Package 'netbiov'

October 16, 2018

Type Package

Title A package for visualizing complex biological network

Version 1.14.0

Date 2015-19-07

Author Shailesh tripathi and Frank Emmert-Streib

Maintainer Shailesh tripathi <shailesh.tripathy@gmail.com>

Description A package that provides an effective visualization of large biological networks

Depends R (>= 3.1.0), igraph (>= 0.7.1)

Suggests BiocStyle,RUnit,BiocGenerics,Matrix

biocViews GraphAndNetwork, Network, Software, Visualization

License GPL (>= 2)

URL http://www.bio-complexity.com

git_url https://git.bioconductor.org/packages/netbiov

git_branch RELEASE_3_7

git_last_commit 86b77d5

git_last_commit_date 2018-04-30

Date/Publication 2018-10-15

R topics documented:

`color_list` *A list object of color vectors*

Description

A list object where each component of list contans a vector of colors.

Usage

```
data("color_list")
```

Format

list object

`gnet_bcell` *An igraph object of B-cell network inferred from gene expression data*

Description

A connected subnetwork of bcell network.

Usage

```
data("gnet_bcell")
```

Format

igaph class object

`level.plot` *Exploratory visualization of information spread of biological networks*

Description

Generates a level plot by selecting a set of nodes at the initial level '0'. Outgoing nodes are plotted on higher levels than level '0', and the incoming nodes are plotted on lower levels. This process is repeated for the nodes at each new level until all nodes are plotted. For this function, the graph can be plotted at each step, like a sequence, by using the 'plotstep' option. Edge colors show a connection between different levels. If the level difference is one, than the edgecolor is "grey" (default). For a level difference of zero, the default edge color is "brown". For level differences greater than one (upward edges), the edgecolor is "blue". If the level difference is less than zero (downward edges) the edgecolor is "red". The level plot function helps to see the information spread in a network, starting from a set of nodes (initial_nodes).

Usage

```
level.plot(x, layout.function=NULL, type=1, initial_nodes=NULL,
init_nodes=0, order_degree = "in", plotsteps = FALSE,
saveplots=FALSE, dirname=NULL, vertex.colors=NULL,
edge.col=NULL, tkplot=FALSE, nodeset=NULL,path.col="green",
col.s1="red", col.s2="yellow", nodes.on.path=TRUE,v.size=2,
e.size=.5, v.lab=FALSE, bg="black", v.lab.cex=0.5,
v.lab.col="skyblue",sf=4,e.path.width=1,e.curve=.5,
level.spread=FALSE)
```

Arguments

`x`	`x` is a graph object created from an adjacency matrix or from a tabular data of two columns using the `graph()` function available in `igraph`.
`layout.function`	`layout.function` is a function class which genetates 2D coordinates for an input graph, this function is used to adjust x coordinates of the plot
`type`	`type` is a numerical value, it can either be 1 or 2, it is used to adjust coordinates generated from `layout.function` option.
`initial_nodes`	`initial_nodes` is a vector of integers of the vertex ids of the graph, from which the plot of a graph starts.
`init_nodes`	`init_nodes` is an integer which provides the total numbers of vertices to initialize the plot of a graph randomly. If `initial_node` is NULL and `init_nodes` is `0` then `level.plot` selects 3 vertices randomly.
`order_degree`	`order_degree` is used to arrange nodes in a given order, based on their degrees from left to right in increasing order. Possible options for the ordering are 'in' and 'out'.
`plotsteps`	`plotsteps` is logical; if 'TRUE', shows a sequence of plots of the nodes found at each steps. Possible options are 'TRUE' and 'FALSE'.

saveplots	saveplots is logical; if 'TRUE' than saves plots of each steps in a pdf file inside a folder with the name plot_steps_dir in the current working directory.
dirname	dirname it is a directory path option if this option is not null then plots of each steps are saved in the given directory.
vertex.colors	is a vector of color names or hexadecimal codes of three colors. The first element is for vertex color at level 0, second element is for vertex color at level above level 0, third element is for vertex color lower than level 0.
edge.col	is a 4 dimensional vector of colors. The first element is for edges that have a level difference between nodes greater than one. The second element is for edges that are connected to nodes with a level difference less than one. The third element is for edges connecting nodes on the same level. The fourth element is for edges with a level difference between connected nodes equal to one.
nodeset	is a list object containing two vector. The first vector contains a set of start nodes and the second vector contains the end nodes. Shortest paths are visualized between the start nodes and end nodes.
path.col	is a character value for a color used to color the shortest paths between start nodes and end nodes, specified innodeset option.
col.s1	color of start nodes specified in nodeset option, possible values can be a color name or a hexadecimal code of a color.
col.s2	color of end nodes specified in nodeset option, possible values can be a color name or a hexadecimal code of a color.
nodes.on.path	is a logical value It shows the nodes which connect start nodes and end nodes specified in nodeset option.
v.size	is a numeric value or a numeric vector which contains numeric values to assign the size of nodes.
e.size	is a numeric value or a numeric vector which contains numeric values to assign the width of edges.
v.lab	is a logical value to show vertex label.
v.lab.cex	is a numerical value to set the font size of vertex labels.
v.lab.col	is a color name or a hexadecimal color code to assign colors to vertex labels.
sf	A numeric value, which is a scaling factor. This is used to adjust the vertex size when v.size input is a numeric vector.
e.path.width	is a numeric value. This is used to adjust the edge-size of the shortest paths between start nodes and end nodes. The start nodes and end nodes are given as an input as a list object for the "nodeset" option.
e.curve	is a positive numeric value that adjusts the curvature of the edges which connect nodes that are on same level.
level.spread	is a logical value, if it is 'TRUE' then the width of each level is the same.
bg	name of a color or a hexadecimal code of a color. This option is used to color the background of the plot. The default option is "black".
tkplot	is logical value, if TRUE use the tkplot function of igraph for plotting the network.
...	... parameter for other inputs.

