HW3

Urminder Singh

February 2, 2018

Sol1

General network features

The general network features like #edges, #nodes, radius etc are shown in Table 1.

```
library("igraph")
library(readxl)
library("MCL")
Y2H_uni <- read.table("Y2H_uniondata.txt")
graph_Y2H_uni <- graph.data.frame(as.matrix(Y2H_uni), directed = FALSE)
graph_Y2H_uni_Details <- c(ecount(graph_Y2H_uni), vcount(graph_Y2H_uni), graph.density(graph_Y2H_uni),
    diameter(graph_Y2H_uni), radius(graph_Y2H_uni), transitivity(graph_Y2H_uni,
        type = "global"), transitivity(graph_Y2H_uni, type = "average"), mean_distance(graph_Y2H_uni))

CCSB_YI1 <- read.table("CCSB_YI1.txt")
graph_CCSB_YI1 <- graph.data.frame(as.matrix(CCSB_YI1), directed = FALSE)
graph_CCSB_YI1_Details <- c(ecount(graph_CCSB_YI1), vcount(graph_CCSB_YI1),
    graph.density(graph_CCSB_YI1), diameter(graph_CCSB_YI1), radius(graph_CCSB_YI1),
    transitivity(graph_CCSB_YI1, type = "global"), transitivity(graph_CCSB_YI1,
        type = "average"), mean_distance(graph_CCSB_YI1))

# wtite to table
header1 <- c("#Edges", "#Vertices", "Density", "Diameter", "Radius", "Clustering_coeff(global)",
    "Clustering_coeff(avg)", "Avg Shortest Path")
tab1 <- data.frame(Feature = header1, Y2H_uni = graph_Y2H_uni_Details, CCSB_YI1 = graph_CCSB_YI1_Details
tab1 %>% knitr::kable(caption = "Graph Features")
```

Table 1: Graph Features

Feature	Y2H_uni	CCSB_YI1
#Edges	2930.0000000	1809.0000000
#Vertices	2018.0000000	1278.0000000
Density	0.0014397	0.0022169
Diameter	14.0000000	14.0000000
Radius	0.0000000	0.0000000
Clustering_coeff(global)	0.0236142	0.0206502
Clustering_coeff(avg)	0.0970026	0.1059977
Avg Shortest Path	5.6109291	5.3641783

Testing for small world property

In a small world network most of the nodes can be reached by other nodes in only a short number of hops. In a small-world network the typical distance D between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network, that is

$$D \sim log(N)$$

It is difficult to statistically test whether a given a network is small-world or not. Here I used the approach as described by Watts and Strogatz (Watts and Strogatz. Nature 393. 1998). We can compare small world network with a random network with the same number on nodes and vertices. We expect that a small world network will have higher clustering coeff than coressponding random network and the avg shortest path is expected to be greater in small world networks than random network. So, I randomized the given networks 3 times to see if the conditions hold.

I observed that the Agv. shortest path in the random networks is longer than the given networks in all three cases. Also, the clustering coefficient of the given networks is greater than those of the random networks in all three cases. Thus, only one condition is satisfied and it is difficult to say with confidence that these networks are small world or not.

```
# generate 3 random netwroks for each graph
rn1a <- erdos.renyi.game(vcount(graph_Y2H_uni), ecount(graph_Y2H_uni), type = "gnm")
rn1b <- erdos.renyi.game(vcount(graph_Y2H_uni), ecount(graph_Y2H_uni), type = "gnm")
rn1c <- erdos.renyi.game(vcount(graph_Y2H_uni), ecount(graph_Y2H_uni), type = "gnm")
rn2a <- erdos.renyi.game(vcount(graph_CCSB_YI1), ecount(graph_CCSB_YI1), type = "gnm")
rn2b <- erdos.renyi.game(vcount(graph_CCSB_YI1), ecount(graph_CCSB_YI1), type = "gnm")
rn2c <- erdos.renyi.game(vcount(graph_CCSB_YI1), ecount(graph_CCSB_YI1), type = "gnm")

# get cc and mean dist
cc1a <- transitivity(rn1a, type = "global")
cc1b <- transitivity(rn1b, type = "global")
cc1c <- transitivity(rn1c, type = "global")
cc2a <- transitivity(rn2a, type = "global")
cc2b <- transitivity(rn2b, type = "global")
cc2c <- transitivity(rn2c, type = "global")

md1a <- mean_distance(rn1a)
md1b <- mean_distance(rn1b)
md1c <- mean_distance(rn1c)
md2a <- mean_distance(rn2a)
md2b <- mean_distance(rn2b)
md2c <- mean_distance(rn2c)

header2 <- c("Clustering_coeff(global)", "Avg Shortest Path")
tab2 <- data.frame(Network = header2, Y2H_uni_randomized_a = c(cc1a, md1a),
    Y2H_uni_randomized_b = c(cc1b, md1b), Y2H_uni_randomized_c = c(cc1c, md1c),
    CCSB_YI1_randomized_a = c(cc2a, md2a), CCSB_YI1_randomized_b = c(cc2b, md2b),
    CCSB_YI1_randomized_c = c(cc2c, md2c))
tab2 <- t(tab2)
tab2 %>% knitr::kable(caption = "Clustering Coeff and Avg shortest path in random networks")
```

Table 2: Clustering Coeff and Avg shortest path in random networks

Network	Clustering_coeff(global)	Avg Shortest Path
Y2H_uni_randomized_a	0.002452125	7.032280404

Y2H_uni_randomized_b	0.0003460607	6.9845307111
Y2H_uni_randomized_c	0.002122391	7.072930195
CCSB_YI1_randomized_a	0.001749271	6.804317558
CCSB_YI1_randomized_b	0.001778656	6.759885866
CCSB_YI1_randomized_c	0.001163242	6.742871016

Testing for power law (scale-free) property

To test if the networks follow powerlaw we first plot the degree distribution to visually see if the degree distribution can be a power law. Then we can use the function *power.law.fit* from igraph package and then reject or accept the null hypothesis, that degree distribution is sampled from a scle free distribution, at a confidence level using the p-value reported. Looking at Fig. 1 which shows the degree distributions of the networks, we can say that both networks follow power law. Then, Table 3. shows the result of *power.law.fit* function. I observed high p-values which indicates that the degree distributions follow a power law. Also the reported power law exponent is between 2 and 3, which strongly suggests that these networks are scale free.

```
# plot degree dist
par(mfrow = c(1, 3))
plot(degree.distribution(graph_Y2H_uni), xlab = "log(k)", ylab = "log(P(k))",
    log = "xy", type = "o", main = "a: Degree Distribution Y2H_uni")
plot(degree.distribution(graph_CCSB_YI1), xlab = "log(k)", ylab = "log(P(k))",
    log = "xy", type = "o", main = "b: Degree Distribution CCSB_YI1")
g_powlaw <- barabasi.game(1000)
plot(degree.distribution(g_powlaw), xlab = "log(k)", ylab = "log(P(k))", log = "xy",
    type = "o", main = "c: Powerlaw distribution")
```

```
# use powerlaw fit
fitA <- power.law.fit(degree(graph_Y2H_uni, mode = "all"), implementation = "plfit")
fitB <- power.law.fit(degree(graph_CCSB_YI1, mode = "all"), implementation = "plfit")
fitC <- power.law.fit(degree(g_powlaw, mode = "all"), implementation = "plfit")
head3 <- c("Y2H_uni", "CCSB_YI1", "Powerlaw graph")
tab3 <- data.frame(Graph = head3, Alpha = c(fitA$alpha, fitB$alpha, fitC$alpha),
    X_min = c(fitA$xmin, fitB$xmin, fitC$xmin), KS_stat = c(fitA$KS.stat, fitB$KS.stat,
        fitC$KS.stat), P_Value = c(fitA$KS.p, fitB$KS.p, fitC$KS.p), logLik = c(fitA$logLik,
        fitB$logLik, fitC$logLik))
tab3 %>% knitr::kable(caption = "Power law fits")
```

