

# Package ‘destiny’

October 17, 2017

**Type** Package

**Title** Creates diffusion maps

**Version** 2.4.5

**Date** 2014-12-19

**Description** Create and plot diffusion maps.

**License** GPL

**Encoding** UTF-8

**Depends** R (>= 3.3.0)

**Imports** methods, graphics, grDevices, utils, stats, Matrix, Rcpp (>= 0.10.3), RcppEigen, Biobase, BiocGenerics, ggplot2, Hmisc, FNN, VIM, proxy, igraph, smoother, scales, scatterplot3d

**LinkingTo** Rcpp, RcppEigen

**SystemRequirements** C++11

**NeedsCompilation** yes

**Enhances** rgl

**Suggests** nbconvertR, testthat

**VignetteBuilder** nbconvertR

**biocViews** CellBiology, CellBasedAssays, Clustering, Software, Visualization

**Collate** 'RcppExports.R' 'aaa.r' 'accessor-generics.r' 'censoring.r' 'colorlegend.r' 'compat.r' 'cube\_helix.r' 'destiny-package.r' 's4-unions.r' 'dist-matrix-coerce.r' 'sigmas.r' 'diffusionmap.r' 'diffusionmap-methods-accession.r' 'diffusionmap-methods.r' 'plohelpers.r' 'diffusionmap-plotting.r' 'dpt-branching.r' 'dpt-helpers.r' 'dpt.r' 'dpt-methods-matrix.r' 'dpt-methods.r' 'dpt-plotting.r' 'eig\_decomp.r' 'utils.r' 'expressionset-helpers.r' 'find\_dm\_k.r' 'gene-relevance.r' 'gene-relevance-plotting.r' 'guo-data.r' 'l\_which.r' 'methods-coercion.r' 'methods-extraction.r' 'methods-update.r' 'predict.r' 'projection-dist.r' 'sigmas-plotting.r'

**RoxygenNote** 6.0.1.9000

**Author** Philipp Angerer [cre, aut],  
Laleh Haghverdi [ctb],

Maren Büttner [ctb],  
 Fabian Theis [ctb],  
 Carsten Marr [ctb],  
 Florian Büttner [ctb]

**Maintainer** Philipp Angerer <philipp.angerer@helmholtz-muenchen.de>

## R topics documented:

coercions	2
colorlegend	4
cube_helix	5
destiny	6
destiny generics	6
DiffusionMap accessors	7
DiffusionMap class	9
DiffusionMap methods	11
dm_predict	11
DPT	12
DPT matrix methods	13
DPT methods	14
eig_decomp	15
ExpressionSet helpers	15
extractions	16
find_dm_k	17
find_sigmas	18
find_tips	19
Gene Relevance	19
Gene Relevance plotting	21
guo	22
l_which	23
plot.DiffusionMap	24
plot.DPT	25
plot.Sigmas	26
projection_dist	27
random_root	28
Sigmas class	28
updateObject-method	30
<b>Index</b>	<b>31</b>

---

coercions

*Coercion methods*

---

### Description

Convert a [DiffusionMap](#) or [DPT](#) object to other classes

**Usage**

```
## 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 <a href="#">DiffusionMap</a> or <a href="#">DPT</a> 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 <a href="#">make.names</a> ) is optional.
...	Passed to <a href="#">as.data.frame</a>
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**

```
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**

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

**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**

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

scale_colour_cube_helix(..., start = 0, r = 0.4, hue = 0.8, gamma = 1,
  light = 0.85, dark = 0.15, reverse = FALSE, discrete = TRUE,
  guide = if (discrete) "legend" else "colourbar")

scale_color_cube_helix(..., start = 0, r = 0.4, hue = 0.8, gamma = 1,
  light = 0.85, dark = 0.15, reverse = FALSE, discrete = TRUE,
  guide = if (discrete) "legend" else "colourbar")

scale_fill_cube_helix(..., start = 0, r = 0.4, hue = 0.8, gamma = 1,
  light = 0.85, dark = 0.15, reverse = FALSE, discrete = TRUE,
  guide = if (discrete) "legend" else "colourbar")
```

**Arguments**

n	Number of colors to return (default: 6)
start	Hue to start helix at ( $\text{start} \in [0, 3]$ , 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 ( $\text{gamma} < 1$ ) or lighter ( $\text{gamma} > 1$ ) 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 <a href="#">discrete_scale</a> or <a href="#">continuous_scale</a>
discrete	If TRUE, return a discrete scale, if FALSE a continuous one (default: TRUE)
guide	Type of scale guide to use. See <a href="#">guides</a>

**Value**

A character vector of hex colors with length n

**Examples**

