

# Package ‘classGraph’

November 18, 2009

**Type** Package

**Title** Construct Graphs of S4 Class Hierarchies

**Version** 0.7-3

**Date** 2009-11-18

**Author** Martin Maechler partly based on code from Robert Gentleman

**Maintainer** Martin Maechler <maechler@stat.math.ethz.ch>

**Depends** methods, graph, Rgraphviz

**Imports** methods, graph, Rgraphviz

**Suggests** Matrix

**Description** Construct directed graphs of S4 class hierarchies and visualize them. In general, these graphs typically are DAGs (directed acyclic graphs), often simple trees in practice.

**License** GPL

**Repository** CRAN

**Date/Publication** 2009-11-18 14:14:08

## R topics documented:

classGraph-package . . . . .	2
bGraph . . . . .	2
class2Graph . . . . .	3
classTree . . . . .	5
mRagraph . . . . .	6
numOutEdges . . . . .	7
plotRag . . . . .	8
subClasses . . . . .	9
superClasses . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

classGraph-package *The R Package 'classGraph'*

---

### Description

The package **classGraph** is package using graph and graph visualization methods to visualize inheritance graphs of S4 classes.

### Details

Package: classGraph  
Type: Package  
Version: 0.7-0  
Date: 2007-02-10  
License: GPL

### Author(s)

Martin Maechler

### See Also

[classTree\(\)](#) is the main function of this package.

---

bGraph

*Create a "Branch Graph" (Simple Tree with Root and Leaves)*

---

### Description

Create a "Branch Graph", i.e., a simple tree with root and  $n$  (simple) branches or leaves.

### Usage

```
bGraph(n, root = "Mom",  
       leaves = paste(l.prefix, seq(length = n), sep = ""),  
       l.prefix = "D", weights = NULL,  
       mode = c("undirected", "directed"))
```

**Arguments**

n	integer specifying the number of leave branches.
root	the node on which to root the tree.
leaves	the nodes to be used as leaves.
l.prefix	a string specifying .....
weights	.....
mode	string indicating which mode is to be used.

**Value**

a graph object of class `graphNEL`.

**Author(s)**

Martin Maechler, Aug.2005

**See Also**

class `graphNEL`; `ftM2graphNEL`.

**Examples**

```
(bg7 <- bGraph(7)) # 8 nodes {Mom, D1..D7}; 7 edges
plot(bg7) # draws the graph

(bgD3 <- bGraph(3, mode="directed"))
plot(bgD3) # directed: using arrows

(bgw2 <- bGraph(2, weights = c(10,1)))
plot(bgw2) # {maybe use lwd for weights in the future?}
if(require("Matrix"))
  show(as(bgw2, "sparseMatrix")) # shows the weights
```

---

class2Graph

*Build the Graph of Super Classes from an S4 Class Definition*


---

**Description**

From an S4 class definition `class`, computes the graph of all super classes, i.e., of all classes that `class` extends.

**Usage**

```
class2Graph(class, fullNames = TRUE, simpleOnly = FALSE,
            bottomUp = FALSE, package = class@package)
```

**Arguments**

class	class name
fullNames	logical indicating if full name should be applied....
simpleOnly	logical, simply passed to <a href="#">getAllSuperClasses</a> (. . ).
bottomUp	logical indicating the <i>direction</i> of the graph.
package	package where the super classes should be gotten from.

**Value**

an R object inheriting from class [graph](#).

**Author(s)**

Robert Gentleman (original code) and Martin Maechler

**See Also**

[classTree](#) which builds the graph of all *subclasses*.

