Package ‘RedeR’

January 31, 2017

**Type** Package

**Title** Interactive visualization and manipulation of nested networks

**Version** 1.22.0

**Date** 2016-03-11

**Author** Mauro Castro, Xin Wang, Florian Markowetz

**Maintainer** Mauro Castro &lt;mauro.a.castro@gmail.com&gt;

**Depends** R (&gt;= 2.15), methods, igraph

**Imports** RCurl, XML, pvclust

**Suggests** PANR

**SystemRequirements** Java Runtime Environment (&gt;= 6)

**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.

**License** GPL (&gt;= 2)

**biocViews** Infrastructure, GraphAndNetwork, Software, Network, Visualization, DataRepresentation


**LazyLoad** yes

**NeedsCompilation** no

### R topics documented:

- RedeR-package .............................................. 2
- addEdgeBetweenContainers ................................. 3
- addEdges .................................................... 4
- addGraph ................................................... 5
- addLegend ................................................... 8
- addNodes .................................................... 10
- addSeries .................................................. 11
- addSubgraph ............................................... 12
- addSubgraph.list .......................................... 13
- att ......................................................... 15
- callId ...................................................... 16
- cea ......................................................... 17
RedeR-package

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

Package: RedeR
Type: Package
License: GPL
LazyLoad: yes

Author(s)
Mauro Castro <mauro.a.castro@gmail.com>

References

See Also
RedPort-class

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

<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>containerA</td>
<td>&lt;string&gt;</td>
</tr>
<tr>
<td>containerB</td>
<td>&lt;string&gt;</td>
</tr>
</tbody>
</table>

Value
Add graph objects.

Note
Prior calling this method invoke RedeR application via XML-RPC server (i.e. ‘calld’).
addEdges

Add edges to RedeR graphs.

Description

Add edges to an active RedeR session.

Usage

addEdges(obj, edges)

Arguments

obj Object of RedPort Class.
edges Edge sequence as an array <array of strings>.

Value

Adds the specified edges to the graph.

Note

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

Author(s)

Mauro Castro
addGraph

See Also
RedPort

Examples

dp <- RedPort('MyPort')
edges <- c("n1", "n2", "n1", "n3", "n1", "n4", "n1", "n5", "n1", "n6", "n1", "n7")

## Not run:
callid(dp)
addEdges(dp, edges)
updateGraph(dp)

## End(Not run)

addGraph Add graphs to RedeR application.

Description
Method to wrap R graphs into RedeR objects and send it to RedeR app.

Usage
addGraph(obj, g, ...)

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) <matrix>.
gscale Expansion factor of the graph area related to the app panel area (default = 75) <numeric>.
zoom Sets the zoom scale for the app panel (range: 0.0 to 100.0; default = 100.0) <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)) <numeric vector>.
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).

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

**ntransform** Logical value, whether to transform nodes in containers (default = FALSE).

**parent** ID of a container already available in the app <string>. Nodes from g will be nested to this container.

**Value**

Submits R graphs to RedeR app.

**Attributes passed by the igraph object**

**Graph attributes:**

- **bgColor** Sets the background color of the app panel <hexadecimal>.
- **zoom** Sets the zoom scale for the app panel (range: 0.0 to 100.0) (Default=100) <numerics>.
- **gscale** Expansion factor of the graph area related to the app panel (range: 0.0 to 100.0) (Default=100) <numerics> (PS. alternative entry to the ‘gscale’ argument above).
- **coordX** Sets the graph x center; x between 0 and 100 sets to visible area <numeric> (PS. alternative entry to the ‘coord’ argument above).
- **coordY** Sets the graph y center; y between 0 and 100 sets to visible area <numeric> (PS. alternative entry to the ‘coord’ argument above).
- **loadEdges** Logical value, whether to send edges to RedeR app (Default=TRUE) (PS. alternative entry to the ‘loadEdges’ argument above).
- **isNest** Logical value, whether to nest all nodes into a new container (Default=FALSE) (PS. 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).
- **nestColor** If isNest=TRUE, this attribute sets the ‘color’ of the new container <hexadecimal>.
- **nestAlias** If isNest=TRUE, this attribute sets the label of the new container <string>.
- **nestFontSize** If isNest=TRUE, this attribute sets the size of the container label (Default=12) <numerics>.
- **nestFontColor** If isNest=TRUE, this attribute sets the ‘color’ of the container label <hexadecimal>.
- **nestFontX** If isNest=TRUE, this attribute sets the x position of the label related to the container (Default=-8) <numerics>.
- **nestFontY** If isNest=TRUE, this attribute sets the y position of the label related to the container (Default=-8) <numerics>.
- **nestShape** If isNest=TRUE, this attribute sets the shape of the container, options: ‘ELLIPSE’, and ‘ROUNDED_RECTANGLE’ (Default=ELLIPSE).
- **nestSize** If isNest=TRUE, this attribute sets the size of the container (Default=NULL) <numerics>.
- **nestLineWidth** If isNest=TRUE, this attribute sets the line width of the container, options: >= 0 (Default=1.0) <numerics>.
- **nestLineColor** If isNest=TRUE, this attribute sets the line color of the container <hexadecimal>.
- **nestImage** If isNest=TRUE, sets the status of the container on the screen: ‘plain’, ‘transparent’, or ‘hide’ (Default=plain).
**nestLineType**  If `isNest=TRUE`, this attribute sets the line type of the container: `<'SOLID'>`, `<'DOTTED'>`, `<'DOTTED_SHORT'>`, `<'LONG_DASH'>` (Default='SOLID').

