# Gene Correlation Network

Miguel Blanco Chillerón (mblancochi@gmail.com)

2024-06-04

```
library(affy)
```

```
## Loading required package: BiocGenerics
```

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

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

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

```
## Loading required package: Biobase
```

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

```
library(annaffy)
```

```
## Loading required package: BiocManager
```

```
## Warning: package 'BiocManager' was built under R version 4.2.3
```

```
## Bioconductor version '3.15' is out-of-date; the current release version '3.19'
##   is available with R version '4.4'; see https://bioconductor.org/install
```

```
## Loading required package: GO.db
```

```
## Loading required package: AnnotationDbi

## Loading required package: stats4

## Loading required package: IRanges

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

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

##
## Attaching package: 'IRanges'

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

##
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.2.3

##
## Attaching package: 'igraph'

## The following object is masked from 'package:IRanges':
##
##     union

## The following object is masked from 'package:S4Vectors':
##
##     union

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

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
library(treemap)
```

```
## Warning: package 'treemap' was built under R version 4.2.3
```

```r
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.2.3
```

## INTRODUCTION

### Study Objective

The objective of the study is to observe the genes and regulatory metabolism that is activated in plants when phosphate levels are low. The goal is to see what adaptations the plant implements to cope with these conditions and whether there are key genes that regulate and coordinate the plant's adaptation mechanisms. Additionally, it will be studied if there is any type of coexpression among the genes expressed under Pi deficiency.

Experimental Design, Description of Each Sample, Description of Each Condition, and Adequacy to Solve the Proposed Problem

We will have a total of 24 samples, half of which will be in a Pi-deficient medium (-Pi) and the other half in a Pi-containing medium serving as control (+Pi). We will also distinguish according to the hours the samples have been in those conditions at the time of sampling, distinguishing between 0 hours, 1 hour, 6 hours, and 24 hours. Finally, we will have 3 replicates for each condition. Therefore, we will have samples with and without Pi in the medium at 0, 1, 6, and 24 hours, making a total of 8 conditions with 3 replicates each, summing up to a total of 24 samples. This way, we can observe the gene expression that occurs in plants with Pi restriction and know that it is due to this by comparing it with the controls. We can also see how this expression evolves according to the time the organism has been under Pi restriction.

### Organism Used: Arabidopsis Thaliana

### Workflow

We process the microarray samples and obtain the gene expressions, group the replicates, and obtain the mean expressions for each condition. We will contrast between conditions and calculate the differentially expressed genes. Then, we calculate the correlation and construct the network by choosing the appropriate threshold while maintaining the average connectivity and a high R as possible. We will also designate those nodes with higher connectivity that are considered key in coregulation, which we will call HUBs. Once we have the network, we will use Hclust and Pam to divide the network into the optimal number of clusters. Once we have this, we will visualize the network in Cytoscape, distinguishing according to different node attributes such as the cluster in which they are located, the Hub score they possess, or others.

## GENE EXPRESSION ANALYSIS, CONSTRUCTION AND VISUALIZATION OF COEXPRESSION NETWORK

### Criterion for Choosing Differentially Expressed Genes

Once the gene expression contrasts between conditions are done, we use eBayes to rank the genes according to the statistical significance of differential expression. From these values, we will use the DEG.selection function, which will return the number of differentially expressed genes based on the values obtained by eBayes and the selected fold change. It will not specify whether these are activated or repressed genes, but for this case, it is not essential to know. The selected fold change will be 1, as chosen by the authors in the article and it seems the most appropriate.

```
microarrays<- ReadAffy(verbose = TRUE)
```

```
## 1 reading C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618324.CEL ...ins
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618324.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618325.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618326.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618327.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618328.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618329.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618330.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618331.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618332.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618333.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618334.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618335.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618336.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618337.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618338.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618339.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618340.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618341.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618342.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618343.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618344.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618345.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618346.CEL
## Reading in : C:/Users/Usuario/OneDrive/Escritorio/Bioinf/Gene_Coexpression_Network/GSM618347.CEL
```

```
microarrays.data <- {affy::rma(microarrays)}
```

```
## Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
## loading 'ath1121501cdf'
```

```
## Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
## loading 'ath1121501cdf'
```

```
##
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

```
expression.level <- exprs(microarrays.data)

dim(expression.level)
```

```
## [1] 22810    24
```

```
sampleID <- c("0_+Pi_1","0_-Pi_1","1_+Pi_1","1_-Pi_1","6_+Pi_1","6_-Pi_1",
              "24_+Pi_1","24_-Pi_1","0_+Pi_2","0_-Pi_2","1_+Pi_2",
              "1_-Pi_2","6_+Pi_2","6_-Pi_2",
              "24_+Pi_2","24_-Pi_2","0_+Pi_3","0_-Pi_3","1_+Pi_3",
              "1_-Pi_3","6_+Pi_3","6_-Pi_3","24_+Pi_3","24_-Pi_3")

colnames(expression.level) <- sampleID


Control_0 <- (expression.level[,"0_+Pi_1"]
              +expression.level[,"0_+Pi_2"] +expression.level[,"0_+Pi_3"])/3

Control_1 <- (expression.level[,"1_+Pi_1"]
              +expression.level[,"1_+Pi_2"] +expression.level[,"1_+Pi_3"])/3

Control_6 <- (expression.level[,"6_+Pi_1"]
              +expression.level[,"6_+Pi_2"] +expression.level[,"6_+Pi_3"])/3

Control_24 <- (expression.level[,"24_+Pi_1"]
               +expression.level[,"24_+Pi_2"] +expression.level[,"24_+Pi_3"])/3


NoPi_0 <- (expression.level[,"0_-Pi_1"]
           +expression.level[,"0_-Pi_2"] +expression.level[,"0_-Pi_3"])/3

NoPi_1 <- (expression.level[,"1_-Pi_1"]
           +expression.level[,"1_-Pi_2"] +expression.level[,"1_-Pi_3"])/3

NoPi_6 <- (expression.level[,"6_-Pi_1"]
           +expression.level[,"6_-Pi_2"] +expression.level[,"6_-Pi_3"])/3

NoPi_24 <- (expression.level[,"24_-Pi_1"]
            +expression.level[,"24_-Pi_2"] +expression.level[,"24_-Pi_3"])/3




mean.expression <- matrix(c(Control_0,Control_1,Control_6,
                            Control_24,NoPi_0,NoPi_1,NoPi_6,NoPi_24),nrow=8, ncol=length(NoPi_0),byrow='

colnames(mean.expression) <- names(Control_0)
rownames(mean.expression) <- c("Control_0", "Control_1",
                               "Control_6","Control_24","NoPi_0",
                               "NoPi_1","NoPi_6","NoPi_24")



library(limma)
```

```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
```

```
##
##     plotMA
```

```
experiment.design <- model.matrix(~ -1+factor(c(1,5,2,6,3,7,4,8,1,5,2
                                                ,6,3,7,4,8,1,5,2,6,3,7,4,8)))
colnames(experiment.design) <- c("Control_0", "Control_1","Control_6",
                                 "Control_24","NoPi_0","NoPi_1","NoPi_6","NoPi_24")

fit <- lmFit(expression.level, experiment.design)

contrast.matrix <- makeContrasts(NoPi_0-Control_0,NoPi_1-Control_1,
                                 NoPi_6-Control_6,NoPi_24-Control_24,
                                 levels=c("Control_0", "Control_1","Control_6",
                                          "Control_24","NoPi_0","NoPi_1","NoPi_6","NoPi_24"))







fit2 <- contrasts.fit(fit, contrast.matrix)
ebayes <- eBayes(fit2)
dim(mean.expression)
```

```
## [1]     8 22810
```

```
Pi.dif<- topTable(ebayes, number=22810,coef=1,sort.by="logFC")




DEG.selection <- function(processed.data,number.comparisons,
                          number.genes,fold.change.threshold,log.pvalue.threshold)
{

  DEGs <- character()

  for(i in 1:number.comparisons)
  {
    DEGs.partial <- topTable(processed.data, number=number.genes,coef=i)
    fold.change <- DEGs.partial[["logFC"]]
    log.p.value <- -log10(DEGs.partial[["adj.P.Val"]])
    probe.names <- rownames(DEGs.partial)

    ## Establish fold change for consider a gen as activated or repressed##
    activated <- (fold.change > fold.change.threshold) & (log.p.value > log.pvalue.threshold)
    inhibited <- (fold.change < - fold.change.threshold) & (log.p.value > log.pvalue.threshold)

    activated.genes <- probe.names[activated]
    inhibited.genes <- probe.names[inhibited]

    DEGs <- c(DEGs, activated.genes, inhibited.genes)
  }
```

```
  return(unique(DEGs))
}

#Differentialy expressed genes and matrix##

diff.expr.genes <- DEG.selection(processed.data=ebayes,number.comparisons=4,
                                 number.genes=22810,fold.change.threshold=1,log.pvalue.threshold=2)
length(diff.expr.genes)
```

```
## [1] 412
```

```
#Create diff.expr matrix, which gives the expression
#value of each differentialy expressed gene in each condition##

diff.expr <- mean.expression[,diff.expr.genes]
dim(diff.expr)
```

```
## [1]   8 412
```

### Determination of Gene Coexpression Matrix and Criteria to Measure Coexpression and the Criteria for Choosing Gene Coexpression

The fundamental decision of choosing a methodology to measure from the gene expression profiles obtained in the previous step the degree of coexpression between them. The criterion followed to determine if two genes are coexpressed under the conditions of the various experiments studied is based on the correlation between their expression profiles.

