Package ‘destiny’

December 21, 2016

Type Package
Title Creates diffusion maps
Version 2.0.3
Date 2014-12-19
Description Create and plot diffusion maps.
License GPL
Encoding UTF-8
Depends R (>= 3.2.0)
Imports methods, graphics, grDevices, utils, stats, Matrix, Rcpp (>= 0.10.3), RcppEigen, Biobase, BiocGenerics, Hmisc, FNN, VIM, proxy, igraph, smoother, scales, scatterplot3d
LinkingTo Rcpp, RcppEigen
SystemRequirements C++11
NeedsCompilation yes
Enhances rgl
Suggests ggplot2, nbconvertR
VignetteBuilder nbconvertR
biocViews CellBiology, CellBasedAssays, Clustering, Software, Visualization
RoxygenNote 5.0.1
Author Philipp Angerer [cre, aut], Laleh Haghverdi [ctb], Maren Büttner [ctb],
Description

Convert a DiffusionMap or DPT object to other classes

Usage

```r
## S4 method for signature 'DiffusionMap'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

fortify.DiffusionMap(model, data, ...)
```
## S4 method for signature 'DPT'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

fortify.DPT(model, data, ...)

## S4 method for signature 'DPT'
as.matrix(x, ...)

### Arguments

- `x, model` A `DiffusionMap` or `DPT` object
- `row.names` NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
- `optional` logical. If TRUE, setting row names and converting column names (to syntactic names: see `make.names`) is optional.
- `...` Passed to `as.data.frame`
- `data` ignored

### Details

`fortify` is a `ggplot2` generic allowing a diffusion map to be used as data parameter in `ggplot` or `qplot`.

### Value

An object of the desired class

### See Also

`DiffusionMap` accessors, extractions, `DiffusionMap` methods for more methods

### Examples

```r
library(Biobase)
data(guo)
dm <- DiffusionMap(guo)
classes <- vapply(as.data.frame(dm), class, character(1L))
stopifnot(all(classes[paste0('DC', 1:20)] == 'numeric'))
stopifnot(all(classes[featureNames(guo)] == 'numeric'))
stopifnot(all(classes[varLabels(guo)] == c('factor', 'integer')))```

---

### colorlegend

**Color legend**

#### Description

Creates a color legend for a vector used to color a plot. It will use the current `palette()` or the specified `pal` as reference.
Usage

colorlegend(col, pal = palette(), log = FALSE, posx = c(0.9, 0.93),
posy = c(0.05, 0.9), main = NULL, cex_main = par("cex.sub"),
cex_axis = par("cex.axis"), col_main = par("col.sub"),
col_lab = par("col.lab"), steps = 5, steps_color = 100, digit = 2,
left = FALSE, ..., cex.main = NULL, cex.axis = NULL, col.main = NULL,
col.lab = NULL)

Arguments

col Vector of factor, integer, or double used to determine the ticks.
pal If col is double, pal is used as a continuous palette, else as categorical one
log Use logarithmic scale?
posx Left and right borders of the color bar relative to plot area (Vector of length 2; 0-1)
posy Bottom and top borders of color bar relative to plot area (Vector of length 2; 0-1)
main Legend title
cex_main Size of legend title font (default: subtitle font size \texttt{par("cex.sub")})
cex_axis Size of ticks/category labels (default: axis font size \texttt{par("cex.axis")})
col_main Color of legend title (default: subtitle color \texttt{par("col.sub")})
col_lab Color of tick or category labels (default: axis color \texttt{par("col.lab")})
steps Number of labels in case of a continuous axis. If 0 or FALSE, draw no ticks
steps_color Number of gradient samples in case of continuous axis
digit Number of digits for continuous axis labels
left logical. If TRUE, invert posx
... Additional parameters for the \texttt{text} call used for labels
cex.main, cex.axis, col.main, col.lab For compatibility with \texttt{par}

Details

When passed a factor or integer vector, it will create a discrete legend, whereas a double vector will result in a continuous bar.

Value

This function is called for the side effect of adding a colorbar to a plot and returns nothing/NULL.

Examples

color_data <- 1:6
par(mar = par("mar") + c(0, 0, 0, 3))
plot(sample(6), col = color_data)
colorlegend(color_data)
**cube_helix**

**Sequential color palette using the cube helix system**

**Description**

Creates a perceptually monotonously decreasing (or increasing) lightness color palette with different tones.

**Usage**

```r
cube_helix(n = 6, start = 0, r = 0.4, hue = 0.8, gamma = 1,
            light = 0.85, dark = 0.15, reverse = FALSE)
```

```r
cube_helix(n = 6, start = 0, r = 0.4, hue = 0.8, gamma = 1,
            light = 0.85, dark = 0.15, reverse = FALSE)
```