```
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	<i>Create and plot diffusion maps</i>
---------	---------------------------------------

---

**Description**

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

**Examples**

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

---

destiny generics	<i>destiny generics</i>
------------------	-------------------------

---

**Description**

destiny provides several generic methods and implements them for the [DiffusionMap](#) and [Sigmas](#) classes.

**Usage**

```
eigenvalues(object)

eigenvalues(object) <- value

eigenvectors(object)

eigenvectors(object) <- value

sigmas(object)

sigmas(object) <- value

dataset(object)

dataset(object) <- value

distance(object)
```

```
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

### Examples

```
data(guo_norm)
dm <- DiffusionMap(guo_norm)
eigenvalues(dm)
eigenvectors(dm)
sigmas(dm)
optimal_sigma(dm)
dataset(dm)
distance(dm)
```

---

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

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

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

```
## S4 method for signature 'DiffusionMap'  
eigenvalues(object)  
  
## S4 replacement method for signature 'DiffusionMap'  
eigenvalues(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](#).

### Usage

```
DiffusionMap(data = stopifnot_distmatrix(distance), sigma = "local",
  k = find_dm_k(n_samples(data, distance) - 1L), n_eigs = min(20L,
  n_samples(data, distance) - 2L), density_norm = TRUE, ...,
  distance = c("euclidean", "cosine", "rankcor"), n_local = seq(to = min(k,
  7L), length.out = min(k, 3L)), rotate = FALSE, 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 and covariates. Provide vars to select specific columns other than the default: all double value columns. If distance is a distance matrix, data has to be a <a href="#">data.frame</a> with covariates only.
sigma	Diffusion scale parameter of the Gaussian kernel. One of 'local', 'global', a <a href="#">numeric</a> global sigma or a <a href="#">Sigmas</a> object. When choosing 'global', a global sigma will be calculated using <a href="#">find_sigmas</a> . (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 <a href="#">find_dm_k</a> ).
n_eigs	Number of eigenvectors/values to return (default: 20)
density_norm	logical. If TRUE, use density normalisation
...	All parameter after this are optional and have to be specified by name
distance	Distance measurement method applied to data or a distance matrix/ <a href="#">dist</a> . Allowed measures: 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(s) determine(s) the local sigma.
rotate	logical. If TRUE, rotate the eigenvalues to get a slimmer diffusion map
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 <a href="#">DPT</a> in the returned object (default: FALSE)

**Value**

A DiffusionMap object:

**Slots**

eigenvalues	Eigenvalues ranking the eigenvectors
eigenvectors	Eigenvectors mapping the datapoints to <code>n_eigs</code> dimensions
sigmas	<a href="#">Sigmas</a> object with either information about the <a href="#">find_sigmas</a> heuristic run or just local or <a href="#">optimal_sigma</a> .
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 <code>k</code> parameter for kNN
n_local	The <code>n_localth</code> nearest neighbor(s) is/are used to determine local kernel density
density_norm	Was density normalization used?
rotate	Were the eigenvectors rotated?
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)

covars <- data.frame(covar1 = letters[1:100])
dists <- dist(matrix(rnorm(100*10), 100))
DiffusionMap(covars, distance = dists)
```

---

DiffusionMap methods    *DiffusionMap methods*

---

### 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

### See Also

[DiffusionMap accessors, extractions, coercions](#) for more methods

### Examples

```
data(guo)  
dm <- DiffusionMap(guo)  
print(dm)  
show(dm)
```

---

dm\_predict      *Predict new data points using an existing DiffusionMap. The resulting matrix can be used in [the plot method for the DiffusionMap](#)*

---

### Description

Predict new data points using an existing DiffusionMap. The resulting matrix can be used in [the plot method for the DiffusionMap](#)

### Usage

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

**Arguments**

<code>dm</code>	A <a href="#">DiffusionMap</a> object.
<code>new_data</code>	New data points to project into the diffusion map. Can be a <a href="#">matrix</a> , <a href="#">data.frame</a> , or an <a href="#">ExpressionSet</a> .
<code>...</code>	Passed to <a href="#">proxy::dist</a> ( <code>new_data</code> , <code>data</code> , <code>dm@distance</code> , ...).
<code>verbose</code>	Show progress messages?

