Graphs in the gRbase package

Søren Højsgaard

August 4, 2012

Contents

1 Introduction

For the R community, the packages igraph, graph, RBGL and Rgraphviz are extremely useful tools for graph operations, manipulation and layout. The gRbase package adds some additional tools to these fine packages. The most important tools are:

1. Undirected and directed acyclic graphs can be specified using formulae or an adjacency list using the functions ug() and dag(). This gives graphs represented as graphNEL objects (the default), as igraphs or as adjacency matrices.

2. Some graph algorithms are implemented in `gRbase`. These can be applied to graphs represented as `graphNEL`s or as matrices.

   The most important algorithms are: `mcs()`,(maximum cardinality search) `moralize()`, (moralization of directed acyclic graph), `rip()`, (RIP ordering of cliques of triangulated undirected graph), `triangulate()`, (triangulate undirected graph).

   Furthermore corresponding to some of the functions in the `graph` and `RBGL` packages there are corresponding matrix versions of these implemented in `gRbase`. These are: `maxCliqueMAT()`.

2 Graphs

Undirected graphs can be created by the `ug()` function and directed acyclic graphs (DAGs) by the `dag()` function.

The graphs can be specified either using formulae or a list of vectors; see examples below.

The representation of a graph can be as a `graphNEL` object, as an `igraph` object or as an adjacency matrix.

2.1 Undirected graphs

An undirected graph is created by the `ug()` function.

As `graphNEL`: The following specifications are equivalent:

```
ug11 <- ug(~a*b*c + c*d + d*e + a*e + f*g)
ug12 <- ug(c("a","b","c"),c("c","d"),c("d","e"),c("a","e"),c("f","g"))
ug13 <- ug(~a*b*c, ~c*d, ~d*e + a*e + f*g)
```

Notice that a ":" can be used instead of "*" in the formula specifications above.

```
ug11

A graphNEL graph with undirected edges
Number of Nodes = 7
Number of Edges = 7
```

As `igraph`: A representation as an `igraph` object can be obtained with one of the following equivalent specifications:

```
ug11i <- ug(~a*b*c + c*d + d*e + a*e + f*g, result="igraph")
ug12i <- ug(c("a","b","c"),c("c","d"),c("d","e"),c("a","e"),c("f","g"),
            result="igraph")
```

```
ug11i

IGRAPH UNW- 7 7 --
+ attr: name (v/c), label (v/c), weight (e/n)
```

As adjacency matrix: A representation as an adjacency matrix can be obtained with one of the following equivalent specifications:

```
ug11m <- ug(~a*b*c + c*d + d*e + a*e + f*g, result="matrix")
ug12m <- ug(c("a","b","c"),c("c","d"),c("d","e"),c("a","e"),c("f","g"),
            result="matrix")
```

```
ug11m

  a b c d e f g
a 0 1 1 0 1 0 0
b 1 0 1 0 0 0 0
c 1 1 0 1 0 0 0
d 0 0 1 0 1 0 0
e 1 0 0 1 0 0 0
f 0 0 0 0 0 0 1
g 0 0 0 0 0 1 0
```

2.2 Directed acyclic graphs (DAGs)

A directed acyclic graph is created by the `dag()` function.

As `graphNEL`: The following specifications are equivalent:

```
dag11 <- dag(~a + b*a + c*a*b + d*c*e + e*a + g*f)
dag12 <- dag("a", c("b","a"), c("c","a","b"), c("d","c","e"),
             c("e","a"),c("g","f"))
dag13 <- dag(~a, ~b*a,  ~c*a*b, ~d*c*e, ~e*a, ~g*f)
```

```
 dag11

A graphNEL graph with directed edges
Number of Nodes = 7
Number of Edges = 7
```

Here `~a` means that "a" has no parents while `~d*b*c` means that "d" has parents "b" and "c". Notice that a ":" can be used instead of "*" the specification.

As `igraph`: A representation as an `igraph` object can be obtained with

```
dag11i <- dag(~a + b*a + c*a*b + d*c*e + e*a + g*f, result="igraph")
dag12i <- dag("a", c("b","a"), c("c","a","b"), c("d","c","e"),
              c("e","a"),c("g","f"), result="igraph")
```

As adjacency matrix: A representation as an adjacency matrix can be obtained with

```
dag11m <- dag(~a + b*a + c*a*b + d*c*e + e*a + g*f, result="matrix")
dag12m <- dag("a", c("b","a"), c("c","a","b"), c("d","c","e"),
              c("e","a"),c("g","f"), result="matrix")
```

3 Graph coercion

Graphs can be coerced between differente representations using `as()`; for example

```
 as(ug11,"igraph")

IGRAPH UNW- 7 7 --
+ attr: name (v/c), label (v/c), weight (e/n)

 as(as(ug11,"igraph"),"matrix")

7 x 7 sparse Matrix of class "dgCMatrix"
  a b c d e f g
a . 1 1 . 1 . .
b 1 . 1 . . . .
c 1 1 . 1 . . .
d . . 1 . 1 . .
e 1 . . 1 . . .
f . . . . . . 1
g . . . . . 1 .

 as(as(as(ug11,"igraph"),"matrix"),"graphNEL")

A graphNEL graph with undirected edges
Number of Nodes = 7
Number of Edges = 7
```

4 Plotting graphs

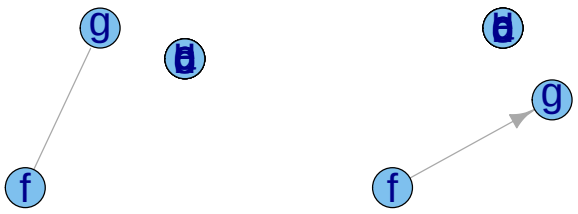
Graphs represented as `graphNEL`: Graphs (represented as `graphNEL` objects) are displayed with `plot()`, but this requires that 1) the `Rgraphviz` package is installed and also 2) that the `Graphviz` program is installed.

