Yeast Genetic Interactions Network

Contents

1 Genetic Interactions

A genetic interaction (GI) between two genes generally indicates that the phenotype of a double mutant differs from what is expected from each individual mutant. The existence of a GI between two genes does not necessarily imply that these two genes code for interacting proteins or that the two genes are even expressed in the same cell. In fact, a GI only implies that the two genes share a functional relationship. These two genes may be involved in the same biological process or pathway; or they may also be involved in compensatory pathways with unrelated apparent function [Genetic interaction networks: better understand to better predict]

2 Data

Here we will explore and analyse the GI network of yeast. Data were downloaded from BIOGRID in December 2016 and were sorted by organism under the hyperlink: **BIOGRID-ORGANISM-3.4.143.tab2.zip**. We will use data from *Saccharomyces cerevisiae*. Files are in Tab2 format, so in order to import them in r, we first have to remove manually the text from the first part of the file.

The file from yeast contains many interactions. From them we have to retrieve the ones we are interested in, which are the ones resulting from Synthetic Genetic Arrays.

```
summary(yeast_BIOGRID$EXPERIMENTAL_SYSTEM)
```

```
## Affinity Capture-Luminescence           Affinity Capture-MS
##                            50                         58455
##          Affinity Capture-RNA      Affinity Capture-Western
```

```
##                        11166                        16569
##          Biochemical Activity          Co-crystal Structure
##                         6650                          881
##              Co-fractionation               Co-localization
##                          975                          675
##               Co-purification          Dosage Growth Defect
##                         4326                         1988
##              Dosage Lethality                Dosage Rescue
##                         1627                         5663
##                   Far Western                         FRET
##                          101                          219
##              Negative Genetic                          PCA
##                       115467                         6577
##        Phenotypic Enhancement        Phenotypic Suppression
##                         7487                         6590
##              Positive Genetic               Protein-peptide
##                        24681                          846
##                    Protein-RNA         Reconstituted Complex
##                          573                         7657
##       Synthetic Growth Defect  Synthetic Haploinsufficiency
##                        25200                          283
##            Synthetic Lethality              Synthetic Rescue
##                        16217                         6905
##                     Two-hybrid
##                        15973
```

These are

- Negative: **Synthetic Lethality** and **Negative Genetic**
- Positive: **Synthetic Rescue** and **Positive Genetic**

2.1 Extract GIN

These four types of genetic relations are defined based on their differentietion from un expected phenotype. The expected phenotype is

Synthetic Lethality means that the second mutation is lethal. **Negative Genetic** means that the phenotype of the two mutations is worse than expected. **Synthetic Rescue** means that even though the first mutation is leathal, the second mutation regains the viability of the organism. **Positive Genetic** means that the phenotype is better (highest fitness) than expected.

