Host-parasite-clocks

Biplabendu Das

14 October, 2021

## Overview/Goals

This document provides a step-by-step tutorial on how to: - build a circadian gene co-expression network (GCN), - how to annotate the network using published data, - infer functions of your gene-clusters-of-interest.

## Step 1: Build circadian GCN

### 1.1 Load data

We will build a circadian GCN for the ant, *Camponotus floridanus*, using time-course RNASeq data collected in Das and de Bekker (2021; bioRxiv). The raw data is deposited on NCBI under accession number **XXXX**.

DESCRIBE THE DATA HERE (briefly)

One would need to perform the usual steps - trimming the reads, mapping the reads to the genome, and quantifying normalized gene counts - to obtain normalized gene expression data from the raw reads. At the end, for each gene in the genome, we should have the normalized expression for each time point, throughout the 24h day.

For the purpose of this tutorial, we assume that you have organized the processed data into a (gene-expr X time-point) format as shown below.

# loading database which contains data for Das and de Bekker 2021 (bioRxiv)  
db <- dbConnect(RSQLite::SQLite(), paste0(path\_to\_repo,"/data/databases/TC5\_data.db"))  
  
# extract the (gene-expr X time-point) data  
dat <-  
 db %>%  
 tbl(., "annot\_fpkm") %>%  
 select(gene\_name, X2F:X24N) %>%  
 collect()  
  
dim(dat)

## [1] 13813 25

### 1.2 Clean data

The above dataset contains all genes (n=13,813) in the ant genome. However, not all of these genes are expressed in the ant brain, and some are expressed at very low levels that are not biologically meaningful. Therefore, we will only keep the genes that are “expressed” (≥1 FPKM) in the ant brain, for at least half of all the sampled time points.

# Which genes are expressed throughout the day in both forager and nurses brains?  
daily.exp.genes <-  
 tbl(db, "expressed\_genes") %>% # note, the information is already available in the database  
 filter(exp\_half\_for == "yes" & exp\_half\_nur == "yes") %>%  
 collect() %>%  
 pull(gene\_name)  
  
# Subset the gene-expr X time-point file  
dat <- dat %>% filter(gene\_name %in% daily.exp.genes)  
dim(dat)

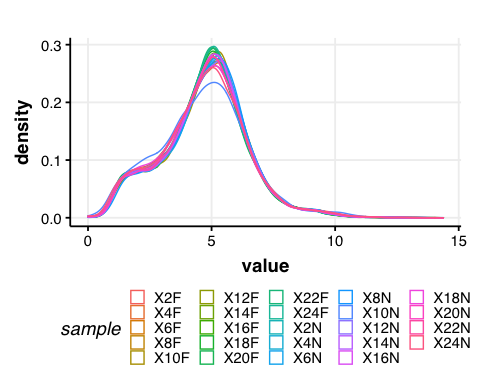
## [1] 9139 25

This is our cleaned, input data file. The daily expression for these 9139 genes that will be used to create the circadian GCN of *Camponotus floridanus*.

### 1.3 Format data

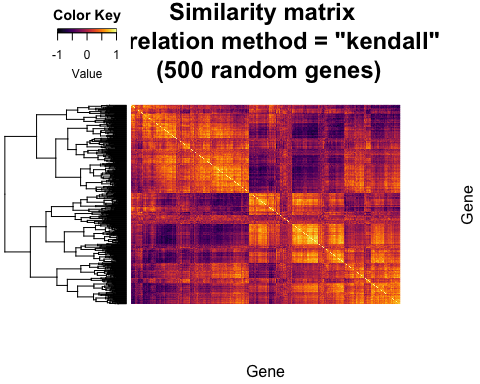
* Log2 transform the data

datExpr = as.data.frame(t(log2(dat[-c(1)]+1)))  
names(datExpr) = dat$gene\_name  
rownames(datExpr) = names(dat)[-c(1)]  
  
# USE THE FOLLOWING CODE TO CHECK IF YOU HAVE ANY BAD SAMPLES #  
 # gsg = goodSamplesGenes(datExpr0, verbose = 3);  
 # gsg$allOK  
 #  
 # sampleTree = hclust(dist(datExpr0), method = "average");  
 # # Plot the sample tree: Open a graphic output window of size 12 by 9 inches  
 # # The user should change the dimensions if the window is too large or too small.  
 # sizeGrWindow(12,9)  
 # #pdf(file = "Plots/sampleClustering.pdf", width = 12, height = 9);  
 # par(cex = 1);  
 # par(mar = c(0,4,2,0))  
 # plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="", cex.lab = 1.5,  
 # cex.axis = 1.5, cex.main = 2)  
  
# save the number of genes and samples  
# that will be used to create the circadian GCN  
nGenes = ncol(datExpr)  
nSamples = nrow(datExpr)  
  
# visualize the log-transformed data  
x = reshape2::melt(as.matrix(t(datExpr)))  
colnames(x) = c('gene\_id', 'sample', 'value')  
ggplot(x, aes(x=value, color=sample)) + geom\_density() + theme\_Publication()



