

BCB570 Assignment 3

Ashish Jain

Question 1:

i).

Network Details

```
library("tidyverse")
library("igraph")
library("curl")

graphDetails <- function(edgeFile, sep) {
  graph <- as.matrix(read.table((curl(edgeFile)), sep = sep))
  graphGraph1 <- graph.data.frame(graph, directed = FALSE)
  graphGraph <- simplify(graphGraph1, remove.multiple = TRUE,
    remove.loops = TRUE, edge.attr.comb = igraph_opt("edge.attr.comb"))
  graphDetails <- c(vcount(graphGraph), ecount(graphGraph),
    graph.density(graphGraph), diameter(graphGraph), radius(graphGraph),
    transitivity(graphGraph, type = "average"), transitivity(graphGraph,
    type = "global"), mean_distance(graphGraph))
  return(list(graphDetails, graphGraph))
}

path <- "https://raw.githubusercontent.com/ashishjain1988/BCB570/master/HW3/"
outputY2H_uniondata <- graphDetails(paste0(path, "Y2H_uniondata.txt"),
  "\t")
outputCCSB_YI1data <- graphDetails(paste0(path, "CCSB_YI1.txt"),
  "\t")
essentialGenes <- read.table(paste0(path, "essentialGenes.txt"))

details <- c("Number of Nodes", "Number of Edges", "Graph Density",
  "Diameter", "Radius", "Average Clustering Coefficient", "Global Clustering Coefficient",
  "Average Shortest Path")

table <- data.frame(Properties = details, `Y2H-Union` = outputY2H_uniondata[[1]],
  `CCSB-YI1` = outputCCSB_YI1data[[1]])
table %>% knitr::kable(caption = "PPI Details of the Network")
```

Table 1: PPI Details of the Network

Properties	Y2H.Union	CCSB.YI1
Number of Nodes	2018.0000000	1278.0000000
Number of Edges	2705.0000000	1641.0000000
Graph Density	0.0013291	0.0020110
Diameter	14.0000000	14.0000000
Radius	0.0000000	0.0000000
Average Clustering Coefficient	0.0970026	0.1059977
Global Clustering Coefficient	0.0236142	0.0206502

Properties	Y2H.Union	CCSB.YI1
Average Shortest Path	5.6109291	5.3641783

```

graphUnion <- outputY2H_uniondata[[2]]
graphCCSB <- outputCCSB_YI1data[[2]]

gdetails <- function(graphGraph1) {
  graphGraph <- simplify(graphGraph1, remove.multiple = TRUE,
    remove.loops = TRUE, edge.attr.comb = igraph_opt("edge.attr.comb"))
  graphDetails <- c(vcount(graphGraph), ecount(graphGraph),
    graph.density(graphGraph), diameter(graphGraph), radius(graphGraph),
    transitivity(graphGraph, type = "average"), transitivity(graphGraph,
      type = "global"), mean_distance(graphGraph))
  return(graphDetails)
}

largestSubNet <- function(graph) {
  dGraph <- decompose.graph(graph)
  lgraph <- dGraph[[1]]
  for (i in 1:length(dGraph)) {
    if (vcount(lgraph) < vcount(dGraph[[i]])) {
      lgraph <- dGraph[[i]]
    }
  }
  return(lgraph)
}

lgraphUnion <- largestSubNet(graphUnion)
lgraphCCSB <- largestSubNet(graphCCSB)

table <- data.frame(Properties = details, `Y2H-Union` = gdetails(lgraphUnion),
  `CCSB-YI1` = gdetails(lgraphCCSB))
table %>% knitr::kable(caption = "PPI Details of largest connected sub-network")

```

Table 2: PPI Details of largest connected sub-network

Properties	Y2H.Union	CCSB.YI1
Number of Nodes	1647.0000000	964.0000000
Number of Edges	2518.0000000	1487.0000000
Graph Density	0.0018576	0.0032036
Diameter	14.0000000	14.0000000
Radius	8.0000000	8.0000000
Average Clustering Coefficient	0.1023266	0.1071888
Global Clustering Coefficient	0.0236669	0.0205214
Average Shortest Path	5.6117474	5.3659488

Small World Network Test

```

getRandomNetworkDetails <- function(graph) {
  randomNetwork <- erdos.renyi.game(vcount(graph), ecount(graph),
    type = "gnm")
  output1 <- average.path.length(graph)
  output2 <- average.path.length(randomNetwork)
  output3 <- transitivity(graph, type = "global")
  output4 <- transitivity(randomNetwork, type = "global")
  return(c(output1, output2, output3, output4))
}

graphUnionProp <- graphUnion %>% getRandomNetworkDetails()
graphCCSBProp <- graphCCSB %>% getRandomNetworkDetails()

details <- c("Average Path Length-Graph", "Average Path Length-Random Graph",
  "Clustering Coefficient-Graph", "Clustering Coefficient-Random Graph")

table <- data.frame(Features = details, `Y2H-Union` = graphUnionProp,
  `CCSB-YI1` = graphCCSBProp)
table %>% knitr::kable(caption = "Small-World Property Test of the Network")

```

Table 3: Small-World Property Test of the Network

Features	Y2H.Union	CCSB.YI1
Average Path Length-Graph	5.6109291	5.3641783
Average Path Length-Random Graph	7.5068465	7.4088676
Clustering Coefficient-Graph	0.0236142	0.0206502
Clustering Coefficient-Random Graph	0.0024858	0.0021692

In order to test the small worldness of a network, we can use the concept described in **Watts and Strogatz's** paper. The idea is to compare the average path length and clustering coefficients with that of a random network with the same number of nodes and vertices. According to them, small world network satisfies the following conditions:

$$\begin{aligned}
(Clustering\ Coefficient)_{Network} &\gg (Clustering\ Coefficient)_{RandomNetwork} \\
(Average\ Path\ Length)_{Network} &\geq (Average\ Path\ Length)_{RandomNetwork}
\end{aligned}$$

From table 3, it can be seen that none of the networks satisfies the small-world network properties. Although, both the networks have very large clustering coefficient compared to the random network with the same number of nodes but the average path length is less than that of the random network.

```

table <- data.frame(Features = details, `Y2H-Union` = getRandomNetworkDetails(lgraphUnion),
  `CCSB-YI1` = getRandomNetworkDetails(lgraphCCSB))
table %>% knitr::kable(caption = "Small-World Property Test for largest connected sub-network")

```

Table 4: Small-World Property Test for largest connected sub-network

Features	Y2H.Union	CCSB.YI1
Average Path Length-Graph	5.6117474	5.3659488

Features	Y2H.Union	CCSB.YI1
Average Path Length-Random Graph	6.5797516	6.0224649
Clustering Coefficient-Graph	0.0236669	0.0205214
Clustering Coefficient-Random Graph	0.0023253	0.0045142

From table 4, it can be seen that none of the largest sub-network satisfies the small-world network properties. Although, both the networks have very large clustering coefficient compared to the random network with same number of nodes but the average path length is less than that of the random network.