```
## Using the cor function, which from the expressions
#of the differentially expressed genes in the different conditions
#will give us the coexpression they present ##
gene.correlation <- cor(diff.expr)
dim(gene.correlation)
```

```
## [1] 412 412
```

### Selection of the Threshold and Network Creation, Graphs, and Visualization in Cytoscape

We must select a threshold from which we will assume that two genes present coexpression. A too high threshold will make the average connectivity drop as there will be little coexpression, while a too low threshold will make the R2 drop, as it will give as coexpressed genes that are not actually coexpressed.

We will establish a criterion in which the average connectivity must be greater than 10. To do this, we create a vector from 0.70 to 0.99 that goes through it every 0.01, returning the vector with all values. For each of these values, we will calculate the average connectivity and R2, and from there we choose the threshold. In this case, the threshold we will choose is 0.97, which gives us a connectivity of 10.19 and an R2 of 0.63.

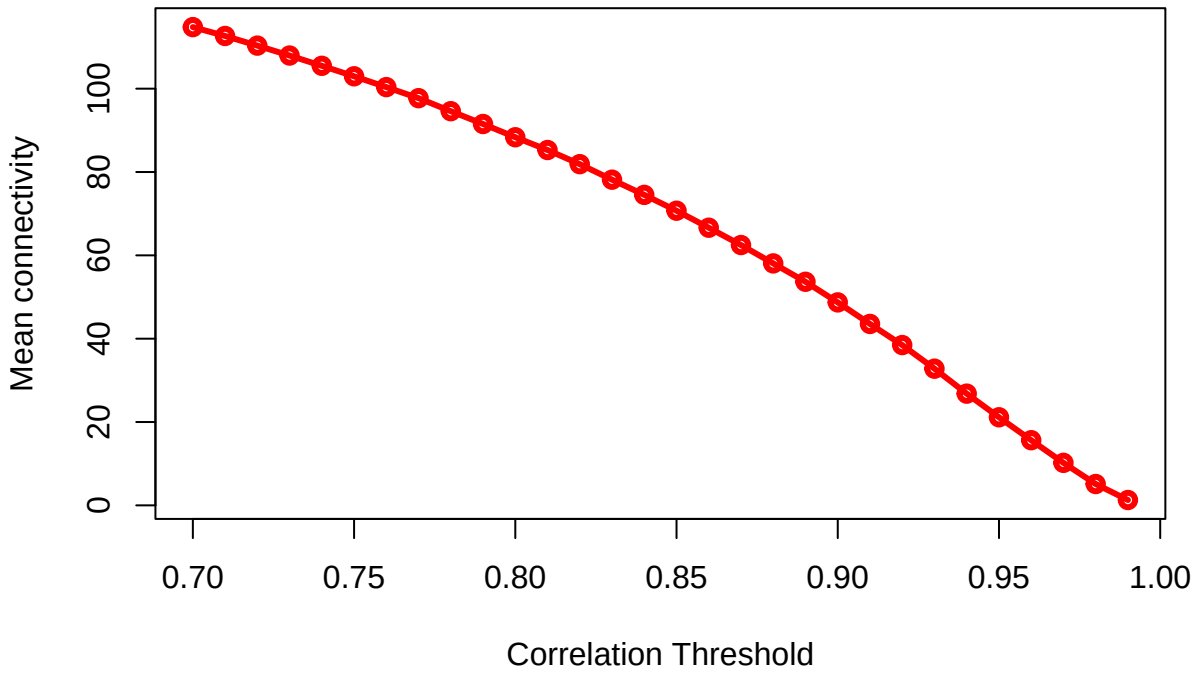
```
## We create the vector with all possible thresholds and two empty vectors that will collect the connec
thresholds <- seq(from=0.70, to=0.99, by=0.01)
mean.connectivities <- vector(length =length(thresholds))
scale.free.R2 <- vector(length = length(thresholds))
```

```
## Loop to calculate for each threshold connectivity and R2 ##
for(i in 1:length(thresholds)) {
  print(thresholds[i])
  ## Gene correlation less than 1 to not include correlations between a gene and itself (which is 1) ##
  current.adjacency <- (gene.correlation > thresholds[i] & gene.correlation < 1)

  threshold.network <- graph.adjacency(current.adjacency, mode="undirected")

  node.degrees <- degree(threshold.network)

  mean.connectivities[i] <- mean(node.degrees)

  degree.frequencies <- table(node.degrees)

  lm.r <- lm(log10(as.numeric(names(degree.frequencies[-1]))) ~ log10(degree.frequencies[-1]))

    s.lm <- summary(lm.r)
  scale.free.R2[i] <- s.lm[["adj.r.squared"]]
}
```

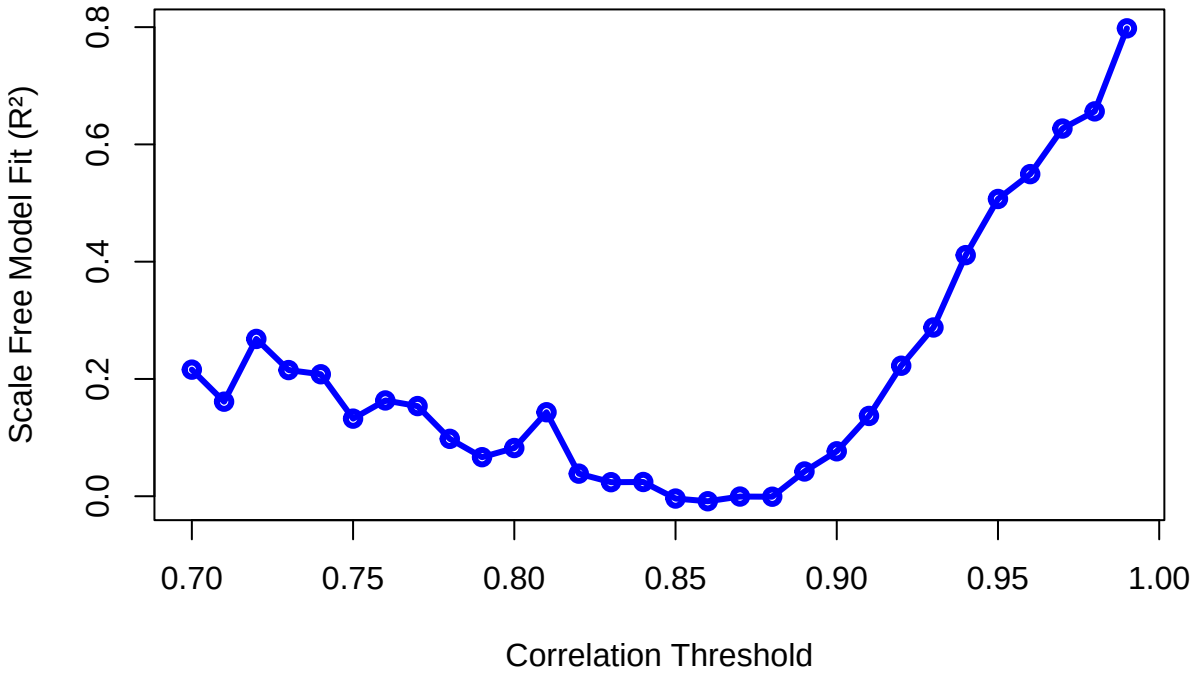
```
## [1] 0.7
## [1] 0.71
## [1] 0.72
## [1] 0.73
## [1] 0.74
## [1] 0.75
## [1] 0.76
## [1] 0.77
## [1] 0.78
## [1] 0.79
## [1] 0.8
## [1] 0.81
## [1] 0.82
## [1] 0.83
## [1] 0.84
## [1] 0.85
## [1] 0.86
## [1] 0.87
## [1] 0.88
## [1] 0.89
## [1] 0.9
## [1] 0.91
## [1] 0.92
## [1] 0.93
## [1] 0.94
## [1] 0.95
## [1] 0.96
## [1] 0.97
## [1] 0.98
## [1] 0.99
```

Plots to show how R2 and average connectivity evolve according to the threshold and assign each threshold value the corresponding average connectivity and R2

