Package 'DataNet'

October 7, 2018

Type Package

Title Tools for Working with Correlation Networks

Version 1.0.1

Description Laying out large networks/graphs can be difficult when there are very many connections/edges. This package implements a grid-constrained layout algorithm which places nodes/vertices in different cells on a grid to maintain an optimal distance from each other. The cells of the grid can then be plotted using colour to highlight properties of the nodes. Connections are not drawn and thus do not obscure important properties of the network.

License LGPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

R topics documented:

`as.distance` *Convert correlations to distances.*

Description

Convert a correlation matrix (or other adjacency matrix) into a distance matrix.

Usage

```
as.distance(network, type = "positive", range = "unit")
```

Arguments

network	A numerical matrix (as returned for example by cor) of correlations between nodes/vertices of a network.
type	One of 'positive' (only positive edges are retained; the default) , 'absolute' (absolute values are retained, so that a negative-weighted edge will have the same distance as if it were positive) or 'scaled' (where negative-weighted edges have greater distances and uncorrelated edges will have intermediate distances).
range	One of 'unit' (edge weights are in the range -1 to 1 as for a correlation matrix; the default) or 'range' (the edge weights will be scaled so that the range of supplied weights ends up on the scale 0 to 1).

Value

A matrix, of the same dimensions (and row/column names) as the input, containing the edge weights as distances. Thus, the most highly-correlated edges will tend to 0 and uncorrelated edges will tend to 1.

Examples

```
## Not run:
require(DataNet)
# Load example data and create distance matrix
data(hipExpr)
network = cor(hipExpr, method="pearson", use="pairwise.complete.obs")
distance.matrix = as.distance(network)

## End(Not run)
```

---

as.edge.list *Convert a distance matrix to an edge list.*

Description

Convert a distance matrix representing an undirected network into an edge list.

Usage

```
as.edge.list(network, threshold)
```

Arguments

network	A numerical matrix (as returned for example by as.distance) of distances between nodes/vertices of a network. Values must be positive and will usually lie in the range 0–1 (although no upper bound is set).
threshold	A threshold above which edges will not be included in the edge list.

Details

This function efficiently transforms the lower triangle of a distance matrix into a data.frame representing the edges of a network.

Value

A `data.frame` representing the edges of a network. The columns `source` and `target` are character vectors containing the IDs of the source and target nodes respectively (the column and row names of the correponding elements of the distance matrix). The column `distance` is a numeric vector containing the value of the of the correponding element of the distance matrix.

Examples

```
## Not run:
require(DataNet)
# Load example data and create distance matrix
data(hipExpr)
network = cor(hipExpr, method="pearson", use="pairwise.complete.obs")
distance.matrix = as.distance(network)

# Convert to edge list
edge.list = as.edge.list(distance.matrix, 0.4)

## End(Not run)
```

`DataNet` *Tools for working with correlation networks.*

Description

The DataNet package provides a fast layout algorithm `fast.force.layout` suitable for large and highly-connected networks. The function `draw.stick.graph` enables large graphs to be plotted in a simple stick graph where vertices are not drawn. Included are two helper functions `as.distance`, to convert correlations to distances, and `as.edge.list`, to reformat a distance matrix to an edge list. There is also an example dataset `hipExpr` provided.

Functions provided

`as.distance`

`as.edge.list`

`fast.force.layout`

`draw.stick.graph`

Data provided

`hipExpr`

`draw.stick.graph` *Plot a network as a graph of edges coloured by weight.*

Description

Plot a network as a graph of edges (vertices are not drawn) coloured by the weight/optimal distance.

Usage

```
draw.stick.graph(edge.list, coordinates, plot.edges = 1:nrow(edge.list),
  plot.vertices = 1:nrow(coordinates),
  edge.colours = colorRampPalette(c("red", "yellow", "white"))(256),
  new = TRUE, return.coordinates = FALSE, ...)
```

Arguments

`edge.list`	A `data.frame` with two `character` columns, 'source' and 'target', containing the vertex IDs, and one `numeric` column, distance, containing the optimal edge distance.
`coordinates`	A `data.frame` with two `numeric` columns, `x` and `y` containing the Cartesian coordinates of the vertices. The vertex IDs (provided as the row names of this `data.frame`) must correspond to the vertices used in the edge list. This object can be generated using the function `fast.force.layout`.
`plot.edges`	A vector, the same length as the number of rows of `edge.list`, specifying which edges are to be drawn (default is all edges). The vector may be logical, numeric or character if the edge.list has rownames).
`plot.vertices`	A vector, the same length as the number of rows of `edge.list`, specifying which vertices (nodes) are to be drawn (default is all vertices). The vector may be logical, numeric or character (corresponding to the rownames of `coordinates`).
`edge.colours`	A vector of colours which will be used as a colour scale from the closest to most distant edges. Colours can be specified in any of the three R formats.
`new`	Logical (`TRUE` by default). Whether to create a new plot. If `FALSE`, the graph will be drawn over the existing plot (enabling additional plotting layers to be added).
`return.coordinates`	Logical (`FALSE` by default). Whether to create a new plot. If `TRUE`, the graph will be drawn over the existing plot (enabling additional plotting layers to be added).
`...`	Addtional arguments to be passed to `segments`.

