Package 'slingshot'

December 24, 2024

Title Tools for ordering single-cell sequencing

Version 2.15.0

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 (>= 4.0), princurve (>= 2.0.4), stats, TrajectoryUtils

Imports graphics, grDevices, igraph, matrixStats, methods, S4Vectors, SingleCellExperiment, SummarizedExperiment

Suggests BiocGenerics, BiocStyle, clusterExperiment, DelayedMatrixStats, knitr, mclust, mgcv, RColorBrewer, rgl, rmarkdown, testthat, uwot, covr

VignetteBuilder knitr

LazyData false

RoxygenNote 7.2.0

Encoding UTF-8

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 devel

git_last_commit cf41eec

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-23

Author Kelly Street [aut, cre, cph],
Davide Risso [aut],
Diya Das [aut],
Sandrine Dudoit [ths],
Koen Van den Berge [ctb],
Robrecht Cannoodt [ctb] (ORCID:
<https://orcid.org/0000-0003-3641-729X>, github: rcannood)

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

Contents

`as.PseudotimeOrdering` *Conversion to PseudotimeOrdering*

Description

This function converts objects that contain `slingshot` results into a `PseudotimeOrdering`.

Usage

```
as.PseudotimeOrdering(x, ...)

## S4 method for signature 'SlingshotDataSet'
as.PseudotimeOrdering(x)

## S4 method for signature 'SingleCellExperiment'
as.PseudotimeOrdering(x)

## S4 method for signature 'PseudotimeOrdering'
as.PseudotimeOrdering(x)
```

Arguments

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

Value

A PseudotimeOrdering object containing the slingshot results from the original object, x.

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
library(SingleCellExperiment)
u <- matrix(rpois(140*50, 5), nrow = 50)
sce <- SingleCellExperiment(assays = list(counts = u),
                            reducedDims = SimpleList(PCA = rd),
                            colData = data.frame(clus = cl))
sce <- slingshot(sce, clusterLabels = 'clus', reducedDim = 'PCA')
as.PseudotimeOrdering(sce)
```

---

as.SlingshotDataSet *Conversion to SlingshotDataSet*

Description

This function converts objects that contain slingshot results into a SlingshotDataSet.

Usage

```
as.SlingshotDataSet(x, ...)

## S4 method for signature 'PseudotimeOrdering'
as.SlingshotDataSet(x)
```

```
## S4 method for signature 'SingleCellExperiment'
as.SlingshotDataSet(x)

## S4 method for signature 'SlingshotDataSet'
as.SlingshotDataSet(x)
```

Arguments

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

Value

A SlingshotDataSet object containing the slingshot results from the original object, x.

See Also

PseudotimeOrdering

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl, start.clus = '1')
as.SlingshotDataSet(pto)
```

---

embedCurves *Embed trajectory in new space*

Description

This function takes the output of slingshot (or getCurves) and attempts to embed the curves in a different coordinate space than the one in which they were constructed. This should be considered a visualization tool, only.

Usage

```
embedCurves(x, newDimRed, ...)

## S4 method for signature 'PseudotimeOrdering,matrix'
embedCurves(
  x,
  newDimRed,
  shrink = NULL,
  stretch = NULL,
```

```
  approx_points = NULL,
  smoother = NULL,
  shrink.method = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment,matrix'
embedCurves(
  x,
  newDimRed,
  shrink = NULL,
  stretch = NULL,
  approx_points = NULL,
  smoother = NULL,
  shrink.method = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment,character'
embedCurves(
  x,
  newDimRed,
  shrink = NULL,
  stretch = NULL,
  approx_points = NULL,
  smoother = NULL,
  shrink.method = NULL,
  ...
)
```

Arguments

x	an object containing slingshot output.
newDimRed	a matrix representing the new coordinate space in which to embed the curves.
...	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.
stretch	numeric factor by which curves can be extrapolated beyond endpoints. Default is 2, see principal_curve.
approx_points	numeric, whether curves should be approximated by a fixed number of points. If FALSE (or 0), no approximation will be performed and curves will contain as many points as the input data. If numeric, curves will be approximated by this number of points; preferably about 100 (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.

Details

Many of the same parameters are used here as in getCurves. This function attempts to translate curves from one reduced dimensional space to another by predicting each dimension as a function of pseudotime (ie. the new curve is determined by a series of scatterplot smoothers predicting the coordinates in the new space as a function of pseudotime). Because the pseudotime values are not changed, this amounts to a single iteration of the iterative curve-fitting process used by getCurves.

