

Package: BayesSpace (via r-universe)

October 9, 2026

Version 1.23.0

Date 2026-01-08

Title Clustering and Resolution Enhancement of Spatial Transcriptomes

Description Tools for clustering and enhancing the resolution of spatial gene expression experiments. BayesSpace clusters a low-dimensional representation of the gene expression matrix, incorporating a spatial prior to encourage neighboring spots to cluster together. The method can enhance the resolution of the low-dimensional representation into ``sub-spots'', for which features such as gene expression or cell type composition can be imputed.

Depends R (>= 4.0.0), SingleCellExperiment

Imports Rcpp (>= 1.0.4.6), stats, methods, purrr, scatter, scan, SummarizedExperiment, coda, rhdf5, S4Vectors, Matrix, magrittr, assertthat, arrow, mclust, RCurl, DirichletReg, xgboost (>= 3.0.0), utils, dplyr, rlang, ggplot2, tibble, rjson, tidyr, scales, microbenchmark, BiocFileCache, BiocSingular, BiocParallel

License MIT + file LICENSE

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppArmadillo, RcppDist, RcppProgress

NeedsCompilation yes

SystemRequirements C++17

Encoding UTF-8

Suggests testthat, knitr, rmarkdown, igraph, spatialLIBD, viridis, patchwork, RColorBrewer, Seurat

VignetteBuilder knitr

biocViews Software, Clustering, Transcriptomics, GeneExpression, SingleCell, ImmunoOncology, DataImport

BugReports <https://github.com/edward130603/BayesSpace/issues>

URL edward130603.github.io/BayesSpace

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libxml2-dev libssl-dev zlib1g-dev
Repository https://bioc.r-universe.dev
Date/Publication 2026-04-28 12:53:57 UTC
RemoteUrl https://github.com/bioc/BayesSpace
RemoteRef HEAD
RemoteSha 0411308424722c3271e205fde699d86389e6ee90

Contents

clusterPlot	2
enhanceFeatures	3
exampleSCE	5
featurePlot	6
getRDS	7
mcmcChain	8
parallelize	9
qTune	10
readVisium	11
spatialCluster	13
spatialEnhance	15
spatialPreprocess	18
Index	20

clusterPlot	<i>Plot spatial cluster assignments.</i>
-------------	--

Description

Plot spatial cluster assignments.

Usage

```
clusterPlot(  
  sce,  
  label = "spatial.cluster",  
  palette = NULL,  
  color = NULL,  
  platform = NULL,  
  is.enhanced = NULL,  
  nsubspots.per.edge = 3,  
  ...  
)
```

Arguments

sce	SingleCellExperiment. If fill is specified and is a string, it must exist as a column in colData(sce).
label	Labels used to color each spot. May be the name of a column in colData(sce), or a vector of discrete values.
palette	Optional vector of hex codes to use for discrete spot values.
color	Optional hex code to set color of borders around spots. Set to NA to remove borders.
platform	Spatial sequencing platform. If "Visium", the hex spot layout will be used, otherwise square spots will be plotted. NOTE: specifying this argument is only necessary if sce was not created by spatialCluster() or spatialEnhance().
is.enhanced	True if sce contains subspot-level data instead of spots. Spatial sequencing platform. If true, the respective subspot lattice for each platform will be plotted. NOTE: specifying this argument is only necessary if sce was not created by spatialCluster() or spatialEnhance().
nsubspots.per.edge	Number of subspots per edge of the square. Only valid when platform is 'ST' or 'VisiumHD'.
...	Additional arguments for geom_polygon(). size, to specify the linewidth of these borders, is likely the most useful.

Value

Returns a ggplot object.

See Also

Other spatial plotting functions: [featurePlot\(\)](#)

Examples

```
sce <- exampleSCE()
clusterPlot(sce)
```

enhanceFeatures

Predict feature vectors from enhanced PCs.

Description

Predict feature vectors from enhanced PCs.

