Graph Visualization for Gene Set Testing

Utah State University – Spring 2014

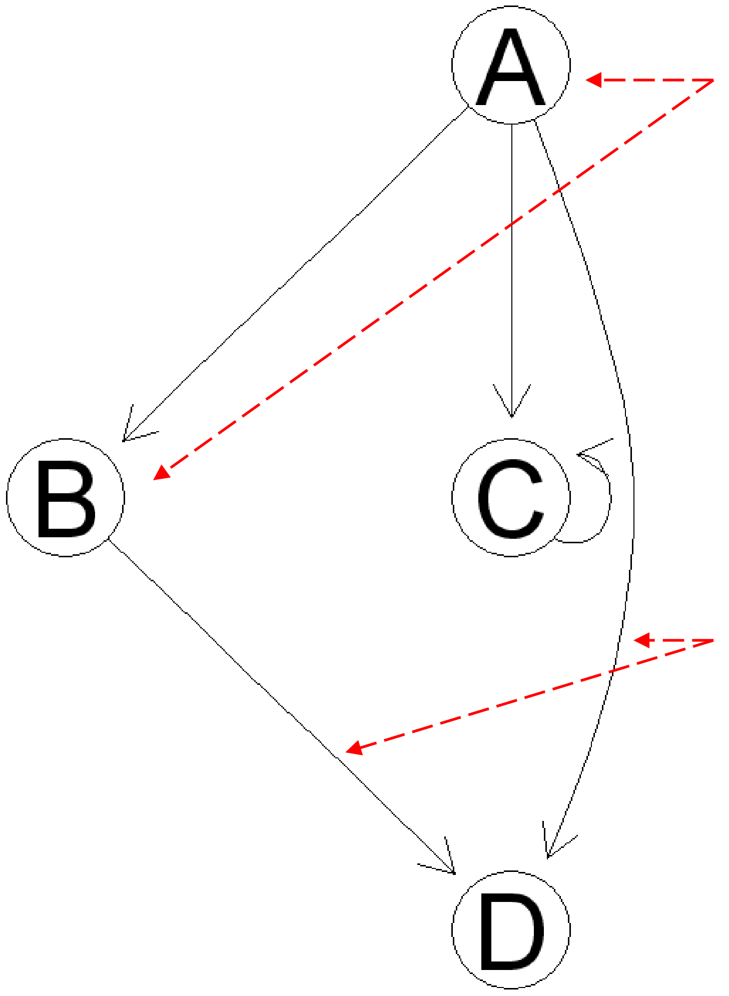
STAT 5570: Statistical Bioinformatics

Notes 4.4

References

- Chapters 19-22 of Bioconductor Monograph (course text)

Graphs – from discrete mathematics



nodes – “entities of interest”
(maybe genes, proteins, or GO terms)

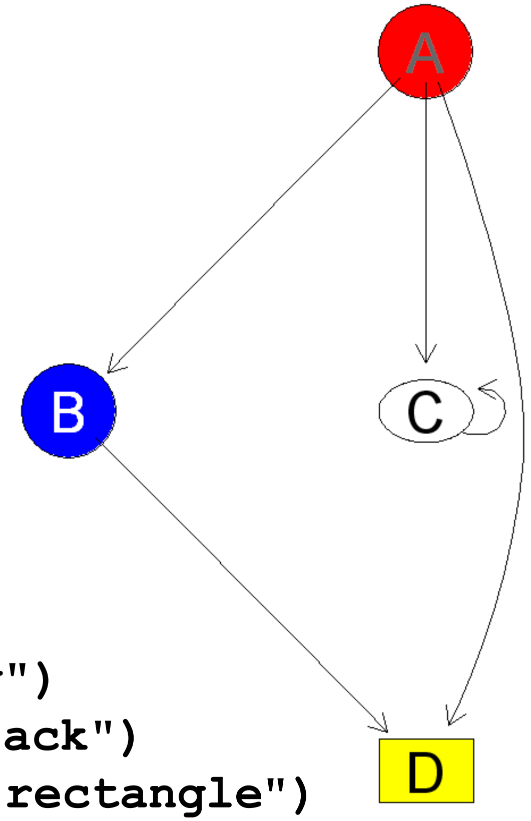
edges – “relationships between entities”
(maybe interactions between genes, or [our focus here] sub-setting of GO terms)

Why graphs?

