**Lab 6**. Co-expression Networks. In a previous lab, we clustered samples/subjects. We used a hierarchical method called UPGMA to visualize the similarity between samples for quality control. Hierarchical methods use trees to visualize relationships and a distance metric. In this lab, we will use focus on clustering genes instead of samples and using network/graph methods to visualize relationships. These networks define connections based on correlation or co-expression between genes. Thus, they are called co-expression networks. Nodes in the network or graph are genes and edges are correlation between the genes. We will identify subclusters of genes that may represent functional groups or modules of genes that may serve some common function.

**Step 0**. Process and filter data following previous labs. Below we do a more strict coefficient of variation (CoV) filter so the co-expression networks are more manageable.

# load gene expression data

load("sense.filtered.cpm.Rdata") # setwd!

# load phenotype (mdd/hc) data

subject.attrs <- read.csv("Demographic\_symptom.csv",

stringsAsFactors = FALSE)

library(dplyr)

# grab intersecting X (subject ids) and Diag (Diagnosis) from columns

phenos.df <- subject.attrs %>%

filter(X %in% colnames(sense.filtered.cpm)) %>%

dplyr::select(X, Diag)

mddPheno <- as.factor(phenos.df$Diag)

**# Normalized and transform**

library(preprocessCore)

mddExprData\_quantile <- normalize.quantiles(sense.filtered.cpm)

mddExprData\_quantileLog2 <- log2(mddExprData\_quantile)

# attach phenotype names and gene names to data

colnames(mddExprData\_quantileLog2) <- mddPheno

rownames(mddExprData\_quantileLog2) <- rownames(sense.filtered.cpm)

**# coefficient of variation filter** sd(x)/abs(mean(x))

CoV\_values <- apply(mddExprData\_quantileLog2,1,

function(x) {sd(x)/abs(mean(x))})

# smaller threshold, the higher the experimental effect relative to the

# measurement precision

**thresh <- .02**

sum(CoV\_values<.**thresh**)

# there is one gene that has 0 variation -- remove

sd\_values <- apply(mddExprData\_quantileLog2,1, function(x) {sd(x)})

rownames(mddExprData\_quantileLog2)[sd\_values==0]

# filter the data matrix

GxS.covfilter <- mddExprData\_quantileLog2[CoV\_values< **thresh** & sd\_values>0,]

dim(GxS.covfilter)

**# convert phenotype to factor**

pheno.factor <- as.factor(colnames(GxS.covfilter))

pheno.factor

str(pheno.factor)

levels(pheno.factor)

**A**. Co-expression Networks. Use the code below to create a co-expression matrix from the gene expression data and an adjacency matrix with a correlation threshold of 0.7.

**1.** What are the sizes of mddCorr and adjMat? How are the two matrices related?

mddCorr: 781x781

adjMat: 781x781

adjMat is just a piece wise function of mddCorr as defined by: {if x > .7: 1 else: 0}

so it marks highly correlated genes with a 1 and less correlated with 0.

mddCorr<-cor(t(GxS.covfilter)) # correlation between genes

thresh<-.7 # controls sparsity of network

# threshold and turn T/F to 1/0

adjMat <- (abs(mddCorr)>thresh)+0

diag(adjMat) <- 0 # remove self-connections

rownames(adjMat) <- row.names(GxS.covfilter)

colnames(adjMat) <- row.names(GxS.covfilter)

Use the code below to create a network graph from regular matrix adjMat. Show the graph and histogram of edges.

**2.** In the network, describe and explain the many unconnected nodes that surround the connected network. In your explanation, consider the histogram and the threshold that was used.

library(igraph)

ig <- graph.adjacency(adjMat, mode = "undirected")