Usage

```

enhanceFeatures(
  sce.enhanced,
  sce.ref,
  feature_names = NULL,
  model = c("xgboost", "dirichlet", "lm"),
  use.dimred = "PCA",
  assay.type = "logcounts",
  altExp.type = NULL,
  feature.matrix = NULL,
  nrounds = 0,
  train.n = round(ncol(sce.ref) * 2/3),
  nthread = 1L
)

```

Arguments

<code>sce.enhanced</code>	SingleCellExperiment object with enhanced PCs.
<code>sce.ref</code>	SingleCellExperiment object with original PCs and expression.
<code>feature_names</code>	List of genes/features to predict expression/values for.
<code>model</code>	Model used to predict enhanced values.
<code>use.dimred</code>	Name of dimension reduction to use.
<code>assay.type</code>	Expression matrix in <code>assays(sce.ref)</code> to predict.
<code>altExp.type</code>	Expression matrix in <code>altExps(sce.ref)</code> to predict. Overrides <code>assay.type</code> if specified.
<code>feature.matrix</code>	Expression/feature matrix to predict, if not directly attached to <code>sce.ref</code> . Must have columns corresponding to the spots in <code>sce.ref</code> . Overrides <code>assay.type</code> and <code>altExp.type</code> if specified.
<code>nrounds</code>	Nonnegative integer to set the <code>nrounds</code> parameter (max number of boosting iterations) for <code>xgboost</code> . <code>nrounds = 100</code> works reasonably well in most cases. If <code>nrounds</code> is set to 0, the parameter will be tuned using a train-test split. We recommend tuning <code>nrounds</code> for improved feature prediction, but note this will increase runtime.
<code>train.n</code>	Number of spots to use in the training dataset for tuning <code>nrounds</code> . By default, 2/3 the total number of spots are used.
<code>nthread</code>	Number of threads to use for <code>xgboost</code> . By default, 1 thread is used.

Details

Enhanced features are computed by fitting a predictive model to a low-dimensional representation of the original expression vectors. By default, a linear model is fit for each gene using the top 15 principal components from each spot, i.e. `lm(gene ~ PCs)`, and the fitted model is used to predict the enhanced expression for each gene from the subspots' principal components.

Diagnostic measures, such as RMSE for `xgboost` or `R.squared` for linear regression, are added to the 'rowData' of the enhanced experiment if the features are an assay of the original experiment. Otherwise they are stored as an attribute of the returned matrix/altExp.

Note that feature matrices will be returned and are expected to be input as $p \times n$ matrices of p -dimensional feature vectors over the n spots.

Value

If `assay.type` or `altExp.type` are specified, the enhanced features are stored in the corresponding slot of `sce.enhanced` and the modified `SingleCellExperiment` object is returned.

If `feature.matrix` is specified, or if a subset of features are requested, the enhanced features are returned directly as a matrix.

Examples

```
set.seed(149)
sce <- exampleSCE()
sce <- spatialCluster(sce, 7, nrep = 100, burn.in = 10)
enhanced <- spatialEnhance(sce, 7, init = sce$spatial.cluster, nrep = 100, burn.in = 10)
enhanced <- enhanceFeatures(enhanced, sce, feature_names = c("gene_1", "gene_2"))
```

exampleSCE	<i>Create minimal SingleCellExperiment for documentation examples.</i>
------------	--

Description

Create minimal `SingleCellExperiment` for documentation examples.

Usage

```
exampleSCE(nrow = 8, ncol = 12, n_genes = 100, n_PCs = 10)
```

Arguments

<code>nrow</code>	Number of rows of spots
<code>ncol</code>	Number of columns of spots
<code>n_genes</code>	Number of genes to simulate
<code>n_PCs</code>	Number of principal components to include

Details

Inspired by `scuttle`'s `mockSCE()`.

Value

A `SingleCellExperiment` object with simulated counts, corresponding logcounts and PCs, and positional data in `colData`. Spots are distributed over an (`nrow` x `ncol`) rectangle.

Examples

```
set.seed(149)
sce <- exampleSCE()
```

featurePlot	<i>Plot spatial gene expression.</i>
-------------	--------------------------------------

Description

Plot spatial gene expression.

