Package ‘SCORPIUS’

January 20, 2025

Type Package

Title Inferring Developmental Chronologies from Single-Cell RNA Sequencing Data

Version 1.0.9

Description An accurate and easy tool for performing linear trajectory inference on single cells using single-cell RNA sequencing data. In addition, 'SCORPIUS' provides functions for discovering the most important genes with respect to the reconstructed trajectory, as well as nice visualisation tools. Cannoodt et al. (2016) <doi:10.1101/079509>.

License GPL-3

Encoding UTF-8

URL `https://github.com/rcannood/SCORPIUS`, `http://rcannood.github.io/SCORPIUS/`

BugReports `https://github.com/rcannood/SCORPIUS/issues`

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

Depends R (>= 3.5.0)

Imports dplyr, dynutils (>= 1.0.3), dynwrap, grDevices, ggplot2 (>= 2.0), lmds, MASS, Matrix, mclust, methods, pbapply, pheatmap, princurve (>= 2.1.4), purrr, ranger, RANN, RColorBrewer, reshape2, stats, tidyr, TSP

Suggests anndata, covr, knitr, reticulate, rmarkdown, Seurat, SingleCellExperiment, testthat (>= 2.1.0)

NeedsCompilation no

Author Robrecht Cannoodt [aut, cre] (<`https://orcid.org/0000-0003-3641-729X`>, rcannood), Wouter Saelens [ctb] (<`https://orcid.org/0000-0002-7114-6248`>, zouter)

Maintainer Robrecht Cannoodt <`rcannood@gmail.com`>

Repository CRAN

Date/Publication 2023-08-07 17:30:05 UTC

Contents

`SCORPIUS-package` *SCORPIUS: Trajectory inference from single-cell RNA sequencing data.*

Description

SCORPIUS orders single cells with regard to an implicit timeline, such as cellular development or progression over time.

Dimensionality Reduction functions

`reduce_dimensionality`

Trajectory Inference functions

`infer_trajectory`, `infer_initial_trajectory`, `reverse_trajectory`, `gene_importances`, `extract_modules`

Visualisation functions

`draw_trajectory_plot`, `draw_trajectory_heatmap`

Datasets

`generate_dataset`, `ginhoux`

References

Cannoodt R. et al., SCORPIUS improves trajectory inference and identifies novel modules in dendritic cell development, bioRxiv (Oct., 2016). doi:10.1101/079509 (PDF).

Examples

```
## Load dataset from Schlitzer et al., 2015
data("ginhoux")

## Reduce dimensionality and infer trajectory with SCORPIUS
space <- reduce_dimensionality(ginhoux$expression, "spearman")
traj <- infer_trajectory(space)

## Visualise
draw_trajectory_plot(
  space,
  path = traj$path,
  progression_group = ginhoux$sample_info$group_name
)
```

---

draw_trajectory_heatmap

Draw time-series heatmap

Description

`draw_trajectory_heatmap` draws a heatmap in which the samples are ranked according their position in an inferred trajectory. In addition, the progression groups and feature modules can be passed along to further enhance the visualisation.

Usage

```
draw_trajectory_heatmap(
  x,
  time,
  progression_group = NULL,
  modules = NULL,
  show_labels_row = FALSE,
  show_labels_col = FALSE,
  scale_features = TRUE,
  progression_group_palette = NULL,
  ...
)
```

Arguments

`x`	A numeric matrix or a data frame with one row per sample and one column per feature.
`time`	A numeric vector containing the inferred time points of each sample along a trajectory.
`progression_group`	`NULL` or a vector (or factor) containing the groupings of the samples (default `NULL`).

`modules`	`NULL` or a data frame as returned by `extract_modules`.
`show_labels_row`	`TRUE` if the labels of the rows are to be plotted (default `FALSE`).
`show_labels_col`	`TRUE` if the labels of the cols are to be plotted (default `FALSE`).
`scale_features`	`TRUE` if the values of each feature is to be scaled (default `TRUE`).
`progression_group_palette`	A named vector palette for the progression group.
`...`	Optional arguments to `pheatmap`

Value

The output of the `pheatmap` function.