Scale Free Network Test

```
## Power law fit
fitUnion <- power.law.fit(degree(graphUnion), 9, implementation = "plfit")
fitCCSB <- power.law.fit(degree(graphCCSB), 9, implementation = "plfit")

details <- c("Y2H-Union", "CCSB-YI1")

table <- data.frame(Graph = details, Alpha = c(fitUnion$alpha,
  fitCCSB$alpha), `X-min` = c(fitUnion$xmin, fitCCSB$xmin),
  `P-Value` = c(fitUnion$KS.p, fitCCSB$KS.p))
table %>% knitr::kable(caption = "Power Law Fit Details of the Network")
```

Table 5: Power Law Fit Details of the Network

Graph	Alpha	X.min	P.Value
Y2H-Union	2.816046	9	0.9999058
CCSB-YI1	2.630039	9	0.9782982

From table 5, it can be seen that Y2H-Union and CCSB-YI network has P-Value greater than 0.05 which means that we are not able to reject the null hypothesis that the original data could have been drawn from the fitted power-law distribution. The hubs in the graphs are also significant as the α is between 2 and 3 in both the cases. From this we can conclude that this network is scale free.

```
## Power law fit
fitUnion <- power.law.fit(degree(lgraphUnion), 9, implementation = "plfit")
fitCCSB <- power.law.fit(degree(lgraphCCSB), 9, implementation = "plfit")

table <- data.frame(Graph = details, Alpha = c(fitUnion$alpha,
  fitCCSB$alpha), `X-min` = c(fitUnion$xmin, fitCCSB$xmin),
  `P-Value` = c(fitUnion$KS.p, fitCCSB$KS.p))
table %>% knitr::kable(caption = "Power Law Fit Details of largest connected sub-network")
```

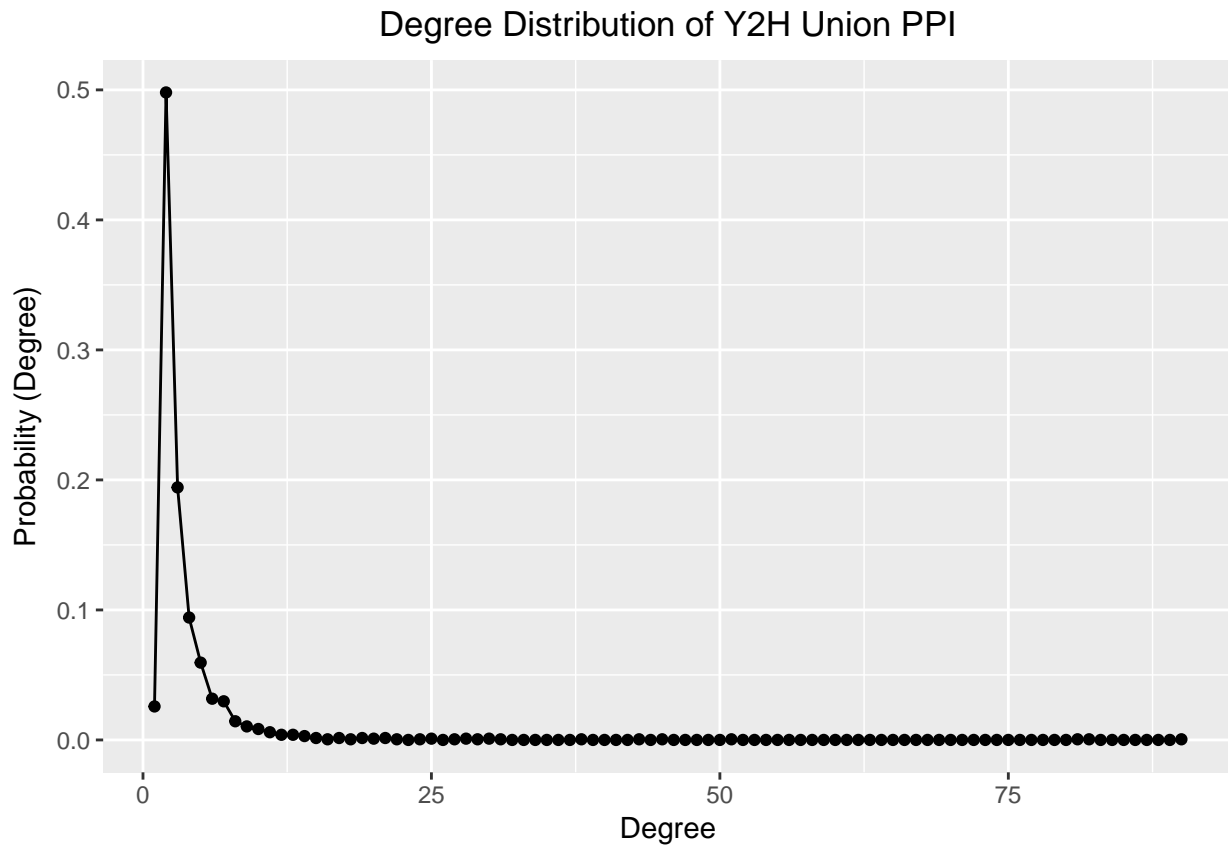
Table 6: Power Law Fit Details of largest connected sub-network

Graph	Alpha	X.min	P.Value
Y2H-Union	2.816046	9	0.9999058
CCSB-YI1	2.630039	9	0.9782982

Also from table 6, it can be seen that the largest subnetwork of Y2H-Union and CCSB-YI1 networks has P-Value greater than 0.05 which means that we are not able to reject the null hypothesis that the original data could have been drawn from the fitted power-law distribution. The hubs in the graphs are also significant as the α is between 2 and 3 in both the cases. From this we can conclude that this network is scale free.