Table 3: Power law fits

Graph	Alpha	X_min	KS_stat	P_Value	logLik
Y2H_uni	2.893596	7	0.0267756	0.9997574	-462.4460
CCSB_YI1	2.625133	4	0.0261695	0.9951312	-608.3298
Powerlaw graph	2.765336	3	0.0144907	1.0000000	-339.1307

Finding hubs

Hubs are the nodes which have high degree in the network. To find the hubs we first empirically decide a degree such that any node having higher degree is a hub. To do this we plot the degree distribution of the network and set a threshhold such that probability of higher degree is lower. From Fig. 2 and 3 we decide that any node having degree more than 11 in Y2H_uni network is a hub and a node having degree more

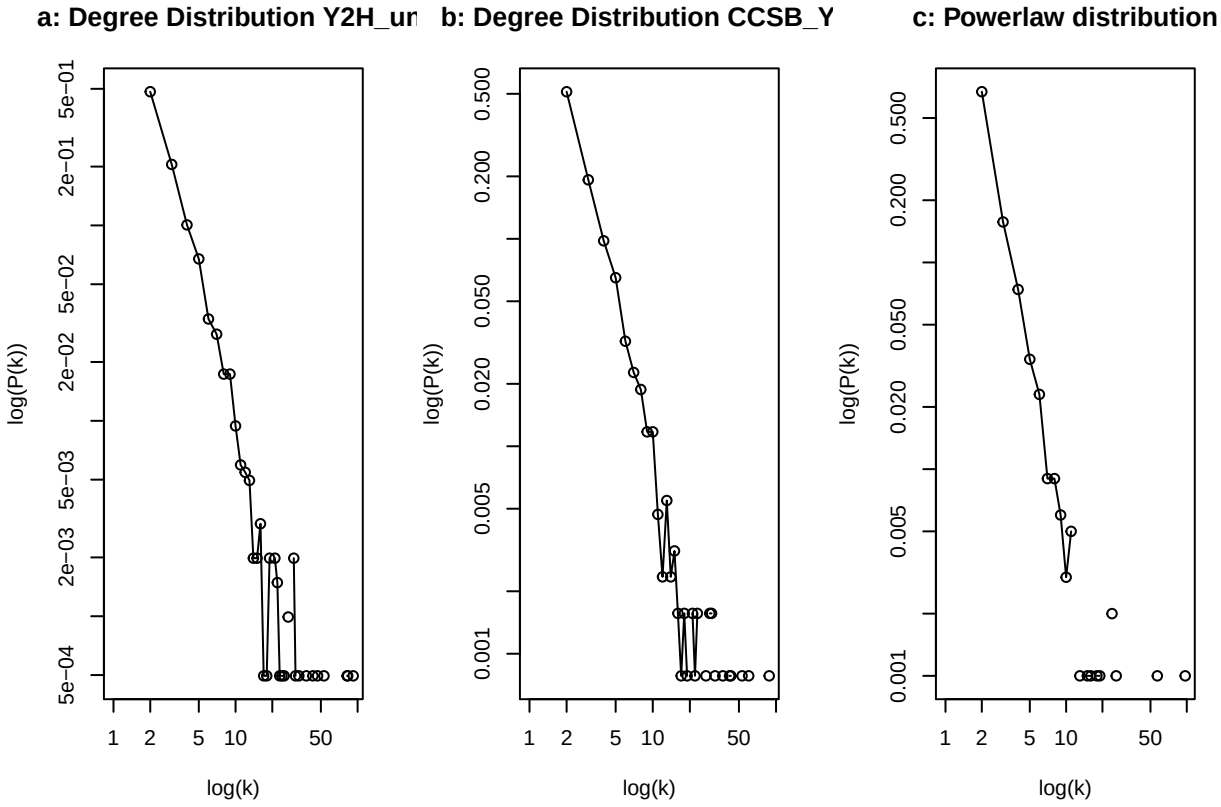

Figure 1: Degree distributions of graphs(log-log scale)

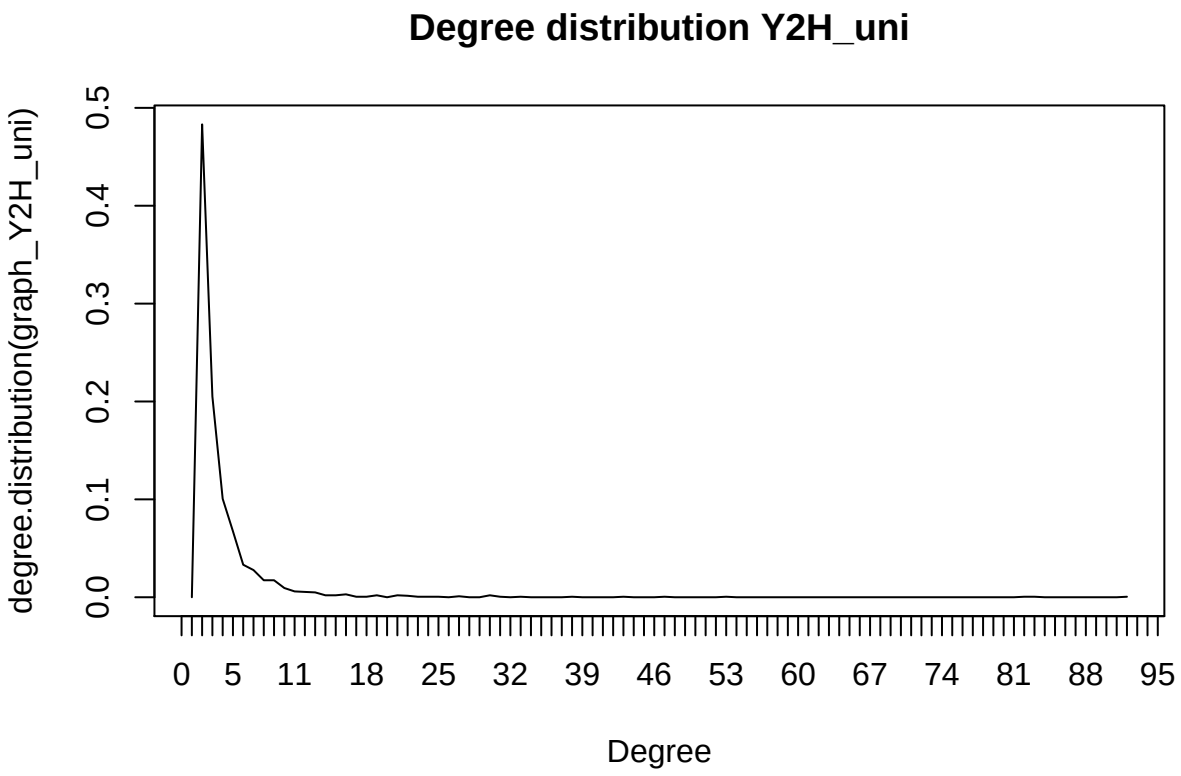

Figure 2: Degree distribution Y2H_uni

than 10 in CCSB_YI1 network is a hub.