**Vertex attributes:**
- **name**  Node attribute 'name' <string>.
- **nodeAlias**  Node attribute 'alias' <string>.
- **nodeBend**  Node attribute 'bend', options: 0-100% (Default=50) <numeric>.
- **coordX**  Node attribute 'x coord' (Default=random coord) <numeric>.
- **coordY**  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=E L L I P S E) <string>.
- **nodeColor**  Node attribute 'color', e.g. "#ff0000" for red <hexadecimal>.
- **nodeWeight**  Node attribute 'weight', options: >= 0 (Default=0) <numeric>.
- **nodeLineWidth**  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>.

**Edge attributes:**
- **arrowDirection**  Edge attribute 'arrow direction', used to set mixed associations in undirected graphs. Options: 0 (A>B), 1 (A->B), -1 (A<lB), 2 (A<-B), -2 (A<l-B), 3 (A <-> B), -3 (A l-B), 4 (A l-> B) and -4 (A l-< B) (Default=0) <integer>.
- **arrowType**  Edge attribute 'arrow type', used to set the association mode in directed graphs. Options: -1, 0 and 1 (Default=1) <integer>.
- **edgeWeight**  Edge attribute 'weight', options: >= 0 (Default=0.0) <numeric>.
- **edgeWidth**  Edge attribute 'width', options: >=0 (Default=1.0) <numeric>.
- **edgeColor**  Edge attribute 'color', e.g. "#ff0000" for red <hexadecimal>.
- **edgeType**  Edge attribute 'color', options: 'SOLID', 'DOTTED', 'DOTTED_SHORT', 'LONG_DASH' (Default='SOLID').
- **arrowLength**  Edge arrow attribute 'length', options: > 0 (Default=10) <numeric>.
- **arrowAngle**  Edge arrow attribute 'angle', options: 0-90 (Default=45) <numeric>.
- **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
rdp <- RedPort('MyPort')

## Not run:
calld(rdp)

###

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

resetd(rdp)

###

g2 <- graph.lattice(c(5,5,5))
addGraph( rdp, g2, layout.kamada.kawai(g2) )

resetd(rdp)

###

g <- barabasi.game(10)
V(g)$name<-letters[1:10]
V(g)$nodeSize<-c(100,rep(30,9))
addGraph( rdp, g, ntransform=TRUE )

sg <- barabasi.game(3)
addGraph( rdp, sg, parent="a" )

resetd(rdp)

###...to check loading an interactome!

data(hs.inter)
system.time( addGraph(rdp, hs.inter, layout=NULL) )

## End(Not run)
```

---

**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, ...)`
**addLegend**

**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>.

... Additional arguments passed to RedeR application.

**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: 8) <numeric>.
- **title** Legend title <string>.
- **dxtitle** Distance (in pixel) from legend title to the main axis (default: 35) <numeric>.
- **size** Legend size; only for addLegend.color and addLegend.shape methods (default: 30) <numeric>.
- **bend** Legend width/height ratio; only for addLegend.color method (default: 0.85) <numeric>.
- **col** Legend color; only for addLegend.size and addLegend.shape methods (default: "#000000") <either hexadecimal or valid color name>.
- **intersp** Legend inter space (only for addLegend.size and addLegend.shape methods) (default: 0) <numeric>.
- **edgelen** Length of the edges in addLegend.size method (default: 50) <numeric>.

**Value**

Send legend objects to RedeR app.

**Author(s)**

Mauro Castro

**See Also**

`addGraph` `att.setv` `att.sete`
addNodes

Description

Method to add nodes to an active RedeR session.

Usage

```r
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. 'calld').
addSeries

Author(s)

Mauro Castro

See Also

RedPort

Examples

```r
rdp <- RedPort('MyPort')
nodes <- c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

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

## End(Not run)
```

Description

Method to send series of graphs to RedeR app.

Usage

```r
addSeries(obj, g, ...)
```

Arguments

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

Details

Additional arguments:

- `setnodes` Logical value, whether to update node attributes in the new item of the series (default = TRUE).
- `setedges` Logical value, whether to add edges and update attributes in the new item of the series (default = TRUE).

