732A51 Bioinformatics

Lab 5

Johannes Hedström, Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2024-12-13

Contents

1 Question 1

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.*

```
data <- read.table("ChG-Miner_miner-chem-gene.tsv", header = FALSE)
colnames(data) <- c("drug","gene")
# create graph
graph <- graph_from_data_frame(data, directed = FALSE)

total_nodes <- vcount(graph)
drug_nodes <- as.numeric(length(unique(data$drug)))
gene_nodes <- as.numeric(length(unique(data$gene)))
edges <- ecount(graph)

# strongest connected components
scc <- largest_component(graph)
nodes_scc <- vcount(scc)
nodes_fraction <- nodes_scc/total_nodes # not identical to bioSNAP
edges_scc <- ecount(scc)
edges_fraction <- edges_scc/edges

graph_diameter <- diameter(graph, directed = FALSE) # not identical to bioSNAP
graph_dist <- distances(graph) # get all distances
graph_dist <- as.vector(graph_dist[graph_dist < Inf]) # remove unconnected nodes
effective_diameter <- quantile(graph_dist, probs = 0.9) # not identical to bioSNAP

summary_stats <- rbind(total_nodes, drug_nodes, gene_nodes, edges, nodes_scc,
                       nodes_fraction, edges_scc, edges_fraction,
                       graph_diameter, effective_diameter)
```

Table 1: Summary statistics for chosen dataset

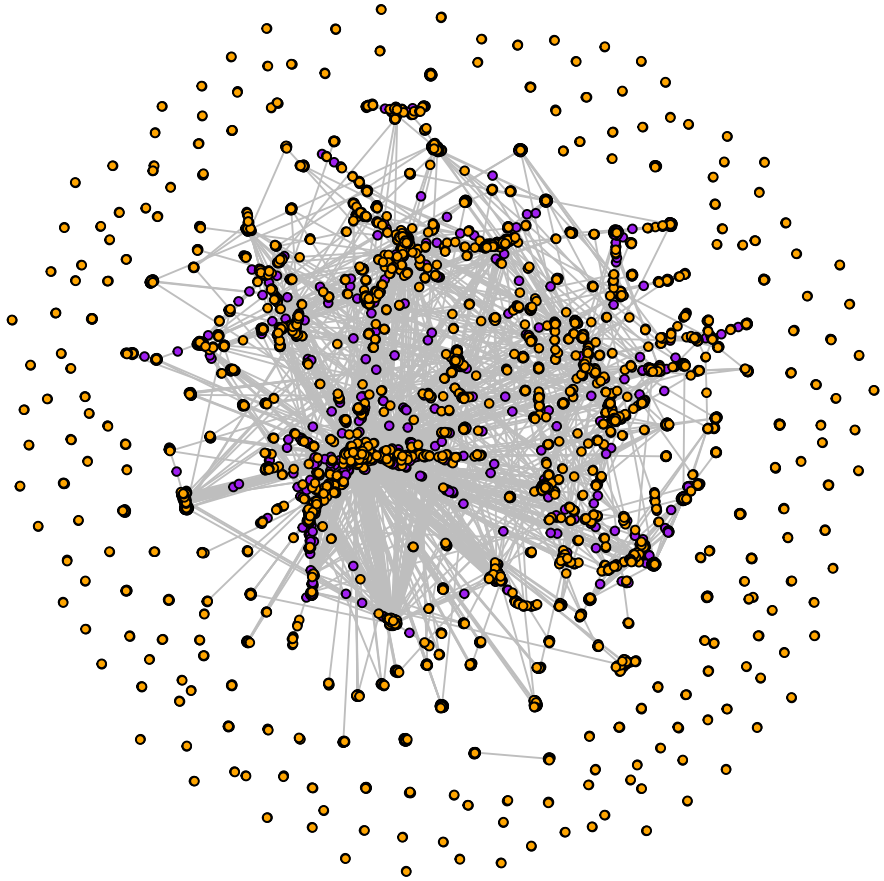
	Dataset statistics
Nodes	7341
Drug nodes	5017
Gene nodes	2324
Edges	15138
Nodes in largest SCC	6621
Fraction of nodes in largest SCC	0.901921
Edges in largest SCC	14581
Fraction of edges in largest SCC	0.963205
Diameter (longest shortest path)	18.000
90-percentile effective diameter	8.000

There are differences in the fraction of nodes in largest SCC, the longest shortest path and the 90-percentile effective diameter between the BioSNAP summary and the summary produced here. The differences are small, and the concrete statistics such as nodes and edges are the same, meaning the constructed graph is very similar but calculated differently. Most likely, the differences comes from choices of computation, such as the diameter being calculated with different algorithms which might handle edge cases in different ways. Also, the graph created here has been created as undirected, but perhaps extra information exists that could make it directed. The fraction of nodes in the largest SCC differ, while the fraction of edges in the largest SCC are identical, which means there is some difference in how the fraction is calculated in the BioSNAP database.

```r
set.seed(123456)
# set node types
V(graph)$type <- ifelse(V(graph)$name %in% data[,1], "Drug", "Gene")

# set colors for nodes
V(graph)$color <- ifelse(V(graph)$type == "Drug", "purple", "orange")

plot(
  graph,
  vertex.color = V(graph)$color,
  vertex.size = 2,
  vertex.label=NA,
  edge.color = "gray",
  edge.size = 0.01,
  main = "Network of drugs and genes"
)
```



