Package ‘slingshot’

October 16, 2019

Title Tools for ordering single-cell sequencing

Version 1.2.0

Description Provides functions for inferring continuous, branching lineage structures in low-dimensional data. Slingshot was designed to model developmental trajectories in single-cell RNA sequencing data and serve as a component in an analysis pipeline after dimensionality reduction and clustering. It is flexible enough to handle arbitrarily many branching events and allows for the incorporation of prior knowledge through supervised graph construction.

License Artistic-2.0

Depends R (>= 3.5), princurve (>= 2.0.4), stats

Imports ape, clusterExperiment, graphics, grDevices, igraph, matrixStats, methods, rgl, SingleCellExperiment, SummarizedExperiment

Suggests BiocGenerics, BiocStyle, destiny, gam, knitr, mclust, RColorBrewer, rmarkdown, testthat, covr

VignetteBuilder knitr

LazyData false

RoxygenNote 6.1.1

biocViews Clustering, DifferentialExpression, GeneExpression, RNASeq, Sequencing, Software, Sequencing, SingleCell, Transcriptomics, Visualization

BugReports `https://github.com/kstreet13/slingshot/issues`

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

git_branch RELEASE_3_9

git_last_commit b8962fe

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author Kelly Street [aut, cre, cph],
Davide Risso [aut],
Diya Das [aut],
Sandrine Dudoit [ths],
Koen Van den Berge [ctb],
Robrecht Cannoodt [ctb]

Maintainer Kelly Street `<street.kelly@gmail.com>`

R topics documented:

`getCurves` *Construct Smooth Lineage Curves*

Description

This function takes a reduced data matrix `n` by `p`, a vector of cluster identities (optionally including `-1`'s for "unclustered"), and a set of lineages consisting of paths through a forest constructed on the clusters. It constructs smooth curves for each lineage and returns the points along these curves corresponding to the orthogonal projections of each data point, along with corresponding arclength (`pseudotime` or `lambda`) values.

Usage

```
getCurves(sds, ...)

## S4 method for signature 'SlingshotDataSet'
getCurves(sds, shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)
```

Arguments

`sds`	The `SlingshotDataSet` for which to construct simultaneous principal curves. This should already have lineages identified by `getLineages`.
`...`	Additional parameters to pass to scatter plot smoothing function, `smoother`.
`shrink`	logical or numeric between 0 and 1, determines whether and how much to shrink branching lineages toward their average prior to the split.

`extend`	character, how to handle root and leaf clusters of lineages when constructing the initial, piece-wise linear curve. Accepted values are `'y'` (default), `'n'`, and `'pc1'`. See 'Details' for more.
`reweight`	logical, whether to allow cells shared between lineages to be reweighted during curve-fitting. If `TRUE`, cells shared between lineages will be iteratively reweighted based on the quantiles of their projection distances to each curve. See 'Details' for more.
`reassign`	logical, whether to reassign cells to lineages at each iteration. If `TRUE`, cells will be added to a lineage when their projection distance to the curve is less than the median distance for all cells currently assigned to the lineage. Additionally, shared cells will be removed from a lineage if their projection distance to the curve is above the 90th percentile and their weight along the curve is less than `0.1`.
`thresh`	numeric, determines the convergence criterion. Percent change in the total distance from cells to their projections along curves must be less than `thresh`. Default is `0.001`, similar to `principal_curve`.
`maxit`	numeric, maximum number of iterations, see `principal_curve`.
`stretch`	numeric factor by which curves can be extrapolated beyond endpoints. Default is 2, see `principal_curve`.
`smoother,`	choice of scatter plot smoother. Same as `principal_curve`, but `"lowess"` option is replaced with `"loess"` for additional flexibility.
`shrink.method`	character denoting how to determine the appropriate amount of shrinkage for a branching lineage. Accepted values are the same as for `kernel` in `density` (default is `"cosine"`), as well as `"tricube"` and `"density"`. See 'Details' for more.
`allow.breaks`	logical, determines whether curves that branch very close to the origin should be allowed to have different starting points.

Details

When there is only a single lineage, the curve-fitting algorithm is nearly identical to that of `principal_curve`. When there are multiple lineages and `shrink > 0`, an additional step is added to the iterative procedure, forcing curves to be similar in the neighborhood of shared points (ie., before they branch).

The `extend` argument determines how to construct the piece-wise linear curve used to initiate the recursive algorithm. The initial curve is always based on the lines between cluster centers and if `extend = 'n'`, this curve will terminate at the center of the endpoint clusters. Setting `extend = 'y'` will allow the first and last segments to extend beyond the cluster center to the orthogonal projection of the furthest point. Setting `extend = 'pc1'` is similar to `'y'`, but uses the first principal component of the cluster to determine the direction of the curve beyond the cluster center. These options typically have little to no impact on the final curve, but can occasionally help with stability issues.