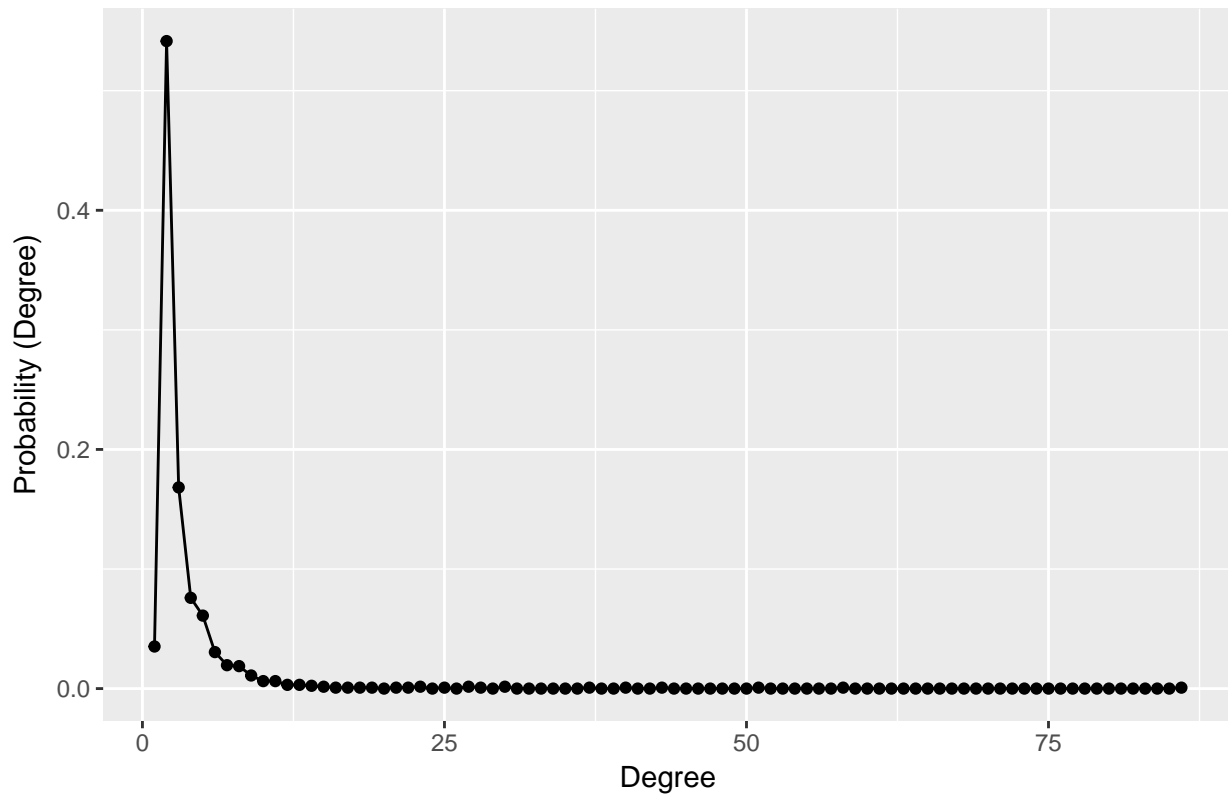
ii). Finding the hubs based on degree centrality

```
data <- data.frame(prob = degree.distribution(graphUnion),
  degree = c(1:length(degree.distribution(graphUnion))))
data %>% ggplot(aes(x = degree, y = prob)) + geom_point() +
  geom_line() + labs(x = "Degree", y = "Probability (Degree)",
  title = "Degree Distribution of Y2H Union PPI") +
  theme(plot.title = element_text(hjust = 0.5))
```



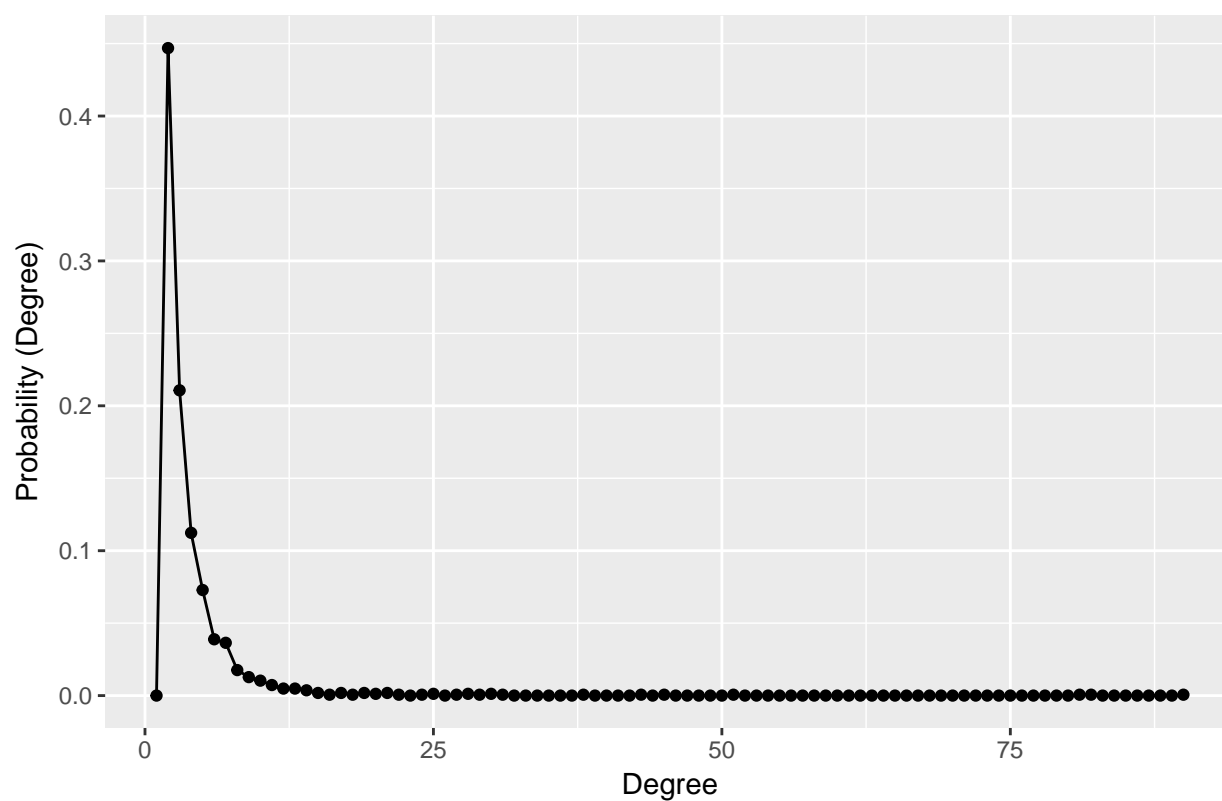
```
data <- data.frame(prob = degree.distribution(graphCCSB),
  degree = c(1:length(degree.distribution(graphCCSB))))
data %>% ggplot(aes(x = degree, y = prob)) + geom_point() +
  geom_line() + labs(x = "Degree", y = "Probability (Degree)",
  title = "Degree Distribution of CCSB YI1 PPI") +
  theme(plot.title = element_text(hjust = 0.5))
```

