

Package ‘SCORPIUS’

June 29, 2018

Type Package

Title Inferring Developmental Chronologies from Single-Cell RNA Sequencing Data

Version 1.0.2

Date 2018-06-12

Description An accurate and easy tool for performing 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

URL <http://github.com/rcannood/SCORPIUS>

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

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

Depends R (>= 3.0.0)

Imports dplyr, dynutils, grDevices, ggplot2 (>= 2.0), magrittr, MASS, mclust, pbapply, pheatmap, prncurve (>= 2.0.2), purrr, ranger, RColorBrewer, reshape2, splines, stats, tidyr, TSP, utils

Suggests knitr, rmarkdown, testthat

NeedsCompilation no

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

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2018-06-29 14:55:11 UTC

R topics documented:

apply_quantile_scale	2
apply_uniform_scale	3
draw_trajectory_heatmap	3
draw_trajectory_plot	5
evaluate_dim_red	6
evaluate_trajectory	6
extract_modules	7
generate_dataset	8
gene_importances	9
ginhoux	10
infer_initial_trajectory	11
infer_trajectory	12
knn	13
knn_distances	14
reduce_dimensionality	14
reverse_trajectory	15
scale_quantile	16
scale_uniform	17
SCORPIUS	18
Index	19

apply_quantile_scale *Apply a quantile scale.*

Description

Anything outside the range of [0, 1] will be set to 0 or 1.

Usage

```
apply_quantile_scale(x, addend, multiplier)
```

Arguments

x	A numeric matrix or data frame.
addend	A minimum vector for each column
multiplier	A scaling vector for each column

Value

The scaled matrix. The numeric centering and scalings used are returned as attributes.

apply_uniform_scale *Apply a uniform scale*

Description

Apply a uniform scale

Usage

```
apply_uniform_scale(x, addend, multiplier)
```

Arguments

x	A numeric matrix or data frame.
addend	A centering vector for each column
multiplier	A scaling vector for each column

Value

The centered, scaled matrix. The numeric centering and scalings used are returned as attributes.

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,  
  ...  
)
```

Arguments

x	A numeric matrix or 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).
...	Optional arguments to pheatmap

Value

The output of the [pheatmap](#) function.

Examples

```
## Not run:
## Generate a dataset
dataset <- generate_dataset(type="s", num_genes=500, num_samples=300, num_groups=4)
expression <- dataset$expression
space <- reduce_dimensionality(expression, correlation_distance, ndim=2)
groups <- dataset$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=groups)

## 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 *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)
```

Arguments

space	A numeric matrix or 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 data frame containing the coordinates of the inferred path.
contour	TRUE if contours are to be drawn around the samples.

Value

A ggplot2 plot.

Examples

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

evaluate_dim_red *Evaluate the dimensionality reduction*

Description

evaluate_dim_red calculates the *accuracy* of the dimensionality reduction by performing 5-nearest neighbour leave-one-out-cross-validation (5NN LOOCV).

Usage

```
evaluate_dim_red(space, progression, k=5)
```

Arguments

space	A numeric vector containing the inferred time points of each sample along a trajectory.
progression	A factor or a numeric vector which represents the progression stages of each sample.
k	The maximum number of nearest neighbours to search (default 5).

Value

The accuracy of a 5NN LOOCV using the dimensionality reduction to predict the progression stage of a sample.

Examples

```
## Generate a dataset
dataset <- generate_dataset(type="s", num_genes=500, num_samples=300, num_groups=4)
space <- reduce_dimensionality(dataset$expression, correlation_distance, ndim=2)

## Evaluate the trajectory timeline
evaluate_dim_red(space, dataset$sample_info$group_name)
```

evaluate_trajectory *Evaluate the inferred timeline*

Description

evaluate_trajectory calculates the *consistency* of the predicted time points of samples versus the known progression stages.

Usage

```
evaluate_trajectory(time, progression)
```

Arguments

time	A numeric vector containing the inferred time points of each sample along a trajectory.
progression	A factor or a numeric vector which represents the progression stages of each sample.

Value

The consistency value for the predicted timeline.

Examples

```
## Generate a dataset
dataset <- generate_dataset(type="s", num_genes=500, num_samples=1000, num_groups=4)
space <- reduce_dimensionality(dataset$expression, correlation_distance, ndim=2)
traj <- infer_trajectory(space)

## Evaluate the trajectory timeline
evaluate_trajectory(traj$time, dataset$sample_info$group_name)
```

extract_modules	<i>Extract modules of features</i>
-----------------	------------------------------------

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 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(type="s", num_genes=500, num_samples=300, num_groups=4)
expression <- dataset$expression
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(expression, correlation_distance, ndim=2)
traj <- infer_trajectory(space)
time <- traj$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(
  type = c("splines", "polynomial"),
  num_samples = 400,
  num_genes = 500,
  num_groups = 4
)