Note that non-linear dimensionality reduction techniques (such as tSNE and UMAP) may produce discontinuities not observed in other spaces. Use caution when embedding curves in these spaces.

Value

a PseudotimeOrdering object containing curves in the new space.

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl, start.clus = '1')
rd2 <- cbind(rd[,2] + rnorm(nrow(rd)), -rd[,1] + rnorm(nrow(rd)))
pto.new <- embedCurves(pto, rd2)
pto.new

plot(rd2, col = cl, asp = 1)
lines(SlingshotDataSet(pto.new), lwd = 3)
```

---

getCurves *Construct Simultaneous Principal Curves*

Description

This function constructs simultaneous principal curves, the second step in Slingshot's trajectory inference procedure. It takes a (specifically formatted) PseudotimeOrdering object, as is returned by the first step, getLineages. The output is another PseudotimeOrdering object, containing the simultaneous principal curves, pseudotime estimates, and lineage assignment weights.

Usage

```
getCurves(data, ...)

## S4 method for signature 'PseudotimeOrdering'
getCurves(
  data,
```

```
  shrink = TRUE,
  extend = "y",
  reweight = TRUE,
  reassign = TRUE,
  thresh = 0.001,
  maxit = 15,
  stretch = 2,
  approx_points = NULL,
  smoother = "smooth.spline",
  shrink.method = "cosine",
  allow.breaks = TRUE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
getCurves(data, ...)

## S4 method for signature 'SlingshotDataSet'
getCurves(data, ...)
```

Arguments

data	a data object containing lineage information provided by getLineages, to be used for constructing simultaneous principal curves. Supported types include SingleCellExperiment, SlingshotDataSet, and PseudotimeOrdering (recommended).
...	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 (default = TRUE).
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 (default), 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 (default), 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 (default = 15), see principal_curve.

stretch	numeric factor by which curves can be extrapolated beyond endpoints. Default is 2, see principal_curve.
approx_points	numeric, whether curves should be approximated by a fixed number of points. If FALSE (or 0), no approximation will be performed and curves will contain as many points as the input data. If numeric, curves will be approximated by this number of points (default = 150 or #cells, whichever is smaller). See 'Details' and principal_curve for more.
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

This function constructs simultaneous principal curves (one per lineage). Cells are mapped to curves by orthogonal projection and pseudotime is estimated by the arclength along the curve (also called lambda, in the principal_curve objects).

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 approx_points argument, which sets the number of points to be used for each curve, can have a large effect on computation time. Due to this consideration, we set the default value to 150 whenever the input dataset contains more than that many cells. This setting should help with exploratory analysis while having little to no impact on the final curves. To disable this behavior and construct curves with the maximum number of points, set approx_points = FALSE.

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 limited impact on the final curve, but can occasionally help with stability issues.

When shink = TRUE, we compute a percent 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 $w_l(0) = 1$ (complete shrinkage), so that the curves will always 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 to have no discernable 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 PseudotimeOrdering object containing the pseudotime estimates and lineage assignment weights in the assays. It will also include the original information provided by getLineages, as well as the following new elements in the metadata:

- curves A list of principal_curve objects.
- slingParams Additional 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")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- getLineages(rd, cl, start.clus = '1')
pto <- getCurves(pto)

# plotting
sds <- as.SlingshotDataSet(pto)
plot(rd, col = cl, asp = 1)
lines(sds, type = 'c', lwd = 3)
```

---

getLineages *Infer Lineage Structure from Clustered Samples*

Description

This function constructs the minimum spanning tree(s) on clusters of cells, the first step in Slingshot's trajectory inference procedure. Paths through the MST from an origin cluster to leaf node clusters are 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.method = "slingshot",
  use.median = FALSE,
  omega = FALSE,
  omega_scale = 1.5,
  times = NULL,
  ...
)