Details

This function starts plotting nodes at level 0. These nodes are either user-defined or randomly selection. The outgoing nodes of the initial node set are plotted on the upper level and incoming nodes are plotted on the lower level. These steps are repeated until all nodes are plotted.

Value

This function plots a graph using 'tkplot' or 'plot' function available in 'igraph'. This function returns a list. The first component of the list contains the x and y coordinates and color ids of the nodes of the graph and the second component contains a vector of edge colors.

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
data("PPI_Athalina")

### Example 1 #####
#####% 5 initial nodes are picked randomly #####
id <- level.plot(g1, init_nodes =5 ,tkplot=FALSE)

### Example 2 #####
#####% initial nodes are given as an input #####
id <- level.plot(g1, initial_nodes =c(1,4,5,7,9,11,25,27,29,100,101),
tkplot=FALSE)

### Example 3 #####
#####% width of each level is same #####
id <- level.plot(g1, initial_nodes
=c(1,4,5,7,9,11,25,27,29,100,101),
level.spread = TRUE,tkplot=FALSE)

### Example 4 #####
#####% plot of graph when nodes are not arranged based on the degree #####
id <- level.plot(g1, initial_nodes
=c(1,4,5,7,9,11,25,27,29,100,101),
order_degree= NULL, level.spread = TRUE,
tkplot=FALSE)

### Example 5 #####
#####% plot of graph when nodes are not arranged based on the degree #####
id <- level.plot(g1, initial_nodes
=c(1,4,5,7,9,11,25,27,29,100,101),
order_degree= NULL, level.spread = TRUE,
tkplot=FALSE)

### Example 6 #####
###% plot of graph using a forcebased algorithm
###in igraph package #####
xx <- level.plot(g1, layout.function=layout.reingold.tilford,
initial_nodes=c(1,4,5,7,9,11,25,27,29,100,101))

xx <- level.plot(g1, layout.function=layout.fruchterman.reingold,type=2,
init_nodes=20)

xx <- level.plot(g1, layout.function=layout.kamada.kawai,
```

```
type=1, init_nodes=20)

### Example 7 #####
## Shortest path between initial set of nodes at level '0'
## to a set of nodes with vertex id 10, 1000, 1001, 1002 ##%
id <- level.plot(g1, initial_nodes=c(101,1,5,7),
vertex.colors=c("white", "white", "white"),
edge.col=c("grey", "grey", "grey", "grey"),
nodeset= list(c(101,1,5,7),c(10,1000,1001,1002)),
tkplot=FALSE, level.spread=TRUE,
order_degree=NULL)

### Example 8 #####
###  A directed network is plotted with level.plot function ##%
g <- barabasi.game(300, power=0)
id <-  level.plot(g, tkplot=FALSE,
level.spread=TRUE, order_degree=NULL)
```

`modules_bcell` *A list object of modules's information of B-cell network*

Description

A list object of modules obtained by using fastgreedy algorithm in igraph. Then we performed the enrichment analysis for GO pathways on modules.

Usage

```
data("modules_bcell")
```

Format

list object

`modules_PPI_Athalina` *A list object of modules's information of PPI interaction of A. thalina*

Description

A list object of modules obtained by using fastgreedy algorithm in igraph. Then we performed the enrichment analysis for GO pathways on modules.

Usage

```
data(modules_PPI_Athalina)
```

Format

list object

mst.plot *Network plot*

Description

Large complex networks can be plotted using the fruchterman-reingold algorithm or the Kamada-kawai algorithm or any given input function on minimum spanning tree of each network. The edges of the minimum spanning tree (MST) are shown in black color, the rest of the edges are shown in a different colors which are a function of distances between the coordinates of the nodes.

Usage

```
 mst.plot(x, layout.function=NULL,colors=NULL,mst.edge.col="white",
vertex.color = "skyblue",tkplot=FALSE,expression=NULL, v.size=FALSE,
e.size=FALSE, mst.e.size=1, edge.col.wt=NULL, v.lab=FALSE,
e.lab=NULL, bg="black",v.lab.cex=0.5, e.lab.cex=0.5,
v.lab.col="blue",lab.dist=0, e.lab.col="blue",
v.sf=c(3,12),e.arrow=.2)
```

Arguments

x	x is a graph object, created using the igraph package.
layout.function	layout.function is a function class. The user can pass his own function to plot the graph. The input function passed to layout.function should return a 2D matrix of coordinates with rows equal to the number of vertices of the input graph and two columns representing x and y coordinates.
mst.edge.col	This option assigns a color to the edges of the minimum spanning tree of graph 'g'. The default color is black for 'tkplot=TRUE'. If 'tkplot=FALSE' it is white.
colors	colors is a vector for the edge colors to color the remaining edges of a graph. The remaining edges are colored based on the distance between the nodes connecting them.
vertex.color	vertex.colors is a vector of colors for the vertices of a graph.
expression	expression is a numeric vector which represents properties of genes or proteins, e.g., their mean expression values or p-values. If this option is given, the nodes are colored with a range of colors from red to blue (red for low expression, blue for high expression) depending on the expression values.
tkplot	tkplot is a boolean variable. If it is FALSE, the graph is plotted with the plot function with a black background.
v.size	is a numeric value or a numeric vector which contains values to assign the size of the nodes.
e.size	is a numeric value or a numeric vector which contains values to assign the width of edgesother than the minimum spanning tree edges of the graph.
mst.e.size	is a numeric value which assigns the edge width to the edges of minimum spanning tree of the input graph.
v.lab	is logical value to show vertex labels.
e.lab	is logical value to show edge labels.

bg	is a character value to color the background.
v.lab.cex	is a numerical value to set the font size of the vertex labels.
e.lab.cex	is a numerical value to set the font size of the edge labels.
lab.dist	is a numerical value to adjust the distance of labels from the nodes.
v.lab.col	is a hexadecimal character value to assign colors to the vertex labels.
e.lab.col	is a hexadecimal character value to assign colors to the edge labels.
e.arrow	is a inumerical value to set the arrow width in a directed network.
v.sf	is a 2 dimensional numeric vector, which represents the minimum and maximum limits of the size of vertices. This is used to adjust the vertex size when the v.size input is a numeric vector.
edge.col.wt	is a vector of numeric values provided for each edge. This is used to color the edges from blue to red from high to low values.
...	... parameter for other inputs.

Value

This function plots the graph object given as an input using 'tkplot' or 'plot' function available in 'igraph' package. This function returns a netbiov class object.

Examples