```r
cube_helix(n = 6, start = 0, r = 0.4, hue = 0.8, gamma = 1,
            light = 0.85, dark = 0.15, reverse = FALSE)
```

**Arguments**

- `n`: Number of colors to return (default: 6)
- `start`: Hue to start helix at (start ∈ [0, 3], default: 0)
- `r`: Number of rotations of the helix. Can be negative. (default: 0.4)
- `hue`: Saturation. 0 means greyscale, 1 fully saturated colors (default: 0.8)
- `gamma`: Emphasize darker (gamma < 1) or lighter (gamma > 1) colors (default: 1)
- `light`: Lightest lightness (default: 0.85)
- `dark`: Darkest lightness (default: 0.15)
- `reverse`: logical. If TRUE, reverse lightness (default: FALSE)
- `...`: parameters passed to `discrete_scale` or `continuous_scale`
- `discrete`: If TRUE, return a discrete scale, if FALSE a continuous one (default: TRUE)
- `guide`: Type of scale guide to use. See `guides`

**Value**

A character vector of hex colors with length `n`
Examples

```r
palette(cube_helix())
image(matrix(1:6), col = 1:6, pch = 19, axes = FALSE)

cr <- scales::colour_ramp(cube_helix(12, r = 3))
r <- runif(100)
plot(1:100, r, col = cr(r), type = 'b', pch = 20)
```

---

**destiny**  
Create and plot diffusion maps

---

**Description**

The main function is `DiffusionMap`, which returns an object you can `plot` (`plot.DiffusionMap` is then called).

**Examples**

```r
demo(destiny, ask = FALSE)
```

---

**destiny generics**  
**destiny generics**

---

**Description**

`destiny` provides several generic methods and implements them for the `DiffusionMap` and `Sigmas` classes.

**Usage**

```r
eigenvalues(object)
eigenvalues(object) <- value
eigenvectors(object)
eigenvectors(object) <- value
sigmas(object)
sigmas(object) <- value
dataset(object)
dataset(object) <- value
distance(object)
```
DiffusionMap accessors

```r
distance(object) <- value

optimal_sigma(object)
```

**Arguments**

- `object`: Object from which to extract or to which to assign a value
- `value`: Value to assign within an object

**Value**

- `eigenvalues` retrieves the numeric eigenvalues
- `eigenvectors` retrieves the eigenvectors matrix
- `sigmas` retrieves the `Sigmas` from an object utilizing it as kernel width
- `dataset` retrieves the data the object was created from
- `distance` retrieves the distance metric used to create the object, e.g. euclidean
- `optimal_sigma` retrieves the numeric value of the optimal sigma or local sigmas

**See Also**

- [DiffusionMap methods](#) and [Sigmas class](#) for implementations

---

**DiffusionMap accessors**

**DiffusionMap accession methods**

**Description**

Get and set eigenvalues, eigenvectors, and sigma(s) of a `DiffusionMap` object or print information about a DiffusionMap

**Usage**

```r
## S4 method for signature 'DiffusionMap'
eigenvalues(object)