When `shink = TRUE`, we compute a shrinkage curve, $w_l(t)$, for each lineage, a non-increasing function of pseudotime that determines how much that lineage should be shrunk toward a shared average curve. We set $w_l(0) = 1$, so that the curves will perfectly overlap the average curve at pseudotime `0`. The weighting curve decreases from `1` to `0` over the non-outlying pseudotime values of shared cells (where outliers are defined by the `1.5*IQR` rule). The exact shape of the curve in this region is controlled by `shrink.method`, and can follow the shape of any standard kernel function's cumulative density curve (or more precisely, survival curve, since we require a decreasing function). Different choices of `shrink.method` seem to have little impact on the final curves, in most cases.

When `reweight = TRUE`, weights for shared cells are based on the quantiles of their projection distances onto each curve. The distances are ranked and converted into quantiles between `0` and `1`, which are then transformed by `1 -q^2`. Each cell's weight along a given lineage is the ratio of this value to the maximum value for this cell across all lineages.

Value

An updated `SlingshotDataSet` object containing the oringinal input, arguments provided to `getCurves` as well as the following new elements:

- curvesA list of `principal_curve` objects.
- slingParamsAdditional parameters used for fitting simultaneous principal curves.

References

Hastie, T., and Stuetzle, W. (1989). "Principal Curves." *Journal of the American Statistical Association*, 84:502–516.

See Also

`slingshot`

Examples

```
data("slingshotExample")
sds <- getLineages(rd, cl, start.clus = '1')
sds <- getCurves(sds)

plot(rd, col = cl, asp = 1)
lines(sds, type = 'c', lwd = 3)
```

getLineages — *Infer Lineage Structure from Clustered Samples*

Description

Given a reduced-dimension data matrix `n` by `p` and a vector of cluster identities (potentially including -1's for "unclustered"), this function infers a forest structure on the clusters and returns paths through the forest that can be interpreted as lineages.

Usage

```
getLineages(data, clusterLabels, ...)

## S4 method for signature 'matrix,matrix'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'matrix,character'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'matrix,ANY'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'SlingshotDataSet,ANY'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'data.frame,ANY'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'matrix,numeric'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)

## S4 method for signature 'matrix,factor'
getLineages(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL)
```

Arguments

`data`	a data object containing the matrix of coordinates to be used for lineage inference. Supported types include `matrix`, `SingleCellExperiment`, and `SlingshotDataSet`.
`clusterLabels`	character, a vector of length `n` denoting cluster labels, optionally including `-1`'s for "unclustered." If `reducedDim` is a `SlingshotDataSet`, cluster labels will be taken from it.
`...`	Additional arguments to specify how lineages are constructed from clusters.
`reducedDim`	(optional) identifier to be used if `reducedDim(data)` contains multiple elements. Otherwise, the first element will be used by default.
`start.clus`	(optional) character, indicates the cluster(s) *from* which lineages will be drawn.
`end.clus`	(optional) character, indicates the cluster(s) which will be forced leaf nodes in their trees.
`dist.fun`	(optional) function, method for calculating distances between clusters. Must take two matrices as input, corresponding to points in reduced-dimensional space. If the minimum cluster size is larger than the number dimensions, the default is to use the joint covariance matrix to find squared distance between cluster centers. If not, the default is to use the diagonal of the joint covariance matrix.
`omega`	(optional) numeric, this granularity parameter determines the distance between every real cluster and the artificial cluster, OMEGA. It is parameterized such that this distance is `omega / 2`, making `omega` the maximum distance between two connected clusters. By default, `omega = Inf`.

Details

The connectivity matrix is learned by fitting a (possibly constrained) minimum-spanning tree on the clusters and the artificial cluster, OMEGA, which is a fixed distance away from every real cluster. This effectively limits the maximum branch length in the MST to twice the chosen distance, meaning that the output may contain multiple trees.

Once the connectivity is known, lineages are identified in any tree with at least two clusters. For a given tree, if there is an annotated starting cluster, every possible path out of a starting cluster and ending in a leaf that isn't another starting cluster will be returned. If no starting cluster is annotated, every leaf will be considered as a potential starting cluster and whichever configuration produces the longest average lineage length (in terms of number of clusters included) will be returned.

Value

An object of class `SlingshotDataSet` containing the arguments provided to `getLineages` as well as the following new elements:

- `lineages` a list of `L` items, where `L` is the number of lineages identified. Each lineage is represented by a character vector with the names of the clusters included in that lineage, in order.
- `connectivity` the inferred cluster connectivity matrix.
- `slingParams$start.given`,`slingParams$end.given` logical values indicating whether the starting and ending clusters were specified a priori.
- `slingParams$dist` the pairwise cluster distance matrix.

Examples

```
data("slingshotExample")
sds <- getLineages(rd, cl, start.clus = '1')

plot(rd, col = cl, asp = 1)
lines(sds, type = 'l', lwd = 3)
```

`newSlingshotDataSet` *Initialize an object of class* `SlingshotDataSet`

Description

Constructs a `SlingshotDataSet` object. Additional helper methods for manipulating `SlingshotDataSet` objects are also described below.

