Package 'destiny'

April 23, 2016

Type Package

Title Creates diffusion maps

Version 1.0.0

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, 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' 'plothelpers.r' 'diffusionmap-plotting.r' 'utils.r' 'expressionset-helpers.r' 'guo-data.r' 'predict.r' 'sigmas-plotting.r'

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` *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

`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
`main.cex`	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 text 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)
```

---

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

Description

Gene expression data of 48 genes and an annotation column `$division` containing the number of divisions after which the cells were harvested.

Usage

```
data(guo)
```

Format

A data frame with 492 rows and 49 variables, the first 48 of which are Ct values for gene expression data

Details

The data is normalized using the mean of two housekeeping genes and the LoD is set to 15.

Value

a 429×49 `data.frame` with the first 48 columns being qPCR Ct values and the last one the "divisions" 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

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"), censor.val = NULL,
  censor.range = NULL, missing.range = NULL, vars = NULL,
  verbose = !is.null(censor.range), .debug.env = NULL)
```

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 betweeen 100 and $n - 1$) NULL or NA are also interpreted as n - 1L.
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) or cosine distance ($1 - corr(c_1, c_2)$).
`censor.val`	Value regarded as uncertain. Either a single value or one for every dimension (Optional, default: CENSOR.VAL)
`censor.range`	Uncertainity 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)
`.debug.env`	If supplied, the rotated transition probaility matrix `M` and the scaling rotation `D.rot` is stored in it prior to eigen decomposition.

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.

`d` Density vector of transition probability matrix

`k` The k parameter for kNN

`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 `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 make.names) 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` *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)
```

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$divisions != 5L, ]
g2 <- guo[guo$divisions == 5L, ]
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 waya 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 *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 *Calculate the average dimensionality for m different gaussian kernel widths (σ).*

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 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: $\log_1 0(min(dist(data)))$)
`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`	Uncertainity 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)
```

---

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

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(lWhich(2, len = 3) == c(FALSE, TRUE, FALSE))
all(lWhich(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

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

Arguments

x	A DiffusionMap
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: 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: palette())
...	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 tick.marks 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 colorlegend options (default: empty list)
interactive	Use plot3d to plot instead of scatterplot3d?
draw.legend	logical. If TRUE, draw color legend (default: TRUE if 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))

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

x	Sigmas object to plot
col	Vector of bar colors or single color for all bars
highlight.col	Color for highest bar. Overrides col
line.col	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 defauts, 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)
```

`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 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

Slots

`log.sigmas` Vector of length m containing the $\log_{10}$ of the σs

`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 σs 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)
```

Index