visEnrichment

October 24, 2014

visEnrichment	*Function to visualise enrichment analysis outputs in the context of the ontology hierarchy*

Description

visEnrichment is supposed to visualise enrichment analysis outputs (represented as an 'Eoutput' object) in the context of the ontology hierarchy (direct acyclic graph; DAG). Only part of DAG induced by those nodes/terms specified in query nodes (and the mode defining the paths to the root of DAG) will be visualised. Nodes in query are framed in black (by default), and all nodes (in query plus induced) will be color-coded according to a given data.type ('zscore'; otherwise taking the form of 10-based negative logarithm for 'adjp' or 'pvalue'). If no nodes in query, the top 5 significant terms (in terms of adjusted p-value) will be used for visualisation

Usage

```
visEnrichment(e, nodes_query = NULL, num_top_nodes = 5,
path.mode = c("all_shortest_paths", "shortest_paths", "all_paths"),
data.type = c("adjp", "pvalue", "zscore"), height = 7, width = 7,
margin = rep(0.1, 4), colormap = c("yr", "bwr", "jet", "gbr", "wyr",
"br",
"rainbow", "wb", "lightyellow-orange"), ncolors = 40, zlim = NULL,
colorbar = T, colorbar.fraction = 0.1, newpage = T,
layout.orientation = c("left_right", "top_bottom", "bottom_top",
"right_left"), node.info = c("both", "none", "term_id", "term_name",
"full_term_name"), graph.node.attrs = NULL, graph.edge.attrs = NULL,
node.attrs = NULL)
```

Arguments

e	an object of S4 class Eoutput
nodes_query	a verctor containing a list of nodes/terms in query. These nodes are used to produce a subgraph of the ontology DAG induced by them. If NULL, the top significant terms (in terms of p-value) will be determined by the next 'num_top_nodes'
num_top_nodes	a numeric value specifying the number of the top significant terms (in terms of p-value) will be used. This parameter does not work if the previous 'nodes_query' has been specified

`path.mode` the mode of paths induced by nodes in query. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

`data.type` a character telling which data type for nodes in query is used to color-code nodes. It can be one of 'adjp' for adjusted p-values (by default), 'pvalue' for p-values and 'zscore' for z-scores. When 'adjp' or 'pvalue' is used, 10-based negative logarithm is taken. For the style of how to color-code, please see the next arguments: colormap, ncolors, zlim and colorbar

`height` a numeric value specifying the height of device

`width` a numeric value specifying the width of device

`margin` margins as units of length 4 or 1

`colormap` short name for the colormap. It can be one of "yr" (yellow-red colormap; by default), "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

`ncolors` the number of colors specified over the colormap

`zlim` the minimum and maximum z/data values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted

`colorbar` logical to indicate whether to append a colorbar. If data is null, it always sets to false

`colorbar.fraction` the relative fraction of colorbar block against the device size

`newpage` logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`layout.orientation` the orientation of the DAG layout. It can be one of "left_right" for the left-right layout (viewed from the DAG root point; by default), "top_bottom" for the top-bottom layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left layout

`node.info` tells the ontology term information used to label nodes. It can be one of "both" for using both of Term ID and Name (the first 15 characters; by default), "none" for no node labeling, "term_id" for using Term ID, "term_name" for using Term Name (the first 15 characters), and "full_term_name" for using the full Term Name

`graph.node.attrs` a list of global node attributes. These node attributes will be changed globally. See 'Note' below for details on the attributes

`graph.edge.attrs` a list of global edge attributes. These edge attributes will be changed globally. See 'Note' below for details on the attributes

`node.attrs` a list of local edge attributes. These node attributes will be changed locally; as such, for each attribute, the input value must be a named vector (i.e. using Term ID as names). See 'Note' below for details on the attributes

Value

An object of class 'Ragraph'

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings
- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings
- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node
- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings
- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

dcEnrichment, dcRDataLoader, dcConverter

Examples

```
# 1) load SCOP.sf (as InfoDataFrame object)
SCOP.sf <- dcRDataLoader(SCOP.sf)
# randomly select 20 domains
data <- sample(rowNames(SCOP.sf), 20)

# 2) perform enrichment analysis, producing an object of S4 class Eoutput
eoutput <- dcEnrichment(data, domain="SCOP.sf", ontology="GOMF")
eoutput

# 3) visualise the top 10 significant terms
# color-coded according to 10-based negative logarithm of p-values
visEnrichment(eoutput)
# color-coded according to zscore
visEnrichment(eoutput, data.type=zscore)

# 4) visualise the top 5 significant terms in the ontology hierarchy
nodes_query <- names(sort(adjp(eoutput))[1:5])
visEnrichment(eoutput, nodes_query=nodes_query)
# change the frame color: highlight (framed in blue) nodes/terms in query
nodes.highlight <- rep("blue", length(nodes_query))
names(nodes.highlight) <- nodes_query
visEnrichment(eoutput, nodes_query=nodes_query,
node.attrs=list(color=nodes.highlight))
```