```
plot(thresholds,mean.connectivities,type="o",col="red",lwd=3,xlab="Correlation Threshold",ylab="Mean co
```




```
plot(thresholds,scale.free.R2,type="o",col="blue",lwd=3,xlim=c(0.70,0.99),
     xlab="Correlation Threshold",ylab="Scale Free Model Fit (R²)")
```




```
names(mean.connectivities) <- thresholds
names(scale.free.R2) <- thresholds

mean.connectivities
```

```
##        0.7       0.71       0.72       0.73       0.74       0.75       0.76
## 114.771845 112.650485 110.330097 107.951456 105.490291 102.975728 100.373786
##       0.77       0.78       0.79        0.8       0.81       0.82       0.83
##  97.718447  94.592233  91.524272  88.344660  85.276699  81.883495  78.160194
##       0.84       0.85       0.86       0.87       0.88       0.89        0.9
##  74.509709  70.708738  66.631068  62.475728  58.067961  53.655340  48.708738
##       0.91       0.92       0.93       0.94       0.95       0.96       0.97
##  43.538835  38.466019  32.796117  26.810680  21.135922  15.631068  10.199029
##       0.98       0.99
##   5.116505   1.281553
```

```
scale.free.R2
```

```
##           0.7          0.71          0.72          0.73          0.74
##  0.2158895296  0.1612478830  0.2682719494  0.2151381915  0.2080221262
##          0.75          0.76          0.77          0.78          0.79
##  0.1323300996  0.1633271202  0.1537607338  0.0978821157  0.0665799915
##           0.8          0.81          0.82          0.83          0.84
##  0.0821998522  0.1432357396  0.0385224599  0.0239631437  0.0242775541
##          0.85          0.86          0.87          0.88          0.89
```

```
## -0.0039582020 -0.0085406668 -0.0005395368 -0.0008435962  0.0421857402
##           0.9          0.91          0.92          0.93          0.94
##  0.0766190843  0.1369142433  0.2225376768  0.2877545351  0.4111638601
##          0.95          0.96          0.97          0.98          0.99
##  0.5069451037  0.5493159477  0.6270428950  0.6563152964  0.7979246099
```

We create a matrix that has differentially expressed genes as rows and columns and that, adjusting to the threshold chosen earlier (0.97), will tell us if two genes present correlation (TRUE) or not (FALSE). According to that matrix, we will generate the network using the graph.adjacency function, specifying that it is undirected. Finally, we save the network in a GML file to visualize it later in Cytoscape.

```
adjacency.097 <- (gene.correlation > 0.97) & (gene.correlation < 1)
gene.coexpression.network <- graph.adjacency(adjacency.097, mode="undirected")
write.graph(gene.coexpression.network,file="ath_gene_coexpression_network.gml",format="gml")
```

Visualizamos la red en Cytoscape

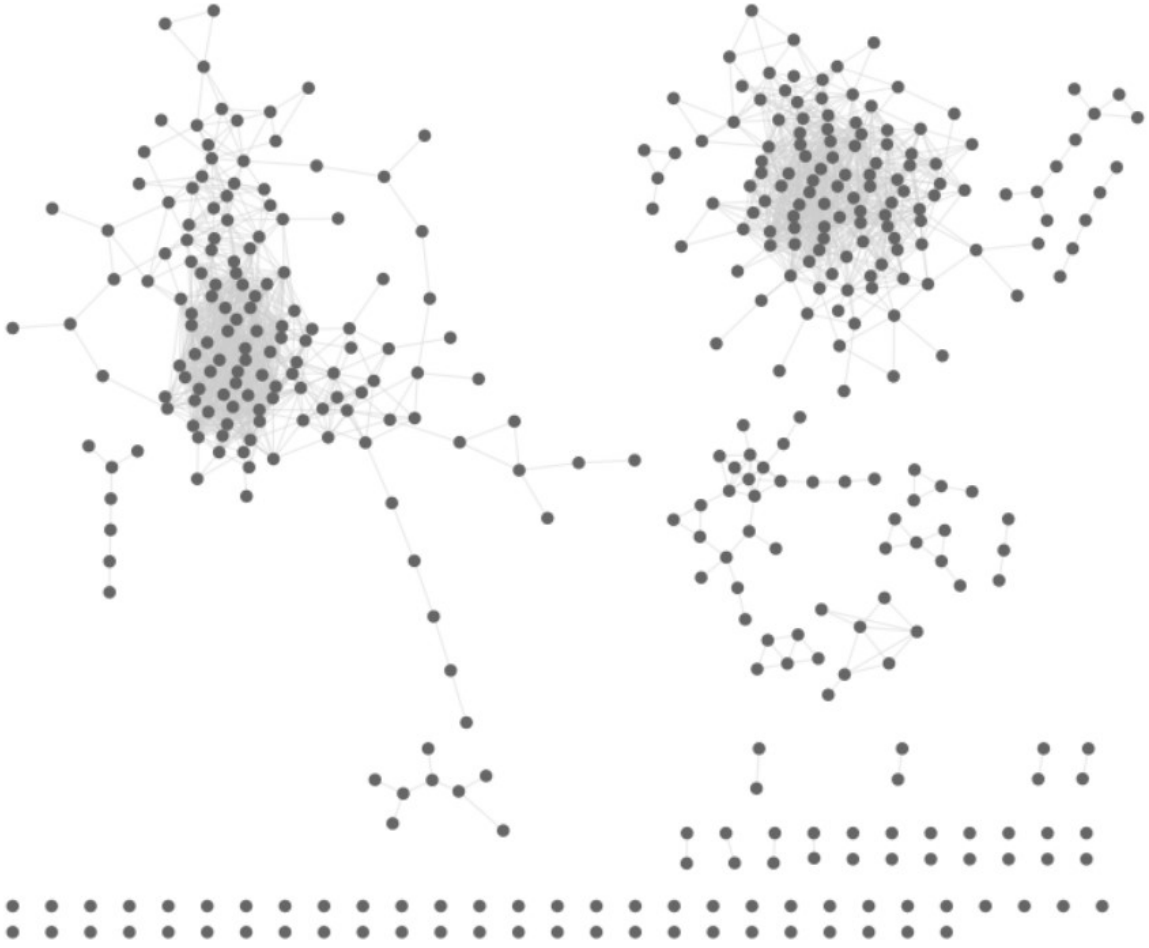


Figure 1: Visualización red

**Study of the Fit of the Generated Network to the Scale-Free Network Property Using Linear Regression and KS Test. Node Degree Distribution**

We will read the previously created network and calculate the node degree distribution with degree.distribution. We will see if our network is scale-free. To do this, we will use two methods, the first is the Kolmogorov-Smirnov method, which will give us a p-value that tells us with what confidence we can reject the null hypothesis that our network is not scale-free. In this case, the obtained p-value is 0.89, so we can say with great confidence that our network is scale-free.

The second method we will use is linear regression. For this, we will calculate the degree of each node (gene) and make a histogram, which should have a negative potential shape since, being scale-free, most nodes should be low degree, and only a few of high degree. To quantify it, we transform it to logarithm and do a linear regression to see if it looks like a straight line. We see that the R2 obtained with the linear regression is quite 0.627, quite poor, we focus more on the KS test to say that the network is scale-free.

```
gene.coexpression.network <- read.graph(file="ath_gene_coexpression_network.gml",format="gml")

network.degree.distribution <- degree.distribution(gene.coexpression.network)
fit.scale.free <- power.law.fit(network.degree.distribution)
fit.scale.free[["KS.p"]]
```