The plotted graph above shows Drug nodes as purple and Gene nodes as orange. There is a large cluster in the center with a lot of close connectivity between groups, yet distances between nodes differ largely and it would probably be apt to create more clusters. Then there is a large ring outer circle of nodes that has no apparent edges connecting them. There are only Genes in the outer circle at least in this format, but there are also overlapping nodes. As there are a lot of nodes and edges there is hard to say anything regarding how nodes are connected and potential directions in the graph, which requires a more formal analysis.

For slightly more detail, here only the largest connected component is computed.

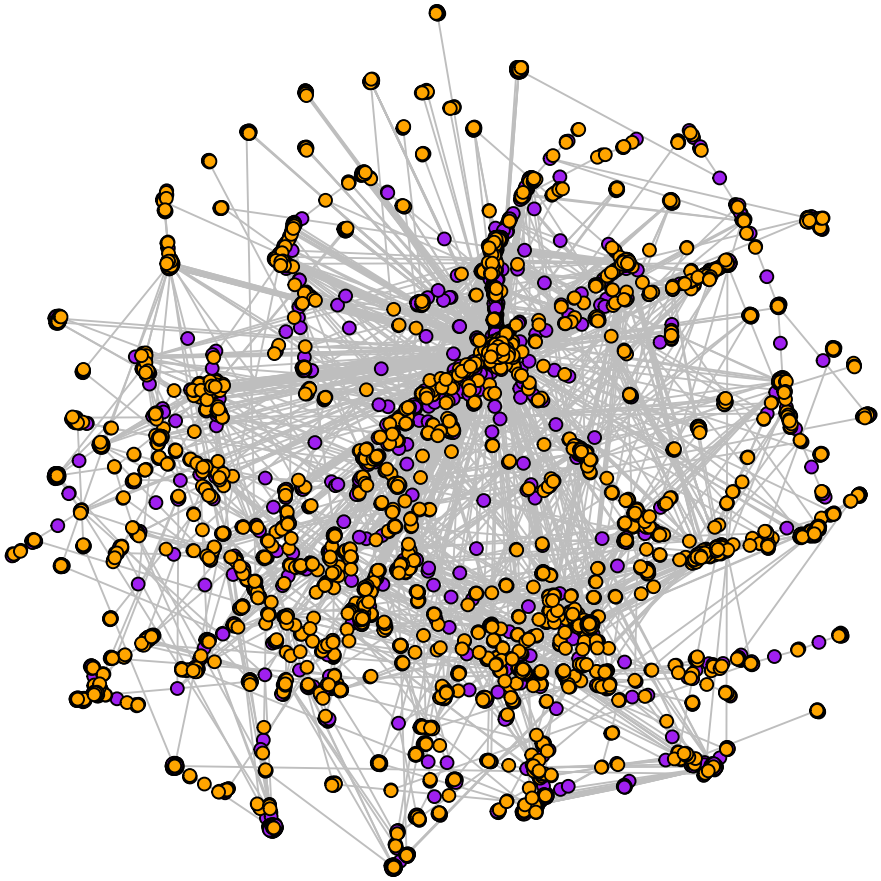
```r
set.seed(123456)
clusters <- components(graph) # find components

# largest component
largest_component <- induced_subgraph(graph, which(clusters$membership == which.max(clusters$csize)))

# set node types
V(largest_component)$type <- ifelse(V(largest_component)$name %in% data[,1], "Drug", "Gene")

# set colors for nodes
V(largest_component)$color <- ifelse(V(largest_component)$type == "Drug", "purple", "orange")

plot(
  largest_component,
  vertex.color = V(largest_component)$color,
  vertex.size = 3,
  vertex.label=NA,
  edge.color = "gray",
  edge.size = 0.01,
  main = "Network of largest connected drugs and genes"
)
```



There is still hard to decipher the network but shows a bit more structure, also a bit smaller for use of ease.

2. *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.*

To perform the cluster analysis the first step is to perform the Spinglass Algorithm from the linked webpage.