There is also an `iplot()` function which converts a graphNEL to an igraph and plots this. To use `iplot()`, no additional software must be installed. There is a lot of room for improvement of the `iplot()` function.

Graphs represented as `igraph`: Graphs (represented as `igraph` objects) are displayed with `plot()`.

Graphs represented as adjacency matrices: There is no plot method for graphs represented as adjacency matrices.

```
par(mfrow=c(1,2))
iplot(ug11)
iplot(dag11)
```



5 Graph queries

The `graph` and `RBGL` packages implement various graph operations for `graphNEL` objects. See the documentation for these packages. The `gRbase` implements a few additional functions, see Section 1. An additional function in `gRbase` for graph operations is `querygraph()`. This function is intended as a wrapper for the various graph operations available in `gRbase`, `graph` and `RBGL`. There are two main virtues of `querygraph()`: 1) `querygraph()` operates on any of the three graph representations described above and 2) `querygraph()` provides a unified interface to the graph operations. The general syntax is

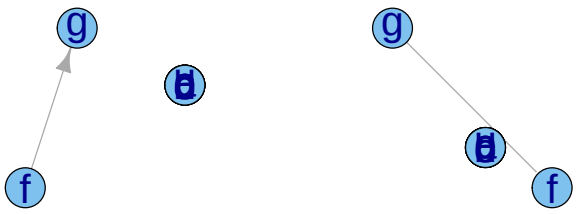
```
args(querygraph)

function (object, op, set = NULL, set2 = NULL, set3 = NULL)
NULL
```

6 More advanced graph operations

A moralized directed acyclic graph is obtained with

```
dag11.mor <- moralize(dag11)
```

```
par(mfrow=c(1,2))
iplot(dag11)
iplot(dag11.mor)
```



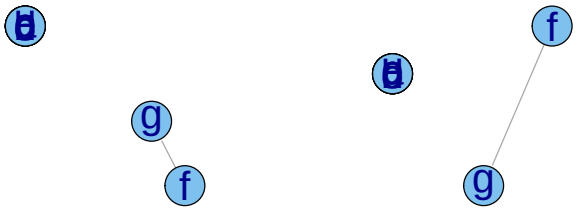
Testing for whether a graph is triangulated is based on Maximum Cardinality Search. If `character(0)` is returned the graph is not triangulated. Otherwise a linear ordering of the nodes is returned.

```
mcs(ug11)

character(0)
```

Triangulate an undirected graph by adding extra edges to the graph:

```
tug11<-triangulate(ug11)

A graphNEL graph with undirected edges
Number of Nodes = 7
Number of Edges = 8
```

```
par(mfrow=c(1,2))
iplot(ug11)
iplot(tug11)
```



A RIP ordering of the cliques of a triangulated graph can be obtained as:

```
r <- rip(tug11)
r

cliques
  1 : c a b
  2 : e a c
  3 : d c e
  4 : g f
separators
  1 :
  2 : a c
  3 : c e
  4 :
parents
  1 : 0
  2 : 1
  3 : 2
  4 : 0
```

7 Time and space considerations

7.1 Space

It is worth noticing that working with graphs representated as `graphNEL` objects is somewhat slower working with graphs represented as adjacency matrices. On the other hand, graph

Consider for example coerction from a `graphNEL` object. This can be obtained with `as()` as shown above or by using `as.adjMAT()` from `gRbase`. The timings are:

```
system.time({for (ii in 1:200) as(ug11, "matrix")})

  user  system elapsed
  0.09    0.00    0.09

system.time({for (ii in 1:200) as.adjMAT(ug11)})

  user  system elapsed
  0.07    0.00    0.06
```

Similarly, consider finding the cliques of an undirected graph represented as a `graphNEL` object or as a matrix:

```
system.time({for (ii in 1:200) maxClique(ug11)})

  user  system elapsed
  0.13    0.00    0.12

system.time({for (ii in 1:200) maxCliqueMAT(ug11m)})

  user  system elapsed
  0.01    0.00    0.01
```

7.2 Space

On the other hand, the `graphNEL` representation is – at least – in principle more economic in terms of space requirements than the adjacency matrix representation (because the adjacency matrix representation uses a 0 to represent a "missing edge". However, in practice the picture is not so clear. Consider the following examples

```
 V <- 1:100
 M <- 1:10
 ## Sparse graph
 ##
 g1 <- randomGraph(V, M, 0.05)
 length(edgeList(g1))

[1] 106

 object.size(g1)

126392 bytes

 object.size(as.adjMAT(g1))

51648 bytes

 ## More dense graph
 ##
 g1 <- randomGraph(V, M, 0.2)
 length(edgeList(g1))

[1] 1542

 object.size(g1)

1206008 bytes

 object.size(as.adjMAT(g1))

51648 bytes

 ## Even more dense graph
 ##
 g1 <- randomGraph(V, M, 0.5)
 length(edgeList(g1))

[1] 4740

 object.size(g1)

3610904 bytes

 object.size(as.adjMAT(g1))

51648 bytes
```