Degree Distribution of CCSB Y11 PPI



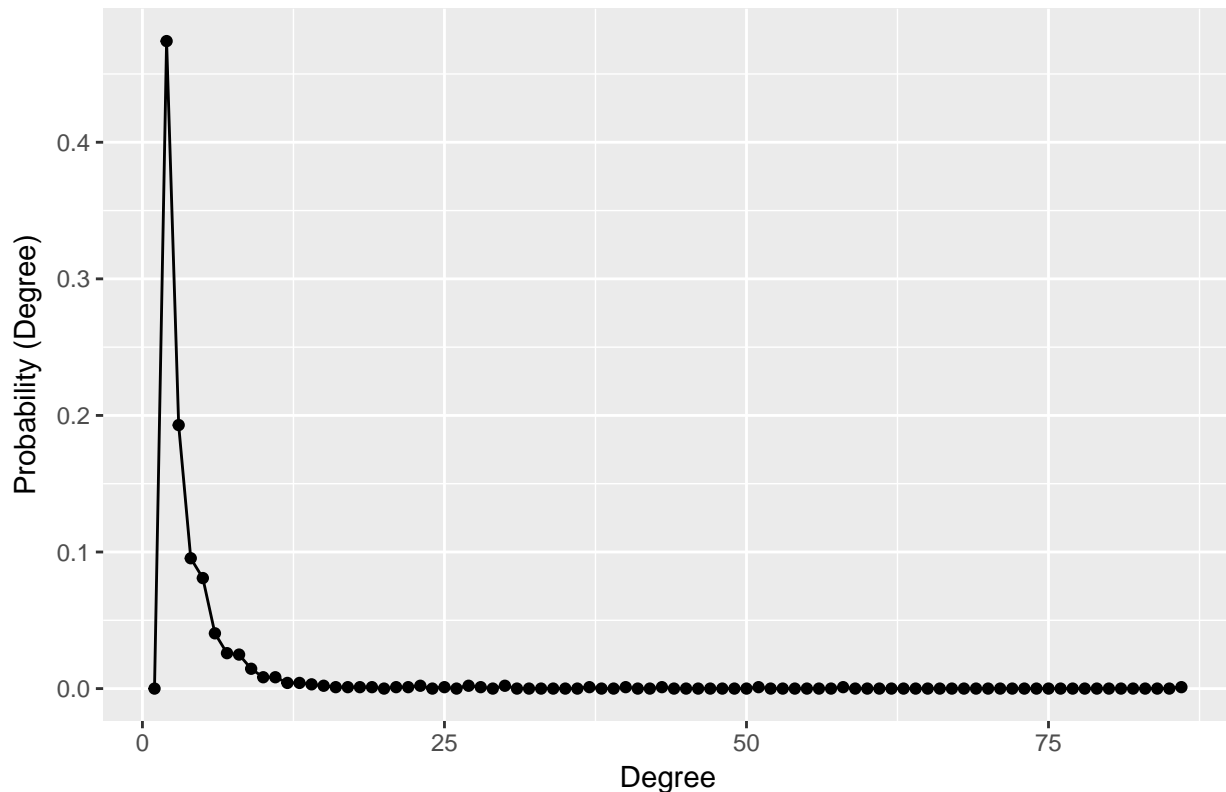
```
data <- data.frame(prob = degree.distribution(lgraphUnion),
  degree = c(1:length(degree.distribution(lgraphUnion))))
data %>% ggplot(aes(x = degree, y = prob)) + geom_point() +
  geom_line() + labs(x = "Degree", y = "Probability (Degree)",
  title = "Degree Distribution of Y2H Union PPI largest connected sub-network") +
  theme(plot.title = element_text(hjust = 0.5))
```

Degree Distribution of Y2H Union PPI largest connected sub-network



```
data <- data.frame(prob = degree.distribution(lgraphCCSB),
  degree = c(1:length(degree.distribution(lgraphCCSB))))
data %>% ggplot(aes(x = degree, y = prob)) + geom_point() +
  geom_line() + labs(x = "Degree", y = "Probability (Degree)",
  title = "Degree Distribution of CCSB YI1 PPI of largest connected sub-network") +
  theme(plot.title = element_text(hjust = 0.5))
```

Degree Distribution of CCSB YI1 PPI of largest connected sub-network



From the degree distribution plot we define hubs to be those genes which have degree greater than or equal to 8.

Intersection of the essential genes and hubs in whole network

```
hubNodesUnion <- names(which(igraph::degree(graphUnion) >= 8,
  arr.ind = T))
hubNodesCCSB <- names(which(igraph::degree(graphCCSB) >= 8, arr.ind = T))
length(hubNodesUnion)
```

```
## [1] 106
```

```
length(hubNodesCCSB)
```

```
## [1] 63
```

```
details <- c("Y2H-Union", "CCSB-YI1")
table <- data.frame(Graph = details, Overlap = c(toString(intersect(hubNodesUnion,
  as.vector(essentialGenes[, 1]))), toString(intersect(hubNodesCCSB,
  as.vector(essentialGenes[, 1])))))
table %>% knitr::kable(caption = "Essential genes overlap with hub genes of the PPI Network")
```


Table 7: Essential genes overlap with hub genes of the PPI Network

Graph	Overlap
Y2H-Union	YDR388W, YGL115W, YLR423C, YPL031C
CCSB-YI1	YPL031C, YLR423C

Intersection of the essential genes and hubs in the largest connected sub-network

```

hubNodesUnion <- names(which(igraph::degree(lgraphUnion) >=
  8, arr.ind = T))
hubNodesCCSB <- names(which(igraph::degree(lgraphCCSB) >= 8,
  arr.ind = T))
length(hubNodesUnion)

## [1] 106

length(hubNodesCCSB)

## [1] 63

# rhubNodesUnion<-rep(intersect(sample(V(lgraphUnion),
# length(hubNodesUnion)),
# as.vector(essentialGenes[,1])),100)

table <- data.frame(Graph = details, Overlap = c(toString(intersect(hubNodesUnion,
  as.vector(essentialGenes[, 1]))), toString(intersect(hubNodesCCSB,
  as.vector(essentialGenes[, 1])))))
table %>% knitr::kable(caption = "Essential genes overlap with hub genes of
  largest connected sub-network")

```

Table 8: Essential genes overlap with hub genes of largest connected sub-network

Graph	Overlap
Y2H-Union	YDR388W, YGL115W, YLR423C, YPL031C
CCSB-YI1	YPL031C, YLR423C

As seen from the above results, only four essential genes overlapped with the hub genes based on the degree distribution. These genes includes YDR388W (8), YGL115W (8), YLR423C (80), and YPL031C (12). Out of these YLR423C, and YPL031C genes found in both the PPI networks as well as in the essential genes. The biological functions of these genes are as following.

- **YDR388W**: The gene translates to a calmodulin-binding actin-associated protein. It play a significant role in endocytic membrane tabulation and constriction, and exocytosis. It also contributes to cell viability following starvation or osmotic stress.
- **YGL115W**: This gene activates glucose-repressed genes, represses glucose-induced genes and play an important role in sporulation, and peroxisome biogenesis.

- **YLR423C**: It is involved in the phagophore assembly site organization. It also acts as a regulatory subunit of an autophagy-specific complex that includes Atg1p and Atg13p. In addition to that, it also stimulates Atg1p kinase activity.
- **YPL031C**: It is involved in the regulation of the cellular response to nutrient levels and environmental conditions and progression through the cell cycle.

iii).

- Both the networks fit the power law and are scale free networks as mentioned in the Yu paper.
- There is a difference in the number of essential genes overlap with the hub genes. In the Yu paper, it is found that there was 20% overlapping of the central genes with that of the essential genes. In our results, we only found a total of 4 genes out of the 466 essential genes in the 108 hub genes in the YIB-Union PPI network (0.8%). We found similar results for the CCSB PPI data. We only found 2 essential genes.

Question 2:

i).