## S4 replacement method for signature 'DiffusionMap'
eigenvalues(object) <- value

## S4 method for signature 'DiffusionMap'
eigenvectors(object)

## S4 replacement method for signature 'DiffusionMap'
eigenvectors(object) <- value

## S4 method for signature 'DiffusionMap'
sigmas(object)

## S4 replacement method for signature 'DiffusionMap'
sigmas(object) <- value
```
## S4 method for signature 'DiffusionMap'
dataset(object)

## S4 replacement method for signature 'DiffusionMap'
dataset(object) <- value

## S4 method for signature 'DiffusionMap'
distance(object)

## S4 replacement method for signature 'DiffusionMap'
distance(object) <- value

## S4 method for signature 'DiffusionMap'
optimal_sigma(object)

### Arguments

- **object**: A DiffusionMap
- **value**: Vector of eigenvalues or matrix of eigenvectors to get/set

### Value

The assigned or retrieved value

### See Also

- extractions, DiffusionMap methods, coercions for more methods

### Examples

data(guo)
dm <- DiffusionMap(guo)
eigenvalues(dm)
eigenvectors(dm)
sigmas(dm)
dataset(dm)
optimal_sigma(dm)

---

**DiffusionMap class**

Create a diffusion map of cells

**Description**

The provided data can be a double matrix of expression data or a data.frame with all non-integer (double) columns being treated as expression data features (and the others ignored), or an ExpressionSet.
DiffusionMap class

Usage
DiffusionMap(data, sigma = "local", k = find_dm_k(nrow(data) - 1L),
    n_eigs = min(20L, nrow(data) - 2L), density_norm = TRUE, ...
    distance = c("euclidean", "cosine", "rankcor"), n_local = 5L,
    censor_val = NULL, censor_range = NULL, missing_range = NULL,
    vars = NULL, verbose = !is.null(censor_range), suppress_dpt = FALSE)

Arguments

- **data**  Expression data to be analyzed. Provide vars to select specific columns other than the default: all double value columns
- **sigma** Diffusion scale parameter of the Gaussian kernel. One of 'local', 'global', a (numeric) global sigma or a Sigmas object. When choosing 'global', a global sigma will be calculated using find_sigmas. (Optional. default: 'local') A larger sigma might be necessary if the eigenvalues can not be found because of a singularity in the matrix
- **k** Number of nearest neighbors to consider (default: a guess between 100 and \( n - 1 \). See find_dm_k).
- **n_eigs** Number of eigenvectors/values to return (default: 20)
- **density_norm** logical. If TRUE, use density normalisation
- **distance** Distance measurement method. Euclidean distance (default), cosine distance \((1 - \text{corr}(c_1, c_2))\) or rank correlation distance \((1 - \text{corr}(\text{rank}(c_1), \text{rank}(c_2)))\).
- **n_local** If sigma == 'local', the n_localth nearest neighbor determines the local sigma.
- **censor_val** Value regarded as uncertain. Either a single value or one for every dimension (Optional, default: censor_val)
- **censor_range** Uncertainty range for censoring (Optional, default: none). A length-2-vector of certainty range start and end. TODO: also allow \(2 \times G\) matrix
- **missing_range** Whole data range for missing value model. Has to be specified if NAs are in the data
- **vars** Variables (columns) of the data to use. Specifying NULL will select all columns (default: All floating point value columns)
- **verbose** Show a progressbar and other progress information (default: do it if censoring is enabled)
- **suppress_dpt** Specify TRUE to skip calculation of necessary (but spacious) information for DPT in the returned object (default: FALSE)

Value
A DiffusionMap object:

Slots

- **eigenvalues** Eigenvalues ranking the eigenvectors
- **eigenvectors** Eigenvectors mapping the datapoints to n_eigs dimensions
- **sigmas** Sigmas object with either information about the find_sigmas heuristic run or just local or optimal_sigma.
data_env Environment referencing the data used to create the diffusion map
eigenvec0 First (constant) eigenvector not included as diffusion component.
transitions Transition probabilities. Can be NULL
d Density vector of transition probability matrix
d_norm Density vector of normalized transition probability matrix
k The k parameter for kNN
n_local The n_localth nearest neighbor is used to determine local kernel density
density_norm Was density normalization used?
distance Distance measurement method used.
censor_val Censoring value
censor_range Censoring range
missing_range Whole data range for missing value model
vars Vars parameter used to extract the part of the data used for diffusion map creation

See Also
DiffusionMap-methods to get and set the slots. find_sigmas to pre-calculate a fitting global sigma parameter

Examples

data(guo)
DiffusionMap(guo)
DiffusionMap(guo, 13, censor_val = 15, censor_range = c(15, 40), verbose = TRUE)

Description
Methods for external operations on diffusion maps

Usage

## S4 method for signature 'DiffusionMap'
print(x)

## S4 method for signature 'DiffusionMap'
show(object)

Arguments

x, object A DiffusionMap

Value

The DiffusionMap object (print), or NULL (show), invisibly
Predict new data points using an existing DiffusionMap. The resulting matrix can be used in the plot method for the DiffusionMap

**Usage**

```r
dm_predict(dm, new_data, verbose = FALSE)
```

**Arguments**

- `dm` A DiffusionMap object
- `new_data` New data points to project into the diffusion map. Can be a matrix, data.frame, or an ExpressionSet.
- `verbose` Show progress messages?

**Value**

A \( \text{nrow(new_data)} \times \text{ncol(eigenvectors(diff))} \) matrix of projected diffusion components for the new data.

**Examples**

```r
data(guo)
g1 <- guo[, guo$num_cells != 32L]
g2 <- guo[, guo$num_cells == 32L]
dm <- DiffusionMap(g1)
dc2 <- dm_predict(dm, g2)
plot(dm, new_dcs = dc2)
```
**DPT**

**Diffusion Pseudo Time**

**Description**

Create pseudotime ordering and assigns cell to one of three branches

**Usage**

\[
\text{DPT}(\text{dm, tips = random\_root(\text{dm}), ..., w\_width = 0.1})
\]

**Arguments**

- **dm**: A DiffusionMap object. Its transition probabilities will be used to calculate the DPT
- **tips**: The cell index/indices from which to calculate the DPT(s) (integer of length 1-3)
- **...**: All parameters after this have to be specified by name
- **w_width**: Window width to use for deciding the branch cutoff

**Details**

Treat it as a matrix of pseudotime by subsetting (`\text{dim nrow ncol as.matrix}`), and as a list of pseudotime, and expression vectors (`\$ [[ names as.data.frame]`).

**Value**

A DPT object:

**Slots**

- **branch** (of integer) recursive branch labels for each cell (row); NA for undeceided. Use `\text{branch\_divide}` to modify this.
- **tips** (of logical) indicating if a cell (row) is a tip of the corresponding branch level (col)
- **dm** DiffusionMap used to create this DPT object

**Examples**

