Package ‘RedeR’

April 4, 2024

Type  Package
Title  Interactive visualization and manipulation of nested networks
Version  2.6.1
Author  Mauro Castro, Xin Wang, Florian Markowetz
Maintainer  Mauro Castro <mauro.a.castro@gmail.com>
Depends  R (>= 4.0), methods
Imports  igraph
Suggests  BiocStyle, knitr, rmarkdown, markdown, TreeAndLeaf
SystemRequirements  Java Runtime Environment (Java>= 11)
Description  RedeR is an R-based package combined with a stand-alone Java application for interactive visualization and manipulation of nested networks.
License  GPL (>= 3)
biocViews  GUI, GraphAndNetwork, Network, NetworkEnrichment, NetworkInference, Software, SystemsBiology
VignetteBuilder  knitr
URL  http://genomebiology.com/2012/13/4/R29
LazyLoad  yes
git_url  https://git.bioconductor.org/packages/RedeR
git_branch  RELEASE_3_18
git_last_commit  0a91264
git_last_commit_date  2024-03-10
Repository  Bioconductor 3.18
Date/Publication  2024-04-03

R topics documented:

 RedeR-package .......................................................... 3
 .rederpost ............................................................. 3
 addEdgeBetweenContainers ........................................ 4
R topics documented:

addEdges ................................................................. 5
addGraph ............................................................... 6
addLegend ............................................................... 10
addNodes ................................................................. 12
att .......................................................................... 13
calld ....................................................................... 15
cea ......................................................................... 16
deleteEdges ............................................................. 18
deleteNodes ............................................................. 19
deleteSelectedEdges .................................................. 20
deleteSelectedNodes .................................................. 21
deSelectEdges .......................................................... 22
deSelectGraph .......................................................... 23
deSelectNodes .......................................................... 24
exitd ......................................................................... 25
getContainerComponets .............................................. 26
getEdgeIDs .............................................................. 27
getEdges ................................................................. 28
getGraph ................................................................. 29
getNodeIDs .............................................................. 30
getNodes ................................................................. 32
getSourceEdgeIDs ..................................................... 33
getTargetEdgeIDs ..................................................... 34
gtoy.rm ................................................................. 35
isRelaxActive ........................................................... 36
mergeNodes ............................................................. 37
mergeOutEdges .......................................................... 38
nesthc ..................................................................... 39
nestNodes ............................................................... 41
ping ......................................................................... 43
RedeR.data .............................................................. 44
RedPort ................................................................. 45
RedPort-class ........................................................... 46
relax ......................................................................... 48
resetd ..................................................................... 49
selectAllEdges .......................................................... 50
selectAllNodes .......................................................... 51
selectEdges ............................................................. 52
selectGraph ............................................................ 53
selectNodes ............................................................ 54
setArrowDirection ...................................................... 55
startRedeR ............................................................... 56
subg ......................................................................... 57
updateContainerSize .................................................. 58
updateCoordXY ........................................................ 59
updateGraph ........................................................... 60
version ................................................................. 61

Index ................................................................. 63
RedeR-package

RedeR: bridging the gap between network analysis and visualization.

Description

RedeR is an R-based package combined with a stand-alone Java application for interactive visualization and manipulation of modular structures, nested networks and multiple levels of hierarchical associations. The software takes advantage of R to run robust statistics, while the R-to-Java interface bridges the gap between network analysis and visualization.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>RedeR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>License:</td>
<td>GPL</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Mauro Castro <mauro.a.castro@gmail.com>

References


See Also

RedPort-class

.rederpost

This function implements an RPC protocol for RedeR.

Description

Post to RedeR’s handler at the Java side.

Usage

.rederpost(obj, method, ..., gdata=list(...))
**addEdgeBetweenContainers**

**Arguments**

- **obj** Object of RedPort Class.
- **method** A method listed in RedeR’s handler at the Java side.
- **...** Additional arguments passed to the Java side.
- **gdata** A list of graph options passed to RedeR’s handler at the Java side.

**Value**

Post to RedeR’s handler at the Java side.

**Note**

This is an internal function.

**Author(s)**

Mauro Castro

**See Also**

RedPort

---

### addEdgeBetweenContainers

*Add edges between containers.*

---

**Description**

Method to add edges between RedeR containers. This method adds non-nested assignments, in contrast to the default behavior that builds nested associations to-and-from containers.

**Usage**

```
addEdgeBetweenContainers(obj, containerA, containerB)
```

**Arguments**

- **obj** Object of RedPort Class.
- **containerA** <string>
- **containerB** <string>

**Value**

Add graph objects.
addEdges

Note
Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').

Author(s)
Mauro Castro

See Also
RedPort

Examples

```r
# Initialize igraph
library(igraph)
el<-matrix(c('n1','n2','n3','n4'), ncol=2, byrow=TRUE)
g <- graph.edgelist(el)

## Not run:
rdp <- RedPort()
callld(rdp)
addGraph( rdp, g, layout.kamada.kawai(g) )
nestNodes( rdp, c('n1','n2') )
nestNodes( rdp, c("n3","n4") )
addEdgeBetweenContainers(rdp, "N0", "N1")
updateGraph(rdp)

## End(Not run)
```

addEdges  

Add edges to RedeR graphs.

Description
Add edges to an active RedeR session.

Usage
addEdges(obj, edges)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object of RedPort Class.</td>
</tr>
<tr>
<td>edges</td>
<td>A vertex sequence &lt;vector of strings&gt; or data frame of ncol=2.</td>
</tr>
</tbody>
</table>
Value

Adds the specified edges to the graph.

Note

Prior calling this method invoke RedeR application via 'calld' function.

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

# edges as a vertex sequence
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

# edges as a data.frame
edges <- data.frame(A=c("n1","n1","n1","n1","n1"),
                    B=c("n2","n3","n4","n5","n6","n7"))

## Not run:
rdp <- RedPort()
calld(rdp)
addEdges(rdp, edges)
updateGraph(rdp)

## End(Not run)
**Arguments**

- **obj**: Object of RedPort Class.
- **g**: An igraph object.
- ... Additional arguments passed to RedeR application.

**Details**

Additional arguments:

- **layout**: Vertex coordinates (graph layout). Accepts matrix with 2 cols (i.e. x and y coords) \(<\text{matrix}>\). Note, xy coords are not updated when "layout=NULL". This allows graph attributes to be updated, keeping the same layout of an active graph.
- **gscale**: Expansion factor of the graph area related to the app panel area (default = 75) \(<\text{numeric}>\).
- **gcoord**: Sets the graph x,y center. Coords between 0 and 100 are set to the visible area of the app panel (default = c(50,50)) \(<\text{numeric vector}>\).
- **zoom**: Sets zoom scale for the app panel (range: 0.0 to 100.0; default = NULL) \(<\text{numeric}>\).
- **gzoom**: Sets zoom scale only for objects on the app panel (range: 0.0 to 100.0; default = NULL) \(<\text{numeric}>\). This option overrides ‘zoom’.
- **isNest**: Logical value, whether to nest all nodes into a new container (default = FALSE). See additional args in `nestNodes`.
- **isAnchor**: If isNest=TRUE, this logical value sets whether to anchor the container in dynamic layouts (default = TRUE).
- **isAssign**: If isNest=TRUE, this logical value sets whether to assign the container name to the nested nodes (default = FALSE).
- **loadEdges**: Logical value, whether to send edges to RedeR app (default = TRUE).
- **ntransform**: Logical value, whether to transform nodes into containers (default = FALSE).
- **parent**: ID of a container already available in the app \(<\text{string}>\). Nodes from ‘g’ will be nested to this container.
- **theme**: Some pre-defined nest attributes. Options: ‘tm0’, ‘tm1’, ‘tm2’, ‘tm3’, ‘tm4’, ‘tm5’, ‘tm6’ \(<\text{string}>\). Alternatively, it can be a list with customized attributes.