Central genes based on the betweenness centrality

In this, we first calculated the betweenness of the all the nodes in both the Y2H Union and CCSN YI1 PPI network. After that, the top 5% genes sorted on the basis of betweenness are taken as the hub genes.

```
top5percentUnion <- vcount(lgraphUnion) * 0.05
centralGenesUnion <- names(sort(igraph::betweenness(lgraphUnion),
  decreasing = TRUE)[1:as.integer(top5percentUnion)]))
# length(centralGenesUnion)
centralGenesUnion
```

```
## [1] "YLR291C" "YLR423C" "YNL189W" "YDR510W" "YBR261C" "YDR100W"
## [7] "YPL049C" "YML051W" "YHR114W" "YGL153W" "YJR091C" "YDL239C"
## [13] "YIR038C" "YDL100C" "YMR047C" "YCL028W" "YPL070W" "YKR034W"
## [19] "YGR120C" "YPR113W" "YHR113W" "YNL044W" "YDR479C" "YJL030W"
## [25] "YOR284W" "YGL127C" "YKL002W" "YOL034W" "YHL004W" "YCR086W"
## [31] "YKL103C" "YCR106W" "YOR355W" "YBR080C" "YFR008W" "YDR448W"
## [37] "YGR099W" "YLR321C" "YGR268C" "YOR111W" "YER081W" "YPR105C"
## [43] "YPL094C" "YIL144W" "YPL004C" "YDR473C" "YLR424W" "YNL288W"
## [49] "YOR047C" "YAL032C" "YLR386W" "YPL031C" "YML099C" "YFL010C"
## [55] "YAR027W" "YOL149W" "YDL012C" "YNL078W" "YIL065C" "YLR132C"
## [61] "YGL181W" "YER125W" "YOR128C" "YML064C" "YBL058W" "YOR380W"
## [67] "YFL039C" "YLR438CA" "YIL074C" "YNL263C" "YNL236W" "YDR311W"
## [73] "YGL225W" "YOR117W" "YKL142W" "YOL082W" "YNL229C" "YIL109C"
## [79] "YOR370C" "YJL058C" "YDR174W" "YLL036C"
```

```
top5percentCCSB <- vcount(lgraphCCSB) * 0.05
centralGenesCCSB <- names(sort(igraph::betweenness(lgraphCCSB),
  decreasing = TRUE)[1:as.integer(top5percentCCSB)]))
# length(centralGenesCCSB)
centralGenesCCSB
```

```
## [1] "YLR291C" "YBR261C" "YLR423C" "YDR510W" "YDR100W" "YCL028W"
## [7] "YPL049C" "YML051W" "YPR113W" "YIR038C" "YNL044W" "YGL122C"
## [13] "YGL153W" "YLR177W" "YDR479C" "YNL189W" "YDL100C" "YDR099W"
## [19] "YBR233W" "YPL267W" "YCR106W" "YKR034W" "YPL070W" "YDR448W"
## [25] "YNL263C" "YOR284W" "YHR113W" "YOL034W" "YLR350W" "YAR027W"
## [31] "YPL094C" "YLR132C" "YKL103C" "YER125W" "YFL039C" "YCR086W"
## [37] "YPL004C" "YLL036C" "YOR380W" "YDL239C" "YPL135W" "YGR268C"
## [43] "YDR473C" "YLR438CA" "YLR424W" "YBR080C" "YGR099W" "YIL109C"
```

```
details <- c("Graph Union", "Graph CCSB")
table <- data.frame(Graph = details, Overlap = c(toString(intersect(centralGenesUnion,
  as.vector(essentialGenes[, 1]))), toString(intersect(centralGenesCCSB,
  as.vector(essentialGenes[, 1])))))
table %>% knitr::kable(caption = "Essential genes overlap with hub genes of largest sub-network")
```

Table 9: Essential genes overlap with hub genes of largest sub-network

Graph	Overlap
Graph Union	YLR423C, YKL002W, YPL031C
Graph CCSB	YLR423C

As seen from the above results, only three essential genes overlapped with the hub genes based on the betweenness. These genes includes YKL002W (7), YLR423C (80), and YPL031C (12). Both the genes YLR423C, and YPL031C pops up as the hub genes based on two centralities we have used.

- **YKL002W**: It is a class E Vps protein of the ESCRT-III complex. It is required for sorting of integral membrane proteins into luminal vesicles of multivesicular bodies, and for delivery of newly synthesized vacuolar enzymes to the vacuole, involved in endocytosis.
- **YLR423C**: It is involved in the phagophore assembly site organization. It also acts as a regulatory subunit of an autophagy-specific complex that includes Atg1p and Atg13p. In addition to that, it also stimulates Atg1p kinase activity.
- **YPL031C**: It is involved in the regulation of the cellular response to nutrient levels and environmental conditions and progression through the cell cycle.

So, out of 465 essential genes we only get 3 genes as the hub nodes in the Y2H union PPI network and 1 gene in the CCSB YI1 PPI network. From these results, we can say that there is no correlation between the central genes based on betweenness and essential genes.

ii).

```
gUniond10 <- delete.edges(lgraphUnion, E(lgraphUnion)[sample(seq_along(E(lgraphUnion)),
  0.1 * ecount(lgraphUnion))])
gUniond25 <- delete.edges(lgraphUnion, E(lgraphUnion)[sample(seq_along(E(lgraphUnion)),
  0.25 * ecount(lgraphUnion))])
gCCSBd10 <- delete.edges(lgraphCCSB, E(lgraphCCSB)[sample(seq_along(E(lgraphCCSB)),
  0.1 * ecount(lgraphCCSB))])
gCCSBd25 <- delete.edges(lgraphCCSB, E(lgraphCCSB)[sample(seq_along(E(lgraphCCSB)),
```

```

0.25 * ecount(lgraphCCSB))])
# write_graph(gUniond10, 'UnionD10.txt', 'ncol')
# write_graph(gUniond25, 'UnionD25.txt', 'ncol')
# write_graph(gCCSBd10, 'CCSBD10.txt', 'ncol')
# write_graph(gCCSBd25, 'CCSBD25.txt', 'ncol')

details <- c("Y2H-Union D10%", "Y2H-Union D25%", "Y2H-CCSB D10%",
            "Y2H-CCSB D25%")
table <- data.frame(Graph = details, `Whole-Net-Degree` = c(1573,
1413, 925, 832), `Whole-Net-Edges` = c(2267, 1889, 1339,
1116), `Sub-Net-Degree` = c(1520, 1303, 881, 779), `Sub-Net-Edges` = c(2230,
1822, 1303, 1086))
table %>% knitr::kable(caption = "Details of the Deleted Networks")

```

Table 10: Details of the Deleted Networks

Graph	Whole.Net.Degree	Whole.Net.Edges	Sub.Net.Degree	Sub.Net.Edges
Y2H-Union D10%	1573	2267	1520	2230
Y2H-Union D25%	1413	1889	1303	1822
Y2H-CCSB D10%	925	1339	881	1303
Y2H-CCSB D25%	832	1116	779	1086