As the original graph isn't connected, which is a requirement for the algorithm, the largest component network will be used.

```
is_connected(graph)
```

```
## [1] FALSE
```

```
is_connected(largest_component)
```

```
## [1] TRUE
```

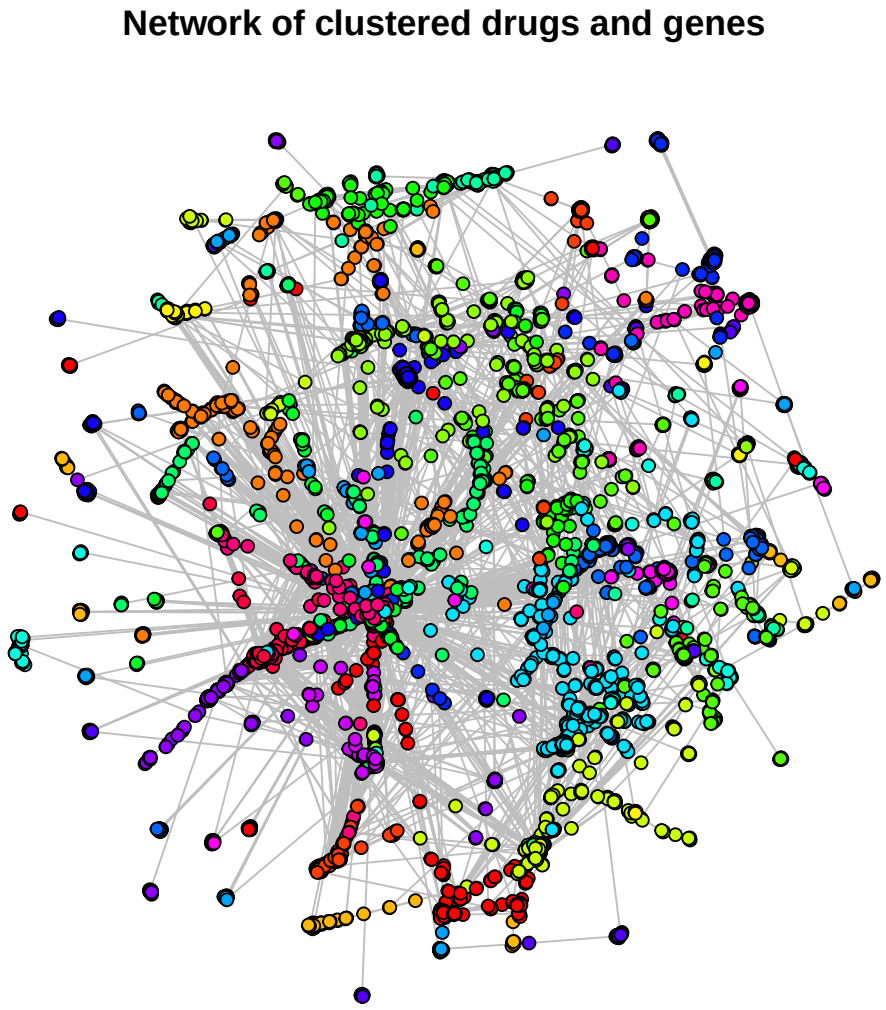
The guide mentions that the spinglass algorithm can differ between runs, but often provide very similar results to the walktrap algorithm which is fairly deterministic. Due to the size of the network, only a single run of the spinglass algorithm will be run for the sake of compute time.

```
set.seed(123456)
g <- largest_component
sgc <- cluster_spinglass(g)

# assign cluster memberships to vertices
V(g)$cluster <- sgc$membership

# assign colors to vertices depending on membership
colors <- rainbow(length(unique(sgc$membership)))
V(g)$color <- colors[V(g)$cluster]

plot(
  g,
  vertex.color = V(g)$color,
  vertex.size = 3,
  vertex.label=NA,
  edge.color = "gray",
  edge.size = 0.01,
  main = "Network of clustered drugs and genes"
)
```



As there are 25 different clusters which also are not very separated in the graph as they are highly connected to each other, some further subsampling will be performed.

```
cluster_sizes <- table(sgc$membership)
cluster_sizes
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
```

```
## 235 235 323 147 124 373 305 385 237 132 259 218 191 507 133 280 166 269 103 190
##  21  22  23  24  25
## 224 141 211 788 445
```