Examples

```
## Generate a dataset
dataset <- generate_dataset(num_genes=500, num_samples=300, num_groups=4)
expression <- dataset$expression
space <- reduce_dimensionality(expression, ndim=2)
groups <- dataset$sample_info$group_name
traj <- infer_trajectory(space)
time <- traj$time

gimp <- gene_importances(expression, traj$time, num_permutations = 0, ntree = 10000)
gene_sel <- gimp[1:50,]
expr_sel <- expression[,gene_sel$gene]

## Draw a time series heatmap
draw_trajectory_heatmap(expr_sel, time)

## Also show the progression groupings
draw_trajectory_heatmap(expr_sel, time, progression_group=groups)

## Use a different palette
draw_trajectory_heatmap(
  expr_sel, time, progression_group=groups,
  progression_group_palette = setNames(RColorBrewer::brewer.pal(4, "Set2"), paste0("Group ", 1:4))
)

## Group the genes into modules and visualise the modules in a heatmap
modules <- extract_modules(scale_quantile(expr_sel))
draw_trajectory_heatmap(expr_sel, time, progression_group=groups, modules=modules)
```

`draw_trajectory_plot` *Visualise SCORPIUS*

Description

`draw_trajectory_plot` is used to plot samples after performing dimensionality reduction. Additional arguments can be provided to colour the samples, plot the trajectory inferred by SCORPIUS, and draw a contour around the samples.

Usage

```
draw_trajectory_plot(
  space,
  progression_group = NULL,
  path = NULL,
  contour = FALSE,
  progression_group_palette = NULL,
  point_size = 2,
  point_alpha = 1,
  path_size = 0.5,
  path_alpha = 1,
  contour_alpha = 0.2
)
```

Arguments

`space`	A numeric matrix or a data frame containing the coordinates of samples.
`progression_group`	`NULL` or a vector (or factor) containing the groupings of the samples (default `NULL`).
`path`	A numeric matrix or a data frame containing the coordinates of the inferred path.
`contour`	`TRUE` if contours are to be drawn around the samples.
`progression_group_palette`	A named vector palette for the progression group.
`point_size`	The size of the points.
`point_alpha`	The alpha of the points.
`path_size`	The size of the path (if any).
`path_alpha`	The alpha of the path (if any).
`contour_alpha`	The alpha of the contour (if any).

Value

A ggplot2 plot.

Examples

```
## Generate a synthetic dataset
dataset <- generate_dataset(num_genes = 500, num_samples = 300, num_groups = 4)
space <- reduce_dimensionality(dataset$expression, ndim = 2)
groups <- dataset$sample_info$group_name

## Simply plot the samples
draw_trajectory_plot(space)

## Colour each sample according to its group
draw_trajectory_plot(space, progression_group = groups)

## Add contours to the plot
draw_trajectory_plot(space, progression_group = groups, contour = TRUE)

## Plot contours without colours
draw_trajectory_plot(space, contour = TRUE)

## Infer a trajectory and plot it
traj <- infer_trajectory(space)
draw_trajectory_plot(space, progression_group = groups, path = traj$path)
draw_trajectory_plot(space, progression_group = groups, path = traj$path, contour = TRUE)

## Visualise gene expression
draw_trajectory_plot(space, progression_group = dataset$expression[,1])
```

`extract_modules` *Extract modules of features*

Description

`extract_modules` uses adaptive branch pruning to extract modules of features, which is typically done on the smoothed expression returned by `gene_importances`.

Usage

```
extract_modules(
  x,
  time = NULL,
  suppress_warnings = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	A numeric matrix or a data frame with *M* rows (one per sample) and *P* columns (one per feature).

`time`	(Optional) Order the modules according to a pseudotime
`suppress_warnings`	Whether or not to suppress warnings when P > 1000
`verbose`	Whether or not Mclust will print output or not
`...`	Extra parameters passed to `Mclust`

Value

A data frame containing meta-data for the features in `x`, namely the order in which to visualise the features in and which module they belong to.

See Also

`gene_importances`

Examples

```
## Generate a dataset and visualise
dataset <- generate_dataset(num_genes=300, num_samples=200, num_groups=4)
expression <- dataset$expression
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(expression, ndim=2)
traj <- infer_trajectory(space)
time <- traj$time
draw_trajectory_plot(space, path=traj$path, group_name)

## Select most important genes (set ntree to at least 10000!)
gimp <- gene_importances(expression, traj$time, num_permutations = 0, ntree = 1000)
gene_sel <- gimp[1:50,]
expr_sel <- expression[,gene_sel$gene]

## Group the genes into modules and visualise the modules in a heatmap
modules <- extract_modules(scale_quantile(expr_sel))
draw_trajectory_heatmap(expr_sel, time, group_name, modules)
```

`generate_dataset` *Generate a synthetic dataset*

Description

`generate_dataset` generates an synthetic dataset which can be used for visualisation purposes.

Usage

```
generate_dataset(
  num_samples = 400,
  num_genes = 500,
  num_groups = 4
)
```

Arguments

`num_samples`	The number of samples the dataset will contain.
`num_genes`	The number of genes the dataset will contain.
`num_groups`	The number of groups the samples will be split up in.

Value

A list containing the expression data and the meta data of the samples.

See Also

SCORPIUS

Examples