Usage

```
featurePlot(
  sce,
  feature,
  assay.type = "logcounts",
  diverging = FALSE,
  low = NULL,
  high = NULL,
  mid = NULL,
  color = NULL,
  platform = NULL,
  is.enhanced = NULL,
  nsubspots.per.edge = 3,
  ...
)
```

Arguments

sce	SingleCellExperiment. If feature is specified and is a string, it must exist as a row in the specified assay of sce.
feature	Feature vector used to color each spot. May be the name of a gene/row in an assay of sce, or a vector of continuous values.
assay.type	String indicating which assay in sce the expression vector should be taken from.
diverging	If true, use a diverging color gradient in featurePlot() (e.g. when plotting a fold change) instead of a sequential gradient (e.g. when plotting expression).
low, mid, high	Optional hex codes for low, mid, and high values of the color gradient used for continuous spot values.
color	Optional hex code to set color of borders around spots. Set to NA to remove borders.
platform	Spatial sequencing platform. If "Visium", the hex spot layout will be used, otherwise square spots will be plotted. NOTE: specifying this argument is only necessary if sce was not created by spatialCluster() or spatialEnhance().

is.enhanced	True if sce contains subspot-level data instead of spots. Spatial sequencing platform. If true, the respective subspot lattice for each platform will be plotted. NOTE: specifying this argument is only necessary if sce was not created by <code>spatialCluster()</code> or <code>spatialEnhance()</code> .
nsubspots.per.edge	Number of subspots per edge of the square. Only valid when platform is 'ST' or 'VisiumHD'.
...	Additional arguments for <code>geom_polygon()</code> . <code>size</code> , to specify the linewidth of these borders, is likely the most useful.

Value

Returns a ggplot object.

See Also

Other spatial plotting functions: [clusterPlot\(\)](#)

Examples

```
sce <- exampleSCE()
featurePlot(sce, "gene_2")
```

getRDS

Download a processed sample from our S3 bucket

Description

Datasets are cached locally using BiocFileCache. The first time using this function, you may need to consent to creating a BiocFileCache directory if one does not already exist.

Usage

```
getRDS(dataset, sample, cache = TRUE)
```

Arguments

dataset	Dataset identifier
sample	Sample identifier
cache	If true, cache the dataset locally with BiocFileCache. Otherwise, download directly from our S3 bucket. Caching saves time on subsequent loads, but consumes disk space.

Details

The following datasets are available via `getRDS`.

Dataset	Sample(s)
2018_thrane_melanoma	ST_mel1_rep2
2020_maynard_prefrontal-cortex	151507, 151508, 151509, 151510, 151669, 151670, 151671, 151672, 151673, 151674, 1
2020_ji_squamous-cell-carcinoma	P4_rep1
2020_10X-IDC	IDC1
2020_10X-demo_ovarian-cancer	whole_transcriptome

Value

sce A `SingleCellExperiment` with positional information in `colData` and PCs based on the top 2000 HVGs

Examples

```
sce <- getRDS("2018_thrane_melanoma", "ST_mel1_rep2", cache = FALSE)
```

mcmcChain	<i>Read MCMC chain associated with a BayesSpace clustering or enhancement</i>
-----------	---

Description

BayesSpace stores the MCMC chain associated with a clustering or enhancement on disk in an HDF5 file. The `mcmcChain()` function reads any parameters specified by the user into a `coda::mcmc` object compatible with `TidyBayes`.

Usage

```
mcmcChain(sce, params = NULL)

removeChain(sce)
```

Arguments

sce	SingleCellExperiment with a file path stored in its metadata.
params	List of model parameters to read

Details

To interact with the HDF5 file directly, obtain the filename from the SingleCellExperiment's meta-data: `metadata(sce)$chain.h5`. Each parameter is stored as a separate dataset in the file, and is represented as a matrix of size (n_iterations x n_parameter_indices). Parameter choices for the spot-level clustering include:

- `z` (cluster assignments)
- `weights` (w_i)
- `mu` (mean vectors)
- `lambda` (precision matrix)
- `plogLik` (pseudo-log-likelihood)

Parameter choices for the subspot-level enhanced clustering include:

- `z` (cluster assignments)
- `weights` (w_i)
- `Y` (enhanced PCs)
- `mu` (mean vectors)
- `lambda` (precision matrix)
- `Ychange` (acceptance rate for the jittering of PCs)

For best results, `Ychange` should average between 0.25 and 0.40.

Value

Returns an `mcmc` object containing the values of the requested parameters over the constructed chain.

Examples

```
set.seed(149)
sce <- exampleSCE()
sce <- spatialCluster(sce, 7, nrep=100, burn.in=10, save.chain=TRUE)
chain <- mcmcChain(sce)
removeChain(sce)
```

parallelize

Parallelization

Description

A convenient wrapper function of `BiocParallel` providing easy parallelization.