```
  data("PPI_Athalina")

  ## Example 1 ####
  kk <- mst.plot(g1)

  ## Example 2 ####
  ## When expression values of genes or nodes are given and
  ## to be plotted as ai color of vertices ###
  id <- mst.plot(g1, expression=rnorm(vcount(g1)), v.size=1)

  ## Example 3 ####
  ## When expression values of genes or nodes are given and to be
  ## plotted as a
  ##color of vertices, also the degree of nodes to be shown
  ## as their vertex-size ###
  id <- mst.plot(g1, expression=rnorm(vcount(g1)),
 v.size=degree(g1), v.sf=c(1,5))

  ## Example 4 ####
  ## When MST edges are highlighted in purple color
## and rest of the edges are plotted with a range of
## heat colors depending on the distance between nodes ###
  id <- mst.plot(g1, mst.edge.col="purple",
 colors=heat.colors(20), vertex.color="yellow",
 v.size=1)


  ## Example 5 ####
  ## Plotting a graph with kamada-kawai layout algorithm ###
  id <- mst.plot(g1, mst.edge.col="purple",
colors=heat.colors(20), vertex.color="white", v.size=1,
layout.function=layout.kamada.kawai)
```

```
  ## Example 6 ####
  ## Plotting a graph with when weights of edges are given ###
  id <- mst.plot(g1, mst.edge.col="purple", edge.col.wt =
runif(ecount(g1), min=1, max=10), vertex.color="yellow",
 v.size=1, layout.function=layout.kamada.kawai)
```

`mst.plot.mod` *Network plot*

Description

Global layout style:

Large complex networks can be plotted using the fruchterman-reingold algorithm, the Kamada-kawai algorithm or any given input function by applying these algorithms to the minimum spanning tree (MST). The edges of the minimum spanning tree are shown in black, all other edges are shown in colors which are a function of the distances between the coordinates of the nodes.

Usage

```
 mst.plot.mod(x, layout.function=NULL,colors=NULL,
mst.edge.col="white", vertex.color = "skyblue",
tkplot=FALSE,expression=NULL, v.size=FALSE, e.size=FALSE,
mst.e.size=1, edge.col.wt=NULL, v.lab=FALSE, e.lab=NULL,
bg="black",v.lab.cex=0.5, e.lab.cex=0.5,v.lab.col="blue",
lab.dist=0, e.lab.col="blue",v.sf=c(3,12), sf=0,
e.arrow=.2, layout.overall=NULL)
```

Arguments

`x`	`x` is a graph object, created using the igraph package.
`layout.function`	`layout.function` is a function class. The user can pass his own function to plot the graph. The input function passed to `layout.function` should return a 2D matrix of coordinates with rows equal to the number of vertices of the input graph and two columns representing x and y coordinates.
`layout.overall`	`layout.overall` is a function class. The user can pass his own function to adjust the abstract view of the graph. The input function passed to `layout.overall` should return a 2D coordinate matrix containing two columns and the number of rows equals the number of vertices in the input network. The user defined function should return a 2D matrix of coordinates with rows equal to the number of vertices of the input graph and two columns representing x and y coordinates.
`mst.edge.col`	This option assigns a color to the edges of the minimum spanning tree of graph 'g'. If mst.edge.col = NULL, then the default color is black for 'tkplot=TRUE'. If 'tkplot=FALSE' it is white.

colors	colors is a vector for the edge colors of the remaining edges in a graph. These remaining edges are colored based on the distance between the nodes connecting them.
vertex.color	vertex.colors is a vector of colors for the vertices of a graph.
expression	expression is a numeric vector which represents properties of genes or proteins, e.g., their mean expression values or p-values. If this option is given, the nodes are colored with a range of colors from red to blue (red for low expression, blue for high expression) depending on the expression values.
tkplot	tkplot is a boolean variable. If it is FALSE, the graph is plotted with the plot function with a black background.
v.size	is a numeric value or a numeric vector which contains values to assign the size of the nodes.
e.size	is a numeric value or a numeric vector which contains values to assign the width of edgesother than the minimum spanning tree edges of the graph.
mst.e.size	is a numeric value which assigns the edge width to the edges of minimum spanning tree of the input graph.
v.lab	is a logical value to show vertex label.
e.lab	is logical value to show edge labels.
bg	is a character value to color the background.
v.lab.cex	is a numerical value to set the font size of the vertex labels.
e.lab.cex	is a numerical value to set the font size of the edge labels.
lab.dist	is a numerical value to adjust the distance of labels from the nodes.
v.lab.col	is a hexadecimal value or a character name of a color to assign colors to the vertex labels.
e.lab.col	is a hexadecimal character value to assign colors to the edge labels.
e.arrow	is a inumerical value to set the arrow width in a directed network.
v.sf	is a 2 dimensional numeric vector, which represents the minimum and maximum limits of the size of the vertices. This is used to adjust the vertex size when the option v.size is used as a vector.
sf	is a numeric value. It is a scaling factor used to scale-up or scale-down the abstract graph.
edge.col.wt	is a vector of numeric values provided for each edge. This is used to color the edges from blue to red from high to low values.
...	... parameter for other inputs.

Value

This function plots the graph object given as an input using 'tkplot' or 'plot' function available in 'igraph' package. This function returns a netbiov class object.

Examples

