MODULE 5: NETWORKS AND GRAPHICAL MODELS

CASE STUDY ACTIVITY TUTORIAL

CASE STUDY 1– DISCOVERING GENES THAT CAUSE AUTISM USING NETWORK THEORY

CASE STUDY ACTIVITY TUTORIAL

CASE STUDY 1– DISCOVERING GENES THAT CAUSE AUTISM USING NETWORK THEORY

Faculty: Caroline Uhler

In this document, we walk through some helpful tips to get you started with building your own network theory application to determine important nodes in a network given knowledge about other nodes. We show how this works for the problem of identifying new candidate genes that might cause Autism. In this tutorial, we provide examples and some pseudo-code for the following programming environment: **R**. We cover the following topics:

Topics

Data: Protein-Protein Interaction Network

For this case study, we will work with the BioGRID (http://thebiogrid.org) dataset, which makes available the protein-protein interaction network for humans. For your convenience, we provide a sample of this data in the file entitled: 'BIOGRID.txt" for use in the rest of this tutorial.

Note that for this tutorial, you will need the igraph library for R. You can install it by invoking the following line of code and choosing the mirror closest to you:

```
install.packages("igraph")
```

The following lines of code in R will allow you to work with this protein-protein interaction data and represent the network in R.

```
library(igraph)
biogrid <- read.delim("./BIOGRID.txt",stringsAsFactors = F)

#View biogrid

names(biogrid)
attach(biogrid)
```

```
HSnet <-
graph.data.frame(data.frame(Entrez.Gene.Interactor.A,Entrez.Gene.Interactor.B),directed=F)

# You can uncomment the following line to plot the network, but it takes a long time

#plot(HSnet)


A <- get.adjacency(HSnet)

# multiple edges
A[1:15,1:15]

# the following is FALSE if the graph is not simple
is.simple(HSnet)

# remove multiple edges and self-loops

HSnet <- simplify(HSnet, remove.multiple = TRUE, remove.loops = TRUE,
        edge.attr.comb = getIgraphOpt("edge.attr.comb"))
is.simple(HSnet)
A <- get.adjacency(HSnet)

# only single edges now
A[1:15,1:15]

# for this application, we remove nodes of very high degree; these are usually house-keeping

genes that are necessary to keep a cell alive, but are usually not specific to a particular
disease.
    overly.attached.proteins <- which(degree(HSnet)>1000)
    HSnet <- delete.vertices(HSnet, overly.attached.proteins )

# the following is TRUE if the graph is connected.
    is.connected(HSnet)
```

Data: Genes Causing Autism

We now need to identify a list of genes that are known to cause Autism. Such data is provided by SFARI Gene (https://gene.sfari.org/autdb/Welcome.do). We provide sample files from this database entitled: “gene-id-table.txt” and “gene-score.csv”. You can use these files in the rest of this tutorial.

The following lines of code in R will guide you in extracting information about the genes that are known to cause Autism:

```
# read the gene-id table
gene.table <- read.delim("gene-id-table.txt")
names(gene.table)
```

```
# read the scores for Autism
gene.score<-read.csv("gene-score-dataset/gene-score.csv",stringsAsFactors=F)
attach(gene.score)
names(gene.score)

# display the scores
unique(Score)

# identify the genes that have significant scores
signif.scores<-c("3","1S","1","2S","2","3S")
signif.genes<-Gene.Symbol[which(Score %in% signif.scores)]
signif.EIDs <- gene.table[which(gene.table[,1] %in% signif.genes),2]

# Now use the protein interaction network HSnet, created previously, to determine the genes
that are present in the network and known to cause Autism
geneEIDs <- as.numeric(V(HSnet)$name)
HSnetN<-HSnet
V(HSnetN)$name<-1:length(V(HSnet))

signif.ids<-which(geneEIDs %in% signif.EIDs)
length(signif.ids)
```

Building an Autism Interactome

Now that we have a protein-protein interaction network and have also identified the genes or nodes in this network that are known to cause Autism, we can build an "Autism Interactome": We want to determine the smallest subnetwork of the full protein-protein interaction network that contains the Autism genes. This will help us to determine new candidate genes for Autism, since these are genes that are central in this Autism Interactome. We build the Autism Interactome using Steiner Trees, i.e. the subnetwork of shortest length that connects a given set of nodes. We have provided a script in R called "steiner_tree.R" for your convenience. This script should be placed in the same directory as your own R script. You can then use it, as shown below.