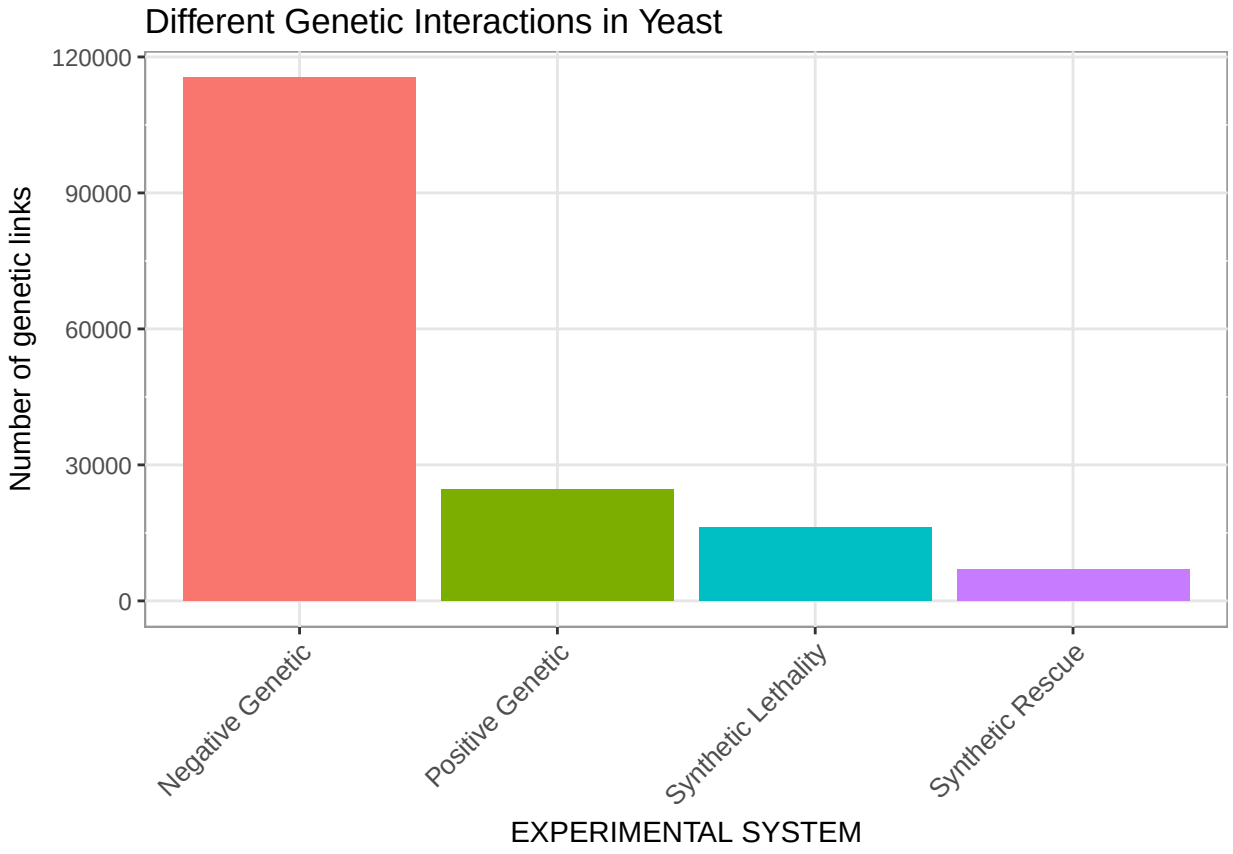
```
yeast_GIN <- subset(x = yeast_BIOGRID, EXPERIMENTAL_SYSTEM=="Synthetic Lethality" | EXPE
yeast_GIN$weights <- with(yeast_GIN, ifelse(EXPERIMENTAL_SYSTEM=="Synthetic Lethality" |

rm(yeast_BIOGRID)
```

```
summary(yeast_GIN$EXPERIMENTAL_SYSTEM)
```

```
## Affinity Capture-Luminescence           Affinity Capture-MS
##                             0                             0
##          Affinity Capture-RNA      Affinity Capture-Western
##                             0                             0
##          Biochemical Activity          Co-crystal Structure
##                             0                             0
##              Co-fractionation               Co-localization
##                             0                             0
##               Co-purification          Dosage Growth Defect
##                             0                             0
##              Dosage Lethality                 Dosage Rescue
##                             0                             0
##                   Far Western                          FRET
##                             0                             0
##              Negative Genetic                           PCA
##                        115467                             0
##        Phenotypic Enhancement        Phenotypic Suppression
##                             0                             0
##              Positive Genetic               Protein-peptide
##                         24681                             0
##                   Protein-RNA         Reconstituted Complex
##                             0                             0
##       Synthetic Growth Defect  Synthetic Haploinsufficiency
##                             0                             0
##           Synthetic Lethality              Synthetic Rescue
##                         16217                          6905
##                   Two-hybrid
##                             0
```

The bar plot of the 4 types of relations.

```
ggplot()+
  geom_bar(data = yeast_GIN, aes(EXPERIMENTAL_SYSTEM, y= ..count.., fill=EXPERIMENTAL_SY
  ggtitle("Different Genetic Interactions in Yeast")+
  labs(x="EXPERIMENTAL SYSTEM", y= "Number of genetic links")+
  theme(panel.grid.major = element_line(colour = "grey90"), panel.background = element_
        axis.text.x = element_text(size  = 10, angle = 45,hjust = 1,vjust = 1))
```