**Value**

Send R graphs to RedeR app.

**Attributes passed by the igraph object**

**Graph attributes:**

- **bgColor**: Sets the background color of the app panel \(<\text{hexadecimal}>\).
- **zoom**: Sets the zoom scale for the app panel (range: 0.0 to 100.0) (Default=100) \(<\text{numeric}>\).
- **gzoom**: Sets zoom scale only for objects on the app panel (range: 0.0 to 100.0) (Default=NULL) \(<\text{numeric}>\). This option overrides ‘zoom’. Obs. this is an alternative entry to the ‘gzoom’ argument above.
gscale  Expansion factor of the graph area related to the app panel (range: 0.0 to 100.0) (Default=100) <numeric>. Obs. this is an alternative entry to the 'gscale' argument above.

isNest  Logical value, whether to nest all nodes into a new container (Default=FALSE). Obs. this is an alternative entry to the 'nest' argument above.

isAnchor  If isNest=TRUE, this logical value sets whether to anchor the container in dynamic layouts (Default=FALSE).

isAssign  If isNest=TRUE, this logical value sets whether to assign the container name to the nested nodes (Default=FALSE).

nestAlias  If isNest=TRUE, this attribute sets the label of the new container <string>.

nestSize  If isNest=TRUE, this attribute sets the size of the container (Default=NULL) <numeric>.

nestShape  If isNest=TRUE, this attribute sets the shape of the container, options: 'ELLIPSE', 'RECTANGLE', 'ROUNDED_RECTANGLE', 'TRIANGLE', 'DIAMOND' (Default=ELLIPSE) <string>.

nestColor  If isNest=TRUE, this attribute sets the 'color' of the new container <hexadecimal>.

nestFontSize  If isNest=TRUE, this attribute sets the size of the container label (Default=18) <numeric>.

nestFontColor  If isNest=TRUE, this attribute sets the 'color' of the container label <hexadecimal>.

nestLineWidth  If isNest=TRUE, this attribute sets the line width of the container, options: >= 0 (Default=1.0) <numeric>.

nestLineColor  If isNest=TRUE, this attribute sets the line color of the container <hexadecimal>.

nestLineType  If isNest=TRUE, this attribute sets the line type of the container: '<SOLID>', '<DOTTED>', '<DASHED>', '<LONG_DASH>' (Default='SOLID').

nestImage  If isNest=TRUE, sets the status of the container on the screen: '<plain>', '<transparent>', or '<hide>' (Default=plain).

Vertex attributes:

name  Node attribute 'name' <string>.

nodeAlias  Node attribute 'alias' <string>.

tabsX  Node attribute 'x coord' (Default=random coord) <numeric>.

tabsY  Node attribute 'y coord' (Default=random coord) <numeric>.

nodeSize  Node attribute 'size', options: >= 0 (Default=20) <numeric>.

nodeShape  Node attribute 'shape', options: 'ELLIPSE', 'RECTANGLE', 'ROUNDED_RECTANGLE', 'TRIANGLE', 'DIAMOND' (Default=ELLIPSE) <string>.

nodeColor  Node attribute 'color', e.g. "#ff0000" for red <hexadecimal>.

tabsLineWidth  Node attribute 'line width', options: >= 0 (Default=1) <numeric>.

nodeLineColor  Node attribute 'line color', e.g. "#ff0000" for red <hexadecimal>.

nodeFontSize  Node attribute 'font size', options: >= 0 (Default=12) <integer>.

nodeFontColor  Node attribute 'font color', e.g. "#ff0000" for red <hexadecimal>.

nodeBend  Node attribute 'bend', options: 0-100% (Default=50) <numeric>.

nodeWeight  Node attribute 'weight', options: >= 0 (Default=0) <numeric>.

Edge attributes:

edgeWeight  Edge attribute 'weight', options: >= 0 (Default=0.0) <numeric>.
**addGraph**

**edgeWidth**  Edge attribute ‘width’, options: $\geq 0$ (Default=1.0) <numeric>.

**edgeColor**  Edge attribute ‘color’, e.g. “#ff0000” for red <hexadecimal>.

**edgeType**  Edge attribute ‘type’, options: 'SOLID', 'DOTTED', 'DASHED', 'LONG_DASH' (Default='SOLID').

**arrowLength**  Edge arrow attribute ‘length’, options: $\geq 0$ (Default=15) <numeric>.

**arrowAngle**  Edge arrow attribute ‘angle’, options: 10-75 degrees (Default=20) <numeric>.

**arrowType**  Edge arrow attribute ‘type’, used to set the associations in directed graphs. Options: -1, 0 and 1 (Default=1) <integer>.

**arrowDirection**  Edge arrow attribute ‘direction’, used to set the associations in undirected graphs. Options: 0 (A-B), 1 (A -> B), -1 (A | B), 2 (A < B), -2 (A |- B), 3 (A <-> B), -3 (A |< B), 4 (A |< B) and -4 (A <- | B) (Default=0) <integer>.

**linkType**  Set assignment type either between nodes and containers or containers and containers. Options: 'nested' and 'notnested' (Default='nested') <string>.

**Note**

In ‘igraph’ package, vertex and edge attributes can be assigned as arbitrary R objects. In order to pass these extensible features to RedeR the attributes must be provided in a valid syntax (see above). Only UNIQUE edges are accepted. If present, mutual/multiple edges will be collapsed to unique edges. In this cases, source-target information is transferred to ‘arrowDirection’ attribute; other attributes will be related to the first edge from the edge list.

**Author(s)**

Mauro Castro

**See Also**

getGraph addLegend nesthc nestNodes mergeOutEdges relax selectNodes att

**Examples**

```r
# Initialize igraph
library(igraph)

## Not run:
rdp <- RedPort()
callld(rdp)

###
g1 <- igraph::graph.empty(n=10, directed=FALSE)
addGraph( rdp, g1, layout.random(g1) )
resetd(rdp)

###
g2 <- igraph::graph.lattice(c(5,5,5))
```
addLegend

Add graph legends to RedeR application.

Description

Methods to send legends to RedeR app.

Usage

addLegend.color(obj, colvec, ...)
addLegend.size(obj, sizevec, ...)
addLegend.shape(obj, shapevec, ...)

Arguments

obj Object of RedPort Class.
colvec Vector with legend colors, either hexadecimal or valid R color names.
sizevec Vector with legend node size, options: > 0 <numeric>.
shapevec Vector with valid shape names: 'ELLIPSE', 'RECTANGLE', 'ROUNDED_RECTANGLE', 'TRIANGLE', 'DIAMOND'.
... Additional arguments passed to RedeR application.
addLegend

Details

Alternatively, colvec, sizevec and shapevec can be igraph objects with legend information previously set by the functions ‘att.setv’ and ‘att.sete’.

Additional arguments:

- **type** Legend type. Options: "node" or "edge" (default: "node") <character>.
- **labvec** Vector with legend labels <character>.
- **position** Position of the legend in RedeR panel. Options: 'topleft', 'topright', 'bottomleft', 'bottomright' (default: addLegend.color "topright", addLegend.size "bottomleft", and addLegend.shape "bottomright") <character>.
- **dxborder** Distance (in pixel) from panel border (default: 5) <numeric>.
- **dyborder** Distance (in pixel) from panel border (default: 5) <numeric>.
- **vertical** Logical value, set vertical/horizontal position of the legend in the app panel (default: TRUE for addLegend.color and addLegend.size and FALSE for addLegend.shape.
- **ftsize** Font size (in pixel) (default: 12) <numeric>.
- **title** Legend title <string>.
- **dxttitle** Distance (in pixel) from legend title to the main axis (adjusted to nchar) (default: 5) <numeric>.
- **size** Legend size; only for addLegend.color and addLegend.shape methods (default: 20) <numeric>.
- **bend** Legend width/height ratio; only for addLegend.color method (default: 0.85) <numeric>.
- **intersp** Legend inter space (only for addLegend.size and addLegend.shape methods) (default: 5) <numeric>.
- **edgelen** Length of the edges in addLegend.size method (default: 50) <numeric>.