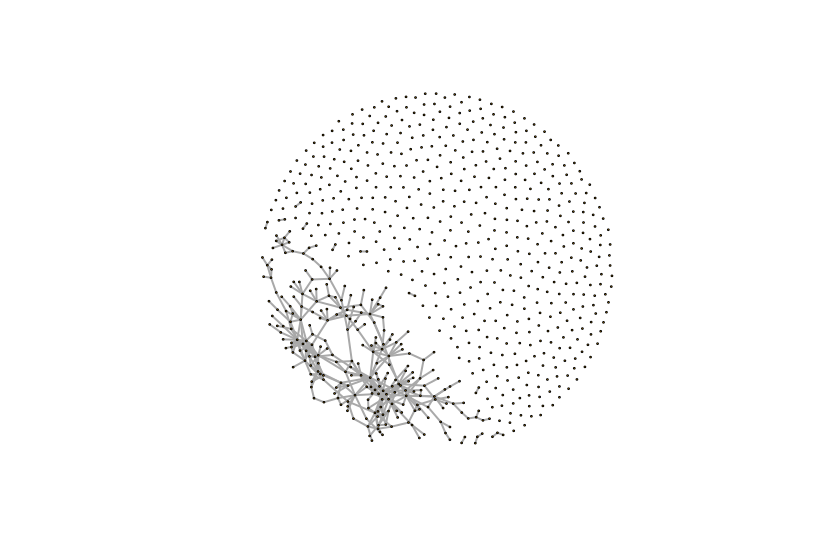
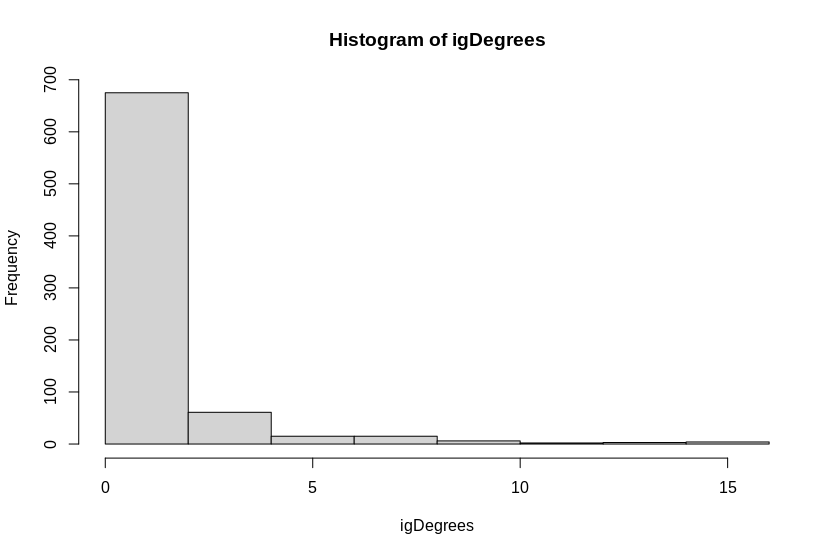
plot(ig, vertex.size=1, vertex.label.color = "black",

edge.width=2, vertex.label=NA)

igDegrees <- rowSums(adjMat) # degree vector for nodes

hist(igDegrees)

Most of the nodes are unconnected. The reason for that is the relatively high bar that we use to mark a node as connected to another (> 0.7). Thus, most of the genes have a correlation lower than .7 with all other genes. We could change that threshold to .3 and the number of nodes with degree >= 1 would increase greatly.



Use the following code to remove genes with 0 degree. Then add code like above to create a histogram and network graph.

**3.** Show the plots; how are they similar or different from the previous plots?

The histograms are similar because the trend of many with low degree and few with high degree remains. they are different because all of the nodes with degree 0 are removed.

The second network graph is basically just a zoomed in graph focused on the section that is connected. Since the nodes with degree 0 were all removed, we just have a loss of information but the information still there is not different.