Usage

```
newSlingshotDataSet(reducedDim, clusterLabels, ...)

## S4 method for signature 'data.frame,ANY'
newSlingshotDataSet(reducedDim, clusterLabels,
  ...)

## S4 method for signature 'matrix,numeric'
```

```
newSlingshotDataSet(reducedDim, clusterLabels,
  ...)

## S4 method for signature 'matrix,factor'
newSlingshotDataSet(reducedDim, clusterLabels,
  ...)

## S4 method for signature 'matrix,ANY'
newSlingshotDataSet(reducedDim, clusterLabels, ...)

## S4 method for signature 'matrix,character'
newSlingshotDataSet(reducedDim, clusterLabels,
  ...)

## S4 method for signature 'matrix,matrix'
newSlingshotDataSet(reducedDim, clusterLabels,
  lineages = list(), adjacency = matrix(NA, 0, 0), curves = list(),
  slingParams = list())
```

Arguments

`reducedDim`	matrix. An `n` by `p` numeric matrix or data frame giving the coordinates of the cells in a reduced dimensionality space.
`clusterLabels`	character. A character vector of length `n` denoting each cell's cluster label.
`...`	additional components of a `SlingshotDataSet` to specify. This may include any of the following:
`lineages`	list. A list with each element a character vector of cluster names representing a lineage as an ordered set of clusters.
`adjacency`	matrix. A binary matrix describing the connectivity between clusters induced by the minimum spanning tree.
`curves`	list. A list of `principal.curve` objects produced by `getCurves`.
`slingParams`	list. Additional parameters used by Slingshot. These may specify how the minimum spanning tree on clusters was constructed: • `start.clus`character. The label of the root cluster. • `end.clus`character. Vector of cluster labels indicating the terminal clusters. • `start.given`logical. A logical value indicating whether the initial state was pre-specified. • `end.given`logical. A vector of logical values indicating whether each terminal state was pre-specified • `dist`matrix. A numeric matrix of pairwise cluster distances. They may also specify how simultaneous principal curves were constructed: • `shrink`logical or numeric between 0 and 1. Determines whether and how much to shrink branching lineages toward their shared average curve. • `extend`character. Specifies the method for handling root and leaf clusters of lineages when constructing the initial, piece-wise linear curve. Accepted values are 'y' (default), 'n', and 'pc1'. See `getCurves` for details. • `reweight`logical. Indicates whether to allow cells shared between lineages to be reweighted during curve-fitting. If `TRUE`, cells shared between lineages will be iteratively reweighted based on the quantiles of their projection distances to each curve.

- reassignlogical. Indicates whether to reassign cells to lineages at each iteration. If TRUE, cells will be added to a lineage when their projection distance to the curve is less than the median distance for all cells currently assigned to the lineage. Additionally, shared cells will be removed from a lineage if their projection distance to the curve is above the 90th percentile and their weight along the curve is less than 0.1.
- shrink.methodcharacter. Denotes how to determine the amount of shrinkage for a branching lineage. Accepted values are the same as for kernel in the density function (default is "cosine"), as well as "tricube" and "density". See getCurves for details.
- Other parameters specified by principal.curve.

Value

A SlingshotDataSet object with all specified values.

Methods (by class)

- reducedDim = data.frame,clusterLabels = ANY: returns a SlingshotDataSet object.
- reducedDim = matrix,clusterLabels = numeric: returns a SlingshotDataSet object.
- reducedDim = matrix,clusterLabels = factor: returns a SlingshotDataSet object.
- reducedDim = matrix,clusterLabels = ANY: returns a SlingshotDataSet object.
- reducedDim = matrix,clusterLabels = character: returns a SlingshotDataSet object.
- reducedDim = matrix,clusterLabels = matrix: returns a SlingshotDataSet object.

Examples

```
rd <- matrix(data=rnorm(100), ncol=2)
cl <- sample(letters[seq_len(5)], 50, replace = TRUE)
sds <- newSlingshotDataSet(rd, cl)
```

---

pairs-SlingshotDataSet
Pairs plot of Slingshot output

Description

A tool for quickly visualizing lineages inferred by slingshot.

Usage

```
## S3 method for class 'SlingshotDataSet'
pairs(x, type = NULL,
  show.constraints = FALSE, col = NULL, pch = 16, cex = 1,
  lwd = 2, ..., labels, horInd = seq_len(nc), verInd = seq_len(nc),
  lower.panel = FALSE, upper.panel = TRUE, diag.panel = NULL,
  text.panel = textPanel, label.pos = 0.5 + has.diag/3,
  line.main = 3, cex.labels = NULL, font.labels = 1,
  row1attop = TRUE, gap = 1)
```

Arguments

`x`	a `SlingshotDataSet` with results to be plotted.
`type`	character, the type of output to be plotted, can be one of `"lineages"`, `curves`, or `both` (by partial matching), see Details for more.
`show.constraints`	logical, whether or not the user-specified initial and terminal clusters should be specially denoted by green and red dots, respectively.
`col`	character, color vector for points.
`pch`	integer or character specifying the plotting symbol, see `par`.
`cex`	numeric, amount by which points should be magnified, see `par`.
`lwd`	numeric, the line width, see `par`.
`...`	additional parameters for `plot` or `axis`, see `pairs`.
`labels`	character, the names of the variables, see `pairs`.
`horInd`	see `pairs`.
`verInd`	see `pairs`.
`lower.panel`	see `pairs`.
`upper.panel`	see `pairs`.
`diag.panel`	see `pairs`.
`text.panel`	see `pairs`.
`label.pos`	see `pairs`.
`line.main`	see `pairs`.
`cex.labels`	see `pairs`.
`font.labels`	see `pairs`.
`row1attop`	see `pairs`.
`gap`	see `pairs`.