Value

Send legend objects to RedeR app.

Author(s)

Mauro Castro

Examples

```r
# Initialize igraph
library(igraph)

## Not run:
rdp <- RedPort()
calld(rdp)

cols<-colorRampPalette(colors=c('red','blue'))(14)
addLegend.color(rdp,cols)
addLegend.color(rdp,cols,type="edge")
```
size<-c(10,20,30,40,50)
addLegend.size(rdp,size)

size<-c(1:10)
addLegend.size(rdp,size,type="edge")

shape<-c('ELLIPSE', 'RECTANGLE', 'ROUNDED_RECTANGLE', 'TRIANGLE', 'DIAMOND')
addLegend.shape(rdp,shape)

shape<-c('SOLID','DOTTED','DASHED','LONG_DASH')
addLegend.shape(rdp,shape,type="edge")

## End(Not run)

---

**addNodes**

*Add nodes to RedeR graphs.*

**Description**

Method to add nodes to an active RedeR session.

**Usage**

```
addNodes(obj, nodes)
```

**Arguments**

- `obj` Object of RedPort Class.
- `nodes` Node sequence as an array <array of strings>

**Value**

Add graph objects.

**Note**

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').

**Author(s)**

Mauro Castro

**See Also**

RedPort
Examples

# Initialize igraph
library(igraph)

nodes<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
 rdp <- RedPort()
callid(rdp)
addNodes(rdp, nodes)
updateGraph(rdp)

## End(Not run)

att

Add, map, and set edge and vertex attributes to the RedeR application.

Description

Given an 'igraph' object 'g', 'att.addv' adds a new attribute with a fixed 'value' to all nodes or
selected nodes using 'filter', while 'att.adde' adds a new attribute with a fixed 'value' to all edges.
The 'att.mapv' and 'att.mape' functions map data frames to the 'g' object, useful for adding batches
of attributes.

The 'att.setv' and 'att.sete' functions set attributes available in the 'g' object, transforming them
into new attribute classes (for examples, numeric values into colors or sizes).

Usage

att.addv(g, to, value, filter = NULL, index = V(g))
att.adde(g, to, value, index = E(g))
att.mapv(g, dat, refcol=1)
att.mape(g, dat, refcol=c(1,2))
att.setv(g, from, to='nodeColor', pal=1, cols=NULL, na.col=grey(0.7),
xlim=c(20,100,1), shapes=NULL, breaks=NULL, categvec=NULL, nquant=NULL,
isrev=FALSE, getleg=TRUE, roundleg=1, title=NULL)

att.sete(g, from, to='edgeColor', pal=1, cols=NULL, na.col=grey(0.7),
xlim=c(20,100,1), shapes=NULL, breaks=NULL, categvec=NULL, nquant=NULL,
isrev=FALSE, getleg=TRUE, roundleg=1, title=NULL)
Arguments

- **g**: An igraph object.
- **from**: An attribute name available in 'g' <string>.
- **to**: A valid RedeR attribute name (see addGraph or type 'att.setv()' and 'att.sete()' for a quick list).
- **value**: A single value to add in a edge or vertex attribute.
- **filter**: A named list of length = 1, used to filter which nodes will receive the attribute. The attribute 'to' will be added to nodes which have the attribute.
- **index**: An optional index to set and attribute of a subset of vertices or edges (see set_vertex_attr and set_edge_attr).
- **breaks**: A numeric vector of two or more breakpoints to be applied to the attribute values.
- **pal**: Color scaling option (1 or 2); 'pal=1' will use a single color pallete, while 'pal=2' will split 'breaks' at the center, generating two color paletes. The 'pal=2' option maybe useful to build separated color paletes for negative and positive values.
- **xlim**: A numeric vector with three boundaries: c(<lower boundary>, <upper boundary>, <NA>). It corresponds to boundary values to be apply to numeric attributes (e.g. nodeSize). Default: c(20,100,1).
- **cols**: Vector of colors (either hexadecimals or valid R color names).
- **na.col**: A color representing eventual NAs. Default: grey(0.7).
- **shapes**: A string vector with valid RedeR shapes (see addGraph or type 'att.setv()' or 'att.sete()').
- **categvec**: Levels to encode attributes as a factor <vector>.
- **nquant**: Number of breakpoints to split attribute values by quantiles <integer>.
- **isrev**: Reverse of attribute values <logical>.
- **getleg**: Optional: return legend values <logical>.
- **dat**: A data frame with the attributes to be mapped to 'g'.
- **refcol**: A reference column in the 'dat' object used to map 'dat' to 'g'. For 'att.mapv', 'refcol' is a single integer value indicating a column with node ids. For 'att.mape', 'refcol' is a vector of two integers indicating two columns with edge ids. Also, for 'att.mapv', 'refcol = 0' indicates 'dat' rownames will be used to map 'dat' to 'g'.
- **roundleg**: Integer indicating the number of decimal places in the legend of numerical attributes.
- **title**: A title legend.

Value

Add, map, and set igraph attributes to the RedeR application.

Author(s)

Mauro Castro, Clarisse Groeneveld.
calld

See Also

addGraph

Examples

library(igraph)
data(ER.deg)

sg <- ER.deg$ceg  # an igraph object
dt <- ER.deg$dat  # a data frame object

# add a new vertex attribute, creating 'nodeFontSize' from a fixed value
sg <- att.addv(sg, to = "nodeFontSize", value = 10)

# same as above, but applied only to three nodes
sg <- att.addv(sg, to = "nodeFontSize", value = 100,
               filter = list("name" = V(sg)$name[1:3]))

# map vertex attributes, adding 'dt' into 'g'
sg <- att.mapv(g=sg, dat=dt, refcol=1)

# set a new vertex attribute, creating 'nodeAlias' from 'symbol'
sg <- att.setv(sg, from="Symbol", to="nodeAlias")

# set a new vertex attribute, creating 'nodeColor' from 'logFC.t3'
sg <- att.setv(sg, from="logFC.t3", to="nodeColor", breaks=seq(-1,1,0.2), pal=2)

# set a new vertex attribute, creating 'nodeSize' from 'ERbdist'
sg <- att.setv(sg, from="ERbdist", to="nodeSize", nquant=10, isrev=TRUE, xlim=c(5,40,1))

Call RedeR app from R.

Description

Method to invoke RedeR application from R.

Usage

calld(obj, ...)

Arguments

obj Object of RedPort Class.

... Additional arguments passed to RedeR application.
Details

Other arguments can be passed to the system in order to open the application.

- **filepath**  Path to 'reder.jar' file  
- **maxlag**  Max acceptable lag time for the R-Java callback confirmation (default=20 s)  
- **checkcalls**  Reports eventual errors from the R-Java callback (default=FALSE)

Value

Systems call to open RedeR application and XML-RPC server.

Author(s)

Mauro Castro

See Also

RedPort addGraph

Examples

```r
dp <- RedPort('MyPort')

## Not run:
callr(dp)

## End(Not run)
```

Description

Simple function for correlation analysis. This function computes a null distribution via permutation and returns the significant correlation values.

Usage

```r
cea(x, sig=0.01, padj.method="fdr", cor.method="spearman", nper=1000,
regulators=NULL, plotcea=TRUE, ...)
```
Arguments

- **x**: A matrix or data frame. Variables should be on rows, samples on columns. The 'x' matrix should be named, with unique row names.
- **sig**: Significance threshold.
- **p.adj.method**: Correction method passed to "p.adjust" function.
- **cor.method**: Correlation method passed to "cor" function.
- **nper**: Number of permutations.
- **plotcea**: Logical value, option to plot density and the null distributions.
- **regulators**: An optional character vector listing variables (e.g. genes) regarded as regulators.
- **...**: Additional arguments passed to plotcea option.