**Value**

A  $nrow(new\_data) \times ncol(eigenvectors(dif))$  matrix of projected diffusion components for the new data.

**Examples**

```
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**

```
DPT(dm, tips = random_root(dm), ..., w_width = 0.1)
```

**Arguments**

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

**Details**

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

**Value**

A DPT object:

**Slots**

branch **matrix** (of **integer**) recursive branch labels for each cell (row); NA for undecided. Use **branch\_divide** to modify this.

tips **matrix** (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**

```
data(guo_norm)
dm <- DiffusionMap(guo_norm)
dpt <- DPT(dm, branching = TRUE)
str(dpt)
```

---

DPT matrix methods      *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,index,index'
x[[i, j, ...]]

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

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

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

**Arguments**

<code>x</code>	DPT object.
<code>i, j</code>	Numeric or logical index.
<code>...</code>	ignored
<code>drop</code>	If <b>TRUE</b> , coerce result to a vector if it would otherwise have <code>1 %in% dim(result)</code> .

**See Also**

[as.matrix.DPT](#)

---

DPT methods

*DPT methods*

---

**Description**

Methods for the [DPT](#) class. `branch_divide` subdivides branches for plotting (see the examples).

**Usage**

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

tips(dpt)

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

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

**Arguments**

<code>dpt, object</code>	DPT object
<code>divide</code>	Vector of branch numbers to use for division
<code>value</code>	Value of slot to set

**Value**

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

**See Also**

[plot.DPT](#) uses `branch_divide` for its `divide` argument.

**Examples**

```
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	<i>Fast eigen decomposition using ARPACK</i>
------------	--

---

**Description**

Fast eigen decomposition using ARPACK

**Usage**

```
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**

```
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 way to create [ExpressionSet](#) objects.

**Usage**

```
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	<a href="#">data.frame</a> to convert to an <a href="#">ExpressionSet</a> .
...	Additional parameters to <a href="#">read.table</a>
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**

```
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
```

---

extractions

*Extraction methods*

---

**Description**

Extraction methods

**Usage**

```
## 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 'DiffusionMap'
x$name

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



**Arguments**

x	<a href="#">DiffusionMap</a> or <a href="#">DPT</a> 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](#) accessors, [DiffusionMap](#) methods, [coercions](#) for more methods

**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 $n \geq big$
small	Number of neighbors considered small. If/where $n \leq small$ , n itself will be returned.
big	Number of neighbors considered big. If/where $n \geq big$ , 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	<i>Calculate the average dimensionality for <math>m</math> different gaussian kernel widths (<math>\sigma</math>).</i>
-------------	--

---

**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**

```
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 <a href="#">data.frame</a> , <a href="#">matrix</a> or <a href="#">ExpressionSet</a> .
step_size	Size of log-sigma steps
steps	Number of steps/calculations
start	Initial value to search from. (Optional. default: $\log_{10}(\min(\text{dist}(\text{data})))$ )
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**

```
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))
```

---

 Gene Relevance

*Gene relevances for entire data set*


---

**Description**

The relevance map is cached insided of the [DiffusionMap](#).

**Usage**

```
gene_relevance(coords, exprs, ..., k = 20, dims = 1:2, verbose = FALSE)

## S4 method for signature 'DiffusionMap,missing'
gene_relevance(coords, exprs, ..., k = 20,
  dims = 1:2, verbose = FALSE)

## S4 method for signature 'matrix,matrix'
gene_relevance(coords, exprs, ..., k = 20,
  dims = 1:2, verbose = FALSE)
```

**Arguments**

coords	A <a href="#">DiffusionMap</a> object or a cells × dims <a href="#">matrix</a> .
exprs	An cells × genes <a href="#">matrix</a> . Only provide if coords is no <a href="#">DiffusionMap</a> .
...	Ignored.
k	Number of nearest neighbors to use
dims	Index into columns of coord
verbose	If TRUE, log additional info to the console

**Value**

A GeneRelevance object:

**Slots**

coords A cells × dims [matrix](#) of coordinates (e.g. diffusion components), reduced to the dimensions passed as dims.

exprs A cells × genes matrix of expressions.

partials Array of partial derivatives wrt to considered dimensions in reduced space. (genes × cells × dimensions)

partials\_norm Matrix with norm of aforementioned derivatives. (n\_genes × cells)

nn\_index Matrix of k nearest neighbor indices. (cells × k)

dims Column index for plotted dimensions. Can [character](#), [numeric](#) or [logical](#).

**See Also**

[Gene Relevance plotting](#): [plot\\_gradient\\_map](#)/[plot\\_gene\\_relevance](#)

**Examples**

