

# Package ‘RedeR’

April 15, 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 <mauro.a.castro@gmail.com>

**Depends** R (>= 2.15), methods, igraph

**Imports** RCurl, XML, pvclust

**Suggests** PANR

**SystemRequirements** Java Runtime Environment (>= 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 (>= 2)

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

**URL** <http://genomebiology.com/2012/13/4/R29>

**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
callD . . . . .	16
cea . . . . .	17

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
RedeR.data . . . . .	42
RedPort . . . . .	43
RedPort-class . . . . .	44
relax . . . . .	46
reseta . . . . .	47
selectAllEdges . . . . .	48
selectAllNodes . . . . .	49
selectEdges . . . . .	50
selectGraph . . . . .	51
selectNodes . . . . .	52
setArrowDirection . . . . .	53
subg . . . . .	54
updateContainerSize . . . . .	54
updateGraph . . . . .	55
version . . . . .	56

<b>Index</b>	<b>58</b>
--------------	-----------

## 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**

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

**Author(s)**

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

**References**

Castro, MAA et al. *RedeR: R/Bioconductor package for representing modular structures, nested networks and multiple levels of hierarchical associations*. Genome Biology 13(4):R29, 2012.

**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**

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

**Value**

Add 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**

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

## Not run:

callD(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**

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

**See Also**[RedPort](#)**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)

## 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**

<code>obj</code>	Object of RedPort Class.
<code>g</code>	An igraph object.
<code>...</code>	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) <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(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 'gcoord' argument above).
- coordY** Sets the graph y center; y between 0 and 100 sets to visible area <numeric> (PS. alternative entry to the 'gcoord' 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= `ELLIPSE`) `<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-l B), 2 (A <-B), -2 (A l-B), 3 (A <-> B), -3 (A l-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**

```

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, ...)

```



**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>colvec</code>	Vector with legend colors, either hexadecimal or valid R color names.
<code>sizevec</code>	Vector with legend node size, options: $> 0$ <numeric>.
<code>shapevec</code>	Vector with valid shape names: 'ELLIPSE', 'RECTANGLE', 'ROUNDED_RECTANGLE', 'TRIANGLE', 'DIAMOND'.
<code>...</code>	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](#)

**Examples**

```

rdp <- RedPort('MyPort')

## Not run:

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', 'DOTTED_SHORT', '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. 'calld').

**Author(s)**

Mauro Castro

**See Also**[RedPort](#)**Examples**

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

## Not run:

  callD(rdp)
  addNodes(rdp, nodes)
  updateGraph(rdp)

## End(Not run)
```

---

addSeries	<i>Add series to RedeR application.</i>
-----------	---

---

**Description**

Method to send series of graphs to RedeR app.

**Usage**

```
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

**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)
```

---

`addSubgraph`*Add subgraphs to RedeR application.*

---

**Description**

Method to send subgraph to RedeR app.

**Usage**

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

**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>g</code>	An igraph object.
<code>nodes</code>	Nodes of the subgraph <array of strings>
<code>...</code>	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.

**Author(s)**

Mauro Castro

**See Also**[addGraph](#) [addSubgraph.list](#)**Examples**

```
rdp <- 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)
```

---

addSubgraph.list	<i>Add a list of subgraphs to RedeR application.</i>
------------------	--

---

**Description**

Method to send subgraphs to RedeR app.

**Usage**

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

**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>g</code>	An igraph object.
<code>nodeList</code>	List of nodes. Will be used to extra subgraphs from <code>g</code> .
<code>...</code>	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:

callD(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)
```

att

*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)
att.sete(g, from, to='edgeColor', pal=1, cols=NULL, na.col=grey(0.7), xlim=c(20,100,1), shapes=NULL)
att.mapv(g, dat, refcol=1)
att.mape(g, dat, refcol=c(1,2))
```