```
## Example 1 ####
g <- barabasi.game(2000, directed=FALSE)
id <- mst.plot.mod(g)

## Example 2 ####
```

```
  ### plotting a graph by combining two algorithms ##%

  fn <- function(g){layout.reingold.tilford(g,
circular=TRUE, root=which.max(degree(g)))}
  id <- mst.plot.mod(g, v.size=1, sf=-20, layout.function=fn,
layout.overall=layout.fruchterman.reingold, mst.e.size=2,
vertex.color="darkgreen")


  data("PPI_Athalina")
  id <- mst.plot.mod(g1, v.size=1, sf=0, layout.function=fn,
layout.overall=layout.fruchterman.reingold, mst.e.size=1,
vertex.color="magenta", colors=heat.colors(20))


  ## Example 3 ####
  ## When expression values of genes or nodes
## are given and to be plotted as a color of vertices ###
  id <- mst.plot.mod(g1, expression=rnorm(vcount(g1)), v.size=1)

  ## Example 4 ####
  ## When expression values of genes or nodes are given
## and to be plotted as a color of vertices,
## also the degree of nodes to be shown as their vertex-size ###
  id <- mst.plot.mod(g1, expression=rnorm(vcount(g1)),
v.size=degree(g1), v.sf=c(1,5))

  ## Example 5 ####
  ## When MST edges are highlighted in purple color and rest
## of the edges are plotted with a range of heat colors
## depending on the distance between nodes ###
  id <- mst.plot.mod(g1, mst.edge.col="purple",
colors=heat.colors(20), vertex.color="yellow", v.size=1)


  ## Example 6 ####
  ## Plotting a graph with kamada-kawai layout algorithm ###
  id <- mst.plot.mod(g1, mst.edge.col="purple",
colors=heat.colors(20), vertex.color="white", v.size=1,
layout.function=layout.kamada.kawai)


  ## Example 7 ####
  ## Plotting a graph with when weights of edges are given ###
  id <- mst.plot.mod(g1, mst.edge.col="purple", edge.col.wt =
runif(ecount(g1), min=1, max=10), vertex.color="yellow", v.size=
1, layout.function=layout.kamada.kawai)
```

`plot.abstract.module` *Visualization of large biological networks.*

Description

Visualization of graphs in a modular form. In this plot, connecting edges between modules are replaced by a single edge and its width is propotional to the number of connections between modules.

Usage

```
## S3 method for class 'abstract.module'
plot(x, layout.function=NULL,mod.list=NULL,
module.function=FALSE, split.graph=7, color.random=FALSE,
modules.color = NULL, col.grad=NULL, mod.edge.col=NULL,
ed.color=NULL,edge.col.random=FALSE, expression = NULL,
exp.by.module=FALSE, tkplot=FALSE, layout.overall = NULL,
sf=0,arrange.bydegree=FALSE,mod.lab=FALSE,node.lab=FALSE,
lab.cex = NULL,lab.color=NULL, lab.dist=NULL, v.size=FALSE,
nodeset=NULL,path.col="green", col.s1="red", col.s2="yellow",
nodes.on.path=TRUE,e.path.width=1, scale.module=NULL,v.sf=5,
e.width=.5,bg="black", e.sf=15, abstract.graph=TRUE,
modules.name.num=TRUE,  v.size.path=TRUE, ...)
```

Arguments

x	x is a graph object created from an adjacency matrix or from a tabular data of two columns using graph() function available in igraph
layout.function	It is a 'function' class or a vector of functions to plot the layout of each module by a function in 'layout.function'.
mod.list	mod.list is a list object, which provides a modular information about the graph, each components of mod.list contains a vector of nodes to be plotted.
module.function	is a logical value for obtaining modules in the network.
random	It is a boolean variable, if 'mod.list' is null, it picks the nodes for modules randomly.
split.graph	split.graph if random is TRUE, it provides no. of modules that a graph to be split.
color.random	If this option is TRUE it will assign random colors to modules.
modules.color	Is a vector of colors to assign a color to each module by the the user.
col.grad	is a vector of colors or can be a list of vectors of colors to assign the colors to the nodes for each modules based on their degree from low to high.
mod.edge.col	is a vector of color to assign the edge color to the edges of a modules.
ed.color	is a scaler of color and assign colors to the edges between modules.
edge.col.random	is a boolean variable and assign colors to the edges of each module randomly.
expression	expression is a numeric vector which represents properties of genes or proteins like mean expression values or p-values, If this option is given, the nodes are colored with a range of colors from red to blue (red for low expression, blue for high expression) depending on the expression values.
exp.by.module	this option is a boolean or a numeric vector which represents the order of modules given as an input in the mod.list option. This option is used to see the variation in expression values of nodes in a particular module by using a range of color from red to blue, red indicate low expression value and blue indicates high expression values.
tkplot	it is a boolean variable, if it is true, function will use 'tkplot' function to plot a graph, if it is false function will use plot function with the black background.

`layout.overall`	this option belongs to the class 'function', for this option any function which returns a two column matrix which should have rows equal to the no of modules for the placement of the modules.
`sf`	is an integer variable is used to scale up or scale down the graph plot.
`arrange.bydegree`	is a boolean variable; if true the coordinates of nodes are assigned by their degree, higher degree nodes are plotted towards center and lower degree nodes are plotted outside.
`mod.lab`	is a boolean variable; prints module labels at center of each module if the module's names are available in 'mod.list' as list names.
`node.lab`	is a boolean variable; this option plots the vertex label, if option is TRUE.
`lab.cex`	is a numeric variable; this determines the size of the label of the vertices or the modules.
`lab.color`	is a string variable; this assigns colors to the label of vertices or the modules.
`lab.dist`	is a numeric variable; this adjusts vertices label.
`nodeset`	is a list object contains two vector, it can be a numeric vector also. First vector contains set of start nodes and second vector contains end nodes. In the case of numeric vector which indicates the module id, it is used to show shortest path between two modules. Shortest paths are visualized between start nodes and end nodes.
`path.col`	is a color vector for coloring the shortest path between start nodes and end nodes.
`col.s1`	is a color vector to color start nodes or modules.
`col.s2`	is a color vector to color end nodes or modules.
`nodes.on.path`	is a logical value which shows nodes which connect start nodes and end nodes.
`e.path.width`	is a vector containing of size 2. This option sets the edge width of the shortest paths between two modules or two set of nodes.
`scale.module`	is a numeric vector of the size of total number of modules. This option scales the size of each module independently.
`v.size`	is a numeric value or a numeric vector which contains numeric values to assign the size of nodes.
`e.width`	is a numeric value which contains numeric values to assign the width to edges.
`v.sf`	is a numeric value. This is used to adjust vertex size when `v.size` input is a numeric vector.
`bg`	is a color value to adjust background color of the plot.
`e.sf`	is a numeric vector of size 2 to adjust the edge width between a range of minimum and maximum.
`abstract.graph`	is a logical value which adjusts the abstract view of modular plot using force-based algorithm or any input function given in `layout.overall` option.
`modules.name.num`	is a logical value for displaying module name or its number
`v.size.path`	is a numeric or logical value which adjusts the size of nodes between which the shortest path to be shown.
`...`	`...` parameter for other inputs.

Value

returns a list object of 'netbiov' class

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

See Also

plot.abstract.nodes, plot.abstract.module

Examples