```

Arguments

type	The type of function used in order to generate the expression data. Must be either "splines" (default) or "polynomial" (or abbreviations thereof).
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

[correlation_distance](#), [reduce_dimensionality](#), [infer_trajectory](#), [draw_trajectory_plot](#)

Examples

```
## Generate a dataset
dataset <- generate_dataset(type = "poly", num_genes = 500, num_samples = 1000, num_groups = 4)

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

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

gene_importances	<i>Calculate the importance of a feature</i>
------------------	--

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

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

Value

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

Examples

```
dataset <- generate_dataset(type="s", num_genes=500, num_samples=300, num_groups=4)
expression <- dataset$expression
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(expression, correlation_distance, ndim=2)
traj <- infer_trajectory(space)
# set ntree to at least 1000!
gene_importances(expression, traj$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/ginhoux_orig.rds", destfile = "1")
ginhoux <- readRDS("local.rds")
# do something with ginhoux
```

Usage

```
ginhoux
```

Format

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

Source

<http://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

Examples

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

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

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

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 data frame containing the coordinates of samples.
k	The number of clusters to cluster the data into.

Value

the initial trajectory obtained by this method

Examples

```
## Generate an example dataset and visualise it
dataset <- generate_dataset(type = "poly", num_genes = 500, num_samples = 1000, num_groups = 4)
space <- reduce_dimensionality(dataset$expression, correlation_distance, ndim=2)
draw_trajectory_plot(space, progression_group = dataset$sample_info$group_name)

## Infer a trajectory through this space
init_traj <- infer_initial_trajectory(space, k = 4)

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

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")
```

Arguments

space	A numeric matrix or data frame containing the coordinates of samples.
k	The number of clusters to cluster the data into.
thresh	principal_curve parameter: convergence threshold on shortest distances to the curve
maxit	principal_curve parameter: maximum number of iterations
stretch	principal_curve parameter: a factor by which the curve can be extrapolated when points are projected
smoother	principal_curve parameter: choice of smoother

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(type = "poly", num_genes = 500, num_samples = 1000, num_groups = 4)
space <- reduce_dimensionality(dataset$expression, correlation_distance, ndim=2)
draw_trajectory_plot(space, progression_group = dataset$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)
```

knn *k Nearest Neighbour indices and distances*

Description

knn returns the indices and distances of the k nearest neighbours of each sample.

Usage

```
knn(dist, k, self_loops = FALSE)
```

Arguments

dist	A numeric matrix, data frame or "dist" object.
k	The maximum number of nearest neighbours to search.
self_loops	TRUE if samples with the same index or name are allowed to be neighbours.

Value

A list containing two matrices indices and distances

Examples

```
## Calculate the kNN distances within a set of samples
x <- matrix(rnorm(50*10, mean=0, sd=1), ncol=10)
dist <- dist(x)
knnd <- knn(dist, 10)
plot(density(knnd$distances))

## Calculate the kNN distances between two sets of samples
y <- matrix(rnorm(100*10, mean=1, sd=2), ncol=10)
dist <- euclidean_distance(x, y)
knnd <- knn(dist, 10)
plot(density(knnd$distances))
```

knn_distances	<i>k</i> Nearest Neighbour distances
---------------	--------------------------------------

Description

knn_distances returns the distances of the k nearest neighbours of each sample.

Usage

```
knn_distances(dist, k, self_loops = FALSE)
```

Arguments

dist	A numeric matrix, data frame or "dist" object.
k	The maximum number of nearest neighbours to search.
self_loops	TRUE if samples with the same index or name are allowed to be neighbours.

Value

A matrix containing the distances of the k nearest neighbours of each sample.

Examples

```
## Calculate the kNN distances within a set of samples
x <- matrix(rnorm(50*10, mean=0, sd=1), ncol=10)
dist <- dist(x)
knnd <- knn_distances(dist, 10)
plot(density(knnd))

## Calculate the kNN distances between two sets of samples
y <- matrix(rnorm(100*10, mean=1, sd=2), ncol=10)
dist <- euclidean_distance(x, y)
knnd <- knn_distances(dist, 10)
plot(density(knnd))
```

reduce_dimensionality	<i>Dimensionality reduction</i>
-----------------------	---------------------------------

