732A51 Bioinformatics

LAB 5 Bioinformatics

Hugo Morvan
William Wiik

STIMA
Department of Computer and Information Science
Linköpings universitet

2024-12-13

Contents

1 Question 1

Go to the webpage http://snap.stanford.edu/biodata/ and choose one of the provided datasets. Download it and reproduce the statistics concerning the graph. If you obtain different values, then discuss this in your report. Visualize the graph. The next step is to try to identify some clusters (communities in the graph). You can follow the tutorial at https://psych-networks.com/r-tutorial-identify-communities-items-networks/ to achieve this. Once you have found some clusters, identify the elements in it and try to find information on this cluster. Is it related to some known biological phenomena? If you do not find anything, then document your search attempts. If it will not be possible to do this question on the whole downloaded graph, then you may take some sub-graph of it.

```
library(qgraph)
```

```
## Warning: package 'qgraph' was built under R version 4.4.2
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.4.2
```

```
library(ape)
```

```
## Warning: package 'ape' was built under R version 4.4.2
```

```
con <- gzfile("ChG-Miner_miner-chem-gene.tsv.gz", "rt")
data <- read.table(con, sep = "\t", header = FALSE)
close(con)
colnames(data) <- c("Drug","Gene")

g <- graph_from_data_frame(data, directed = TRUE)
g
```

```
## IGRAPH 61b6077 DN-- 7341 15138 --
## + attr: name (v/c)
## + edges from 61b6077 (vertex names):
##  [1] DB00357->P05108 DB02721->P00325 DB00773->P23219 DB07138->Q16539
##  [5] DB08136->P24941 DB01242->P23975 DB01238->P08173 DB00186->P48169
##  [9] DB00338->P10635 DB01151->P08913 DB01244->P05023 DB01745->P07477
## [13] DB01996->P08254 DB04800->P18031 DB08352->Q16539 DB00133->P21549
## [17] DB00163->P21266 DB00197->P10632 DB06777->P08684 DB01151->P10635
## [21] DB00356->P08684 DB01589->P34903 DB01272->P20645 DB08846->Q14534
## [25] DB01151->P33261 DB01076->P04035 DB00605->Q03181 DB08515->P49721
## [29] DB02401->P07195 DB01057->P06276 DB03286->P11217 DB08814->Q9Y233
## + ... omitted several edges
```

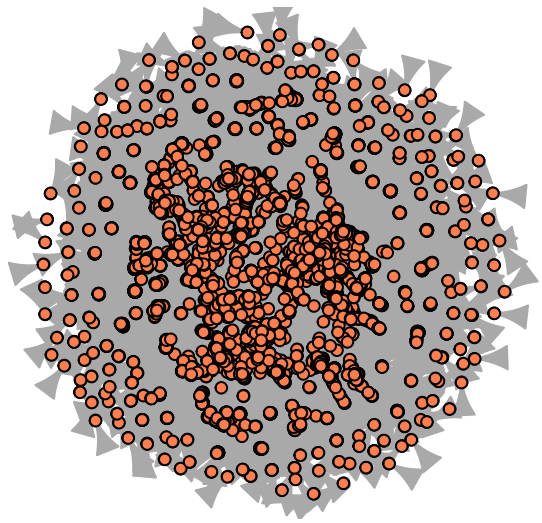
```
# Number of nodes and edges
cat("Number of nodes:", vcount(g), "\n")
```

```
## Number of nodes: 7341
```

```
cat("Number of edges:", ecount(g), "\n")
```

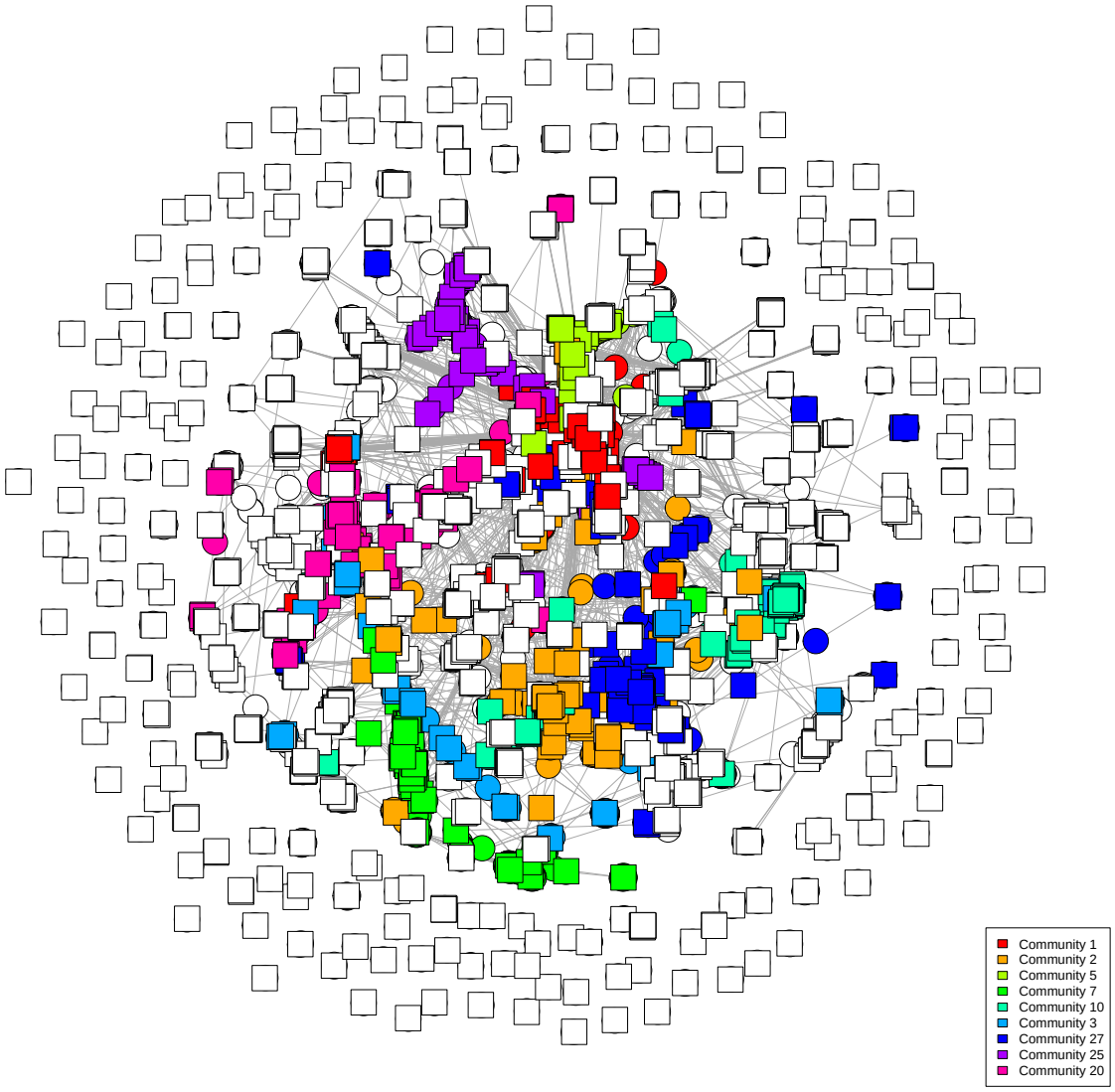
```
## Number of edges: 15138
```

```r
plot(g, vertex.size=5,vertex.label=NA, edge.width=0.01,vertex.color="coral", main="ChG-Miner_miner-chem-
```




```r
# Function to create and plot a bipartite network from an edge matrix (We got help from Claude.ai)
plot_bipartite_network <- function(edge_matrix, title = "Bipartite Network",plot_graph = TRUE, top_n = 9
  # Create unique node names
  drugs <- unique(edge_matrix[,1])
  genes <- unique(edge_matrix[,2])

  # Create the graph
  g <- make_empty_graph(directed = FALSE)

  # Add all nodes
  all_nodes <- c(drugs, genes)
  g <- add_vertices(g, length(all_nodes), name = all_nodes)

  # Create type attribute for bipartite graph
  V(g)$type <- c(rep(TRUE, length(drugs)), rep(FALSE, length(genes)))
  V(g)$shape <- ifelse(V(g)$type, "circle", "square")

  edges <- apply(edge_matrix, 1, function(x) c(which(V(g)$name == x[1]), which(V(g)$name == x[2])))
```

```r
  # Add all edges at once
  g <- add_edges(g, unlist(edges))


  comms <- cluster_louvain(g)
  freq_memb <- table(membership(comms))
  big_mems <- freq_memb[order(freq_memb,decreasing = T)][1:top_n]

  right_index <- as.vector(as.numeric(names(big_mems)))
  mems <- membership(comms)
  comm_col <- rainbow(top_n)[mems]
  comm_col <- rep("#FFFFFF",length(comm_col))

  topncol <- c("#FF0000", "#FFAA00", "#AAFF00", "#00FF00" ,"#00FFAA",
               "#00AAFF", "#0000FF" ,"#AA00FF", "#FF00AA")

  for(i in 1:length(right_index)) {
    comm_col[which(mems == right_index[i])] <- topncol[i]
  }

  # Plot the network
  if(plot_graph) {

    plot(g,
         vertex.label = V(g)$name,
         vertex.shape = V(g)$shape,
         vertex.color = comm_col,
         vertex.label.cex = 0.01,  # Adjust label size
         vertex.size = 5,         # Node size
         vertex.label.color = "black",
         main = title#,
         #layout = layout_as_bipartite  # Specific layout for bipartite graphs
    )
    legend("bottomright", legend = paste("Community", right_index),
           fill = topncol)

    # Return the graph object for further analysis if needed
    return(g)
  } else {
    return(comms)
  }
}

# Example usage:
# Create a sample edge matrix (drug-gene connections)
edge_matrix <- data

# Plot the network
plot_bipartite_network(edge_matrix, "Drug-Gene Interaction Network")
```




```
## IGRAPH 7744741 UN-B 7341 15138 --
## + attr: name (v/c), type (v/l), shape (v/c)
## + edges from 7744741 (vertex names):
##  [1] DB00357--P05108 DB02721--P00325 DB00773--P23219 DB07138--Q16539
##  [5] DB08136--P24941 DB01242--P23975 DB01238--P08173 DB00186--P48169
##  [9] DB00338--P10635 DB01151--P08913 DB01244--P05023 DB01745--P07477
## [13] DB01996--P08254 DB04800--P18031 DB08352--Q16539 DB00133--P21549
## [17] DB00163--P21266 DB00197--P10632 DB06777--P08684 DB01151--P10635
## [21] DB00356--P08684 DB01589--P34903 DB01272--P20645 DB08846--Q14534
```