Value

Submits series of R graphs to RedeR app.

Author(s)

Mauro Castro
addSubgraph

See Also
addGraph

Examples

rdp <- RedPort('MyPort')

## Not run:
callld(rdp)

###

g1 <- graph.lattice(c(3,3,3))
addGraph( rdp, g1, layout.kamada.kawai(g1) )
V(g1)$nodeColor<-heat.colors(vcount(g1))
addSeries( rdp, g1)

## End(Not run)

Description

Method to send subgraph to RedeR app.

Usage

addSubgraph(obj, g, nodes, ...)

Arguments

obj Object of RedPort Class.
g An igraph object.
nodes Nodes of the subgraph <array of strings>
... Additional arguments passed to RedeR application.

Details

Additional arguments:

gatt A list of graph attributes. See attribute syntax in addGraph
gscale Expansion factor of the graph area related to the app panel (default = 75) <numerics>.
gcoord Sets the graph x,y center. Coords between 0 and 100 are set to the visible area of the app panel (default = c(75,75)) <numeric vector>.
theme Some pre-defined nest attributes. Options: 'tm0','tm1','tm2','tm3','tm4','tm5'

Value

Extracts subgraphs from 'igraph' objects and sends the result to the RedeR app.
addSubgraph.list

Add a list of subgraphs to RedeR application.

Description
Method to send subgraphs to RedeR app.

Usage
addSubgraph.list(obj, g, nodeList, ...)

Examples

```r
drp <- RedPort('MyPort')
## Not run:
calld(rdp)
g <- graph.lattice(c(5,5,5))
#. extracts a subgraph from g and sends to RedeR:
addSubgraph( rdp, g, nodes=c(1:10) )
#. sets some attributes on g prior to extraction!
g$isNest<-TRUE
g$nestColor="ff0000"
g$scale=50
addSubgraph( rdp, g, nodes=c(1:10) )
#. alternatively, sets an independent list of attributes:
att <-list()
att$isNest<-TRUE
att$nestColor="0000ff"
att$scale=50
att$coordX=25
att$coordY=25
addSubgraph( rdp, g, nodes=c(20:30), gatt=att )
#. for further attributes see 'addGraph' function!
## End(Not run)
```
Arguments

obj  Object of RedPort Class.
g  An igraph object.
nodeList  List of nodes. Will be used to extra subgraphs from g.
...  Additional arguments passed to RedeR application.

Details

Additional arguments:

gridRows  Number of lines to layout the subgraph panel (default = 2) <integer>
gridScale  Expansion factor of the grid area in the app panel. Options: 0.0 to 100 (default = 50) <numeric>.
gscale  Expansion factor each subgraph related to the app panel (default = 20) <numeric>.
gatt  Either a list or data frame with graph attributes (for data frames, attribute names on cols). See attribute syntax in addGraph
update  String argument: if 'all' it forces to update node/edge attributes of a graph already available in the app panel; if 'partial', only node attributes are updated (default = NULL).
theme  Some pre-defined nest attributes. Options: 'tm0', 'tm1', 'tm2', 'tm3', 'tm4', 'tm5', 'tm6'.

Value

Extracts subgraphs from 'igraph' objects and sends the result to the RedeR app.

Author(s)

Mauro Castro

See Also

addSubgraph addGraph

Examples

rdp <- RedPort('MyPort')
## Not run:
callrd(rdp)
g <- graph.lattice(c(5,5,5))
#..extract subgraphs from g and send to RedeR:
nl<-list(c(1:10),c(15:20))
att<-data.frame(isNest=c(TRUE,TRUE), nestColor=c("#0000ff","#ff0000"))
addSubgraph.list( rdp, g, nodeList=nl, gridRows=1, gatt=att, gridScale=80)
#..for further attributes see 'addGraph' function!
## End(Not run)
Map and set edge and vertex attributes to RedeR application.

Description
These functions map data frames containing edge/vertex attributes to an igraph object and set attributes to RedeR.

Usage
```
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)
att.mapv(g, dat, refcol=1)
att.mape(g, dat, refcol=c(1,2))
```

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()' or 'att.sete()').
- `breaks`: A numeric vector of two or more breakpoints to be applied to the attribute values.
- `pal`: Default color palette. Options: 1 or 2.
- `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 hexadecimal 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`: Optional: levels to encode attributes as a factor <vector>.
- `nquant`: Optional: number of breakpoints to split attribute values by quantiles <integer>.
- `isrev`: Optional: reversed version of attribute values <logical>.
- `getleg`: Optional: return legend values <logical>.
- `dat`: A data frame with the attributes to be mapped to 'g'.
- `refcol`: The reference columns in the 'data' object with either node ids (one column <integer>) or edge ids (two columns <vector of two integers>).
- `roundleg`: Integer indicating the number of decimal places (round) in the legend of numerical attributes.
- `title`: Optional: legend title.