```
## [1] 0.8905287
```

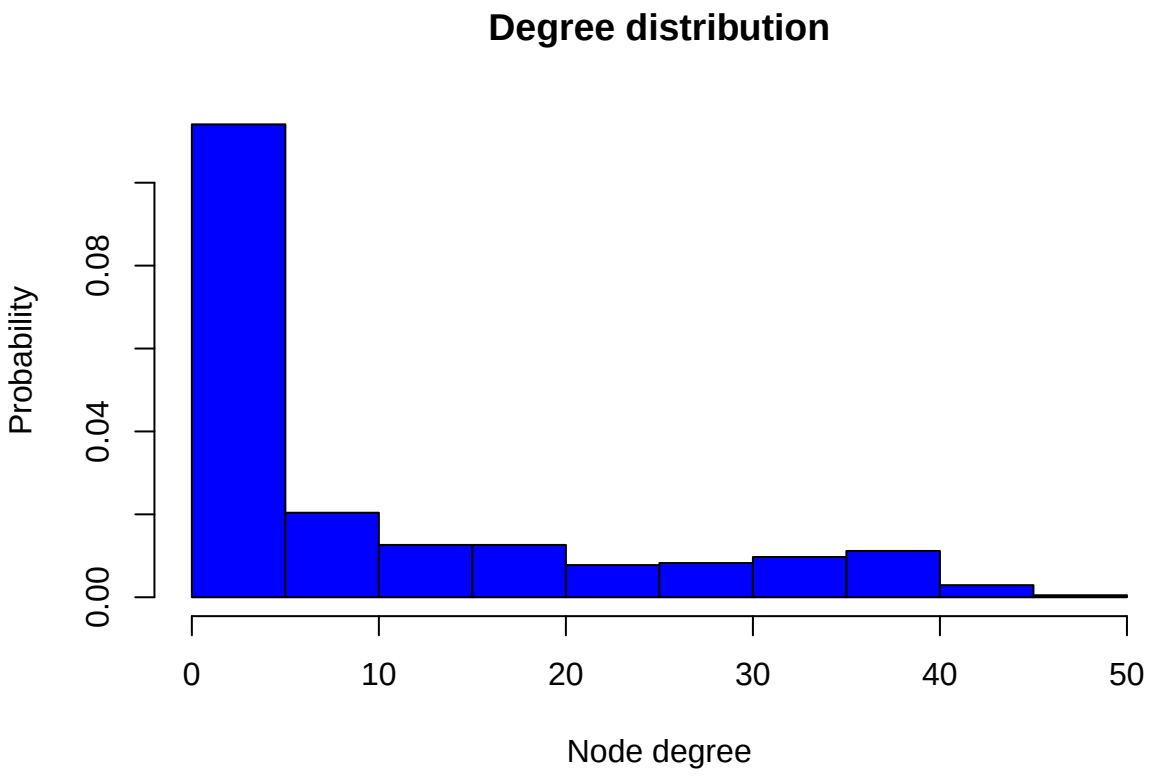
```
# node degrees##
network.degrees <- degree(gene.coexpression.network)
# frequency of each node##
degree.frequencies <- table(network.degrees)
# Eliminate grade 0 nodes to be able to apply log10##
degree.frequencies.no.0 <- degree.frequencies[-1]


sum(degree.frequencies)
```

```
## [1] 412
```

```
sum(degree.frequencies.no.0)
```

```
## [1] 344
```

```
degree.histogram <- hist(network.degrees,freq=FALSE,col="blue",
                         xlab="Node degree", ylab="Probability",main="Degree distribution")
```




```r
# log transformation
log10.degrees.frequencies <- log10(degree.frequencies.no.0)
log10.node.degrees <- log10(as.numeric(names(degree.frequencies.no.0)))

# Lineal regression
lm.r <- lm(log10.degrees.frequencies ~ log10.node.degrees)
summary(lm.r)
```

```
##
## Call:
## lm(formula = log10.degrees.frequencies ~ log10.node.degrees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52121 -0.09557  0.02126  0.14357  0.54185
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.71391    0.12748  13.444  < 2e-16 ***
## log10.node.degrees -0.84292    0.09846  -8.561 9.35e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2476 on 42 degrees of freedom
## Multiple R-squared:  0.6357, Adjusted R-squared:  0.627
## F-statistic: 73.29 on 1 and 42 DF,  p-value: 9.351e-11
```

**Determination and Representation of HUBs, GO Enrichment

In networks, there are some points that have a greater number of neighbors, which are called HUBs. In this case, they are genes that are believed to have a significant implication as their activation or repression can influence many genes, causing a cascade that makes the organism react by producing many different proteins.

We will calculate the HUBs of our generated network, for this, each gene will be assigned a HUB score, which is a score based on the attributes that a node presents to be considered a HUB. In our case, we will select as HUBs the top 10% of genes with the highest score. We generate a table with the hubs, their function, genBank... with the aafTableAnn function, then we introduce the hubs in GO and do the gene enrichment to see in which cellular processes they are involved, obtaining as a result that they are related to active transport processes through the membrane, specifically with secondary antiporter type transport.

```
## Identification of the hubs and generation of attribute, txt and html files.
network.hub.scores <- hub.score(gene.coexpression.network)
hub.score.attributes <-network.hub.scores[["vector"]]
write.table(hub.score.attributes,file="hub_score_attributes.txt", col.names = F, quote = F, sep = "\t")
quantile.090 <- quantile(hub.score.attributes,prob=0.90)
hubs.values <- hub.score.attributes[hub.score.attributes > quantile.090]
hubs.names <- names(hubs.values)
write.table(hubs.names, file = "hubs.names.txt", col.names = F, quote = F, sep = "\t")
hubs.table <- aafTableAnn(hubs.names, "ath1121501.db", aaf.handler())
```

```
## Loading required package: ath1121501.db
```

```
## Loading required package: org.At.tair.db
```

```
##
```

```
##
```

```
## Warning in chkPkgs(chip): The ath1121501.db package does not appear to contain
## annotation data.
```

```
## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().
```

```
## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().
```

```
saveHTML(hubs.table, file="hubs_table.html")
```

```
## HUBs gene enrichment, treemap in Revigo
revigo.names <- c("term_ID","description","frequency","value","uniqueness","dispensability","representa
revigo.data <- rbind(c("GO:0042910","xenobiotic transmembrane transporter activity",0.178371650913369,
                       4.5543957967264,0.375461362756366,0,"xenobiotic transmembrane transporter activi
c("GO:0015291","secondary active transmembrane transporter activity",1.0386256999114,
  3.58838029403677,0.209449543444262,0.40070127,"xenobiotic transmembrane transporter activity"),
c("GO:0015297","antiporter activity",0.474138311739169,
  3.71669877129645,0.22194168062897,0.68398558,"xenobiotic transmembrane transporter activity"))
```

```
stuff <- data.frame(revigo.data)
names(stuff) <- revigo.names
stuff$value <- as.numeric( as.character(stuff$value) )
stuff$frequency <- as.numeric( as.character(stuff$frequency) )
stuff$uniqueness <- as.numeric( as.character(stuff$uniqueness) )
stuff$dispensability <- as.numeric( as.character(stuff$dispensability) )

## by default, outputs to a PDF file
pdf( file="revigo_HUBs_treemap.pdf", width=16, height=9 ) ## width and height are in inches
treemap(
  stuff,
  index = c("representative","description"),
  vSize = "value",
  type = "categorical",
  vColor = "representative",
  title = "Revigo TreeMap",
  inflate.labels = FALSE,
  lowerbound.cex.labels = 0,
  position.legend = "none"
)
dev.off()
```

```
## pdf
##   2
```

HTML HUBs Table txt HUBs Table

Visualize the network according to the HUB score

Gene Enrichment Treemap HUBs

Calculation of Clustering Coefficient and Comparison with Randomly Generated Networks

We will calculate the clustering coefficient, which is a measure of how small the paths between nodes in a network are. If the clustering coefficient is high, it means that the nodes in that network are very clustered and connected. If a network meets the properties of being scale-free and having a very high clustering coefficient, it is called a small-world network.

To know if our network is small-world, first, we calculate the average clustering coefficient of our network with the transitivity function, then we will generate 1000 random networks using the Barabasi model (Barabasi.game), in which nodes are added to the network and the new node will join with higher probability to the existing node with the highest degree. Once we have the networks, we will measure the average clustering coefficient of each one and compare it with ours.

For each added node, we will add the average number of edges our network has, for example, our network has 2101 edges and 412 nodes, so each node has an average of 5 edges. We generate an empty vector in which we will collect the coefficients of each of the random networks. We do a for loop to make the 1000 networks and automatically calculate their coefficient and store it in the vector. We calculate the transitivity of our network, in this case, we get that it is 0.6234, which is quite high transitivity. Now we will see how many of the randomly generated networks have a higher average clustering coefficient than ours.

The result is that 0 of the randomly generated networks have a higher clustering coefficient than ours, so we can say it is a SMALL-WORLD network, with a probability of error less than, at least, 10^-3. We will also calculate the individual transitivity of each node in our network, to visualize it in a gradient using Cytoscape, where the redder nodes are those with a higher clustering coefficient and the yellower ones have a lower one.