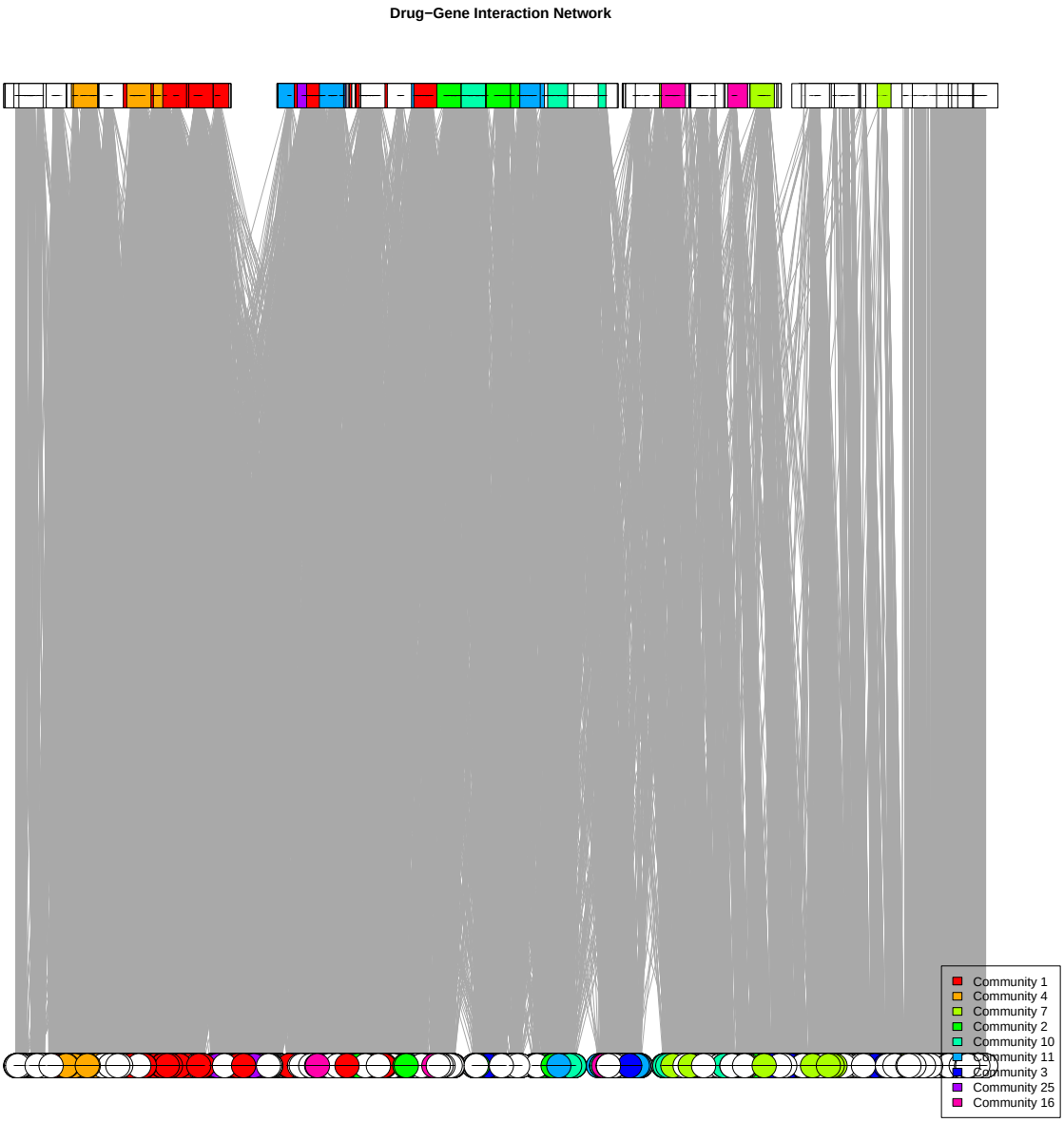
```
## [25] DB01151--P33261 DB01076--P04035 DB00605--Q03181 DB08515--P49721
## [29] DB02401--P07195 DB01057--P06276 DB03286--P11217 DB08814--Q9Y233
## + ... omitted several edges
```

Drug-Gene interactions are plotted. The 9 biggest communities have been visualized with colors, the rest are white. Notice that a lot of Gene-Drug interaction happen only once, resulting in a lot of isolated pairs on the outskirt of the plot. Drugs are represented by circles and Gene by Squares.

