

Package ‘slanter’

September 14, 2020

Version 0.1-0

Date 2020-08-13

Title Slanted Matrices and Ordered Clustering

Description Slanted matrices and ordered clustering for better visualization of similarity data.

RoxygenNote 6.1.1

Encoding UTF-8

License MIT + file LICENSE

Imports Matrix, pheatmap, pracma, stats

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Oren Ben-Kiki [aut, cre],
Weizmann Institute of Science [cph]

Maintainer Oren Ben-Kiki <oren@ben-kiki.org>

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2020-09-14 09:10:03 UTC

R topics documented:

meristems	2
oclust	2
reorder_frame	3
reorder_hclust	4
sheatmap	4
slanted_orders	6
slanted_reorder	7

Index	8
--------------	----------

meristems	<i>Sample RNA data of similarity between batches of 1000 cells of tomato meristem cells.</i>
-----------	--

Description

This is a simple matrix where each entry is the similarity (correlation) between a pair of batches. Negative correlations were changed to zero to simplify the analysis.

Usage

```
data(meristems)
```

Format

A simple square matrix.

Examples

```
data(meristems)
similarity <- meristems
similarity[similarity < 0] = 0
slanter::sheatmap(meristems, order_data=similarity, show_rownames=FALSE, show_colnames=FALSE)
```

oclust	<i>Hierarchically cluster ordered data.</i>
--------	---

Description

Given a distance matrix for sorted objects, compute a hierarchical clustering preserving this order. That is, this is similar to `hclust` with the constraint that the result's order is always 1:N.

Usage

```
oclust(distances, method = "ward.D2", order = NULL, members = NULL)
```

Arguments

distances	A distances object (as created by <code>stats::dist</code>).
method	The clustering method to use (only <code>ward.D</code> and <code>ward.D2</code> are supported).
order	If specified, assume the data will be re-ordered by this order.
members	Optionally, the number of members for each row/column of the distances (by default, one each).

Details

If an order is specified, assumes that the data will be re-ordered by this order. That is, the indices in the returned `hclust` object will refer to the post-reorder data locations, ****not**** to the current data locations.

This can be applied to the results of `slanted_reorder`, to give a "plausible" clustering for the data.

Value

A clustering object (as created by `hclust`).

Examples

```
clusters <- slanter::oclust(dist(mtcars), order=1:dim(mtcars)[1])
clusters$order
```

reorder_frame	<i>Reorder the rows of a frame.</i>
---------------	-------------------------------------

Description

You'd expect `data[order,]` to "just work". It doesn't for data frames with a single column, which happens for annotation data, hence the need for this function. Sigh.

Usage

```
reorder_frame(frame, order)
```

Arguments

frame	A data frame to reorder the rows of.
order	An array containing indices permutation to apply to the rows.

Value

The data frame with the new row orders.

Examples

```
df <- data.frame(foo=c(1, 2, 3))
df[c(1,3,2),]
slanter::reorder_frame(df, c(1,3,2))
```

reorder_hclust	<i>Given a clustering of some data, and some ideal order we'd like to use to visualize it, reorder (but do not modify) the clustering to be as consistent as possible with this ideal order.</i>
----------------	--

Description

Given a clustering of some data, and some ideal order we'd like to use to visualize it, reorder (but do not modify) the clustering to be as consistent as possible with this ideal order.

Usage

```
reorder_hclust(clusters, order)
```

Arguments

clusters	The existing clustering of the data.
order	The ideal order we'd like to see the data in.

Value

A reordered clustering which is consistent, wherever possible, the ideal order.

Examples

```
clusters <- hclust(dist(mtcars))
clusters$order
clusters <- slanter::reorder_hclust(clusters, 1:length(clusters$order))
clusters$order
```

sheatmap	<i>Plot a heatmap with values as close to the diagonal as possible.</i>
----------	---

Description

Given a matrix expressing the cross-similarity between two (possibly different) sets of entities, this will reorder it to move the high values close to the diagonal, for a better visualization.

Usage

```
sheatmap(data, ..., order_data = NULL, annotation_col = NULL,
  annotation_row = NULL, order_rows = TRUE, order_cols = TRUE,
  squared_order = TRUE, same_order = FALSE, cluster_rows = TRUE,
  cluster_cols = TRUE, oclust_rows = TRUE, oclust_cols = TRUE,
  clustering_distance_rows = "euclidian",
  clustering_distance_cols = "euclidian",
  clustering_method = "ward.D2", clustering_callback = NA)
```

Arguments

<code>data</code>	A rectangular matrix to plot, of non-negative values (unless <code>order_data</code> is specified).
<code>...</code>	Additional flags to pass to <code>pheatmap</code> .
<code>order_data</code>	An optional matrix of non-negative values of the same size to use for computing the orders.
<code>annotation_col</code>	Optional data frame describing each column.
<code>annotation_row</code>	Optional data frame describing each row.
<code>order_rows</code>	Whether to reorder the rows. Otherwise, use the current order.
<code>order_cols</code>	Whether to reorder the columns. Otherwise, use the current order.
<code>squared_order</code>	Whether to reorder to minimize the l2 norm (otherwise minimizes the l1 norm).
<code>same_order</code>	Whether to apply the same order to both rows and columns (if reordering both).
<code>cluster_rows</code>	Whether to cluster the rows, or the clustering to use.
<code>cluster_cols</code>	Whether to cluster the columns, or the clustering to use.
<code>oclust_rows</code>	Whether to use <code>oclust</code> instead of <code>hclust</code> for the rows (if clustering them).
<code>oclust_cols</code>	Whether to use <code>oclust</code> instead of <code>hclust</code> for the columns (if clustering them).
<code>clustering_distance_rows</code>	The default method for computing row distances (by default, euclidian).
<code>clustering_distance_cols</code>	The default method for computing column distances (by default, euclidian).
<code>clustering_method</code>	The default method to use for hierarchical clustering (by default, <code>ward.D2</code> and <code>*not* complete</code>).
<code>clustering_callback</code>	Is not supported.