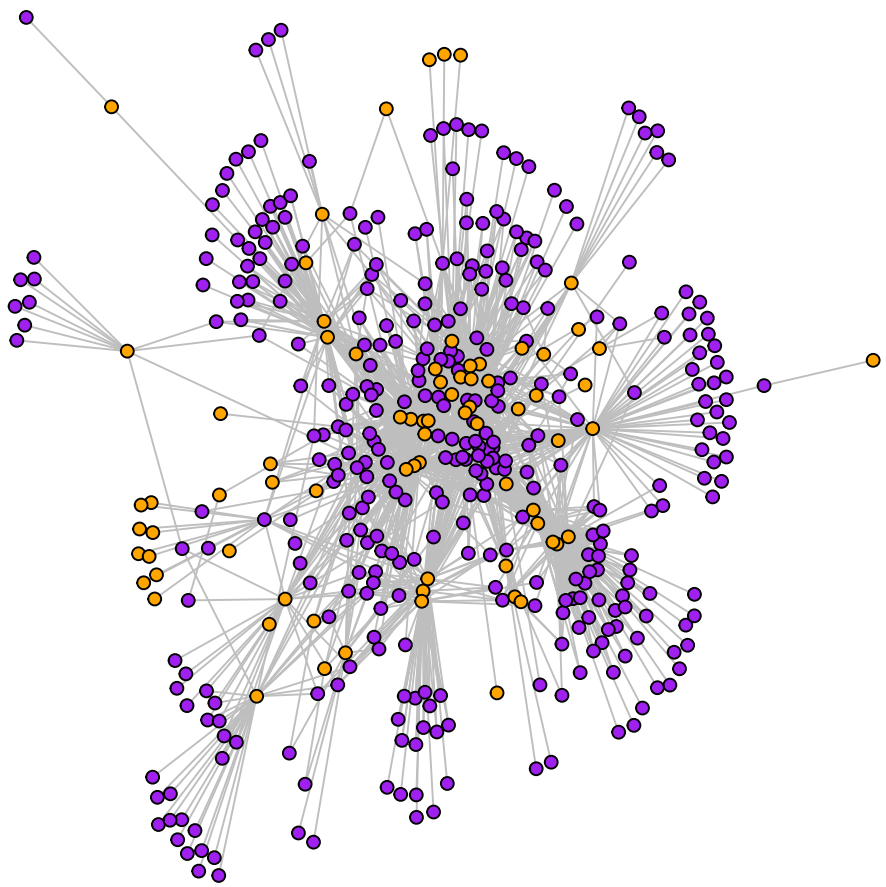
Let's take a look at the biggest cluster

```
cluster25 <- V(g)[sgc$membership == 25]
cluster25_graph <- induced_subgraph(g, cluster25)

# set node types
V(cluster25_graph)$type <- ifelse(V(cluster25_graph)$name %in% data[,1], "Drug", "Gene")

# set colors for nodes
V(cluster25_graph)$color <- ifelse(V(cluster25_graph)$type == "Drug", "purple", "orange")


plot(
  cluster25_graph,
  #vertex.label = V(cluster25_graph)$name,
  vertex.size = 3,
  vertex.label=NA,
  edge.color = "gray",
  edge.size = 0.01,
  main = "Network of biggest cluster of drugs and genes"
)
```