```
## Generate a dataset
dataset <- generate_dataset(num_genes = 200, num_samples = 400, num_groups = 4)

## Reduce dimensionality and infer trajectory with SCORPIUS
space <- reduce_dimensionality(dataset$expression, ndim = 2)
traj <- infer_trajectory(space)

## Visualise
draw_trajectory_plot(space, path=traj$path, progression_group=dataset$sample_info$group_name)
```

`gene_importances` *Calculate the importance of a feature*

Description

Calculates the feature importance of each column in `x` in trying to predict the time ordering.

Usage

```
gene_importances(
  x,
  time,
  num_permutations = 0,
  ntree = 10000,
  ntree_perm = ntree/10,
  mtry = ncol(x) * 0.01,
  num_threads = 1,
  ...
)
```

Arguments

`x`	A numeric matrix or a data frame with *M* rows (one per sample) and *P* columns (one per feature).
`time`	A numeric vector containing the inferred time points of each sample along a trajectory as returned by `infer_trajectory`.
`num_permutations`	The number of permutations to test against for calculating the p-values (default: 0).
`ntree`	The number of trees to grow (default: 10000).
`ntree_perm`	The number of trees to grow for each of the permutations (default: ntree / 10).
`mtry`	The number of variables randomly samples at each split (default: 1% of features).
`num_threads`	Number of threads. Default is 1.
`...`	Extra parameters passed to `ranger`.

Value

a data frame containing the importance of each feature for the given time line

Examples

```
dataset <- generate_dataset(num_genes=500, num_samples=300, num_groups=4)
expression <- dataset$expression
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(expression, ndim=2)
traj <- infer_trajectory(space)
# set ntree to at least 1000!
gene_importances(expression, traj$time, num_permutations = 0, ntree = 1000)
```

`ginhoux` *scRNA-seq data of dendritic cell progenitors.*

Description

This dataset contains the expression values of the top 2000 most variable genes for 248 dendritic cell progenitors. Each cell is in one of three maturation stages: MDP, CDP or PreDC. The levels of the factor in `sample.info` are ordered according to the maturation process.

The number of genes had to be reduced specifically for reducing the package size of SCORPIUS. Use the following code to download the original data:

```
download.file("https://github.com/rcannood/SCORPIUS/raw/master/data-raw/ginhoux_orig.rds", destfile
ginhoux <- readRDS("local.rds")
# do something with ginhoux
```

Usage

```
ginhoux
```

Format

A list containing two data frames, `expression` (248x2000) and `sample_info` (248x1).

Source

`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE60783`

References

Schlitzer A, Sivakamasundari V, Chen J, Sumatoh HR et al. Identification of cDC1- and cDC2-committed DC progenitors reveals early lineage priming at the common DC progenitor stage in the bone marrow. Nat Immunol 2015 Jul;16(7):718-28. PMID: 26054720

See Also

SCORPIUS

`infer_initial_trajectory` *Infer an initial trajectory through space*

Description

`infer_initial_trajectory` infers an initial trajectory for `infer_trajectory` by clustering the points and calculating the shortest path through cluster centers. The shortest path takes into account the euclidean distance between cluster centers, and the density between those two points.

Usage

```
infer_initial_trajectory(space, k)
```

Arguments

`space`	A numeric matrix or a data frame containing the coordinates of samples.
`k`	The number of clusters

Value

the initial trajectory obtained by this method

infer_trajectory *Infer linear trajectory through space*

Description

`infer_trajectory` infers a trajectory through samples in a given space in a four-step process:

1. Perform *k*-means clustering
2. Calculate distance matrix between cluster centers using a custom distance function
3. Find the shortest path connecting all cluster centers using the custom distance matrix
4. Iteratively fit a curve to the given data using principal curves

Usage

```
infer_trajectory(
  space,
  k = 4,
  thresh = 0.001,
  maxit = 10,
  stretch = 0,
  smoother = "smooth_spline",
  approx_points = 100
)
```

Arguments

`space`	A numeric matrix or a data frame containing the coordinates of samples.
`k`	The number of clusters to cluster the data into.
`thresh`	convergence threshold on shortest distances to the curve.
`maxit`	maximum number of iterations.
`stretch`	A stretch factor for the endpoints of the curve, allowing the curve to grow to avoid bunching at the end. Must be a numeric value between 0 and 2.
`smoother`	choice of smoother. The default is `"smooth_spline"`, and other choices are `"lowess"` and `"periodic_lowess"`. The latter allows one to fit closed curves. Beware, you may want to use `iter = 0` with `lowess()`.
`approx_points`	Approximate curve after smoothing to reduce computational time. If `FALSE`, no approximation of the curve occurs. Otherwise, `approx_points` must be equal to the number of points the curve gets approximated to; preferably about 100.

Value

A list containing several objects:

- `path`: the trajectory obtained by principal curves.
- `time`: the time point of each sample along the inferred trajectory.

See Also

reduce_dimensionality, draw_trajectory_plot

Examples