```
  data("PPI_Athalina")
  data("modules_PPI_Athalina")

  ## Example 1 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network,
##modules are colored randomly, module information is given as a
##list object ###%
  id <- plot.abstract.module(g1,mod.list = lm,
layout.function=layout.graphopt, color.random = TRUE ,
tkplot=FALSE,node.lab=FALSE,v.size=1)


  ## Example 2 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network,
## modules are colored randomly, module information is  predicted
##using 'fastgreedy' algorithm ###%
  id <- plot.abstract.module(g1,
layout.function=layout.graphopt, color.random = TRUE ,
tkplot=FALSE,node.lab=FALSE,v.size=1)


  ## Example 3 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network
## when expression value of genes are given ###%
  id <- plot.abstract.module(g1, layout.function=layout.graphopt,
 color.random = TRUE,cexpression=rnorm(vcount(g1)), tkplot=FALSE,
node.lab=FALSE,v.size=1)

  ## Example 4 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network
##when expression value of modules 1, 2 and 3 are shown by colors
##by ranking independently from each other ###%
  id <- plot.abstract.module(g1,
layout.function=layout.graphopt, modules.color="grey",
expression=rnorm(vcount(g1)), tkplot=FALSE,node.lab=FALSE,v.size=
1, exp.by.module=c(1,2,3))

  ## Example 5 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network
##by emphasizing module labels ###%
  id <- plot.abstract.module(g1, mod.list=lm,
layout.function=layout.graphopt, modules.color="grey",
tkplot=FALSE, mod.lab=TRUE  ,v.size=1, lab.color="green" )
```

```
  ## Example 6 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network
##highlighting shortest paths between modules 1, 5 and 7, 18 ###%
  id <- plot.abstract.module(g1, mod.list=lm,
layout.function=layout.graphopt, modules.color="grey",
tkplot=FALSE, nodeset=c(1,5,7,18), sf=-10, v.size=1)

  ## Example 7 #######%
  ##% Abstract modular layout plot of A. Thalina PPI network
##combining two layouts ###%
  fn <- function(g)layout.star(g,
center=which.max(degree(g))-1)
  id <- plot.abstract.module(g1,mod.list = lm,
layout.function=layout.graphopt, layout.overall=fn,
color.random = TRUE , tkplot=FALSE,node.lab=FALSE,v.size=1)
```

`plot.abstract.nodes` *Exploratory visualization of information spread of biological networks*

Description

Modular layout style:

Visualization of modules in an abstract way. In the function `plot.abstract.nodes`, nodes in a module are replaced by a single node and the relative size of a node is proportional to the total number of nodes in a module. In the functions `plot.abstract.module`, `plot.abstract.nodes`, edges between two modules are replaced by a single edge and the total number of edges between two modules are reflected by the relative edge-width.

Usage

```
## S3 method for class 'abstract.nodes'
plot(x, mod.list=NULL,  rest.module=TRUE,
color.random=FALSE, nodes.color = NULL, edge.colors=NULL,
layout.function=NULL,tkplot=FALSE,v.sf = 0, e.sf = 0,lab.color=NULL,
lab.cex=NULL, lab.dist=NULL, bg="black", ... )
```

Arguments

`x`	`x` is a graph object created from an adjacency matrix or from a tabular data of two columns using `graph()` function available in `igraph`
`mod.list`	`mod.list` is a list object each components of mod.list consists a vector of module to be plotted.
`random`	is a logical value, if modules information is not provided then random nodes for the modules are selected.
`rest.module`	is a logical value, rest of the nodes to be plotted as a one module (if TRUE) or independently (if FALSE)
`color.random`	is a logical value, if this option is TRUE it will assign random color to each module or nodes.

nodes.color	assigns colors to nodes of recreated graph
edge.colors	is a vector of colors, assigned to color edges depending on their width from small to large width.
layout.function	is a function class, this corrosponds to a layout function given as an input to plot a graph. Input function should return a two column matrix which has same no of rows as the modules in the graph
tkplot	is a boolean variable, if it is true function will use this option to plot a graph, if it is false function will use plot function with a black background.
v.sf	is a numeric value, this option to increase or decrease the size of nodes
e.sf	is a numeric value, this option to increase or decrease width of edges
lab.color	colors the vertices and edge labels
lab.cex	is anumeric value, increase or decrease the font size of labels
lab.dist	is a numeric value, adjust vertex labels
bg	is a color value to adjust background color of the plot.
...	... parameter for other inputs.

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
  require(igraph)
  data("PPI_Athalina")
  data("modules_PPI_Athalina")
  plot.abstract.nodes(g1, mod.list = lm,
edge.colors = c("red","green","blue","orange"))
```

---

plot.modules *Visualization of large biological networks.*

Description

Modular layout style:

Visualization of graphs in a modular form.