Cluster 25 is plotted above. Again, Drug nodes are colored purple and Gene nodes are colored orange. There is a big amount of drug nodes in this cluster compared to genes. Below follows the associated labels with the cluster.

```
cluster25_vec <- V(cluster25_graph)$name
cluster25_vec
```

```
##   [1] "DB01242" "DB01238" "DB01151" "DB00734" "DB01221" "DB00988" "DB00373"
```

```
##   [8] "DB01247" "DB05079" "DB05280" "DB00477" "DB01454" "DB01224" "DB01365"
##  [15] "DB00696" "DB01085" "DB06678" "DB00458" "DB00182" "DB00934" "DB00575"
##  [22] "DB06016" "DB01403" "DB00852" "DB00670" "DB01102" "DB00268" "DB06216"
##  [29] "DB00411" "DB04820" "DB01622" "DB00543" "DB00830" "DB00598" "DB01620"
##  [36] "DB00408" "DB04836" "DB00540" "DB04855" "DB00729" "DB00952" "DB09082"
##  [43] "DB00767" "DB00195" "DB01142" "DB00496" "DB00835" "DB08800" "DB01392"
##  [50] "DB01363" "DB06096" "DB00321" "DB00340" "DB00489" "DB01162" "DB05816"
##  [57] "DB01267" "DB00476" "DB01149" "DB05469" "DB00831" "DB00484" "DB06707"
##  [64] "DB05461" "DB03907" "DB00692" "DB00589" "DB00248" "DB01136" "DB04946"
##  [71] "DB00413" "DB01231" "DB00454" "DB00933" "DB00725" "DB00334" "DB01591"
##  [78] "DB00342" "DB01186" "DB00726" "DB01359" "DB00363" "DB00368" "DB00370"
##  [85] "DB00246" "DB06282" "DB00968" "DB01200" "DB08897" "DB01288" "DB00612"
##  [92] "DB09016" "DB01577" "DB06694" "DB05849" "DB00960" "DB06726" "DB04829"
##  [99] "DB06148" "DB01038" "DB00942" "DB05752" "DB01239" "DB00667" "DB01576"
## [106] "DB06714" "DB00420" "DB00751" "DB00221" "DB00935" "DB04832" "DB01625"
## [113] "DB00728" "DB06701" "DB00574" "DB00804" "DB00457" "DB00409" "DB01063"
## [120] "DB00715" "DB08927" "DB05766" "DB01235" "DB07513" "DB04844" "DB00777"
## [127] "DB05271" "DB04884" "DB00315" "DB00656" "DB01366" "DB00747" "DB01472"
## [134] "DB00264" "DB00320" "DB05743" "DB02095" "DB00247" "DB00953" "DB00699"
## [141] "DB04861" "DB04818" "DB08807" "DB00202" "DB00800" "DB00679" "DB01407"
## [148] "DB01579" "DB01114" "DB00366" "DB01036" "DB04843" "DB01624" "DB00723"
## [155] "DB00191" "DB00392" "DB00376" "DB07641" "DB03336" "DB05509" "DB00998"
## [162] "DB00572" "DB01210" "DB01615" "DB00332" "DB01106" "DB04821" "DB01626"
## [169] "DB01580" "DB00346" "DB01442" "DB00508" "DB00397" "DB00714" "DB08815"
## [176] "DB01255" "DB04850" "DB01488" "DB00865" "DB00809" "DB00985" "DB00422"
## [183] "DB01084" "DB08810" "DB04846" "DB00927" "DB04842" "DB01069" "DB00521"
## [190] "DB01105" "DB06207" "DB05687" "DB05492" "DB05049" "DB09014" "DB01253"
## [197] "DB00668" "DB02509" "DB01193" "DB00719" "DB07512" "DB08516" "DB09202"
## [204] "DB01623" "DB01618" "DB01409" "DB06262" "DB00629" "DB05080" "DB00211"
## [211] "DB00925" "DB06706" "DB04924" "DB01175" "DB01463" "DB06698" "DB04889"
## [218] "DB00450" "DB00217" "DB00280" "DB00986" "DB08799" "DB06711" "DB08806"
## [225] "DB00517" "DB00940" "DB01364" "DB00462" "DB00505" "DB00590" "DB01237"
## [232] "DB00204" "DB06077" "DB01170" "DB08049" "DB04888" "DB00669" "DB00323"
## [239] "DB00216" "DB08804" "DB09076" "DB01001" "DB04857" "DB06702" "DB00298"
## [246] "DB01291" "DB04890" "DB06709" "DB05339" "DB08805" "DB03894" "DB05892"
## [253] "DB00601" "DB06288" "DB06144" "DB07919" "DB00805" "DB00797" "DB00372"
## [260] "DB01621" "DB04948" "DB00391" "DB00816" "DB05316" "DB08801" "DB00614"
## [267] "DB09128" "DB00866" "DB05032" "DB04307" "DB05227" "DB01295" "DB06764"
## [274] "DB00245" "DB00424" "DB00494" "DB00557" "DB00434" "DB05395" "DB00344"
## [281] "DB08480" "DB08176" "DB01297" "DB00579" "DB01445" "DB00706" "DB01214"
## [288] "DB01146" "DB09207" "DB06691" "DB00875" "DB00915" "DB00427" "DB01019"
## [295] "DB04885" "DB05026" "DB07543" "DB00209" "DB01619" "DB00785" "DB01614"
## [302] "DB00449" "DB04970" "DB08082" "DB06623" "DB00226" "DB00901" "DB01148"
## [309] "DB01089" "DB00260" "DB00353" "DB05042" "DB04677" "DB00335" "DB01425"
## [316] "DB00388" "DB00964" "DB02211" "DB04837" "DB05369" "DB00950" "DB00937"
## [323] "DB06412" "DB00697" "DB00737" "DB05205" "DB00206" "DB06700" "DB08802"
## [330] "DB00841" "DB01608" "DB02105" "DB05381" "DB00187" "DB01203" "DB00902"
## [337] "DB05432" "DB00610" "DB05012" "DB05422" "DB00386" "DB00190" "DB00383"
## [344] "DB01616" "DB06766" "DB01246" "DB05307" "DB04908" "DB09018" "DB00387"
```

```
## [351] "DB04947" "DB05152" "DB00792" "DB00782" "DB08808" "DB05419" "DB00272"
## [358] "DB00219" "DB05688" "DB00405" "DB00354" "DB04365" "DB00341" "DB08347"
## [365] "DB07462" "DB00771" "DB04017" "DB05345" "DB08936" "P23975"  "P08173"
## [372] "P08913"  "P08588"  "P21397"  "P35367"  "P35462"  "P17405"  "P31645"
## [379] "P35368"  "P25100"  "Q9UK17"  "P08172"  "P28335"  "P18089"  "P08908"
## [386] "P28223"  "P07550"  "P11229"  "P13945"  "P28221"  "P27338"  "P35348"
## [393] "P18825"  "Q01959"  "P46098"  "P08912"  "O95069"  "P20309"  "P30939"
## [400] "P28222"  "P47898"  "P21728"  "P21964"  "P34969"  "P41595"  "P14416"
## [407] "P50406"  "Q9H3N8"  "P21917"  "P21918"  "P15336"  "P20711"  "Q12809"
## [414] "Q8WYK2"  "P25021"  "Q05940"  "Q9Y2D1"  "Q99720"  "Q9H252"  "P54219"
## [421] "P28566"  "Q9NZV8"  "Q99712"  "Q9Y5N1"  "Q9NYX4"  "Q9UNX9"  "P41146"
## [428] "O95665"  "P01138"  "P18850"  "O00264"  "P26447"  "P18847"  "Q9NS40"
## [435] "P16860"  "P18846"  "P18848"  "Q14500"  "P78508"  "Q96RJ0"  "P17544"
## [442] "Q16620"  "P17302"  "Q16568"  "O95644"
```