- Knowledge representation [our focus here]
  - visualization of pathways, for example
- Exploratory Data Analysis (EDA)
  - guide discovery of interesting phenomena
  - mapping expression values onto graphs, for example
- Statistical Inference
  - compare observed graph to a “random” graph

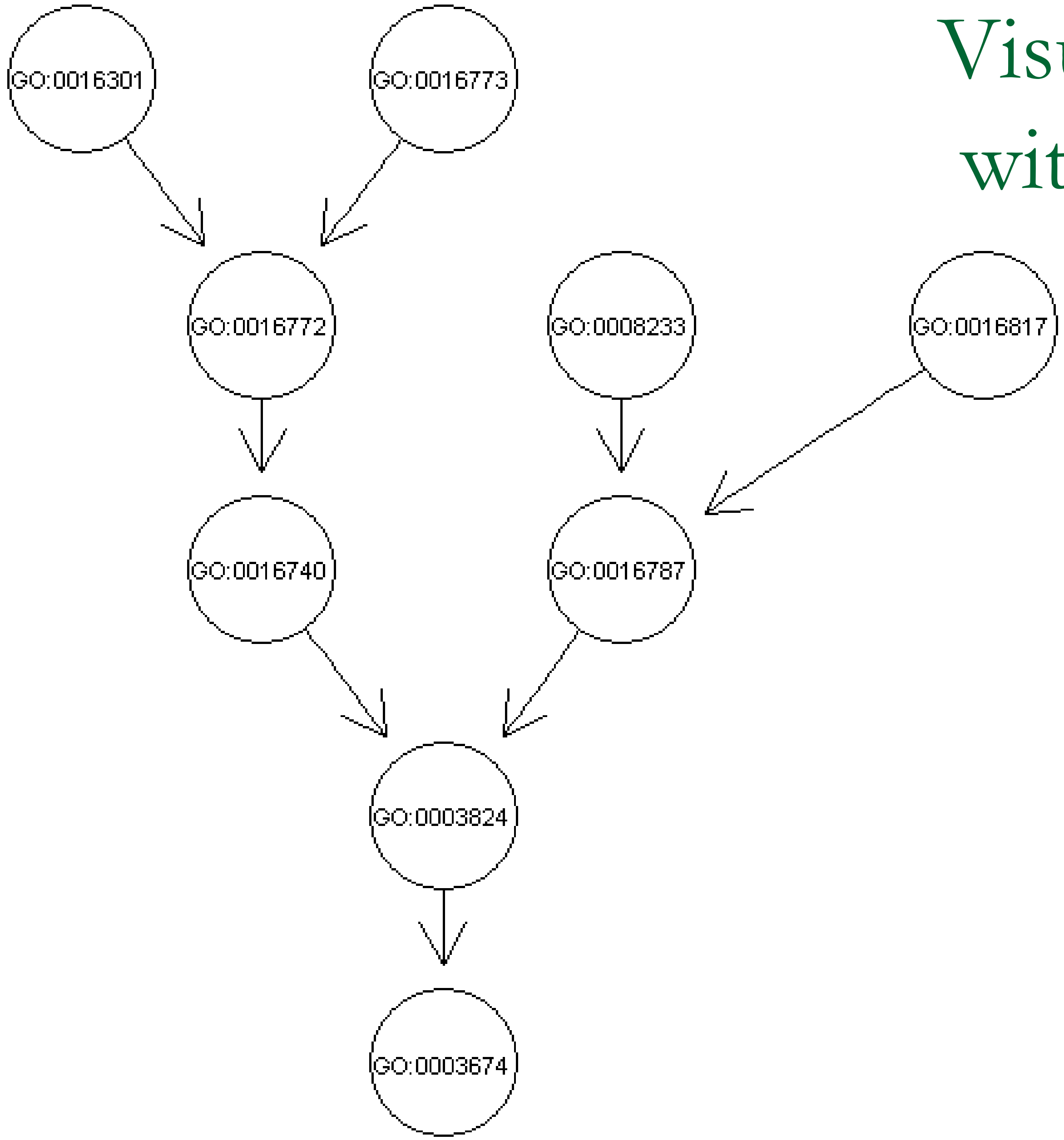
```
# R code to generate the simple graph on slide 3
library(graph); library(Rgraphviz)

# List nodes
MyNodes <- c(  "A",
               "B",
               "C",
               "D"   )

# For each node above (and in the same order), list nodes
# for which the node is a "child" (i.e., the nodes to which
# the node will have an edge pointing), with an empty list
# if it is not a "child" for anything.
MyEdges <- list(  A = list(edges=c("C","B","D")),
                  B = list(edges=c("D")),
                  C = list(edges=c("C")),
                  D = list()    )

g <- new("graphNEL",nodes=MyNodes,
         edgeL=MyEdges,edgemode="directed")
plot(g)
```