```r
2101/412
```

```
## [1] 5.099515
```

```r
number.of.added.edges <- c(rep(5,412))
random.scale.free.graph <- barabasi.game(n=412,out.seq=number.of.added.edges,directed=FALSE)
transitivity(random.scale.free.graph)
```

```
## [1] 0.06315598
```

```r
clustering.coefficients <- vector(length=1000)

for(i in 1:1000) {
  random.scale.free.graph <- barabasi.game(n=412,out.seq=number.of.added.edges,directed=FALSE)
  clustering.coefficients[i] <- transitivity(random.scale.free.graph)
}
network.clustering.coefficient <- transitivity(gene.coexpression.network, type = "global")
sum(clustering.coefficients > network.clustering.coefficient) / 1000
```

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

```r
node.transitivity <- transitivity(gene.coexpression.network,type = "local",isolates = "zero")
names(node.transitivity) <- names(V(gene.coexpression.network))
write.table(node.transitivity,file="node_transitivity.txt", quote = F, sep ="\t")
```

Visualize the network according to the clustering coefficient

## CLUSTERING ANALYSIS

### Objective

The objective of clustering is to group the genes that constitute our network according to the coexpression that occurs between them, to be able to see more clearly and realistically how the entire biological system works, since genes do not act individually, but groups of them lead the organism to express different phenotypes. In the specific case of our network, we will see which groups of genes are coexpressed according to the time they have been under phosphate restriction and what function they have, as well as if they have any repercussion on the expression of other groups or clusters that we have in our network.

To do the clustering, we will have to choose between two possible methods, the first is hierarchical clustering (hclust), in which each element starts individually and groups according to the similarities they present, we also have to choose the right moment to cut the tree, that is, we have to choose the moment when each individual is separated into the group that corresponds according to their characteristics but does not reach the moment when individuals are forced to enter a group that does not belong to them. The first step is to generate a similarity matrix which will be 1 minus the gene coexpression matrix, then generate a distance matrix from the previous similarity matrix that hclust needs to recalculate the distance after dividing the groups. The second method is partitioning around medoids or PAM. In this case, it receives the number of groups we want to form (we will try from 2 to 10). From here, a central element is chosen for each group that will act as a central element, and from there each element is joined to the group it best fits, so that the intercluster distance is the greatest possible and the intracluster distance is the smallest possible.

To know which method and the number of groups we have to generate, we will calculate the silhouette of each process, which will give us the goodness of the clustering. To do this, we calculate the silhouettes of each method and accumulate them in a vector, then we represent them in a plot and choose the method that has the highest silhouette.

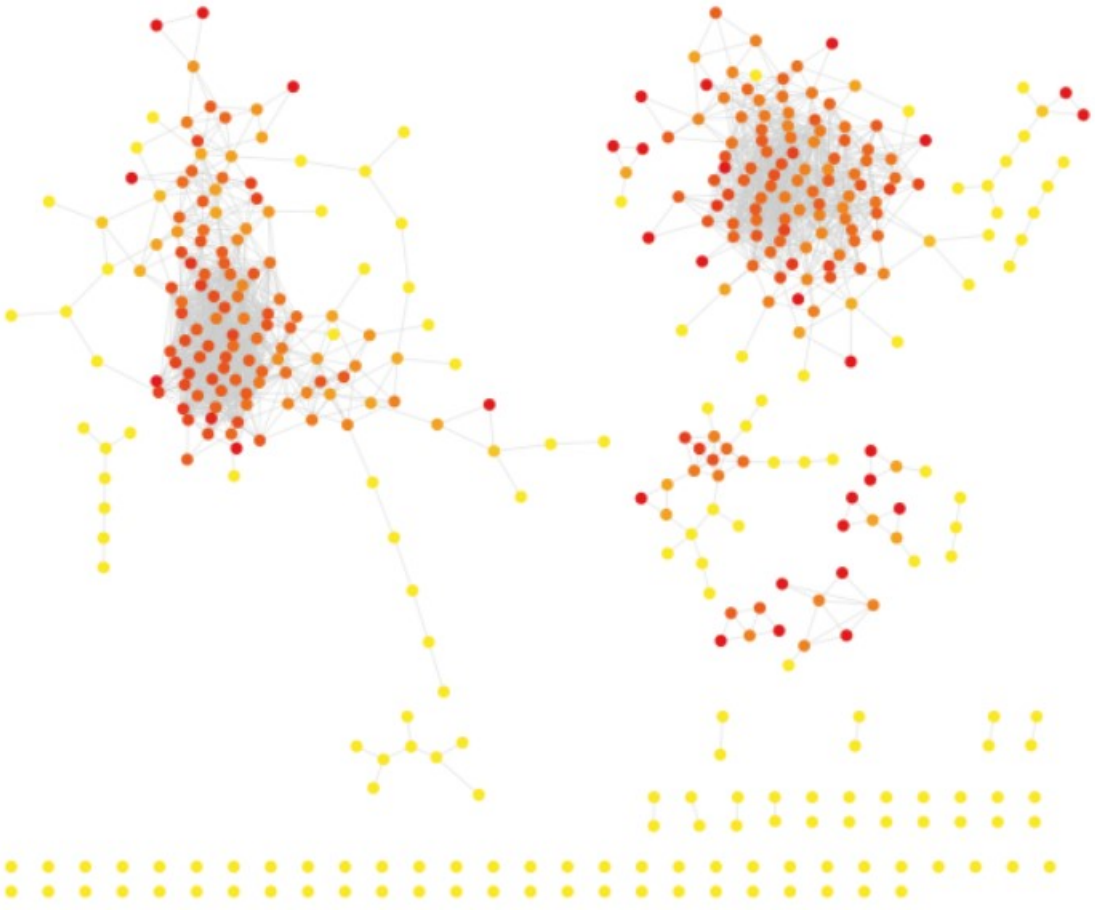


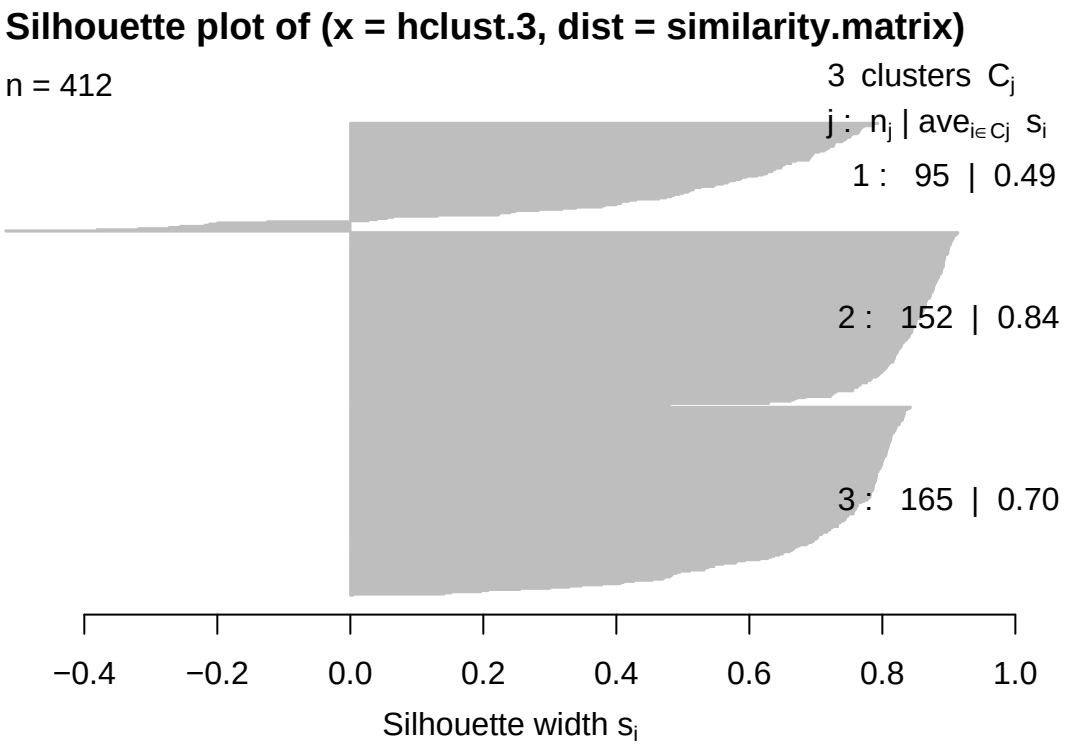
Figure 2: Visualización de la red según coeficiente de agrupamiento

In our case, the method we will choose is PAM3, that is, our clustering has been generated using the partitioning around medoids method dividing the individuals into 3 groups. As we see in the graph, we could have chosen PAM2 as they present a very similar silhouette. However, if we go to the functionality of the genes, in the case of putting 3 clusters, cluster 1 contains genes that do not regulate the expression of any of the other two clusters and performs a completely different function. If we select PAM2 and divide only into 2 clusters, we would be forcing these genes to enter one of the two clusters, when in my opinion they should not do so and should be in a separate cluster. On the other hand, the other two clusters are clearly related, both in function and expression, as the expression of one silences the other, so I understand that they have opposite functions for the same processes, and depending on the conditions of the medium one is expressed, and the other is repressed.

All this can be seen in the graph comparing the average expressions of each cluster of each sample. In each sample, we see the expression of the three clusters of that sample. For example, at x=1 we are seeing the average expression of the 3 clusters (green, red, and blue) of the control sample at hour 0. Seeing the graph, we can conclude that clusters 2 and 3 are opposite. In control conditions, cluster 2 is expressed much more than cluster 3, the first being at an average expression level between 8 and 9 and the second around 6. In phosphate restriction conditions, the expression of cluster 2 drops to levels between 6 and 7, while cluster 3 reaches 8. Cluster 1 on the other hand increases its expression significantly in phosphate restriction conditions but has no notable relationship with any of the other two clusters, so we understand that it is related to a completely different response that also occurs in the absence of phosphate.