1.1 Graph 2

```
# Function to create and plot a bipartite network from an edge matrix
plot_bipartite_network_2 <- function(edge_matrix, title = "Bipartite Network", top_n = 9) {
  # Create unique node names
  drugs <- unique(edge_matrix[,1])
  genes <- unique(edge_matrix[,2])

  # Create the graph
  g <- make_empty_graph(directed = FALSE)

  # Add all nodes
  all_nodes <- c(drugs, genes)
  g <- add_vertices(g, length(all_nodes), name = all_nodes)

  # Create type attribute for bipartite graph
  V(g)$type <- c(rep(TRUE, length(drugs)), rep(FALSE, length(genes)))

  # Add edges
  edges <- apply(edge_matrix, 1, function(x) c(which(V(g)$name == x[1]), which(V(g)$name == x[2])))

  # Add all edges at once
  g <- add_edges(g, unlist(edges))
  # Set node colors based on type
  #V(g)$color <- ifelse(V(g)$type, "lightblue", "lightgreen")

  comms <- cluster_louvain(g)
  freq_memb <- table(membership(comms))
  big_mems <- freq_memb[order(freq_memb,decreasing = T)][1:top_n]

  right_index <- as.vector(as.numeric(names(big_mems)))
  mems <- membership(comms)
  comm_col <- rainbow(top_n)[mems]
  comm_col <- rep("#FFFFFF",length(comm_col))

  topncol <- c("#FF0000", "#FFAA00", "#AAFF00", "#00FF00" ,"#00FFAA",
               "#00AAFF", "#0000FF" ,"#AA00FF", "#FF00AA")

  for(i in 1:length(right_index)) {
    comm_col[which(mems == right_index[i])] <- topncol[i]
  }
```

```r
  V(g)$shape <- ifelse(V(g)$type, "circle", "square")
  # Plot the network

  plot(g,
       vertex.label = V(g)$name,
       vertex.color = comm_col,
       vertex.label.cex = 0.1,  # Adjust label size
       vertex.size = 5,         # Node size
       vertex.label.color = "black",
       main = title,
       layout = layout_as_bipartite  # Specific layout for bipartite graphs
  )
  legend("bottomright", legend = paste("Community", right_index),
           fill = topncol)
  # Return the graph object for further analysis if needed
  return(g)
}

# Example usage:
# Create a sample edge matrix (drug-gene connections)
edge_matrix <- data

# Plot the network
plot_bipartite_network_2(edge_matrix, "Drug-Gene Interaction Network")
```




```
## IGRAPH 7f1cec9 UN-B 7341 15138 --
## + attr: name (v/c), type (v/l), shape (v/c)
## + edges from 7f1cec9 (vertex names):
##  [1] DB00357--P05108 DB02721--P00325 DB00773--P23219 DB07138--Q16539
##  [5] DB08136--P24941 DB01242--P23975 DB01238--P08173 DB00186--P48169
##  [9] DB00338--P10635 DB01151--P08913 DB01244--P05023 DB01745--P07477
## [13] DB01996--P08254 DB04800--P18031 DB08352--Q16539 DB00133--P21549
## [17] DB00163--P21266 DB00197--P10632 DB06777--P08684 DB01151--P10635
## [21] DB00356--P08684 DB01589--P34903 DB01272--P20645 DB08846--Q14534
```

```
## [25] DB01151--P33261 DB01076--P04035 DB00605--Q03181 DB08515--P49721
## [29] DB02401--P07195 DB01057--P06276 DB03286--P11217 DB08814--Q9Y233
## + ... omitted several edges
```

The Drug-Gene network can also be viewed as a bipartite graph, since Drug never connect to other drugs directly, and likewise for genes. Drugs are represented by circles and Gene by Squares.

```
top_n <- 9
communitys <- plot_bipartite_network(edge_matrix, plot_graph = FALSE)
freq_memb <- table(membership(communitys))
big_mems <- freq_memb[order(freq_memb,decreasing = T)][1:top_n]

right_index <- as.vector(as.numeric(names(big_mems)))

biggest_com <- communitys[[right_index[1]]]


# install.packages("BiocManager")
# BiocManager::install("clusterProfiler")
# BiocManager::install("org.Hs.eg.db")
# devtools::install_github("YuLab-SMU/clusterProfiler")

library(clusterProfiler)
```

```
## Warning: package 'clusterProfiler' was built under R version 4.4.2
##
## clusterProfiler v4.14.4 Learn more at https://yulab-smu.top/contribution-knowledge-mining/
##
## Please cite:
##
## S Xu, E Hu, Y Cai, Z Xie, X Luo, L Zhan, W Tang, Q Wang, B Liu, R Wang,
## W Xie, T Wu, L Xie, G Yu. Using clusterProfiler to characterize
## multiomics data. Nature Protocols. 2024, doi:10.1038/s41596-024-01020-z
##
## Attaching package: 'clusterProfiler'
## The following object is masked from 'package:igraph':
##
##     simplify
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(org.Hs.eg.db)
```

```
## Loading required package: AnnotationDbi
## Loading required package: stats4
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:igraph':
##
##     normalize, path, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: IRanges

## Warning: package 'IRanges' was built under R version 4.4.2

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:clusterProfiler':
##
##     rename

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:clusterProfiler':
##
##     slice
```

```
## The following object is masked from 'package:grDevices':
##
##     windows
```

```
##
## Attaching package: 'AnnotationDbi'
```

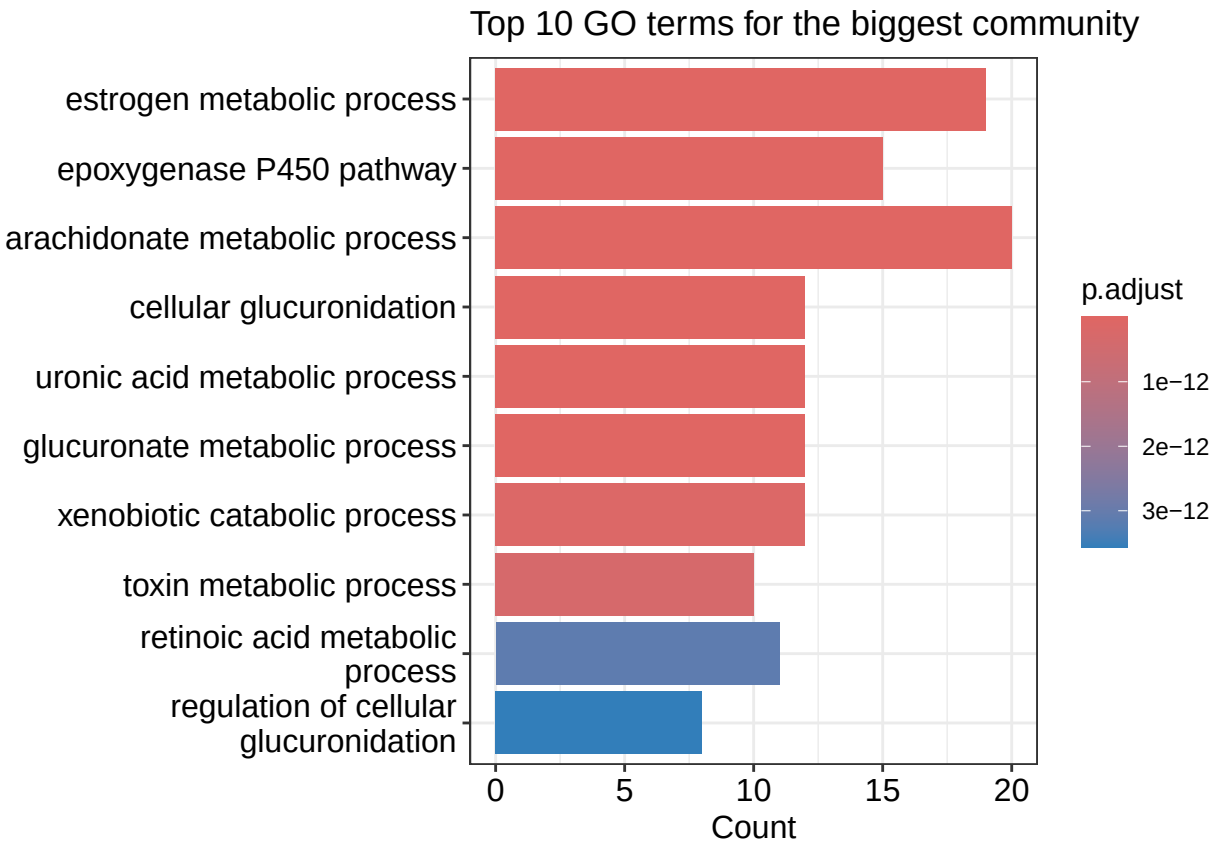
```
## The following object is masked from 'package:clusterProfiler':
##
##     select
```

```
##
```

```
# Perform GO enrichment analysis for the current community
go_results <- enrichGO(gene = biggest_com,
                       OrgDb = org.Hs.eg.db, keyType = "UNIPROT", ont = "BP")

top_go_terms <- go_results@result[order(go_results@result$p.adjust), ]
head(top_go_terms$Description, 10) # strongly related to neurobiology
```

```
##  [1] "estrogen metabolic process"
##  [2] "epoxygenase P450 pathway"
##  [3] "arachidonate metabolic process"
##  [4] "cellular glucuronidation"
##  [5] "uronic acid metabolic process"
##  [6] "glucuronate metabolic process"
##  [7] "xenobiotic catabolic process"
##  [8] "toxin metabolic process"
##  [9] "retinoic acid metabolic process"
## [10] "regulation of cellular glucuronidation"
```

```
barplot(go_results, showCategory = 10, title = "Top 10 GO terms for the biggest community")
```