## S4 method for signature 'matrix,character'
getLineages(data, clusterLabels, ...)

## S4 method for signature 'matrix,ANY'
getLineages(data, clusterLabels, ...)

## S4 method for signature 'SlingshotDataSet,ANY'
getLineages(data, clusterLabels, ...)

## S4 method for signature 'PseudotimeOrdering,ANY'
getLineages(data, clusterLabels, ...)

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

## S4 method for signature 'matrix,numeric'
getLineages(data, clusterLabels, ...)

## S4 method for signature 'matrix,factor'
getLineages(data, clusterLabels, ...)

## S4 method for signature 'SingleCellExperiment,ANY'
getLineages(data, clusterLabels, reducedDim = NULL, ...)
```

Arguments

data	a data object containing the matrix of coordinates to be used for lineage inference. Supported types include matrix, SingleCellExperiment, SlingshotDataSet, and PseudotimeOrdering.
clusterLabels	each cell's cluster assignment. This can be a single vector of labels, or a #cells

by #clusters matrix representing weighted cluster assignment. Either representation may optionally include a "-1" group meaning "unclustered."

`...` Additional arguments to specify how lineages are constructed from clusters.

`reducedDim` (optional) the dimensionality reduction to be used. Can be a matrix or a character identifying which element of `reducedDim(data)` is to be used. If multiple dimensionality reductions are present and this argument is not provided, the first element will be used by default.

`start.clus` (optional) character, indicates the starting cluster(s) from which lineages will be drawn.

`end.clus` (optional) character, indicates which cluster(s) will be forced to be leaf nodes in the graph.

`dist.method` (optional) character, specifies the method for calculating distances between clusters. Default is "slingshot", see `createClusterMST` for details.

`use.median` logical, whether to use the median (instead of mean) when calculating cluster centroid coordinates.

`omega` (optional) numeric or logical, this granularity parameter determines the distance between every real cluster and the artificial cluster, `.OMEGA`. In practice, this makes `omega` the maximum allowable distance between two connected clusters. By default, `omega = Inf`. If `omega = TRUE`, the maximum edge length will be set to the median edge length of the unsupervised MST times a scaling factor (`omega_scale`, default = `1.5`). This value is provided as a potentially useful rule of thumb for datasets with outlying clusters or multiple, distinct trajectories. See `outgroup` in `createClusterMST`.

`omega_scale` (optional) numeric, scaling factor to use when `omega = TRUE`. The maximum edge length will be set to the median edge length of the unsupervised MST times `omega_scale` (default = `3`). See `outscale` in `createClusterMST`.

`times` numeric, vector of external times associated with either clusters or cells. See `defineMSTPaths` for details.

Details

Given a reduced-dimension data matrix `n` by `p` and a set of cluster identities (potentially including a "-1" group for "unclustered"), this function infers a tree (or forest) structure on the clusters. This work is now mostly handled by the more general function, `createClusterMST`.

The graph of this structure is learned by fitting a (possibly constrained) minimum-spanning tree on the clusters, plus 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 the chosen distance, meaning that the output may contain multiple trees.

Once the graph 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, one will be chosen by a heuristic method, but this is not recommended.

Value

An object of class PseudotimeOrdering. Although the final pseudotimes have not yet been calculated, the assay slot of this object contains two elements: pseudotime, a matrix of NA values; and weights, a preliminary matrix of lineage assignment weights. The reducedDim and clusterLabels matrices will be stored in the cellData. Additionally, the metadata slot will contain an igraph object named mst, a list of parameter values named slingParams, and a list of lineages (ordered sets of clusters) named lineages.

Examples

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

# plotting
sds <- as.SlingshotDataSet(pto)
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. We now recommend using PseudotimeOrdering objects, instead.

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.

- reassign logical. 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.method character. 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.

See Also

PseudotimeOrdering

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 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")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl, start.clus = "1")
pairs(SlingshotDataSet(pto))
```

---

plot-SlingshotDataSet *Plot Slingshot output*

Description

Tools for visualizing lineages inferred by `slingshot`.

Usage

```
## S3 method for class 'SlingshotDataSet'
plot(
  x,
  type = NULL,
  linInd = NULL,
  show.constraints = FALSE,
  add = FALSE,
  dims = seq_len(2),
  asp = 1,
  cex = 2,
  lwd = 2,
  col = 1,
  ...
)

## S3 method for class '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.
linInd	integer, an index indicating which lineages should be plotted (default is to plot all lineages). If col is a vector, it will be subsetted by linInd.
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")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl, start.clus = "1")
plot(SlingshotDataSet(pto), type = 'b')

# add to existing plot
sds <- as.SlingshotDataSet(pto)
plot(rd, col = 'grey50', asp = 1)
lines(sds, lwd = 3)
```

plot3d-SlingshotDataSet

Plot Slingshot output in 3D

Description

Tools for visualizing lineages inferred by `slingshot`.