Usage

```
paraLapply(  
  X,  
  FUN,  
  BPPARAM = NULL,  
  cores = 1L,  
  type = c("serial", "fork", "sock"),  
  verbose = FALSE,  
  ...  
)
```

Arguments

X	Any object for which methods length, [, and [[are implemented (passed to bplapply).
FUN	The function to be applied to each element of X (passed to bplapply).
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to BiocParallel functions.
cores	The number of threads to use. The results are invariate to the value of cores.
type	One of "serial", "fork", or "sock". When cores is one, type is always "serial". Both "fork" and "sock" are for multi-threading. "fork" is faster, but only supports linux and macos. "sock" supports linux, macos, and windows.
verbose	Whether to print debug information or not.
...	Additional parameters passed to bplapply.

Value

See lapply.

qTune	<i>Tuning the choice of q (number of clusters) before running spatial-Cluster</i>
-------	---

Description

Before running spatialCluster(), we recommend tuning the choice of q by choosing the q that minimizes the model’s negative log likelihood over early iterations. qTune() computes the average negative log likelihood for a range of q values over iterations 100:1000, and qPlot() displays the results.

Usage

```
qPlot(sce, qs = seq(3, 7), force.retune = FALSE, ...)  
  
qTune(sce, qs = seq(3, 7), burn.in = 100, nrep = 1000, cores = 1L, ...)
```

Arguments

sce	A SingleCellExperiment object containing the spatial data.
qs	The values of q to evaluate.
force.retune	If specified, existing tuning values in sce will be overwritten.
...	Other parameters are passed to spatialCluster().
burn.in, nrep	Integers specifying the range of repetitions to compute.
cores	The number of threads to use. The results are invariate to the value of cores.

Details

qTune() takes the same parameters as spatialCluster() and will run the MCMC clustering algorithm up to nrep iterations for each value of q. The first burn.in iterations are discarded as burn-in and the log likelihood is averaged over the remaining iterations.

qPlot() plots the computed negative log likelihoods as a function of q. If qTune() was run previously, i.e. there exists an attribute of sce named "q.logliks", the pre-computed results are displayed. Otherwise, or if force.retune is specified, qplot() will automatically run qTune() before plotting (and can take the same parameters as spatialCluster()).

Value

qTune() returns a modified sce with tuning log likelihoods stored as an attribute named "q.logliks".

qPlot() returns a ggplot object.

Examples

```
set.seed(149)
sce <- exampleSCE()
sce <- qTune(sce, seq(3, 7), burn.in = 10, nrep = 100)
qPlot(sce)
```

readVisium

Load a Visium spatial dataset as a SingleCellExperiment.

Description

Load a Visium spatial dataset as a SingleCellExperiment.

Usage

```
readVisium(
  dirname,
  rm.feats.pat = c("^NegControl.*", "^BLANK.*", "^DEPRECATED.*")
)
```

```

read10Xh5(
  dirname,
  fname = "filtered_feature_bc_matrix.h5",
  rm.feats.pat = c("^NegControl.*", "^BLANK.*", "^DEPRECATED.*")
)

counts2h5(dirname)

```

Arguments

<code>dirname</code>	Path to spaceranger output directory (e.g. "sampleID/outs/"). This directory must contain the counts matrix and feature/barcode TSVs in <code>filtered_feature_bc_matrix/</code> for <code>readVisium</code> , or in <code>filtered_feature_bc_matrix.h5</code> for <code>read10Xh5</code> . Besides, it must also contain a file for spot positions named <code>spatial/tissue_positions_list.csv</code> (before Space Ranger V2.0) or <code>spatial/tissue_positions.csv</code> (since Space Ranger V2.0), as well as a file containing scale factors named <code>spatial/scalefactors_json.json</code> . (To understand the output directory, refer to the corresponding 10X Genomics help page .)
<code>rm.feats.pat</code>	Patterns for features (genes) to remove.
<code>fname</code>	File name of the h5 file. It should be inside <code>dirname</code> . (By default "filtered_feature_bc_matrix.h5")

Details

We store two variables associated with downstream BayesSpace functions in a list called `BayesSpace.data` in the `SingleCellExperiment`'s metadata.

- `platform` is set to "Visium", and is used to determine spot layout and neighborhood structure.
- `is.enhanced` is set to `FALSE` to denote the object contains spot-level data.

Value

`SingleCellExperiment` containing the counts matrix in `counts` and spatial data in `colData`. Array coordinates for each spot are stored in columns `array_row` and `array_col`, while image coordinates are stored in columns `pxl_row_in_fullres` and `pxl_col_in_fullres`.