Value
Map/set RedeR attributes to igraph objects.

Author(s)
Mauro Castro
Call RedeR app from R.

Description
Method to invoke RedeR application from R.

Usage

calld(obj, ...)

Arguments

obj Object of RedPort Class.
...

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 <string>

maxlag Max acceptable lag time for the R-Java callback confirmation (default=20 s) <numeric>

checkcalls Reports eventual errors from the R-Java callback (default=FALSE) <logical>

Value
Systems call to open RedeR application and XML-RPC server.
Co-expression analysis.

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

Usage
cea(x, sig=0.01, p.adj.method="fdr", cor.method="spearman", nper=1000, plotcea=TRUE, ...)

Arguments
- **x**: A matrix or data frame.
- **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.
- **...**: 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>)._
deleteEdges

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

Usage
deleteEdges(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>Edge sequence as an array &lt;array of strings&gt;</td>
</tr>
</tbody>
</table>

Value
Removes the specified edges from the graph.

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

Author(s)
Mauro Castro

See Also
RedPort

Examples

dp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
call(rdp)
addEdges(rdp, edges)
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. `call`).

Author(s)
Mauro Castro

See Also
RedPort
deleteSelectedEdges

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’).

Author(s)

Mauro Castro

See Also

RedPort, selectAllEdges, selectEdges, deSelectEdges
Examples

rdp <- RedPort('MyPort')
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")
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. 'calld').

Author(s)

Mauro Castro

See Also

RedPort, selectAllNodes, selectNodes, deSelectNodes
Examples

dep <- RedPort('MyPort')
edges <- c("n1", "n2", "n3", "n4", "n5", "n6", "n7")

## Not run:

calld(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. 'calld').

Author(s)

Mauro Castro

See Also

RedPort
```r
rdp <- RedPort('MyPort')
edges<-c("n1", "n2", "n3", "n1", "n3", "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

```r
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. `calld`).

## Author(s)

Mauro Castro

## See Also

- `RedPort`
- `selectNodes`
- `selectEdges`
- `selectGraph`
deSelectNodes

Examples

```r
dep <- RedPort('MyPort')
edges<-c("n1","n2","n3","n4","n5","n6","n7")

## Not run:
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

```r
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. 'calld').

Author(s)

Mauro Castro

See Also

RedPort
duplicateGraph

Examples

```r
dup <- RedPort('MyPort')
edges<-c("n1","n2","n3","n4","n5","n6","n7")

## Not run:
callId(dup)
addEdges(dup, edges)
selectNodes(dup,c("n3","n4","n5"))
deSelectNodes(dup)
updateGraph(dup)

## End(Not run)
```

duplicateGraph  
*Duplicate graphs in RedeR application.*

Description

Method to duplicate graphs and subgraphs of a network.

Usage

```r
duplicateGraph(obj, ...)
```

Arguments

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

Details

Additional arguments:

- `isToCopyEdges` Logical value, whether to include edges to the copy (default = TRUE).
- `isDefaultCopy` Logical value, whether to duplicate the complete network or to copy only the original graph (default = TRUE).
- `nodes` Optional: nodes to be duplicated <array of strings> (p.s. in this case, isDefaultCopy=TRUE).

Value

Duplicates graphs in RedeR app.

Author(s)

Mauro Castro

See Also

`addGraph`
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

See Also
RedPort

Examples
rdp <- RedPort('MyPort')

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

## End(Not run)
**getContainerComponents**  
*Get container components.*

**Description**

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

**Usage**

```r
getContainerComponents(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
drp <- RedPort('MyPort')
el<-matrix(c('n1','n2','n3','n4'), ncol=2, byrow=TRUE)
g <- graph.edgelist(el)
## Not run:
calld(rdp)
addGraph(rdp, g, layout.kamada.kawai(g) )
nestNodes(rdp, c('n1','n2') )
nestNodes(rdp, c('n3','n4') )
updateGraph(rdp)
getContainerComponents(rdp, "N0")
## End(Not run)
```
getEdgeIDs

Get edge IDs.

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

Usage
getEdgeIDs(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. 'calld').

Author(s)
Mauro Castro

See Also
RedPort getGraph

Examples

```r
rdp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")
## Not run:
  calld(rdp)
  addEdges(rdp, edges)
  updateGraph(rdp)
  getEdgeIDs(rdp)
## End(Not run)
```
getEdges

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'.

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

rdp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

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

## End(Not run)
getGraph

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

Usage
getGraph(obj, ...)