Details

If `type == 'lineages'`, straight line connectors between cluster centers will be plotted. If `type == 'curves'`, simultaneous princiapl curves will be plotted.

When `type` is not specified, the function will first check the `curves` slot and plot the curves, if present. Otherwise, `lineages` will be plotted, if present.

Value

returns `NULL`.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = "1")
pairs(sds, type = 'curves')
```

plot-SlingshotDataSet *Plot Slingshot output*

Description

Tools for visualizing lineages inferred by `slingshot`.

Usage

```
## S4 method for signature 'SlingshotDataSet,ANY'
plot(x, type = NULL,
  show.constraints = FALSE, add = FALSE, dims = seq_len(2),
  asp = 1, cex = 2, lwd = 2, col = 1, ...)

## S4 method for signature 'SlingshotDataSet'
lines(x, type = NULL, dims = seq_len(2),
  ...)
```

Arguments

`x`	a `SlingshotDataSet` with results to be plotted.
`type`	character, the type of output to be plotted, can be one of `"lineages"`, `"curves"`, or `"both"` (by partial matching), see Details for more.
`show.constraints`	logical, whether or not the user-specified initial and terminal clusters should be specially denoted by green and red dots, respectively.
`add`	logical, indicates whether the output should be added to an existing plot.
`dims`	numeric, which dimensions to plot (default is `1:2`).
`asp`	numeric, the y/x aspect ratio, see `plot.window`.
`cex`	numeric, amount by which points should be magnified, see `par`.
`lwd`	numeric, the line width, see `par`.
`col`	character or numeric, color(s) for lines, see `par`.
`...`	additional parameters to be passed to `lines`.

Details

If `type == 'lineages'`, straight line connectors between cluster centers will be plotted. If `type == 'curves'`, simultaneous principal curves will be plotted.

When `type` is not specified, the function will first check the `curves` slot and plot the curves, if present. Otherwise, `lineages` will be plotted, if present.

Value

returns `NULL`.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = "1")
plot(sds, type = 'b')

# add to existing plot
plot(rd, col = 'grey50')
lines(sds, lwd = 3)
```

---

plot3d-SlingshotDataSet
Plot Slingshot output in 3D

Description

Tools for visualizing lineages inferred by `slingshot`.

Usage

```
## S3 method for class 'SlingshotDataSet'
plot3d(x, type = NULL, add = FALSE,
  dims = seq_len(3), aspect = "iso", ...)
```

Arguments

`x`	a `SlingshotDataSet` with results to be plotted.
`type`	character, the type of output to be plotted, can be one of `"lineages"`, `curves`, or `both` (by partial matching), see Details for more.
`add`	logical, indicates whether the output should be added to an existing plot.
`dims`	numeric, which dimensions to plot (default is `1:3`).
`aspect`	either a logical indicating whether to adjust the aspect ratio or a new ratio, see `plot3d`.
`...`	additional parameters to be passed to `lines3d`.

Details

If `type == 'lineages'`, straight line connectors between cluster centers will be plotted. If `type == 'curves'`, simultaneous princiapl curves will be plotted.

When `type` is not specified, the function will first check the `curves` slot and plot the curves, if present. Otherwise, `lineages` will be plotted, if present.

Value

returns `NULL`.

Examples

```
## Not run:
library(rgl)
data("slingshotExample")
rd <- cbind(rd, rnorm(nrow(rd)))
sds <- slingshot(rd, cl, start.clus = "1")
plot3d(sds, type = 'b')

# add to existing plot
plot3d(rd, col = 'grey50', aspect = 'iso')
plot3d(sds, lwd = 3, add = TRUE)

## End(Not run)
```

`plotGenePseudotime` *Plot Gene Expression by Pseudotime*

Description

Show the gene expression pattern for an individual gene along lineages inferred by `slingshot`.

Usage

```
plotGenePseudotime(data, ...)

## S4 method for signature 'SlingshotDataSet'
plotGenePseudotime(data, gene, exprs,
  loess = TRUE, loessCI = TRUE, ...)

## S4 method for signature 'SingleCellExperiment'
plotGenePseudotime(data, gene, exprs,
  loess = TRUE, loessCI = TRUE, ...)
```

Arguments

`data`	an object containing `slingshot` output, either a `SlingshotDataSet` or a `SingleCellExperiment` object.
`...`	additional parameters to be passed to `plot`.
`gene`	the gene to be plotted. If `exprs` is provided, this may be either the gene name or its row index in `exprs`. Otherwise, this is assumed to be a vector of scaled expression values.
`exprs`	the genes-by-samples matrix of scaled expression values (log counts or normalized log counts).
`loess`	logical, whether to include a loess fit in each plot (default is `TRUE`).
`loessCI`	logical, whether to include a confidence band around the loess curve (default is `TRUE`).