I found two hubs in Y2H_uni and two hubs in CCSB_YI1 which are present in the Yeast_deletion project. These two hubs were the same for both the networks (see Table 4.). They are *YPL031c* and *YLR423c*. The ORF *YPL031c* is described as "cyclin-dependent protein kinase" and *YLR423c* is defind as "protein involved in authophagy".

```
# first plot degree dist to find degree threshhold
plot(degree.distribution(graph_Y2H_uni), type = "l", xlab = "Degree", main = "Degree distribution Y2H_u
    xaxt = "n")
axis(side = 1, at = c(0:100))
```

```
plot(degree.distribution(graph_CCSB_YI1), type = "l", xlab = "Degree", main = "Degree distribution CCSB
    xaxt = "n")
axis(side = 1, at = c(0:100))
```

```
# find hubs in the networks
hubs_Y2H_uni <- names(which(degree(graph_Y2H_uni) > 11))
hubs_CCSB_YI1 <- names(which(degree(graph_CCSB_YI1) > 10))
# hubs_common <- intersect(hubs_Y2H_uni,hubs_CCSB_YI1) read yeast deletion
# data
Yeast_deletionProject <- read_excel("Yeast_deletionProject.xlsx")
# remove extra whitespace from first col
ORFs_Yeast_proj <- as.data.frame(apply(Yeast_deletionProject, 2, function(x) gsub("\\s+",
    "", x)))$ORF
```

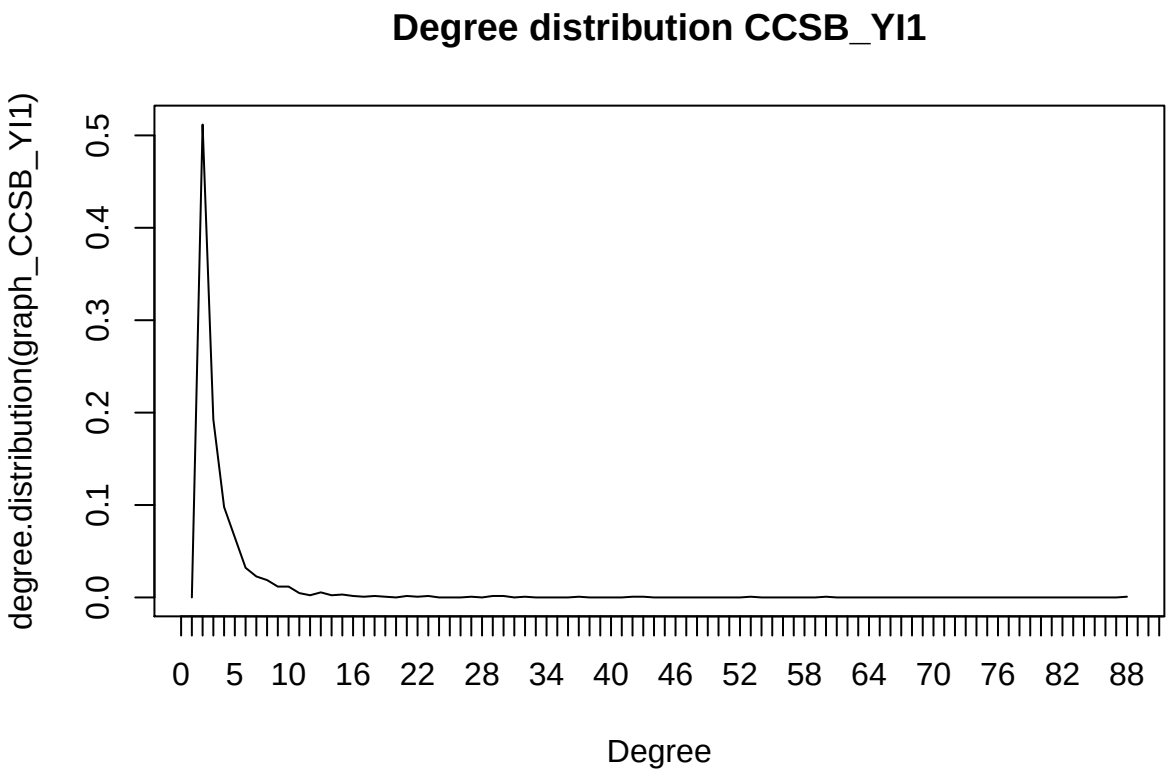

Figure 3: Degree distribution CCSB_YI1

```r
# find common ORFs
hubs_common_Y2H_uni <- intersect(hubs_Y2H_uni, ORFs_Yeast_proj)
hubs_common_CCSB_YI1 <- intersect(hubs_CCSB_YI1, ORFs_Yeast_proj)
hubs_common <- intersect(hubs_common_CCSB_YI1, hubs_common_Y2H_uni)
head4 <- c("Y2H_uni", "CCSB_YI1", "Intersection")
tab4 <- data.frame(Network = head4, Hubs_common = c(paste(hubs_common_Y2H_uni,
    collapse = ","), paste(hubs_common_CCSB_YI1, collapse = ","), paste(hubs_common,
    collapse = ",")))

tab4 %>% knitr::kable(caption = "Hubs in the network common with YeastDeletion Project data")
```

Table 4: Hubs in the network common with YeastDeletion Project data

Network	Hubs_common
Y2H_uni	YLR423C,YPL031C
CCSB_YI1	YPL031C,YLR423C
Intersection	YPL031C,YLR423C

Next, to test the significance of hubs, I took, randomly, a set of genes and checked whether at least 2 of them will appear in the yeast deletion project or not. The number of genes were same as the number of hubs found in the network i.e. 55 for Y2H_uni and 40 for CCSB_YI1 and since I found 2 genes from the hubs essential I am checking how likely it is it to sample a subset and get 2 or more genes in the essential gene list. For each of these networks, I chose set of genes then checked if at least 2 of them were present in the yeast deletion project. I repeated this process 100 times. My null hypothesis is that nodes chosen at random are essential. I define my test statistic as #observations where essential gene are more than or equal to 2.