We have deleted 10% and 25% edges from the largest connected sub-network of both the PPI networks. After that, we extracted the largest connected sub-network from that PPI network and performed the MCL clustering in cytoscape. We have used the **Cluster Maker2** plugin for carrying out the MCL clustering. In table 10, the number of nodes and edges for both complete and largest connected subnetwork is shown for the deleted networks.

```

outputgraphUnionClust <- graphDetails(paste0(path, "Union-lsub-clustered.csv"),
",")
outputgraphUnionD10Clust <- graphDetails(paste0(path, "Union-D10-Clust.csv"),
",")
outputgraphUnionD25Clust <- graphDetails(paste0(path, "Union-D25-Clust.csv"),
",")
outputgraphCCSBClust <- graphDetails(paste0(path, "CCSB-lsub-clustered.csv"),
",")
outputgraphCCSBD10Clust <- graphDetails(paste0(path, "CCSB-D10-clust.csv"),
",")
outputgraphCCSBD25Clust <- graphDetails(paste0(path, "CCSB-D25-Clust.csv"),
",")

lgraphUnionClust <- largestSubNet(outputgraphUnionClust[[2]])
lgraphUnionD10Clust <- largestSubNet(outputgraphUnionD10Clust[[2]])
lgraphUnionD25Clust <- largestSubNet(outputgraphUnionD25Clust[[2]])
lgraphCCSBClust <- largestSubNet(outputgraphCCSBClust[[2]])
lgraphCCSBD10Clust <- largestSubNet(outputgraphCCSBD10Clust[[2]])
lgraphCCSBD25Clust <- largestSubNet(outputgraphCCSBD25Clust[[2]])

details <- c("Number of Nodes", "Number of Edges", "Graph Density",
            "Diameter", "Radius", "Average Clustering Coefficient", "Global Clustering Coefficient",
            "Average Shortest Path")
table <- data.frame(Properties = details, `Y2H-Union` = gdetails(lgraphUnionClust),

```

```
`Y2H-UnionD10` = gdetails(lgraphUnionD10Clust), `Y2H-UnionD25` = gdetails(lgraphUnionD25Clust))
table %>% knitr::kable(caption = "Details of the Largest Modules in Y2H Union PPI")
```

Table 11: Details of the Largest Modules in Y2H Union PPI

Properties	Y2H.Union	Y2H.UnionD10	Y2H.UnionD25
Number of Nodes	59.0000000	44.0000000	21.0000000
Number of Edges	73.0000000	68.0000000	20.0000000
Graph Density	0.0426651	0.0718816	0.0952381
Diameter	6.0000000	4.0000000	2.0000000
Radius	3.0000000	2.0000000	1.0000000
Average Clustering Coefficient	0.0635721	0.2038390	0.0000000
Global Clustering Coefficient	0.0034762	0.0235110	0.0000000
Average Shortest Path	2.8392753	2.6109937	1.9047619

```
table <- data.frame(Properties = details, `CCSB-YI1` = gdetails(lgraphCCSBClust),
`CCSB-YI1D10` = gdetails(lgraphCCSBD10Clust), `CCSB-YI1D25` = gdetails(lgraphCCSBD25Clust))
table %>% knitr::kable(caption = "Details of the Largest Modules in CCSB-YI1 PPI")
```

Table 12: Details of the Largest Modules in CCSB-YI1 PPI

Properties	CCSB.YI1	CCSB.YI1D10	CCSB.YI1D25
Number of Nodes	49.0000000	37.0000000	22.0000000
Number of Edges	72.0000000	48.0000000	21.0000000
Graph Density	0.0612245	0.0720721	0.0909091
Diameter	4.0000000	4.0000000	4.0000000
Radius	2.0000000	2.0000000	2.0000000
Average Clustering Coefficient	0.1296706	0.2793464	0.0000000
Global Clustering Coefficient	0.0122449	0.0188679	0.0000000
Average Shortest Path	2.3494898	2.0345345	2.1558442

Comparison of Modules

```
# To get the maximum overlap clusters
getMaxOverlapsClusters <- function(graph1, graph2) {
  dGraph1 <- decompose.graph(graph1)
  dGraph2 <- decompose.graph(graph2)
  largestIntersect <- 0
  cluster1 <- 0
  cluster2 <- 0
  node1 <- 0
  node2 <- 0
  for (i in 1:length(dGraph1)) {
    if (vcount(dGraph1[[i]]) >= 5) {
      for (j in 1:length(dGraph2)) {
        if (vcount(dGraph2[[j]]) >= 5) {
          inter <- intersect(names(V(dGraph1[[i]])),
                             names(V(dGraph2[[j]])))

```

```

        if (length(inter) > largestIntersect) {
          largestIntersect <- length(inter)
          cluster1 <- i
          cluster2 <- j
          node1 <- vcount(dGraph1[[i]])
          node2 <- vcount(dGraph2[[j]])
        }
      }
    }
  }
  return(c(cluster1, cluster2, largestIntersect, node1, node2))
}

UnionVSCCSB <- getMaxOverlapsClusters(outputgraphUnionClust[[2]],
  outputgraphCCSBClust[[2]])
UnionVSUnionD10 <- getMaxOverlapsClusters(outputgraphUnionClust[[2]],
  outputgraphUnionD10Clust[[2]])
UnionVSUnionD25 <- getMaxOverlapsClusters(outputgraphUnionClust[[2]],
  outputgraphUnionD25Clust[[2]])
CCSBVSCCSBD10 <- getMaxOverlapsClusters(outputgraphCCSBClust[[2]],
  outputgraphCCSBD10Clust[[2]])
CCSBVSCCSBD25 <- getMaxOverlapsClusters(outputgraphCCSBClust[[2]],
  outputgraphCCSBD25Clust[[2]])

details <- c("Y2H-Unionv/sCCSB-YI1", "Y2H-Unionv/sY2H-UnionD10",
  "Y2H-Unionv/sY2H-UnionD25", "CCSB-YI1v/sCCSB-YI1D10", "CCSB-YI1v/sCCSB-YI1D25")
table <- data.frame(Comparisons = details, `Module Numbers` = c(paste0(UnionVSCCSB[1],
  " and ", UnionVSCCSB[2]), paste0(UnionVSUnionD10[1], " and ",
  UnionVSUnionD10[2]), paste0(UnionVSUnionD25[1], " and ",
  UnionVSUnionD25[2]), paste0(CCSBVSCCSBD10[1], " and ", CCSBVSCCSBD10[2]),
  paste0(CCSBVSCCSBD25[1], " and ", CCSBVSCCSBD25[2])), `#Overlap Genes` = c(UnionVSCCSB[3],
  UnionVSUnionD10[3], UnionVSUnionD25[3], CCSBVSCCSBD10[3],
  CCSBVSCCSBD25[3]), `#Gene in Module 1` = c(UnionVSCCSB[4],
  UnionVSUnionD10[4], UnionVSUnionD25[4], CCSBVSCCSBD10[4],
  CCSBVSCCSBD25[4]), `#Gene in Module 2` = c(UnionVSCCSB[5],
  UnionVSUnionD10[5], UnionVSUnionD25[5], CCSBVSCCSBD10[5],
  CCSBVSCCSBD25[5]))
table %>% knitr::kable(caption = "Overlap Details of the Modules")