Value

returns `NULL`.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = "1")
ex <- matrix(c(rchisq(100,1),rchisq(20,3),rchisq(20,6)),nrow=1)
rownames(ex) <- 'Gene-1'
plotGenePseudotime(sds, 'Gene-1', ex)
```

`predict,SlingshotDataSet-method`
Predict from a Slingshot model

Description

Map new observations onto simultaneous principal curves fitted by `slingshot`.

Usage

```
## S4 method for signature 'SlingshotDataSet'
predict(object, newdata = NULL)
```

Arguments

`object`	a `SlingshotDataSet` containing simultaneous principal curves to use for prediction.
`newdata`	a matrix or data frame of new points in the same reduced-dimensional space as the original input to `slingshot` (or `getLineages`).

Details

This function is a method for the generic function `predict` with `signature(object = "SlingshotDataSet")`. If no `newdata` argument is provided, it will return the original results, given by `object`.

Value

A `SlingshotDataSet` object based on the input `newdata`. New cells are treated as "unclustered" and the `lineages` and `adjacency` slots are intentionally left blank, to distinguish these results from the original `slingshot` output. The `curves` slot represents the projections of each new cell onto the existing curves. As with standard `slingshot` output, the lineage-specific pseudotimes and assignment weights can be accessed via the functions `slingPseudotime` and `slingCurveWeights`.

See Also

`slingshot`, `SlingshotDataSet`

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = '1')

x <- cbind(runif(100, min = -5, max = 10), runif(100, min = -4, max = 4))
predict(sds, x)
```

slingAdjacency *Extract Slingshot adjacency matrix*

Description

Extract the adjacency matrix from an object containing `slingshot` output.

Usage

```
slingAdjacency(x)

## S4 method for signature 'SlingshotDataSet'
slingAdjacency(x)

## S4 method for signature 'SingleCellExperiment'
slingAdjacency(x)
```

Arguments

`x`	an object containing `slingshot` output.

Value

the matrix of connections between clusters, inferred by the MST.

Methods (by class)

- `SlingshotDataSet`: returns the adjacency matrix between clusters from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the adjacency matrix between clusters from a `SingleCellExperiment` object.

Examples

```
data("slingshotExample")
sds <- getLineages(rd, cl)
slingAdjacency(sds)
```

slingCurves *Extract simultaneous principal curves*

Description

Extract the simultaneous principal curves from an object containing `slingshot` output.

Usage

```
slingCurves(x)

## S4 method for signature 'SlingshotDataSet'
slingCurves(x)

## S4 method for signature 'SingleCellExperiment'
slingCurves(x)
```

Arguments

`x`	an object containing `slingshot` output.

Value

the list of smooth lineage curves, each of which is a `principal.curve` object.

Methods (by class)

- `SlingshotDataSet`: returns the list of smooth lineage curves from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the list of smooth lineage curves from a `SingleCellExperiment` object.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl)
slingCurves(sds)
```

`slingLineages` *Extract the Slingshot lineages*

Description

Extract lineages (represented by ordered sets of clusters) from a `SlingshotDataSet`.

Usage

```
slingLineages(x)

## S4 method for signature 'SlingshotDataSet'
slingLineages(x)

## S4 method for signature 'SingleCellExperiment'
slingLineages(x)
```

Arguments

`x`	an object containing `slingshot` output.

Value

the list of lineages, represented by ordered sets of clusters.

Methods (by class)

- `SlingshotDataSet`: returns the list of lineages, represented by ordered sets of clusters from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the list of lineages, represented by ordered sets of clusters from a `SingleCellExperiment` object.

Examples

```
data("slingshotExample")
sds <- getLineages(rd, cl)
slingLineages(sds)
```

slingParams — *Methods for parameters used by Slingshot*

Description

Extracts additional control parameters used by Slingshot in lineage inference and fitting simultaneous principal curves.

Usage

```
slingParams(x)

## S4 method for signature 'SlingshotDataSet'
slingParams(x)

## S4 method for signature 'SingleCellExperiment'
slingParams(x)
```

Arguments

`x`	an object containing `slingshot` output.

Value

the list of additional parameters used by Slingshot.

Methods (by class)

- `SlingshotDataSet`: returns the list of additional parameters used by Slingshot from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the list of additional parameters used by Slingshot from a `SingleCellExperiment` object.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = '5')
slingParams(sds)
```

`slingPseudotime` *Get Slingshot pseudotime values*

Description

Extract the matrix of pseudotime values or cells' weights along each lineage.