**Examples**

```
require("graph")
cg <- class2Graph("graphNEL") # simple : graphNEL |-> graph
plot(cg)

if(require("Matrix")) {
  cg2 <- class2Graph("dgCMatrix")
  as(cg2, "sparseMatrix")
  plot(cg2)
  ## alternative: don't show the initial "Matrix:"
  cg2. <- class2Graph("dgCMatrix", fullNames=FALSE)
  plot(cg2.)
  ## 'simpleOnly' does not change anything here :
  stopifnot(identical(cg2.,
    class2Graph("dgCMatrix", fullNames=FALSE, simpleOnly = TRUE)))

  ## very simple, since "sparseMatrix" only extends "Matrix" :
  cg3 <- class2Graph("sparseMatrix")
  plot(cg3)
}
```

---

classTree	<i>builds a directed graph, typically a tree from a class Object</i>
-----------	--

---

## Description

From an S4 class, by investigating all subclasses, an inheritance graph is built, a directed graph, often a tree.

## Usage

```
classTree(Cl, all = FALSE, ...)
```

## Arguments

Cl	class name ...
all	logical indicating if all instead of just direct sub-classes should be used.
...	....

## Value

an R object inheriting from class [graph](#).

## Author(s)

Martin Maechler

## See Also

[class2Graph](#), ...

## Examples

```
trGclass <- classTree("graph")
as(trGclass, "matrix")
plot(trGclass) # using package 'Rgraphviz'
```

## Description

My constructor of an [Ragraph](#) object, a kind of “laid-out” graph, from package **Rgraphviz**. This allows more customization in plotting than just calling `plot(gr, ...)` for a [graph](#) object from package **graph**.

## Usage

```
mRagraph(gr, lType, fixedsize = FALSE,
         fill = c("lightblue", "gray90"),
         color = c("blue3", "gray60"),
         labcol = c("blue3", "green4", "purple"))
```

## Arguments

<code>gr</code>	an R object of class <a href="#">graph</a> (from package <b>graph</b> ), in our case typically the result of <code>classTree()</code> .
<code>lType</code>	a string specifying the layout type, see <code>agopen()</code> in package <b>Rgraphviz</b> for the possibilities.
<code>fixedsize</code>	logical indicating if the ellipses should all get the same size – or should rather adapt to the situation.
<code>fill</code>	character vector of length 2....
<code>color</code>	character vector of length 2....
<code>labcol</code>	vector of labels to be used ....

## Value

an object of class [Ragraph](#), produced by an appropriate call to `agopen`.

## Author(s)

Martin Maechler

## See Also

the customized plotting function `plotRag`.

**Examples**

```

if(require("Matrix")) {
  trMatrix <- classTree("Matrix")
  trMatrix
  RtrM <- mRagraph(trMatrix)
  RtrM # (the show method will hopefully improve)
  str(RtrM, max=2) # shows a bit more

  plot(RtrM)
}

```

---

numOutEdges

*For each Node of a Directed Graph give the Number Outgoing Edges*


---

**Description**

In a directed or undirected graph, for each node count the number of edges “leaving” that nodes.

**Usage**

```
numOutEdges(g)
```

**Arguments**

`g` an R object of class `graph` (from package **graph**).

**Value**

an `integer` vector the same length as `nodes(g)` giving the number of edges that “go out” from each node.

**Author(s)**

Martin Maechler

**See Also**

`edgeL` on which this function is built, and `leaves`, both from package **graph**.

**Examples**

```

## Simplistic leaves() definition for *directed graphs* :
## { compare with graph::leaves() }
is.leaf <- function(g) numOutEdges(g) == 0 ## (also exists hiddenly..)
Leaves <- function(g) nodes(g)[is.leaf(g)]
Leaves(bGraph(4, mode = "directed"))

```

---

plotRag

*Plot an Ragraph (using Rgraphviz)*


---

### Description

Plot an [Ragraph](#) object (a kind of “laid-out” graph, from package **Rgraphviz**). This the simply uses the `plot` method from package **Rgraphviz** (i.e., `selectMethod(plot, "Ragraph")`) and additionally adds a “footnote”-like subtitle.