```
## Generate an example dataset and visualise it
dataset <- generate_dataset(num_genes = 200, num_samples = 400, num_groups = 4)
space <- reduce_dimensionality(dataset$expression, ndim = 2)
draw_trajectory_plot(space, progression_group = dataset$sample_info$group_name)

## Infer a trajectory through this space
traj <- infer_trajectory(space)

## Visualise the trajectory
draw_trajectory_plot(space, path=traj$path, progression_group = dataset$sample_info$group_name)
```

---

reduce_dimensionality *Dimensionality reduction*

Description

reduce_dimensionality performs an eigenanalysis of the given dissimilarity matrix and returns coordinates of the samples represented in an ndim-dimensional space.

Usage

```
reduce_dimensionality(
  x,
  dist = c("spearman", "pearson", "euclidean", "cosine", "manhattan"),
  ndim = 3,
  num_landmarks = 1000
)
```

Arguments

x	a numeric matrix
dist	the distance metric to be used; can be any of the metrics listed in dynutils::calculate_distance().
ndim	the maximum dimension of the space which the data are to be represented in; must be in 1, 2, . . . , n-1.
num_landmarks	the number of landmarks to be selected.

Value

A matrix containing the coordinates of each sample, represented in an ndim-dimensional space.

See Also

SCORPIUS

Examples

```
## Generate an example dataset
dataset <- generate_dataset(num_genes = 200, num_samples = 400, num_groups = 4)

## Reduce the dimensionality of this dataset
space <- reduce_dimensionality(dataset$expression, ndim = 2)

## Visualise the dataset
draw_trajectory_plot(space, progression_group = dataset$sample_info$group_name)
```

`reverse_trajectory` *Reverse a trajectory*

Description

Since the direction of the trajectory is not specified, the ordering of a trajectory may be inverted using `reverse_trajectory`.

Usage

```
reverse_trajectory(trajectory)
```

Arguments

`trajectory`	A trajectory as returned by `infer_trajectory`.

Value

The same trajectory, but in the other direction.

See Also

`infer_trajectory`

Examples

```
## Generate an example dataset and infer a trajectory through it
dataset <- generate_dataset(num_genes = 200, num_samples = 400, num_groups = 4)
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(dataset$expression, ndim = 2)
traj <- infer_trajectory(space)

## Visualise the trajectory
draw_trajectory_plot(space, group_name, path = traj$path)

## Reverse the trajectory
reverse_traj <- reverse_trajectory(traj)
draw_trajectory_plot(space, group_name, path = reverse_traj$path)

plot(traj$time, reverse_traj$time, type = "l")
```

ti_scorpius *Infer a trajectory using SCORPIUS*

Description

Pass this object to `dynwrap::infer_trajectory()`.

Usage

```
ti_scorpius(
  distance_method = "spearman",
  ndim = 3L,
  k = 4L,
  thresh = 0.001,
  maxit = 10L,
  stretch = 0,
  smoother = "smooth_spline"
)
```

Arguments

`distance_method`	A character string indicating which correlationcoefficient (or covariance) is to be computed. One of "pearson", "spearman" (default), or "cosine". Domain: spearman, pearson, cosine. Default: spearman. Format: character.
`ndim`	The number of dimensions in the new space. Domain: U(2, 20). Default: 3. Format: integer.
`k`	The number of clusters to cluster the data into to construct the initial trajectory. Domain: U(1, 20). Default: 4. Format: integer.
`thresh`	`principal_curve` parameter; convergence threshhold on shortest distances to the curve. Domain: e^U(-11.51, 11.51). Default: 0.001. Format: numeric.
`maxit`	`principal_curve` parameter; maximum number of iterations. Domain: U(0, 50). Default: 10. Format: integer.
`stretch`	`principal_curve` parameter; a factor by which the curve can be extrapolated when points are projected. Domain: U(0, 5). Default: 0. Format: numeric.
`smoother`	`principal_curve` parameter; choice of smoother. Domain: smooth_spline, lowess, periodic_lowess. Default: smooth_spline. Format: character.

Value

A dynwrap TI method.

Index