Usage

```
## S3 method for class 'modules'
plot(x, layout.function=NULL,mod.list=NULL,
module.function=FALSE, split.graph=7, color.random=FALSE,
modules.color = NULL, col.grad=NULL, mod.edge.col=NULL,
ed.color=NULL,edge.col.random=FALSE, expression = NULL,
exp.by.module=FALSE, tkplot=FALSE, layout.overall = NULL,
sf=0,arrange.bydegree=FALSE,mod.lab=FALSE,node.lab=FALSE,
lab.cex = NULL,lab.color=NULL, lab.dist=NULL, v.size=FALSE,
nodeset=NULL,path.col="green", col.s1="red", col.s2="yellow",
nodes.on.path=TRUE,e.path.width=c(1,1),
scale.module=NULL,v.sf=5,e.width=.5,bg="black",
abstract.graph=TRUE, modules.name.num = TRUE, v.size.path=TRUE,...)
```

Arguments

x	x is a graph object created from an adjacency matrix or from a tabular data of two columns using graph() function available in igraph
layout.function	is a 'function' class or a vector of functions to plot the layout of each module by a function in 'layout.function'.
mod.list	mod.list is a list object, which provides a modular information about the graph, each components of mod.list contains a vector of nodes to be plotted.
module.function	is a logical value for obtaining modules in the network.
random	is a boolean variable, if 'mod.list' is null, it picks the nodes for modules randomly.
split.graph	split.graph if random is TRUE, it provides no. of modules that a graph to be split.
color.random	If this option is TRUE it will assign random colors to modules.
modules.color	Is a vector of colors to assign a color to each module by the the user.
col.grad	is a vector of colors or can be a list of vectors of colors to assign the colors to the nodes for each modules based on their degree from low to high.
mod.edge.col	is a vector of color to assign the edge color to the edges of a modules.
ed.color	is a scaler of color and assign colors to the edges between modules.
edge.col.random	is a boolean variable and assign colors to the edges of each module randomly.
expression	expression is a numeric vector which represents properties of genes or proteins like mean expression values or p-values, If this option is given, the nodes are colored with a range of colors from red to blue (red for low expression, blue for high expression) depending on the expression values.
exp.by.module	this option is a boolean or a numeric vector which represents the order of modules given as an input in the mod.list option. This option is used to see the variation in expression values of nodes in a particular module by using a range of color from red to blue, red indicate low expression value and blue indicates high expression values.

tkplot	it is a boolean variable, if it is true, function will use 'tkplot' function to plot a graph, if it is false function will use plot function with the black background.
layout.overall	this option belongs to the class 'function', for this option any function which returns a two column matrix which should have rows equal to the no of modules for the placement of the modules.
sf	is an integer variable is used to scale up or scale down the graph plot.
arrange.bydegree	is a boolean variable; if true the coordinates of nodes are assigned by their degree, higher degree nodes are plotted towards center and lower degree nodes are plotted outside.
mod.lab	is a boolean variable; prints module labels at center of each module if the module's names are available in 'mod.list' as list names.
node.lab	is a boolean variable; this option plots the vertex label, if option is TRUE.
lab.cex	is a numeric variable; this determines the size of the label of the vertices or the modules.
lab.color	is a string variable; this assigns colors to the label of vertices or the modules.
lab.dist	is a numeric variable; this adjusts vertices label.
nodeset	is a list object contains two vector, it can be a numeric vector also. First vector contains set of start nodes and second vector contains end nodes. In the case of numeric vector which indicates the module id, it is used to show shortest path between two modules. Shortest paths are visualized between start nodes and end nodes.
path.col	is a color vector for coloring the shortest path between start nodes and end nodes.
col.s1	is a color vector to color start nodes or modules.
col.s2	is a color vector to color end nodes or modules.
nodes.on.path	is a logical value which shows nodes which connect start nodes and end nodes.
e.path.width	is a vector containing of size 2. This option sets the edge width of the shortest paths between two modules or two set of nodes.
scale.module	is a numeric vector of the size of total number of modules. This option scales the size of each module independently.
v.size	is a numeric value or a numeric vector which contains numeric values to assign the size of nodes.
e.width	is a numeric value to assign the width to edges.
v.sf	is a numeric value. This is used to adjust vertex size when v.size input is a numeric vector.
bg	is a color value to adjust background color of the plot.
abstract.graph	is a logical value which adjusts the abstract view of modular plot using force-based algorithm or any input function given in layout.overall option.
modules.name.num	is a logical value for displaying module name or its number
v.size.path	is a numeric or logical value which adjusts the size of nodes between which the shortest path to be shown.
...	... parameter for other inputs.