the common point between these is that they are all metabolic processes.

```
library(ggplot2)
```

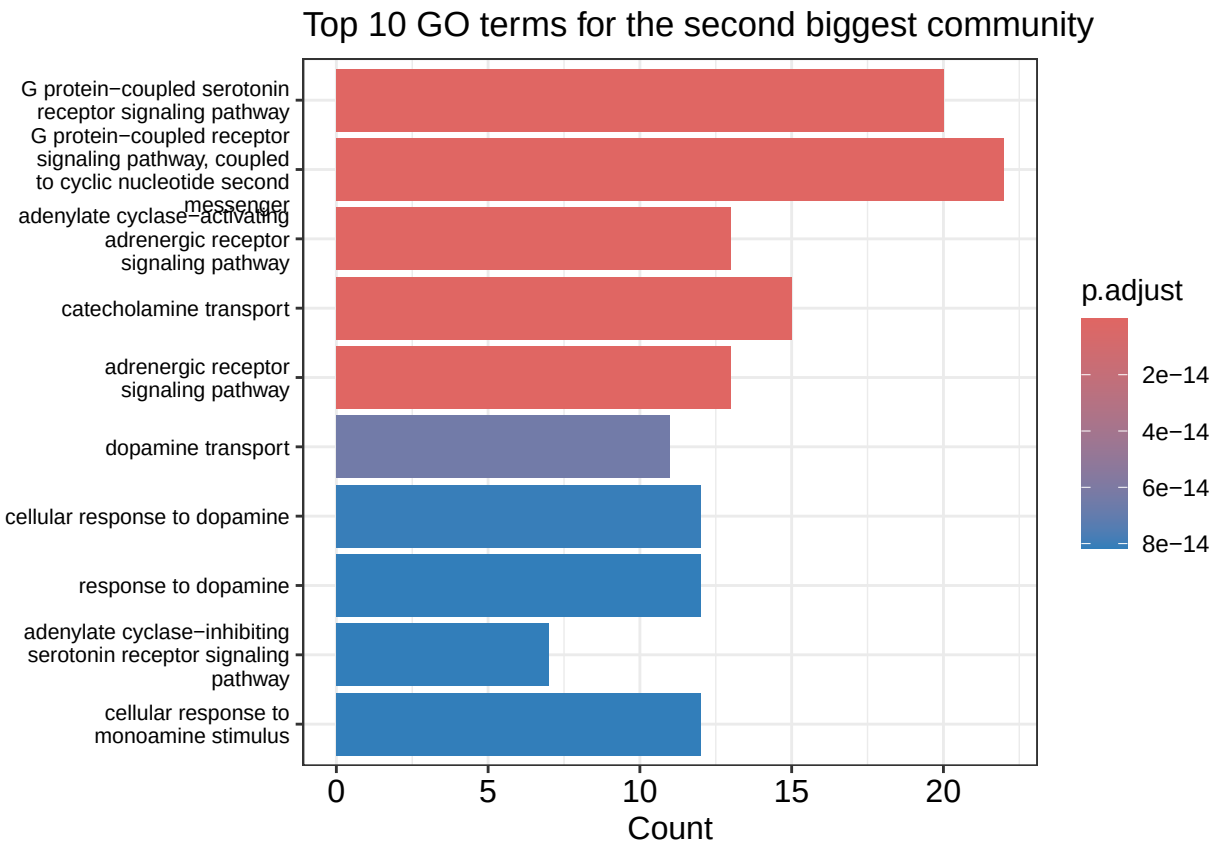
```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
sec_biggest_com <- communitys[[right_index[2]]]


# Perform GO enrichment analysis for the current community
go_results <- enrichGO(gene = sec_biggest_com,
                       OrgDb = org.Hs.eg.db, keyType = "UNIPROT", ont = "BP")

top_go_terms <- go_results@result[order(go_results@result$p.adjust), ]
head(top_go_terms$Description, 10)
```

```
##  [1] "G protein-coupled serotonin receptor signaling pathway"
##  [2] "G protein-coupled receptor signaling pathway, coupled to cyclic nucleotide second messenger"
##  [3] "adenylate cyclase-activating adrenergic receptor signaling pathway"
##  [4] "catecholamine transport"
##  [5] "adrenergic receptor signaling pathway"
##  [6] "dopamine transport"
##  [7] "cellular response to dopamine"
```

```
##  [8] "response to dopamine"
##  [9] "adenylate cyclase-inhibiting serotonin receptor signaling pathway"
## [10] "cellular response to monoamine stimulus"
```

```
barplot(go_results, showCategory = 10, title = "Top 10 GO terms for the second biggest community") +
  theme(axis.text.y = element_text(size = 8))
```



The common point between these seems to be that these are all signaling pathways or response mechanisms.

2 Question 2

Recreate one of the three analyses that can be found on https://strimmerlab.github.io/software/genenet/index.html. Document and discuss all your steps. In the analyses there is the step where you select the edges to keep. There a particular criterion is chosen for edge inclusion. Vary this criterion and explore how the resulting clusters will differ with the changes. Take one found cluster, identify the elements in it and try to find information on this cluster. Is it related to some known biological phenomena? If you do not find anything, then document your search attempts.