Usage

```
slingPseudotime(x, ...)

slingCurveWeights(x)

## S4 method for signature 'SlingshotDataSet'
slingPseudotime(x, na = TRUE)

## S4 method for signature 'SingleCellExperiment'
slingPseudotime(x, na = TRUE)

## S4 method for signature 'SlingshotDataSet'
slingCurveWeights(x)

## S4 method for signature 'SingleCellExperiment'
slingCurveWeights(x)
```

Arguments

`x`	an object containing `slingshot` output.
`...`	additional parameters to be passed to object-specific methods.
`na`	logical. If `TRUE` (default), cells that are not assigned to a lineage will have a pseudotime value of `NA`. Otherwise, their arclength along each curve will be returned.

Value

an `n` by `L` matrix representing each cell's pseudotime along each lineage.

an `n` by `L` matrix of cell weights along each lineage.

Methods (by class)

- `SlingshotDataSet`: returns the matrix of pseudotime values from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the matrix of pseudotime values from a `SingleCellExperiment` object.

- `SlingshotDataSet`: returns the matrix of cell weights along each lineage from a `SlingshotDataSet` object.
- `SingleCellExperiment`: returns the matrix of cell weights along each lineage from a `SingleCellExperiment` object.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl)
slingPseudotime(sds)
data("slingshotExample")
sds <- slingshot(rd, cl)
slingCurveWeights(sds)
```

`slingshot` *Perform lineage inference with Slingshot*

Description

Perform lineage inference with Slingshot

Given a reduced-dimensional data matrix n by p and a vector of cluster labels (or matrix of soft cluster assignments, potentially including a -1 label for "unclustered"), this function performs lineage inference using a cluster-based minimum spanning tree and constructing simulatenous principal curves for branching paths through the tree.

This wrapper function performs lineage inference in two steps: (1) identify lineage structure with a cluster-based minimum spanning tree with the `getLineages` function and (2) construct smooth representations of each lineage using simultaneous principal curves from the function `getCurves`.

Usage

```
slingshot(data, clusterLabels, ...)

## S4 method for signature 'matrix,character'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'matrix,matrix'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'SlingshotDataSet,ANY'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'data.frame,ANY'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'matrix,numeric'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'matrix,factor'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'matrix,ANY'
slingshot(data, clusterLabels, reducedDim = NULL,
  start.clus = NULL, end.clus = NULL, dist.fun = NULL,
  omega = NULL, lineages = list(), shrink = TRUE, extend = "y",
  reweight = TRUE, reassign = TRUE, thresh = 0.001, maxit = 15,
  stretch = 2, smoother = "smooth.spline", shrink.method = "cosine",
  allow.breaks = TRUE, ...)

