellType, 6
rasterizeGeneExpression, 8
rasterizeMatrix, *9*, 10