2.2 Duplicates and sign consistency

We check for the unique links.In order to check this we first concatenate the node IDs for each link and then we check for uniqueness.

```
yeast_GIN$links <- with(yeast_GIN,paste(yeast_GIN$INTERACTOR_A,yeast_GIN$INTERACTOR_B,se
length(unique(yeast_GIN$links)) # how many unique links
```

```
## [1] 144166
```

But the number of links is:

```
nrow(yeast_GIN)
```

```
## [1] 163270
```

So there are 19104 duplicates. Next we want to label each link to see if it's duplicated and after that we count the duplicates of each link in order to find the origin of the duplicates.

```
yeast_GIN$duplicate <- duplicated(yeast_GIN$links) | duplicated(yeast_GIN$links, fromLas

yeast_GIN_dublicates <- subset(x = yeast_GIN, yeast_GIN$duplicate==TRUE)
# count the number of occurences of each link
```

```r
GIN_occurences <- yeast_GIN %>%
  group_by(links) %>%
  summarise (n = n())

#summary(GIN_occurences)
```

```r
GIN_occurences$link <- c(seq(1:length(GIN_occurences$links)))
GIN_dupl_freq<-aggregate(rep.int(1, length(GIN_occurences$link))~GIN_occurences$n, FUN=s
names(GIN_dupl_freq)<-c("Copies per link","Count of links")
kable(GIN_dupl_freq, align = 'c', caption = "The number of copies of each link in the da
```

Table 1: The number of copies of each link in the data yeast GIN

Copies per link	Count of links
1	129604
2	11381
3	2331
4	593
5	148
6	55
7	25
8	8
9	8
10	2
11	5
12	2
13	1
14	1
16	2