Details

Additional arguments:

- **n.breaks**: If plotcea=TRUE, 'n.breaks' sets the number of histogram breaks (Default=100 <integer>).
- **plotnull**: If plotcea=TRUE, 'plotnull' sets whether to plot the null distribution (Default=TRUE <logical>).
- **avnull**: If plotcea=TRUE, 'avnull' takes the average null distribution (Default=TRUE <logical>).
- **nullcol**: If plotcea=TRUE, 'nullcol' sets the color of the null distribution (Default="black" <character>).

Value

An adjacency matrix with significant correlation values.

Author(s)

Mauro Castro

See Also

cor p.adjust

Examples

```r
#--- Load a gene expression matrix
data(ER.deg)
x <- ER.deg$exp

#--- NOTE1: 'x' should have variables on rows and samples on columns!
#--- NOTE2: 'x' should be named, with unique names on rows.
x[1:3,1:3]
# GSM286031 GSM286032 GSM286033
#1 7.106765 7.225175 7.217780
#10005 8.108632 8.012673 7.932592
```
#10007 10.225053 10.068826 10.236548

--- Get a subset, with 100 genes, for a quick demonstration!
x <- x[sample(1:nrow(x))[1:100],]

## Not run:
--- Run correlation analysis, estimating significant
--- associations by permutation.
--- NOTE: set 'nper' for at least 1000
res <- cea(x=x, nper=100)

## End(Not run)

---

deleteEdges

Remove edges from RedeR graphs.

Description

Method to remove edges between nodes in an active RedeR session.

Usage

deleteEdges(obj, edges)

Arguments

obj Object of RedPort Class.
edges A vertex sequence <vector of strings> or data frame of ncol=2.

Value

Removes the specified edges from the graph.

Note

Prior calling this method invoke RedeR application via 'call' function.

Author(s)

Mauro Castro

See Also

RedPort
Examples

# Initialize igraph
library(igraph)

# edges as a vertex sequence
d shalt<="c("n1","n2","n1","n3","n1","n4","
# edges as a data.frame
d shalt<="c("n1","n1","n1","n1")
B<="c("n2","n3","n4","n5","n6","n7")

## Not run:
rdp <- RedPort()
calld(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
deleteEdges(rdp, c("n1","n3","n1","n7") )
updateGraph(rdp)

## End(Not run)

deleteNodes

Remove nodes from RedeR graphs.

Description

Method to remove nodes from an active RedeR session.

Usage

deleteNodes(obj, nodes)

Arguments

obj          Object of RedPort Class.

nodes         Node sequence as an array <array of strings>

Value

Remove graph objects.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').
Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callId(rdp)
addEdges(rdp, edges)
deleteNodes(rdp, c("n1","n3") )
updateGraph(rdp)

## End(Not run)

deleSelectedEdges  Delete selected edges in RedeR graphs.

Description

Remove all edges selected in an active RedeR session.

Usage

deleteSelectedEdges(obj)

Arguments

obj Object of RedPort Class.

Value

Remove graph objects.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callId').
deleteSelectedNodes

Author(s)

Mauro Castro

See Also

RedPort, selectAllEdges, selectEdges, deSelectEdges

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callld(rdp)
addEdges(rdp, edges)
selectEdges(rdp,"n1","n3")
deleteSelectedEdges(rdp)
updateGraph(rdp)

## End(Not run)

deleteSelectedNodes  Delete selected nodes in RedeR graphs.

Description

Remove all selected nodes from an active RedeR session.

Usage

deleteSelectedNodes(obj)

Arguments

obj  Object of RedPort Class.

Value

Remove graph objects.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').
deSelectEdges

Author(s)
Mauro Castro

See Also
RedPort, selectAllNodes, selectNodes, deSelectNodes

Examples

```r
# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callid(rdp)
addEdges(rdp, edges)
selectNodes(rdp,c("n3","n4"))
deleteSelectedNodes(rdp)
updateGraph(rdp)

## End(Not run)
```

deselectEdges = Unmark selected edges.

Description

Unmark all selected edges in an active RedeR session.

Usage

deselectEdges(obj)

Arguments

obj Object of RedPort Class.

Value

Unmark edges.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callid').
**deSelectGraph**

*Author(s)*
Mauro Castro

*See Also*
RedPort

*Examples*

```r
# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
calld(rdp)
addEdges(rdp, edges)
selectEdges(rdp,"n1","n3")
deselectEdges(rdp)
updateGraph(rdp)

## End(Not run)
```

---

**deSelectGraph**  
*Unmark selected graph objects.*

**Description**

Unmark all selected objects in an active RedeR session.

**Usage**

deselectGraph(obj)

**Arguments**

obj  
Object of RedPort Class.

**Value**

Unmark graph.

**Note**

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `callld`).
Author(s)

Mauro Castro

See Also

RedPort, selectNodes, selectEdges, selectGraph

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

rdp <- RedPort()
calld(rdp)
addEdges(rdp, edges)
selectGraph(rdp)
deSelectGraph(rdp)
updateGraph(rdp)

## End(Not run)

deSelectNodes

Unmark selected nodes.

Description

Unmark all selected nodes in an active RedeR session.

Usage

deSelectNodes(obj)

Arguments

obj Object of RedPort Class.

Value

Unmark nodes.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. ‘callld’).
exitd

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callld(rdp)
addEdges(rdp, edges)
selectNodes(rdp,c("n3","n4","n5"))
deSelectNodes(rdp)
updateGraph(rdp)

## End(Not run)

exitd  Exit RedeR R-to-Java interface.

Description

Exit R interface and close the active RedeR session.

Usage

exitd(obj)

Arguments

obj Object of RedPort Class.

Value

Exit software.

Author(s)

Mauro Castro
getContainerComponets

Description

Method to get components (nested objects) of a specific container from an active RedeR session.

Usage

getContainerComponets(obj, container)

Arguments

obj Object of RedPort Class.
container Name of the container in the graph <string>

Value

Returns all nested objects assigned to a container <array of strings>

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').

Author(s)

Mauro Castro

See Also

RedPort
**Examples**

```r
# Initialize igraph
library(igraph)

el < matrix(c('n1','n2','n3','n4'), ncol=2, byrow=TRUE)
g <- graph.edgelist(el)

## Not run:

rdp <- RedPort()
calld(rdp)
addGraph( rdp, g, layout.kamada.kawai(g) )
nestNodes( rdp, c('n1','n2') )
nestNodes( rdp, c("n3","n4") )
updateGraph(rdp)
getContainerComponets(rdp, "N0")

## End(Not run)
```

**Description**

Method to get ids of all edges from an active RedeR application.

**Usage**

```r
gedgeIDs(obj, ...)
```

**Arguments**

- `obj` Object of RedPort Class.
- `...` Additional arguments passed to RedeR application.

**Details**

Additional arguments:

- `type` Filter options. Valid arguments: '<node>', '<container>' or '<all>'. Default='node'.
- `status` Filter options. Valid arguments: '<selected>', '<nonselected>' or '<all>'. Default='all'

**Value**

Returns edges<array of integers>

**Note**

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `callld`).
getEdges

Get edges.

Description

Method to get all edges from an active RedeR application.

Usage

getEdges(obj, ...)

Arguments

obj Object of RedPort Class.
...

Additional arguments passed to RedeR application.

Details

Additional arguments:

status Filter options. Valid arguments: <'selected'>, <'nonselected'> or <'all'>. Default='selected'

type Filter options. Valid arguments: <'node'>, <'container'> or <'all'>. Default='node'.
**getGraph**

**Value**

Returns edges <array of strings>

**Note**

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `calld`).