Usage

```
plot3d.SlingshotDataSet(
  x,
  type = NULL,
  linInd = NULL,
  add = FALSE,
  dims = seq_len(3),
  aspect = "iso",
  size = 10,
  col = 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.
`linInd`	integer, an index indicating which lineages should be plotted (default is to plot all lineages). If `col` is a vector, it will be subsetted by `linInd`.
`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`.
`size`	numeric, size of points for MST (default is `10`), see `plot3d`.
`col`	character or numeric, color(s) for lines, see `par`.
`...`	additional parameters to be passed to `lines3d`.

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

```
library(rgl)
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
rd <- cbind(rd, rnorm(nrow(rd)))
pto <- slingshot(rd, cl, start.clus = "1")
sds <- SlingshotDataSet(pto)
plot3d.SlingshotDataSet(sds, type = 'b')

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

---

predict,PseudotimeOrdering-method

Predict from a Slingshot model

Description

Map new observations onto simultaneous principal curves fitted by slingshot.

Usage

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

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

Arguments

object	a PseudotimeOrdering or 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 inputs being either a PseudotimeOrdering or SlingshotDataSet. If no newdata argument is provided, it will return the original results, given by object.

Value

An object of the same type as `object`, based on the input `newdata`. New cells are treated as "unclustered", but other metadata is preserved. 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`

Examples

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

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

---

slingBranchGraph *Construct graph of slingshot branch labels*

Description

Builds a graph describing the relationships between the different branch assignments.

Usage

```
slingBranchGraph(x, ...)

## S4 method for signature 'ANY'
slingBranchGraph(x, thresh = NULL, max_node_size = 100)
```

Arguments

`x`	an object containing `slingshot` output, generally either a `PseudotimeOrdering` or `SingleCellExperiment`.
`...`	additional arguments passed to object-specific methods.
`thresh`	weight threshold for assigning cells to lineages. A cell's weight on a certain lineage must be greater than this value (default = `1/L`, for `L` lineages).
`max_node_size`	the `size` of the largest node in the graph, for plotting (all others will be drawn proportionally). Default is `100`. See `igraph.plotting` for more details.

Value

an `igraph` object representing the relationships between lineages.

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl)
slingBranchGraph(pto)
```

---

slingBranchID *Get slingshot branch labels*

Description

Summarizes the lineage assignment weights from slingshot results as a single vector. This is represented by a categorical variable indicating which lineage (or combination of lineages) each cell is assigned to.

Usage

```
slingBranchID(x, ...)

## S4 method for signature 'ANY'
slingBranchID(x, thresh = NULL)
```

Arguments

x	an object containing slingshot output, generally either a PseudotimeOrdering or SingleCellExperiment.
...	additional arguments passed to object-specific methods.
thresh	weight threshold for assigning cells to lineages. A cell's weight on a certain lineage must be at least this value (default = 1/L, for L lineages).

Value

a factor variable that assigns each cell to a particular lineage or set of lineages.

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl)
slingBranchID(pto)
```

slingClusterLabels *Extract cluster labels used by Slingshot*

Description

Extract the cluster labels used by slingshot.

Usage

```
slingClusterLabels(x)

## S4 method for signature 'PseudotimeOrdering'
slingClusterLabels(x)

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

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

Arguments

x	an object containing slingshot output.

Value

Typically returns a matrix of cluster assignment weights (#cells by #clusters). Rarely, a vector of cluster labels.

Examples

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

slingCurves *Extract simultaneous principal curves*

Description

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

Usage

```
slingCurves(x, ...)

## S4 method for signature 'PseudotimeOrdering'
slingCurves(x, as.df = FALSE)

## S4 method for signature 'SingleCellExperiment'
slingCurves(x, ...)

## S4 method for signature 'SlingshotDataSet'
slingCurves(x, as.df = FALSE)
```

Arguments

x	an object containing slingshot output.
...	additional parameters to be passed to object-specific methods.
as.df	logical, whether to format the output as a data.frame, suitable for plotting with ggplot.

Value

A list of smooth lineage curves, each of which is a principal_curve object.

Examples

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

---

slingLineages *Extract the Slingshot lineages*

Description

Extract lineages (represented by ordered sets of clusters) identified by slingshot.

Usage

```
slingLineages(x)

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

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

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

Arguments

x	an object containing slingshot output.