```
similarity.matrix <- 1 - gene.correlation

## hclust uses the similarity matrix as distances and the average method
## to recalculate distances, calculating the hierarchical clustering
hierarchical.clustering <- hclust(as.dist(similarity.matrix), method="average")

## The cutree function allows cutting the tree using different heights, and therefore
## producing a different number of clusters.
hclust.2 <- cutree(hierarchical.clustering, k=2)
hclust.3 <- cutree(hierarchical.clustering, k=3)
hclust.4 <- cutree(hierarchical.clustering, k=4)
hclust.5 <- cutree(hierarchical.clustering, k=5)
hclust.6 <- cutree(hierarchical.clustering, k=6)
hclust.7 <- cutree(hierarchical.clustering, k=7)
hclust.8 <- cutree(hierarchical.clustering, k=8)
hclust.9 <- cutree(hierarchical.clustering, k=9)
hclust.10 <- cutree(hierarchical.clustering, k=10)

## The pam function uses the similarity
#matrix as distances to determine clusters
## according to the partitioning around
#medoids method. Centroid refinement loop and group refinement.
pam.2 <- pam(as.dist(similarity.matrix), k=2, diss=TRUE)
pam.3 <- pam(as.dist(similarity.matrix), k=3, diss=TRUE)
pam.4 <- pam(as.dist(similarity.matrix), k=4, diss=TRUE)
pam.5 <- pam(as.dist(similarity.matrix), k=5, diss=TRUE)
pam.6 <- pam(as.dist(similarity.matrix), k=6, diss=TRUE)
pam.7 <- pam(as.dist(similarity.matrix), k=7, diss=TRUE)
pam.8 <- pam(as.dist(similarity.matrix), k=8, diss=TRUE)
pam.9 <- pam(as.dist(similarity.matrix), k=9, diss=TRUE)
pam.10 <- pam(as.dist(similarity.matrix), k=10, diss=TRUE)

## The silhouette function allows us to calculate the silhouette of a clustering which serves as a meas
## for the goodness of said clustering.
```

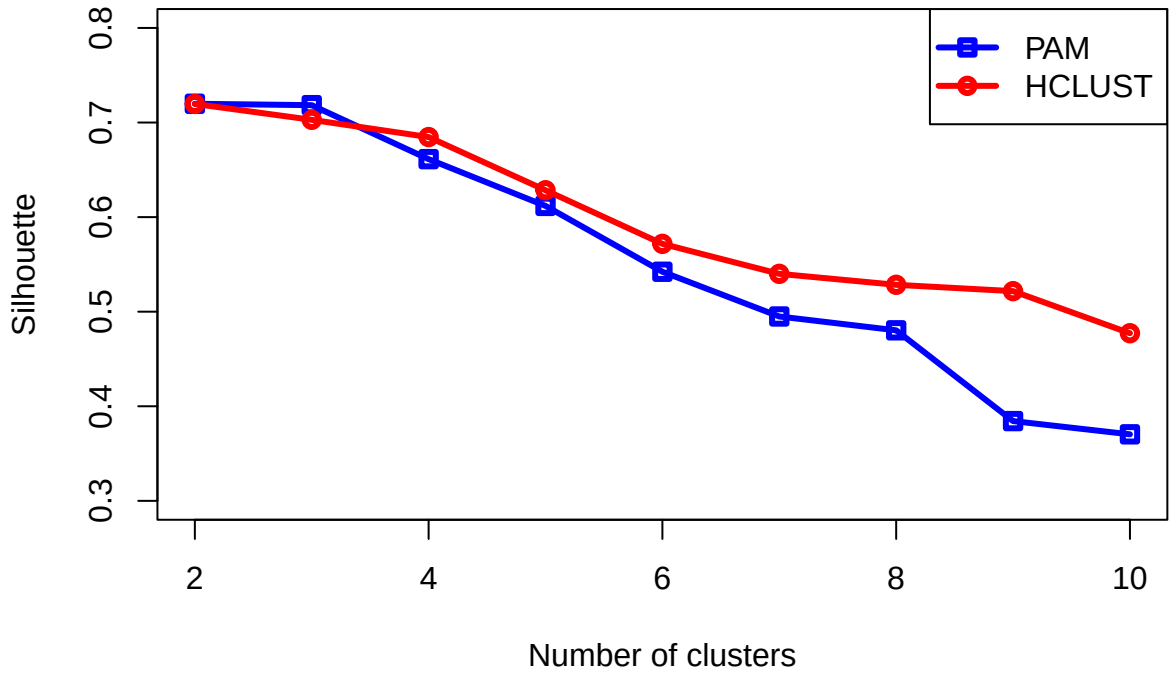
```
sil2 <- silhouette(hclust.2, dist=similarity.matrix)
sil3 <- silhouette(hclust.3, dist=similarity.matrix)
sil4 <- silhouette(hclust.4, dist=similarity.matrix)
sil5 <- silhouette(hclust.5, dist=similarity.matrix)
sil6 <- silhouette(hclust.6, dist=similarity.matrix)
sil7 <- silhouette(hclust.7, dist=similarity.matrix)
sil8 <- silhouette(hclust.8, dist=similarity.matrix)
sil9 <- silhouette(hclust.9, dist=similarity.matrix)
sil10 <- silhouette(hclust.10, dist=similarity.matrix)

plot(sil3, border="grey")
```




Average silhouette width : 0.7

```
hclust.sil.values <- c(summary(sil2)[["avg.width"]], summary(sil3)[["avg.width"]], summary(sil4)[["avg.

sil2 <- silhouette(pam.2)
sil3 <- silhouette(pam.3)
sil4 <- silhouette(pam.4)
sil5 <- silhouette(pam.5)
sil6 <- silhouette(pam.6)
sil7 <- silhouette(pam.7)
sil8 <- silhouette(pam.8)
sil9 <- silhouette(pam.9)
sil10 <- silhouette(pam.10)

pam.sil.values <- c(summary(sil2)[["avg.width"]], summary(sil3)[["avg.width"]], summary(sil4)[["avg.wid
```

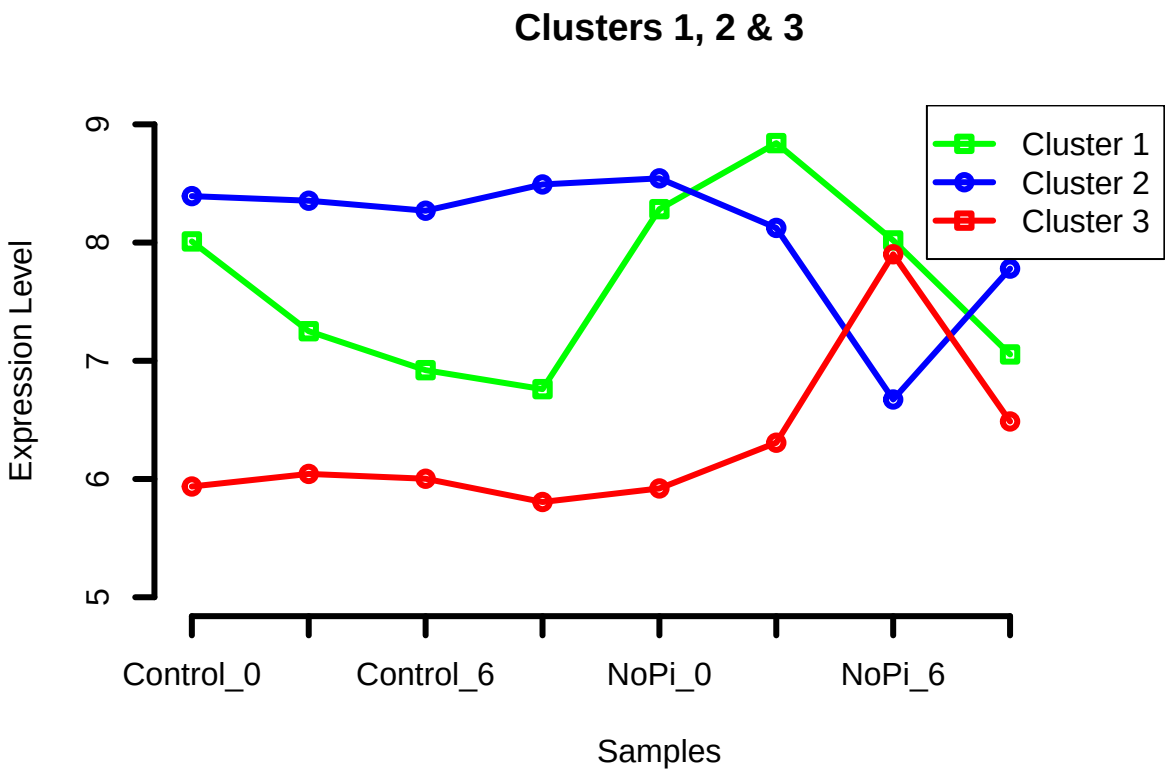
```r
## We represent for the two clustering methods, hierarchical and pam, and for different numbers
## of clusters the corresponding silhouette to choose the best combination of clustering method
## and number of clusters.
plot(2:10, pam.sil.values, type="o", col="blue", pch=0, ylim=c(0.3,0.8), xlab="Number of clusters", yla
lines(2:10, hclust.sil.values, type="o", col="red", pch=1, xlab="", ylab="", lwd=3)
legend("topright", legend=c("PAM", "HCLUST"), col=c("blue", "red"), pch=c(0, 1), lwd=3)
```