## S4 method for signature 'ClusterExperiment,ANY'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)

## S4 method for signature 'SingleCellExperiment,ANY'
slingshot(data, clusterLabels,
  reducedDim = NULL, start.clus = NULL, end.clus = NULL,
  dist.fun = NULL, omega = NULL, lineages = list(), shrink = TRUE,
  extend = "y", reweight = TRUE, reassign = TRUE, thresh = 0.001,
  maxit = 15, stretch = 2, smoother = "smooth.spline",
  shrink.method = "cosine", allow.breaks = TRUE, ...)
```

Arguments

`data`	a data object containing the matrix of coordinates to be used for lineage inference. Supported types include `matrix`, `SingleCellExperiment`, and `SlingshotDataSet`.
`clusterLabels`	character, a vector of length `n` denoting cluster labels, optionally including `-1`'s for "unclustered." If `reducedDim` is a `SlingshotDataSet`, cluster labels will be taken from it.
`...`	Additional parameters to pass to scatter plot smoothing function, `smoother`.
`reducedDim`	(optional) identifier to be used if `reducedDim(data)` contains multiple elements. Otherwise, the first element will be used by default.
`start.clus`	(optional) character, indicates the cluster(s) of origin. Lineages will be represented by paths coming out of this cluster.
`end.clus`	(optional) character, indicates the cluster(s) which will be forced leaf nodes. This introduces a constraint on the MST algorithm.
`dist.fun`	(optional) function, method for calculating distances between clusters. Must take two matrices as input, corresponding to subsets of `reducedDim`. If the minimum cluster size is larger than the number dimensions, the default is to use the joint covariance matrix to find squared distance between cluster centers. If not, the default is to use the diagonal of the joint covariance matrix.
`omega`	(optional) numeric, this granularity parameter determines the distance between every real cluster and the artificial cluster, OMEGA. It is parameterized such that this distance is `omega / 2`, making `omega` the maximum distance between two connected clusters. By default, `omega = Inf`.
`lineages`	list generated by `getLineages`, denotes lineages as ordered sets of clusters and contains the `K x K` connectivity matrix constructed on the clusters by `getLineages`.
`shrink`	logical or numeric between 0 and 1, determines whether and how much to shrink branching lineages toward their average prior to the split.
`extend`	character, how to handle root and leaf clusters of lineages when constructing the initial, piece-wise linear curve. Accepted values are `'y'` (default), `'n'`, and `'pc1'`. See 'Details' for more.
`reweight`	logical, whether to allow cells shared between lineages to be reweighted during curve-fitting. If `TRUE`, cells shared between lineages will be iteratively reweighted based on the quantiles of their projection distances to each curve. See 'Details' for more.
`reassign`	logical, whether to reassign cells to lineages at each iteration. If `TRUE`, cells will be added to a lineage when their projection distance to the curve is less than the median distance for all cells currently assigned to the lineage. Additionally, shared cells will be removed from a lineage if their projection distance to the curve is above the 90th percentile and their weight along the curve is less than `0.1`.
`thresh`	numeric, determines the convergence criterion. Percent change in the total distance from cells to their projections along curves must be less than `thresh`. Default is `0.001`, similar to `principal_curve`.
`maxit`	numeric, maximum number of iterations, see `principal_curve`.
`stretch`	numeric factor by which curves can be extrapolated beyond endpoints. Default is 2, see `principal_curve`.
`smoother,`	choice of scatter plot smoother. Same as `principal_curve`, but `"lowess"` option is replaced with `"loess"` for additional flexibility.

`shrink.method`	character denoting how to determine the appropriate amount of shrinkage for a branching lineage. Accepted values are the same as for `kernel` in `density` (default is `"cosine"`), as well as `"tricube"` and `"density"`. See 'Details' for more.
`allow.breaks`	logical, determines whether curves that branch very close to the origin should be allowed to have different starting points.

Details

The `connectivity` matrix is learned by fitting a (possibly constrained) minimum-spanning tree on the clusters and the artificial cluster, OMEGA, which is a fixed distance away from every real cluster. This effectively limits the maximum branch length in the MST to twice the chosen distance, meaning that the output may contain multiple trees.

Once the `connectivity` is known, lineages are identified in any tree with at least two clusters. For a given tree, if there is an annotated starting cluster, every possible path out of a starting cluster and ending in a leaf that isn't another starting cluster will be returned. If no starting cluster is annotated, every leaf will be considered as a potential starting cluster and whichever configuration produces the longest average lineage length (in terms of number of clusters included) will be returned.

When there is only a single lineage, the curve-fitting algorithm is nearly identical to that of `principal_curve`. When there are multiple lineages and `shrink == TRUE`, an additional step is added to the iterative procedure, forcing curves to be similar in the neighborhood of shared points (ie., before they branch).

The `extend` argument determines how to construct the piece-wise linear curve used to initiate the recursive algorithm. The initial curve is always based on the lines between cluster centers and if `extend = 'n'`, this curve will terminate at the center of the endpoint clusters. Setting `extend = 'y'` will allow the first and last segments to extend beyond the cluster center to the orthogonal projection of the furthest point. Setting `extend = 'pc1'` is similar to `'y'`, but uses the first principal component of the cluster to determine the direction of the curve beyond the cluster center. These options typically have little to no impact on the final curve, but can occasionally help with stability issues.

When `shink == TRUE`, we compute a shrinkage curve, $w_l(t)$, for each lineage, a non-increasing function of pseudotime that determines how much that lineage should be shrunk toward a shared average curve. We set $w_l(0) = 1$, so that the curves will perfectly overlap the average curve at pseudotime `0`. The weighting curve decreases from `1` to `0` over the non-outlying pseudotime values of shared cells (where outliers are defined by the `1.5*IQR` rule). The exact shape of the curve in this region is controlled by `shrink.method`, and can follow the shape of any standard kernel function's cumulative density curve (or more precisely, survival curve, since we require a decreasing function). Different choices of `shrink.method` seem to have little impact on the final curves, in most cases.

When `reweight = TRUE`, weights for shared cells are based on the quantiles of their projection distances onto each curve. The distances are ranked and converted into quantiles between `0` and `1`, which are then transformed by `1 -q^2`. Each cell's weight along a given lineage is the ratio of this value to the maximum value for this cell across all lineages.

Value

An object of class `SlingshotDataSet` containing the arguments provided to `slingshot` as well as the following output:

- `lineages` a list of `L` items, where `L` is the number of lineages identified. Each lineage is represented by a character vector with the names of the clusters included in that lineage, in order.

- `connectivity` the inferred cluster connectivity matrix.
- `slingParams`Additional parameters used for lineage inference or fitting simultaneous principal curves. This may include the elements `start.given` and `end.given`, logical values indicating whether the starting and ending clusters were specified a priori. Additionally, this will always include `dist`, the pairwise cluster distance matrix.
- curvesA list of `principal_curve` objects.

References

Hastie, T., and Stuetzle, W. (1989). "Principal Curves." *Journal of the American Statistical Association*, 84:502–516.

Examples

```
data("slingshotExample")
sds <- slingshot(rd, cl, start.clus = '1')

plot(rd, col = cl, asp = 1)
lines(sds, lwd = 3)
```

`SlingshotDataSet` *Extract Slingshot output*

Description

This is a convenience function to extract a `SlingshotDataSet` from an object containing `slingshot` output.

Usage

```
SlingshotDataSet(data, ...)

