

Package ‘destiny’

October 12, 2016

Type Package

Title Creates diffusion maps

Version 1.2.1

Date 2014-12-19

Description Create and plot diffusion maps.

License GPL

Encoding UTF-8

Depends R (>= 3.2.0), methods, Biobase

Imports graphics, Rcpp (>= 0.10.3), RcppEigen, BiocGenerics, Matrix,
Hmisc, FNN, VIM, proxy, igraph, scatterplot3d

LinkingTo Rcpp, RcppEigen

SystemRequirements C++11

NeedsCompilation yes

Enhances rgl

Suggests RColorBrewer, ggplot2, nbconvertR

VignetteBuilder nbconvertR

biocViews CellBiology, CellBasedAssays, Clustering, Software,
Visualization

Collate 'RcppExports.R' 'censoring.r' 'destiny-package.r'
's4-null-unions.r' 'dist-matrix-coerce.r' 'sigmas.r'
'diffusionmap.r' 'diffusionmap-coercions.r'
'diffusionmap-methods.r' 'plotheelpers.r'
'diffusionmap-plotting.r' 'utils.r' 'expressionset-helpers.r'
'guo-data.r' 'predict.r' 'sigmas-plotting.r' 'update-methods.r'

RoxygenNote 5.0.1

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:

colorlegend	2
cube.helix	3
destiny	4
DiffusionMap class	5
DiffusionMap coercions	6
DiffusionMap methods	7
dm.predict	9
ExpressionSet helpers	9
find.dm.k	11
find.sigmas	11
guo	12
IWhich	13
plot.DiffusionMap	14
plot.Sigmas	15
Sigmas class	16
updateObject,DiffusionMap-method	18
Index	19

 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, main.cex = 1, main.col = par("fg"),
  lab.col = par("fg"), steps = 5, color.steps = 100, digit = 2,
  left = FALSE, ...)
```

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>main.cex</code>	Size of legend title font

main.col	Color of legend title
lab.col	Color of tick or category labels
steps	Number of labels in case of a continuous axis
color.steps	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 <code>text</code> call used for labels

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)
```

Arguments

n	Number of colors to return (default: 6)
start	Hue to start helix at (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 (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)

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 <- colorRamp(cube.helix(12, r = 3))
r <- runif(100)
plot(1:100, r, col = rgb(cr(r), max = 255), 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).

Details

The `sigma` parameter for [DiffusionMap](#) can be determined using [find.sigmas](#) if your dataset is small enough.

Examples

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

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, sigma = NULL, k = find.dm.k(nrow(data) - 1L),
  n.eigs = min(20L, nrow(data) - 2L), density.norm = TRUE, ...,
  distance = c("euclidean", "cosine", "rankcor"), censor.val = NULL,
  censor.range = NULL, missing.range = NULL, vars = NULL,
  verbose = !is.null(censor.range))
```

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. Either a number or a Sigmas object. (Optional. default: will be calculated using find.sigmas) 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 $n - 1$) NULL or NA are also interpreted as $n - 1$ L.
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. Euclidean distance (default), cosine distance ($1 - \text{corr}(c_1, c_2)$) or rank correlation distance ($1 - \text{corr}(\text{rank}(c_1), \text{rank}(c_2))$).
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)

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 [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
`phi0` Density vector of normalized transition probability matrix
`d` Density vector of transition probability matrix
`k` The k parameter for kNN
`density.norm` Was density normalization used?
`distance` Distance measurement method used.
`sensor.val` Censoring value
`sensor.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 sigma parameter

Examples

```
data(guo)
DiffusionMap(guo)
DiffusionMap(guo, 13, censor.val = 15, censor.range = c(15, 40), verbose = TRUE)
```

DiffusionMap coercions

DiffusionMap coercions

Description

Convert a diffusionmap to other classes

Usage

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

Arguments

x, model	A DiffusionMap
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 <code>make.names</code>) is optional.
...	Other parameters
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](#)

Examples

```
data(guo)  
dm <- DiffusionMap(guo)  
df <- as.data.frame(dm)  
df$DC1 # A diffusion component  
df$Actb # A gene expression
```

DiffusionMap methods *DiffusionMap methods*

Description

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

Usage