**Arguments**

<code>g</code>	An igraph object.
<code>from</code>	An attribute name available in 'g' <string>.
<code>to</code>	A valid RedeR attribute name (see <a href="#">addGraph</a> or type 'att.setv()' or 'att.sete()').
<code>breaks</code>	A numeric vector of two or more breakpoints to be applied to the attribute values.
<code>pal</code>	Default color palette. Options: 1 or 2.
<code>xlim</code>	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).
<code>cols</code>	Vector of colors (either hexadecimals or valid R color names).
<code>na.col</code>	A color representing eventual NAs. Default: grey(0.7)
<code>shapes</code>	A string vector with valid RedeR shapes (see <a href="#">addGraph</a> or type 'att.setv()' or 'att.sete()').
<code>categvec</code>	Optional: levels to encode attributes as a factor <vector>.
<code>nquant</code>	Optional: number of breakpoints to split attribute values by quantiles <integer>.
<code>isrev</code>	Optional: reversed version of attribute values <logical>.
<code>getleg</code>	Optional: return legend values <logical>.
<code>dat</code>	A data frame with the attributes to be mapped to 'g'.
<code>refcol</code>	The reference columns in the 'data' object with either node ids (one column <integer>) or edge ids (two columns <vector of two integers>).
<code>roundleg</code>	Integer indicating the number of decimal places (round) in the legend of numerical attributes.
<code>title</code>	Optional: legend title.

**Value**

Map/set RedeR attributes to igraph objects.

**Author(s)**

Mauro Castro

**See Also**[addGraph](#)**Examples**

```

data(ER.deg)

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

# maps the data frame to the igraph object
sg <- att.mapv(g=sg, dat=dt, refcol=1)

# Sets graph attributes to RedeR!

# sets gene symbol do nodeAlias attribute
sg <- att.setv(sg, from="Symbol", to="nodeAlias")

# sets numerical value to nodeColor attribute
sg <- att.setv(sg, from="logFC.t3", to="nodeColor", breaks=seq(-1,1,0.2), pal=2)

# sets numerical value to nodeSize attribute
sg <- att.setv(sg, from="ERbdist", to="nodeSize", nquant=10, isrev=TRUE, xlim=c(5,40,1))

```

---

`calld`*Call RedeR app from R.*

---

**Description**

Method to invoke RedeR application from R.

**Usage**

```
calld(obj, ...)
```

**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>...</code>	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 <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.



**Author(s)**

Mauro Castro

**See Also**[RedPort addGraph](#)**Examples**

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

## Not run:

  callD(rdp)

## End(Not run)
```

cea

*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>).

**Value**

An adjacency matrix with significant correlation values.

**Author(s)**

Mauro Castro

**See Also**

[cor.p.adjust](#)

**Examples**

```
data(ER.deg)
#--- a gene expression matrix
exp <- ER.deg$exp
#--- a sample from gx!!
idx <- sample(1:nrow(exp))[1:100]
exp <- exp[idx,]

## Not run:

res <- cea(x=exp, nper=100) #ps set 'nper' for at least 1000

## 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	Edge sequence as an array <array of strings>

**Value**

Removes the specified edges from the graph.

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

## Not run:

callD(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. 'callD').

**Author(s)**

Mauro Castro

**See Also**

[RedPort](#)

## 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)
  deleteNodes(rdp, c("n1", "n3") )
  updateGraph(rdp)

## End(Not run)
```

---

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. 'callD').

## 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

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

## 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")
deSelectEdges(rdp)
updateGraph(rdp)

## End(Not run)
```

---

deSelectGraph	<i>Unmark selected graph objects.</i>
---------------	---------------------------------------

---

## 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. 'callD').

## Author(s)

Mauro Castro

## See Also

[RedPort](#), [selectNodes](#), [selectEdges](#), [selectGraph](#)

## 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)
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. 'callD').

