Graphs in the **gRbase** package

Søren Højsgaard

January 8, 2013

Contents

1 Introduction **1**

2 Graphs **2**
2.1 Undirected graphs 2
2.2 Directed acyclic graphs (DAGs) 4

3 Graph coercion **5**

4 Plotting graphs **6**

5 Graph queries **7**

6 Advanced graph operations **8**
6.1 Moralization 8
6.2 Maximum cardinality search 9
6.3 Triangulation 10
6.4 RIP ordering / junction tree 12

7 Time and space considerations **13**
7.1 Time 13
7.2 Space 14

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 in one of the following four forms:

   - `graphNEL` objects (the default),
   - adjacency matrices. There are two type of adjacency matrices in this context: A "standard" matrix in `R` and a sparse matrix (or to be precise: a `dgCMatrix` in the `Matrix` package).
   - `igraphs`

2. Some graph algorithms are implemented in **gRbase**. These can be applied to graphs represented as `graphNEL`s and matrices. Some can also be applied to `igraph`s but we are not consistent with respect to this. 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).

3. 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.

2.1 Undirected graphs

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

As `graphNEL`: The following specifications are equivalent (notice that "`:`" and "`*`" can be used interchangably):

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

```
ug11

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

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
```

```
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

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 `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)
```

2.2 Directed acyclic graphs (DAGs)

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

As `graphNEL`: The following specifications are equivalent (notice that “`:`” and “`*`” can be used interchangably):

```
dag11 <- dag(~a + b:a + c:a:b + d:c:e + e:a + g:f)
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)
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”.

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")
```

```
dag11m

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

```
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")
```

```
dag11M

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

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")
```

3 Graph coercion

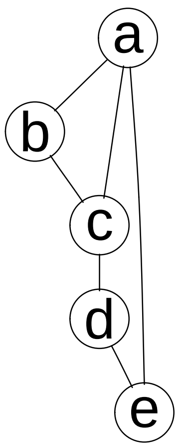
Graphs can be coerced between differente representations using `as()`; for example

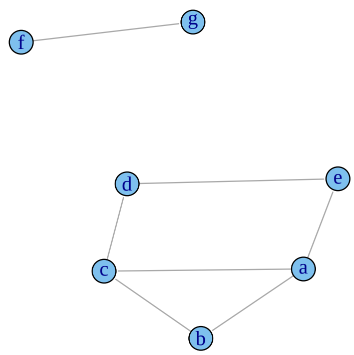
```
 as(ug11,"igraph")

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

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

  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

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

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

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

4 Plotting graphs

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

```
par(mfrow=c(1,2))
plot(ug11)
plot(ug11i)
```





Graphs represented as adjacency matrices: There is no plot method for graphs represented as adjacency matrices in **gRbase** but the `gplot()` function in the **sna** package has these facilities:

```
par(mfrow=c(1,2))
library(sna)
gplot(ug11m, label=colnames(ug11m),gmode="graph")
gplot(dag11m, label=colnames(dag11m))
```

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[1] 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
```

[1]Actually not quite yet, but it will be so in the future.

6 Advanced graph operations

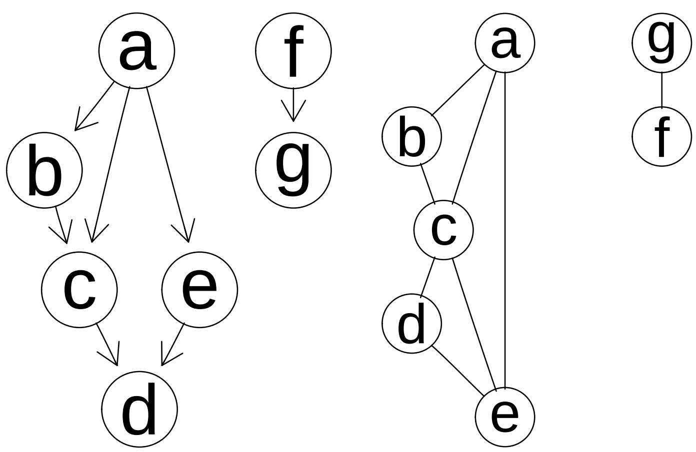
6.1 Moralization

```
apropos("^moralize\\.")

[1] "moralize.Matrix"   "moralize.graphNEL" "moralize.igraph"
[4] "moralize.matrix"
```

A moralized directed acyclic graph is obtained with

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

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



For the alternative representations

```
moralize(dag11m)

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

moralize(dag11M)

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

moralize(dag11i)

IGRAPH UN-- 7 8 --
+ attr: name (v/c), label (v/c)
```

6.2 Maximum cardinality search

```
apropos("^mcs\\.")