From my test, I get a very high p-value (0.8) which suggests that from any randomly chosen set of genes, there could be a at least N number of genes present in the essential gene list, where N is the number of essential genes which are also hubs. Thus, it implies that grouping genes by centrality doesn't make them essential and if we select a random set of genes they are also equally or more probable to be present in the essential gene list.

```r
# test essentiality are randomly selected genes essential?
N = 100
n1 = length(hubs_Y2H_uni)
C1 <- length(hubs_common_Y2H_uni)
T1 = 0
# repeat N times
nodes_Y2H <- names(degree(graph_Y2H_uni))
for (i in 1:N) {
    # print(i) sample n1 genes from Y2H_uni
    thissample <- sample(nodes_Y2H, n1)
    intersect(thissample, ORFs_Yeast_proj)
    if (length(intersect(thissample, ORFs_Yeast_proj)) >= C1) {
        T1 = T1 + 1
    }

}

print("Pvalue is")
```

```
## [1] "Pvalue is"
```

```
print(T1/N)
```

```
## [1] 0.81
```

Sol1 c

The Yu paper reports that the networks are scale free and follows power law. I observed the same in this report. The Yu paper found around 90 of essential genes overlapping with their central genes. However, in my report I only found 2 which is contradictory to Yu paper.

Sol 2

Table 5. shows the various properties of the BioGrid network. This network has greater number of nodes as compared to Y2H and CCSB networks. This graph is denser than the other two and has global clustering coeff of 0.0428, which is higher as compared to other networks. I further checked the degree distribution and whether it follows power law or not. The results are in Table 6 which shows the results of the function power.law.fit on the three networks. We can see that all networks have comparable α and high p-values. This means BioGrid network also follows power law and is scale-free.

I also checked whether the BioGrid network behave likes a small world network. The results in Table 7 show that clustering coeff of BioGrid network is greater than that of a random network with same density but the mean shortest path is shorter than that of the random network. This is exactly same behaviour as of the other two networks i.e. Y2H_uni and CCSB_YI1. Based on these observations, we can say that the BioGrid network is pretty similar to the other smaller networks.

Looking at these properties we can safely assume that we are dealing with scale-free networks as even larger networks have very similar properties to the smaller ones.

```
# read BioGrid2018_uni-2
BioGrid18_data <- read.table("BioGrid2018_uni-2.txt")
graph_BioGrid18 <- graph.data.frame(as.matrix(BioGrid18_data), directed = FALSE)
graph_BioGrid18_Details <- c(ecount(graph_BioGrid18), vcount(graph_BioGrid18),
    graph.density(graph_BioGrid18), diameter(graph_BioGrid18), radius(graph_BioGrid18),
    transitivity(graph_BioGrid18, type = "global"), transitivity(graph_BioGrid18,
        type = "average"), mean_distance(graph_BioGrid18))
head5 <- c("#Edges", "#Vertices", "Density", "Diameter", "Radius", "Clustering_coeff(global)",
    "Clustering_coeff(avg)", "Avg Shortest Path")
tab5 <- data.frame(Feature = head5, BioGrid18 = graph_BioGrid18_Details)
tab5 %>% knitr::kable(caption = "BioGrid Network Features")
```

Table 5: BioGrid Network Features

Feature	BioGrid18
#Edges	9.753900e+04
#Vertices	5.960000e+03
Density	5.492700e-03
Diameter	6.000000e+00
Radius	3.000000e+00
Clustering_coeff(global)	4.275510e-02
Clustering_coeff(avg)	3.332975e-01
Avg Shortest Path	2.335485e+00

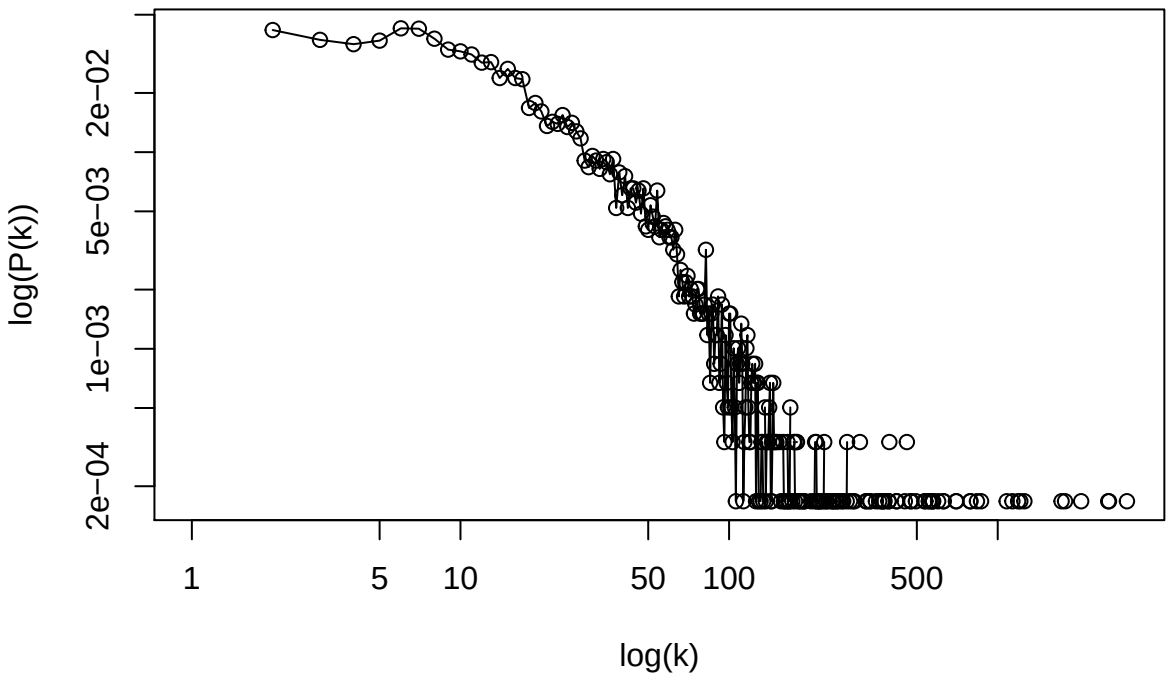

Figure 4: Degree distribution BioGrid18

```
plot(degree.distribution(graph_BioGrid18), xlab = "log(k)", ylab = "log(P(k))",
    log = "xy", type = "o")
```

```
fitD <- power.law.fit(degree(graph_BioGrid18, mode = "all"), implementation = "plfit")
head6 <- c("Y2H_uni", "CCSB_YI1", "BioGrid18")
tab6 <- data.frame(Graph = head3, Alpha = c(fitA$alpha, fitB$alpha, fitD$alpha),
    X_min = c(fitA$xmin, fitB$xmin, fitD$xmin), KS_stat = c(fitA$KS.stat, fitB$KS.stat,
        fitD$KS.stat), P_Value = c(fitA$KS.p, fitB$KS.p, fitD$KS.p), logLik = c(fitA$logLik,
        fitB$logLik, fitD$logLik))
tab6 %>% knitr::kable(caption = "BioGrid Network Powerlaw fit results")
```