```r
## Visualization of clusters

clustering.pam.3 <- pam.3[["clustering"]]
write.table(clustering.pam.3, file="pam_3.txt", quote=FALSE, sep="\t")

library(annaffy)

cluster1.pam3 <- names(which(pam.3[["clustering"]] == 1))
cluster2.pam3 <- names(which(pam.3[["clustering"]] == 2))
cluster3.pam3 <- names(which(pam.3[["clustering"]] == 3))

cluster1.pam3.table <- aafTableAnn(cluster1.pam3, "ath1121501.db", aaf.handler())
cluster2.pam3.table <- aafTableAnn(cluster2.pam3, "ath1121501.db", aaf.handler())
cluster3.pam3.table <- aafTableAnn(cluster3.pam3, "ath1121501.db", aaf.handler())
head(cluster3.pam3)
```

```
## [1] "263231_at" "257947_at" "266368_at" "253203_at" "260203_at" "250605_at"
```

```
saveHTML(cluster1.pam3.table, file="cluster_1_annotation.html")
saveHTML(cluster2.pam3.table, file="cluster_2_annotation.html")
saveHTML(cluster3.pam3.table, file="cluster_3_annotation.html")

# saveText(cluster1.pam3.table, file="cluster_1_annotation.txt")
# saveText(cluster2.pam3.table, file="cluster_2_annotation.txt")
# saveText(cluster3.pam3.table, file="cluster_3_annotation.txt")

## Expression of genes that are differentially expressed in each cluster ##
expr.cluster1.pam3 <- diff.expr[, cluster1.pam3]
expr.cluster2.pam3 <- diff.expr[, cluster2.pam3]
expr.cluster3.pam3 <- diff.expr[, cluster3.pam3]

## Mean expression of each cluster ##
mean.profile.cluster1.pam3 <- rowMeans(expr.cluster1.pam3)
mean.profile.cluster2.pam3 <- rowMeans(expr.cluster2.pam3)
mean.profile.cluster3.pam3 <- rowMeans(expr.cluster3.pam3)

## Visualization of the mean expression of clusters in each sample ##
samples <- c("Control_0", "Control_1", "Control_6",
             "Control_24", "NoPi_0", "NoPi_1", "NoPi_6", "NoPi_24")

plot(axes=F, mean.profile.cluster1.pam3, type="o", col="green",
     xlim=c(1, 9), ylim=c(5, 9), xlab="Samples", ylab="Expression Level", lwd=3, pch=0, main="Clusters
axis(side=2, lwd=3)
axis(side=1, at=seq(from=1, to=8), labels=samples, lwd=3)
lines(mean.profile.cluster2.pam3, type="o", col="blue", lwd=3, pch=1)
lines(mean.profile.cluster3.pam3, type="o", col="red", lwd=3, pch=1)
legend("topright", legend=c("Cluster 1", "Cluster 2", "Cluster 3"), col=c("green", "blue", "red"), pch=
```




**Visualizamos la red en cytoscape según el cluster en el que se encuentra cada nodo Cluster 1 -> Verde Cluster 2 -> Azul Cluster 3 -> Rojo**

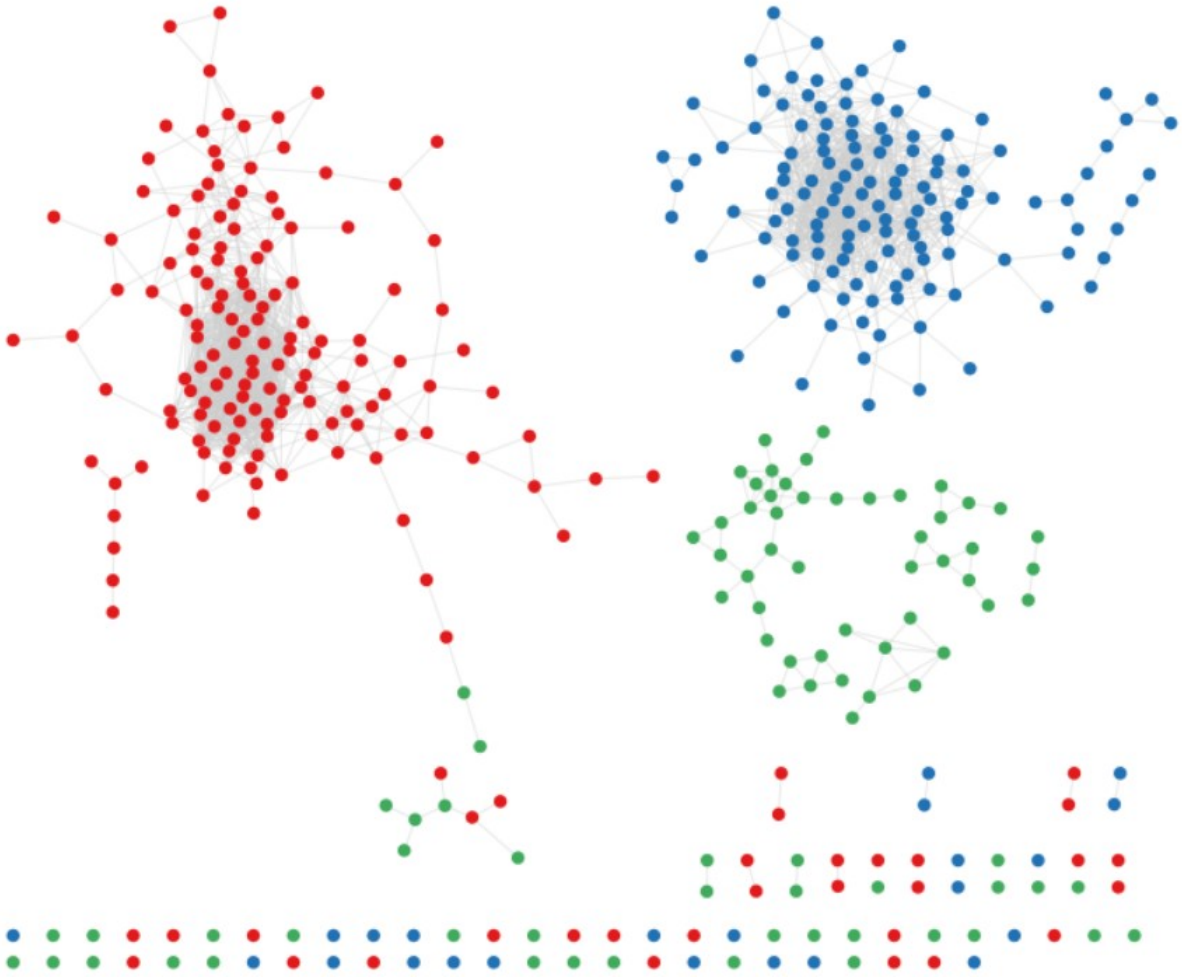


Table HTML Clúster 1 Table HTML Clúster 2 Table HTML Clúster 3 Table txt Clúster 1 Table txt Clúster 2 Table txt Clúster 3 Table Ontología génica GEO clúster 1 Table Ontología génica GEO clúster 2 Table Ontología génica GEO clúster 3

**Displaying the Processes in which Our Clusters are Involved Using REVIGO Treemaps**