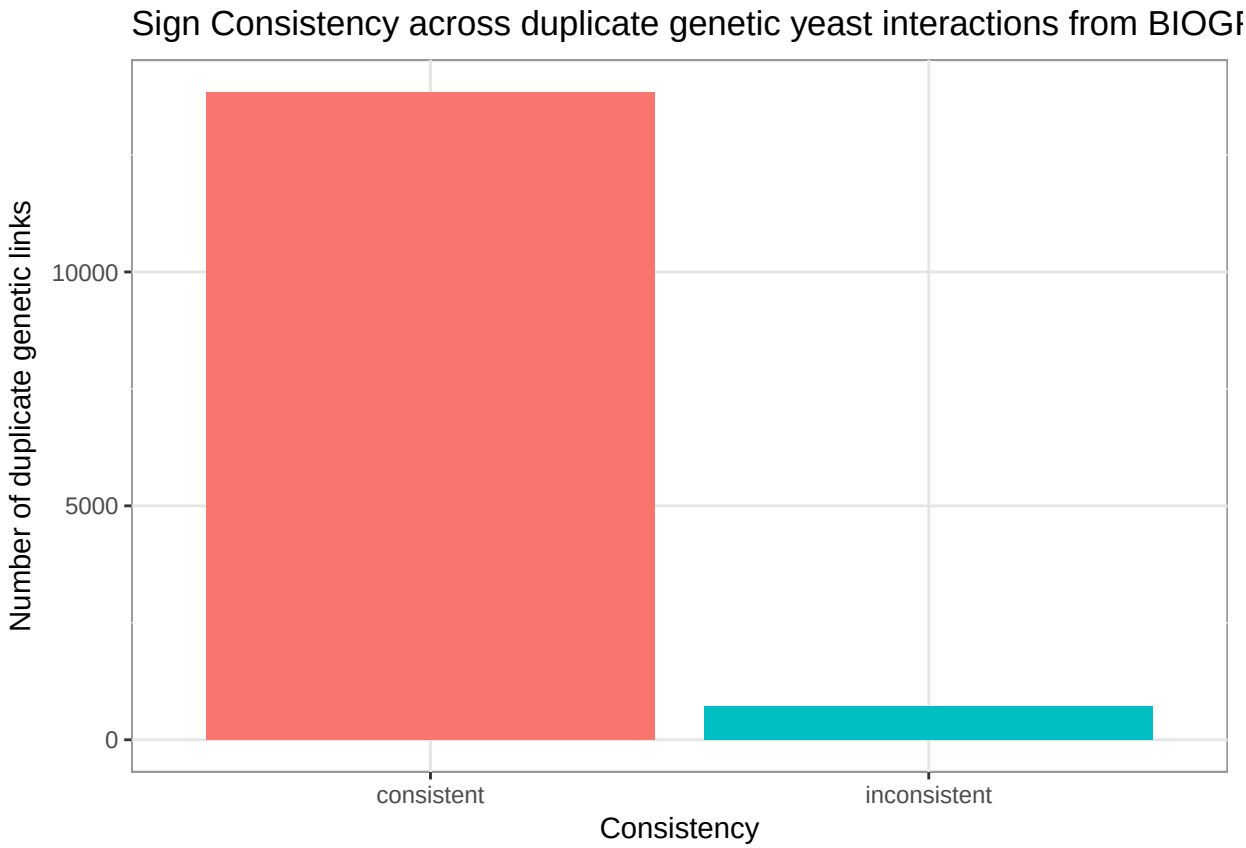
So some links appear multiple times. The link between YCR066W and YJL092W appears 16 times in the dataset. This is due to the different sources which discoved an already known genetic relation. Next we have to delete duplicates. In order to do so we have to examine the nature of the genetic interaction for each duplicate. If between duplicates there are contradictions of the sign of the genetic interaction we have to delete the interaction. If not, we'll keep one version of the interaction.

```r
## duplicated links

#sort links so the duplicates are adjacent?
duplicated_interactions <- yeast_GIN_dublicates[order(yeast_GIN_dublicates$links),]
#yeast_GIN2 <- head(duplicated_interactions,n = 200)
yeast_GIN_unique_dup <- as.data.frame(matrix())
```

```
yeast_GIN_unique_dup <- as.data.frame(unique(duplicated_interactions$links))
yeast_GIN_unique_dup2 <- as.data.frame(yeast_GIN_unique_dup)

ptm <- proc.time()
for(i in 1:nrow(yeast_GIN_unique_dup)){

  tmp_dupl_links <- subset(yeast_GIN_dublicates,yeast_GIN_dublicates$links==yeast_GIN_un

  if (abs(sum(tmp_dupl_links$weights))==nrow(tmp_dupl_links)){

    yeast_GIN_unique_dup2[i,2] <- "consistent"
    #print(paste0("consistent ",yeast_GIN_unique_dup[i,1]))

    } else {
      yeast_GIN_unique_dup2[i,2] <- "inconsistent"
      #print(paste("inconsistent ",yeast_GIN_unique_dup[i,1]))
    }

  yeast_GIN_unique_dup2[i,3] <- length(which(tmp_dupl_links$weights>0))#Positive_Intera
  yeast_GIN_unique_dup2[i,4] <- length(which(tmp_dupl_links$weights<0)) #Positive_Inter
  tmp_dupl_links <- c()

}

colnames(yeast_GIN_unique_dup2) <- c("Dublicated_Interactions","Consistency","Positive_I
proc.time() - ptm
```

```
##    user  system elapsed
##  43.073   7.770  52.446
```

```
#consistency
ggplot()+
  geom_bar(data = yeast_GIN_unique_dup2, aes(Consistency, y= ..count.., fill=Consistency
  ggtitle("Sign Consistency across duplicate genetic yeast interactions from BIOGRID")+
  labs(x="Consistency", y= "Number of duplicate genetic links")+
  theme(panel.grid.major = element_line(colour = "grey90"), panel.background = element_
```




```
count_consistency_GIN <- yeast_GIN_unique_dup2 %>%
  group_by(Consistency) %>%
  summarise (n = n())


kable(count_consistency_GIN, align = 'c', caption = "Sign consistency across duplicated
```