Details

If you have an a-priori order for the rows and/or columns, you can prevent reordering either or both by specifying `order_rows=FALSE` and/or `order_cols=FALSE`. Otherwise, `slanted_orders` is used to compute the "ideal" slanted order for the data.

By default, the rows and columns are ordered independently from each other. If the matrix is asymmetric but square (e.g., a matrix of weights of a directed graph such as a K-nearest-neighbors graph), then you can specify `same_order=TRUE` to force both rows and columns to the same order.

There are four options for controlling clustering:

* By default, `sheatmap` will generate a clustering tree using `oclust`, to generate the "best" clustering that is also compatible with the slanted order.

* Request that `sheatmap` will use the same `hclust` as `pheatmap` (e.g., `oclust_rows=FALSE`). In this case, the tree is reordered to be the "most compatible" with the target slanted order. That is, `sheatmap` will invoke `reorder_hclust` so that, for each node of the tree, the order of the two subtrees will be chosen to best match the target slanted order. The end result need not be identical to the slanted order, but is as close as possible given the `hclust` clustering tree.

* Specify an explicit clustering (e.g., `cluster_rows=hclust(...)`). In this case, `sheatmap` will again merely reorder the tree but will not modify it.

In addition, you can give this function any of the `pheatmap` flags, and it will just pass them on. This allows full control over the diagram's features.

Note that `clustering_callback` is not supported. In addition, the default `clustering_method` here is `ward.D2` instead of `complete`, since the only methods supported by `oclust` are `ward.D` and `ward.D2`.

Value

Whatever `pheatmap` returns.

Examples

```
slanter::sheatmap(cor(t(mtcars)))
slanter::sheatmap(cor(t(mtcars)), oclust_rows=FALSE, oclust_cols=FALSE)
pheatmap::pheatmap(cor(t(mtcars)))
```

<code>slanted_orders</code>	<i>Compute rows and columns orders which move high values close to the diagonal.</i>
-----------------------------	--

Description

For a matrix expressing the cross-similarity between two (possibly different) sets of entities, this produces better results than clustering (e.g. as done by `pheatmap`). This is because clustering does not care about the order of each two sub-partitions. That is, clustering is as happy with $((2,1), (4,3))$ as it is with the more sensible $((1,2), (3,4))$. As a result, visualizations of similarities using naive clustering can be misleading.

Usage

```
slanted_orders(data, order_rows = TRUE, order_cols = TRUE,
  squared_order = TRUE, same_order = FALSE, max_spin_count = 10)
```

Arguments

<code>data</code>	A rectangular matrix containing non-negative values.
<code>order_rows</code>	Whether to reorder the rows.
<code>order_cols</code>	Whether to reorder the columns.
<code>squared_order</code>	Whether to reorder to minimize the l2 norm (otherwise minimizes the l1 norm).
<code>same_order</code>	Whether to apply the same order to both rows and columns.
<code>max_spin_count</code>	How many times to retry improving the solution before giving up.

Value

A list with two keys, `rows` and `cols`, which contain the order.

Examples

```
slanter::slanted_orders(cor(t(mtcars)))
```

slanted_reorder	<i>Reorder data rows and columns to move high values close to the diagonal.</i>
-----------------	---

Description

Given a matrix expressing the cross-similarity between two (possibly different) sets of entities, this uses `slanted_orders` to compute the "best" order for visualizing the matrix, then returns the reordered data. Commonly used in `heatmap(slanted_reorder(data), ...)`, and of course `heatmap` does this internally for you.

Usage

```
slanted_reorder(data, order_data = NULL, order_rows = TRUE,
  order_cols = TRUE, squared_order = TRUE, same_order = FALSE)
```

Arguments

<code>data</code>	A rectangular matrix to reorder, of non-negative values (unless <code>order_data</code> is specified).
<code>order_data</code>	An optional matrix of non-negative values of the same size to use for computing the orders.
<code>order_rows</code>	Whether to reorder the rows.
<code>order_cols</code>	Whether to reorder the columns.
<code>squared_order</code>	Whether to reorder to minimize the l_2 norm (otherwise minimizes the l_1 norm).
<code>same_order</code>	Whether to apply the same order to both rows and columns.

Value

A matrix of the same shape whose rows and columns are a permutation of the input.

Examples

```
slanter::slanted_reorder(cor(t(mtcars)))
```

Index

* **datasets**

meristems, [2](#)

meristems, [2](#)

oclust, [2](#)

reorder_frame, [3](#)

reorder_hclust, [4](#)

sheatmap, [4](#)

slanted_orders, [6](#)

slanted_reorder, [7](#)