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

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
rdp <- RedPort('MyPort')
## Not run:
calld(rdp)
#ps. first add a graph (e.g. see samples in RedeR or 'addGraph' method)!
g <- getGraph(rdp)
## End(Not run)
**getNodeIDs**

Get node IDs.

**Description**

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

**Usage**

`getNodeIDs(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 node attributes <array of numerics>

**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
dp <- RedPort('MyPort')
edges<-c("n1", "n2", "n1", "n3", "n1", "n4", "n1", "n5", "n1", "n6", "n1", "n7")

## Not run:
calld(dp)
addEdges(dp, edges)
updateGraph(dp)
getNodeIDs(dp)

## End(Not run)
```
getNodes

Get nodes.

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. 'calld').

Author(s)
Mauro Castro

See Also
RedPort getGraph

Examples
rdp <- RedPort('MyPort')
edges<-c("n1","n2","n3","n1","n4","n5","n1","n6","n1","n7")

## Not run:
calld(rdp)
addEdges(rdp, edges)
updateGraph(rdp)
getNodes(rdp)

## End(Not run)
**getDescription**

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

**Usage**

```
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. `callld`).

**Author(s)**

Mauro Castro

**See Also**

RedPort `getGraph`

**Examples**

```
rdp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
callld(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**  
`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`).

**Author(s)**  
Mauro Castro

**See Also**  
RedPort `getGraph`

**Examples**  
```r  
rdp <- RedPort('MyPort')  
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")  
## Not run:  
calld(rdp)  
addEdges(rdp, edges)  
updateGraph(rdp)  
getTargetEdgeIDs(rdp)  
## End(Not run)```
**gtoy.rm**

**Random graphs and modules.**

**Description**

A very simple function to generate random graphs with modular structures.

**Usage**

```r
gtoy.rm(m=3, nmax=30, nmin=3, p1=0.5, p2=0.05, p3=0.9)
```

**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.

**Value**

Returns a igraph object.

**Author(s)**

Mauro Castro

**Examples**

```r
# g <- gtoy.rm()
```

---

**isDynamicsActive**

**Inquires about RedeR current state.**

**Description**

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

**Usage**

```r
isDynamicsActive(obj)
```

**Arguments**

- `obj` Object of RedPort Class.
mergeNodes

Value
Returns 1<integer> if true, 0<integer> otherwise.

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

Author(s)
Mauro Castro

See Also
RedPort

Examples
rdp <- RedPort('MyPort')

## Not run:
calld(rdp)
isDynamicsActive (rdp)
  # 1 or 0

## End(Not run)

mergeNodes Merge nodes.

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

Usage
mergeNodes(obj, nodes)

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>nodes</td>
<td>Node sequence &lt;array of strings&gt;</td>
</tr>
</tbody>
</table>

Value
Add/change graph objects.

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

Author(s)
Mauro Castro

See Also
RedPort

Examples

```r
drp <- RedPort('MyPort')
nodes<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
callId(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
mergeOutEdges(obj,...)

Arguments

obj Object of RedPort Class.

... Additional arguments passed to RedeR application.

Details

Additional arguments:

rescale Logical value. Whether to rescale the out-edge width to fit container size limits; if false, it will run a simple sum (default=TRUE).

lb Custom lower bound to rescale edge width (default=NULL) <numerics>.

ub Custom upper bound to rescale edge width between containers (default=NULL) <numerics>.

nlev Number of levels to be merged in the hierarchy (default=1) <integer>.

Value
Add/change edge assigments.
nesthc

Note

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

Author(s)

Mauro Castro

See Also

RedPort

Examples

```r
dep <- RedPort('MyPort')
e1 <- matrix(c("n1","n2","n1","n3","n1","n4","n2","n5","n2","n6","n2","n7"), ncol=2, byrow=TRUE)
#g <- graph.edgelist(e1)

## Not run:
calld(rdp)
addGraph( rdp, g, layout.kamada.kawai(g) )
nestNodes( rdp, c("n1","n2") )
mergeOutEdges(rdp)
updateGraph(rdp)

## End(Not run)
```

---

nesthc  

Nest hclust objects to containers.

Description

Method to nest nodes in an active RedeR session.

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>obj</code></td>
<td>Object of RedPort Class.</td>
</tr>
<tr>
<td><code>hc</code></td>
<td>Either an object of hclust or pvclust class.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Additional arguments passed to RedeR application; if a &quot;pvclust&quot; object, it also passes arguments for &quot;pvpick&quot; function (e.g. to set the p-value threshold).</td>
</tr>
</tbody>
</table>
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.

- **plothc** 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 and plot corresponding hclust object.

Note

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

Author(s)

Mauro Castro

See Also

RedPort nestNodes gtoy.rm
Examples

```r
# g <- gtoy.rm()
# hc <- hclust(dist(get.adjacency(g)))
# plot(hc)