Table 2: Sign consistency across duplicated genetic interactions in yeast GIN

Consistency	n
consistent	13844
inconsistent	718

Next we have to keep only the unique interactions and remove the inconsistent interactions from the original dataset.

```
ddd <- subset(yeast_GIN_unique_dup2,yeast_GIN_unique_dup2$Consistency=="inconsistent")

yeast_GIN$Consistency <- !(yeast_GIN$links %in% ddd$Dublicated_Interactions)
yeast_GIN$duplicate_not_first <- duplicated(yeast_GIN$links)
yeast_GIN_final <- subset(yeast_GIN, yeast_GIN$duplicate_not_first==FALSE)
```

```
yeast_GIN_final <- subset(yeast_GIN_final, yeast_GIN_final$Consistency==TRUE)


nrow(yeast_GIN_final)==length(unique(yeast_GIN_final$links)) # check unique links
```

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

```
nrow(yeast_GIN_final)
```

```
## [1] 143448
```

```
a <- as.character(yeast_GIN_final$INTERACTOR_A)
b <- as.character(yeast_GIN_final$INTERACTOR_B)
yeast_GIN_genes <- as.data.frame(unique(c(a,b)))
yeast_GIN_genes <- yeast_GIN_final$INTERACTOR_B #as.factor(yeast_GIN_final$INTERACTOR_B

nrow(yeast_GIN_genes)
```

```
## NULL
```

The removal of duplicates and the complete deletion of links with duplicates and inconsistent signs leads to the final edgelist. The final edgelist of yeast GIN consists of **143448** interactions and **5433** genes.

```
count_weights_GIN <- yeast_GIN_final %>%
  group_by(weights) %>%
  summarise (n = n())


kable(count_weights_GIN, align = 'c', caption = "Weight frequences of the yeast GIN")
```

Table 3: Weight frequences of the yeast GIN

weights	n
-1	114864
1	28584

2.3 Loops

Next, we need to examine if there are any loops in the network. Which means that we have to look for interactions that have the same gene in edge list.

```
loop_finder <- function(x){
  d<- c()
  for(i in 1:nrow(x)){
```

```
    if(as.character(x[i,1])==as.character(x[i,2])){
      d <- c(d,as.character(x[i,1]))
    }else{}
  }
    d <-as.data.frame(d)
    colnames(d)[1]<-"genes_loop"
    d
  }
loops <- loop_finder(yeast_GIN_final)
nrow(loops)
```

```
## [1] 8
```

There are 8 loops in the yeast GIN. What do they mean? Do they have biological meaning?

3 Network analysis

Import the network in igraph. We choose to delete the loops (8).

```
library(igraph)

yeast_GIN_Net <- as.data.frame(yeast_GIN_final[,-c(3:11,13:16)])
g_yeast_GIN <- graph_from_data_frame(yeast_GIN_Net,directed = T)
summary(g_yeast_GIN)
```

```
## IGRAPH DN-- 5433 143448 --
## + attr: name (v/c), weights (e/n)
```

```
is.simple(g_yeast_GIN)
```

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

```
#It has multiple links?
E(g_yeast_GIN)$multiple <- which_multiple(g_yeast_GIN)
summary(E(g_yeast_GIN)$multiple) # all FALSE
```

```
##    Mode   FALSE    NA's
## logical  143448       0
```

```
# It has loops?
E(g_yeast_GIN)$loop <- which_loop(g_yeast_GIN)
summary(E(g_yeast_GIN)$loop) # 8 TRUE loops
```

```
##    Mode   FALSE    TRUE    NA's
## logical  143440       8       0
```

```
#Remove loops
g_yeast_GIN <- igraph::simplify(g_yeast_GIN)
# Assign attributes
yeast_GIN_Net$sign_color <- with(yeast_GIN_Net, ifelse(weights<0, paste0("red"), paste0
E(g_yeast_GIN)$weights <- yeast_GIN_Net$weights
E(g_yeast_GIN)$sign_color <- yeast_GIN_Net$sign_color
is.simple(g_yeast_GIN)
```

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

```
summary(g_yeast_GIN)
```

```
## IGRAPH DN-- 5433 143440 --
## + attr: name (v/c), weights (e/n), sign_color (e/c)
```