### 1.4 Calculate gene-gene similarity

## Calculate Kendall's tau-b correlation for each gene-gene pair  
#  
# sim\_matrix <- cor((datExpr), method = "kendall") # this step takes time  
# save(sim\_matrix, file = paste0(path\_to\_repo, "/results/temp\_files/sim\_matrix\_for\_nur\_TC5.RData")) # might be useful to save the sim\_matrix and  
load(paste0(path\_to\_repo, "/results/temp\_files/sim\_matrix\_for\_nur\_TC5.RData")) # load it up  
  
## Let's display a chunk of the matrix (code from Hughitt 2016; github)  
heatmap\_indices <- sample(nrow(sim\_matrix), 500)  
gplots::heatmap.2(t(sim\_matrix[heatmap\_indices, heatmap\_indices]),  
 col=inferno(100),  
 labRow=NA, labCol=NA,  
 trace='none', dendrogram='row',  
 xlab='Gene', ylab='Gene',  
 main='Similarity matrix \n correlation method = "kendall" \n (500 random genes)',  
 density.info='none', revC=TRUE)



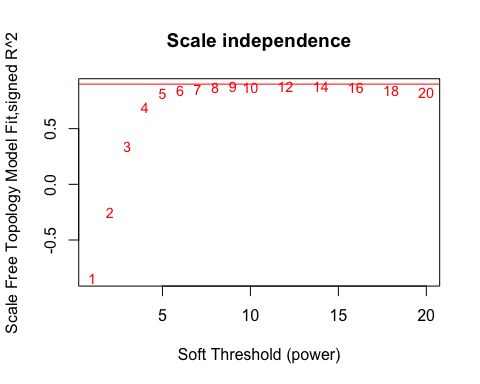
### 1.5 Create adjacency matrix

* To create the adjacency matrix, we need to first identify the soft-thresholding power

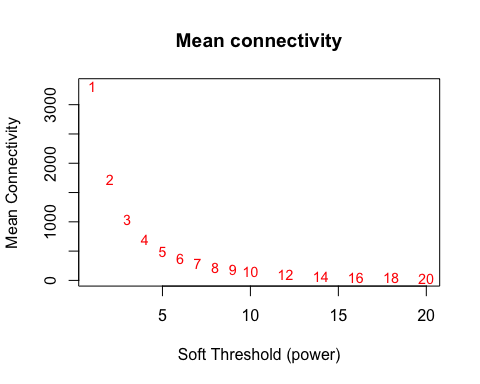
# Choose a set of soft-thresholding powers  
powers = c(c(1:10), seq(from = 12, to=20, by=2))  
# # Call the network topology analysis function  
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)

## pickSoftThreshold: will use block size 4895.  
## pickSoftThreshold: calculating connectivity for given powers...  
## ..working on genes 1 through 4895 of 9139  
## ..working on genes 4896 through 9139 of 9139  
## Power SFT.R.sq slope truncated.R.sq mean.k. median.k. max.k.  
## 1 1 0.845 1.900 0.995 3310.0 3390.00 4730  
## 2 2 0.248 0.276 0.930 1720.0 1710.00 3200  
## 3 3 0.343 -0.284 0.907 1050.0 988.00 2410  
## 4 4 0.696 -0.580 0.922 701.0 616.00 1930  
## 5 5 0.818 -0.762 0.951 499.0 402.00 1600  
## 6 6 0.847 -0.896 0.942 371.0 272.00 1360  
## 7 7 0.854 -0.992 0.933 285.0 190.00 1180  
## 8 8 0.868 -1.060 0.935 225.0 136.00 1030  
## 9 9 0.879 -1.110 0.940 181.0 99.10 919  
## 10 10 0.874 -1.160 0.928 148.0 73.40 824  
## 11 12 0.879 -1.220 0.928 103.0 42.20 676  
## 12 14 0.879 -1.280 0.921 74.8 25.40 568  
## 13 16 0.874 -1.310 0.916 56.1 15.80 485  
## 14 18 0.842 -1.360 0.884 43.2 10.10 420  
## 15 20 0.827 -1.390 0.874 34.0 6.64 367

# Plot the results:  
# sizeGrWindow(9, 5)  
# par(mfrow = c(1,2));  
cex1 = 0.9;  
# Scale-free topology fit index as a function of the soft-thresholding power  
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])\*sft$fitIndices[,2],  
 xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",  
 main = paste("Scale independence"));  
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])\*sft$fitIndices[,2],  
 labels=powers,cex=cex1,col="red");  
# this line corresponds to using an R^2 cut-off of h  
abline(h=0.90,col="red")



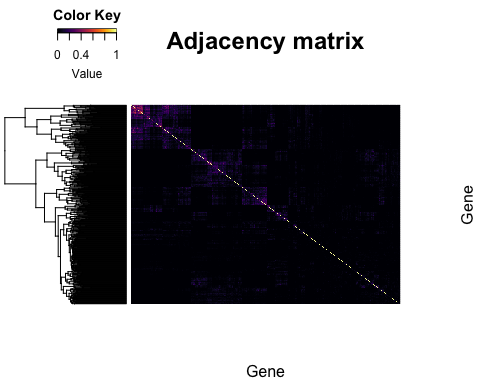
# Mean connectivity as a function of the soft-thresholding power  
plot(sft$fitIndices[,1], sft$fitIndices[,5],  
 xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",  
 main = paste("Mean connectivity"))  
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")



NOTE: The scale-free topology fit index reaches ~0.9 at a soft-thresholding-power=9, and it does not improve drastically beyond that.