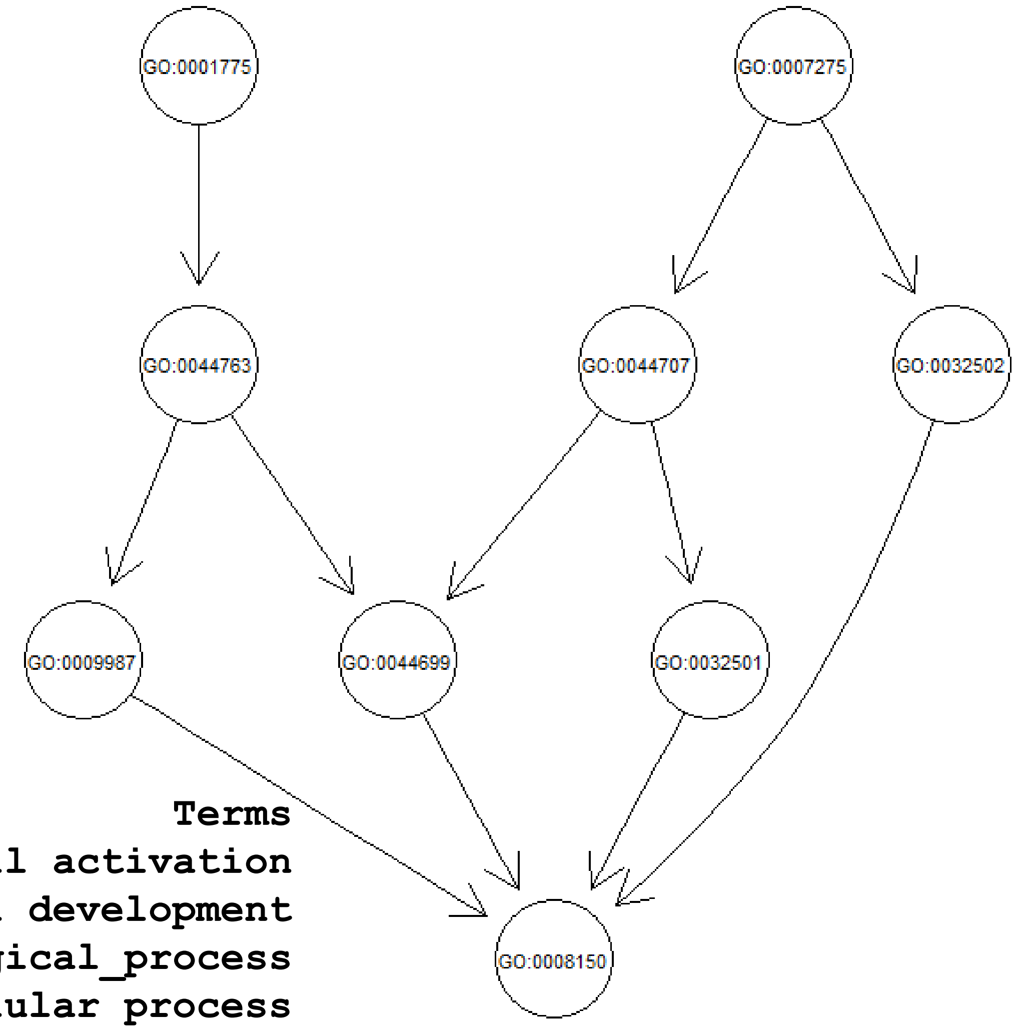
(Don’t worry about code here; just note that you can change the appearance of graph to emphasize various characteristics.)



```
## Revise the graph a little
# create a 'copy' graph to modify
nAgo <- makeNodeAttrs(g)
ag.obj <-  agopen(g, recipEdges="distinct",
  layoutType="dot", nodeAttrs=nAgo, name="")
# define changes - in same order as nodes
fill.colors <- c("red","blue","white","yellow")
text.colors <- c("grey40","white","black","black")
node.shape <- c("circle","circle","ellipse","rectangle")
# make the changes to nodes
for(i in 1:length(MyNodes))
 {  ag.obj@AgNode[[i]]@fillcolor <- fill.colors[i]
    ag.obj@AgNode[[i]]@txtLabel@labelColor <- text.colors[i]
    ag.obj@AgNode[[i]]@shape <- node.shape[i] }
# plot the revised object
plot(ag.obj)
```