Create the adjacency matrix.

```
adjacency_weight_yeast_GIN <- as_adjacency_matrix(g_yeast_GIN, attr = "weights",names =
adjacency_weight_yeast_GIN <- as.data.frame(as.matrix(adjacency_weight_yeast_GIN))
sum(sapply(adjacency_weight_yeast_GIN,is.character)) # if 0 then there are no character
```

```
## [1] 0
```

```
#write.table(adjacency_weight_yeast_GIN,file = "adjacency_matrix_yeast_GIN.txt",sep =
```

To make further calculations we isolate the giant component.

```
igraph::is.connected(g_yeast_GIN)
```

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

```
decg<-decompose.graph(g_yeast_GIN, min.vertices = 10) #upografima tis megalis sinistosa
gcomp_GIN<-decg[[1]]
igraph::is.connected(gcomp_GIN)
```

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

```
summary(gcomp_GIN)
```

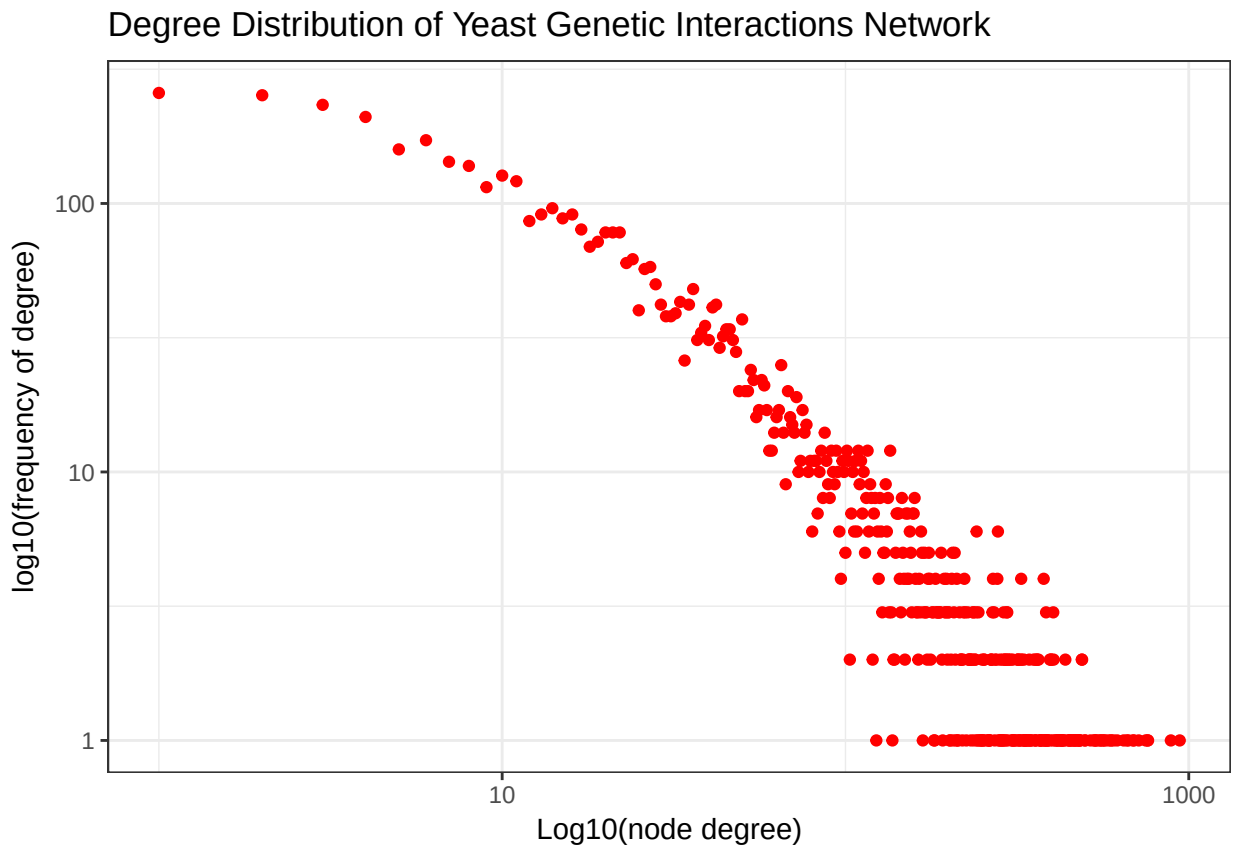
```
## IGRAPH DN-- 5422 143431 --
## + attr: name (v/c), weights (e/n), sign_color (e/c)
```

Only 11 genes and 9 links weren't connected with the giant component.