2.1 Load GeneNet package

```
library("GeneNet")
```

```
## Loading required package: corpcor
## Loading required package: longitudinal
## Loading required package: fdrtool
```

```
library("Rgraphviz")
```

```
## Loading required package: graph
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:igraph':
##
##     normalize, path, union
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min
##
## Attaching package: 'graph'
## The following objects are masked from 'package:ape':
##
##     complement, degree, edges
## The following objects are masked from 'package:igraph':
##
##     degree, edges, intersection
## Loading required package: grid
```

Loading the needed dataset; E. Coli data set (9 time points for 102 genes):

```
data(ecoli)
dim(ecoli)
```

```
## [1]   9 102
```

2.2 Estimation of partial correlations

Estimate matrix of partial correlation using a shrinkage estimator: A shrinkage estimator is a type of statistical method for estimating genetic effects. Shrinkage estimators involve "shrinking" or reducing the estimates

towards zero, thereby reducing variability and improving precision compared to non-shrunken (ordinary) least squares estimates.

```
pc = ggm.estimate.pcor(ecoli)
```

```
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.1804
```

```
dim(pc)
```

```
## [1] 102 102
```

Assign p-values, q-values and empirical posterior probabilities to all 5151 potential edges in the network:

```
ecoli.edges = network.test.edges(pc, direct=TRUE, fdr=TRUE)
```

```
## Estimate (local) false discovery rates (partial correlations):
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```

```
##
## Estimate (local) false discovery rates (log ratio of spvars):
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```

```
dim(ecoli.edges)
```

```
## [1] 5151   10
```

The table lists all edges in the order strength of partial correlations:

```
ecoli.edges[1:5,]
```

```
##         pcor node1 node2         pval         qval      prob    log.spvar
## 1  0.2318566    51    53 2.220446e-16 3.612205e-13 1.0000000 -0.043537019
## 2  0.2240555    52    53 2.220446e-16 3.612205e-13 1.0000000 -0.040249854
## 3  0.2150782    51    52 2.220446e-16 3.612205e-13 1.0000000 -0.003287165
## 4  0.1732886     7    93 