Package ‘SCORPIUS’

June 9, 2021

Type Package

Title Inferring Developmental Chronologies from Single-Cell RNA Sequencing Data

Version 1.0.8

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.


License GPL-3

Encoding UTF-8

URL https://github.com/rcannood/SCORPIUS

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

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr, R.rsp

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 covr, knitr, rmarkdown, R.rsp, 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 2021-06-09 12:40:02 UTC
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

**draw_trajectory_heatmap**

**Examples**

```r
## 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
)
```

**Description**

draw_trajectory_heatmap draws a heatmap in which the samples are ranked according to 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 \texttt{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 \texttt{pheatmap}

Value
The output of the \texttt{pheatmap} function.

Examples

```r
## Not run:
## 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$\texttt{time}, num\_permutations = 0, ntree = 10000)
gene\_sel <- gimp[1:50,]
expr\_sel <- expression[,gene\_sel$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)

## End(Not run)
```
draw_trajectory_plot

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,  # A numeric matrix or a data frame containing the coordinates of samples.
  progression_group = NULL,  # NULL or a vector (or factor) containing the groupings of the samples (default NULL).
  path = NULL,  # A numeric matrix or a data frame containing the coordinates of the inferred path.
  contour = FALSE,  # TRUE if contours are to be drawn around the samples.
  progression_group_palette = NULL,  # A named vector palette for the progression group.
  point_size = 2,  # The size of the points.
  point_alpha = 1,  # The alpha of the points.
  path_size = 0.5,  # The size of the path (if any).
  path_alpha = 1,  # The alpha of the path (if any).
  contour_alpha = 0.2  # The alpha of the contour (if any).
)

Arguments

space  
progression_group  
path  
contour  
progression_group_palette  
point_size  
point_alpha  
path_size  
path_alpha  
contour_alpha

Value

A ggplot2 plot.
Examples

```r
## 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**

```r
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).
generate_dataset

Generate a synthetic dataset

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

```r
## 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 a synthetic dataset which can be used for visualisation purposes.

Usage
generate_dataset(
  num_samples = 400,
  num_genes = 500,
  num_groups = 4
)
gene_importances

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 = 500, num_samples = 1000, 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 sampled 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)

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:

ginhoux <- readRDS("local.rds")
# do something with ginhoux
infer_initial_trajectory

Usage

ginhoux

Format

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

Source


References


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

**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**

```r
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.
reduce_dimensionality

See Also

reduce_dimensionality, draw_trajectory_plot

Examples

## Generate an example dataset and visualise it
dataset <- generate_dataset(num_genes = 500, num_samples = 1000, 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 = 500, num_samples = 1000, 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 = 500, num_samples = 1000, 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

```r
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 correlation coefficient (or covariance) is to be computed. One of "pearson", "spearman" (default), or "cosine". Domain: spearman, pearson, cosine. Default: spearman. Format: character.
- **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 threshold on shortest distances to the curve. Domain: e^U(-11.51, 11.51). Default: 0.001. Format: numeric.
- **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.
ti_scorpius_run_fun

---

**ti_scorpius_run_fun**  
*Run scorpius using the dynwrap pipeline*

---

**Description**

Run scorpius using the dynwrap pipeline

**Usage**

```r
ti_scorpius_run_fun(expression, priors, parameters, seed = NULL, verbose = 0)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Expression matrix</td>
</tr>
<tr>
<td>priors</td>
<td>Priors</td>
</tr>
<tr>
<td>parameters</td>
<td>Parameters</td>
</tr>
<tr>
<td>seed</td>
<td>Random seed</td>
</tr>
<tr>
<td>verbose</td>
<td>Verbosity level</td>
</tr>
</tbody>
</table>
Index

* datasets
  ginhoux, 9
  draw_trajectory_heatmap, 2, 3
  draw_trajectory_plot, 2, 5, 12
  dynutils::calculate_distance(), 12
  dynwrap::infer_trajectory(), 14
  extract_modules, 2, 4, 6
  gene_importances, 2, 6, 7, 8
  generate_dataset, 2, 7
  ginhoux, 2, 9
  infer_initial_trajectory, 2, 10
  infer_trajectory, 2, 9, 10, 11, 13
  Mclust, 7
  pheatmap, 4
  ranger, 9
  reduce_dimensionality, 2, 12, 12
  reverse_trajectory, 2, 13
  SCORPIUS, 8, 10, 12
  SCORPIUS (SCORPIUS-package), 2
  SCORPIUS-package, 2
  ti_scorpius, 14
  ti_scorpius_run_fun, 15