3.1 Degree distribution

One of the first things to examine is the degree distribution of the graph.

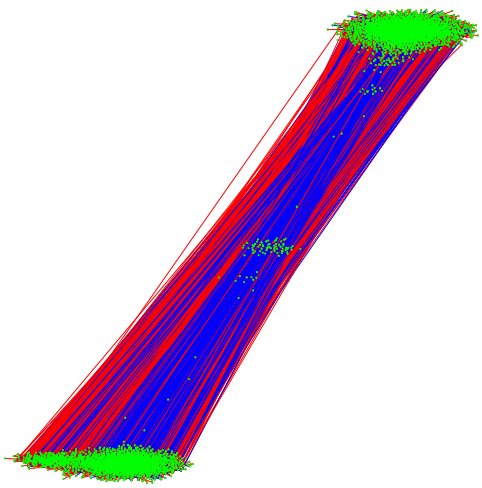
```r
# Degree distribution of the graph

de <- igraph::degree(gcomp_GIN)
de <- as.data.frame(de)
de$node <- c(seq(1:length(de$de)))
dd<-aggregate(rep.int(1, length(de$node))~de$de, FUN=sum)
names(dd)<-c("val","freq")

ggplot()+
  geom_point(data = dd, aes(x = val,y = freq ),color="red")+
  scale_y_log10()+
  scale_x_log10()+
  ggtitle("Degree Distribution of Yeast Genetic Interactions Network")+
  labs(x="Log10(node degree)", y="log10(frequency of degree)")+
  theme_bw()
```



Plot of the network.

```r
layg <- layout_nicely(gcomp_GIN)
V(gcomp_GIN)$coord1 <-layg[ ,1]     ## network plots
V(gcomp_GIN)$coord2 <-layg[ ,2]
```

```
plot(gcomp_GIN,
     layout=layg,
     vertex.shape="circle",
     vertex.size=0.0095,
     vertex.color="green",
     vertex.frame.color="green",
     vertex.label=NA,
     #vertex.label.cex = vcount(g)*0.00004,
     #vertex.label.color ="black",
     edge.color=E(gcomp_GIN)$sign_color,
     edge.width=0.03,
     edge.arrow.size = 0.00001,
     edge.arrow.width = 0.0000005,
     #edge.label=round(x = E(g)$weight,digits = 3),
     #edge.label.cex = 0.3,
     #edge.label.color = E(g)$color,
     margin = 0,
     main = "",
     sub = "")
title(main = paste("yeast GIN network "), cex.main= 1.5, cex.sub = 1.2,outer = F)
```




```
#sss <- induced_subgraph(gcomp_GIN, v = nei[[1]])

#plot(sss,edge.color=E(gcomp_GIN)$sign_color)

#nei <- neighborhood(gcomp_GIN,order = 1,nodes = "YOL001W")

yeast_GIN_final$AD <- paste0(yeast_GIN_final[,1],",",yeast_GIN_final[,12])
```

```
yeast_GIN_final$BD <- paste0(yeast_GIN_final[,2],",",yeast_GIN_final[,12])

AD_freq <- table(yeast_GIN_final$AD)
AD_freq <- as.data.frame(AD_freq)
BD_freq <- table(yeast_GIN_final$BD)


AD_freq_a <- strsplit(as.character(AD_freq$Var1),split = ",")
AD_freq_a <- as.data.frame(AD_freq_a)
AD_freq_a <- t(AD_freq_a)

AD_freq$gene <- AD_freq_a[,1]
AD_freq$weight <- AD_freq_a[,2]

AD_freq_a_1 <- AD_freq[order(AD_freq$weight,AD_freq$Freq,decreasing = T),]
AD_freq_a_neg <- subset(x = AD_freq_a_1,AD_freq_a_1$weight==-1)
#AD_freq_a_1 <- AD_freq[order(AD_freq$Freq,decreasing = T),]
```