3.108624e-15 3.792816e-12 0.9999945 -0.025293430
## 5 -0.1341889    29    86 1.120812e-09 1.093997e-06 0.9999516  0.022305368
##    pval.dir  qval.dir     prob.dir
## 1 0.3803869 0.7557272 1.110223e-16
## 2 0.4173922 0.7724561 1.110223e-16
## 3 0.9471949 0.8851073 1.110223e-16
## 4 0.6103234 0.8323249 1.110223e-16
## 5 0.6531371 0.8415749 1.110223e-16
```

2.3 Decide which edges to include in the network

In these section are presented different methods to decide which edges to include in the final network. One method is to use various False Discovery Rate (FDR) criterion. Another method is to ahve a fix number of

edges, and to only include the strongest edges.

To obtain a graph you need to select top ranking edges according to a suitable criterion. Here are some suggestions:

1. Use local fdr cutoff 0.2, i.e. include all edges with posterior probability of at least 0.8.

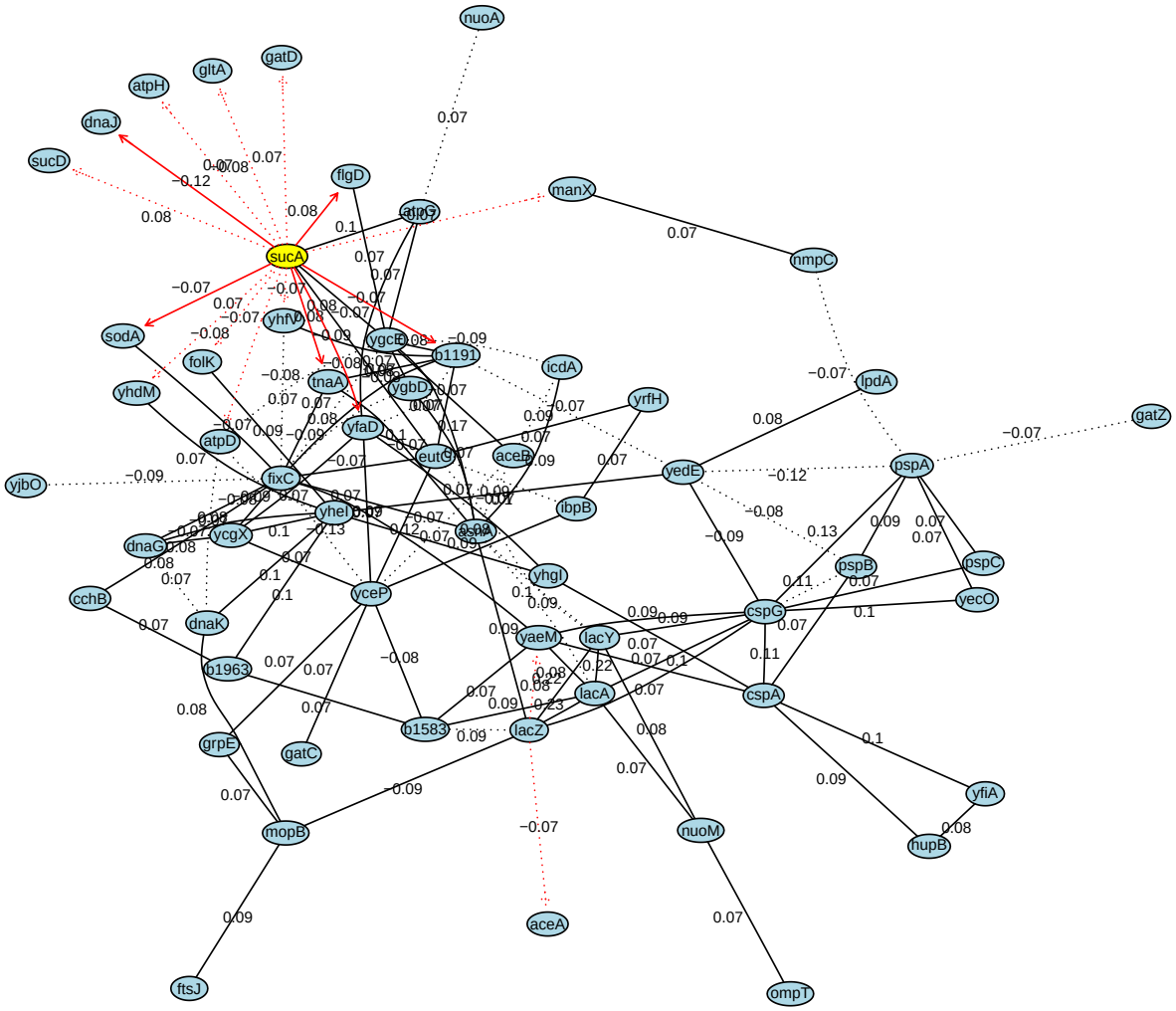
```
ecoli.net = extract.network(ecoli.edges)
```

```
##
## Significant edges:  125
##     Corresponding to  2.43 %  of possible edges
##
## Significant directions:  377
##     Corresponding to  7.32 %  of possible directions
## Significant directions in the network:  17
##     Corresponding to  13.6 %  of possible directions in the network
```

```
dim(ecoli.net)
```

```
## [1] 125  11
```

```
node.labels = colnames(ecoli)
gr = network.make.graph(ecoli.net, node.labels, drop.singles=TRUE)
globalAttrs = list()
globalAttrs$edge = list(color = "black", lty = "solid", lwd = 1, arrowsize=1)
globalAttrs$node = list(fillcolor = "lightblue", shape = "ellipse", fixedsize = FALSE)

nodeAttrs = list()
nodeAttrs$fillcolor = c('sucA' = "yellow")

edi = edge.info(gr)
edgeAttrs = list()
edgeAttrs$dir = edi$dir # set edge directions
edgeAttrs$lty = ifelse(edi$weight < 0, "dotted", "solid") # negative correlation -> dotted
edgeAttrs$color = ifelse(edi$dir == "none", "black", "red")
edgeAttrs$label = round(edi$weight, 2) # use partial correlation as edge labels
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



2. Use local fdr cutoff 0.1, i.e. i.e. include all edges with posterior probability of at least 0.9.

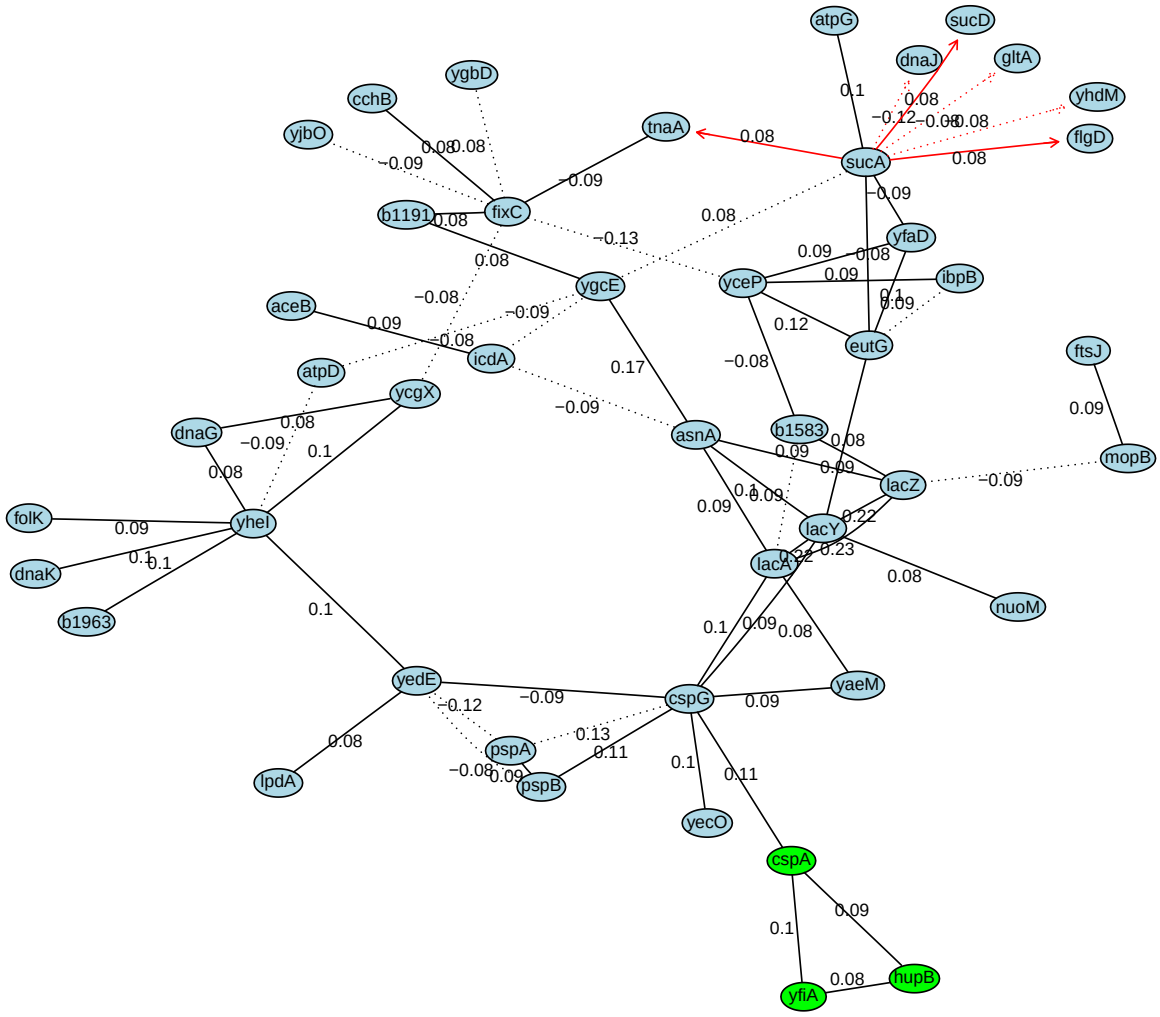
```
ecoli.net = extract.network(ecoli.edges, cutoff.ggm=0.9, cutoff.dir=0.9)
```