```
data(guo_norm)
dm <- DiffusionMap(guo_norm)
gr <- gene_relevance(dm)
# now plot them!
```

---

Gene Relevance plotting

*Plot gene relevance or gradient map*

---

## Description

`plot(gene_relevance, 'Gene')` plots the gradient map of this gene, `plot(gene_relevance)` a relevance map of a selection of genes. Alternatively, you can use `plot_gradient_map` or `plot_gene_relevance` on a [GeneRelevance](#) or [DiffusionMap](#) object, or with two matrices.

## Usage

```
## S4 method for signature 'GeneRelevance,character'
plot(x, y, ...)

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

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

plot_gradient_map(coords, exprs, ..., gene, pal = cube_helix)

## S4 method for signature 'matrix,matrix'
plot_gradient_map(coords, exprs, ..., gene,
  pal = cube_helix)

## S4 method for signature 'DiffusionMap,missing'
plot_gradient_map(coords, exprs, ..., gene,
  pal = cube_helix)

## S4 method for signature 'GeneRelevance,missing'
plot_gradient_map(coords, exprs, ..., gene,
  pal = cube_helix)

plot_gene_relevance(coords, exprs, ..., iter_smooth = 2L, genes = 5L,
  pal = palette())

## S4 method for signature 'matrix,matrix'
plot_gene_relevance(coords, exprs, ...,
  iter_smooth = 2L, genes = 5L, pal = palette())

## S4 method for signature 'DiffusionMap,missing'
plot_gene_relevance(coords, exprs, ...,
  iter_smooth = 2L, genes = 5L, pal = palette())

## S4 method for signature 'GeneRelevance,missing'
plot_gene_relevance(coords, exprs, ...,
  iter_smooth = 2L, genes = 5L, pal = palette())
```

**Arguments**

x	<a href="#">GeneRelevance</a> object.
y, gene	Gene name or index to create gradient map for. (a number or string)
...	Passed to <code>plot_gradient_map/plot_gene_relevance</code>
coords	A <a href="#">DiffusionMap/GeneRelevance</a> object or a cells $\times$ dims <a href="#">matrix</a> .
exprs	An cells $\times$ genes <a href="#">matrix</a> . Only provide if coords is a matrix.
pal	Palette. Either A colormap function or a list of colors.
iter_smooth	Number of label smoothing iterations to perform on relevance map. The higher the more homogenous and the less local structure.
genes	Genes to based relevance map on or number of genes to use. (vector of strings or one number) You can also pass an index into the gene names. (vector of numbers or logicals with length > 1)

**Value**

ggplot2 plot, when plotting a relevance map with a list member \$ids containing the IDs used.

**Examples**

```
data(guo_norm)
dm <- DiffusionMap(guo_norm)
gr <- gene_relevance(dm)
plot(gr) # or plot_gene_relevance(dm)
plot(gr, 'Fgf4') # or plot_gradient_map(dm, 'Fgf4')

guo_norm_mat <- t(Biobase::exprs(guo_norm))
pca <- prcomp(guo_norm_mat)$x
plot_gene_relevance(pca, guo_norm_mat)
plot_gradient_map(pca, guo_norm_mat, gene = 'Fgf4')
```

---

guo

*Guo at al. mouse embryonic stem cell qPCR data*

---

**Description**

Gene expression data of 48 genes and an annotation column \$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**

<http://www.sciencedirect.com/science/article/pii/S1534580710001103>

---

l_which	<i>Logical which</i>
---------	----------------------

---

**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**

```
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**

```
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 = 1:3, new_dcs = NULL, new_data = NULL,
  col = NULL, col_by = NULL, col_limits = NULL, col_new = "red",
  pal = NULL, pal_new = 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) NULL)
```