Value

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

Examples

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

---

slingMST *Extract Slingshot minimum spanning tree*

Description

Extract the minimum spanning tree from an object containing slingshot output.

Usage

```
slingMST(x, ...)

## S4 method for signature 'PseudotimeOrdering'
slingMST(x, as.df = FALSE)

## S4 method for signature 'SingleCellExperiment'
slingMST(x, ...)

## S4 method for signature 'SlingshotDataSet'
slingMST(x, as.df = FALSE)
```

Arguments

x	an object containing slingshot output.
...	additional parameters to be passed to object-specific methods.
as.df	logical, whether to format the output as a data.frame, suitable for plotting with ggplot.

Value

In most cases, output is an igraph object representing the MST. If x is a SlingshotDataSet, then output is an adjacency matrix representing the MST.

Examples

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

---

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 'PseudotimeOrdering'
slingParams(x)

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

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

Arguments

x	an object containing slingshot output.

Value

The list of additional parameters used by Slingshot. These include parameters related to the cluster-based minimum spanning tree:

- start.clus character. The label of the root cluster, or a vector of cluster labels giving the root clusters of each disjoint component of the graph.
- end.clus character. Vector of cluster labels indicating 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
- `omega` numeric or logical. Granularity parameter determining the maximum edge length for building the MST. See `getLineages`.
- `omega_scale` numeric. Scaling factor used for setting maximum edge length when `omega = TRUE`. See `getLineages`.

They may also specify how simultaneous principal curves were constructed (for a complete listing, see `getCurves`:

- `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.
- `reassign` logical. 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.method` character. 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.
- approx_points numeric. Number of points to use in estimating curves. See `getCurves` for details.
- allow.breaks logical. Whether to allow curves that diverge very early on in a trajectory to have different starting points.
- Other parameters specified by `principal_curve`.

Examples

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

slingPseudotime *Get Slingshot pseudotime values*

Description

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

Usage

```
slingPseudotime(x, ...)

slingCurveWeights(x, ...)

slingAvgPseudotime(x, ...)

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

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

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

## S4 method for signature 'PseudotimeOrdering'
slingCurveWeights(x, as.probs = FALSE)

## S4 method for signature 'SingleCellExperiment'
slingCurveWeights(x, as.probs = FALSE)

## S4 method for signature 'SlingshotDataSet'
slingCurveWeights(x, as.probs = FALSE)

## S4 method for signature 'ANY'
slingAvgPseudotime(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.
as.probs	logical. If FALSE (default), output will be the weights used to construct the curves, appropriate for downstream analysis of individual lineages (ie. a cell shared between two lineages can have two weights of 1). If TRUE, output will be

scaled to represent probabilistic assignment of cells to lineages (ie. a cell shared between two lineages will have two weights of `0.5`).

Value

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

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

`slingAvgPseudotime`: a length `n` vector of average cell pseudotimes, where the average is a weighted average across lineages, weighted by the assignment weights.

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
pto <- slingshot(rd, cl, start.clus = '1')
slingPseudotime(pto)
slingCurveWeights(pto)
slingAvgPseudotime(pto)
```

---

slingReducedDim *Extract dimensionality reduction used by Slingshot*

Description

Extract the dimensionality reduction used by `slingshot`.