```

Table 13: Overlap Details of the Modules

Comparisons	Module.Numbers	X.Overlap.Genes	X.Gene.in.Module.1	X.Gene.in.Module.2
Y2H-Unionv/sCCSB-YI1	2 and 1	32	34	49
Y2H-Unionv/sY2H-UnionD10	1 and 3	34	59	36
Y2H-Unionv/sY2H-UnionD25	1 and 2	18	59	20
CCSB-YI1v/sCCSB-YI1D10	1 and 1	36	49	37
CCSB-YI1v/sCCSB-YI1D25	2 and 1	19	31	22

From table 13, it can be seen that the maximum overlap between the Y2H Union and CCSB YI1 PPI network clusters is of 32 genes. It should be noted that this overlap is between the cluster 2 of the Y2H Union and cluster 1 CCSB YI1 PPI network. When we tried to compare the top most cluster of both these networks it

gives us an overlap of only 1 gene. As, we know that Y2H Union is a superset of the CCSB Y1I network, we can observe that results from network analysis depend a lot on the initial dataset. If the initial data is changed, our whole results gets changed which we found in our results. The same thing we found when we compared the deleted networks with the original ones.

iii).

For this question, we selected the largest modules of both the PPI networks based on the number of nodes in the network as shown in Figure 1 and 2. After that, we used the **Yeast Genome** website to carry out the GO terms enrichment analysis to get the common functions associated with the genes in the cluster. **Revigo** is used to group the GO terms.

Figure 3 consists of one part of the output that we get after entering the genes from the largest connected cluster of Y1-Union PPI. The genes in the cluster are associated with various biosynthetic and metabolic processes which are essential for the survival of yeast. In Figure 4, it is seen that we have four major groups of GO Biological processes including single organism metabolism, organophosphate biosynthesis, single organism process, and pyrimidine containing compound metabolism. All of these groups are in fact essential for yeast growth and development thus we can say from this observation that this is a very important cluster of genes for yeast development.

Figure 5 consists of one part of the output that we get after entering the genes from the largest connected cluster of CCSB Y1I PPI. The genes in the cluster are associated with various glycoprotein and lipid metabolism and biosynthetic processes. These processes are essential for the growth and development of yeast. We also used Revigo to group the GO terms. In Figure 6, it is seen that we have two major groups of GO Biological processes including lipid and glycoprotein metabolism. These biological processes are observed to be important for the growth and development of yeast.