# remove genes that have no connections

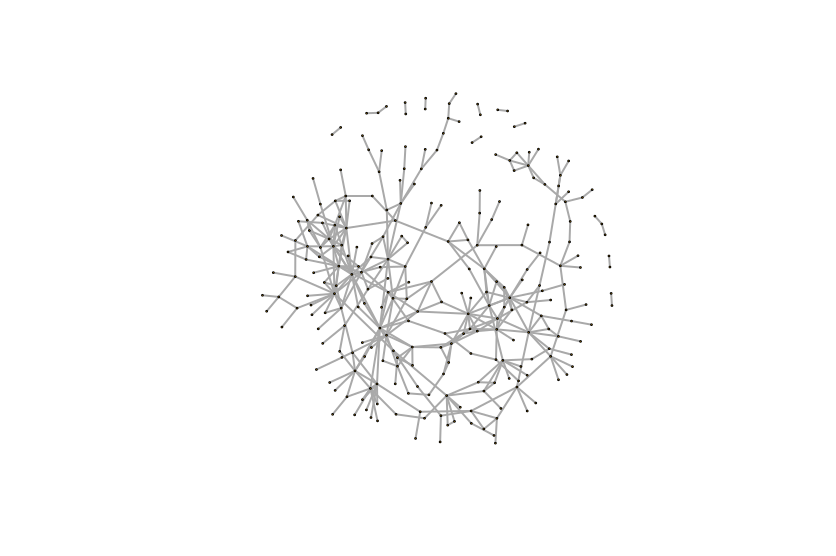
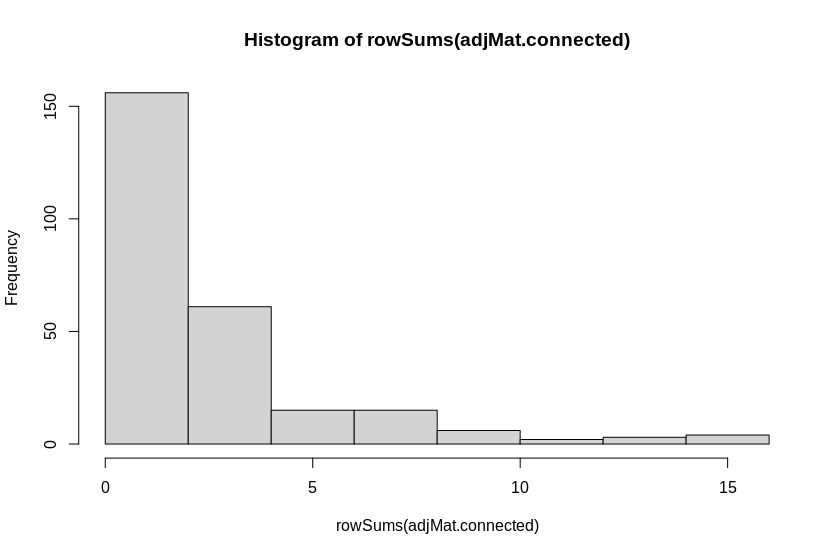
adjMat.connected <- adjMat[igDegrees!=0,igDegrees!=0]

ig.connected <- graph.adjacency(adjMat.connected, mode = "undirected")

plot(ig.connected, vertex.size=1, vertex.label.color = "black",

edge.width=2, vertex.label=NA)

hist(rowSums(adjMat.connected))



**B**. Co-expression network clustering. Use the fastgreedy.community graph clustering algorithm from igraph to partition the network. It uses a spectral (eigenvalue) algorithm called Modularity for clustering. Modularity is a network-based method for clustering, in contrast to the tree-based method we used before.

**4.** How many clusters are there? What are there sizes?

# fast community detection algorithm

igc.clusts <- fastgreedy.community(ig.connected)

sizes(igc.clusts)

igc.membership <- membership(igc.clusts)

[in]length(igc.clusts) [out] 22

sizes(igc.clusts)

22 clusters

[in] sizes(igc.clusts)

[out]1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

46 34 45 20 20 21 19 13 5 9 6 3 3 2 2 2 2 2 2 2 2 2

total number of nodes in all clusters:

[in] length(igc.membership)

[out] 262

Use the code below to color-code the nodes in the network by cluster membership.

**5.** Show the plot.

igc.colors <- igc.membership

color.pallet <- rainbow(length(igc.clusts)) # discrete color for each cluster

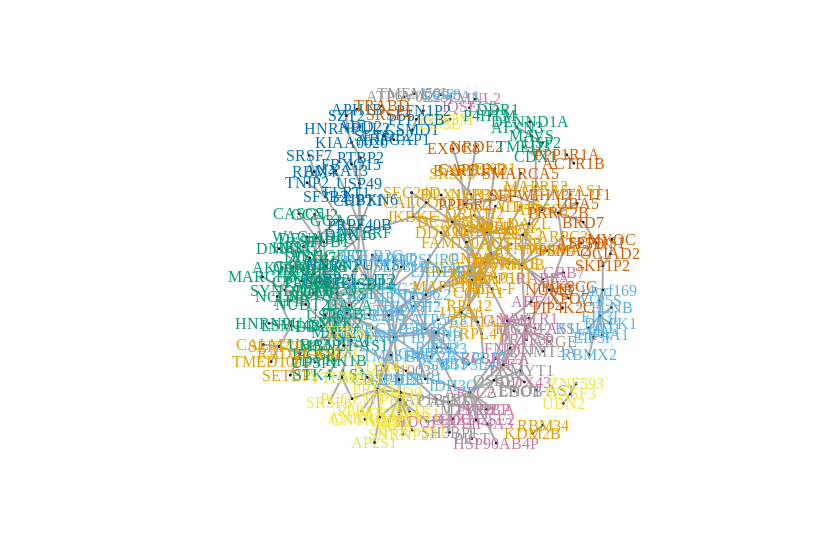
for (i in (1:length(igc.clusts))){ # change membership to the color pallet

igc.colors[i==igc.colors]<-color.pallet[i]

