Graphs with Bioconductor

Wolfgang Huber

August 16, 2005

Loading all the packages that are needed for this exercise

```
> library("graph")
> library("Rgraphviz")
> library("RBGL")
> library("GO")
> library("GOstats")
> library("ALL")
> library("hgu95av2")
> library("genefilter")
```

Creating our first graph from scratch

```
> myNodes <- c("s", "p", "q", "r")
> myEdges <- list(s = list(edges = c("p", "q")), p = list(edges = c("p",
+     "q")), q = list(edges = c("p", "r")), r = list(edges = c("s")))
> g <- new("graphNEL", nodes = myNodes, edgeL = myEdges, edgemode = "directed")
> g
> plot(g)
```

Querying nodes and edges

```
> nodes(g)
> edges(g)
> degree(g)
> plot(g)
```

Graph manipulation

```
> g1 = addNode("t", g)
> g2 = removeNode("p", g)
> g3 = addEdge("t", "s", g1, weights = pi/2)
> plot(g3)
```

Loading a slightly more complex example graph

```
> data("integrinMediatedCellAdhesion")
> IMCAGraph
> nodes(IMCAGraph)
> edges(IMCAGraph)
```

Exploring it

```
> class(IMCAGraph)
> adj(IMCAGraph, "SOS")
> s = acc(IMCAGraph, "SOS")
> s
```

To demonstrate set operations on graphs, we generate two graphs ug1 and ug2 using the *randomGraph* function.

```
> V <- letters[1:4]
> set.seed(4713)
> ug1 <- randomGraph(V, 1, 0.55)
> ug2 <- randomGraph(V, 1, 0.55)
> plot(ug1)
> plot(ug2)
> plot(complement(ug1))
> plot(intersection(ug1, ug2))
> plot(union(ug1, ug2))
```

Switching between graph representations: from-to table, adjacency matrix, graphNEL

```
> ft = cbind(1:4, c(2, 3, 1, 4))
> ft
> m = ftM2adjM(ft)
> g = ftM2graphNEL(ft)
> plot(g)
```

GXL

```
> edit(file = system.file("GXL", "kmstEx.gxl", package = "graph"))
```

A large random edge graph (the layout will take a while to compute)

```
> re = randomEGraph(paste(1:100), edges = 50)
> plot(re, "neato")
```

Connected components

```
> cc = connComp(re)
> cc[1:5]
> table(listLen(cc))
> wh = which.max(listLen(cc))
> sg = subGraph(cc[[wh]], re)
> plot(sg, "neato")
```

The induced GO graph of GO category 0003700

```
> get("GO:0003700", GOTERM)
> tfg = GOGraph("GO:0003700", GOMFPARENTS)
```

Plotting it

```
> plot(tfg)
> get("GO:0003677", GOTERM)
```

Fine control of node and edge rendering

In the following exercise we will learn how to control the node rendering, e.g. their colors and shapes. First, we create a data structure that holds these attributes.

```
> nda = makeNodeAttrs(re)
> nda
```

Then we create a vector of color names and assign different colors to each connected component:

```
> colors = rainbow(length(cc))
> ord = order(listLen(cc))
> for (i in seq(along = ord)) nda$fillcolor[cc[[ord[i]]]] = colors[i]
```

To fine control graph layouts, the Rgraphviz package allows to separate the tasks of *layout* and plotting

```
> la = agopen(re, nodeAttrs = nda, layoutType = "neato", name = "whatever")
> la
> plot(la)
```

... and so on:

```
> nda$shape[cc[[4]]] = "box"
> la2 = agopen(re, nodeAttrs = nda, layoutType = "neato", name = "whatever")
> plot(la2)
```

Even finer control of the graph rendering is possible by accessing the AgNode and Edge objects in the `la` object

```
> class(la)
> slotNames(la)
> AgEdge(la)
> slotNames(AgEdge(la)[[1]])
> AgEdge(la)[[1]]@color
```

A GOstats analysis

First we load the ALL data set, and select only those samples with a B-cell tumor (s1) and one of the two genotypes "BCR/ABL" or "NEG".

```
> data(ALL)
> s1 = grep("^B", as.character(ALL$BT))
> s2 = which(as.character(ALL$mol) %in% c("BCR/ABL", "NEG"))
> eset = ALL[, intersect(s1, s2)]
```

Then we select those genes whose IQR is among the top 50% of all IQRs:

```
> n = nrow(pData(eset))
> q25 = rowQ(exprs(eset), n * 0.25)
> q75 = rowQ(exprs(eset), n * 0.75)
> iqr = q75 - q25
> sel = (iqr >= median(iqr))
> eset = eset[sel, ]
```

Now can calculate for each gene the t-tstatistic for the comparison between NEG and BCR/ABL, and seelect the top 100 genes.

```
> tt = rowttests(eset, "mol")
> hist(tt$p.value, 200, col = "orange")
> diffps = geneNames(eset)[order(tt$p.value)[1:100]]
> diffps
```

From the probe set IDs, we obtain the unique Locuslink IDs

```
> diffll = unlist(mget(diffps, hgu95av2LOCUSID))
> diffll
> diffll = as.character(unique(diffll))
```

and apply hypergeometric tests to find GO categories that are enriched in our gene list

```
> gGhyp = GOHyperG(diffll)
> hist(gGhyp$pv)
> goCats = names(which(gGhyp$pv < 0.01))
> mget(goCats, GOTERM)
```

We can also construct and plot the induced GO graph

```
> gGO = makeGOGraph(diffll, "MF")
> gGO
> natt = makeNodeAttrs(gGO, shape = "ellipse", label = substr(nodes(gGO),
+     7, 10))
> natt$fillcolor[goCats] = "red"
> lgGO = agopen(gGO, recipEdges = "distinct", layoutType = "dot",
+     nodeAttrs = natt, name = "")
> plot(lgGO)
```