## Author(s)

Mauro Castro

## See Also

[RedPort](#)



## 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)
selectNodes(rdp,c("n3","n4","n5"))
deSelectNodes(rdp)
updateGraph(rdp)

## End(Not run)
```

---

duplicateGraph

*Duplicate graphs in RedeR application.*

---

## Description

Method to duplicate graphs and subgraphs of a network.

## Usage

```
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](#)

**Examples**

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

## Not run:

callD(rdp)

###

g1 <- graph.lattice(c(3,3,3))
addGraph( rdp, g1, layout.kamada.kawai(g1) )
duplicateGraph(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

**See Also**

[RedPort](#)

**Examples**

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

## Not run:

callD(rdp)
exitd(rdp)

## End(Not run)
```

---

getContainerComponets *Get container components.*

---

### 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

```
rdp <- 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)
getContainerComponets(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

```
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`*Get edges.*

---

**Description**

Method to get all edges from an active RedeR application.

**Usage**

```
getEdges(obj, ...)
```

**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>...</code>	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`*Get RedeR graph.*

---

**Description**

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

**Usage**

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

**Arguments**

<code>obj</code>	Object of RedPort Class.
<code>...</code>	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'`).

**attribs** 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	<i>Get node IDs.</i>
------------	----------------------

---

### 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

```
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)
  getNodeIDs(rdp)

## End(Not run)
```

---

`getNodes`*Get nodes.*

---

### Description

Method to get node list from an active RedeR session.

### Usage

```
getNodes(obj, ...)
```

### Arguments

<code>obj</code>	Object of RedPort Class.
<code>...</code>	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", "n1", "n3", "n1", "n4", "n1", "n5", "n1", "n6", "n1", "n7")

## Not run:

  callD(rdp)
  addEdges(rdp, edges)
  updateGraph(rdp)
  getNodes(rdp)

## End(Not run)
```



---

getSourceEdgeIDs	<i>Get source-edge IDs.</i>
------------------	-----------------------------

---

### Description

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. '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)
  getSourceEdgeIDs(rdp)

## End(Not run)
```

---

getTargetEdgeIDs	<i>Get target-edge IDs.</i>
------------------	-----------------------------

---

### 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

```
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**

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

**Arguments**

<code>m</code>	Number of modules.
<code>nmax</code>	The maximum number of vertices in each module.
<code>nmin</code>	The minimum number of vertices in each module.
<code>p1</code>	Probability for adding new vertices to a module.
<code>p2</code>	Probability for drawing an edge between modules.
<code>p3</code>	Probability for drawing an edge within modules.

**Value**

Returns a `igraph` object.

**Author(s)**

Mauro Castro

**Examples**

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

---

`isDynamicsActive`*Inquires about RedeR current state.*

---

**Description**

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

**Usage**

```
isDynamicsActive(obj)
```

**Arguments**

<code>obj</code>	Object of RedPort Class.
------------------	--------------------------

**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**

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**

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

## Not run:

callD(rdp)
addNodes(rdp, nodes)
mergeNodes(rdp,c("n2","n3","n4"))
updateGraph(rdp)

## End(Not run)
```

---

mergeOutEdges	<i>Merge out-edges between connected containers and transfers edges from nodes to containers.</i>
---------------	---

---

**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 assignments.

**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')
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**

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).

## 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

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

#rdp <- RedPort('MyPort')

## Not run:

  callD(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

```
nestNodes(obj, nodes, ...)
```

## Arguments

<code>obj</code>	Object of RedPort Class.
<code>nodes</code>	<array of strings>
<code>...</code>	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.



**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**

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

## Not run:

  callD(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. 'callD').

**Author(s)**

Mauro Castro

**See Also**[RedPort](#)**Examples**

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

## Not run:

callD(rdp)

ping (rdp)

# "R interface is ready to use!"

## End(Not run)
```

---

RedeR.data

*Pre-processed dataset for RedeR case study.*

---

**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**

```
data(Carroll12006)
```

**Format**

Carroll12006 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.dfc.harvard.edu/brownlab/datasets/index> (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 binding 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 log<sub>2</sub> gene expression values of DE genes.

**ER.deg\$ceg** Co-expression gene network of early ER-responsive genes computed by the function `cea` [cea](#).

## References

Carroll JS et al., Genome-wide analysis of estrogen receptor binding sites. Nat Genet. 38(11):1289-97, 2006.

## Examples

```
data(Carroll12006)
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)
```

## Arguments

<code>title</code>	A character string representing the XML-RPC port.
<code>host</code>	The domain name of the machine that is running the RedeR XML-RPC server.
<code>port</code>	An integer specifying the port on which the XML-RPC server should listen.

