Package ‘BayesSpace’

June 11, 2021

Version 1.3.0
Date 2021-05-12
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, purrr, scater, scran, SummarizedExperiment, coda, rhdf5, S4Vectors, Matrix, assertthat, mclust, RCurl, DirichletReg, xgboost, utils, ggplot2, scales, BiocFileCache, BiocSingular
License MIT + file LICENSE
RoxygenNote 7.1.1
LinkingTo Rcpp, RcppArmadillo, RcppDist, RcppProgress
NeedsCompilation yes
SystemRequirements C++11
Encoding UTF-8
Suggests testthat, knitr, rmarkdown, igraph, spatialLIBD, dplyr, 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
git_url https://git.bioconductor.org/packages/BayesSpace
git_branch master
git_last_commit 03d5e05
clusterPlot

Plot spatial cluster assignments.

Description

Plot spatial cluster assignments.

Usage

```r
clusterPlot(
  sce,
  label = "spatial.cluster",
  palette = NULL,
  color = NULL,
  platform = NULL,
  is.enhanced = NULL,
  ...
)
```
**enhanceFeatures**

**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()`.
- **...**: 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**

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

---

**enhanceFeatures**

Predict feature vectors from enhanced PCs.

**Description**

Predict feature vectors from enhanced PCs.
enhanceFeatures

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)
)

Arguments

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

```r
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"))
```

---

**Description**

Create minimal SingleCellExperiment for documentation examples.

**Usage**

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

**Arguments**

- `nrow`: Number of rows of spots
- `ncol`: Number of columns of spots
- `n_genes`: Number of genes to simulate
- `n_PCs`: 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**

```r
set.seed(149)
sce <- exampleSCE()
```
**featurePlot**

*Plot spatial gene expression.*

**Description**

Plot spatial gene expression.

**Usage**

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

**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 spatialCluster() or spatialEnhance().
- **...** Additional arguments for geom_polygon(). size, to specify the linewidth of these borders, is likely the most useful.
getRDS

Value

Returns a ggplot object.

See Also

Other spatial plotting functions: `clusterPlot()`

Examples

```r
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

```r
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.

Value

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

Examples

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

Read MCMC chain associated with a BayesSpace clustering or enhancement

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

```r
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 metadata: `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)

Value

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

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

Description

Before running `spatialCluster()`, we recommend tuning the choice of \( q \) by choosing the \( q \) that maximizes 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

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

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.

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

```r
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

```r
readVisium(dirname)
```

### Arguments

- `dirname`  
  Path to spaceranger output directory (e.g. "sampleID/outs/"). This directory must contain the counts matrix and feature/barcode TSVs in `filtered_feature_bc_matrix/`, and the spot positions at `spatial/tissue_positions_list.csv`. (These are default locations for spaceranger outputs.)

### 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 `row` and `col`, while image coordinates are stored in columns `imagerow` and `imagecol`.

### Examples

```r
## Not run:
sce <- readVisium("path/to/outs/")
## End(Not run)
```
spatialCluster

spatialCluster  Spatial clustering

Description

Cluster a spatial expression dataset.

Usage

spatialCluster(
  sce,
  q,
  use.dimred = "PCA",
  d = 15,
  platform = c("Visium", "ST"),
  init = NULL,
  init.method = c("mclust", "kmeans"),
  model = c("t", "normal"),
  precision = c("equal", "variable"),
  nrep = 50000,
  burn.in = 1000,
  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' for square lattice geometry. Specifying this parameter is optional when analyzing SingleCellExperiments processed using readVisium or spatialPreprocess, 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')
**spatialCluster**

- **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.
- **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.
- `mcmcChain` for examining the full MCMC chain associated with the clustering.

**Examples**

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

Enhance spot resolution

Description

Enhanced clustering of a spatial expression dataset to subspot resolution.

Usage

spatialEnhance(
  sce,
  q,
  platform = c("Visium", "ST"),
  use.dimred = "PCA",
  d = 15,
  init = NULL,
  init.method = c("spatialCluster", "mclust", "kmeans"),
  model = c("t", "normal"),
  nrep = 2e+05,
  gamma = NULL,
  mu0 = NULL,
  lambda0 = NULL,
  alpha = 1,
  beta = 0.01,
  save.chain = FALSE,
  chain.fname = NULL,
  burn.in = 10000,
  jitter_scale = 5,
  jitter_prior = 0.3,
  verbose = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sce</td>
<td>A SingleCellExperiment object containing the spatial data.</td>
</tr>
<tr>
<td>q</td>
<td>The number of clusters.</td>
</tr>
<tr>
<td>platform</td>
<td>Spatial transcriptomic platform. Specify 'Visium' for hex lattice geometry or 'ST' for square lattice geometry. Specifying this parameter is optional when analyzing SingleCellExperiments processed using <code>readVisium</code>, <code>spatialPreprocess</code>, or <code>spatialCluster</code>, as this information is included in their metadata.</td>
</tr>
<tr>
<td>use.dimred</td>
<td>Name of a reduced dimensionality result in <code>reducedDims(sce)</code>. If provided, cluster on these features directly.</td>
</tr>
<tr>
<td>d</td>
<td>Number of top principal components to use when clustering.</td>
</tr>
<tr>
<td>init</td>
<td>Initial cluster assignments for spots.</td>
</tr>
<tr>
<td>init.method</td>
<td>If <code>init</code> is not provided, cluster the top <code>d</code> PCs with this method to obtain initial cluster assignments.</td>
</tr>
</tbody>
</table>
model Error model. ('normal' or 't')
nrep The number of MCMC iterations.
gamma Smoothing parameter. (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.
burn.in Number of iterations to exclude as burn-in period. The MCMC iterations are currently thinned to every 100; accordingly burn.in is rounded down to the nearest multiple of 100.
jitter_scale 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 jitter_scale so that Ychange is on average around 25%-40%.
jitter_prior Scale factor for the prior variance, parameterized as the proportion (default = 0.3) of the mean variance of the PCs. We suggest making jitter_prior smaller if the jittered values are not expected to vary much from the overall mean of the spot.
verbose Log progress to stderr.

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.
- **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.
- **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.
- **imagerow**: Pixel row of the subspot. This is the parent spot’s row plus an offset based on the subspot’s position within the spot.
- **imagecol**: Pixel col of the subspot. This is the parent spot’s col plus an offset based on the subspot’s position within the spot.
spatialPreprocess

Description

Preprocess a spatial dataset for BayesSpace

Value

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').

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

```r
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

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

Arguments

- **sce**
  - SingleCellExperiment to preprocess

- **platform**
  - Spatial sequencing platform. Used to determine spot layout and neighborhood structure (Visium = hex, ST = square).

- **n.PCs**
  - Number of principal components to compute. We suggest using the top 15 PCs in most cases.
spatialPreprocess

- **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.

**Value**

SingleCellExperiment with PCA and BayesSpace metadata

**Examples**

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

* spatial plotting functions
  
  clusterPlot.2
  featurePlot.6

BiocSingularParam, 16

clusterPlot.2, 7, 12, 15

enhanceFeatures.3, 15

exampleSCE.5

featurePlot.3, 6, 12

getRDS.7

mcmcChain, 8, 12, 15

qPlot (qTune), 9

qTune, 9

readVisium, 10, 11–13

removeChain (mcmcChain), 8

spatialCluster, 11, 13, 15

spatialEnhance, 12, 13

spatialPreprocess, 11–13, 15