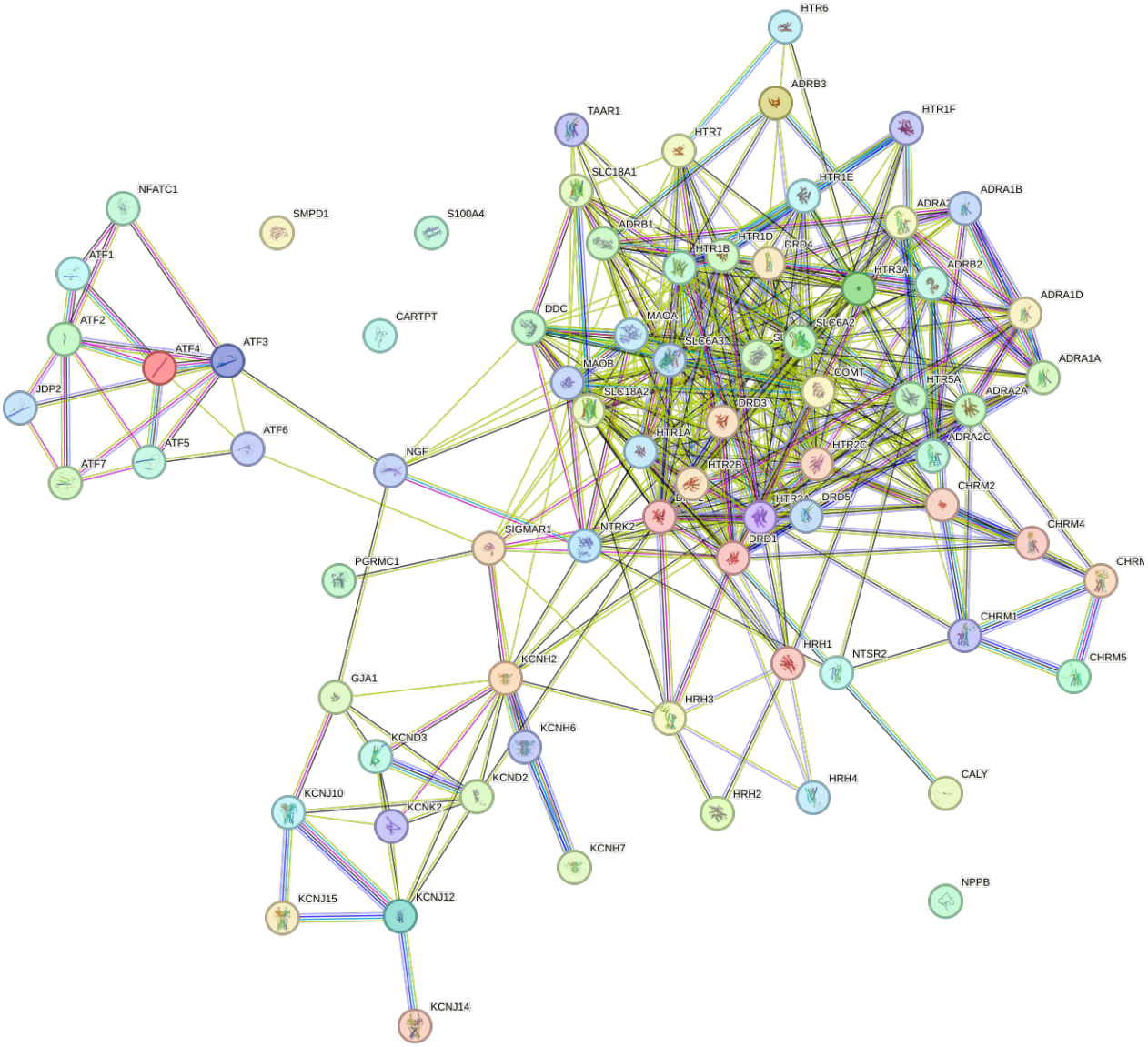
To analyze this, all proteins will be ran in the STRING database (https://string-db.org/).