```
eigenvalues(dm)

eigenvalues(dm) <- value

eigenvectors(dm)

eigenvectors(dm) <- value

sigmas(dm)

sigmas(dm) <- value

dataset(dm)

dataset(dm) <- value

## S4 method for signature 'DiffusionMap'
optimal.sigma(object)

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

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

Arguments

```
dm, x, object  A DiffusionMap
value          Vector of eigenvalues or matrix of eigenvectors to get/set
```

Value

The assigned or retrieved value

See Also

[DiffusionMap](#)

Examples

```
data(guo)
dm <- DiffusionMap(guo)
eigenvalues(dm)
eigenvectors(dm)
sigmas(dm)
dataset(dm)
optimal.sigma(dm)
```

dm.predict	<i>Predict new data points using an existing DiffusionMap. The resulting matrix can be used in the plot method for the DiffusionMap</i>
------------	---

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)
```

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 .

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)
```

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	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

```
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
```

find.dm.k	<i>Find a suitable k</i>
-----------	--------------------------

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 m different gaussian kernel widths (σ).</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

<code>data</code>	Data set with n samples. Can be a data.frame , matrix or ExpressionSet .
<code>step.size</code>	Size of log-sigma steps
<code>steps</code>	Number of steps/calculations
<code>start</code>	Initial value to search from. (Optional. default: $\log_1 0(\min(\text{dist}(\text{data})))$)
<code>sample.rows</code>	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)
<code>early.exit</code>	logical. If TRUE, return if the first local maximum is found, else keep running
<code>...</code>	All parameter after this are optional and have to be specified by name
<code>censor.val</code>	Value regarded as uncertain. Either a single value or one for every dimension
<code>censor.range</code>	Uncertainty range for censoring. A length-2-vector of certainty range start and end. TODO: also allow $2 \times G$ matrix
<code>missing.range</code>	Whole data range for missing value model. Has to be specified if NAs are in the data
<code>vars</code>	Variables (columns) of the data to use. Specifying TRUE will select all columns (default: All floating point value columns)
<code>verbose</code>	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)
```

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 `sensor.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>

lWhich

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

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

Arguments

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

Details

Either nms or len has to be specified.

Value

Logical vector of length len or the same length as nms

Examples

```
all(1Which(2, len = 3) == c(FALSE, TRUE, FALSE))
all(1Which(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 = palette(), ..., mar = NULL,
  ticks = FALSE, axes = TRUE, box = FALSE, legend.main = col.by,
  legend.opts = list(), interactive = FALSE, draw.legend = !is.null(col)
  && length(col) > 1 && !is.character(col), consec.col = TRUE)
```

```
## 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. (default: no more points)
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: <code>par('fg')</code>)
col.by	Specify a dataset(x) or phenoData(dataset(x)) 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. (default: red)
<code>pal</code>	Palette used to map the <code>col</code> vector to colors (default: <code>palette()</code>)
<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>tick.marks</code> is TRUE)
<code>box</code>	logical. If TRUE, draw plot frame (default: TRUE or the same as <code>axes</code> if specified)
<code>legend.main</code>	Title of legend. (default: nothing unless <code>col.by</code> is given)
<code>legend.opts</code>	Other <code>colorlegend</code> options (default: empty list)
<code>interactive</code>	Use <code>plot3d</code> to plot instead of <code>scatterplot3d</code> ?
<code>draw.legend</code>	logical. If TRUE, draw color legend (default: TRUE if <code>col</code> is given and a vector to be mapped)
<code>consec.col</code>	If <code>col</code> or <code>col.by</code> refers to an integer column, with gaps (e.g. <code>c(5, 0, 0, 3)</code>) use the palette color consecutively (e.g. <code>c(3, 1, 1, 2)</code>)

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`

Plot Sigmas object

Description

Plot `Sigmas` object

Usage

```
## S4 method for signature 'Sigmas,missing'
plot(x, col = par("fg"),
     highlight.col = "#E41A1C", line.col = "#999999", type = c("b", "b"),
     pch = c(par("pch"), 4L), only.dim = FALSE, ..., xlab = NULL,
     ylab = NULL, main = "")
```