## S4 method for signature 'SingleCellExperiment'
SlingshotDataSet(data)

## S4 method for signature 'SlingshotDataSet'
SlingshotDataSet(data)
```

Arguments

`data`	an object containing `slingshot` output.
`...`	additional arguments to pass to object-specific methods.

Value

A `SlingshotDataSet` object containing the output of `slingshot`.

SlingshotDataSet-class *Class* SlingshotDataSet

Description

The SlingshotDataSet class holds data relevant for performing lineage inference with the slingshot package, primarily a reduced dimensional representation of the data and a set of cluster labels. All slingshot methods can take an object of the class SlingshotDataSet as input and will output the same.

Extract cluster labels, either a character vector or matrix of weights.

Usage

```
## S4 method for signature 'SlingshotDataSet'
show(object)

## S4 method for signature 'SlingshotDataSet'
reducedDim(x)

## S4 method for signature 'SlingshotDataSet'
reducedDims(x)

## S4 method for signature 'SlingshotDataSet'
clusterLabels(x)

## S4 method for signature 'SlingshotDataSet,ANY,ANY,ANY'
x[i, j]
```

Arguments

object	a SlingshotDataSet object.
x	a SlingshotDataSet object.
i	indices to be applied to rows (cells) of the reduced dimensional matrix and cluster labels.
j	indices to be applied to the columns (dimensions) of the reduced dimensional matrix.

Details

Warning: this will remove any existing lineages or curves from the SlingshotDataSet object.

Value

The accessor functions reducedDim, clusterLabels, lineages, adjacency, curves, and slingParams return the corresponding elements of a SlingshotDataSet. The functions pseudotime and curveWeights extract useful output elements of a SlingshotDataSet, provided that curves have already been fit with either slingshot or getCurves.

A matrix of cluster weights for each cell or a vector of cluster assignments.

Methods (by generic)

- show: a short summary of SlingshotDataSet object.
- reducedDim: returns the matrix representing the reduced dimensional dataset.
- reducedDims: returns the matrix representing the reduced dimensional dataset.
- clusterLabels: extracts cluster labels from a SlingshotDataSet object.
- [: Subset dataset and cluster labels.

Slots

reducedDim matrix. An n by p numeric matrix or data frame giving the coordinates of the cells in a reduced dimensionality space.

clusterLabels matrix or character. An n by K matrix of weights indicating each cell's cluster assignment or a character vector of cluster assignments, which will be converted into a binary matrix.

lineages list. A list with each element a character vector of cluster names representing a lineage as an ordered set of clusters.

adjacency matrix. A binary matrix describing the adjacency between clusters induced by the minimum spanning tree.

curves list. A list of principal_curve objects produced by getCurves.

slingParams list. Additional parameters used by Slingshot. These may specify how the minimum spanning tree on clusters was constructed:

- start.cluscharacter. The label of the root cluster.
- end.cluscharacter. Vector of cluster labels indicating the terminal clusters.
- start.givenlogical. A logical value indicating whether the initial state was pre-specified.
- end.givenlogical. A vector of logical values indicating whether each terminal state was pre-specified
- distmatrix. A numeric matrix of pairwise cluster distances.

They may also specify how simultaneous principal curves were constructed:

- shrinklogical or numeric between 0 and 1. Determines whether and how much to shrink branching lineages toward their shared average curve.
- extendcharacter. Specifies the method for handling root and leaf clusters of lineages when constructing the initial, piece-wise linear curve. Accepted values are 'y' (default), 'n', and 'pc1'. See getCurves for details.
- reweightlogical. Indicates whether to allow cells shared between lineages to be reweighted during curve-fitting. If TRUE, cells shared between lineages will be iteratively reweighted based on the quantiles of their projection distances to each curve.
- reassignlogical. Indicates whether to reassign cells to lineages at each iteration. If TRUE, cells will be added to a lineage when their projection distance to the curve is less than the median distance for all cells currently assigned to the lineage. Additionally, shared cells will be removed from a lineage if their projection distance to the curve is above the 90th percentile and their weight along the curve is less than 0.1.
- shrink.methodcharacter. Denotes how to determine the amount of shrinkage for a branching lineage. Accepted values are the same as for kernel in the density function (default is "cosine"), as well as "tricube" and "density". See getCurves for details.
- Other parameters specified by principal_curve.

Examples

```
rd <- matrix(data=rnorm(100), ncol=2)
cl <- sample(letters[seq_len(5)], 50, replace = TRUE)
sds <- newSlingshotDataSet(rd, cl)
clusterLabels(sds)
```

`slingshotExample` *Bifurcating lineages data*

Description

A simulated low-dimensional representation of two bifurcating lineages (`rd`) and a vector of cluster labels generated by $k$-means with $K = 5$ (`cl`).

Usage

```
rd

cl
```

Format

A matrix of coordinates in two dimensions, representing $140$ cells, and a numeric vector $140$ corresponding cluster labels.

Source

Simulated data provided with the `slingshot` package.

Examples

```
data("slingshotExample")
slingshot(rd, cl)
```

Index