Now, we can go ahead and create our adjacency matrix.

## Specify the soft-thresholding-power  
soft.power = 9  
  
## Construct adjacency matrix  
# adj\_matrix <- adjacency.fromSimilarity(sim\_matrix,  
# power=soft.power,  
# type='signed'  
# )  
# save(adj\_matrix, file = paste0(path\_to\_repo, "/results/temp\_files/adj\_matrix\_for\_nur\_TC5.RData")) # might be useful to save the sim\_matrix and  
load(paste0(path\_to\_repo, "/results/temp\_files/adj\_matrix\_for\_nur\_TC5.RData")) # load it up  
  
  
# Convert adj\_matrix to matrix  
gene\_ids <- rownames(adj\_matrix)  
  
adj\_matrix <- matrix(adj\_matrix, nrow=nrow(adj\_matrix))  
rownames(adj\_matrix) <- gene\_ids  
colnames(adj\_matrix) <- gene\_ids  
  
## Same heatmap as before, but now with the power-transformed adjacency matrix  
gplots::heatmap.2(t(adj\_matrix[heatmap\_indices, heatmap\_indices]),  
 col=inferno(100),  
 labRow=NA, labCol=NA,  
 trace='none', dendrogram='row',  
 xlab='Gene', ylab='Gene',  
 main='Adjacency matrix',  
 density.info='none', revC=TRUE)



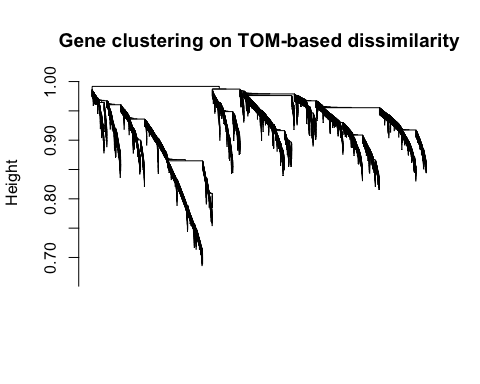
## Delete similarity matrix to free up memory  
rm(sim\_matrix)  
gc()

## used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)  
## Ncells 3910307 208.9 6997548 373.8 NA 6997548 373.8  
## Vcells 92801433 708.1 300696801 2294.2 16384 456586820 3483.5

## Step 2: Identify gene clusters

### 2.1 Create topological overalp matrix

# Turn adjacency into topological overlap  
# TOM = TOMsimilarity(adj\_matrix);  
# dissTOM = 1-TOM  
# save(dissTOM, file = paste0(path\_to\_repo, "/results/temp\_files/dissTOM\_for\_nur\_TC5.RData")) # might be useful to save the sim\_matrix and  
load(paste0(path\_to\_repo, "/results/temp\_files/dissTOM\_for\_nur\_TC5.RData")) # load it up  
  
# Call the hierarchical clustering function  
geneTree = hclust(as.dist(dissTOM), method = "average")  
  
# Plot the resulting clustering tree (dendrogram)  
# sizeGrWindow(12,9)  
plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based dissimilarity",  
 labels = FALSE, hang = 0.04)



### 2.2 Identify clusters

# We like large modules, so we set the minimum module size relatively high:  
minModuleSize = 30;  
  
# Module identification using dynamic tree cut:  
dynamicMods= cutreeDynamic(dendro = geneTree,  
 distM = dissTOM,  
 method = "hybrid",  
 verbose = 4,  
 deepSplit = 3, # see WGCNA for more info on tuning parameters  
 pamRespectsDendro = FALSE,  
 minClusterSize = minModuleSize);

## ..cutHeight not given, setting it to 0.99 ===> 99% of the (truncated) height range in dendro.  
## ..Going through the merge tree  
##   
## ..Going through detected branches and marking clusters..  
## ..Assigning Tree Cut stage labels..  
## ..Assigning PAM stage labels..  
## ....assigned 5531 objects to existing clusters.  
## ..done.

# view number of genes in each module  
table(dynamicMods)

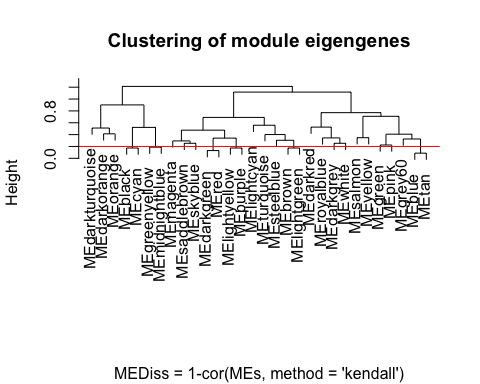
## dynamicMods  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16   
## 1337 1172 742 723 720 579 515 506 369 299 269 265 199 149 134 134   
## 17 18 19 20 21 22 23 24 25 26 27 28 29 30   
## 112 107 99 91 88 79 75 72 66 56 55 50 45 32

# Convert numeric lables into colors  
dynamicColors = labels2colors(dynamicMods)  
table(dynamicColors)

## dynamicColors  
## black blue brown cyan darkgreen   
## 515 1172 742 149 79   
## darkgrey darkorange darkred darkturquoise green   
## 72 56 88 75 720   
## greenyellow grey60 lightcyan lightgreen lightyellow   
## 269 112 134 107 99   
## magenta midnightblue orange pink purple   
## 369 134 66 506 299   
## red royalblue saddlebrown salmon skyblue   
## 579 91 45 199 50   
## steelblue tan turquoise white yellow   
## 32 265 1337 55 723