# rdp <- RedPort('MyPort')

## Not run:
callId(rdp)
addGraph(rdp, g)
nesthc(rdp, hc)

## End(Not run)
```

---

**nestNodes**

Nest nodes to containers.

**Description**

Method to nest nodes in an active RedeR session.

**Usage**

```r
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).
- `gscale` Expansion factor of the nest area related to a parent nest – or related to the app panel (default = 40) <numerics>.
- `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.
ping

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
drp <- RedPort('MyPort')
e1<-matrix(c('n1','n2','n3','n4'), ncol=2, byrow=TRUE)
#g <- graph.edgelist(el)

## Not run:

  call(rdp)
  addGraph( rdp, g, layout.kamada.kawai(g) )
  nestNodes( rdp, c('n1','n2') )
  nestNodes( rdp, c('n3','n4') )

## End(Not run)
```

Description
Test RedeR R-to-Java interface.

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. 'calld').
Pre-processed dataset for RedeR case study.

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

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’).

Carroll2006exp data.frame with log2 gene expression dataset.
Carroll2006$expdata.frame with log2 gene expression dataset.
Carroll2006$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**


**Examples**

```r
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**

```r
RedPort(title = 'default', host = '127.0.0.1', port = 9091)
```

**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.

**Value**

An object of the RedPort Class.

**Author(s)**

Mauro Castro
RedPort-class

See Also
callId

Examples

```r
dp <- 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.
port: The port number to the XML-RPC server.
jclass: The RedeR Java class that should wrap up R graphics.

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
getContainerComponents
mergeNodes
addEdges
addEdgeBetweenContainers
deleteEdges
setArrowDirection
Methods to wrap up attributes and add/get graphs to/from RedeR:

- `addGraph`
- `getGraph`
- `addSubgraph`
- `addSeries`
- `duplicateGraph`

Other methods to manipulate RedeR graphs:

- `updateGraph`
- `selectEdges`
- `selectNodes`
- `selectAllEdges`
- `selectAllNodes`
- `selectGraph`
- `deSelectEdges`
- `deSelectNodes`
- `deSelectGraph`
- `deleteSelectedEdges`
- `deleteSelectedNodes`
- `isDynamicsActive`

Methods to establish RedeR server connection:

- `ping`
- `version`
- `callld`
- `exitd`
- `resetd`

Details

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

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

This function starts the dynamic layout and sets the force-directed options available in RedeR app.

Usage

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

Arguments

obj Object of RedPort Class.
p1 Edge target length (in pixels; >= 1 ) <numeric>.
p2 Edge stiffness (arbitrary unities; >= 1 ) <numeric>.
p3 Node repel factor (arbitrary unities; >= 1 ) <numeric>.
p4 Node perimeter effect (in pixels; >= 1 ) <numeric>.
p5 Node speed limit (arbitrary unities; >= 1 ) <numeric>.
p6 Nest-nest edge target length, i.e., edge target between linked containers (in pixels; >= 1 ) <numeric>.
p7 Nest-node repel factor, i.e., repulsion among containers and out-nodes (arbitrary unities; >= 1 ) <numeric>.
p8 Repulsion radius, i.e., this parameter limits the repel factor range (given in p1 units; >= 1 ) <numeric>.
ps Panel settings: logical value, whether to start interactive panel.

Details

One of the most versatile features of RedeR is the ability to deal with nested network objects using dynamic simulation, which makes it possible to represent, for example, subnetworks and time-series onto the same graph in a user-friendly routine. The simulation uses force-directed algorithms as 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. 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 iteratively (and interactively) until the system reaches the equilibrium state. The parameters controlling the dynamics are arbitrarily set to layout sparse networks with a 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
**resetd**

*Reset RedeR app.*

**Description**

Reset the active RedeR session.

**Usage**

`resetd(obj)`

**Arguments**

- `obj` Object of RedPort Class.

**Value**

Reset the software panel.

**Author(s)**

Mauro Castro

**See Also**

`RedPort`
selectAllEdges

Examples

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

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

## End(Not run)
```

### selectAllEdges

*Select all edges.*

**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
drp <- RedPort('MyPort')
edges<-c('n1','n2','n3','n4','n5','n6','n7')

## Not run:
calld(rdp)
addEdges(rdp, edges)
selectAllEdges(rdp)
```
selectAllNodes

updateGraph(rdp)

## End(Not run)

selectAllNodes  selectAllNodes

Description

Mark all nodes in an active RedeR application.

Usage

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. 'callld').

Author(s)

Mauro Castro

See Also

RedPort, deleteSelectedNodes

Examples

rdp <- RedPort('MyPort')
edges<-c("n1","n2","n3","n4","n5","n6","n7")

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

## End(Not run)
selectEdges

Description
Select edges in an active RedeR application.

Usage
selectEdges(obj, nodeA, nodeB)

Arguments
- **obj**: Object of RedPort Class.
- **nodeA**: <string>
- **nodeB**: <string>

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

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
dp <- RedPort('MyPort')
edges<-c("n1", "n2", "n1", "n3", "n1", "n4", "n1", "n5", "n1", "n6", "n1", "n7")

## Not run:
calld(rdp)
adEdges(rdp, edges)
selectEdges(rdp,"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**