```r
data(guo\_norm)
\text{dm} \leftarrow \text{DiffusionMap(guo\_norm)}
\text{dpt} \leftarrow \text{DPT(dm, branching = TRUE)}
\text{str(dpt)}
```
DPT matrix methods

Description
Treat DPT object as a matrix of cell-by-cell DPT distances.

Usage

## S4 method for signature 'DPT', index, index, logicalOrMissing
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DPT', index, missing, logicalOrMissing
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DPT', missing, index, logicalOrMissing
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DPT', missing, missing, logicalOrMissing
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DPT'
nrow(x)

## S4 method for signature 'DPT'
col(x)

## S4 method for signature 'DPT'
dim(x)

See Also

as.matrix.DPT

DPT methods

Description
Methods for the DPT class. branch_divide subdivides branches for plotting (see the examples).

Usage

branch_divide(dpt, divide = integer(0L))

## S4 method for signature 'DPT'
dataset(object)

## S4 replacement method for signature 'DPT'
dataset(object) <- value
eig_decomp

**Arguments**

- **dpt, object**: DPT object
- **divide**: Vector of branch numbers to use for division
- **value**: Value of slot to set

**Value**

- `branch_divide` and `dataset <-` return the changed object, `dataset` the extracted data.

**See Also**

- `plot.DPT` uses `branch_divide` for its `divide` argument.

**Examples**

```r
data(guo_norm)
dpt <- DPT(DiffusionMap(guo_norm))
dpt_9_branches <- branch_divide(dpt, 1:3)
plot(dpt_9_branches, col_by = 'branch')
```

eig_decomp

**Description**

Fast eigen decomposition using ARPACK

**Usage**

```r
eig_decomp(M, n_eigs, sym = isSymmetric(M))
```

**Arguments**

- **M**: A matrix (e.g. from the Matrix package)
- **n_eigs**: Number of eigenvectors to return
- **sym**: TRUE if `M` is symmetric

**Value**

- `n` eigenvectors of the transition matrix

**Examples**

```r
eig_decomp(cbind(c(1,-1), c(-1,1)), 2)
```
ExpressionSet helpers

Convert object to ExpressionSet or read it from a file

Description

These functions present quick ways to create ExpressionSet objects.

Usage

```r
as.ExpressionSet(x, ...) 
## S4 method for signature 'data.frame'
as.ExpressionSet(x, annotation_cols = !sapply(x, is.double))
read.ExpressionSet(file, header = TRUE, ...)
```

Arguments

- `x` data.frame to convert to an ExpressionSet.
- `...` Additional parameters to `read.table`
- `annotation_cols` The data.frame columns used as annotations. All others are used as expressions. (Logical, character or numerical index array)
- `file` File path to read ASCII data from
- `header` Specifies if the file has a header row.

Details

They work by using all continuous (double) columns as expression data, and all others as sample annotations.

Value

an ExpressionSet object

See Also

`read.table` on which `read.ExpressionSet` is based, and `ExpressionSet`.

Examples

```r
library(Biobase)
df <- data.frame(Time = seq_len(3), # integer column
                Actb = c(0.05, 0.3, 0.8),
                Gapdh = c(0.2, 0.03, 0.1))
set <- as.ExpressionSet(df)
rownames(exprs(set)) == c("Actb", "Gapdh")
phenoData(set)$Time == 1:3
```
extraction methods

Description

Extraction methods

Usage

```r
## S4 method for signature 'DiffusionMap'
names(x)