Usage

```
slingReducedDim(x)

## S4 method for signature 'PseudotimeOrdering'
slingReducedDim(x)

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

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

Arguments

`x`	an object containing `slingshot` output.

Value

A matrix of coordinates.

Examples

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

`slingshot` *Perform trajectory inference with Slingshot*

Description

Perform trajectory inference with Slingshot

Perform trajectory inference by (1) identifying lineage structure with a cluster-based minimum spanning tree, and (2) constructing smooth representations of each lineage using simultaneous principal curves. This function wraps the `getLineages` and `getCurves` functions and is the primary function of the `slingshot` package.

Usage

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

## S4 method for signature 'matrix,character'
slingshot(
  data,
  clusterLabels,
  reducedDim = NULL,
  start.clus = NULL,
  end.clus = NULL,
  dist.method = "slingshot",
  use.median = FALSE,
  omega = FALSE,
  omega_scale = 1.5,
  times = NULL,
  shrink = TRUE,
  extend = "y",
  reweight = TRUE,
  reassign = TRUE,
  thresh = 0.001,
  maxit = 15,
  stretch = 2,
  approx_points = NULL,
  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.method = "slingshot",
  use.median = FALSE,
  omega = FALSE,
  omega_scale = 1.5,
  times = NULL,
  shrink = TRUE,
  extend = "y",
  reweight = TRUE,
  reassign = TRUE,
  thresh = 0.001,
  maxit = 15,
  stretch = 2,
  approx_points = NULL,
  smoother = "smooth.spline",
  shrink.method = "cosine",
  allow.breaks = TRUE,
  ...
)

## S4 method for signature 'SlingshotDataSet,ANY'
slingshot(data, clusterLabels, ...)

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

## S4 method for signature 'matrix,numeric'
slingshot(data, clusterLabels, ...)

## S4 method for signature 'matrix,factor'
slingshot(data, clusterLabels, ...)

## S4 method for signature 'matrix,ANY'
slingshot(data, clusterLabels, ...)

## S4 method for signature 'ClusterExperiment,ANY'
slingshot(
  data,
  clusterLabels,
  reducedDim = NULL,
  start.clus = NULL,
```

```
  end.clus = NULL,
  dist.method = "slingshot",
  use.median = FALSE,
  omega = FALSE,
  omega_scale = 1.5,
  times = NULL,
  shrink = TRUE,
  extend = "y",
  reweight = TRUE,
  reassign = TRUE,
  thresh = 0.001,
  maxit = 15,
  stretch = 2,
  approx_points = NULL,
  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.method = "slingshot",
  use.median = FALSE,
  omega = FALSE,
  omega_scale = 1.5,
  times = NULL,
  shrink = TRUE,
  extend = "y",
  reweight = TRUE,
  reassign = TRUE,
  thresh = 0.001,
  maxit = 15,
  stretch = 2,
  approx_points = NULL,
  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, SlingshotDataSet, and PseudotimeOrdering.
clusterLabels	each cell's cluster assignment. This can be a single vector of labels, or a #cells by #clusters matrix representing weighted cluster assignment. Either representation may optionally include a "-1" group meaning "unclustered."
...	Additional parameters to pass to scatter plot smoothing function, smoother.
reducedDim	(optional) the dimensionality reduction to be used. Can be a matrix or a character identifying which element of reducedDim(data) is to be used. If multiple dimensionality reductions are present and this argument is not provided, the first element will be used by default.
start.clus	(optional) character, indicates the starting cluster(s) from which lineages will be drawn.
end.clus	(optional) character, indicates which cluster(s) will be forced to be leaf nodes in the graph.
dist.method	(optional) character, specifies the method for calculating distances between clusters. Default is "slingshot", see createClusterMST for details.
use.median	logical, whether to use the median (instead of mean) when calculating cluster centroid coordinates.
omega	(optional) numeric, this granularity parameter determines the distance between every real cluster and the artificial cluster, .OMEGA. In practice, this makes omega the maximum allowable distance between two connected clusters. By default, omega = Inf. If omega = TRUE, the maximum edge length will be set to the median edge length of the unsupervised MST times a scaling factor (omega_scale, default = 3). This value is provided as a potentially useful rule of thumb for datasets with outlying clusters or multiple, distinct trajectories. See outgroup in createClusterMST.
omega_scale	(optional) numeric, scaling factor to use when omega = TRUE. The maximum edge length will be set to the median edge length of the unsupervised MST times omega_scale (default = 1.5). See outscale in createClusterMST.
times	numeric, vector of external times associated with either clusters or cells. See defineMSTPaths for details.
shrink	logical or numeric between 0 and 1, determines whether and how much to shrink branching lineages toward their average prior to the split (default = TRUE).
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 (default), 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 (default), 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 (default = 15), see `principal_curve`.

`stretch` numeric factor by which curves can be extrapolated beyond endpoints. Default is 2, see `principal_curve`.

`approx_points` numeric, whether curves should be approximated by a fixed number of points. If `FALSE` (or 0), no approximation will be performed and curves will contain as many points as the input data. If numeric, curves will be approximated by this number of points (default = `150` or `#cells`, whichever is smaller). See 'Details' and `principal_curve` for more.

`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

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 trajectory inference using a cluster-based minimum spanning tree on the clusters and simultaneous principal curves for smooth, branching paths.

The graph of this structure is learned by fitting a (possibly constrained) minimum-spanning tree on the clusters, plus 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 the chosen distance, meaning that the output may contain multiple trees.

Once the graph 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, one will be chosen by a heuristic method, but this is not recommended.

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 `approx_points` argument, which sets the number of points to be used for each curve, can have a large effect on computation time. Due to this consideration, we set the default value to `150` whenever the input dataset contains more than that many cells. This setting should help with exploratory analysis while having little to no impact on the final curves. To disable this behavior and construct curves with the maximum number of points, set `approx_points = FALSE`.

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 limited impact on the final curve, but can occasionally help with stability issues.

When shink = TRUE, we compute a percent 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 $w_l(0) = 1$ (complete shrinkage), so that the curves will always 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 to have no discernable 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 PseudotimeOrdering containing the pseudotime estimates and lineage assignment weights in the assays. The reducedDim and clusterLabels matrices will be stored in the cellData. Additionally, the metadata slot will contain an igraph object named mst, a list of parameter values named slingParams, a list of lineages (ordered sets of clusters) named lineages, and a list of principal_curve objects named curves.

References

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

Street, K., et al. (2018). "Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics." *BMC Genomics*, 19:477.

Examples

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

# plotting
sds <- as.SlingshotDataSet(pto)
plot(rd, col = cl, asp = 1)
lines(sds, type = 'c', lwd = 3)
```