}

plot(ig.connected, vertex.size=1, vertex.label.color = igc.membership,

vertex.color = igc.membership, edge.width=2)



**6.** Show the top Reactome pathway for the two largest clusters and the overlapping genes. The following code will create a file with a column of genes for module 1. You can open your clust1.txt file in R and paste the column into MSigDB. When you use Reactome change the FDR to 1 from 0.05 on MSigDB.

clust1.genes<-names(igc.membership)[igc.membership==1]

write.table(clust1.genes,file="clust1.txt",row.names=F,col.names=F,quote=F)

largest cluster: REACTOME\_NERVOUS\_SYSTEM\_DEVELOPMENT

|  |
| --- |
| RPL12 |
| RPL41 |
| PSME4 |
| CAP2 |
| ARPC3 |
| RHOB |

Second largest cluster: REACTOME\_CYTOCHROME\_C\_MEDIATED\_APOPTOTIC\_RESPONSE

|  |
| --- |
| APIP |
| DIABLO |
| UACA |

**C.** Introduction to GWAS (genome-wide association study) data. GWAS data consists of thousands or millions of single nucleotide polymorphisms (SNPs) for each subject. Each SNP can has two alleles, which can have values (e.g., A or a). A subject can then have one of three possible genotypes for the SNP (AA, Aa or aa). The ultimate goal of GWAS is to see if there are statistical genetic associations between SNPs and case/control phenotypes or quantitative phenotypes.

Reading PLINK files. PLINK is an extensive C++ software for analyzing genome-wide association studies (GWAS) of diseases and traits. PLINK is also the name of the standard file format for GWAS data. The data is organized into two files. The ped (pedigree) file contains most of the data, where each row contains data for a subject. The first 6 columns contain subject ids, demographic and phenotype information. The sixth column is the phenotype (disease status or trait). The remaining columns are the SNP data. Each SNP requires two columns, one for each allele of the genotype.

(https://www.cog-genomics.org/plink2/formats#ped)

The ped file does not contain information about the SNP names. The map file provides the SNP identifiers (rs-ids) and information about each SNP (chromosome and location). Each row of the map file is a SNP, and these rows map to the SNP columns in the ped file.

(https://www.cog-genomics.org/plink2/formats#map)

Typically GWAS data are very large, and R may not be the best platform for the initial analysis. However, in this lab we will use a small tutorial data set from the PLINK website. The data is provided on Harvey.

<http://zzz.bwh.harvard.edu/plink/tutorial.shtml>

Use the code below with snpStats to read the PLINK data in files extra.ped and extra.map. **7.** What is the size (dim) of genotypes? How many SNPs and subjects are in the data?

89 subjects, 17 SNP

**8.** How many subjects are in each group (1 and 0) of the phenotype (hint: use table)?

[in]print(sum(phenotype==0))

[out]41

[in]print(sum(phenotype==1))

[out] 48

library(snpStats) # install first

ex.data <- read.pedfile(file="extra.ped", snps="extra.map")

ex.data$fam

phenotype <- ex.data$fam$affected-1 # change pheno from 1/2 to 0/1

genotypes <- ex.data$genotypes # encoded as AA/AB/BB

Use the code below to convert the genotype data to a data frame of factors with the column names of the data frame as rs-numbers.

**9.** How many subjects have each genotype for SNP rs630969? For each genotype for this SNP, what is the mixture of 0 and 1 phenotypes? You can use the A/A, A/B, B/B encoding.

snp.ids <- as.character(ex.data$map$snp.names)

genotypes.df <- data.frame(as(genotypes, "character"))

table(genotypes.df$rs630969)

table(phenotype,genotypes.df$rs630969)

phenotype A/A A/B B/B

0 11 17 13

1 10 21 17

**10.** What are the two alleles and three possible genotypes for SNP rs630969?

ex.data$map

snprow <- which(ex.data$map$snp.names=="rs630969")

ex.data$map$allele.1[snprow]

ex.data$map$allele.2[snprow]

“C”, “A”

AA, AC, AA