### 2.3 Merge similar modules

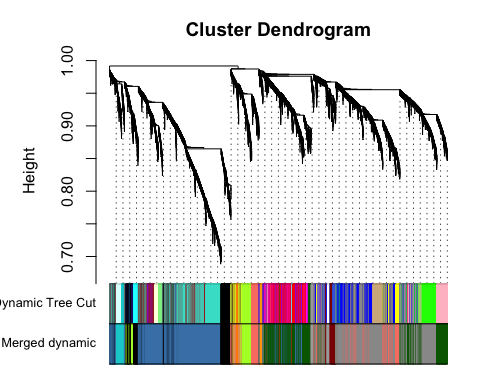
# Calculate eigengenes  
MEList = moduleEigengenes(datExpr, colors = dynamicColors)  
MEs = MEList$eigengenes  
  
# Calculate dissimilarity of module eigengenes  
MEDiss = 1-cor(MEs, method = "kendall");  
  
# Cluster module eigengenes  
METree = hclust(as.dist(MEDiss), method = "average");  
# Plot the result  
# sizeGrWindow(7, 8)  
plot(METree, main = "Clustering of module eigengenes",  
 xlab = "", sub = "MEDiss = 1-cor(MEs, method = 'kendall')")  
  
# We choose a height cut of 0.2, corresponding to correlation of 0.8, to merge  
MEDissThres = 0.2 # user-specified parameter value; see WGCNA manual for more info  
  
# Plot the cut line into the dendrogram  
abline(h=MEDissThres, col = "red")



# Call an automatic merging function  
merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = MEDissThres, verbose = 3)

## mergeCloseModules: Merging modules whose distance is less than 0.2  
## multiSetMEs: Calculating module MEs.  
## Working on set 1 ...  
## moduleEigengenes: Calculating 30 module eigengenes in given set.  
## multiSetMEs: Calculating module MEs.  
## Working on set 1 ...  
## moduleEigengenes: Calculating 13 module eigengenes in given set.  
## multiSetMEs: Calculating module MEs.  
## Working on set 1 ...  
## moduleEigengenes: Calculating 12 module eigengenes in given set.  
## Calculating new MEs...  
## multiSetMEs: Calculating module MEs.  
## Working on set 1 ...  
## moduleEigengenes: Calculating 12 module eigengenes in given set.

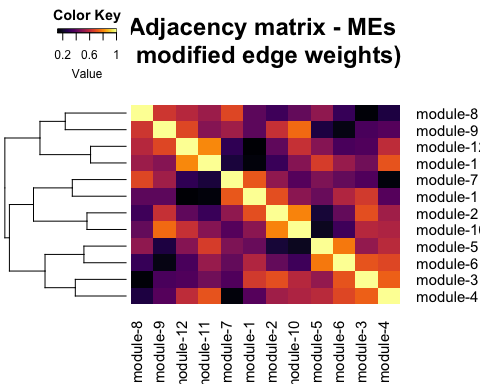
# The merged module colors  
mergedColors = merge$colors;  
# Eigengenes of the new merged modules:  
mergedMEs = merge$newMEs;  
  
# sizeGrWindow(12, 9)  
plotDendroAndColors(geneTree,  
 cbind(dynamicColors, mergedColors),  
 c("Dynamic Tree Cut", "Merged dynamic"),  
 dendroLabels = FALSE, hang = 0.03,  
 addGuide = TRUE, guideHang = 0.05)



# Rename to moduleColors  
moduleColors = mergedColors  
  
# Construct numerical labels corresponding to the colors  
colorOrder = c("grey", standardColors(50));  
moduleLabels = match(moduleColors, colorOrder)-1

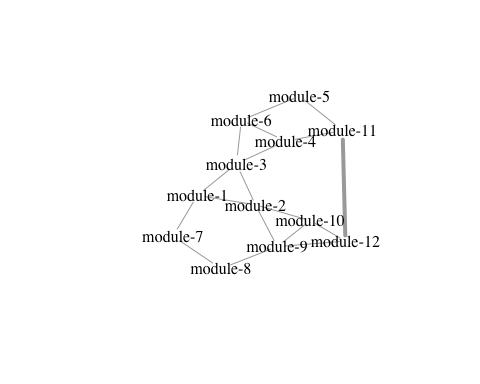
### 2.4 Calculate module-module similarity

# Calculate similarity of the eigen-genes  
sim\_matrix\_ME <- cor(mergedMEs, method = "kendall")  
  
# calculate adj\_matrix  
adj\_matrix\_ME <- adjacency.fromSimilarity(sim\_matrix\_ME,  
 power=1, # DO NOT power transform  
 type='signed'  
)  
  