Examples

```

## Not run:
sce <- readVisium("path/to/outs/")

## End(Not run)

```

spatialCluster	<i>Spatial clustering</i>
----------------	---------------------------

Description

Cluster a spatial expression dataset.

Usage

```
spatialCluster(
  sce,
  q,
  use.dimred = "PCA",
  d = 15,
  platform = c("Visium", "VisiumHD", "ST"),
  init = NULL,
  init.method = c("mclust", "kmeans"),
  model = c("t", "normal"),
  precision = c("equal", "variable"),
  nrep = 50000,
  burn.in = 1000,
  thin = 100,
  gamma = NULL,
  mu0 = NULL,
  lambda0 = NULL,
  alpha = 1,
  beta = 0.01,
  save.chain = FALSE,
  chain.fname = NULL
)
```

Arguments

sce	A SingleCellExperiment object containing the spatial data.
q	The number of clusters.
use.dimred	Name of a reduced dimensionality result in reducedDims(sce). If provided, cluster on these features directly.
d	Number of top principal components to use when clustering.
platform	Spatial transcriptomic platform. Specify 'Visium' for hex lattice geometry or 'ST' and 'VisiumHD' for square lattice geometry. Specifying this parameter is optional when analyzing SingleCellExperiments processed using readVisium , spatialPreprocess , or spatialCluster , as this information is included in their metadata.
init	Initial cluster assignments for spots.
init.method	If init is not provided, cluster the top d PCs with this method to obtain initial cluster assignments.

model	Error model. ('normal' or 't')
precision	Covariance structure. ('equal' or 'variable' for EEE and VVV covariance models, respectively.)
nrep	The number of MCMC iterations.
burn.in	The number of MCMC iterations to exclude as burn-in period.
thin	Thinning rate.
gamma	Smoothing parameter. Defaults to 2 for platform="ST" and 3 for platform="Visium". (Values in range of 1-3 seem to work well.)
mu0	Prior mean hyperparameter for mu. If not provided, mu0 is set to the mean of PCs over all spots.
lambda0	Prior precision hyperparam for mu. If not provided, lambda0 is set to a diagonal matrix $0.01I$.
alpha	Hyperparameter for Wishart distributed precision lambda.
beta	Hyperparameter for Wishart distributed precision lambda.
save.chain	If true, save the MCMC chain to an HDF5 file.
chain.fname	File path for saved chain. Tempfile used if not provided.

Details

The input SCE must have row and col columns in its colData, corresponding to the array row and column coordinates of each spot. These are automatically parsed by [readVisium](#) or can be added manually when creating the SCE.

Cluster labels are stored in the spatial.cluster column of the SCE, and the cluster initialization is stored in cluster.init.

Value

Returns a modified sce with cluster assignments stored in colData under the name spatial.cluster.

See Also

[spatialPreprocess](#) for preparing the SCE for clustering, [spatialEnhance](#) for enhancing the clustering resolution, [clusterPlot](#) for visualizing the cluster assignments, [featurePlot](#) for visualizing expression levels in spatial context, and [mcmcChain](#) for examining the full MCMC chain associated with the clustering.

Examples

```
set.seed(149)
sce <- exampleSCE()
sce <- spatialCluster(sce, 7, nrep = 100, burn.in = 10)
```

spatialEnhance	<i>Enhance spot resolution</i>
----------------	--------------------------------

Description

Enhanced clustering of a spatial expression dataset to subspot resolution.

Usage

```
spatialEnhance(
  sce,
  q,
  platform = c("Visium", "VisiumHD", "ST"),
  use.dimred = "PCA",
  d = 15,
  nsubspots.per.edge = 3,
  init = NULL,
  init.method = c("spatialCluster", "mclust", "kmeans"),
  model = c("t", "normal"),
  nrep = 1e+05,
  gamma = NULL,
  mu0 = NULL,
  lambda0 = NULL,
  alpha = 1,
  beta = 0.01,
  save.chain = FALSE,
  chain.fname = NULL,
  burn.in = 10000,
  thin = 100,
  jitter.scale = 5,
  jitter.prior = 0.3,
  adapt.before = burn.in,
  cores = 1,
  verbose = FALSE
)

coreTune(sce, test.cores = detectCores(), test.times = 1, ...)

adjustClusterLabels(sce, burn.in)
```