Details

This function provides a convenient way to plot results of `fast.force.layout`. The edge list, which can be derived from a distance matrix using `as.edge.list`, is the same as provided to `fast.force.layout` which returns the 'coordinates' object also required for this function.

Value

If return.coordinates is TRUE, then a data.frame will be returned containing the same information as the edge.list but with additional columns describing the Cartesian coordinates of the source and target vertices as well as the colour in which the edge was plotted. The default (return.coordinates = FALSE) returns nothing.

Examples

```
## Not run:
require(DataNet)
# Load example data and create distance matrix
data(hipExpr)
network = cor(hipExpr, method="pearson", use="pairwise.complete.obs")
distance.matrix = as.distance(network)

# Convert to edge list
edge.list = as.edge.list(distance.matrix, 0.4)

# Calculate coordinates
coordinates = fast.force.layout(edge.list)

# Plot graph
draw.stick.graph(edge.list, coordinates)


## End(Not run)
```

---

fast.force.layout *Force-directed graph layout.*

Description

Plotting function for network vertices laid out on a grid.

Usage

```
fast.force.layout(edge.list, extend = NULL, iterations = 1, k = 0.5,
  repelling.edges = 1)
```

Arguments

edge.list	A data.frame representing the edges of a network. The columns source and target are character vectors containing the IDs of the source and target nodes respectively. The column distance is a numeric vector containing the value of the of the correponding element of the distance matrix.
extend	A data.frame of vertex coordinates, as returned by this function. If supplied, then these coordinates will be used as starting postions. The default value of NULL will place all vertices at the origin. It is important that the random seed is set to be the same for all iterations if a reproducible layout is required. This can be done by calling set.seed before performing the layout.

`iterations` An integer specifying the number of iterations to be performed. In each iteration, the length of every edge is compared against the desired distance and the position of the associated vertices adjusted accordingly.

`k` A `double` specifying the fraction of the displacement (the difference between the desired and actual edge lengths) by which each edge length is altered in each itereation. The default value is 0.5 (half the displacement).

`repelling.edges` The fraction of edges which are randomly selected in each iteration for calculation of repulsion. Optimally, this value is set at 1 (all edges; default) but for large networks this significantly affects performance. A value of 0.01 (10 computation times while retaining a reasonable layout.

Value

A `data.frame` of vertex coordinates with columns x and y. The vertex IDs are provided as row names.

Examples

```
## Not run:
require(DataNet)
# Load example data and create distance matrix
data(hipExpr)
network = cor(hipExpr, method = "pearson", use = "pairwise.complete.obs")
distance.matrix = as.distance(network)

# Convert to edge list
edge.list = as.edge.list(distance.matrix, 0.4)

# Calculate coordinates
coordinates = fast.force.layout(edge.list, iterations = 100)

# Plot graph
draw.stick.graph(edge.list, coordinates)


## End(Not run)
```

`generate.network` *Generate test graphs.*

Description

Create edge lists for several standard graphs for testing puposes.

Usage

```
generate.network(graph)
```

Arguments

`graph` A `character` string specifying the graph to be generated.

Details

This function provides a convenient way to generate several useful graphs for testing and parameter optimisation. Three graphs are provided: `circle` (26 vertices connected in a ring), `cube` (8 vertices connected as if they were corners of a 3-dimensional cube) and `knuth` or `lesmis` (the classic Les Miserables character co-mention graph from Donald Knuth; D. E. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison-Wesley, Reading, MA (1993)).

Value

A `data.frame` containing an edge list ready to be passed to `fast.force.layout`.

Examples

```
## Not run:
require(DataNet)

# Generate graph as an edge list
edge.list = generate.network("circle")
coordinates = fast.force.layout(edge.list, iterations = 100)
draw.stick.graph(edge.list, coordinates)

edge.list = generate.network("cube")
coordinates = fast.force.layout(edge.list, iterations = 100)
draw.stick.graph(edge.list, coordinates, edge.colours = "black")

edge.list = generate.network("knuth")
coordinates = fast.force.layout(edge.list, iterations = 100)
draw.stick.graph(edge.list, coordinates, edge.colours = colorRampPalette(c("red", "darkblue"))(256), lwd = 1.


## End(Not run)
```

---

hipExpr *Expression data for 1000 genes.*

Description

A dataset containing the expression of 1000 genes/transcripts measured in brain (hippocampus) tissue of different strains of mice. This example dataset contains 1000 selected genes to provide a dataset of manageable size for testing correlation and network functions.

Usage

```
hipExpr
```

Format

A data frame with 74 rows and 1000 variables:

columns Gene/transcript expression in arbitrary units (log2 transformed). Column names are ENSEMBL gene identifiers.

Details

Note: by being stripped of their context, the data for the 1000 genes in this dataset have lost their biological significance. This dataset is intended only for testing functions requiring a particular data structure.

Source

Details about the original dataset are provided at http://www.genenetwork.org/webqtl/main.py?FormID=sharinginfo&GN_AccessionId=112

Index