# coerce into a matrix  
  
## GET THE NAMES OF THE MODULES  
# module\_ids <- rownames(adj\_matrix\_ME)  
## CHANGE THE NAMES OF THE MODULES  
module\_ids <- data.frame(old\_labels = rownames(adj\_matrix\_ME),  
 new\_labels = paste0("module-", 1:nrow(adj\_matrix\_ME)))  
  
adj\_matrix\_ME <- matrix(adj\_matrix\_ME, nrow=nrow(adj\_matrix\_ME))  
rownames(adj\_matrix\_ME) <- module\_ids$new\_labels  
colnames(adj\_matrix\_ME) <- module\_ids$new\_labels  
  
gplots::heatmap.2(t(adj\_matrix\_ME),  
 col=inferno(100),  
 # labRow=NA, labCol=NA,  
 trace='none', dendrogram='row',  
 xlab='', ylab='',  
 # main='Similarity matrix - MEs \n correlation method = "kendall")',  
 main='Adjacency matrix - MEs \n modified edge weights)',  
 density.info='none', revC=TRUE)



### 2.5 Visualize the network

pacman::p\_load(igraph)  
  
# get rid of low correlations (0.6 & 0.8 are arbitrary)  
adj\_matrix\_ME[adj\_matrix\_ME < 0.6] <- 0  
adj\_matrix\_ME[adj\_matrix\_ME < 0.8 & adj\_matrix\_ME>0] <- 0.5  
adj\_matrix\_ME[adj\_matrix\_ME >= 0.8] <- 1  
  
# build\_network  
network <- graph.adjacency(adj\_matrix\_ME,  
 mode = "upper",  
 weighted = T,  
 diag = F)  
  
# simplify network  
network <- igraph::simplify(network) # removes self-loops  
  
# E(network)$width <- E(network)$weight + min(E(network)$weight) + 1 # offset=1  
  
colors <- mergedMEs %>% names() %>% str\_split("ME", 2) %>% sapply("[", 2)  
V(network)$color <- colors  
  
genes\_ME <- factor(moduleColors, levels=colors) %>% summary()  
V(network)$size <- log2(genes\_ME)\*2  
  
V(network)$label.color <- "black"  
V(network)$frame.color <- "white"  
  
E(network)$width <- E(network)$weight^2\*4  
E(network)$edge.color <- "gray80"  
  
# par(mar=c(0,0,0,0))  
# remove unconnected nodes  
# network <- delete.vertices(network, degree(network)==0)  
# plot(network,  
# layout=layout.fruchterman.reingold  
# # layout = layout.kamada.kawai  
# # layout = layout.kamada.kawai  
# )  
  
  
## Circular layout  
plot(network,  
 layout=layout.kamada.kawai,  
 # layout=layout.fruchterman.reingold  
 # layout=layout.graphopt  
 # layout=layout\_in\_circle,  
 # vertex.label=NA  
 # vertex.size=hub.score(network)$vector\*30  
 vertex.shape="none"  
)



## Step 3: Annotate the network

### 3.1 Define your genes of interest

# DEFINE GENES OF INTEREST  
  
rhy.trait.24 <- tbl(db, "ejtk\_all") %>% select(gene\_name:rhy) %>% collect()  
# pull the genes  
for.rhy <- rhy.trait.24 %>% filter(caste=="for" & rhy=="yes") %>% pull(gene\_name)  
nur.rhy <- rhy.trait.24 %>% filter(caste=="nur" & rhy=="yes") %>% pull(gene\_name)  
rhy.genes <- dplyr::union(for.rhy, nur.rhy)  
  
rhy.trait.8 <- tbl(db, "ejtk\_8h\_all") %>% select(gene\_name:rhy) %>% collect()  
for.rhy.8 <- rhy.trait.8 %>% filter(caste=="for" & rhy=="yes") %>% pull(gene\_name)  
nur.rhy.8 <- rhy.trait.8 %>% filter(caste=="nur" & rhy=="yes") %>% pull(gene\_name)  
  
rhy.trait.12 <- tbl(db, "ejtk\_12h\_all") %>% select(gene\_name:rhy) %>% collect()  
for.rhy.12 <- rhy.trait.12 %>% filter(caste=="for" & rhy=="yes") %>% pull(gene\_name)  
nur.rhy.12 <- rhy.trait.12 %>% filter(caste=="nur" & rhy=="yes") %>% pull(gene\_name)  
  
# DRGs  
for24.nur8 <- intersect(for.rhy, nur.rhy.8)

### 3.2 Where are my genes of interest located?

pacman::p\_load(GeneOverlap)  
# https://www.bioconductor.org/packages/devel/bioc/vignettes/GeneOverlap/inst/doc/GeneOverlap.pdf  
  
# Make a list that returns gene names for a given cluster  
module\_color = colors  
module = names(mergedMEs)  
module\_colors <-  
 data.frame(module\_label=module) %>%  
 mutate(module\_color = str\_replace(module\_label, "ME", ""))  
  
module\_genes <- list()  
module\_color <- module\_colors$module\_color  
# Get the genes from each of the modules  
for (i in 1:length(module\_color)) {  
  
 module\_genes[[i]] <- names(datExpr)[which(moduleColors==module\_color[[i]])]  
 names(module\_genes)[[i]] <- module\_color[[i]]  
}  
# check the result | works  
names(module\_genes) <- module\_ids$new\_labels  
# module\_genes['salmon']

# Let's change the name of the modules for easy reading  
# names(module\_genes) <- paste0("module-", 1:length(module\_genes))