Arguments

sce	A SingleCellExperiment object containing the spatial data.
q	The number of clusters.
platform	Spatial transcriptomic platform. Specify 'Visium' for hex lattice geometry or 'ST' and 'VisiumHD' for square lattice geometry. Specifying this parameter is

	optional when analyzing SingleCellExperiments processed using readVisium , spatialPreprocess , or spatialCluster , as this information is included in their metadata.
<code>use.dimred</code>	Name of a reduced dimensionality result in <code>reducedDims(sce)</code> . If provided, cluster on these features directly.
<code>d</code>	Number of top principal components to use when clustering.
<code>nsubspots.per.edge</code>	Number of subspots per edge of the square. Only valid when <code>platform</code> is 'ST' or 'VisiumHD'.
<code>init</code>	Initial cluster assignments for spots.
<code>init.method</code>	If <code>init</code> is not provided, cluster the top <code>d</code> PCs with this method to obtain initial cluster assignments.
<code>model</code>	Error model. ('normal' or 't')
<code>nrep</code>	The number of MCMC iterations.
<code>gamma</code>	Smoothing parameter. (Values in range of 1-3 seem to work well.)
<code>mu0</code>	Prior mean hyperparameter for <code>mu</code> . If not provided, <code>mu0</code> is set to the mean of PCs over all spots.
<code>lambda0</code>	Prior precision hyperparam for <code>mu</code> . If not provided, <code>lambda0</code> is set to a diagonal matrix $0.01I$.
<code>alpha</code>	Hyperparameter for Wishart distributed precision <code>lambda</code> .
<code>beta</code>	Hyperparameter for Wishart distributed precision <code>lambda</code> .
<code>save.chain</code>	If true, save the MCMC chain to an HDF5 file.
<code>chain.fname</code>	File path for saved chain. Tempfile used if not provided.
<code>burn.in</code>	Number of iterations to exclude as burn-in period. The MCMC iterations are currently thinned to every 100; accordingly <code>burn.in</code> is rounded down to the nearest multiple of 100. If a value no larger than 1 is set, it is considered as a percentage. It is always considered as percentage for <code>adjustClusterLabels</code> .
<code>thin</code>	Thinning rate.
<code>jitter.scale</code>	Controls the amount of jittering. Small amounts of jittering are more likely to be accepted but result in exploring the space more slowly. We suggest tuning <code>jitter.scale</code> so that <code>Ychange</code> is on average around 25%-40%. <code>Ychange</code> can be accessed via <code>mcmcChain()</code> . Alternatively, set it to 0 to activate adaptive MCMC.
<code>jitter.prior</code>	Scale factor for the prior variance, parameterized as the proportion (default = 0.3) of the mean variance of the PCs. We suggest making <code>jitter.prior</code> smaller if the jittered values are not expected to vary much from the overall mean of the spot.
<code>adapt.before</code>	Adapting the MCMC chain before the specified number or proportion of iterations (by default equal to <code>burn.in</code> ; set to 0 to always adapt). Only valid when <code>jitter.scale</code> is 0.
<code>cores</code>	The number of threads to use. The results are invariant to the value of <code>cores</code> .
<code>verbose</code>	Log progress to <code>stderr</code> .

<code>test.cores</code>	Either a list of, or a maximum number of cores to test. In the latter case, a list of values (power of 2) will be created
<code>test.times</code>	Times to repeat the benchmarking with microbenchmark.
<code>...</code>	Arguments for <code>spatialEnhance</code> (except for <code>cores</code>).

Details

The enhanced `SingleCellExperiment` has most of the properties of the input SCE - `rowData`, `colData`, `reducedDims` - but does not include expression data in counts or logcounts. To impute enhanced expression vectors, please use `[enhanceFeatures()]` after running `spatialEnhance`.

The `colData` of the enhanced `SingleCellExperiment` includes the following columns to permit referencing the subspots in spatial context and linking back to the original spots:

- `spot.idx`: Index of the spot this subspot belongs to (with respect to the input SCE).
- `subspot.idx`: Index of the subspot within its parent spot.
- `spot.row`: Array row of the subspot's parent spot.
- `spot.col`: Array col of the subspot's parent spot.
- `array_row`: Array row of the subspot. This is the parent spot's row plus an offset based on the subspot's position within the spot.
- `array_col`: Array col of the subspot. This is the parent spot's col plus an offset based on the subspot's position within the spot.
- `pxl_row_in_fullres`: Pixel row of the subspot. This is the parent spot's row plus an offset based on the subspot's position within the spot.
- `pxl_col_in_fullres`: Pixel col of the subspot. This is the parent spot's col plus an offset based on the subspot's position within the spot.