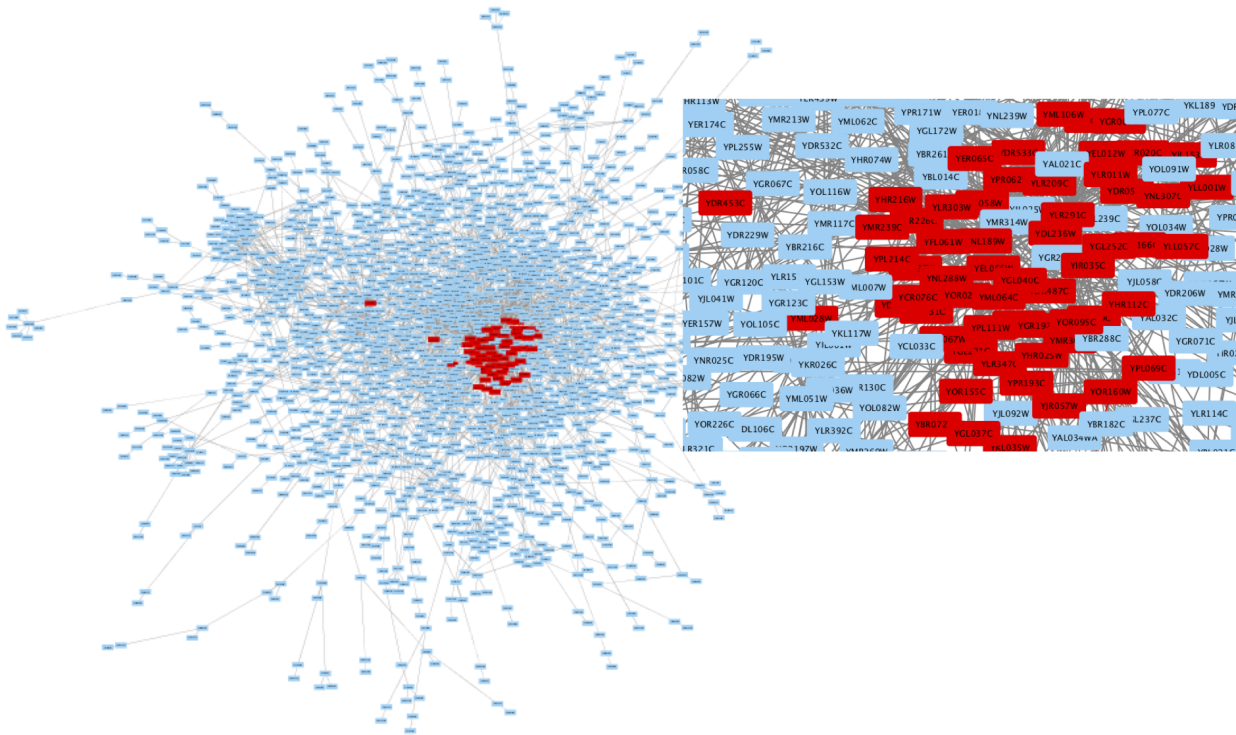


Figure 1: Y1-Union PPI Module 1 - Cytoscape

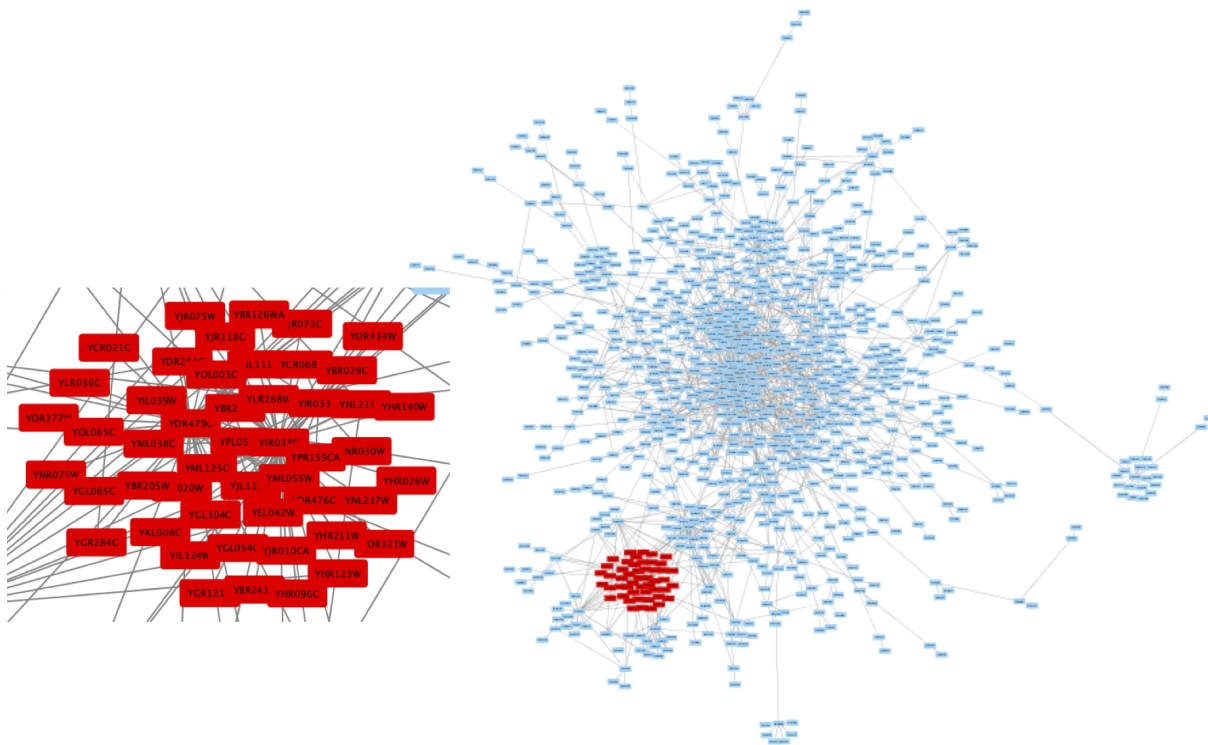


Figure 2: CCSB PPI Module 1 - Cytoscape

Gene Ontology Enrichment

GO terms enriched for items in this list.

Number of Genes in this list not analysed in this widget: 2

Test Correction

Holm-Bonferroni

Max p-value

0.05

Ontology

biological_process

Background population

Default

Change

View

Download


<input type="checkbox"/> GO Term	p-Value 	Matches
<input type="checkbox"/> single-organism biosynthetic process [GO:0044711]	4.944653e-8	28
<input type="checkbox"/> single-organism metabolic process [GO:0044710]	6.027952e-7	38
<input type="checkbox"/> small molecule metabolic process [GO:0044281]	1.008751e-5	27
<input type="checkbox"/> pyrimidine-containing compound metabolic process [GO:0072527]	0.015796	6
<input type="checkbox"/> single-organism process [GO:0044699]	0.023002	50
<input type="checkbox"/> organophosphate biosynthetic process [GO:0090407]	0.035087	12

Figure 3: Y1-Union PPI Module 1 - Yeast genome Results

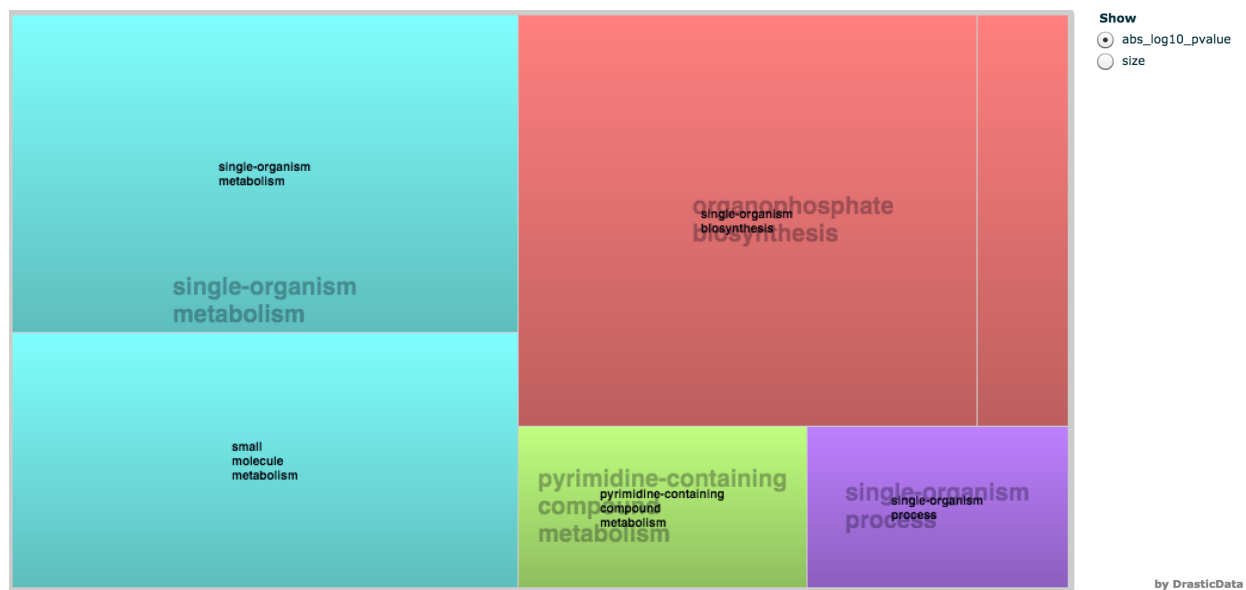


Figure 4: Y1-Union PPI Module 1 - Revigo Treemap

Gene Ontology Enrichment

GO terms enriched for items in this list.

Number of Genes in this list not analysed in this widget: 3

Test Correction

Max p-value

Ontology

Holm-Bonferroni

0.05

biological_process

Background population

Default

Change

View

Download


<input type="checkbox"/> GO Term	p-Value 	Matches
<input type="checkbox"/> lipid metabolic process [GO:0006629]	6.366361e-6	16
<input type="checkbox"/> cellular lipid metabolic process [GO:0044255]	2.120038e-5	15
<input type="checkbox"/> protein glycosylation [GO:0006486]	2.360629e-5	9
<input type="checkbox"/> macromolecule glycosylation [GO:0043413]	2.360629e-5	9
<input type="checkbox"/> glycoprotein biosynthetic process [GO:0009101]	3.305823e-5	9
<input type="checkbox"/> glycoprotein metabolic process [GO:0009100]	4.568843e-5	9
<input type="checkbox"/> glycosylation [GO:0070085]	6.236615e-5	9
<input type="checkbox"/> mannosylation [GO:0097502]	2.534937e-4	7
<input type="checkbox"/> phospholipid metabolic process [GO:0006644]	9.615770e-4	10
<input type="checkbox"/> carbohydrate derivative biosynthetic process [GO:1901137]	0.006195	11
<input type="checkbox"/> glycerophospholipid metabolic process [GO:0006650]	0.012372	8
<input type="checkbox"/> oligosaccharide lipid intermediate biosynthetic process	0.013885	4

Figure 5: CCSB PPI Module 1 - Yeast genome Results



Figure 6: CCSB PPI Module 1 - Revigo Treemap