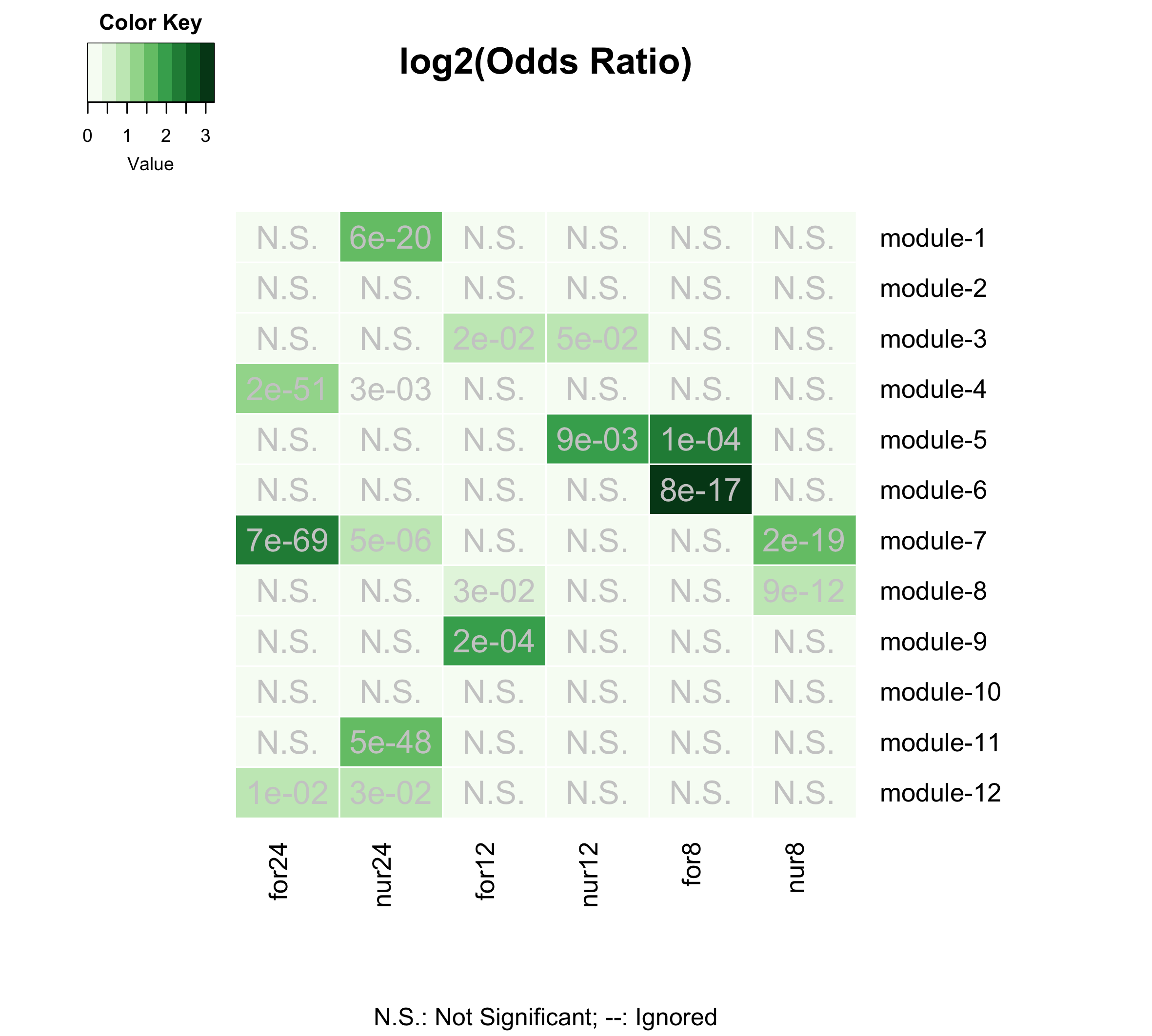
## MAKE YOUR LIST OF GENES OF INTEREST ##  
  
# LIST ONE - WGCNA modules  
list1 <- module\_genes  
sapply(list1, length)

## module-1 module-2 module-3 module-4 module-5 module-6 module-7 module-8   
## 403 56 922 2269 127 179 664 2616   
## module-9 module-10 module-11 module-12   
## 209 66 1533 95

## LIST TWO - rhythmic genes  
list2 <- list(for.rhy, nur.rhy, for.rhy.12, nur.rhy.12, for.rhy.8, nur.rhy.8)  
names(list2) <- c("for24", "nur24", "for12","nur12", "for8", "nur8")  
sapply(list2, length)

## for24 nur24 for12 nur12 for8 nur8   
## 3569 1367 148 193 229 550

## CHECK FOR OVERLAP  
  
## make a GOM object  
gom.1v2 <- newGOM(list1, list2,  
 genome.size = nGenes)  
png(paste0(path\_to\_repo, "/results/figures/gom\_1v2.png"),   
 width = 20, height = 18, units = "cm", res = 300)  
drawHeatmap(gom.1v2,  
 adj.p=T,  
 cutoff=0.05,  
 what="odds.ratio",  
 # what="Jaccard",  
 log.scale = T,  
 note.col = "grey80")  
trash <- dev.off()