Value

returns a list object of 'netbiov' class

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

See Also

plot.abstract.nodes, plot.abstract.module

Examples

```
  data("PPI_Athalina")
  data("modules_PPI_Athalina")

  ## Example 1 #######%
  ##% Modular layout plot of A. Thalina PPI network, modules
##are colored randomly, module information is given as a list
##object ###%
  id <- plot.modules(g1,mod.list = lm,
layout.function=layout.graphopt, color.random = TRUE ,
tkplot=FALSE,node.lab=FALSE,v.size=1)

  ## Example 2 #######%
  ##% Modular layout plot of A. Thalina PPI network, modules
##are colored randomly, module information is  predicted using
##'fastgreedy' algorithm ###%
  id <- plot.modules(g1, layout.function=layout.graphopt,
color.random = TRUE , tkplot=FALSE,node.lab=FALSE,v.size=1)

  ## Example 3 #######%
  ##% Modular layout plot of A. Thalina PPI network when
##expression value of genes are given ###%
  id <- plot.modules(g1, layout.function=layout.graphopt,
color.random = TRUE, expression=rnorm(vcount(g1)),
tkplot=FALSE,node.lab=FALSE,v.size=1)

  ## Example 4 #######%
  ##% Modular layout plot of A. Thalina PPI network when
##expression value of modules 1, 2 and 3 are shown by colors by
##ranking independently from each other ###%
  id <- plot.modules(g1, layout.function=layout.graphopt,
modules.color="grey", expression=rnorm(vcount(g1)),
tkplot=FALSE,node.lab=FALSE,v.size=1, exp.by.module=c(1,2,3))

  ## Example 5 #######%
  ##% Modular layout plot of A. Thalina PPI network by
##emphasizing module labels ###%
  id <- plot.modules(g1, mod.list=lm,
layout.function=layout.graphopt, modules.color="grey",
tkplot=FALSE, mod.lab=TRUE,v.size=1, lab.color="green" )

  ## Example 6 #######%
  ##% Modular layout plot of A. Thalina PPI network
##highlighting shortest paths between modules 1, 5 and 7, 18 ###%
```

```
  id <- plot.modules(g1, mod.list=lm,
layout.function=layout.graphopt, modules.color="grey",
tkplot=FALSE, nodeset=c(1,5,7,18), sf=-10, v.size=1)

  ## Example 7 #######%
  ##% Modular layout plot of A. Thalina PPI network combining
##two layouts ###%
  fn <- function(g)layout.star(g,
center=which.max(degree(g))-1)
  id <- plot.modules(g1,mod.list = lm,
layout.function=layout.graphopt, layout.overall=fn, color.random
= TRUE , tkplot=FALSE,node.lab=FALSE,v.size=1)

  ## Example 8 #######%
  ##% Modular layout plot of A. Thalina PPI network by scaling
##up second module and scaling down the expansion of other modules ###%
  fn <- function(g)layout.star(g,
center=which.max(degree(g))-1)
  sm <- rep(1, length(lm))
  sm[2] <- 40
  id <- plot.modules(g1,mod.list = lm, layout.function=layout.graphopt,
layout.overall=fn, color.random=TRUE,tkplot=FALSE,v.size=2, scale.module=sm,
mod.edge.col="green")
```

`plot.netbiov` *Visualization of large biological networks.*

Description

Plots netbiov object

Usage

```
## S3 method for class 'netbiov'
plot(x, ...)
## S3 method for class 'netbiov'
tkplot(x, ...)
```

Arguments

`x`	`x` is a netbiov object contains different properties of graph
`...`	`...` parameter for other inputs.

Value

returns invisible 'NULL' value.

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
data("PPI_Athalina")
id <- plot.modules(g1)
plot.netbiov(id)
```

`plot.NetworkSperical` *Exploratory visualization of information spread of biological networks*

Description

Global layout style:

A visualization of a network in a spherical form. The node with the highest degree is placed in the center, and its neighbors are plotted around this node in a circular manner. Whenever a node is encountered with multiple neighbors, the neighbors are plotted into the direction of that node. This process continues until all nodes are placed. This gives a compact spherical view of the network.

Usage

```
## S3 method for class 'NetworkSperical'
plot(x, mo="in", tkplot = FALSE,
v.lab=FALSE, v.size=1, bg="black", ...)
```

Arguments

`x`	`x` is a graph object created from an adjacency matrix or from a tabular data of two columns using `graph()` function available in `igraph`.
`mo`	`mo` represents the mode of nodes. Can be either `in` or `out`.
`tkplot`	it is a boolean variable, if it is true, function will use 'tkplot' function to plot a graph, if it is false function will use plot function with the black background.
`v.lab`	`v.lab` is a logical value to show vertex label.
`v.size`	`v.size` is a numeric value to assign the size of nodes.
`bg`	bgis a color value to adjust background color of the plot.
`...`	`...` parameter for other inputs.

Value

Plots the input graph object using `tkplot` function.

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
#Example 1
 g <- barabasi.game(500, directed = TRUE)
 xx <- plot.NetworkSperical(g,  mo = "in", tkplot=FALSE)

# Example 2
g <- erdos.renyi.game(100, p=.1)
xx <- plot.NetworkSperical(g)
```

`plot.NetworkSperical.startSet`

Exploratory visualization of information spread of biological networks

---

Description

Global layout style:

A visualization of network in a 'star-like' form. An initial set of nodes are placed in the corner of polygons. The neighbors are then plotted above these nodes in an arc form. Whenever a node is encountered with multiple neighbors, neighbors are plotted in the direction of that node in an arc form. This process continues until all the nodes are placed. This gives a star-like view of the network.

Usage

```
## S3 method for class 'NetworkSperical.startSet'
plot(x, mo="in", nc=5, tkplot=FALSE,
v.lab=FALSE, v.size=2, bg="black", ...)
```

Arguments

`x`	`x` is a graph object created from an adjacency matrix or from a tabular data of two columns using `graph()` function available in `igraph`.
`mo`	`mo` represents the mode of nodes. Options are `in` or `out`.
`nc`	`nc` is a numeric value representing the total number of corners to be plotted, initially at the center position of a polygon.
`tkplot`	it is a boolean variable, if it is true, function will use 'tkplot' function to plot a graph, if it is false function will use plot function with the black background.
`...`	`...` parameter for other inputs.
`v.lab`	`v.lab` is a logical value to show vertex label.
`v.size`	`v.size` is a numeric value to assign the size of nodes.
`bg`	bgis a color value to adjust background color of the plot.