```r
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

**See Also**

- `RedPort`, `deleteSelectedNodes`, `deleteSelectedEdges`, `deSelectGraph`

**Examples**

```r
rdp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:

calld(rdp)
addEdges(rdp, edges)
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**

```r
selectNodes(obj, nodes, nt=NULL)
```

**Arguments**

- `obj` Object of RedPort Class.
- `nodes` Names of nodes (or containers) <string or array of strings>
- `nt` Optional 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`).

**Author(s)**
Mauro Castro

**See Also**
RedPort, deleteSelectedNodes

**Examples**

```r
dp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
calld(dp)
addEdges(dp, edges)
selectNodes(dp,c("n3","n4","n5"))
updateGraph(dp)

## End(Not run)
```
Description

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

Usage

setArrowDirection(obj, nodeA, nodeB, direction)

Arguments

- obj: Object of RedPort Class.
- nodeA: Name <string>
- nodeB: Name <string>
- direction: Options: 0 (A-B), 1 (A->B), 2 (A<-B) or 3 (A<->B) <integer>

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

```r
dp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")
## Not run:
call(rdp)
addEdges(rdp, edges)
setArrowDirection(rdp, "n1", "n2", 2)
updateGraph(rdp)
## End(Not run)
```
**subg**

*Subgraph of a graph.*

**Description**

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

**Usage**

```r
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 an igraph object.

**Author(s)**

Mauro Castro

**Examples**

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

---

**updateContainerSize**

*Update container size.*

**Description**

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

**Usage**

```r
updateContainerSize(obj)
```
**updateGraph**

**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. `calld`).

**Author(s)**

Mauro Castro

**See Also**

RedPort

**Examples**

```r
drp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

## Not run:
calld(drp)
addEdges(drp, edges)
nestNodes( drp, c("n2","n3") )
updateContainerSize(drp)
updateGraph(drp)

## End(Not run)
```

---

**Description**

Updates an active RedeR application session.

**Usage**

`updateGraph(obj)`

**Arguments**

| obj | Object of RedPort Class. |

**Value**

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

Author(s)
Mauro Castro

See Also
RedPort

Examples
```r
dp <- RedPort('MyPort')
edges<-c("n1","n2","n1","n3","n1","n4","n1","n5","n1","n6","n1","n7")

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

## End(Not run)
```

---

version

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check RedeR application version.</td>
</tr>
</tbody>
</table>

Usage
```
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. 'calld').

Author(s)
Mauro Castro

See Also
RedPort
**Examples**

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

## Not run:

calld(rdp)
version(rdp)
  # "RedeR v1.0"

## End(Not run)
```
Index

*Topic **attributes**
  att, 15

*Topic **cea**
  cea, 17

*Topic **classes**
  RedPort-class, 44

*Topic **dataset**
  RedR.data, 42

*Topic **graphs**
  RedPort-class, 44

*Topic **graph**
  addEdgeBetweenContainers, 3
  addEdges, 4, 44
  addNodes, 10
  addSeries, 11
  addSubgraph, 12
  addSubgraph.list, 13
  calld, 16
  deleteEdges, 18
  deleteNodes, 19
  deleteSelectedEdges, 20
  deleteSelectedNodes, 21
  deSelectEdges, 22
  deSelectGraph, 23
  deSelectNodes, 24
  duplicateGraph, 25
  exitd, 26
  getContainerComponets, 27
  getEdgeIDs, 28
  getEdges, 29
  getGraph, 30
  getNodeIDs, 31
  getNodes, 32
  getSourceEdgeIDs, 33
  getTargetEdgeIDs, 34
  gtoy.rm, 35
  isDynamicsActive, 35
  mergeNodes, 36
  mergeOutEdges, 37
  nesthc, 38
  nestNodes, 40
  ping, 41
  RedPort, 43
  resetd, 47
  selectAllEdges, 48
  selectAllNodes, 49
  selectEdges, 50
  selectGraph, 51
  selectNodes, 52
  setArrowDirection, 53
  updateContainerSize, 54
  updateGraph, 55
  version, 56