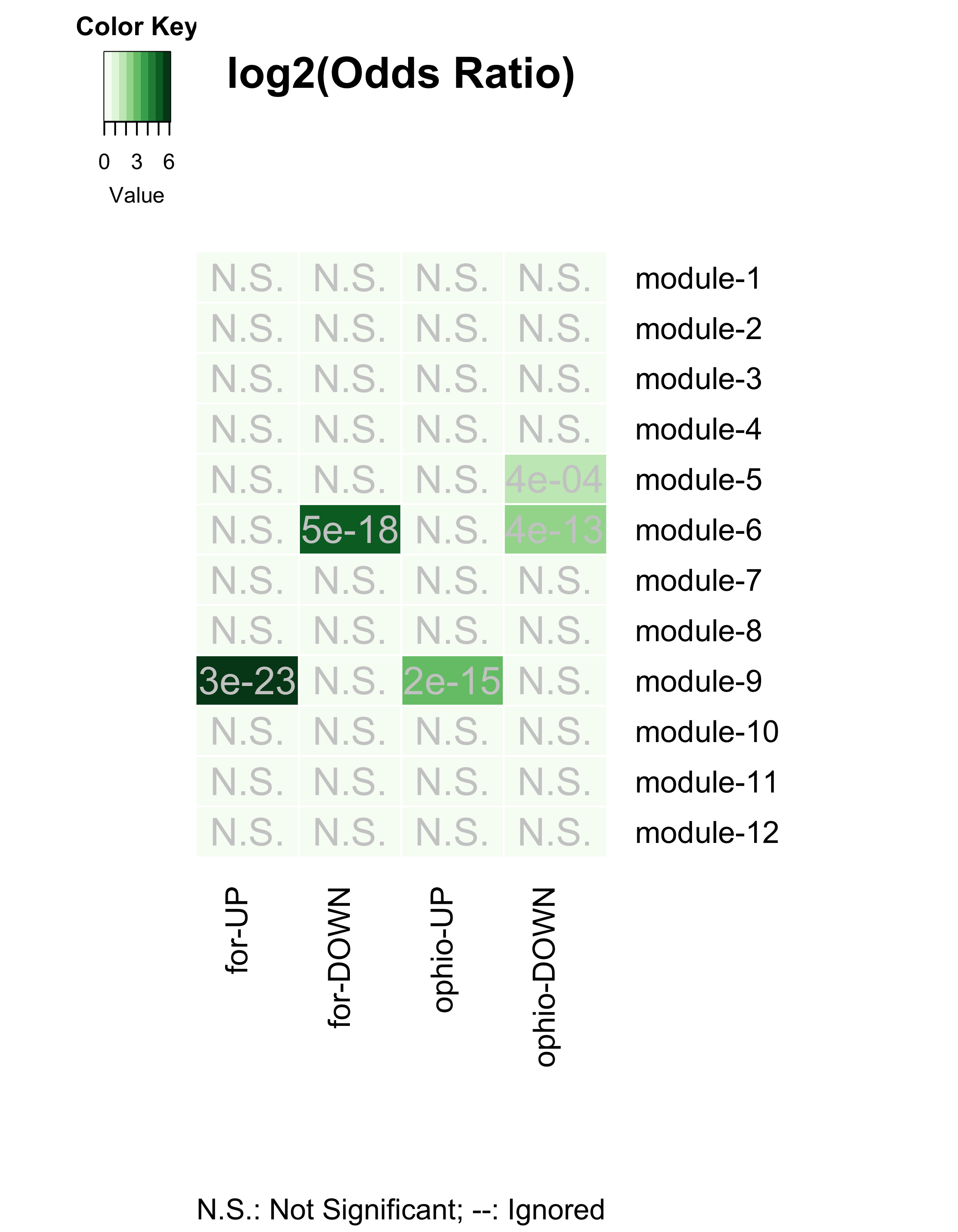
Gene-clusters with 24h-rhythmic genes

The above plot shows the gene-clusters that contain most of our 24h-rhythmic genes of interest.

HOW TO READ THE FIGURE? - Briefly explain here - Talk about Odds-Ratio - Darker the green, more significant is the overlap (also indicated by the adj\_pval)

Next, we can try to identify the ant gene-clusters that underlie behavioral plasticity, as well as the ant clusters that are affected during behavioral manipulation induced by *Ophiocordyceps*.