First, extract all proteins into a list. This file is uploaded on the "multiple proteins" section on STRING (https://string-db.org/cgi/input?sessionId=bWLThyUmK3Hr&input_page_active_form=multiple_identifiers)

```
genes <- cluster25_vec[!grepl("^D", cluster25_vec)]

#writeLines(genes, "cluster25_genes.txt")
```

The proteins are linked as follows,



Out of the 75 gene proteins, 35 are mentioned in a paper "Functional characterization of G-protein-coupled receptors: a bioinformatics approach" by Tovo-Rodrigues L, Roux A, Hutz MH, Rohde LA, Woods AS where they discuss how G protein-coupled receptors (GPCRs) involved in synaptic transmission have flexible regions that allow them to interact with other proteins and adapt to different functions. These regions are especially common in certain parts of the receptors and contain patterns important for protein regulation and forming connections. This flexibility and adaptability may play a key role in how the nervous system functions and adjusts to changes.

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 E. coli data (Schäfer and Strimmer 2005)

```
# Load GeneNet package

library("GeneNet")

data(ecoli)
dim(ecoli)
```

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

Loading tha dataset and its longitude data, with 120 variables measured at 9 different timepoints.

```
#'
#' # Estimation of partial correlations

#' Estimate matrix of partial correlation using a shrinkage estimator:
pc = ggm.estimate.pcor(ecoli)
```

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

```
dim(pc)
```

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

This step estimates the partial correlation for all pairs of genes in the dataset. The output will be a square matrix of partial correlations.

```
#' 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
```

We use the network.test.edges function to assign p-values, q-values, and posterior probabilities to the potential edges in the network and save in a table

```
#' 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
```

This prints a lists the edges with the top 5 highest partial correlation value.

```
#'
#' # Decide which edges to include in the network

#' 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
```

When using a local cutoff of 0.2 we get 125 significant edges, which is 2.43% of all possible in the network. The number of significant directions in the network is 17, this suggests that of all the potential directional relationships in the selected 125 significant edges, 13.6% were deemed significant

```
#' 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
```

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

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

Include edges with a posterior probability of at least 0.9.

```
#' 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
```

Instead of having a specific cutoff, use a max number of edges, number in in this example is 70.

```r
#'
#' 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
```

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

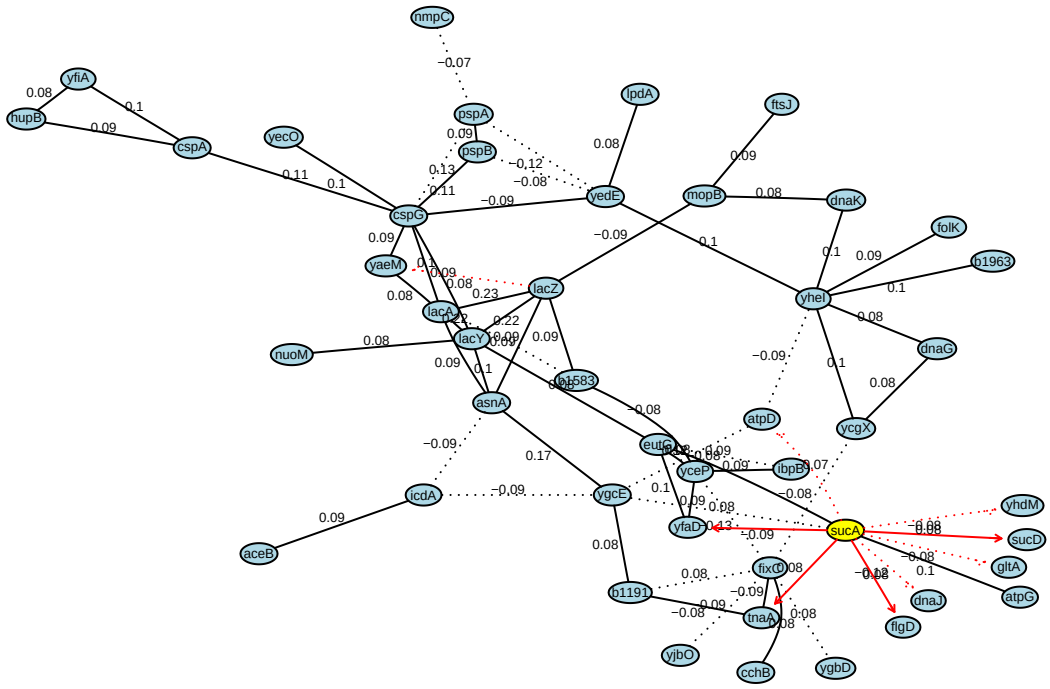
```
##  sucA  cspG  fixC  yheI  lacA  lacY  lacZ  asnA  eutG  yceP  yedE  ygcE  pspA
##    11     8     7     7     6     6     6     5     5     5     5     5     4
##  atpD b1191 b1583  cspA  icdA  mopB  pspB  tnaA  yaeM  ycgX  yfaD  dnaG  dnaK
##     3     3     3     3     3     3     3     3     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
```

```r
#' 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")

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
```

```
#+ fig.width=8, fig.height=7
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



Visualizing the extracted network, to explore the structure, relationships and patterns between the genes.

The gene sucA is highlighted and it seems important as its the only gene that connects to some(sucD, atpG etc.) of the other genes in the network. It might have a significant biological relevance like being a hub gene or highly connected gene.

2.1.1 Varying the criterion

```
ecoli.net = extract.network(ecoli.edges,method.ggm="qval", cutoff.ggm=0.02, cutoff.dir=0.9)
```

```
##
## Significant edges:  48
##     Corresponding to  0.93 %  of possible edges
##
## Significant directions:  269
##     Corresponding to  5.22 %  of possible directions
## Significant directions in the network:  3
##     Corresponding to  6.25 %  of possible directions in the network
```

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

```
## [1] 48 11
```

Changing the criterion for selecting significant partial correlations and significant directions, now going by q-value instead. Only picking edges with a q-value lower than 0.02.

Now we only get 48 significant edges and 3 of them are directed in the network.

