# How To use graph

November 25, 2003

## Introduction

The *graph* package constitutes a preliminary approach to an implementation of graphs (nodes and edges) in R. The approach is quite simplistic and will hopefully evolve as we learn what things need to be done and what is or is not efficient.

The reader is likely to find it very helpful to have the *Rgraphviz* package also attached. It will be important to be able to visualize some of the many different graphs that can be drawn.

## Classes

The basic class, *graph*, is a virtual class and all other classes will extend this class. There are three main implementations available. Which is best will depend on the particular data set and what the user wants to do with it.

The class *graphNEL* is a node and edge-list representation of a graph. That is the graph is comprised of two components a list of nodes and a list of the out edges for each node.

The class *graphAM* is an adjacency matrix implementation. It will be developed next and will use the *SparseM* package if it is available.

The class *clusterGraph* is a special form of graph for clustering. In this graph each cluster is a completely connected component (a clique) and there are no between cluster edges.

## Some Examples

From Chris Volinsky.

```
> V <- LETTERS[1:4]
> edL1 <- vector("list", length = 4)
> names(edL1) <- V
> for (i in 1:4) edL1[[i]] <- list(edges = c(2, 1, 4, 3)[i], weights = sqrt(i))
> gR <- new("graphNEL", nodes = V, edgeL = edL1)
> edL2 <- vector("list", length = 4)
> names(edL2) <- V
> for (i in 1:4) edL2[[i]] <- list(edges = c(2, 1, 2, 1)[i], weights = sqrt(i))
> gR2 <- new("graphNEL", nodes = V, edgeL = edL2)
> edgemode(gR2) <- "directed"
> set.seed(123)
> gR3 <- randomGraph(LETTERS[1:4], M <- 1:2, p = 0.5)
> x1 <- intersection(gR, gR3)
> x1
```

```
A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 2
```

```
> x2 <- union(gR, gR3)
> x2
```

```
A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 4
```

```
> x3 <- complement(gR)
> x3
```

```
A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 4
```

Notice that while the graphs `gR` and `gR2` have different sets of edge weights these are lost when the `union`, `intersection` and `complement` are taken. It is not clear how they should be treated and in the current implementation they are ignored and replaced by weight 1 in the output.

## Random Graphs

Two basic strategies for finding random graphs have been included. One is the random edge model. In this graph the nodes are considered fixed and the edges are sampled, uniformly at random, from the set of possible edges in the complete graph.

The function `randomEGraph` will generate graphs using the random edge model.

```
> set.seed(333)
> V = letters[1:12]
> g1 = randomEGraph(V, 0.1)
> g1
```

```
A graph with  undirected  edges
Number of Nodes = 12
Number of Edges = 8
```

A different method for generating random graphs has a sort of latent variable approach. In this model we presume that there are some latent variables that are appropriate for all nodes. Then two nodes will have an edge between them if the underlying latent variables are the same.

```
> set.seed(23)
> V <- LETTERS[1:20]
> M <- 1:4
> g1 <- randomGraph(V, M, 0.2)
```

We can find out about `g1` by typing its name or by applying various functions to it.

```
> g1
```

```
A graph with  undirected  edges
Number of Nodes = 20
Number of Edges = 58
```

```
> g1cc <- connComp(g1)
> g1cc

[[1]]
[1] "A"

[[2]]
 [1] "B" "C" "D" "E" "F" "G" "I" "J" "M" "N" "O" "Q" "R" "S" "T"

[[3]]
[1] "H"

[[4]]
[1] "K"

[[5]]
[1] "L"

[[6]]
[1] "P"

> g1.sub <- subGraph(g1cc[[2]], g1)
> g1.sub

A graph with  undirected  edges
Number of Nodes = 15
Number of Edges = 58
```

The function `connComp` returns the connected components of a graph. There are many different times when different operations might be applied to the connected components separately.

# Direct Manipulation of Nodes and Edges

A number of functions have been added that allow the user to directly manipulate nodes and edges in graphs. Note that all of these functions make copies of the graph and manipulate the copies. The original graph should not be affected. Clearly this will not be the best approach for large graphs – then we might need to do something different.

Some examples. You will probably get more out of these examples if you use *Rgraphviz* to view the graphs.

```
> V <- LETTERS[1:4]
> edL1 <- vector("list", length = 4)
> names(edL1) <- V
> for (i in 1:4) edL1[[i]] <- list(edges = c(2, 1, 4, 3)[i], weights = sqrt(i))
> gR <- new("graphNEL", nodes = V, edgeL = edL1)
> edL2 <- vector("list", length = 4)
> names(edL2) <- V
> for (i in 1:4) edL2[[i]] <- list(edges = c(2, 1, 2, 1)[i], weights = sqrt(i))
> gR2 <- new("graphNEL", nodes = V, edgeL = edL2)
> edL3 <- vector("list", length = 4)
> for (i in 1:4) edL3[[i]] <- list(edges = (i%%4) + 1, weights = i)
> names(edL3) <- V
> gR3 <- new("graphNEL", nodes = V, edgeL = edL3, "directed")
> x1 <- intersection(gR, gR2)
> x1

A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 1

> x2 <- union(gR, gR2)
> x2

A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 3

> x3 <- complement(gR)
> x3

A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 4

> v1 <- clearNode("A", gR)
> v1

A graph with  undirected  edges
Number of Nodes = 4
Number of Edges = 1

> v2 <- removeNode("B", gR)
> v2

A graph with  undirected  edges
Number of Nodes = 3
Number of Edges = 1

> v3 <- addNode("M", gR)
> v3

A graph with  undirected  edges
Number of Nodes = 5
Number of Edges = 2

> v4 <- addEdge("M", "A", v3, 1)
> v5 <- addEdge("A", c("C", "D"), v4, 1)
> inEdges(c("M", "B"), v5)

$M
[1] "A"

$B
[1] "A"

> c1 <- combineNodes(c("A", "M"), v5, "S")
> c2 <- combineNodes(c("A", "C"), v5, "S")
> inEdges(c("C", "B"), gR3)
```

```
$C
[1] "B"

$B
[1] "A"

> g4 <- addNode("X", gR3)
> g5 <- addEdge("X", "C", g4, 1)
> g6 <- combineNodes(c("B", "D"), g5, "E")
```