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

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

## Arguments

<code>x</code>	A <a href="#">DiffusionMap</a>
<code>dims, y</code>	Diffusion components (eigenvectors) to plot (default: first three components; 1:3)
<code>new_dcs</code>	An optional matrix also containing the rows specified with <code>y</code> and plotted. (default: no more points)
<code>new_data</code>	A data set in the same format as <code>x</code> that is used to create <code>new_dcs &lt;- dm_predict(dif, new_data)</code>
<code>col</code>	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: <code>par('fg')</code> )
<code>col_by</code>	Specify a <code>dataset(x)</code> or <code>phenoData(dataset(x))</code> column to use as color
<code>col_limits</code>	If <code>col</code> is a continuous (=double) vector, this can be overridden to map the color range differently than from min to max (e.g. specify <code>c(0, 1)</code> )
<code>col_new</code>	If <code>new_dcs</code> is given, it will take on this color. A vector is also possible. (default: red)
<code>pal</code>	Palette used to map the <code>col</code> vector to colors. (default: use <code>cube_helix</code> for continuous and <code>palette()</code> for discrete data)
<code>pal_new</code>	Palette used to map the <code>col_new</code> vector to colors. (default: see <code>pal</code> argument)
<code>...</code>	Parameters passed to <code>plot</code> , <code>scatterplot3d</code> , or <code>plot3d</code> (if <code>interactive == TRUE</code> )
<code>mar</code>	Bottom, left, top, and right margins (default: <code>par(mar)</code> )
<code>ticks</code>	logical. If TRUE, show axis ticks (default: FALSE)
<code>axes</code>	logical. If TRUE, draw plot axes (default: Only if <code>ticks</code> is TRUE)



box	logical. If TRUE, draw plot frame (default: TRUE or the same as axes if specified)
legend_main	Title of legend. (default: nothing unless col_by is given)
legend_opts	Other <a href="#">colorlegend</a> options (default: empty list)
interactive	Use <a href="#">plot3d</a> to plot instead of <a href="#">scatterplot3d</a> ?
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.DPT

*Plot DPT*

---

### Description

Plots diffusion components from a Diffusion Map and the accompanying Diffusion Pseudo Time ([DPT](#))

### Usage

```
plot.DPT(x, root = NULL, paths_to = integer(0L), dcs = 1:2,
  divide = integer(0L), w_width = 0.1, col_by = "dpt",
  col_path = rev(palette()), col_tip = "red", ..., col = NULL,
  legend_main = col_by)
```

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

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

**Arguments**

x	A <a href="#">DPT</a> 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 <a href="#">branch_divide</a> )
w_width	Window width for smoothing the path (see <a href="#">smth.gaussian</a> )
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
...	Graphical parameters supplied to <a href="#">plot.DiffusionMap</a>
col	See <a href="#">plot.DiffusionMap</a> . This overrides col_by
legend_main	See <a href="#">plot.DiffusionMap</a> .
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**

```
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**

```
## 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	Sigmas 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 'x'))
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**

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

---

projection_dist	<i>Projection distance</i>
-----------------	----------------------------

---

**Description**

Projection distance

**Usage**

```
projection_dist(dm, new_dcs = NULL, ..., new_data, verbose = FALSE)
```

**Arguments**

dm	A <a href="#">DiffusionMap</a> object.
new_dcs	Diffusion component matrix of which to calculate the distance to the data.
...	Passed to <a href="#">proxy::dist</a> if new_data was passed.
new_data	New data points to project into the diffusion map. Can be a <a href="#">matrix</a> , <a href="#">data.frame</a> , or an <a href="#">ExpressionSet</a> .
verbose	If <a href="#">TRUE</a> , log additional info to the console.

**Examples**

```

data(guo_norm)
g2_32 <- guo_norm[, guo_norm$num_cells < 64]
g64 <- guo_norm[, guo_norm$num_cells == 64]
dm <- DiffusionMap(g2_32)
d <- projection_dist(dm, new_data = g64)

```

---

random_root	<i>Find a random root cell index</i>
-------------	--------------------------------------