Visualizing GO with Graphs

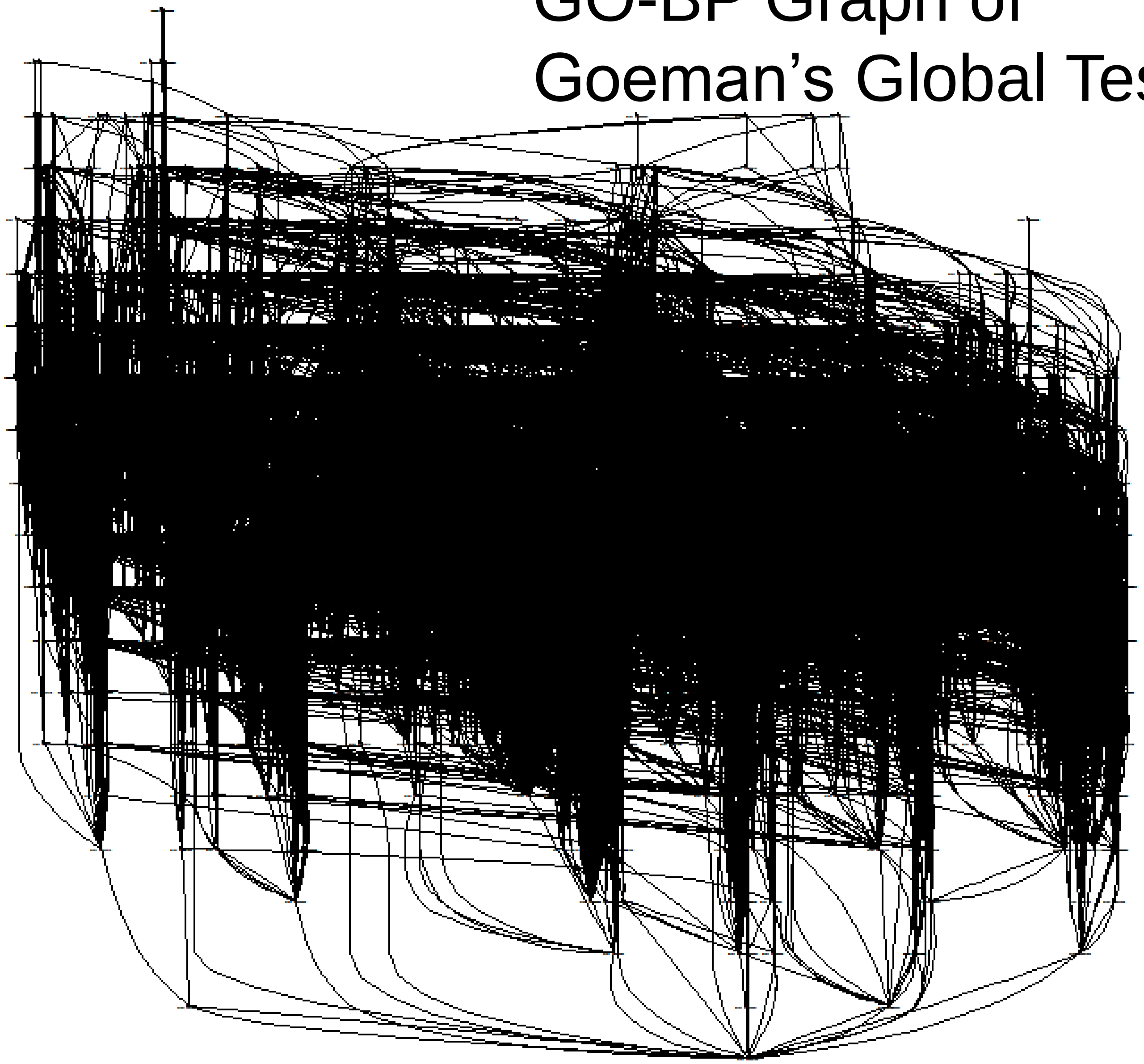




```
                                                 Terms
GO:0001775                             cell activation
GO:0007275  multicellular organismal development
GO:0008150                          biological_process
GO:0009987                            cellular process
GO:0032501          multicellular organismal process
GO:0032502                       developmental process
GO:0044699                     single-organism process
GO:0044707 single-multicellular organism process
GO:0044763          single-organism cellular process
```

```
## A simple example of GO graph
library(GOstats); library(GO.db)
library(Rgraphviz); library(annotate)

# list all child nodes, get parents within BP ontology,
# and make graph
GO.vec <- c("GO:0001775","GO:0009987","GO:0007275")
g <- GOGraph(GO.vec, GOBPPARENTS)
g <- removeNode("all",g)
plot(g)

# Get legend of GO Terms, knowing that these are
# from BP ontology
GO.vec <- sort(names(nodes(g)))
Terms <- getGOTerm(GO.vec)$BP
legend <- data.frame(Terms)
legend
```