**Author(s)**

Mauro Castro

**See Also**

RedPort getGraph

**Examples**

```r
# Initialize igraph
library(igraph)

edges<-c("n1", "n2", "n1", "n3", "n1", "n4", "n1", "n5", "n1", "n6", "n1", "n7")

## Not run:
rdp <- RedPort()
calld(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
getEdges(rdp)

## End(Not run)
```

---

**Description**

Method to get and wrap up RedeR graphs into R objects.

**Usage**

```r
getGraph(obj, ...)
```

**Arguments**

- `obj` Object of RedPort Class.
- `...` Additional arguments passed to RedeR application.
getNodeIDs

Get node IDs.

Description

Method to get node attributes ‘node IDs’ from an active RedeR session.

Usage

getNodeIDs(obj, ...)

Details

Additional arguments:

status  Filter options for RedeR graph status. Valid arguments: <'selected'>, <'nonselected'> or <'all'> (default='all').

type    Filter options for RedeR graph objects. Valid arguments: <'node'>, <'container'> or <'all'> (default='node').

attrs   Filter options for RedeR graph attributes. Valid arguments: <'plain'>, <'minimal'> or <'all'> (default='plain').

Value

Returns igraph objects.

Author(s)

Mauro Castro

See Also

addGraph RedPort

Examples

# Initialize igraph
library(igraph)

## Not run:
rdp <- RedPort()
callid(rdp)

#ps. first add a graph (e.g. see samples in RedeR or 'addGraph' method)!
g <- getGraph(rdp)

## End(Not run)
getNodeIDs

Arguments

obj Object of RedPort Class.

... Additional arguments passed to RedeR application.

Details

Additional arguments:

- **type** Filter options. Valid arguments: <'node'>, <'container'> or <'all'>. Default='node'.
- **status** Filter options. Valid arguments: <'selected'>, <'nonselected'> or <'all'>. Default='all'

Value

Returns node attributes <array of numerics>

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callId').

Author(s)

Mauro Castro

See Also

RedPort getGraph

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

rdp <- RedPort()
callId(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
getNodeIDs(rdp)

## End(Not run)
getNodes

Description

Method to get node list from an active RedeR session.

Usage

getNodes(obj, ...)

Arguments

obj Object of RedPort Class.
...
Additional arguments passed to RedeR application.

Details

Additional arguments:

status Filter options. Valid arguments: `<selected>`, `<nonselected>` or `<all>`. Default='selected'
type Filter options. Valid arguments: `<node>`, `<container>` or `<all>`. Default='node'.

Value

Returns nodes <array of strings>

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `callid`).

Author(s)

Mauro Castro

See Also

RedPort getGraph

Examples

```r
# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

rdp <- RedPort()
```
**getSourceEdgeIDs**

Get source-edge IDs.

### Description

Method to get IDs of all 'source' edges from an active RedeR session.

### Usage

```r
getSourceEdgeIDs(obj, ...) 
```

### Arguments

- `obj` Object of RedPort Class.
- `...` Additional arguments passed to RedeR application.

### Details

Additional arguments:

- `type` Filter options. Valid arguments: `<node'>, `<container'> or `<all'>. Default=`node'.
- `status` Filter options. Valid arguments: `<selected'>, `<nonselected'> or `<all'>. Default=`all'

### Value

Returns 'source' edges <array of integers>

### Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').

### Author(s)

Mauro Castro

### See Also

- `RedPort getGraph`
Examples

```r
# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

rdp <- RedPort()
calld(rdp)
addeDges(rdp, edges)
updateGraph(rdp)
getSourceEdgeIDs(rdp)

## End(Not run)
```

---

**getTargetEdgeIDs**  
*Get target-edge IDs.*

Description

Method to get IDs of all 'target' edges from an active RedeR session.

Usage

```r
getTargetEdgeIDs(obj, ...)
```

Arguments

- **obj**: Object of RedPort Class.
- **...**: Additional arguments passed to RedeR application.

Details

Additional arguments:

- **type**: Filter options. Valid arguments: `<node>`, `<container>` or `<all>`. Default='node'.
- **status**: Filter options. Valid arguments: `<selected>`, `<nonselected>` or `<all>`. Default='all'

Value

Returns 'target' edges <array of integers>

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `calld`).
gtoy.rm

Author(s)
Mauro Castro

See Also
RedPort getGraph

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

rdp <- RedPort()
callfd(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
gTargetEdgeIDs(rdp)

## End(Not run)

---

**Description**

A simple function to simulate random graphs with modular structures.

**Usage**

```r
gtoy.rm(m=5, nmax=30, nmin=5, p1=0.5, p2=0.03, p3=0.7, gg=NULL, nn=vcount(gg), noise.range=c(0.2, 0.6), plot=FALSE, fname=NULL)
```

**Arguments**

- `m`: Number of modules.
- `nmax`: The maximum number of vertices in each module.
- `nmin`: The minimum number of vertices in each module.
- `p1`: Probability for adding new vertices to a module.
- `p2`: Probability for drawing an edge between modules.
- `p3`: Probability for drawing an edge within modules.
- `gg`: An igraph object generated by the 'gtoy.rm' function. This option will add a random graph to the 'gg' object, following Erdos-Renyi model (see `erdos.renyi.game`).
When 'gg' is provided, this option defines the number of new vertices to add to the 'gg' object.

When 'gg' is provided, this option sets the upper and lower bounds of a uniform random distribution assigned to the random graph by the \texttt{runif} function, which is applied with default settings.

A logical value. This option generates a boxplot for the simulated signal generated for each graph module.

A string. The name of the pdf file in which the plot will be saved.

Value

Returns a igraph object.

Author(s)

Mauro Castro

Examples

\texttt{#g<-gtoy.rm()}

---

\textbf{isRelaxActive}  \textit{Inquires about RedeR current state.}

Description

Inquires whether 'dynamics' algorithm is active in RedeR application.

Usage

\texttt{isRelaxActive(obj)}

Arguments

\texttt{obj} Object of RedPort Class.

Value

Returns 1\texttt{<integer>} if true, 0\texttt{<integer>} otherwise.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').

Author(s)

Mauro Castro
mergeNodes

See Also

RedPort

Examples

```r
dp <- RedPort()
## Not run:
calld(rdp)
isRelaxActive(rdp)
   # 1 or 0
## End(Not run)
```

mergeNodes  Merge nodes.

Description

Merge nodes in an active RedeR session and build a new group.

Usage

```r
mergeNodes(obj, nodes)
```

Arguments

- **obj**: Object of RedPort Class.
- **nodes**: Node sequence <array of strings>

Value

Add/change graph objects.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').

Author(s)

Mauro Castro

See Also

RedPort
Examples

```r
# Initialize igraph
library(igraph)

nodes<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
calld(rdp)
addNodes(rdp, nodes)
mergeNodes(rdp,c("n2","n3","n4"))
updateGraph(rdp)

## End(Not run)
```

mergeOutEdges  
Merge out-edges between connected containers and transfers edges from nodes to containers.

Description

Method to assign out-edges to containers in an active RedeR session. This method transfers edges from nodes to the respective containers.

Usage

```r
mergeOutEdges(obj,...)
```

Arguments

obj Object of RedPort Class.

... Additional arguments passed to RedeR application.

Details

Additional arguments:

- `nlevels` Number of levels to be merged in the hierarchical network (default=2) \(<\text{integer}>\).
- `rescale` Logical value. Whether to rescale out-edge width to not overextend container size; if false, it will run a simple sum when combining out-edges between containers (default=TRUE).
- `lb` Custom lower bound to rescale edge width between containers (default=NA) \(<\text{numeric}>\).
- `ub` Custom upper bound to rescale edge width between containers (default=NA) \(<\text{numeric}>\).

Value

Add/change edge assignments.
### nesthc

**Nest hclust objects to containers.**

### Description

Method to nest nodes in an active RedeR session.