---

**Description**

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

**Usage**

```
random_root(dm_or_dpt)
```

**Arguments**

dm\_or\_dpt      A [DiffusionMap](#) or [DPT](#) object

**Value**

A cell index

**Examples**

```

data(guo)
dm <- DiffusionMap(guo)
random_root(dm)

```

---

Sigmas class	<i>Sigmas Object</i>
--------------	----------------------

---

**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)

```

---

updateObject-method     *Update old [DiffusionMaps](#) or [Sigmas](#) to a newer version*

---

### Description

Update old [DiffusionMaps](#) or [Sigmas](#) to a newer version

### Usage

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

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

### Arguments

object	A <a href="#">DiffusionMap</a> or <a href="#">Sigmas</a> 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, [22](#)
  - [,DPT,index,index,logicalOrMissing-method (DPT matrix methods), [13](#)
  - [,DPT,index,missing,logicalOrMissing-method (DPT matrix methods), [13](#)
  - [,DPT,missing,index,logicalOrMissing-method (DPT matrix methods), [13](#)
  - [,DPT,missing,missing,logicalOrMissing-method (DPT matrix methods), [13](#)
  - [.DPT (DPT matrix methods), [13](#)
  - [[,DPT,character,missing-method (extractions), [16](#)
  - [[,DPT,index,index-method (DPT matrix methods), [13](#)
  - [[,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.DiffusionMap (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, [14](#)
- as.matrix.DPT (coercions), [2](#)
- branch\_divide, [13](#), [26](#)
- branch\_divide (DPT methods), [14](#)
- character, [20](#)
- coercions, [2](#), [8](#), [11](#), [17](#)
- colorlegend, [4](#), [25](#)
- continuous\_scale, [5](#)
- cube\_helix, [5](#), [24](#)
- data.frame, [9](#), [12](#), [15](#), [18](#), [27](#)
- data:guo (guo), [22](#)
- data:guo\_norm (guo), [22](#)
- dataset (destiny generics), [6](#)
- dataset,DiffusionMap-method (DiffusionMap accessors), [7](#)
- dataset,DPT-method (DPT methods), [14](#)
- dataset<- (destiny generics), [6](#)
- dataset<- ,DiffusionMap-method (DiffusionMap accessors), [7](#)
- dataset<- ,DPT-method (DPT methods), [14](#)
- destiny, [6](#)
- destiny generics, [6](#)
- destiny-package (destiny), [6](#)
- DiffusionMap, [2](#), [3](#), [6](#), [7](#), [11–13](#), [17](#), [19–24](#), [27–30](#)
- DiffusionMap (DiffusionMap class), [9](#)
- DiffusionMap accessors, [3](#), [7](#), [11](#), [17](#)
- DiffusionMap class, [9](#)
- DiffusionMap methods, [3](#), [7](#), [8](#), [11](#), [17](#)
- DiffusionMap-class (DiffusionMap class), [9](#)
- DiffusionMap-methods, [10](#)
- DiffusionMap-methods (DiffusionMap methods), [11](#)
- dim,DPT-method (DPT matrix methods), [13](#)
- dim.DPT (DPT matrix methods), [13](#)
- discrete\_scale, [5](#)
- dist, [9](#)
- distance (destiny generics), [6](#)
- distance,DiffusionMap-method (DiffusionMap accessors), [7](#)
- distance<- (destiny generics), [6](#)
- distance<- ,DiffusionMap-method (DiffusionMap accessors), [7](#)
- dm\_predict, [11](#), [24](#)
- DPT, [2](#), [3](#), [10](#), [12](#), [14](#), [17](#), [19](#), [25](#), [26](#), [28](#)
- DPT matrix methods, [13](#)
- DPT methods, [14](#)
- DPT-class (DPT), [12](#)

- eig\_decomp, 15
- eigenvalues (destiny generics), 6
- 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
- ExpressionSet, 9, 12, 15, 16, 18, 22, 23, 27
- ExpressionSet helpers, 15
- Extract, 17
- extractions, 3, 8, 11, 16
  
- find\_dm\_k, 9, 17
- find\_sigmas, 9, 10, 18, 28, 29
- find\_tips, 19
- fortify, 3
- fortify.DiffusionMap (coercions), 2