The following lines of code will identify and visually represent the Steiner Tree. The genes that have with significant scores for causing Autism are shown in red, orange and yellow, depending on their significance values. The output graphic is saved in the same folder as a pdf file entitled "**ASD_interactome.pdf**":

```
source('steiner_tree.R')

# Identify the Steiner Tree and note the time this function call takes
system.time(HS.stree <- steiner_tree(terminals=signif.ids, graph=HSnetN))

# Output the overlap between significant Autism and vertices in the Steiner Tree
length(intersect(signif.ids,V(HS.stree)$name))
labels<-gene.table[as.numeric(V(HS.stree)$name),1]
labels<-as.character(labels)
```

```
# identify the genes that have significant scores, and assign the color “red” to them
colors<-rep("skyblue",length(V(HS.stree)))
colors[which(as.numeric(V(HS.stree)$name) %in% signif.ids)] = "red"

# assign colors to the vertices of the tree
V(HS.stree)$color = colors

# plot and save to file
pdf("ASD_interactome.pdf",width=12, height=12)
system.time(plot(HS.stree,vertex.label=labels,vertex.size=5,vertex.label.cex=0.8))
dev.off()
```

Analysis and Properties of the Autism Interactome

We can now analyze the properties of the Autism Interactome and compare it to randomly generated Interactomes. The following lines of code in R will help us with these comparisons. Note that you will need the “**sna**” library for R. You can download and install it by calling the following function*:* ***install.packages(“sna”)*** and then choosing the nearest mirror to your location.

```
library(sna, quietly=TRUE)

# a function that computes the connectivity scores for a network
# here the scores are diameters of the network and average geodesic distance between any two nodes
c.scores<-function(graph) {
        n<-length(V(graph))
        sp<-shortest.paths(graph)
        neighbors<-sum(sp==1)/2
        neighbors2<-sum(sp==2)/2
        return(c(2*neighbors/(n*(n-1)),2*neighbors2/(n*(n-1))))
}
clus<-clusters(HSnetN, mode=c("weak"))
connected.ids<-which(clus$membership==1)
length(connected.ids)

# Generate N randomly chosen subnetworks. Note: this will take a while if N is set large.
N<-50
strees<-list(N)
effs<-numeric(N)
nei<-numeric(N)
nei2<-numeric(N)
for (i in 1:N){
        new.ids<-sample(x=connected.ids,size=length(signif.ids))
        strees[[i]] <- steiner_tree(terminals=new.ids, graph=HSnetN)
        effs[i]<-efficiency(get.adjacency(strees[[i]],sparse=F))
        cs<-c.scores(strees[[i]])
        nei[i]<-cs[1]
```

```
        nei2[i]<-cs[2]
}

# print the efficiencies and connectivity scores for each of the N random graphs
effs
nei
nei2

# Finally, print the efficiency score and connectivity scores for the Autism Interactome
efficiency(get.adjacency(HS.stree,sparse=F))
c.scores(HS.stree)
```

Identifying New Candidate Genes

Next, we would like to use the Autism Interactome to identify new candidate genes that could cause Autism. We do this using some of the centrality measures discussed in this module. In particular, nodes with high betweenness centrality are good candidates for Autism, since these are genes that lie on the largest number of shortest paths between any pair of genes in the Autism Interactome, i.e. these are nodes through which a lot of information flows. Hence, if genes with a high betweenness centrality have a defect, then it is to be expected that this will cause changes in the expression of many genes in the Autism Interactome, in particular also of the already known causative genes for Autism.

The following lines of code in R can help identify the vertices with the highest betweeness centrality in the Steiner Tree, which are **not** already known to be significant for Autism:

```
# compute the betweeness centrality scores for each node
betweeness_centrality_scores = igraph::betweenness(HS.stree)

# now identify only those NOT already known to be significant
significant_centrality = c()
count = 0
for (i in 1:length(betweeness_centrality_scores)){
        if (!(as.numeric(names(betweeness_centrality_scores[i])) %in% signif.ids)) {
                significant_centrality = c(significant_centrality,
betweeness_centrality_scores[i])
        }
}

# sort
significant_centrality  = sort(significant_centrality, decreasing=TRUE)
length(significant_centrality)
```