## Value

An object of the RedPort Class.

## Author(s)

Mauro Castro

**See Also**[calld](#)**Examples**

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

---

RedPort-class

*Class "RedPort"*

---

**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:**[getNode](#)[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](#)

**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**  
**calld**  
**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**

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

---

relax	<i>relax</i>
-------	--------------

---

### 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; $\geq 1$ ) <numeric>.
p2	Edge stiffness (arbitrary unities; $\geq 1$ ) <numeric>.
p3	Node repel factor (arbitrary unities; $\geq 1$ ) <numeric>.
p4	Node perimeter effect (in pixels; $\geq 1$ ) <numeric>.
p5	Node speed limit (arbitrary unities; $\geq 1$ ) <numeric>.
p6	Nest-nest edge target length, i.e., edge target between linked containers (in pixels; $\geq 1$ ) <numeric>.
p7	Nest-node repel factor, i.e., repulsion among containers and out-nodes (arbitrary unities; $\geq 1$ ) <numeric>.
p8	Repulsion radius, i.e., this parameter limits the repel factor range (given in p1 unites; $\geq 1$ ) <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

## References

Brandes U. Drawing graphs: methods and models. In: Lecture notes in computer science. Kaufmann M. and Wagner D. (Ed), vol. 2025. Heidelberg: Springer; 2001: 71-86.

Fruchterman TMJ, Reingold EM. Graph drawing by force-directed placement. Software: Practice and Experience 1991, 21(11):1129-1164.

## Examples

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

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

## Not run:

callD(rdp)
addGraph( rdp, g, layout.random(g) )
relax(rdp)

## End(Not run)
```

---

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](#)

**Examples**

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

## Not run:

  calld(rdp)
  resetd(rdp)

## End(Not run)
```

---

selectAllEdges	<i>Select all edges.</i>
----------------	--------------------------

---

**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**

```
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**

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

## Not run:

  calld(rdp)
  addEdges(rdp, edges)
  selectAllEdges(rdp)
```



```
updateGraph(rdp)
## End(Not run)
```

---

selectAllNodes	<i>selectAllNodes</i>
----------------	-----------------------

---

## 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. 'callD').

## Author(s)

Mauro Castro

## See Also

[RedPort](#), [deleteSelectedNodes](#)

## 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)
selectAllNodes(rdp)
updateGraph(rdp)

## End(Not run)
```

---

selectEdges	<i>selectEdges</i>
-------------	--------------------

---

### 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

```
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")
  updateGraph(rdp)

## End(Not run)
```

---

selectGraph	<i>Select graph.</i>
-------------	----------------------

---

**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

**See Also**

[RedPort](#), [deleteSelectedNodes](#), [deleteSelectedEdges](#), [deSelectGraph](#)

**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)
  selectGraph(rdp)
  updateGraph(rdp)

## End(Not run)
```

---

selectNodes	<i>Select nodes.</i>
-------------	----------------------

---

### 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, 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

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

## Not run:

  callD(rdp)
  addEdges(rdp, edges)
  selectNodes(rdp,c("n3","n4","n5"))
  updateGraph(rdp)

## End(Not run)
```

---

setArrowDirection      *Set arrow direction.*

---

### 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

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

## Not run:

  callD(rdp)
  addEdges(rdp, edges)
  setArrowDirection(rdp, "n1", "n2", 2)
  updateGraph(rdp)

## End(Not run)
```

---

subg	<i>Subgraph of a graph.</i>
------	-----------------------------

---

**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	<i>Update container size.</i>
---------------------	-------------------------------

---

**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. 'callD').

**Author(s)**

Mauro Castro

**See Also**

[RedPort](#)

**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)
  nestNodes( rdp, c("n2", "n3") )
  updateContainerSize(rdp)
  updateGraph(rdp)

## End(Not run)
```

---

updateGraph	<i>Update RedeR graphs.</i>
-------------	-----------------------------

---

**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**

```
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)