### Usage

```r
nesthc(obj, hc, ...)
```

### Arguments

- **obj**: Object of RedPort Class.
- **hc**: Either an object of hclust or pvclust class.
- **...**: Additional arguments passed to RedeR application; if a "pvclust" object, it also passes arguments for "pvpick" function (e.g. to set the p-value threshold).

### Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callId').

### Author(s)

Mauro Castro

### See Also

`RedPort`

### Examples

```r
# Initialize igraph
library(igraph)
el <- matrix(c("n1","n2","n1","n3","n1","n4","n2","n5","n2","n6","n2","n7"), ncol=2, byrow=TRUE)
g <- graph.edgelist(el)

## Not run:

rdp <- RedPort()
calld(rdp)
addGraph( rdp, g, layout.kamada.kawai(g) )
nestNodes( rdp, c("n1","n2") )
mergeOutEdges(rdp)
updateGraph(rdp)

## End(Not run)
```
Details

Additional arguments:

- **cutlevel**: Numeric value indicating the point where the hclust object should be cut (default = 2). The distance is related to the option 'metric'. For "rootdist" and "leafdist", the cut level is related to the n steps required to get to the root’s level or to the leaf’s level, respectively (n>=1). For 'height', the cut is related to the corresponding dendrogram height <numeric>.

- **metric**: Metric used to cut the hclust object at the top level (Options: "rootdist", "leafdist" or "height"; default="rootdist") <string>.

- **nmemb**: Minimum number of members for a nest (>=2) <numeric>.

- **nlev**: Maximum number of levels of a nested sequence (default=2) <numeric>.

- **grid**: Number of rows and cols to lay out graphs in the panel (default = c(2,3)) <numeric>.

- **gridScale**: Expansion factor of the grid area in the app panel. Options: 0.0 to 100 (default = 75) <numeric>.

- **gscale**: Expansion factor to set the nest area related to the parents – or related to the app panel. Provided as a vector with three numbers, c(n1,n2,n3): n1 is related to nests at the first level of the hierarchy (i.e. nests rooted to the panel); n2 is related to nests from single branches, and n3 nests from double branches (default = c(30,75,45)) <numeric>.

- **isAnchor**: Logical value; it sets whether to anchor containers in dynamic layouts.

- **isAssign**: Logical value; it sets whether to assign container names to nested nodes.

- **theme**: Some pre-defined nest attributes. Options: 'tm0','tm1','tm2','tm3','tm4','tm5', 'tm6' (default: 'tm6') <string>. Alternatively, it can be a list with customized attributes.

- **nlinewidth**: Line width of a nested series containers.

- **nfontsz**: Label font size a nested series containers.

- **plotch**: Logical value; whether to plot the corresponding hclust object (i.e. dendrogram).

- **col**: A color vector; it is used to color labels in both containers and corresponding hclust object (i.e. dendrogram nodes).

- **cex**: Numeric character expansion factor of dendrogram text and labels.

- **xlab**: A label for the dendrogram x axis.

- **ylab**: A label for the dendrogram y axis.

Value

Add/change graph objects, plot corresponding hclust object, and return corresponding ids.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').

Author(s)

Mauro Castro
nestNodes

See Also

RedPort nestNodes gtoy.rm

Examples

# Initialize igraph
library(igraph)

g <- gtoy.rm()
hc <- hclust(dist(get.adjacency(g)))

#plot(hc)

## Not run:
rdp <- RedPort()
calld(rdp)
addGraph(rdp,g)
ids <- nesthc(rdp, hc)

## End(Not run)

nestNodes

Nest nodes to containers.

Description

Method to nest nodes in an active RedeR session.

Usage

nestNodes(obj, nodes, ...)

Arguments

obj Object of RedPort Class.
nodes <array of strings>
... Additional arguments passed to RedeR application.

Details

Additional arguments:

nestImage  Status of the container on the screen: '<plain>', '<transparent>', or '<hide>' (default = 'plain').
isAssign Logical value, whether to assign the container name to the nested nodes (default = TRUE).
isAnchor Logical value, whether is to anchor the container in dynamic layouts (default = FALSE).
**nestNodes**

**gscale**  Expansion factor of the nest area related to a parent nest – or related to the app panel (default = 40) <numeric>.

**gcoord** Sets the nest c(x,y) center related to the parent center. Coords between 0 and 100 are set to the inner area (default = NULL) <numeric vector>.

**parent** Nest ID of a parent nest. Must be used with 'isAssign=TRUE' (default = NULL).

**gatt** A list with graph attributes. See nest attribute syntax in addGraph

**theme** Some pre-defined nest attributes. Options: `tm0', 'tm1', 'tm2', 'tm3', 'tm4', 'tm5', 'tm6' <string>. Alternatively, it can be a list with customized attributes.

### Value

Add/change graph objects.

### Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `callld`).

### Author(s)

Mauro Castro

### See Also

*RedPort*

### Examples

```r
# Initialize igraph
library(igraph)

el<-matrix(c('n1','n2','n3','n4'), ncol=2, byrow=TRUE)
g <- graph.edgelist(el)

## Not run:
rdp <- RedPort()
callld(rdp)
addGraph( rdp, g, layout.kamada.kawai(g) )
nestNodes( rdp, c('n1','n2'))
nestNodes( rdp, c("n3","n4") )

## End(Not run)
```
ping

Test RedeR R-to-Java interface.

Description

Test R interface and the connection to an active RedeR session.

Usage

ping(obj)

Arguments

obj Object of RedPort Class.

Value

"R interface is ready to use!"

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').

Author(s)

Mauro Castro

See Also

RedPort

Examples

rdp <- RedPort('MyPort')

## Not run:
callld(rdp)
ping (rdp)

## End(Not run)
Description

Preprocessed data from a time-course gene expression and ChIP-on-chip analysis of estrogen receptor (ER) binding sites in MCF7 breast cancer cell line (Carroll et al, 2006).

Usage

```r
data(Carroll2006)
```

Format

`Carroll2006` List containing 'exp', 'tgs', 'ids', and 'bdsites' R objects.

Details

The gene expression dataset consists of 12 time-course Affymetrix U133Plus2.0 microarrays: 3 replicates at 0h, 3 replicates at 3h, 3 replicates at 6h and 3 replicates at 12h. The original dataset is available at GEO database (GSE11324). The gene ER binding site dataset consists of a Bed file of ER ChIP-on-chip experiment. The original dataset is available at http://research.dfci.harvard.edu/brownlab/datasets/index.php (ER sites from the Bed file '1E-5.bed').

- **Carroll2006$exp** data.frame with log2 gene expression dataset.
- **Carroll2006$tgs** data.frame with microarray details (e.g. targets for limma analysis).
- **Carroll2006$ids** data.frame with gene ids used in RedeR case study.
- **Carroll2006$bdsites** data.frame with ER binding sites mapped to genome build GRCh37.
- **hs.inter** Human interactome extracted from the Human Protein Reference Database (HPRD) in April 2011 (igraph object) (‘name’ attribute is mapped to ENTREZ ID).
- **ER.limma** data-frame containing pre-processed results from limma analysis and ER binding sites mapped to differentially expressed (DE) genes. Content: annotation (ENTREZ and Symbol), time-course fold change (logFC.t3, logFC.t6, logFC.t12), p values (p.value.t3, p.value.t6, p.value.t12), DE genes (degenes.t3, degenes.t6, degenes.t12) and distance of the closest ER bd site to the TSS – in kb (ERbdist).
- **ER.deg$dat** Summary from ER.limma data object with extracted data for differentially expressed genes only.
- **ER.deg$exp** Data matrix with log2 gene expression values of DE genes.
- **ER.deg$ceg** Co-expression gene network of early ER-responsive genes computed by the function `cea`.

References

RedPort

Examples

data(Carroll2006)
data(hs.inter)
data(ER.limma)
data(ER.deg)

RedPort

The constructor for the RedPort class.