GO-BP Graph of Goeman's Global Test



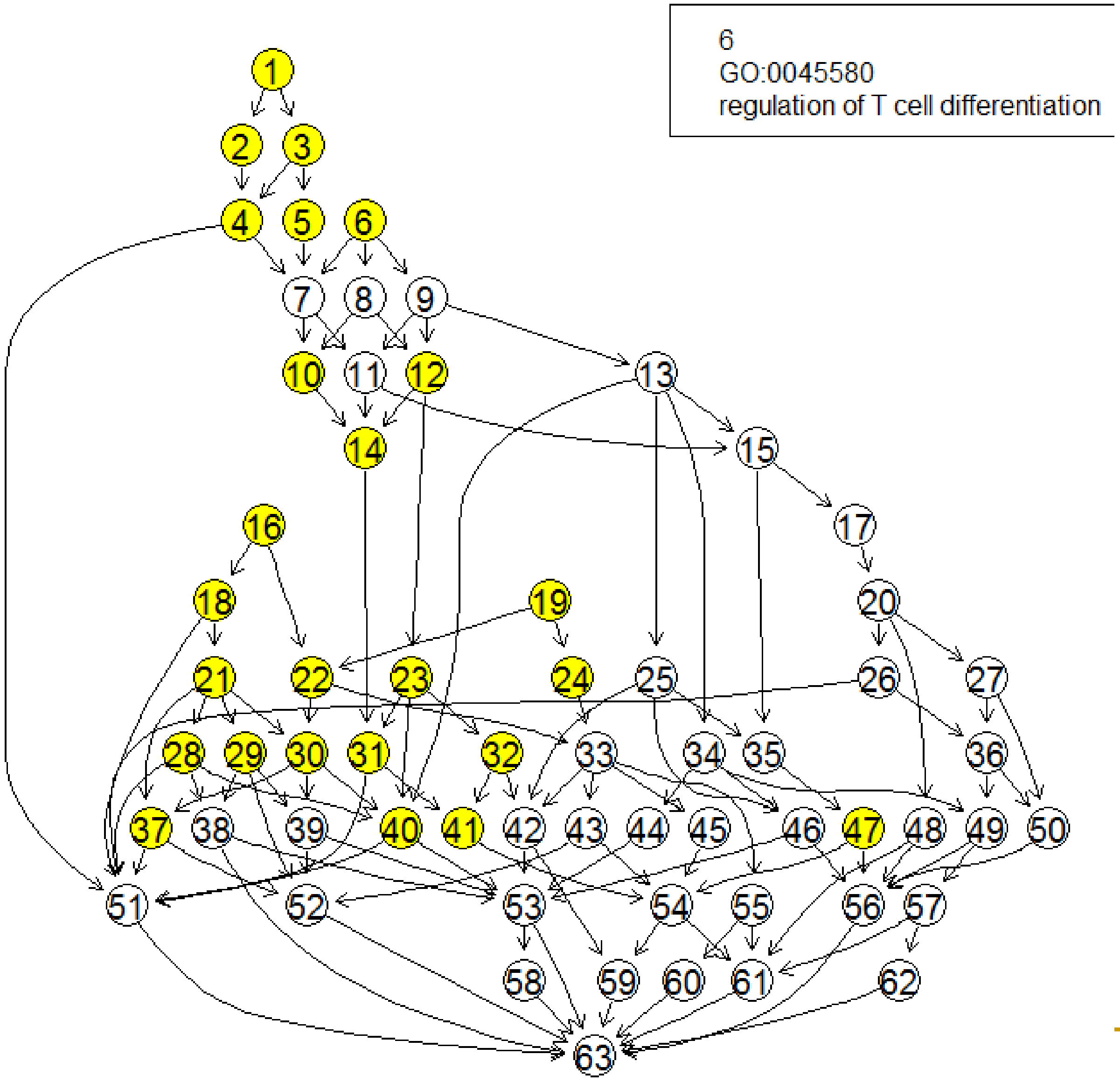
These are all of the BP GO terms tested by the Global Test for the ALL data.