Table 6: BioGrid Network Powerlaw fit results

Graph	Alpha	X_min	KS_stat	P_Value	logLik
Y2H_uni	2.893596	7	0.0267756	0.9997574	-462.4460
CCSB_YI1	2.625133	4	0.0261695	0.9951312	-608.3298
Powerlaw graph	2.689456	45	0.0294566	0.2970416	-5334.4151

```
# check for small world generate 3 random netwroks for each graph
rn3a <- erdos.renyi.game(vcount(graph_BioGrid18), ecount(graph_BioGrid18), type = "gnm")
cc3a <- transitivity(rn3a, type = "global")
md3a <- mean_distance(rn3a)
head7 <- c("Clustering_coeff(global)", "Avg Shortest Path")
tab7 <- data.frame(Feature = head7, BioGrid18 = c(transitivity(graph_BioGrid18,
    type = "global"), mean_distance(graph_BioGrid18)), Random_Net = c(cc3a,
    md3a))
tab7 %>% knitr::kable(caption = "BioGrid Network vs Random Network")
```

```
getLargestCC <- function(g) {
    # decompose graph into connected comps
    decomp <- decompose.graph(g)
    # say first component is largest
    largest <- decomp[[1]]
    for (i in 1:length(decomp)) {
        thisg <- decomp[[i]]
        # if there is largere component
        if (vcount(thisg) > vcount(largest)) {
            largest <- thisg
        }
    }
    # return largest
    return(largest)
}

graph_Y2H_uni_lcc <- getLargestCC(graph_Y2H_uni)
graph_CCSB_YI1_lcc <- getLargestCC(graph_CCSB_YI1)
graph_BioGrid18_lcc <- getLargestCC(graph_BioGrid18)
graph_Y2H_uni_lcc_Details <- c(ecount(graph_Y2H_uni_lcc), vcount(graph_Y2H_uni_lcc),
    graph.density(graph_Y2H_uni_lcc), diameter(graph_Y2H_uni_lcc), radius(graph_Y2H_uni_lcc),
    transitivity(graph_Y2H_uni_lcc, type = "global"), transitivity(graph_Y2H_uni_lcc,
        type = "average"), mean_distance(graph_Y2H_uni_lcc))
graph_CCSB_YI1_lcc_Details <- c(ecount(graph_CCSB_YI1_lcc), vcount(graph_CCSB_YI1_lcc),
    graph.density(graph_CCSB_YI1_lcc), diameter(graph_CCSB_YI1_lcc), radius(graph_CCSB_YI1_lcc),
    transitivity(graph_CCSB_YI1_lcc, type = "global"), transitivity(graph_CCSB_YI1_lcc,
        type = "average"), mean_distance(graph_CCSB_YI1_lcc))
graph_BioGrid18_lcc_Details <- c(ecount(graph_BioGrid18_lcc), vcount(graph_BioGrid18_lcc),
    graph.density(graph_BioGrid18_lcc), diameter(graph_BioGrid18_lcc), radius(graph_BioGrid18_lcc),
    transitivity(graph_BioGrid18_lcc, type = "global"), transitivity(graph_BioGrid18_lcc,
        type = "average"), mean_distance(graph_BioGrid18_lcc))
# wtite to table
head8 <- c("#Edges", "#Vertices", "Density", "Diameter", "Radius", "Clustering_coeff(global)",
    "Clustering_coeff(avg)", "Avg Shortest Path")
tab8 <- data.frame(Feature = head8, Y2H_uni_lcc = graph_Y2H_uni_lcc_Details,
    CCSB_YI1_lcc = graph_CCSB_YI1_lcc_Details, BioGrid_lcc = graph_BioGrid18_lcc_Details)
tab8 %>% knitr::kable(caption = "Largest connected component properties")
```

Table 8: Largest connected component properties

Feature	Y2H_uni_lcc	CCSB_YI1_lcc	BioGrid_lcc
#Edges	2682.0000000	1598.0000000	9.753900e+04
#Vertices	1647.0000000	964.0000000	5.960000e+03
Density	0.0019786	0.0034427	5.492700e-03
Diameter	14.0000000	14.0000000	6.000000e+00
Radius	8.0000000	8.0000000	3.000000e+00
Clustering_coeff(global)	0.0236669	0.0205214	4.275510e-02
Clustering_coeff(avg)	0.1023266	0.1071888	3.332975e-01
Avg Shortest Path	5.6117474	5.3659488	2.335485e+00

```
# find most essential genes by betweenness Assuming Top 5% are essential
bw_Y2H_uni_lcc <- betweenness(graph_Y2H_uni_lcc, directed = F)
```

```
bw_CCSB_YI1_lcc <- betweenness(graph_CCSB_YI1_lcc, directed = F)
bw_Y2H_uni_lcc <- names(head(sort(bw_Y2H_uni_lcc, decreasing = T), vcount(graph_Y2H_uni_lcc) *
    0.05))
bw_CCSB_YI1_lcc <- names(head(sort(bw_CCSB_YI1_lcc, decreasing = T), vcount(graph_CCSB_YI1_lcc) *
    0.05))
# find common with Yeast_deletion data
bw_common_Y2H_uni <- intersect(bw_Y2H_uni_lcc, ORFs_Yeast_proj)
bw_common_CCSB_YI1 <- intersect(bw_CCSB_YI1_lcc, ORFs_Yeast_proj)
bw_common <- intersect(bw_common_Y2H_uni, bw_common_CCSB_YI1)
head9 <- c("Y2H_uni_lcc", "CCSB_YI1_lcc", "Intersection")
tab9 <- data.frame(Network = head4, Hubs_common = c(paste(bw_common_Y2H_uni,
    collapse = ","), paste(bw_common_CCSB_YI1, collapse = ","), paste(bw_common,
    collapse = ",")))
tab9 %>% knitr::kable(caption = "Essential hubs found using betweenness")
```

Table 9: Essential hubs found using betweenness

Network	Hubs_common
Y2H_uni	YLR423C,YKL002W,YPL031C
CCSB_YI1	YLR423C
Intersection	YLR423C

```
print("Hubs by betweennes in Y2H_uni_lcc")
```

```
## [1] "Hubs by betweennes in Y2H_uni_lcc"
```

```
bw_Y2H_uni_lcc
```

```
##  [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"
```

```
print("Hubs by betweennes in CCSB_YI1_lcc")
```

```
## [1] "Hubs by betweennes in CCSB_YI1_lcc"
```

```
bw_CCSB_YI1_lcc
```

```
##  [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"
```