Value

Plots the input graph object using `tkplot` function.

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
 n <- 500
g <- barabasi.game(n, directed = TRUE)
plot.NetworkSperical.startSet(g, mo = "in", nc = 5, tkplot=FALSE)
```

---

plot.spiral.graph *Exploratory visualization of information spread of biological networks*

Description

Plots a graph in a spirical fashion, highly connected nodes are placed at center.

Usage

```
## S3 method for class 'spiral.graph'
plot(x, tp=61, vertex.color=NULL,
color.random=FALSE, rank.function=NULL,tkplot=FALSE,
v.size=2, e.size=.5,e.curve=.5, v.lab=FALSE, bg="black",
e.col="grey", skip=0, ...)
```

Arguments

x	x is a graph object created from an adjacency matrix or from a tabular data of two columns using graph() function available in igraph
tp	is a numeric value, a tuning parameter to get different spirical shapes.
color.random	is alogical value, if TRUE, picks colors randomly and assign to nodes depending on their degree
rank.function	is a 'function' class returns a two column matrix with number of rows equal to the total nodes in the graph.
tkplot	it is a boolean variable, if it is true, function will use 'tkplot' function to plot a graph, if it is false function will use plot function with the black background.
v.size	is a numeric value or a numeric vector which contains numeric values to assign the size of nodes.
e.size	is a numeric value to assign the width to edges.
e.curve	is a positive numeric value that adjusts the curvature of the edges.
v.lab	is a logical value to show vertex label.
bg	name of a color or a hexadecimal code of a color. This option is used to color the background of the plot. The default option is "black".
e.col	is a character or a hexadecimal color value to set the color of edges.

vertex.color	is a vector of colors assign colors to nodes depending on their degree, from high to low.
skip	is an integer value to plot nodes away from the center.
...	... parameter for other inputs.

Details

This layout function plots network in a spiral fashion.

Value

returns a list object of 'netbiov' class

Author(s)

Shailesh Tripathi, Frank Emmert-Streib

References

http://bio-complexity.com/

Examples

```
  g <- barabasi.game(500)
  x <- plot.spiral.graph(g, 121 )
  x <- plot.spiral.graph(g, 120,rank.function=layout.reingold.tilford,
vertex.color="red", e.col="green")
```

PPI_Athalina *An igraph object of PPI interaction of A. thalina*

Description

A connected subnetwork of PPI interaction of A. thalina.

Usage

```
data(PPI_Athalina)
```

Format

igaph class object

Examples

```
data("PPI_Athalina")
#xx <- plot.modules(g1)
```

splitg.mst *Network plot*

Description

Modular layout style:

A large complex network is plotted by splitting it into its modules. The positions of the vertices in each subnetwork are determined by using the fruchterman-reingold algorithm or the Kamada-kawai algorithm for the minimum spanning tree of each subnetwork. The edges of the minimum spanning tree are shown in black color.

Usage

```
splitg.mst(x,layout.function=NULL, mod.list=NULL,
colors=NULL,mst.edge.col="white",
vertex.color = c("red","green","blue","orange"),
random.v.color=FALSE,in.con.ed.col=NULL,tkplot=FALSE,
v.size=2, e.size=.5, mst.e.size=1, v.lab=FALSE,
bg="black", v.lab.cex=0.5, e.lab.cex=0.5,
v.lab.col="skyblue", lab.dist=0, v.sf=4, sf.modules = 5, ...)
```

Arguments

x	x is a graph object, created using igraph package.
layout.function	is a 'function' class or a vector of functions to plot the layout of each module by a function in 'layout.function'.
mod.list	mod.list is a list object, which provides a modular information about the graph, each components of mod.list contains a vector of nodes to be plotted.
v.size	is a numeric value or a numeric vector which contains values to assign the size of the nodes.
e.size	is a numeric value to assign the width to edges.
e.lab.cex	is a numeric variable; this determines the size of the labels of the vertices or the modules.
lab.dist	is a numeric variable; this adjusts labels of vertices.
sf.modules	is an integer variable is used to scale up or scale down the graph plot.
v.sf	is a numeric value. This is used to adjust vertex size when v.size input is a numeric vector.
mst.edge.col	This option assigns a color to the edges of the minimum spanning tree of each module of graph 'g'. The default color is black for 'tkplot=TRUE'. If 'tkplot=FALSE' it is white.
random	random is a logical value, this option is used to choose nodes of split graphs randomly
colors	colors is a vector of colors. This option is a vector of the edge colors to assign colors to the edges of the graph.

vertex.color	vertex.colors is a vector of colors to assign colors to the vertices of the modules of the graph.
random.v.color	is a logical value, this option is used to assign colors to the vertices colors of the modules, colors for modules are picked randomly.
in.con.ed.col	is a scaler, assign colors to the edges which are showing connections between the modules.
tkplot	it is a boolean variable, if it is true function will use tkplot function to plot a graph, if it is false function will use plot function with a black background.
mst.e.size	is a numeric value which assigns the edge width to the edges of minimum spanning tree of the input graph.
v.lab	is a logical value to show vertex label.
v.lab.cex	is a numerical value to set the font size of vertex labels.
v.lab.col	is a hexadecimal character value to assign colors to vertex labels.
sf	is a numeric value. This is used to adjust vertex size when v.size input is a numeric vector.
bg	is a color value for background.
...	... parameter for other inputs.

Value

This function plots a graph using 'tkplot' function available in the 'igraph'. This function returns a list,first component of list is a graph object, second component of the list contains x and y coordinates, third component of list contains color ids of edges of the graph etc.

Examples

```
data("PPI_Athalina")
data("modules_PPI_Athalina")
      id <- splitg.mst(g1, mod.list=lm, random.v.color=TRUE, tkplot=FALSE )
```

Index