```
##
## Significant edges:  65
##     Corresponding to  1.26 %  of possible edges
##
## Significant directions:  269
##     Corresponding to  5.22 %  of possible directions
## Significant directions in the network:  6
##     Corresponding to  9.23 %  of possible directions in the network
```

```r
dim(ecoli.net)
```

```
## [1] 65 11
```

```r
gr = network.make.graph(ecoli.net, node.labels, drop.singles=TRUE)
globalAttrs = list()
globalAttrs$edge = list(color = "black", lty = "solid", lwd = 1, arrowsize=1)
globalAttrs$node = list(fillcolor = "lightblue", shape = "ellipse", fixedsize = FALSE)

nodeAttrs = list()
nodeAttrs$fillcolor = c('sucA' = "yellow")

nodeAttrs$fillcolor = c('cspA' = "green",'hupB' = "green",'yfiA' = "green")

edi = edge.info(gr)
edgeAttrs = list()
edgeAttrs$dir = edi$dir # set edge directions
edgeAttrs$lty = ifelse(edi$weight < 0, "dotted", "solid") # negative correlation -> dotted
edgeAttrs$color = ifelse(edi$dir == "none", "black", "red")
edgeAttrs$label = round(edi$weight, 2) # use partial correlation as edge labels
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



3. Include a fixed number of edges, say the 70 strongest edges

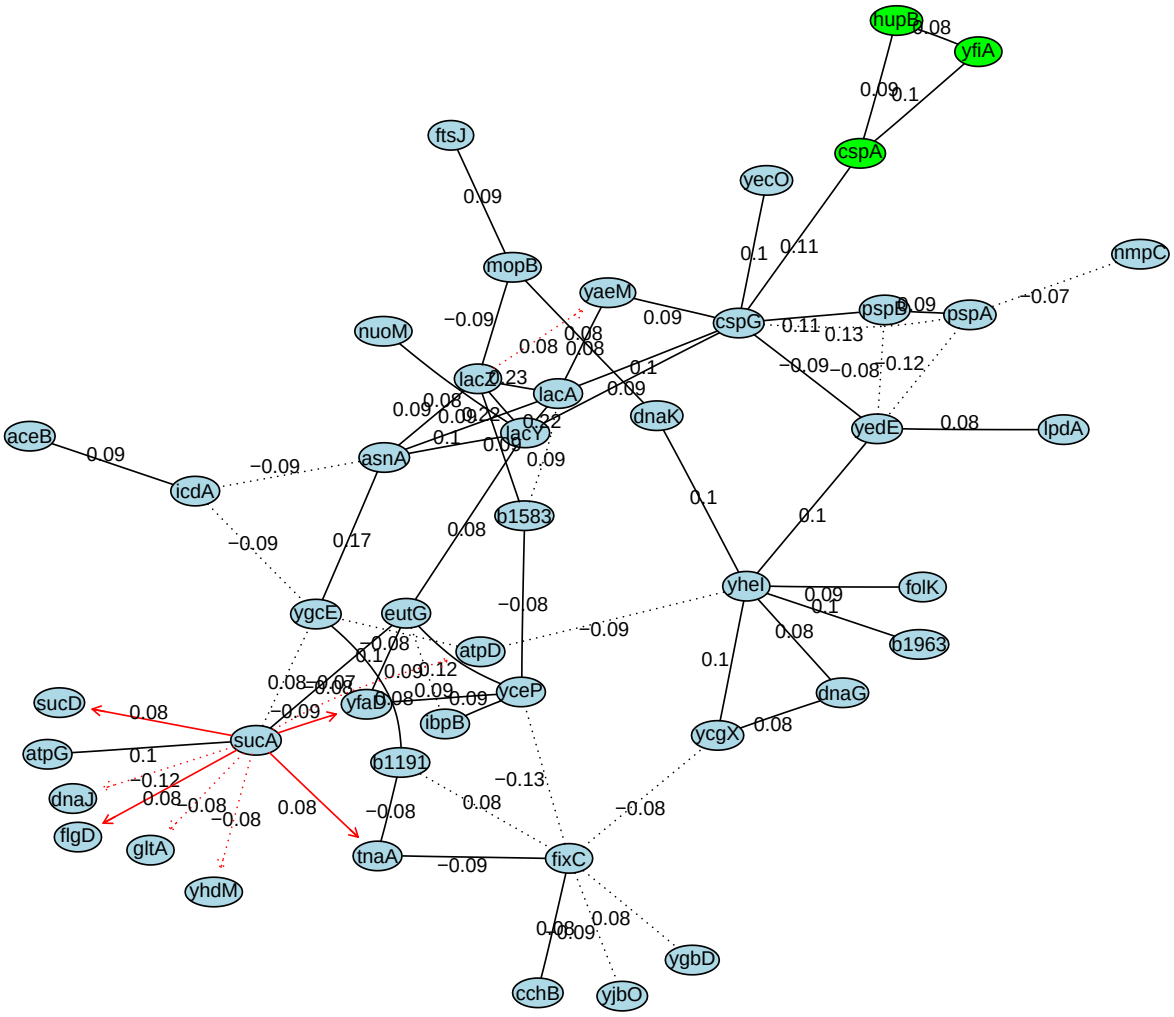
```
ecoli.net = extract.network(ecoli.edges, method.ggm="number", cutoff.ggm=70)
```

```
##
## Significant edges:  70
##     Corresponding to  1.36 %  of possible edges
##
## Significant directions:  377
##     Corresponding to  7.32 %  of possible directions
## Significant directions in the network:  9
##     Corresponding to  12.86 %  of possible directions in the network
```

```r
dim(ecoli.net)
```

```
## [1] 70 11
```

```r
gr = network.make.graph(ecoli.net, node.labels, drop.singles=TRUE)
globalAttrs = list()
globalAttrs$edge = list(color = "black", lty = "solid", lwd = 1, arrowsize=1)
globalAttrs$node = list(fillcolor = "lightblue", shape = "ellipse", fixedsize = FALSE)

nodeAttrs = list()
nodeAttrs$fillcolor = c('sucA' = "yellow")

nodeAttrs$fillcolor = c('cspA' = "green",'hupB' = "green",'yfiA' = "green")

edi = edge.info(gr)
edgeAttrs = list()
edgeAttrs$dir = edi$dir # set edge directions
edgeAttrs$lty = ifelse(edi$weight < 0, "dotted", "solid") # negative correlation -> dotted
edgeAttrs$color = ifelse(edi$dir == "none", "black", "red")
edgeAttrs$label = round(edi$weight, 2) # use partial correlation as edge labels
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



Plot network

For plotting we use the graph and Rgraphviz packages from Bioconductor.

```
library("Rgraphviz")
```

Create graph object from the list of edges:

```
node.labels = colnames(ecoli)
gr = network.make.graph(ecoli.net, node.labels, drop.singles=TRUE)
table(  edge.info(gr)$dir )
```

```
##
## forward    none
##       9      61
```

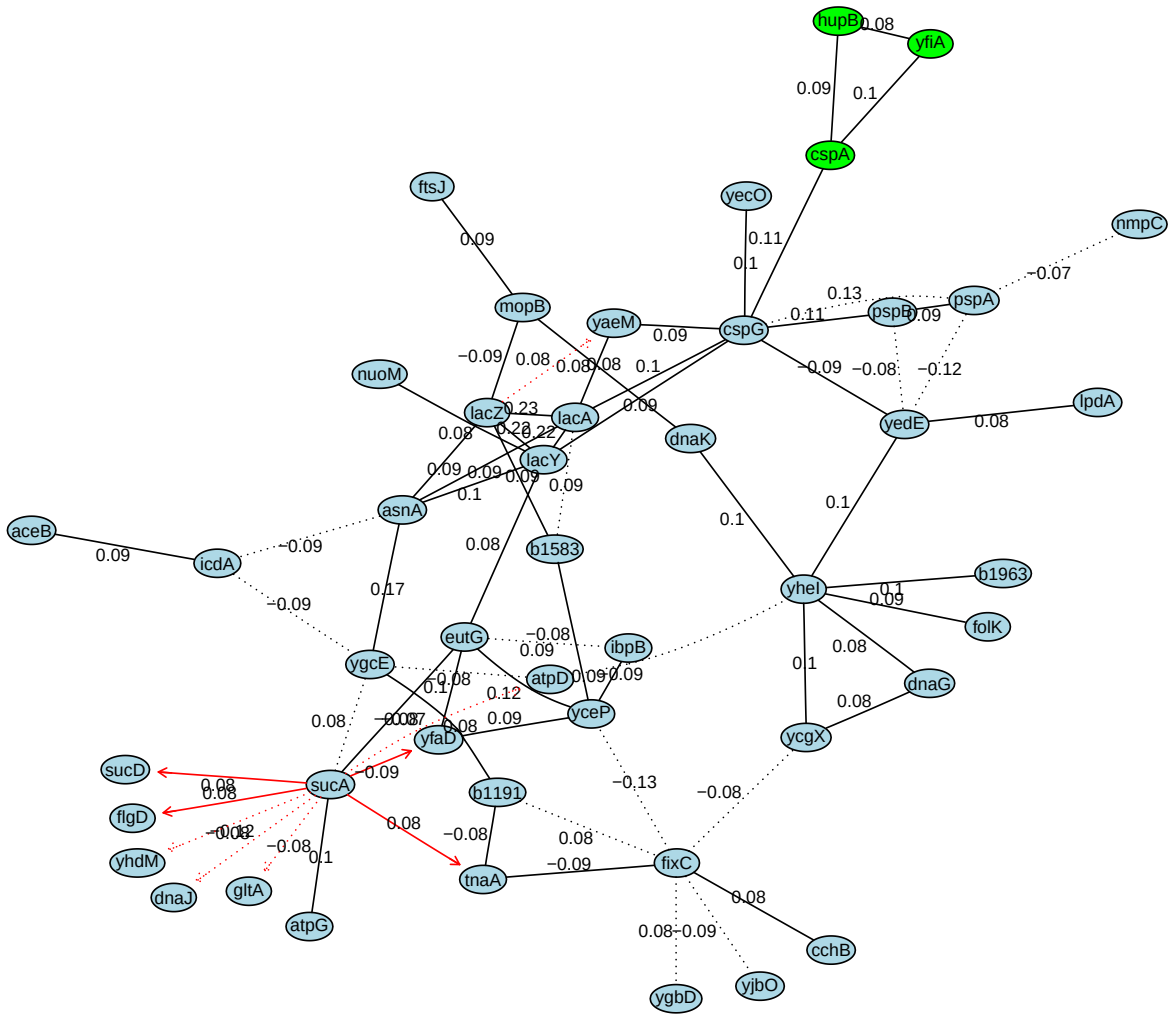
```
sort( node.degree(gr), decreasing=TRUE)
```