## S4 method for signature 'DPT'
names(x)

## S4 method for signature 'DiffusionMap,character,missing'
x[[i, j, ...]]

## S4 method for signature 'DPT,character,missing'
x[[i, j, ...]]

## S4 method for signature 'DPT,index,index'
x[[i, j, ...]]

## S4 method for signature 'DiffusionMap'
x$name

## S4 method for signature 'DPT'
x$name
```

Arguments

- `x` `DiffusionMap` or `DPT` object
- `i`, `name` Name of a diffusion component ‘DCx’, ‘DPTx’, ‘Branch’ or column from the data
- `j` N/A
- `...` ignored

Value

The names or data row, see respective generics.

See Also

`Extract`, `names` for the generics. `DiffusionMap` accessor, `DiffusionMap` methods, coercions for more methods
find_dm_k

Examples

data(guo)
dm <- DiffusionMap(guo)
dm$DC1 # A diffusion component
dm$Actb # A gene expression vector
dm$num_cells # Phenotype metadata

dpt <- DPT(dm)
dm$Branch
dm$DPT1

find_dm_k  

Find a suitable k

Description

The k parameter for the k nearest neighbors used in DiffusionMap should be as big as possible while still being computationally feasible. This function approximates it depending on the size of the dataset n.

Usage

find_dm_k(n, min_k = 100L, small = 1000L, big = 10000L)

Arguments

n Number of possible neighbors (nrow(dataset) - 1)
min_k Minimum number of neighbors. Will be chosen for n ≥ big
small Number of neighbors considered small. If/where n ≤ small, n itself will be returned.
big Number of neighbors considered big. If/where n ≥ big, min_k will be returned.

Value

A vector of the same length as n that contains suitable k values for the respective n

Examples

curve(find_dm_k(n), 0, 13000, xname = 'n')
curve(find_dm_k(n) / n, 0, 13000, xname = 'n')
find_sigmas

Calculate the average dimensionality for m different gaussian kernel widths ($\sigma$).

Description

The sigma with the maximum value in average dimensionality is close to the ideal one. Increasing step number gets this nearer to the ideal one.

Usage

```r
find_sigmas(data, step_size = 0.1, steps = 10L, start = NULL,
            sample_rows = 500L, early_exit = FALSE, ..., censor_val = NULL,
            censor_range = NULL, missing_range = NULL, vars = NULL,
            verbose = TRUE)
```

Arguments

- `data`: Data set with $n$ samples. Can be a `data.frame`, `matrix` or `ExpressionSet`.
- `step_size`: Size of log-sigma steps.
- `steps`: Number of steps/calculations.
- `start`: Initial value to search from. (Optional. default: $\log_10(\min(dist(data)))$)
- `sample_rows`: Number of random rows to use for sigma estimation or vector of row indices/names to use. In the first case, only used if actually smaller than the number of available rows (Optional. default: 500)
- `early_exit`: logical. If TRUE, return if the first local maximum is found, else keep running.
- `...`: All parameter after this are optional and have to be specified by name.
- `censor_val`: Value regarded as uncertain. Either a single value or one for every dimension.
- `censor_range`: Uncertainty range for censoring. A length-2-vector of certainty range start and end. TODO: also allow $2 \times G$ matrix.
- `missing_range`: Whole data range for missing value model. Has to be specified if NAs are in the data.
- `vars`: Variables (columns) of the data to use. Specifying TRUE will select all columns (default: All floating point value columns).
- `verbose`: logical. If TRUE, show a progress bar and plot the output.

Value

Object of class `Sigmas`.

See Also

`Sigmas`, the class returned by this; `DiffusionMap`, the class this is used for.

Examples

```r
data(guo)
sigs <- find_sigmas(guo, verbose = TRUE)
DiffusionMap(guo, sigs)
```
find_tips

Find tips in a DiffusionMap object

Description
Find tips in a DiffusionMap object

Usage
find_tips(dm_or_dpt, root = random_root(dm_or_dpt))

Arguments
dm_or_dpt  A DiffusionMap or DPT object
root       Root cell index from which to find tips. (default: random)

Value
An integer vector of length 3

Examples
data(guo)
dm <- DiffusionMap(guo)
is_tip <- l_which(find_tips(dm), len = ncol(dataset(dm)))
plot(dm, col = factor(is_tip))

---

guo
Guo at al. mouse embryonic stem cell qPCR data

Description
Gene expression data of 48 genes and an annotation column $\text{num_cells}$ containing the cell stage at which the embryos were harvested.

Usage
data(guo)
data(guo_norm)

Format
An ExpressionSet with 48 features, 428 samples and 2 phenoData annotations.

Details
The data is normalized using the mean of two housekeeping genes. The difference between guo and guo_norm is the LoD being set to 10 in the former, making it usable with the censor_val parameter of DiffusionMap.
Value

an `ExpressionSet` with 48 features and 428 samples containing qPCR Ct values and a "num.cells" sample annotation.

Author(s)

Guoji Guo, Mikael Huss, Guo Qing Tong, Chaoyang Wang, Li Li Sun, Neil D. Clarke, Paul Robson
<robsonp@gis.a-star.edu.sg>

References


---

### `l_which`

#### Logical which

Description

Inverse of `which`. Converts an array of numeric or character indices to a logical index array. This function is useful if you need to perform logical operation on an index array but are only given numeric indices.

Usage

```r
l_which(idx, nms = seq_len(len), len = length(nms), useNames = TRUE)
```

Arguments

- `idx` Numeric or character indices.
- `nms` Array of names or a sequence. Required if `idx` is a character array
- `len` Length of output array. Alternative to `nms` if `idx` is numeric
- `useNames` Use the names of `nms` or `idx`

Details

Either `nms` or `len` has to be specified.

Value

Logical vector of length `len` or the same length as `nms`

Examples