Sol3b

I created additional 4 networks by removing 10% and 25% of edges randomly from Y2H and CCSB networs. Then, I used the R package *MCL* to do the MCL clustering. Table 10. and 11 shows the number of clusters which I got from these different networks. I further found the largest clusters in these networks and compared then within each group that is Y2H_uni and CCSB_YI1 See Table 12. For the three networks in Y2H_uni group i.e. the largest connected component of Y2H_uni and two networks which were obtaind by removing 10% and 25% edges, the largest clusters have 8 common genes. In CCSB there was only one common gene, *YPR113w*, among the three largest clusters which was also present in largest clusters of Y2H_uni networks. I also found intersection of largest clusters of Y2H_uni and CCSB_YI1 networks and found that only two genes over lap i.e. *YHL004w* and *YPR113w*. I also observed that after removing edges randomly the size of largest cluster became larger and total number of cluster went down. To see complete list of these genes please see the code snippet.

```
# run MCL on networks
adj_Y2H_uni_lcc <- as_adj(graph_Y2H_uni_lcc, type = "both", attr = NULL, edges = FALSE,
    names = TRUE)
Y2H_uni_lcc_mcl <- mcl(x = adj_Y2H_uni_lcc, addLoops = TRUE, ESM = TRUE)
# randomly remove 10% and 25% edges
graph_Y2H_uni_lcc_10 <- delete.edges(graph_Y2H_uni_lcc, head(sample(E(graph_Y2H_uni_lcc)),
    ecount(graph_Y2H_uni_lcc) * 0.1))
graph_Y2H_uni_lcc_25 <- delete.edges(graph_Y2H_uni_lcc, head(sample(E(graph_Y2H_uni_lcc)),
    ecount(graph_Y2H_uni_lcc) * 0.25))
adj_Y2H_uni_lcc_10 <- as_adj(graph_Y2H_uni_lcc_10, type = "both", attr = NULL,
    edges = FALSE, names = TRUE)
adj_Y2H_uni_lcc_25 <- as_adj(graph_Y2H_uni_lcc_25, type = "both", attr = NULL,
    edges = FALSE, names = TRUE)
Y2H_uni_lcc_mcl_10 <- mcl(x = adj_Y2H_uni_lcc_10, addLoops = TRUE, ESM = TRUE)
Y2H_uni_lcc_mcl_25 <- mcl(x = adj_Y2H_uni_lcc_25, addLoops = TRUE, ESM = TRUE)
head10 <- c("Y2H_uni_lcc", "Y2H_uni_lcc_10", "Y2H_uni_lcc_25")
tab10 <- data.frame(Network = head10, Num_clusters = c(Y2H_uni_lcc_mcl$K, Y2H_uni_lcc_mcl_10$K,
    Y2H_uni_lcc_mcl_25$K), Num_iter = c(Y2H_uni_lcc_mcl$n.iterations, Y2H_uni_lcc_mcl_10$n.iterations,
    Y2H_uni_lcc_mcl_25$n.iterations))
tab10 %>% knitr::kable(caption = "MCL result on Y2H's largest CC")
```

Table 10: MCL result on Y2H's largest CC

Network	Num_clusters	Num_iter
Y2H_uni_lcc	423	67
Y2H_uni_lcc_10	418	65
Y2H_uni_lcc_25	392	64

```
# for CCSB network
adj_CCSB_YI1_lcc <- as_adj(graph_CCSB_YI1_lcc, type = "both", attr = NULL, edges = FALSE,
    names = TRUE)
CCSB_YI1_lcc_mcl <- mcl(x = adj_CCSB_YI1_lcc, addLoops = TRUE, ESM = TRUE)
# randomly remove 10% and 25% edges
graph_CCSB_YI1_lcc_10 <- delete.edges(graph_CCSB_YI1_lcc, head(sample(E(graph_CCSB_YI1_lcc)),
    ecount(graph_CCSB_YI1_lcc) * 0.1))
```

```
graph_CCSB_YI1_lcc_25 <- delete.edges(graph_CCSB_YI1_lcc, head(sample(E(graph_CCSB_YI1_lcc)),
    ecount(graph_CCSB_YI1_lcc) * 0.25))
adj_CCSB_YI1_lcc_10 <- as_adj(graph_CCSB_YI1_lcc_10, type = "both", attr = NULL,
    edges = FALSE, names = TRUE)
adj_CCSB_YI1_lcc_25 <- as_adj(graph_CCSB_YI1_lcc_25, type = "both", attr = NULL,
    edges = FALSE, names = TRUE)
CCSB_YI1_lcc_mcl_10 <- mcl(x = adj_CCSB_YI1_lcc_10, addLoops = TRUE, ESM = TRUE)
CCSB_YI1_lcc_mcl_25 <- mcl(x = adj_CCSB_YI1_lcc_25, addLoops = TRUE, ESM = TRUE)
head11 <- c("CCSB_YI1_lcc", "CCSB_YI1_lcc_10", "CCSB_YI1_lcc_25")
tab11 <- data.frame(Network = head11, Num_clusters = c(CCSB_YI1_lcc_mcl$K, CCSB_YI1_lcc_mcl_10$K,
    CCSB_YI1_lcc_mcl_25$K), Num_iter = c(CCSB_YI1_lcc_mcl$n.iterations, CCSB_YI1_lcc_mcl_10$n.iterations
    CCSB_YI1_lcc_mcl_25$n.iterations))
tab11 %>% knitr::kable(caption = "MCL result on CCSB_YI1's largest CC")
```