```
##  sucA  cspG  fixC  yheI  lacA  lacY  lacZ  asnA  eutG  yceP  yedE  ygcE  pspA
##    11     8     7     7     6     6     6     5     5     5     5     5     4
##  atpD b1191 b1583  cspA  icdA  mopB  pspB  tnaA  yaeM  ycgX  yfaD  dnaG  dnaK
##     3     3     3     3     3     3     3     3     3     3     3     2     2
##  hupB  ibpB  yfiA  aceB  atpG b1963  cchB  dnaJ  flgD  folK  ftsJ  gltA  lpdA
##     2     2     2     1     1     1     1     1     1     1     1     1     1
##  nmpC  nuoM  sucD  yecO  ygbD  yhdM  yjbO
##     1     1     1     1     1     1     1
```

Set node and edge attributes for more beautiful graph plotting:

```
globalAttrs = list()
globalAttrs$edge = list(color = "black", lty = "solid", lwd = 1, arrowsize=1)
globalAttrs$node = list(fillcolor = "lightblue", shape = "ellipse", fixedsize = FALSE)

nodeAttrs = list()
nodeAttrs$fillcolor = c('sucA' = "yellow")

nodeAttrs$fillcolor = c('cspA' = "green",'hupB' = "green",'yfiA' = "green")

edi = edge.info(gr)
edgeAttrs = list()
edgeAttrs$dir = edi$dir # set edge directions
edgeAttrs$lty = ifelse(edi$weight < 0, "dotted", "solid") # negative correlation -> dotted
edgeAttrs$color = ifelse(edi$dir == "none", "black", "red")
edgeAttrs$label = round(edi$weight, 2) # use partial correlation as edge labels
```

```
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



We observe that low values for the FDR cutoff value produces less significant edges in the network. We can demonstrate the opposite by having a higher FDR cutoff, which results in many more significant edges, and thus a denser graph.

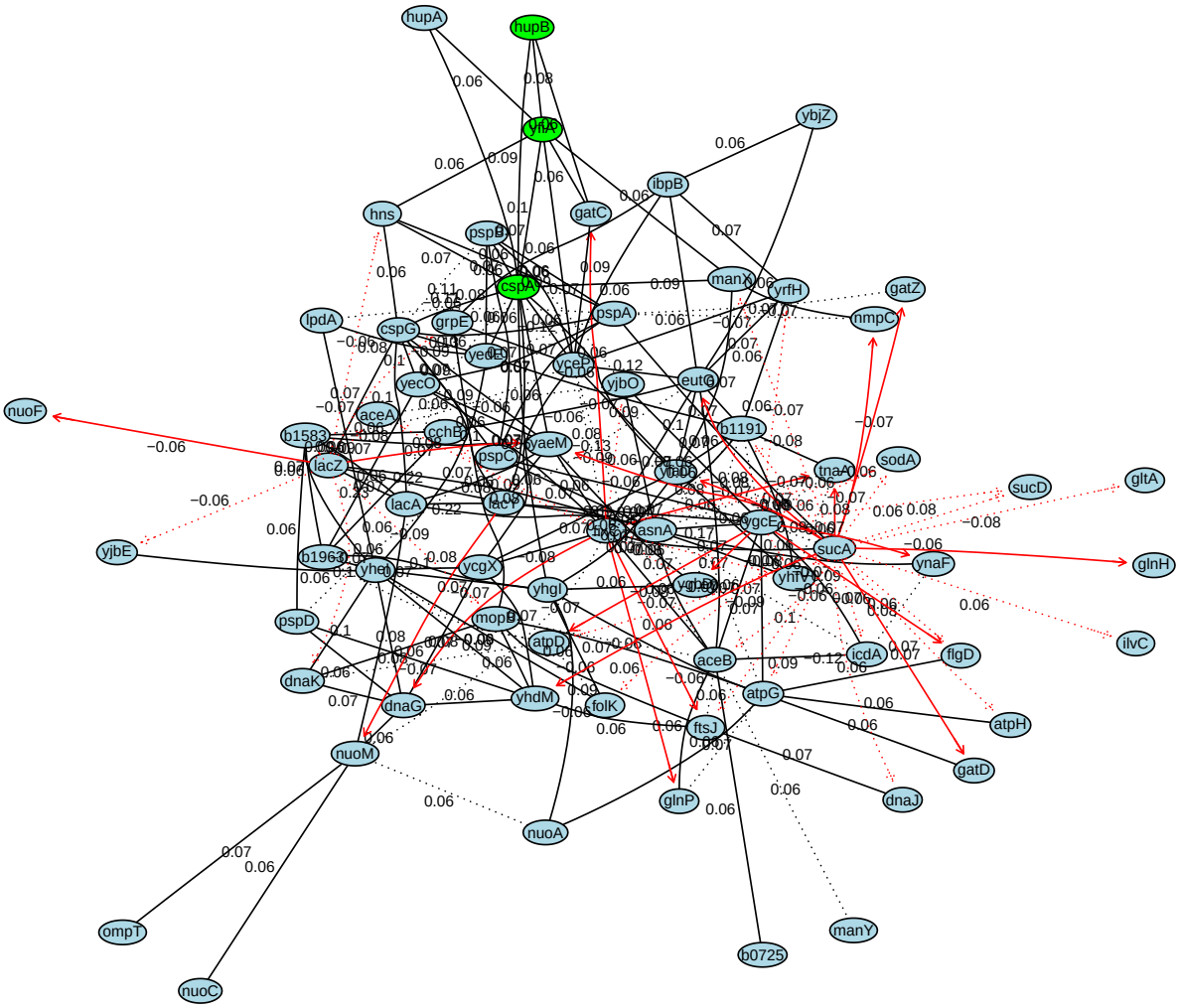
```
ecoli.net = extract.network(ecoli.edges, cutoff.ggm=0.5, cutoff.dir=0.5)
```

```
##
## Significant edges:  232
##     Corresponding to  4.5 %  of possible edges
##
## Significant directions:  651
##     Corresponding to  12.64 %  of possible directions
## Significant directions in the network:  54
##     Corresponding to  23.28 %  of possible directions in the network
```

```r
dim(ecoli.net)
```

```
## [1] 232  11
```

```r
gr = network.make.graph(ecoli.net, node.labels, drop.singles=TRUE)
globalAttrs = list()
globalAttrs$edge = list(color = "black", lty = "solid", lwd = 1, arrowsize=1)
globalAttrs$node = list(fillcolor = "lightblue", shape = "ellipse", fixedsize = FALSE)

nodeAttrs = list()
nodeAttrs$fillcolor = c('sucA' = "yellow")
nodeAttrs$fillcolor = c('cspA' = "green",'hupB' = "green",'yfiA' = "green")
#cspA - hupB - yfiA
edi = edge.info(gr)
edgeAttrs = list()
edgeAttrs$dir = edi$dir # set edge directions
edgeAttrs$lty = ifelse(edi$weight < 0, "dotted", "solid") # negative correlation -> dotted
edgeAttrs$color = ifelse(edi$dir == "none", "black", "red")
edgeAttrs$label = round(edi$weight, 2) # use partial correlation as edge labels
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



2.4 Cluster analysis:

Finding a common factor between these:

cspA - hupB - yfiA triangle/cluster in E. Coli data.

yfiA: https://ecocyc.org/gene?orgid=ECOLI&id=EG11151

hupB: https://ecocyc.org/gene?orgid=ECOLI&id=EG10467

cspA: https://ecocyc.org/gene?orgid=ECOLI&id=EG10166

All three genes appear to be involved in complex regulatory processes related to gene expression, protein interactions, or cellular stress responses. They also seems to have a role in stress response, particularly cold shock and is related to hibernation factors. It is hard to have a better understanding of these proteins' functions

without a deeper biologic understanding, but we think that this cluster plays a role in protecting E. Coli in cold environment.