SlingshotDataSet *Extract Slingshot output*

Description

This is a convenience function to extract a SlingshotDataSet from an object containing slingshot output. However, we now recommend using a PseudotimeOrdering object, in most cases. The SlingshotDataSet is, however, still used for plotting purposes.

Usage

```
SlingshotDataSet(data, ...)

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

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

## S4 method for signature 'PseudotimeOrdering'
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.

See Also

PseudotimeOrdering, as.SlingshotDataSet

Examples

```
data("slingshotExample")
rd <- slingshotExample$rd
cl <- slingshotExample$cl
library(SingleCellExperiment)
u <- matrix(rpois(140*50, 5), nrow = 50)
sce <- SingleCellExperiment(assays = list(counts = u),
                            reducedDims = SimpleList(PCA = rd),
                            colData = data.frame(clus = cl))
sce <- slingshot(sce, clusterLabels = 'clus', reducedDim = 'PCA')
SlingshotDataSet(sce)
```

SlingshotDataSet-class *Class* SlingshotDataSet

Description

This was the original class for storing slingshot results, but we now generally reommend using the PseudotimeOrdering class, instead. Most slingshot functions will still work with SlingshotDataSet objects, but will return PseudotimeOrdering objects, by default. To update old SlingshotDataSet objects, we have provided the as.PseudotimeOrdering conversion function. The only functions that require SlingshotDataSet objects are the plotting functions.

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.

Usage

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

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

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

Arguments

object	a SlingshotDataSet object.
x	a SlingshotDataSet object.

Value

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

Methods (by generic)

- show: a short summary of a SlingshotDataSet object.
- reducedDim: returns the matrix representing the reduced dimensional dataset.

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.clus` character. The label of the root cluster, or a vector of cluster labels giving the root clusters of each disjoint component of the graph.
- `end.clus` character. Vector of cluster labels indicating 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
- `omega` numeric or logical. Granularity parameter determining the maximum edge length for building the MST. See `getLineages`.
- `omega_scale` numeric. Scaling factor used for setting maximum edge length when `omega = TRUE`. See `getLineages`.

They may also specify how simultaneous principal curves were constructed (for a complete listing, see `getCurves`:

- `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.
- `reassign` logical. 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.method` character. 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.
- approx_points numeric. Number of points to use in estimating curves. See `getCurves` for details.

- allow.breaks logical. Whether to allow curves that diverge very early on in a trajectory to have different starting points.
- Other parameters specified by `principal_curve`.

See Also

`PseudotimeOrdering`

`slingshotExample` *Bifurcating lineages data*

Description

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

Usage

```
data("slingshotExample")
```

Format

`rd` is a matrix of coordinates in two dimensions, representing 140 cells. `cl` is a numeric vector of 140 corresponding cluster labels for each cell.

Source

Simulated data provided with the `slingshot` package.

Examples

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

Index