Table 11: MCL result on CCSB_YI1's largest CC

Network	Num_clusters	Num_iter
CCSB_YI1_lcc	232	20
CCSB_YI1_lcc_10	231	69
CCSB_YI1_lcc_25	224	21

```
# find largest clusters in Y2h components
findlargestcluster <- function(graph, mcldata) {
    res <- 0
    cres <- V(graph)[which(mcldata$Cluster %in% c(0))]
    t <- mcldata$K
    for (i in c(0:t)) # print (i)
    thiscluster <- V(graph)[which(mcldata$Cluster %in% c(i))]$name
    # print (length(thiscluster))
    if (length(thiscluster) > length(cres)) {
        cres <- thiscluster
        res <- i
    }
    return(cres)
}
print("Largest clusters in Y2H_uni_lcc networks")
```

```
## [1] "Largest clusters in Y2H_uni_lcc networks"
```

```
lc_Y2H_uni_lcc_mcl <- findlargestcluster(graph_Y2H_uni_lcc, Y2H_uni_lcc_mcl)
lc_Y2H_uni_lcc_mcl
```

```
## + 16/1647 vertices, named, from 05526ff:
##  [1] YAL032C YER174C YGR099W YGR268C YHL004W YIL109C YIL122W YIL144W
##  [9] YJR011C YKL103C YLR321C YML015C YMR314W YNL086W YPL124W YPR113W
```

```
lc_Y2H_uni_lcc_mcl_10 <- findlargestcluster(graph_Y2H_uni_lcc_10, Y2H_uni_lcc_mcl_10)
lc_Y2H_uni_lcc_mcl_10
```

```
## + 91/1647 vertices, named, from d59038a:
##  [1] YAL040C  YBL035C  YBL102W  YBR166C  YBR186W  YCL024W  YCR022C
##  [8] YCR087CA YDL055C  YDL073W  YDL217C  YDR017C  YDR052C  YDR159W
## [15] YDR190C  YDR428C  YER062C  YER188W  YGL116W  YGR004W  YGR092W
## [22] YGR136W  YGR146C  YGR163W  YGR213C  YHL004W  YHR014W  YHR096C
```

```
## [29] YIL018W  YIL053W  YIL084C  YIL109C  YIL144W  YJL030W  YJL112W
## [36] YJL138C  YJR011C  YJR097W  YJR136C  YKL038W  YKL103C  YKL192C
## [43] YKR034W  YLR170C  YLR206W  YML015C  YMR121C  YMR314W  YMR317W
## [50] YNL044W  YNL086W  YNL263C  YNR004W  YPR113W  YLR096W  YLR032W
## [57] YDR239C  YMR255W  YGL247W  YGR140W  YFL003C  YKL003WA YPL235W
## [64] YJR117W  YGR129W  YJR110W  YPR051W  YOR385W  YLR319C  YNL016W
## + ... omitted several vertices
```

```
lc_Y2H_uni_lcc_mcl_25 <- findlargestcluster(graph_Y2H_uni_lcc_25, Y2H_uni_lcc_mcl_25)
lc_Y2H_uni_lcc_mcl_25
```

```
## + 239/1647 vertices, named, from d5949e6:
##   [1] YAL021C  YAL032C  YAL049C  YBL014C  YBL045C  YBL051C  YBR040W
##   [8] YBR111C  YBR122C  YBR126WA YBR134W  YBR154C  YBR155W  YBR187W
##  [15] YBR190W  YBR198C  YBR223C  YBR264C  YBR270C  YBR278W  YCL024W
##  [22] MEL1     TORF19   TORF21   YCR004C  YCR011C  YCR036W  YCR050C
##  [29] YCR068W  YCR088W  YDL031W  YDL066W  YDL105W  YDL106C  YDL111C
##  [36] YDL118W  YDL127W  YDL168W  YDL195W  YDL217C  YDL224C  YDL236W
##  [43] YDL246C  YDR002W  YDR004W  YDR005C  YDR016C  YDR051C  YDR059C
##  [50] YDR070C  YDR098C  YDR159W  YDR189W  YDR218C  YDR366C  YDR388W
##  [57] YDR389W  YDR422C  YDR453C  YDR472W  YDR503C  YDR518W  YDR529C
##  [64] YDR541C  YEL012W  YER025W  YER039C  YER112W  YER130C  YER157W
## + ... omitted several vertices
```

```
lc_Y2H <- intersection(lc_Y2H_uni_lcc_mcl, lc_Y2H_uni_lcc_mcl_10, lc_Y2H_uni_lcc_mcl_25)
print("Intersection of largest clusters in Y2H_uni_lcc networks")
```

```
## [1] "Intersection of largest clusters in Y2H_uni_lcc networks"
```

```
lc_Y2H
```

```
## + 3/1647 vertices, named, from d5949e6:
## [1] YHL004W YIL144W YMR314W
```

```
print("Largest clusters in CCSB_YI1_lcc networks")
```

```
## [1] "Largest clusters in CCSB_YI1_lcc networks"
```

```
lc_CCSB_YI1_lcc_mcl <- findlargestcluster(graph_CCSB_YI1_lcc, CCSB_YI1_lcc_mcl)
lc_CCSB_YI1_lcc_mcl
```

```
## + 6/964 vertices, named, from 0558f24:
## [1] YPR113W YHL004W YDR520C YOR174W YDR532C YLR206W
```

```
lc_CCSB_YI1_lcc_mcl_10 <- findlargestcluster(graph_CCSB_YI1_lcc_10, CCSB_YI1_lcc_mcl_10)
lc_CCSB_YI1_lcc_mcl_10
```

```
## + 66/964 vertices, named, from 6b77958:
##  [1] YLR257W  YLR120WA YGR268C  YLR082C  YDR195W  YHL004W  YLR219W
##  [8] YCL021WA YIL053W  YJL048C  YNL075W  YER126C  TORF21   YHR131C
## [15] YCR004C  TORF19   YPR153W  YHR184W  YKL008C  YNR067C  YNL044W
## [22] YDR513W  YBR201W  YKL192C  YHR057C  YDR328C  YBR195C  YBR243C
## [29] YHR180W  YML101C  YDL184C  YOR288C  YMR233W  YNL042W  YDL064W
## [36] YGR057C  YLR006C  YGL065C  YGL240W  YGL247W  YEL072W  TORF1
## [43] YGL252C  YPR037C  YNR014W  YDL073W  YML018C  YKL212W  YDR116C
## [50] YJR135C  YKR022C  YOR174W  YJL112W  YBR170C  YLR206W  YBR133C
## [57] YOL070C  YOR355W  YDR123C  YBR284W  YPL203W  YNL127W  YDL235C
## [64] YOR347C  YJR130C  YGL015C
```

```
lc_CCSB_YI1_lcc_mcl_25 <- findlargestcluster(graph_CCSB_YI1_lcc_25, CCSB_YI1_lcc_mcl_25)
lc_CCSB_YI1_lcc_mcl_25
```

```
## + 156/964 vertices, named, from 6b784aa:
##   [1] YGR024C  YJL137C  YBR166C  YMR289W  YDL144C  YGR268C  YLR175W
##   [8] YNL138W  YPL077C  YGR082W  YMR052W  YNL329C  YJR021C  YAL046C
##  [15] YLR132C  YGR223C  YJL075C  YOR379C  YOL012C  YNL198C  YDL133CA
##  [22] YLL010C  YHL004W  YMR210W  YJR022W  YDR085C  YHR134W  YDR239C
##  [29] YDL209C  YBR102C  YHR113W  YLR219W  YMR033W  YOR385W  YIR034C
##  [36] YLR392C  YGR286C  YOR304CA YLR393W  YHR123W  YEL034W  YDR518W
##  [43] YGR262C  YLR208W  YHR072WA YNL063W  YGR119C  YCR088W  YMR257C
##  [50] YGL071W  YLR279W  YDR364C  YNL222W  YJR118C  YDR107C  YLR262C
##  [57] YIL122W  YDR520C  YGR130C  YPR153W  YMR059W  YIL105C  YNL149C
##  [64] YPL053C  YFL047W  YPR049C  YPR193C  YER188W  YOR283W  YJR136C
## + ... omitted several vertices
```

```
lc_CCSB <- intersection(lc_CCSB_YI1_lcc_mcl, lc_CCSB_YI1_lcc_mcl_10, lc_CCSB_YI1_lcc_mcl_25)
print("Intersection of largest clusters in CCSB_YI1_lcc networks")
```

```
## [1] "Intersection of largest clusters in CCSB_YI1_lcc networks"
```

```
lc_CCSB
```

```
## + 2/964 vertices, named, from 6b784aa:
## [1] YHL004W YLR206W
```

```
head12 <- c("lc_Y2H_uni_lcc_mcl", "lc_Y2H_uni_lcc_mcl_10", "lc_Y2H_uni_lcc_mcl_25",
    "lc_CCSB_YI1_lcc_mcl", "lc_CCSB_YI1_lcc_mcl_10", "lc_CCSB_YI1_lcc_mcl_25")
tab12 <- data.frame(ClusterIn = head12, Size_of_largest_cluster = c(length(lc_Y2H_uni_lcc_mcl),
    length(lc_Y2H_uni_lcc_mcl_10), length(lc_Y2H_uni_lcc_mcl_25), length(lc_CCSB_YI1_lcc_mcl),
    length(lc_CCSB_YI1_lcc_mcl_10), length(lc_CCSB_YI1_lcc_mcl_25)))
tab12 %>% knitr::kable(caption = "Comparision of clusters found by MCL")
```