Arguments

<code>x</code>	Sigmas object to plot
<code>col</code>	Vector of bar colors or single color for all bars
<code>highlight.col</code>	Color for highest bar. Overrides <code>col</code>
<code>line.col</code>	Color for the line and its axis
<code>type</code>	Plot type of both lines. Can be a vector of length 2 to specify both separately (default: 'b' aka "both lines and points")
<code>pch</code>	Point identifier for both lines. Can be a vector of length 2 to specify both separately (default: <code>par(pch)</code> and 4 (a 'x'))
<code>only.dim</code>	logical. If TRUE, only plot the derivative line
<code>...</code>	Options passed to the call to plot
<code>xlab</code>	X label. NULL to use default
<code>ylab</code>	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.
<code>main</code>	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)
```

Sigmas class

Sigmas Object

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(...)

optimal.sigma(object)

## 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 [Diffusion-Map](#).

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

Slots

`log.sigmas` Vector of length m containing the \log_{10} of the σ s
`dim.norms` Vector of length $m - 1$ containing the average dimensionality $\langle p \rangle$ for the respective kernel widths
`optimal.sigma` Mean of the two σ 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,DiffusionMap-method

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

Description

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

Usage

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

Arguments

object	A DiffusionMap object created with an older destiny release
...	ignored
verbose	tells what is being updated

Value

A [DiffusionMap](#) object that is valid when used with the current destiny release

Index

*Topic **data**
guo, 12

as.data.frame, DiffusionMap-method
(DiffusionMap coercions), 6
as.ExpressionSet (ExpressionSet
helpers), 9
as.ExpressionSet, data.frame-method
(ExpressionSet helpers), 9
as.ExpressionSet-method (ExpressionSet
helpers), 9

colorlegend, 2, 15
cube.helix, 3

data.frame, 5, 9, 10, 12
data:guo (guo), 12
dataset (DiffusionMap methods), 7
dataset<- (DiffusionMap methods), 7
destiny, 4
destiny-package (destiny), 4
DiffusionMap, 4, 7–9, 11–14, 16–18
DiffusionMap (DiffusionMap class), 5
DiffusionMap class, 5
DiffusionMap coercions, 6
DiffusionMap methods, 7
DiffusionMap-class (DiffusionMap
class), 5
DiffusionMap-coercions (DiffusionMap
coercions), 6
DiffusionMap-methods, 6
DiffusionMap-methods (DiffusionMap
methods), 7
dm.predict, 9

eigenvalues (DiffusionMap methods), 7
eigenvalues<- (DiffusionMap methods), 7
eigenvectors (DiffusionMap methods), 7
eigenvectors<- (DiffusionMap methods), 7
ExpressionSet, 5, 9, 10, 12, 13

ExpressionSet helpers, 9

find.dm.k, 11
find.sigmas, 4–6, 11, 16, 17
fortify, 7
fortify.DiffusionMap (DiffusionMap
coercions), 6

ggplot, 7
guo, 12

lWhich, 13

matrix, 5, 9, 12

optimal.sigma, 6
optimal.sigma (Sigmas class), 16
optimal.sigma, DiffusionMap-method
(DiffusionMap methods), 7
optimal.sigma, Sigmas-method (Sigmas
class), 16

palette, 2, 15
par, 14
phenoData, 13
plot, 4, 15
plot, DiffusionMap, missing-method
(plot.DiffusionMap), 14
plot, DiffusionMap, numeric-method
(plot.DiffusionMap), 14
plot, Sigmas, missing-method
(plot.Sigmas), 15
plot.DiffusionMap, 4, 14
plot.Sigmas, 15
plot3d, 15
print, DiffusionMap-method
(DiffusionMap methods), 7
print, Sigmas-method (Sigmas class), 16

qplot, 7

read.ExpressionSet (ExpressionSet
helpers), 9
read.table, 10

scatterplot3d, 15
show, DiffusionMap-method (DiffusionMap
methods), 7
show, Sigmas-method (Sigmas class), 16
Sigmas, 5, 6, 12, 15, 17
Sigmas (Sigmas class), 16
sigmas (DiffusionMap methods), 7
Sigmas class, 16
Sigmas-class (Sigmas class), 16
Sigmas-methods (Sigmas class), 16
sigmas<- (DiffusionMap methods), 7

text, 3
the plot method for the DiffusionMap, 9

updateObject, DiffusionMap-method, 18

values (DiffusionMap methods), 7
values<- (DiffusionMap methods), 7
vectors (DiffusionMap methods), 7
vectors<- (DiffusionMap methods), 7

which, 13