```r
all(l_which(2, len = 3L) == c(FALSE, TRUE, FALSE))
all(l_which(c("a", "c"), letters[1:3]) == c(TRUE, FALSE, TRUE))
```
plot.DiffusionMap

3D or 2D plot of diffusion map

Description

If you want to plot the eigenvalues, simply plot(eigenvalues(dm)[start:end], ...)

Usage

plot.DiffusionMap(x, dims, new_dcs = NULL, col = NULL, col_by = NULL,
col_limits = NULL, col_new = "red", pal = NULL, ..., mar = NULL,
ticks = FALSE, axes = TRUE, box = FALSE, legend_main = col_by,
legend_opts = list(), interactive = FALSE,
draw_legend = !is.null(col_by) || (length(col) > 1 & !is.character(col)),
consec_col = TRUE, col_na = "grey", plot_more = function(p, ..., rescale
= NULL)

## S4 method for signature 'DiffusionMap,numeric'
plot(x, y, ...)

## S4 method for signature 'DiffusionMap,missing'
plot(x, y, ...)

Arguments

x
A DiffusionMap
dims, y
Diffusion components (eigenvectors) to plot (default: first three components; 1:3)
new_dcs
An optional matrix also containing the rows specified with y and plotted. (def-

col
Single color string or vector of discrete or categoric values to be mapped to
colors. E.g. a column of the data matrix used for creation of the diffusion map.
(default: par("fg"))
col_by
Specify a dataset(x) or phenoData(dataset(x)) column to use as color
col_limits
If col is a continuous (=double) vector, this can be overridden to map the color
range differently than from min to max (e.g. specify c(0, 1))
col_new
If new_dcs is given, it will take on this color. (default: red)
pal
Palette used to map the col vector to colors. (default: use cube_helix for continuous and palette() for discrete data)
...
Parameters passed to plot, scatterplot3d, or plot3d (if interactive == TRUE)
mar
Bottom, left, top, and right margins (default: par(mar))
ticks
logical. If TRUE, show axis ticks (default: FALSE)
axes
logical. If TRUE, draw plot axes (default: Only if ticks is TRUE)
box
logical. If TRUE, draw plot frame (default: TRUE or the same as axes if speci-

legend_main
Title of legend. (default: nothing unless col_by is given)
legend_opts
Other colorlegend options (default: empty list)
Use `plot3d` to plot instead of `scatterplot3d`?

**draw_legend**

Logical. If TRUE, draw color legend (default: TRUE if col_by is given or col is given and a vector to be mapped)

**consec_col**

If col or col_by refers to an integer column, with gaps (e.g. c(5,0,0,3)) use the palette color consecutively (e.g. c(3,1,1,2))

**col_na**

Color for NA in the data. Specify NA to hide.

**plot_more**

Function that will be called while the plot margins are temporarily changed (its p argument is the rgl or scatterplot3d instance or NULL, its rescale argument is NULL or of the shape list(from = c(a, b), to = c(c, d)))

**Details**

If you specify negative numbers as diffusion components (e.g. `plot(dm, c(-1,2)))`, then the corresponding components will be flipped.

**Value**

The return value of the underlying call is returned, i.e. a scatterplot3d or rgl object.

**Examples**

data(guo)
plot(DiffusionMap(guo))
**plot.Sigmas**

### Arguments

- **x**  
  A DPT object.

- **paths_to**  
  Numeric Branch IDs. Are used as target(s) for the path(s) to draw.

- **dcs**  
  The dimensions to use from the DiffusionMap.

- **divide**  
  If col_by = 'branch', this specifies which branches to divide. (see `branch_divide`)

- **w_width**  
  Window width for smoothing the path (see `smth.gaussian`)

- **col_by**  
  Color by 'dpt' (DPT starting at `branches[[1]]`), 'branch', or a variable of the data.

- **col_path**  
  Colors for the path or a function creating n colors

- **col_tip**  
  Color for branch tips

- **col**  
  See `plot.DiffusionMap`. This overrides col_by

- **legend_main**  
  See `plot.DiffusionMap`.

- **y, root**  
  Root branch ID. Will be used as the start of the DPT. (default: lowest branch ID) (If longer than size 1, will be interpreted as c(root, branches))

### Value

The return value of the underlying call is returned, i.e. a scatterplot3d or rgl object for 3D plots.

### Examples

```r
data(guo_norm)
dm <- DiffusionMap(guo_norm)
dpt <- DPT(dm, branching = TRUE)
plot(dpt)
plot(dpt, 2L, col_by = 'branch')
plot(dpt, 1L, 2:3, col_by = 'num_cells')
plot(dpt, col_by = 'DPT3')
```

---

**plot.Sigmas**  
**Plot Sigmas object**

### Description

Plot Sigmas object

### Usage