Table 12: Comparision of clusters found by MCL

ClusterIn	Size_of_largest_cluster
lc_Y2H_uni_lcc_mcl	16
lc_Y2H_uni_lcc_mcl_10	91
lc_Y2H_uni_lcc_mcl_25	239
lc_CCSB_YI1_lcc_mcl	6
lc_CCSB_YI1_lcc_mcl_10	66
lc_CCSB_YI1_lcc_mcl_25	156

Sol 3c

I randomly chose the cluster number 21 in Y2H's largest connected component network. This cluster had 4 genes. Then, i checked to GO terms using https://www.yeastgenome.org/cgi-bin/GO/goTermFinder.pl. In the results I found that 4 out of 4 genes were associated with GO term binding and 2 genes were associated with Histone binding both cases p-value was lower that 0.05. It does seems like that this cluster of gene are involved in similar functions although rigorous analysis is required to further confirm this.

```
# get all genes in cluster 21
c1_graph_Y2H_uni_lcc <- as.data.frame(V(graph_Y2H_uni_lcc)[which(Y2H_uni_lcc_mcl$Cluster %in%
    c(21))]$name)
```

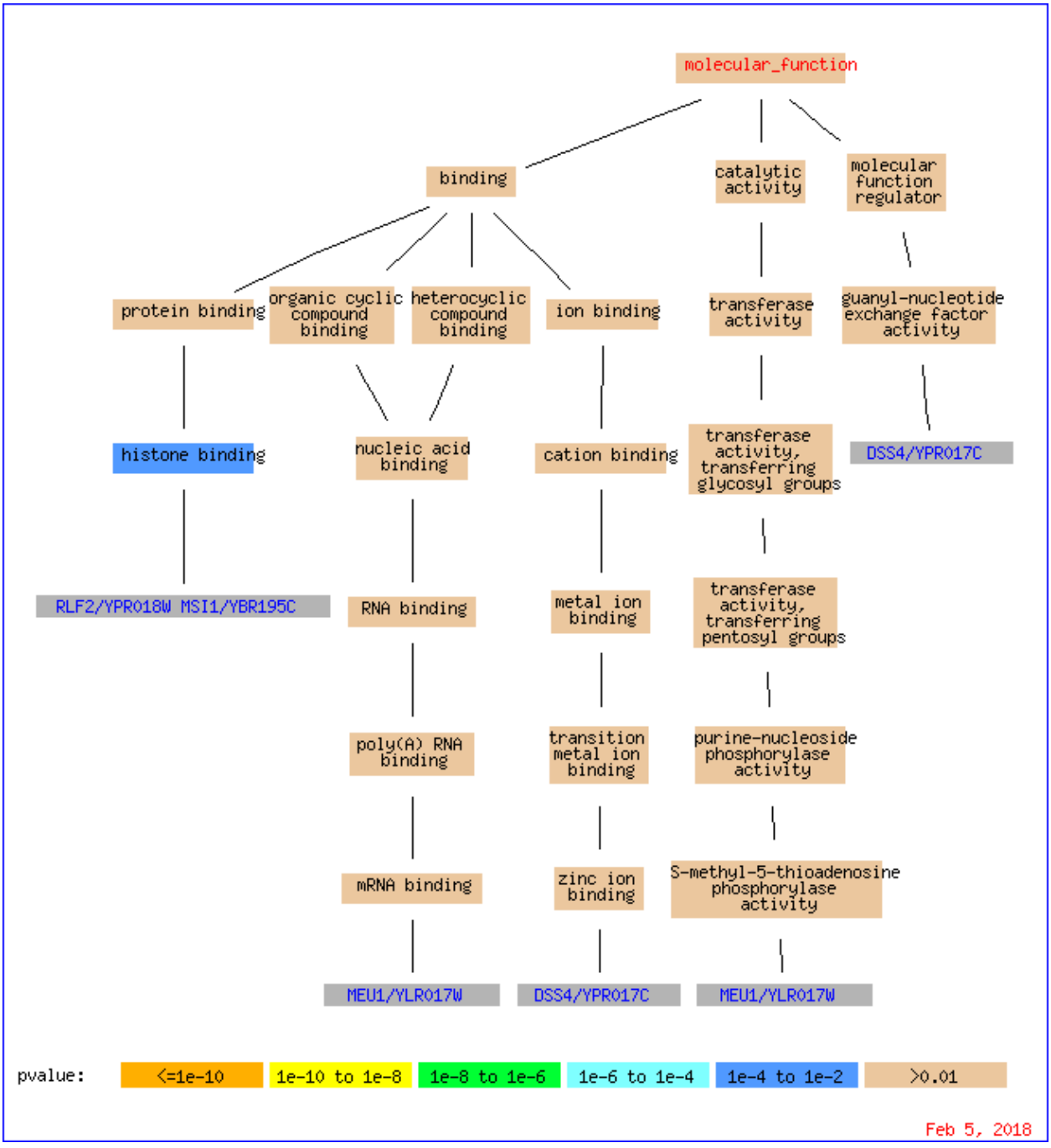

Figure 5: GO terms for genes in cluster 21

```
c1_graph_Y2H_uni_lcc
```

```
##   V(graph_Y2H_uni_lcc)[which(Y2H_uni_lcc_mcl$Cluster %in% c(21))]$name
## 1                                                                YBR195C
## 2                                                                YLR017W
## 3                                                                YPR018W
## 4                                                                YPR017C
```

```
library(readr)
GOres <- read_delim("GOres.txt", "\t", escape_double = FALSE, trim_ws = TRUE)
print("GO term search results:")
```

```
## [1] "GO term search results:"
```

```
GOres
```

```
## # A tibble: 3 x 8
##    GOID GO_term   `Cluster frequen~ `Background frequen~ `P-value`     FDR
##   <int> <chr>     <chr>             <chr>                    <dbl>   <dbl>
## 1 42393 histone ~ 2 out of 4 genes~ 41 out of 7165 back~  0.000570  0
## 2    NA <NA>      <NA>              <NA>                  NA        NA
## 3  5488 binding   4 out of 4 genes~ 1999 out of 7165 ba~  0.0181    0.0100
## # ... with 2 more variables: `Expected FP` <dbl>, `Gene(s) annotated to
## #   the term` <chr>
```