Value

`spatialEnhance` returns a new `SingleCellExperiment` object. By default, the assays of this object are empty, and the enhanced resolution PCs are stored as a reduced dimensionality result accessible with `reducedDim(sce, 'PCA')`.

`coresTune` returns the output of microbenchmark.

`adjustClusterLabels` adjusts the cluster labels from the MCMC samples via `burn.in`, the percentage of samples to drop. The MCMC chain must be retained.

See Also

[spatialCluster](#) for clustering at the spot level before enhancing, [clusterPlot](#) for visualizing the cluster assignments, [enhanceFeatures](#) for imputing enhanced expression, and [mcmcChain](#) for examining the full MCMC chain associated with the enhanced clustering. .

Examples

```
set.seed(149)
sce <- exampleSCE()
sce <- spatialCluster(sce, 7, nrep = 100, burn.in = 10)
enhanced <- spatialEnhance(sce, 7, nrep = 100, burn.in = 10)
```

spatialPreprocess *Preprocess a spatial dataset for BayesSpace*

Description

Adds metadata required for downstream analyses, and (optionally) performs PCA on log-normalized expression of top HVGs.

Usage

```
spatialPreprocess(
  sce,
  platform = c("Visium", "VisiumHD", "ST"),
  n.PCs = 15,
  n.HVGs = 2000,
  skip.PCA = FALSE,
  log.normalize = TRUE,
  assay.type = "logcounts",
  BSPARAM = ExactParam(),
  BPPARAM = SerialParam()
)
```

Arguments

sce	SingleCellExperiment to preprocess
platform	Spatial sequencing platform. Used to determine spot layout and neighborhood structure (Visium = hex, VisiumHD = square, ST = square).
n.PCs	Number of principal components to compute. We suggest using the top 15 PCs in most cases.
n.HVGs	Number of highly variable genes to run PCA upon.
skip.PCA	Skip PCA (if dimensionality reduction was previously computed.)
log.normalize	Whether to log-normalize the input data with scater. May be omitted if log-normalization previously computed.
assay.type	Name of assay in sce containing normalized counts. Leave as "logcounts" unless you explicitly pre-computed a different normalization and added it to sce under another assay. Note that we do not recommend running BayesSpace on PCs computed from raw counts.
BSPARAM	A BiocSingularParam object specifying which algorithm should be used to perform the PCA. By default, an exact PCA is performed, as current spatial datasets are generally small (<10,000 spots). To perform a faster approximate PCA, please specify FastAutoParam() and set a random seed to ensure reproducibility.
BPPARAM	A BiocParallelParam object specifying whether to model the gene variation in parallel or not (default to SerialParam()). To perform faster modeling, please specify SnowParam() or MulticoreParam() .

Value

SingleCellExperiment with PCA and BayesSpace metadata

Examples

```
sce <- exampleSCE()  
sce <- spatialPreprocess(sce)
```

Index

* **spatial plotting functions**

clusterPlot, [2](#)

featurePlot, [6](#)

adjustClusterLabels (spatialEnhance), [15](#)

BiocParallelParam, [18](#)

BiocSingularParam, [18](#)

clusterPlot, [2](#), [7](#), [14](#), [17](#)

coreTune (spatialEnhance), [15](#)

counts2h5 (readVisium), [11](#)

enhanceFeatures, [3](#), [17](#)

exampleSCE, [5](#)

featurePlot, [3](#), [6](#), [14](#)

getRDS, [7](#)

mcmcChain, [8](#), [14](#), [17](#)

paraLapply (parallelize), [9](#)

parallelize, [9](#)

qPlot (qTune), [10](#)

qTune, [10](#)

read10Xh5 (readVisium), [11](#)

readVisium, [11](#), [13](#), [14](#), [16](#)

removeChain (mcmcChain), [8](#)

spatialCluster, [13](#), [13](#), [16](#), [17](#)

spatialEnhance, [14](#), [15](#)

spatialPreprocess, [13](#), [14](#), [16](#), [18](#)