Description

Constructor to build RedeR interface via XML-RPC (remote procedure call) server.

Usage

RedPort(title = 'default', host = '127.0.0.1', port = 9091, checkJava = FALSE)

Arguments

title A character string representing the XML-RPC port.
host The domain name of the machine that is running the RedeR XML-RPC server.
port An integer specifying the port on which the XML-RPC server should listen.
checkJava Check Java Runtime Environment (JRE) installed on the system.

Value

An object of the RedPort Class.

Author(s)

Mauro Castro

See Also

calld

Examples

rdp <- RedPort('MyPort')
Description

A class providing access to the RedeR application.

Slots

- title: The name of the XML-RPC port.
- uri: The uri to the XML-RPC server.
- host: The host to the XML-RPC server.
- port: The port number to the XML-RPC server.

Methods

- **Get node attributes from a RedeR session:**
  - getNodes
  - getNodeIDs

- **Get edge attributes from a RedeR session:**
  - getEdges
  - getEdgeIDs
  - getSourceEdgeIDs
  - getTargetEdgeIDs

- **Methods that change graph structure:**
  - addGraph
  - getGraph
  - addNodes
  - deleteNodes
  - nestNodes
  - updateContainerSize
  - mergeOutEdges
  - getContainerComponets
  - mergeNodes
  - addEdges
  - addEdgeBetweenContainers
  - deleteEdges
  - setArrowDirection

- Other methods to manipulate RedeR graphs:
  - updateGraph
RedPort methods invoke RedeR application via XML-RPC (remote procedure call) server. For each R method listed above there is a Java mirror that executes a callback procedure. Therefore, the Java callback engine must be initialized before any callback from RedeR (i.e. start the Java application).

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Creates a RedeR object by calling the constructor
rdp <- RedPort('MyPort')
Description

RedeR’s hierarchical force-directed interactive layout.

Usage

relax(obj, p1=100, p2=100, p3=100, p4=100, p5=100, p6=10,
p7=10, p8=100, p9=10)

Arguments

<table>
<thead>
<tr>
<th>arg</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object of RedPort Class.</td>
</tr>
<tr>
<td>p1</td>
<td>Edge target length (in pixels; &gt;= 1 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p2</td>
<td>Edge stiffness (arbitrary unities; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p3</td>
<td>Node repulsion factor (arbitrary unities; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p4</td>
<td>Node perimeter effect (in pixels; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p5</td>
<td>Node speed limit (arbitrary unities; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p6</td>
<td>Repulsion radius, i.e., this parameter limits the repulsion factor range (given in p1 units; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p7</td>
<td>Central pull (arbitrary unities; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p8</td>
<td>Nest-nest edge target length, i.e., edge target between linked containers (in pixels; &gt;= 1 ) &lt;numeric&gt;.</td>
</tr>
<tr>
<td>p9</td>
<td>Nest-node repulsion factor, i.e., repulsion among containers and out-nodes (arbitrary unities; &gt;= 0 ) &lt;numeric&gt;.</td>
</tr>
</tbody>
</table>

Details

RedeR’s interactive layout uses a force-directed algorithm described elsewhere (Brandes 2001; Fruchterman and Reingold 1991). Here we adapted the method to deal with nested networks. In force-directed graphs, each edge can be regarded as a spring - with a given target length - and can either exert a repulsive or attractive force on the connected nodes, while nodes are analogous to mutually repulsive charged particles that move according to the applied forces. In RedeR, the simulation is additionally constrained by the hierarchical structure of the network. For example, a nested node is constrained to its parent-node by opposing forces applied by the nest, which is regarded as a special node whose nested objects can reach a local equilibrium independently from other network levels. The simulation is adjusted by global options and evolves until the system reaches the equilibrium state. The default values are set to layout sparse networks with few nodes (e.g. 10-100 nodes). For large and dense networks better results can be achieved interactively by tuning one or more parameters.

Author(s)

Mauro Castro
References

Examples

```r
# Initialize igraph
library(igraph)

g <- graph.lattice(c(5,5,5))

## Not run:
rdp <- RedPort()
calld(rdp)
addGraph( rdp, g, layout.random(g) )
relax(rdp)

## End(Not run)
```

resetd
Reset RedeR app.

Description
Reset the active RedeR session.

Usage

```r
resetd(obj)
```

Arguments

- `obj` Object of RedPort Class.

Value

Reset the software panel.

Author(s)
Mauro Castro

See Also

- `RedPort`
selectAllEdges

Select all edges.

Examples

```r
rdp <- RedPort('MyPort')

## Not run:
calld(rdp)
resetd(rdp)

## End(Not run)
```

Description

Method to mark all edges in an active RedeR application. Selected objects are put available for other methods. It can be done interactively as well.

Usage

```r
selectAllEdges(obj)
```

Arguments

- `obj`: Object of RedPort Class.

Value

Mark edges.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `calld`).

Author(s)

Mauro Castro

See Also

`RedPort`, `deleteSelectedEdges`
Examples

```r
# Initialize igraph
library(igraph)

dges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

  rdp <- RedPort()
  calld(rdp)
  addEdges(rdp, edges)
  selectAllEdges(rdp)
  updateGraph(rdp)

## End(Not run)
```

Description

Mark all nodes in an active RedeR application.

Usage

```r
selectAllNodes(obj)
```

Arguments

- **obj**: Object of RedPort Class.

Value

Mark nodes.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `calld`).

Author(s)

Mauro Castro

See Also

RedPort, deleteSelectedNodes
selectEdges

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callld(rdp)
addEdges(rdp, edges)
selectAllNodes(rdp)
updateGraph(rdp)

## End(Not run)

selectEdges

Description
Select edges in an active RedeR application.

Usage
selectEdges(obj, edges)

Arguments
obj
Object of RedPort Class.

edges
A vertex sequence <vector of strings> or data frame of ncol=2.

Value
Mark edges – which can be handled by other methods.

Note
Prior calling this method invoke RedeR application via `callld` function.

Author(s)
Mauro Castro

See Also
RedPort, deleteSelectedEdges
Examples

# Initialize igraph
library(igraph)

# edges as a vertex sequence
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

# edges as a data.frame
edges <- data.frame(A=c("n1","n1","n1","n1","n1","n1"),
                      B=c("n2","n3","n4","n5","n6","n7"))

## Not run:
rdp <- RedPort()
calld(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
selectEdges(rdp,c("n1","n3"))
updateGraph(rdp)

## End(Not run)

---

selectGraph 

Select graph.

Description

Method to mark all objects in an active RedeR application. Selected objects are put available for other methods. It can be done interactively as well.

Usage

selectGraph(obj)

Arguments

obj Object of RedPort Class.

Value

Mark graph.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').

Author(s)

Mauro Castro
selectNodes

See Also

RedPort, deleteSelectedNodes, deleteSelectedEdges, deSelectGraph

Examples

# Initialize igraph
library(igraph)

dges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
  rdp <- RedPort()
  calld(rdp)
  addEdges(rdp, dges)
  selectGraph(rdp)
  updateGraph(rdp)

## End(Not run)

selectNodes Select nodes.

Description

Method to select nodes in an active RedeR application. Selected objects are put available for other methods. It can be done interactively as well.

Usage

selectNodes(obj, nodes, anchor=FALSE, nt=NULL)

Arguments

obj Object of RedPort Class.

nodes Names of nodes (or containers) <string or array of strings>

anchor Option to anchor selected nodes: this will prevent the relax function from applying the relaxing algorithm on the selected nodes <boolean>

nt Option for nested nodes: to restrict searching to a specific container <string>

Value

Mark nodes – which can be handled by other methods.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'calld').
setArrowDirection

Author(s)
Mauro Castro

See Also
RedPort, deleteSelectedNodes

Examples
# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callid(rdp)
addEdges(rdp, edges)
selectNodes(rdp,c("n3","n4","n5"))
updateGraph(rdp)

## End(Not run)

setArrowDirection (obj, nodeA, nodeB, direction)

Description
Method to set edge attribute 'arrow direction' in active RedeR sessions.