[1] "mcs.Matrix"   "mcs.graphNEL" "mcs.igraph"   "mcs.matrix"
```

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)

mcs(ug11m)

character(0)

mcs(ug11M)

character(0)

mcs(ug11i)

character(0)
```

```
mcs(dag11.mor)

[1] "a" "b" "c" "e" "d" "g" "f"

mcs(as(dag11.mor,"matrix"))

[1] "a" "b" "c" "e" "d" "g" "f"

mcs(as(dag11.mor,"Matrix"))

[1] "a" "b" "c" "e" "d" "g" "f"

mcs(as(dag11.mor,"igraph"))

[1] "a" "b" "c" "e" "d" "g" "f"

mcs(dag11)

character(0)
```

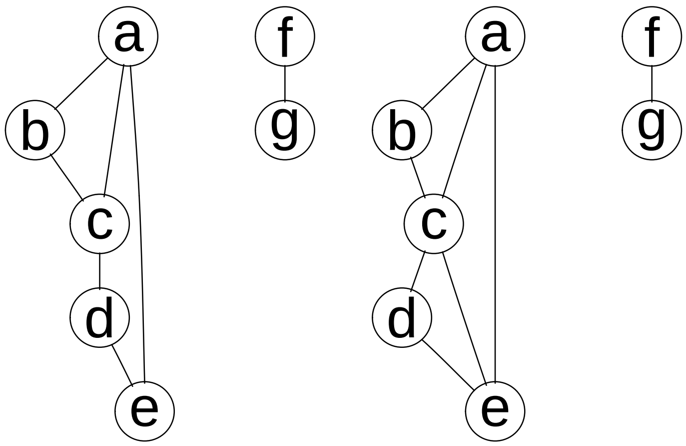
6.3 Triangulation

```
apropos("^triangulate\\.")

[1] "triangulate.Matrix"   "triangulate.graphNEL" "triangulate.igraph"
[4] "triangulate.matrix"
```

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

 (tug11m<-triangulate(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 1 0 0
d 0 0 1 0 1 0 0
e 1 0 1 1 0 0 0
f 0 0 0 0 0 0 1
g 0 0 0 0 0 1 0

 (tug11M<-triangulate(ug11M))

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 1 . .
d . . 1 . 1 . .
e 1 . 1 1 . . .
f . . . . . . 1
g . . . . . 1 .

 (tug11i<-triangulate(ug11i))

IGRAPH UN-- 7 8 --
+ attr: name (v/c), label (v/c)
```

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



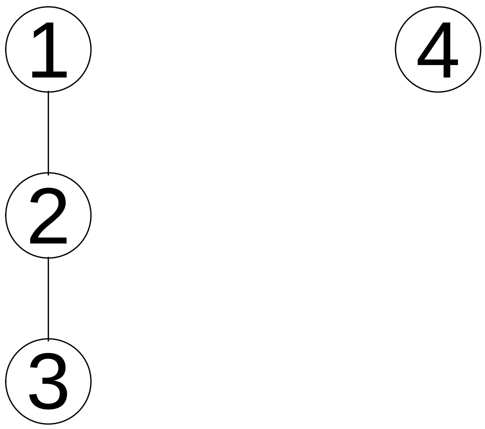
6.4 RIP ordering / junction tree

```
 apropos("^rip\\.")

[1] "rip.Matrix"   "rip.graphNEL" "rip.igraph"   "rip.matrix"
```

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

```
 rr <- rip(tug11)
 rr

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

 rr <- rip(tug11m)
 rr <- rip(tug11M)
```

```
plot(rr)
```



7 Time and space considerations

7.1 Time

It is worth noticing that working with graphs representated as `graphNEL` objects is somewhat slower working with graphs represented as adjacency matrices. 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)})     ## in RBGL

  user  system elapsed
  0.12    0.00    0.12

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

  user  system elapsed
  0.01    0.00    0.02
```

Working with sparse matrices rather than standard matrices slows indexing down:

```
system.time({for (ii in 1:2000) ug11m[2,]})

  user  system elapsed 
     0       0       0 

system.time({for (ii in 1:2000) ug11M[2,]})

  user  system elapsed 
  0.72    0.00    0.72 
```

However, **gRbase** has some functionality for indexing sparse matrices quickly:

```
system.time({for (ii in 1:2000) sp_getXj(ug11M,2)})

  user  system elapsed 
  0.03    0.00    0.03 
```

7.2 Space

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". The sparse matrix representation is clearly only superior to the standard matrix representation if the graph is sparse:

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

[1] 118

 object.size(g1)

135608 bytes

 object.size(as.adjMAT(g1))

51648 bytes

 object.size(as.adjMAT(g1, "Matrix"))

15936 bytes

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

[1] 4547

 object.size(g1)

3465768 bytes

 object.size(as.adjMAT(g1))

51648 bytes

 object.size(as.adjMAT(g1, "Matrix"))

122232 bytes
```