There are 11,390 nodes in the graph.

```
# Run globaltest on all BP terms in ALL data
# -- similar to slides 21-22 of Notes 4.3

library(affy); library(ALL); data(ALL)
eset <- exprs(ALL)
T.cell <- c(rep(0,95),rep(1,33))

Eset <- new("ExpressionSet", exprs=eset)
pData(Eset) <- data.frame(trt=as.character(T.cell))
annotation(Eset) <- "hgu95av2"

library(globaltest)
print(date())
gt.GO <- gtGO(trt, Eset, ontology='BP')
print(date())  # This took about  4 minutes

result <- data.frame(GO.ID=names(gt.GO),
     alias=gt.GO@extra[,2],
     pvalue=gt.GO@result[,1]) # dim is 11390 by 3
```

```
# Now make the full graph of BP terms:

g <- GOGraph(as.character(result$GO.ID), GOBPPARENTS)

# Plot the graph
library(Rgraphviz)
g1 <- removeNode("all",g)
plot(g1)
```

GO-MF Sub-Graph of Goeman's Global Test



Show only the most significant nodes (colored yellow here) and their parents

```
                                                       Terms
1                              positive thymic T cell selection
2                                     positive T cell selection
3                                       thymic T cell selection
4                                              T cell selection
5                             T cell differentiation in thymus
6                         regulation of T cell differentiation
7                                        T cell differentiation
8                              regulation of T cell activation
9                     regulation of lymphocyte differentiation
10                                            T cell activation
11                                  lymphocyte differentiation
12                          regulation of lymphocyte activation
...
58                                        biological regulation
59                                             cellular process
60                                                    signaling
61                                     single-organism process
62                               multicellular organismal process
63                                           biological_process
```

```
# Define function to make interactive graph; don't worry about syntax;
# see use on next slide
interactive.graph <- function(GO.Graph, color.nodes, interact=FALSE,
legend.pos="bottomleft", print.legend=FALSE)
{
nodes <- buildNodeList(GO.Graph)
focusnode <- sapply(nodes, name) %in% color.nodes
names(focusnode) <- names(nodes)
nodefill <- ifelse(focusnode, "yellow", "white")
nAttrs <- list(); nAttrs$fillcolor <- nodefill
nAttrs$label <- 1:length(names(nodes)); names(nAttrs$label) <- names(nodes)
pg <- plot(GO.Graph, nodeAttrs = nAttrs)
x <- getNodeXY(pg)$x; y <- getNodeXY(pg)$y
ordering <- sort.list(order(-y, x)); nAttrs$label <- ordering
names(nAttrs$label) <- names(nodes); plot(GO.Graph, nodeAttrs = nAttrs)
Terms <- sapply(lookUp(names(nodes)[sort.list(ordering)], "GO",
"TERM"), Term); names(Terms) <- NULL; legend <- data.frame(Terms)
if(print.legend){print(legend)}
if(interact)
{ repeat {
p <- locator(n = 1); if (is.null(p)) break()
pg <- plot(GO.Graph, nodeAttrs = nAttrs)
x <- getNodeXY(pg)$x; y <- getNodeXY(pg)$y
distance <- abs(p$x - x) + abs(p$y - y); idx <- which.min(distance)
legend(legend.pos, legend=c(nAttrs$label[idx], names(focusnode)[idx],
Term(lookUp(names(focusnode)[idx], "GO", "TERM")[[1]])), bg = "white")
} } }
```

```
# get top GO terms (result object from slide 11)
top.GO <- as.character(result$GO.ID[1:25])

# make graph object (within BP ontology)
g.sub <- GOGraph(top.GO, GOBPPARENTS)
g.sub <- removeNode("all",g.sub)

# draw interactive graph (ESC to top interactive)
interactive.graph(g.sub,top.GO, interact=TRUE,
   legend.pos="topright", print.legend=TRUE)
```

Variations on Graphs and Visualization (not covered here, maybe later in this class discussion)

- Can test for significance controlling for graph structure
- Can search for most informative (least redundant) sub-graph
- Can color nodes by P-value or LFC, including in KEGG pathways (KEGGgraph package, a good 6000-level project)
- Can use meta-analysis approaches to combine genes' P-values within each node

Summary and possible directions

- Goeman’s global test
  - test sets of genes for DE
  - not the same as “over-representation”
  - valid: subject sampling, self-contained null
- Graphs
  - visualize GO nesting
  - good for highlighting “significant” terms
  - also KEGG pathways (KEGGgraph package)