```
#REVIGO treemap Clúster 1#

revigo.names <- c("term_ID","description","frequency","value","uniqueness","dispensability","representa
revigo.data <- rbind(c("GO:0050896","response to stimulus",14.6729908229443,5.84466396253494,1,0,"respo
c("GO:0070482","response to oxygen levels",0.0402346840881025,17.4497716469449,0.349336395705353,0,"res
c("GO:0006950","response to stress",4.7761769556883,7.56383735295924,0.417501412801227,0.51732357,"resp
c("GO:0009628","response to abiotic stimulus",0.571567920937768,9.15614457737684,0.51017980149819,0.357
c("GO:0033554","cellular response to stress",3.00034034530875,12.0535477349869,0.379674522395061,0.4995
c("GO:0042221","response to chemical",3.72771115376065,8.44977164694491,0.429624086770263,0.44897129,"r
c("GO:0051716","cellular response to stimulus",11.8892878096202,11.2765443279648,0.371605939151585,0.63
c("GO:0070887","cellular response to chemical stimulus",1.82177410918801,14.600326278519,0.378390771494
c("GO:0071456","cellular response to hypoxia",0.0143929759889949,17.6819366650372,0.217398187277304,0.6

stuff <- data.frame(revigo.data);
names(stuff) <- revigo.names;
```

```
stuff$value <- as.numeric( as.character(stuff$value) );
stuff$frequency <- as.numeric( as.character(stuff$frequency) );
stuff$uniqueness <- as.numeric( as.character(stuff$uniqueness) );
stuff$dispensability <- as.numeric( as.character(stuff$dispensability) );

# by default, outputs to a PDF file
pdf( file="revigo1_treemap.pdf", width=16, height=9 ) # width and height are in inches

# check the tmPlot command documentation for all possible parameters - there are a lot more
treemap(
  stuff,
  index = c("representative","description"),
  vSize = "value",
  type = "categorical",
  vColor = "representative",
  title = "Revigo TreeMap",
  inflate.labels = FALSE,      # set this to TRUE for space-filling group labels - good for posters
  lowerbound.cex.labels = 0,   # try to draw as many labels as possible (still, some small squares may
 # bg.labels = "#CCCCCCAA",   # define background color of group labels
                               # "#CCCCCC00" is fully transparent, "#CCCCCCAA" is semi-transparent gr
  position.legend = "none"
)

dev.off()
```

```
## pdf
##   2
```

```
#REVIGO treemap clúster 2#

revigo.names <- c("term_ID","description","frequency","value","uniqueness","dispensability","representa
revigo.data <- rbind(c("GO:0005975","carbohydrate metabolic process",5.7234973704885,3.1681302257195,1,
c("GO:0042178","xenobiotic catabolic process",0.0433778633642065,3.35556141053216,1,0,"xenobiotic catab

stuff <- data.frame(revigo.data);
names(stuff) <- revigo.names;

stuff$value <- as.numeric( as.character(stuff$value) );
stuff$frequency <- as.numeric( as.character(stuff$frequency) );
stuff$uniqueness <- as.numeric( as.character(stuff$uniqueness) );
stuff$dispensability <- as.numeric( as.character(stuff$dispensability) );

# by default, outputs to a PDF file
pdf( file="revigo2_treemap.pdf", width=16, height=9 ) # width and height are in inches

# check the tmPlot command documentation for all possible parameters - there are a lot more
treemap(
  stuff,
  index = c("representative","description"),
  vSize = "value",
  type = "categorical",
  vColor = "representative",
  title = "Revigo TreeMap",
```

```
  inflate.labels = FALSE,      # set this to TRUE for space-filling group labels - good for posters
  lowerbound.cex.labels = 0,   # try to draw as many labels as possible (still, some small squares may
  #bg.labels = "#CCCCCCAA",   # define background color of group labels
                               # "#CCCCCC00" is fully transparent, "#CCCCCCAA" is semi-transparent gr
  position.legend = "none"
)

dev.off()
```

```
## pdf
##   2
```

```
#REVIGO treemap clúster 3#

revigo.names <- c("term_ID","description","frequency","value","uniqueness","dispensability","representa
revigo.data <- rbind(c("GO:0010035","response to inorganic substance",0.276188643142887,3.1414628024303
c("GO:0050896","response to stimulus",14.6729908229443,3.62160209905186,1,0,"response to stimulus"));

stuff <- data.frame(revigo.data);
names(stuff) <- revigo.names;

stuff$value <- as.numeric( as.character(stuff$value) );
stuff$frequency <- as.numeric( as.character(stuff$frequency) );
stuff$uniqueness <- as.numeric( as.character(stuff$uniqueness) );
stuff$dispensability <- as.numeric( as.character(stuff$dispensability) );

# by default, outputs to a PDF file
pdf( file="revigo3_treemap.pdf", width=16, height=9 ) # width and height are in inches

# check the tmPlot command documentation for all possible parameters - there are a lot more
treemap(
  stuff,
  index = c("representative","description"),
  vSize = "value",
  type = "categorical",
  vColor = "representative",
  title = "Revigo TreeMap",
  inflate.labels = FALSE,      # set this to TRUE for space-filling group labels - good for posters
  lowerbound.cex.labels = 0,   # try to draw as many labels as possible (still, some small squares may
  #bg.labels = "#CCCCCCAA",   # define background color of group labels
                               # "#CCCCCC00" is fully transparent, "#CCCCCCAA" is semi-transparent gr
  position.legend = "none"
)

dev.off()
```

```
## pdf
##   2
```

Treemap Revigo cluster 1

Treemap Revigo cluster 2

Treemap Revigo cluster 3

Conclusion

From the data we have obtained throughout the analysis, we can draw some conclusions about the functional and biological significance the studied genes may have and the study of the organism's response as a whole. As we have seen, the genes that showed coexpression have been grouped into 3 different clusters. Cluster 1 functions independently of the other two, it is expressed immediately after the plant is exposed to phosphate restriction, reaching its expression peak about 1 hour after restriction. This cluster influences various processes, but in general, it focuses on processes related to cellular stress signal transduction. This is consistent with that rapid and abrupt manifestation, as they are genes whose function is to alert the organism to this Pi deficiency to which it is being exposed.

Clusters 2 and 3 are clusters related to transport processes and function antagonistically. To be more specific, phenotypically, these genes cause the overgrowth of root hairs in plants under Pi deficiency. This makes clear physiological sense since the function of these root hairs is to increase the absorption surface of the plant's roots to maximize the uptake of essential nutrients and salts. The plant under low Pi conditions detects this metabolite deficiency and tries to maximize its absorption surface to mitigate it. Specifically, it seems that both clusters control genes related to cell wall degradation, a process that seems counterproductive for producing root hairs.

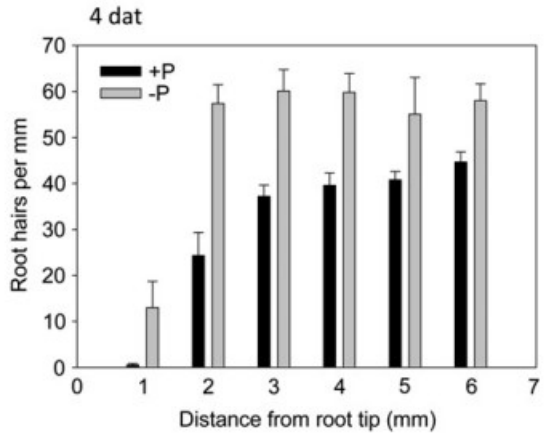
Cluster 2 contains genes related to cell wall degradation, such as At1g05650 and At5g04960, related to the production of polygalacturonases and pectinesterases respectively, enzymes involved in cell wall degradation. Cluster 3, on the other hand, contains genes that inhibit the cell wall degrading genes, that is, those of cluster 2. For example, At1g02810, which is an inhibitor of pectinesterases.

If we look again at the graph of the expressions of each cluster, we see that under phosphate restriction conditions, approximately between 4 and 8 hours, it is clearly seen how the overexpression of the genes of cluster 3 occurs and the partial silencing of cluster 2, which, as we explained before, has a very clear physiological sense.

In short, the plant's response to Pi restriction is mainly the elongation and creation of root hairs in the roots, a process that is favored by the non-degradation of the cell wall, which it achieves by overexpressing cluster 3 and reducing the expression of cluster 2. On the other hand, cluster 1 contains genes that alert the cell to the nutrient deficiency it is exposed to.

Article Conclusions

In the article, the conclusions reached and the analysis performed are quite similar to ours. In addition to the computational analysis, they also perform experiments to validate these conclusions, such as this one where they measure the number of root hairs per mm in plants that have been in Pi restriction conditions for 4 days and compare it with a control:




We clearly see that along the entire root, the number of root hairs is significantly higher in plants grown in Pi- medium, which experimentally evidences the information provided by the gene analysis performed.

**Bibliography**

Artícle:

Thomas J. Buckhout, Wen-Dar Lin, Ya-Yun Liao, Chao-Yu Pan, Thomas J.W.Yang and Wolfgang Schmidt(March 2011), *Plant Physiology*, , Vol. 155, pp. 1383–1402.

Gorilla GO: http://cbl-gorilla.cs.technion.ac.il/