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_fun, ndim = 3, landmark_method = c("naive",
  "none"), num_landmarks = 1000, rescale = T)
```

Arguments

x	a numeric matrix
dist_fun	the distance function to be used; must have exactly two arguments, namely <code>dist_fun(x, y)</code> .
ndim	the maximum dimension of the space which the data are to be represented in; must be in 1, 2, ..., n-1.
landmark_method	Must be "naive" for now. Other landmark methods will be supported in the future.
num_landmarks	the number of landmarks to be selected.
rescale	A logical indicating whether or not the returned space should be rescaled and centered.

Value

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

See Also

[correlation_distance](#), [scale_uniform](#), [draw_trajectory_plot](#)

Examples

```
## Generate an example dataset
dataset <- generate_dataset(type = "poly", num_genes = 500, num_samples = 1000, num_groups = 4)

## Reduce the dimensionality of this dataset
space <- reduce_dimensionality(dataset$expression, correlation_distance, 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(type="poly", num_genes=500, num_samples=1000, num_groups=4)
group_name <- dataset$sample_info$group_name
space <- reduce_dimensionality(dataset$expression, correlation_distance, 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)

## It's the same but reversed?!
plot(traj$time, reverse_traj$time, type="l")
```

scale_quantile

Calculate and apply a quantile scale

Description

Calculate and apply a quantile scale

Usage

```
scale_quantile(x, outlier_cutoff = 0.05)
```

Arguments

`x` A numeric matrix or data frame.
`outlier_cutoff` The quantile cutoff for outliers (default 0.05).

Value

The centered, scaled matrix. The numeric centering and scalings used are returned as attributes.

Examples

```
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200*2, sd = 10, mean = 5), ncol=2)

## Center the dataset at c(0, 0) with a minimum of c(-.5, -.5) and a maximum of c(.5, .5)
x_scaled <- scale_quantile(x)

## Plot rescaled data
plot(x_scaled)

## Show ranges of each column
apply(x_scaled, 2, range)
```

`scale_uniform`*Scaling and centering of matrix-like objects*

Description

`scale_uniform` uniformly scales a given matrix such that the returned space is centered on center, and each column was scaled equally such that the range of each column is at most `max_range`.

Usage

```
scale_uniform(x, center = 0, max_range = 1)
```

Arguments

<code>x</code>	A numeric matrix or data frame.
<code>center</code>	The new center point of the data.
<code>max_range</code>	The maximum range of each column.

Value

The centered, scaled matrix. The numeric centering and scalings used are returned as attributes.

Examples

```
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200*2, sd = 10, mean = 5), ncol=2)

## Center the dataset at c(0, 0) with a minimum of c(-.5, -.5) and a maximum of c(.5, .5)
x_scaled <- scale_uniform(x, center=0, max_range=1)

## Plot rescaled data
plot(x_scaled)

## Show ranges of each column
apply(x_scaled, 2, range)
```

SCORPIUS

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.

Distance functions

`correlation_distance`, `euclidean_distance`

Dimensionality Reduction functions

`reduce_dimensionality`, `scale_uniform`, `scale_quantile`

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`

Evaluation functions

`evaluate_trajectory`, `evaluate_dim_red`

Index

*Topic **datasets**

ginhoux, [10](#)

[apply_quantile_scale](#), [2](#)
[apply_uniform_scale](#), [3](#)

[correlation_distance](#), [9](#), [15](#), [18](#)

[draw_trajectory_heatmap](#), [3](#), [18](#)
[draw_trajectory_plot](#), [5](#), [9](#), [12](#), [15](#), [18](#)

[euclidean_distance](#), [18](#)
[evaluate_dim_red](#), [6](#), [18](#)
[evaluate_trajectory](#), [6](#), [18](#)
[extract_modules](#), [4](#), [7](#), [18](#)

[gene_importances](#), [7](#), [8](#), [9](#), [18](#)
[generate_dataset](#), [8](#), [18](#)
[ginhoux](#), [10](#), [18](#)

[infer_initial_trajectory](#), [11](#), [18](#)
[infer_trajectory](#), [9](#), [11](#), [12](#), [15](#), [16](#), [18](#)

[knn](#), [13](#)
[knn_distances](#), [14](#)

[Mclust](#), [7](#)

[pheatmap](#), [4](#)
[principal_curve](#), [12](#)

[ranger](#), [9](#)
[reduce_dimensionality](#), [9](#), [12](#), [14](#), [18](#)
[reverse_trajectory](#), [15](#), [18](#)

[scale_quantile](#), [16](#), [18](#)
[scale_uniform](#), [15](#), [17](#), [18](#)
[SCORPIUS](#), [18](#)
[SCORPIUS-package \(SCORPIUS\)](#), [18](#)