- fortify.DPT (coercions), 2
  
- Gene Relevance, 19
- Gene Relevance plotting, 20, 21
- gene\_relevance (Gene Relevance), 19
- gene\_relevance, DiffusionMap, missing-method (Gene Relevance), 19
- gene\_relevance, matrix, matrix-method (Gene Relevance), 19
- GeneRelevance, 21, 22
- GeneRelevance (Gene Relevance), 19
- GeneRelevance-class (Gene Relevance), 19
- ggplot, 3
- guides, 5
- guo, 22
- guo\_norm (guo), 22
  
- integer, 13
  
- l\_which, 23
- logical, 13, 14, 20
  
- matrix, 9, 12, 13, 18, 20, 22, 27
  
- names, 17
- names, DiffusionMap-method (extractions), 16
- names, DPT-method (extractions), 16
- names.DiffusionMap (extractions), 16
- names.DPT (extractions), 16
- ncol, DPT-method (DPT matrix methods), 13
- ncol.DPT (DPT matrix methods), 13
- nrow, DPT-method (DPT matrix methods), 13
- nrow.DPT (DPT matrix methods), 13
- Numeric, 14
- numeric, 9, 20
  
- optimal\_sigma, 10
- optimal\_sigma (destiny generics), 6
- optimal\_sigma, DiffusionMap-method (DiffusionMap accessors), 7
- optimal\_sigma, Sigmas-method (Sigmas class), 28
  
- palette, 4, 24
- par, 4, 24
- phenoData, 22
- plot, 6, 24
- plot, DiffusionMap, missing-method (plot.DiffusionMap), 24
- plot, DiffusionMap, numeric-method (plot.DiffusionMap), 24
- plot, DPT, missing-method (plot.DPT), 25
- plot, DPT, numeric-method (plot.DPT), 25
- plot, GeneRelevance, character-method (Gene Relevance plotting), 21
- plot, GeneRelevance, missing-method (Gene Relevance plotting), 21
- plot, GeneRelevance, numeric-method (Gene Relevance plotting), 21
- plot, Sigmas, missing-method (plot.Sigmas), 26
- plot.DiffusionMap, 6, 24, 26
- plot.DPT, 14, 25
- plot.GeneRelevance (Gene Relevance plotting), 21
- plot.Sigmas, 26
- plot3d, 24, 25
- plot\_gene\_relevance (Gene Relevance plotting), 21
- plot\_gene\_relevance, DiffusionMap, missing-method (Gene Relevance plotting), 21
- plot\_gene\_relevance, GeneRelevance, missing-method (Gene Relevance plotting), 21
- plot\_gene\_relevance, matrix, matrix-method (Gene Relevance plotting), 21
- plot\_gradient\_map (Gene Relevance plotting), 21
- plot\_gradient\_map, DiffusionMap, missing-method (Gene Relevance plotting), 21
- plot\_gradient\_map, GeneRelevance, missing-method (Gene Relevance plotting), 21
- plot\_gradient\_map, matrix, matrix-method (Gene Relevance plotting), 21



print,DiffusionMap-method  
    (DiffusionMap methods), 11  
print,Sigmas-method (Sigmas class), 28  
print.DiffusionMap (DiffusionMap  
    methods), 11  
projection\_dist, 27  
proxy::dist, 12, 27

qplot, 3

random\_root, 28  
read.ExpressionSet (ExpressionSet  
    helpers), 15  
read.table, 15, 16

scale\_color\_cube\_helix (cube\_helix), 5  
scale\_colour\_cube\_helix (cube\_helix), 5  
scale\_fill\_cube\_helix (cube\_helix), 5  
scatterplot3d, 24, 25  
show,DiffusionMap-method (DiffusionMap  
    methods), 11  
show,Sigmas-method (Sigmas class), 28  
show.DiffusionMap (DiffusionMap  
    methods), 11  
Sigmas, 6, 7, 9, 10, 18, 19, 26, 29, 30  
Sigmas (Sigmas class), 28  
sigmas (destiny generics), 6  
Sigmas class, 7, 28  
sigmas,DiffusionMap-method  
    (DiffusionMap accessors), 7  
Sigmas-class (Sigmas class), 28  
Sigmas-methods (Sigmas class), 28  
sigmas<- (destiny generics), 6  
sigmas<-,DiffusionMap-method  
    (DiffusionMap accessors), 7  
smth.gaussian, 26

text, 4  
the plot method for the DiffusionMap,  
    11  
tips (DPT methods), 14  
TRUE, 14, 27

updateObject,DiffusionMap-method  
    (updateObject-method), 30  
updateObject,Sigmas-method  
    (updateObject-method), 30  
updateObject-method, 30

which, 23