## End(Not run)
```

---

version

*Version*

---

**Description**

Check RedeR application version.

**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**
    - RedeR.data, 42
  - \*Topic **graphs**
    - RedPort-class, 44
  - \*Topic **graph**
    - addEdgeBetweenContainers, 3
    - addEdges, 4
    - addGraph, 5
    - 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
    - reseted, 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**
    - RedeR-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

- Carroll2006 (RedeR.data), 42
- cea, 17, 43
- cor, 18
- deleteEdges, 18, 44
- deleteEdges, RedPort-method  
(deleteEdges), 18
- deleteNodes, 19, 44
- deleteNodes, RedPort-method  
(deleteNodes), 19
- deleteSelectedEdges, 20, 45, 48, 50, 51
- deleteSelectedEdges, RedPort-method  
(deleteSelectedEdges), 20
- deleteSelectedNodes, 21, 45, 49, 51, 52
- deleteSelectedNodes, RedPort-method  
(deleteSelectedNodes), 21
- deSelectEdges, 20, 22, 45
- deSelectEdges, RedPort-method  
(deSelectEdges), 22
- deSelectGraph, 23, 45, 51
- deSelectGraph, RedPort-method  
(deSelectGraph), 23
- deSelectNodes, 21, 24, 45
- deSelectNodes, RedPort-method  
(deSelectNodes), 24
- duplicateGraph, 25, 45
- duplicateGraph, RedPort-method  
(duplicateGraph), 25
- ER.deg (RedeR.data), 42
- ER.limma (RedeR.data), 42
- exitd, 26, 45
- exitd, RedPort-method (exitd), 26
- getContainerComponets, 27, 44
- getContainerComponets, RedPort-method  
(getContainerComponets), 27
- getEdgeIDs, 28, 44
- getEdgeIDs, RedPort-method (getEdgeIDs),  
28
- getEdges, 29, 44
- getEdges, RedPort-method (getEdges), 29
- getGraph, 7, 28, 29, 30, 31–34, 44, 45
- getGraph, RedPort-method (getGraph), 30
- getNodeIDs, 31, 44
- getNodeIDs, RedPort-method (getNodeIDs),  
31
- getNodes, 32, 44
- getNodes, RedPort-method (getNodes), 32
- getSourceEdgeIDs, 33, 44
- getSourceEdgeIDs, RedPort-method  
(getSourceEdgeIDs), 33
- getTargetEdgeIDs, 34, 44
- getTargetEdgeIDs, RedPort-method  
(getTargetEdgeIDs), 34
- gtoy.rm, 35, 39
- hs.inter (RedeR.data), 42
- isDynamicsActive, 35, 45
- isDynamicsActive, RedPort-method  
(isDynamicsActive), 35
- mergeNodes, 36, 44
- mergeNodes, RedPort-method (mergeNodes),  
36
- mergeOutEdges, 7, 37, 44
- mergeOutEdges, RedPort-method  
(mergeOutEdges), 37
- nesthc, 7, 38
- nesthc, RedPort-method (nesthc), 38
- nestNodes, 5, 7, 39, 40, 44
- nestNodes, RedPort-method (nestNodes), 40
- p.adjust, 18
- ping, 41, 45
- ping, RedPort-method (ping), 41
- RedeR (RedeR-package), 2
- RedeR-package, 2
- RedeR.data, 42
- RedPort, 4, 5, 11, 17, 19–24, 26–34, 36–39,  
41, 42, 43, 45, 47–53, 55, 56
- RedPort-class, 44
- relax, 7, 46
- relax, RedPort-method (relax), 46
- resetd, 45, 47
- resetd, RedPort-method (resetd), 47
- selectAllEdges, 20, 45, 48
- selectAllEdges, RedPort-method  
(selectAllEdges), 48
- selectAllNodes, 21, 45, 49
- selectAllNodes, RedPort-method  
(selectAllNodes), 49
- selectEdges, 20, 23, 45, 50
- selectEdges, RedPort-method  
(selectEdges), 50
- selectGraph, 23, 45, 51
- selectGraph, RedPort-method  
(selectGraph), 51
- selectNodes, 7, 21, 23, 45, 52
- selectNodes, RedPort-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](#)