```r
## S4 method for signature 'Sigmas,missing'
plot(x, col = par("fg"),
     col_highlight = "#E41A1C", col_line = "#999999", type = c("b", "b"),
     pch = c(par("pch"), 4L), only_dim = FALSE, ..., xlab = NULL,
     ylab = NULL, main = "")
```
Arguments

- **x**: Sigma object to plot
- **col**: Vector of bar colors or single color for all bars
- **col_highlight**: Color for highest bar. Overrides col
- **col_line**: Color for the line and its axis
- **type**: Plot type of both lines. Can be a vector of length 2 to specify both separately (default: 'b' aka "both lines and points")
- **pch**: Point identifier for both lines. Can be a vector of length 2 to specify both separately (default: par(pch) and 4 (a ‘×’))
- **only_dim**: logical. If TRUE, only plot the derivative line
- **...**: Options passed to the call to plot
- **xlab**: X label. NULL to use default
- **ylab**: Either one y label or y labels for both plots. NULL to use both defaults, a NULL in a list of length 2 to use one default.
- **main**: Title of the plot

Value

This method plots a Sigma object to the current device and returns nothing/NULL

Examples

```r
data(guo)
sigs <- find_sigmas(guo)
plot(sigs)
```

---

**random_root**

Find a random root cell index

Description

Finds a cell that has the maximum DPT distance from a randomly selected one.

Usage

```r
random_root(dm_or_dpt)
```

Arguments

- **dm_or_dpt**: A DiffusionMap or DPT object

Value

A cell index

Examples

```r
data(guo)
dm <- DiffusionMap(guo)
random_root(dm)
```
Sigmas class

Description

Holds the information about how the sigma parameter for a DiffusionMap was obtained, and in this way provides a plotting function for the find_sigmas heuristic. You should not need to create a Sigmas object yourself. Provide sigma to DiffusionMap instead or use find_sigmas.

Usage

Sigmas(...)

## S4 method for signature 'Sigmas'
optimal_sigma(object)

## S4 method for signature 'Sigmas'
print(x)

## S4 method for signature 'Sigmas'
show(object)

Arguments

object, x  Sigmas object
...
See “Slots” below

Details

A Sigmas object is either created by find_sigmas or by specifying the sigma parameter to DiffusionMap.

In the second case, if the sigma parameter is just a number, the resulting Sigmas object has all slots except of optimal_sigma set to NULL.

Value

Sigmas creates an object of the same class
optimal_sigma retrieves the numeric value of the optimal sigma or local sigmas

Slots

log_sigmas  Vector of length m containing the log_{10} of the \sigma s
dim_norms  Vector of length m − 1 containing the average dimensionality \langle p \rangle for the respective kernel widths
optimal_sigma  Multiple local sigmas or the mean of the two global \sigma s around the highest \langle p \rangle (c(optimal_idx, optimal_idx+1L))
optimal_idx  The index of the highest \langle p \rangle.
avrd_norms  Vector of length m containing the average dimensionality for the corresponding sigma.
See Also

`find_sigmas`, the function to determine a locally optimal sigma and returning this class.

Examples

data(guo)
sigs <- find_sigmas(guo, verbose = FALSE)
optimal_sigma(sigs)
print(sigs)

Description

Update old DiffusionMaps or Sigmas to a newer version.

Usage

```r
## S4 method for signature 'DiffusionMap'
updateObject(object, ..., verbose = FALSE)
```