### Usage

```
plotRag(ragr, sub, subArgs = .optRagargs(), ...)
```

### Arguments

<code>ragr</code>	an object of class <a href="#">Ragraph</a> (as defined in the <b>Rgraphviz</b> package).
<code>sub</code>	a “footnote” or subtitle to be added to <code>plot(ragr, ...)</code> . By default gives the number of nodes and edges.
<code>subArgs</code>	~~Describe subArgs here~~
<code>...</code>	further arguments passed to <code>plot(.)</code> , i.e., the plot method for <a href="#">Ragraph</a> objects.

### Author(s)

Martin Maechler

### See Also

[mRagraph](#), [Ragraph](#) ~~~

### Examples

```
if(require("Matrix")) {
  trMatrix <- classTree("Matrix")
  trMatrix
  RtrM <- mRagraph(trMatrix)
  RtrM # (the show method will hopefully improve)
  str(RtrM, max=2) # shows a bit more

  plot(RtrM) ## almost the same as
  plotRag(RtrM, subArgs=.optRagargs(adj = 0.5))
  ## which just gives "<n> nodes with <m> edges"
}
```

---

`subClasses`*All Subclasses of a Given S4 Class*

---

**Description**

Return all subclasses of a given S4 class; either only the direct sub classes are also those “further away” (*distance* > 1).

**Usage**

```
subClasses(Cl, directOnly = TRUE, complete = TRUE, ...)
```

**Arguments**

<code>Cl</code>	a class representation or a class name ( <a href="#">character</a> ).
<code>directOnly</code>	logical indicating if you <i>direct</i> subclasses are desired (or also the ones with <i>distance</i> > 1).
<code>complete</code>	logical,.. as in....
<code>...</code>	.....

**Value**

a [character](#) vector of class names.

**Author(s)**

Martin Maechler

**See Also**

[superClasses](#); [Classes](#) in general.

**Examples**

```
subClasses("graph") # -> the direct ones
subClasses("graph", directOnly = FALSE) # the same: has only direct subclasses
if(require("Matrix")) {
  print( subClasses("sparseMatrix") )
  print( subClasses("sparseMatrix", directOnly = FALSE) )# much more
}
```

---

superClasses      *List of Super Classes of a Given S4 Class*

---

**Description**

Give a [list](#) of all super classes of a specific S4 class (definition).

**Usage**

```
superClasses(x)
```

**Arguments**

`x`                    a class representation as returned by [getClassDef](#).

**Value**

a list of length-1 [character](#) strings, typically with a "package" attribute each.

**Author(s)**

Robert Gentleman and Martin Maechler

**See Also**

[subClasses](#), ...

**Examples**

```
superClasses(getClassDef("graphNEL"))

if(require("Matrix")) {
  scL <- superClasses(getClassDef("dgeMatrix"))
  str(scL) # a list of two
} # 'Matrix'
```

# Index

- \*Topic **classes**
  - class2Graph, 3
  - classTree, 4
  - subClasses, 8
  - superClasses, 9
- \*Topic **graphs**
  - bGraph, 2
  - class2Graph, 3
  - classTree, 4
  - mRagraph, 5
  - numOutEdges, 6
- \*Topic **hplot**
  - plotRag, 7
- \*Topic **manip**
  - mRagraph, 5
- \*Topic **package**
  - classGraph-package, 1
- \*Topic **utilities**
  - numOutEdges, 6
- agopen, 5
- bGraph, 2
- character, 8, 9
- class2Graph, 3, 5
- Classes, 8
- classGraph (*classGraph-package*), 1
- classGraph-package, 1
- classTree, 2, 4, 4, 5
- edgeL, 6
- ftM2graphNEL, 3
- getAllSuperClasses, 3
- getClassDef, 9
- graph, 3-6
- graphNEL, 2, 3
- integer, 6
- leaves, 6
- list, 9
- mRagraph, 5, 7
- nodes, 6
- numOutEdges, 6
- plot, 7
- plotRag, 6, 7
- Ragraph, 5, 7
- subClasses, 8, 9
- superClasses, 8, 9