Usage
setArrowDirection(obj, nodeA, nodeB, direction)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object of RedPort Class.</td>
</tr>
<tr>
<td>nodeA</td>
<td>Name &lt;string&gt;</td>
</tr>
<tr>
<td>nodeB</td>
<td>Name &lt;string&gt;</td>
</tr>
<tr>
<td>direction</td>
<td>Options: 0 (A-B), 1 (A-&gt;B), 2 (A&lt;-B) or 3 (A&lt;-&gt;B) &lt;integer&gt;</td>
</tr>
</tbody>
</table>

Value
Sets edge attribute <integer>
Note

The direction is set according to the edge order in the app (i.e. the edge list available inside RedeR). So, if a request for direction "1" places nodeA='B' and nodeB='A', then the direction will appear as A->B in the app.

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callid(rdp)
addEdges(rdp, edges)
setArrowDirection(rdp, "n1", "n2", 2)
updateGraph(rdp)

## End(Not run)

---

### startRedeR

**Warnings for RedeR transitional version.**

Description

These new functions are part of a new launch pipeline introduced in RedeR version 3 or later. However, a user might attempt to use them on an outdated version of RedeR. To mitigate this, a warning will be issued on the transitional version.

Usage

startRedeR(...)
addGraphToRedeR(...)
resetRedeR(...)
getGraphFromRedeR(...)
relaxRedeR(...)
addLegendToRedeR(...)
subg

Arguments

... not used.

Value

Warnings

Examples

# Return warnings.

---

subg Subgraph of a graph.

Description

Creates a subgraph containing only nodes specified from a data frame, including all edges among neighbors.

Usage

subg(g, dat, refcol=1, maincomp=TRUE, connected=TRUE, transdat=TRUE)

Arguments

g An igraph object.
dat A data frame with node ids and attributes to be mapped to 'g'.
refcol The reference column (node ids) in the 'dat' object.
maincomp Logical value, whether to return only the main component of the subgraph.
connected Logical value, whether to return only connected nodes.
transdat Logical value, whether to transfer node attributes from the 'dat' object to the subgraph.

Value

Returns a igraph object.

Author(s)

Mauro Castro

Examples

data(hs.inter)
data(ER.deg)
#subnet <- subg(g=hs.inter, dat=ER.deg$dat, refcol=1)
updateContainerSize

Description

Updates the size of all containers in an active RedeR session.

Usage

updateContainerSize(obj)

Arguments

obj Object of RedPort Class.

Value

Updates RedeR’s container objects.

Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. 'callld').

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callld(rdp)
addEdges(rdp, edges)
nestNodes( rdp, c("n2","n3") )
updateContainerSize(rdp)
updateGraph(rdp)

## End(Not run)
updateCoordXY

Update node coordinates.

Description

This function updates node coordinates of an igraph object with the node coordinates from the RedeR interface.

Usage

updateCoordXY(obj, g, delNodes=FALSE, delEdges=FALSE)

Arguments

obj Object of RedPort Class.

g An igraph object, which will be updated with the graph layout displayed in the RedeR interface. Note: 'g' must be the same igraph object sent to the RedeR interface by the addGraph function.

delNodes Option to delete nodes from 'g' when these nodes are not displayed in the RedeR interface.

delEdges Option to delete edges from 'g' when these edges are not displayed in the RedeR interface.

Value

An update igraph object.

Note

Prior calling this method make sure that there is an active RedeR session.

Author(s)

Mauro Castro

See Also

RedPort

Examples

# Initialize igraph
library(igraph)

g <- igraph::sample_pa(10, directed=FALSE)
V(g)$name <- letters[1:10]

## Not run:
updateGraph

Update RedeR graphs.

Description

This function updates an active RedeR application session.

Usage

updateGraph(obj)

Arguments

obj Object of RedPort Class.

Value

An updated RedeR application session.

Note

Prior calling this method make sure that there is an active RedeR session.

Author(s)

Mauro Castro

See Also

RedPort
### Examples

```r
# Initialize igraph
library(igraph)

dges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
rdp <- RedPort()
callrd(rd)
addEdges(rdp, edges)
updateGraph(rdp)

## End(Not run)
```

---

<table>
<thead>
<tr>
<th>version</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Description

Check RedeR application version.

### Usage

```r
version(obj)
```

### Arguments

- **obj**: Object of RedPort Class.

### Value

Returns the version of the current RedeR application that is listening a specified XML-RPC port.

### Note

Prior calling this method invoke RedeR application via XML-RPC server (i.e. `callrd`).

### Author(s)

Mauro Castro

### See Also

- RedPort
Examples

```r
dp <- RedPort('MyPort')

## Not run:

calld(dp)
version(dp)

## End(Not run)
```
Index

* attributes
  att, 13
* cea
  cea, 16
* classes
  RedPort-class, 46
* dataset
  RedeR.data, 44
* graphs
  RedPort-class, 46
* graph
  .rederpost, 3
  addEdgeBetweenContainers, 4
  addEdges, 5
  addGraph, 6
  addNodes, 12
calld, 15
deleteEdges, 18
deleteNodes, 19
deleteSelectedEdges, 20
deleteSelectedNodes, 21
deselectEdges, 22
deselectGraph, 23
deselectNodes, 24
exitd, 25
getContainerComponents, 26
getEdgeIDs, 27
getEdges, 28
getGraph, 29
getNodeIDs, 30
getNodes, 32
getSourceEdgeIDs, 33
getTargetEdgeIDs, 34
gtoy.rm, 35
isRelaxActive, 36
mergeNodes, 37
mergeOutEdges, 38
nesthc, 39
nestNodes, 41
ping, 43
RedPort, 45
resetd, 49
selectAllEdges, 50
selectAllNodes, 51
selectEdges, 52
selectGraph, 53
selectNodes, 54
setArrowDirection, 55
updateContainerSize, 58
updateCoordXY, 59
updateGraph, 60
version, 61
* layout
  relax, 48
* legend
  addLegend, 10
* package
  RedeR-package, 3
* subgraph
  subg, 57
  .rederpost, 3
  .rederpost,RedPort-method(.rederpost), 3
  addEdgeBetweenContainers, 4, 46
  addEdgeBetweenContainers,RedPort-method
  (addEdgeBetweenContainers), 4
  addEdges, 5, 46
  addEdges,RedPort-method(addEdges), 5
  addGraph, 6, 14–16, 30, 42, 46, 59
  addGraph,RedPort-method(addGraph), 6
  addGraphToRedeR (startRedeR), 56
  addLegend, 9, 10
  addLegendToRedeR (startRedeR), 56
  addNodes, 12, 46
  addNodes,RedPort-method(addNodes), 12
  att, 9, 13
calld, 15, 45, 47
selectAllNodes, RedPort-method (selectAllNodes), 51
selectEdges, 21, 24, 47, 52
selectEdges, RedPort-method (selectEdges), 52
selectGraph, 24, 47, 53
selectGraph, RedPort-method (selectGraph), 53
selectNodes, 9, 22, 24, 47, 54
selectNodes, RedPort-method (selectNodes), 54
set_edge_attr, 14
set_vertex_attr, 14
setArrowDirection, 46, 55
setArrowDirection, RedPort-method (setArrowDirection), 55
startRedeR, 56
subg, 57
updateContainerSize, 46, 58
updateContainerSize, RedPort-method (updateContainerSize), 58
updateCoordXY, 59
updateCoordXY, RedPort-method (updateCoordXY), 59
updateGraph, 46, 60
updateGraph, RedPort-method (updateGraph), 60
version, 47, 61
version, RedPort-method (version), 61