*Topic **layout**
  relax, 46

*Topic **legend**
  addLegend, 8

*Topic **package**
  RedR-package, 2

*Topic **subgraph**
  subg, 54
  addEdgeBetweenContainers, 3, 44
  addEdgeBetweenContainers, RedPort-method
    (addEdgeBetweenContainers), 3
  addEdges, 4, 44
  addEdges, RedPort-method (addEdges), 4
  addGraph, 5, 9, 12–17, 25, 30, 40, 44, 45
  addGraph, RedPort-method (addGraph), 5
  addLegend, 7, 8
  addNodes, 10, 44
  addNodes, RedPort-method (addNodes), 10
  addSeries, 11, 45
  addSeries, RedPort-method (addSeries), 11
  addSubgraph, 12, 14, 45
  addSubgraph, RedPort-method
    (addSubgraph), 12
  addSubgraph.list, 13, 13
  addSubgraph.list, RedPort-method
    (addSubgraph.list), 13
  att, 7, 15
  att.sete, 9
  att.setv, 9
  calld, 16, 44, 45
  calld, RedPort-method (calld), 16
INDEX

Carroll2006 (RedeR.data), 42
cia, 17, 43
cor, 18
deleteEdges, 18, 44
deleteEdges, RedePort-method (deleteEdges), 18
deleteNodes, 19, 44
deleteNodes, RedePort-method (deleteNodes), 19
deleteSelectedEdges, 20, 45, 48, 50, 51
deleteSelectedEdges, RedePort-method (deleteSelectedEdges), 20
deleteSelectedNodes, 21, 45, 49, 51, 52
deleteSelectedNodes, RedePort-method (deleteSelectedNodes), 21
deSelectEdges, 20, 22, 45
deSelectEdges, RedePort-method (deSelectEdges), 22
deSelectGraph, 23, 45, 51
deSelectGraph, RedePort-method (deSelectGraph), 23
deSelectNodes, 21, 24, 45
deSelectNodes, RedePort-method (deSelectNodes), 24
duplicateGraph, 25, 45
duplicateGraph, RedePort-method (duplicateGraph), 25

ER.deg (RedeR.data), 42
ER.limma (RedeR.data), 42
exitd, 26, 45
exitd, RedePort-method (exitd), 26
getContainerComponets, 27, 44
getContainerComponets, RedePort-method (getContainerComponets), 27
gEdgeIDs, 28, 44
gEdgeIDs, RedePort-method (gEdgeIDs), 28
gEdges, 29, 44
gEdges, RedePort-method (gEdges), 29
gGetGraph, 7, 28, 29, 30, 31–34, 44, 45
gGetGraph, RedePort-method (getGraph), 30
ggetNodeIDs, 31, 44
ggetNodeIDs, RedePort-method (getNodeIDs), 31
gNodes, 32, 44
gNodes, RedePort-method (gNodes), 32
ggetSourceEdgeIDs, 33, 44
ggetSourceEdgeIDs, RedePort-method (getSourceEdgeIDs), 33
gTargetEdgeIDs, 34, 44
gTargetEdgeIDs, RedePort-method (gTargetEdgeIDs), 34
htoy.rm, 35, 39
hs.inter (RedeR.data), 42
isDynamicsActive, 35, 45
isDynamicsActive, RedePort-method (isDynamicsActive), 35
mergeNodes, 36, 44
mergeNodes, RedePort-method (mergeNodes), 36
mergeOutEdges, 7, 37, 44
mergeOutEdges, RedePort-method (mergeOutEdges), 37
nesthc, 7, 38
nesthc, RedePort-method (nesthc), 38
nestNodes, 5, 7, 39, 40, 44
nestNodes, RedePort-method (nestNodes), 40
p.adjust, 18
ping, 41, 45
ping, RedePort-method (ping), 41
RedeR (RedeR-package), 2
RedeR-package, 2
RedeR.data, 42
RedePort, 4, 5, 11, 17, 19–24, 26–34, 36–39, 41, 42, 43, 45, 47–53, 55, 56
RedePort-class, 44
relax, 7, 46
relax, RedePort-method (relax), 46
resetd, 45, 47
resetd, RedePort-method (resetd), 47
selectAllEdges, 20, 45, 48
selectAllEdges, RedePort-method (selectAllEdges), 48
selectAllNodes, 21, 45, 49
selectAllNodes, RedePort-method (selectAllNodes), 49
selectEdges, 20, 23, 45, 50
selectEdges, RedePort-method (selectEdges), 50
selectGraph, 23, 45, 51
selectGraph, RedePort-method (selectGraph), 51
selectNodes, 7, 21, 23, 45, 52
selectNodes, RedePort-method (selectNodes), 52
setArrowDirection, 44, 53
setArrowDirection, RedPort-method  
(setArrowDirection), 53
subg, 54
updateContainerSize, 44, 54
updateContainerSize, RedPort-method  
(updateContainerSize), 54
updateGraph, 45, 55
updateGraph, RedPort-method  
(updateGraph), 55
version, 45, 56
version, RedPort-method (version), 56