```
#'
#' 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
##       3      45
```

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

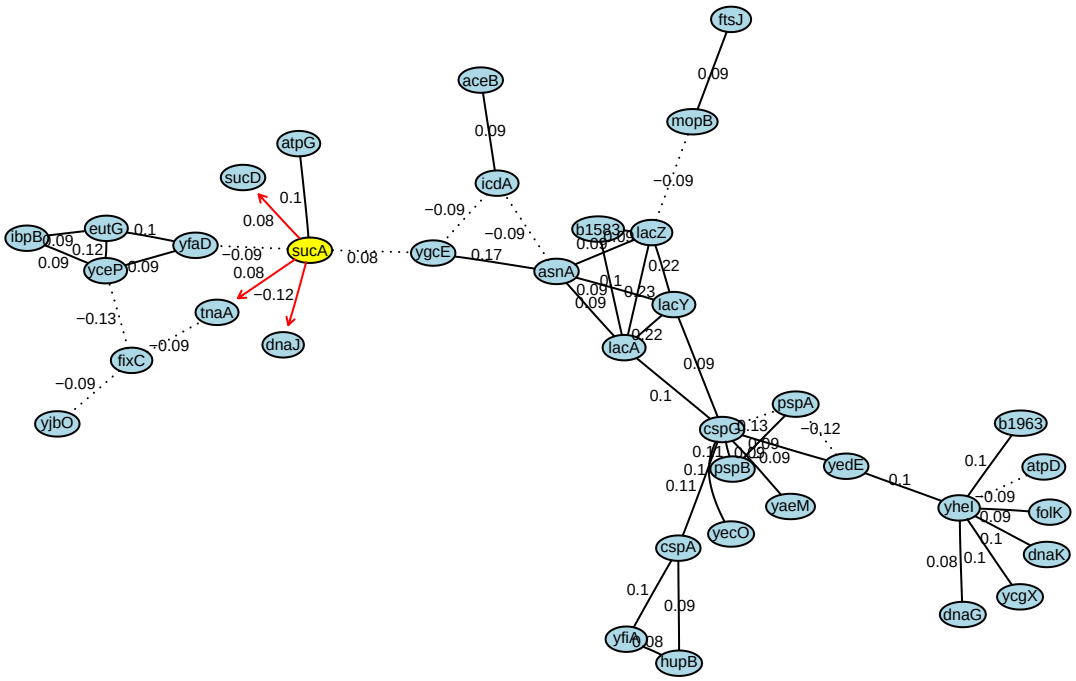
```
##  cspG  yheI  sucA  asnA  lacA  lacZ  lacY  yceP  cspA  eutG  fixC  icdA  pspA
##     8     7     6     5     5     5     4     4     3     3     3     3     3
##  yedE  yfaD  ygcE b1583  hupB  ibpB  mopB  pspB  tnaA  yfiA  aceB  atpD  atpG
##     3     3     3     2     2     2     2     2     2     2     1     1     1
## b1963  dnaG  dnaJ  dnaK  folK  ftsJ  sucD  yaeM  ycgX  yecO  yjbO
##     1     1     1     1     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")

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
```

```
#+ fig.width=8, fig.height=7
plot(gr, attrs = globalAttrs, nodeAttrs = nodeAttrs, edgeAttrs = edgeAttrs, "fdp")
```



asnA, lacY, lacA,b1583, lacZ

A cluster here could be asnA, lacY, lacA, b1583, lacZ as they are well connected to each other but no that much to other genes, only positive partial correlations within and half of the edges to other genes outside the cluster have negative partial correlation.

2.1.2 Cluster

The lac operon of three genes, lacY, lacA and lacZ, which is an operon that is required for the transport of metabolism of lactose in E-coli, so for these three to be in the same cluster seems logical.

Asparagine synthetase A (AsnA) is an enzyme in E. coli that helps produce the amino acid asparagine. It does this by converting another amino acid, aspartate, into asparagine, using ammonia as part of the reaction. So its metabolic but not in the same way as the lac operons and E-coli have lots of genes with metabolic functions, but the function to produce asparagine is very specific to Asna.

Didn't manage to find anything that relates b1583(ynfB) to the other genes in the cluster.