```r
## S4 method for signature 'Sigmas'
updateObject(object, ..., verbose = FALSE)
```

Arguments

- **object**: A DiffusionMap or Sigmas object created with an older destiny release.
- **...**: ignored.
- **verbose**: tells what is being updated.

Value

A DiffusionMap or Sigmas object that is valid when used with the current destiny release.
Index

*Topic data
  guo, 19
  [.DPT (DPT matrix methods), 13
  [,DPT,character,missing-method (extractions), 16
  [,DiffusionMap,character,missing-method (extractions), 16
  $,DPT-method (extractions), 16
  $,DiffusionMap-method (extractions), 16
  as.data.frame, 3, 12
  as.data.frame,DiffusionMap-method (coercions), 2
  as.data.frame,DPT-method (coercions), 2
  as.data.frame.DPT (coercions), 2
  as.ExpressionSet (ExpressionSet helpers), 15
  as.ExpressionSet,data.frame-method (ExpressionSet helpers), 15
  as.ExpressionSet-method (ExpressionSet helpers), 15
  as.matrix, 12
  as.matrix,DPT-method (coercions), 2
  as.matrix,DPT (coercions), 2
  branch_divide, 12, 23
  branch_divide (DPT methods), 13
  coercions, 2, 8, 11, 16
  colorlegend, 3, 21
  continuous_scale, 5
  cube_helix, 5, 21
  data.frame, 8, 11, 15, 18
  data:guo (guo), 19
  data:guo_norm (guo), 19
  dataset (destiny generics), 6
  dataset,DiffusionMap-method (DiffusionMap accessors), 7
  dataset,DPT-method (DPT methods), 13
  dataset<-(destiny generics), 6
  dataset<-,DiffusionMap-method (DiffusionMap accessors), 7
  dataset<-,DPT-method (DPT methods), 13
  destiny, 6
  destiny generics, 6
  destiny-package (destiny), 6
  DiffusionMap, 2, 3, 6, 7, 10–12, 16–19, 21, 24–26
  DiffusionMap (DiffusionMap class), 8
  DiffusionMap accessors, 3, 7, 11, 16
  DiffusionMap class, 8
  DiffusionMap methods, 3, 7, 8, 10, 16
  DiffusionMap-class (DiffusionMap class), 8
  DiffusionMap-methods, 10
  DiffusionMap-methods (DiffusionMap methods), 10
  dim.DPT (DPT matrix methods), 13
  discrete_scale, 5
  distance (destiny generics), 6
  distance,DiffusionMap-method (DiffusionMap accessors), 7
  distance<-(destiny generics), 6
  distance<-,DiffusionMap-method (DiffusionMap accessors), 7
  dm_predict, 11
  DPT, 2, 3, 9, 12, 13, 16, 19, 22–24
  DPT matrix methods, 13
  DPT methods, 13
  DPT-class (DPT), 12
  eig_decomp, 14
  eigenvalues,DiffusionMap-method (DiffusionMap accessors), 7
  eigenvalues<-(destiny generics), 6
  eigenvalues<-,DiffusionMap-method (DiffusionMap accessors), 7
  eigenvectors (destiny generics), 6
  eigenvectors,DiffusionMap-method (DiffusionMap accessors), 7
  eigenvectors<-(destiny generics), 6
  eigenvectors<-,DiffusionMap-method (DiffusionMap accessors), 7

27
ExpressionSet, 8, 11, 15, 18–20
ExpressionSet helpers, 15
Extract, 16
extractions, 3, 8, 11, 16
find_dm_k, 9, 17
find_sigmoids, 9, 10, 18, 25, 26
find_tips, 19
fortify, 3
fortify.DiffusionMap (coercions), 2
fortify.DPT (coercions), 2
find_dm_k, 9, 17
find_sigmoids, 9, 10, 18, 25, 26
find_tips, 19
fortify, 3
fortify.DiffusionMap (coercions), 2
fortify.DPT (coercions), 2
ggplot, 3
guides, 5
guo, 19
guo_norm (guo), 19
integer, 12
l_which, 20
logical, 12
matrix, 8, 11, 12, 18
names, 16
names, DiffusionMap-method (extractions), 16
names, DPT-method (extractions), 16
names, DPT (extractions), 16
ncol.DPT (DPT matrix methods), 13
nrow.DPT (DPT matrix methods), 13
numeric, 9
optimal_sigma, 9
optimal_sigma (destiny generics), 6
optimal_sigma, DiffusionMap-method (DiffusionMap accessors), 7
optimal_sigma, Sigmas-method (Sigmas class), 25
palette, 3, 21
par, 4, 21
phenoData, 19
plot, 6, 21
plot, DiffusionMap, missing-method (plot.DiffusionMap), 21
plot, DiffusionMap, numeric-method (plot.DiffusionMap), 21
plot, DPT, missing-method (plot.DPT), 22
plot, DPT, numeric-method (plot.DPT), 22
plot, Sigmas, missing-method (plot.Sigmas), 23
plot, DiffusionMap, 6, 21, 23
plot.DPT, 14, 22
plot.Sigmas, 23
plot3d, 21, 22
print, DiffusionMap-method (DiffusionMap methods), 10
print, Sigmas-method (Sigmas class), 25
print, DiffusionMap (DiffusionMap methods), 10
qplot, 3
random_root, 24
read.ExpressionSet (ExpressionSet helpers), 15
read.table, 15
scale_color_cube_helix (cube_helix), 5
scale_colour_cube_helix (cube_helix), 5
scale_fill_cube_helix (cube_helix), 5
scatterplot3d, 21, 22
show, DiffusionMap-method (DiffusionMap methods), 10
show, Sigmas-method (Sigmas class), 25
show, DiffusionMap (DiffusionMap methods), 10
Sigmas, 6, 7, 9, 18, 23, 25, 26
Sigmas (Sigmas class), 25
sigmas (destiny generics), 6
Sigmas class, 7, 25
sigmas, DiffusionMap-method (DiffusionMap accessors), 7
Sigmas-class (Sigmas class), 25
Sigmas-methods (Sigmas class), 25
sigmas<-.destiny generics, 6
sigmas<-.DiffusionMap-method (DiffusionMap accessors), 7
smth.gaussian, 23
text, 4
the plot method for the DiffusionMap, 11
updateObject, DiffusionMap-method (updateObject-method), 26
updateObject, Sigmas-method (updateObject-method), 26
updateObject-method, 26
which, 20