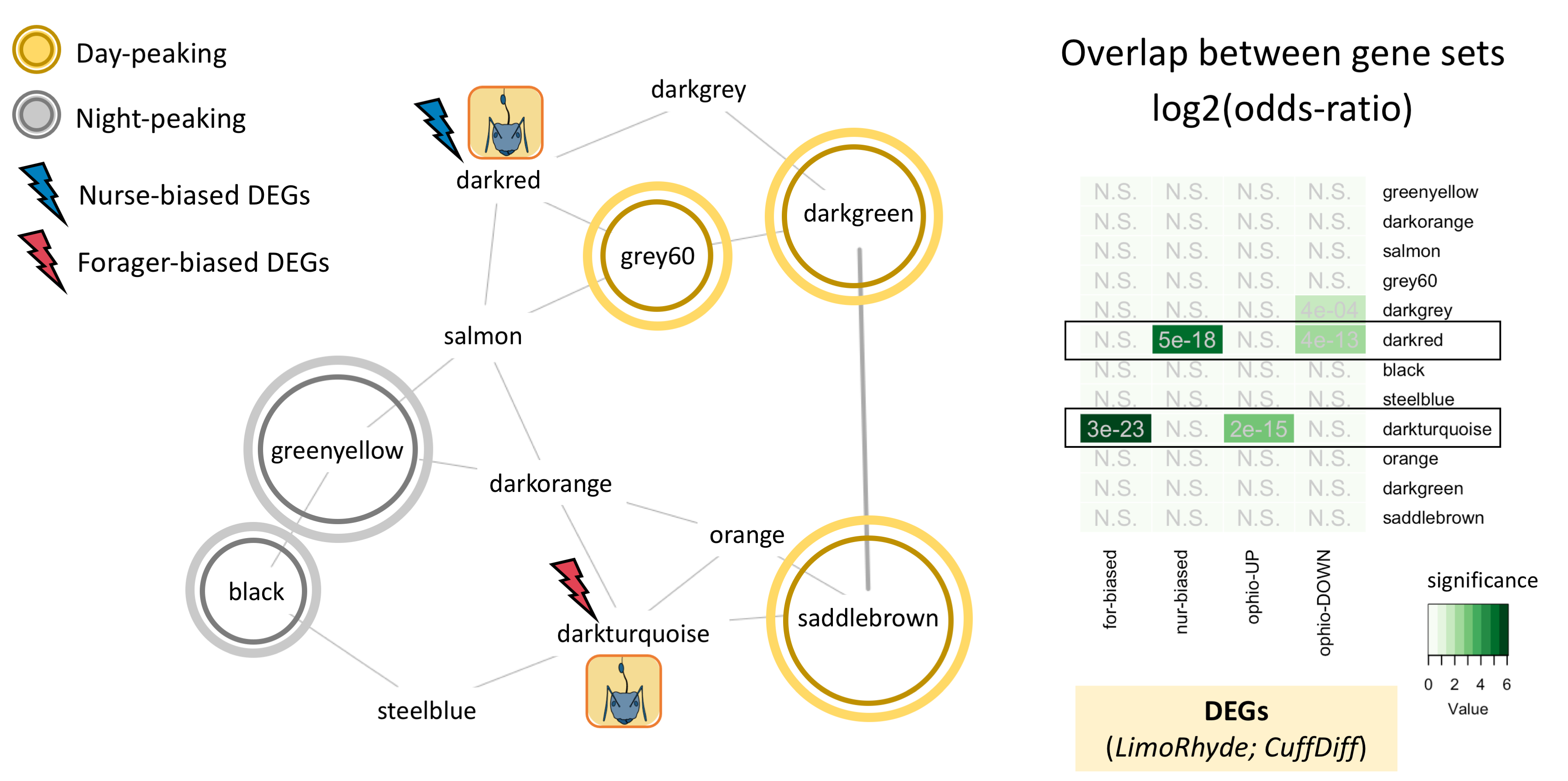
## Genes underlying behavioral plasticity  
 ## DEGS (foragers v. nurses)  
 # genes higher expressed in forager brains (v. nurse brains)  
 for.up <- tbl(db, "TC5\_DEGs\_all") %>% filter(upregulation=="for") %>% collect() %>% pull(gene\_name)  
 # genes lower expressed in for. brains (v. nurse brains)  
 for.down <- tbl(db, "TC5\_DEGs\_all") %>% filter(upregulation=="nur") %>% collect() %>% pull(gene\_name)  
  
## Genes underlying parasite-induced behavioral manipulation  
 ## DEGs (ophio-ant v. control-ant)  
 ophio.dat <- tbl(db, "ophio\_biting\_control") %>% collect() %>% select(gene, value\_1, value\_2, q\_value:logFC)  
 ophio.dat <- ophio.dat %>%  
 filter(abs(logFC) >= 1 & significant=="yes" & q\_value < 0.05) %>%  
 mutate(ophio = ifelse(logFC > 0, "down", "up"))  
 # genes higher expressed in ant heads during Ophio-manipulated biting (v. controls)  
 ophio.up <- ophio.dat %>% filter(ophio=="up") %>% pull(gene)  
 # genes lower expressed in ant heads during manipulated biting (v. controls)  
 ophio.down <- ophio.dat %>% filter(ophio=="down") %>% pull(gene)  
  
## LIST THREE - genes underlying behavioral plasticity and parasitic behavioral manipulation  
list3 <- list(for.up, for.down, # same as list three  
 ophio.up, ophio.down)  
names(list3) <- c("for-UP", "for-DOWN",  
 "ophio-UP", "ophio-DOWN")  
  
## CHECK FOR OVERLAP  
  
## make a GOM object  
gom.1v3 <- newGOM(list1, list3,  
 genome.size = nGenes)  
## visualize the overlaps  
png(paste0(path\_to\_repo, "/results/figures/gom\_1v3.png"),   
 width = 14, height = 18, units = "cm", res = 400)  
drawHeatmap(gom.1v3,  
 adj.p=T,  
 cutoff=0.05,  
 what="odds.ratio",  
 # what="Jaccard",  
 log.scale = T,  
 note.col = "grey80")  
trash <- dev.off()



Gene-clusters underlying behavioral plasiticity and parasitic behavioral manipulation

The figure above clearly indicates that the gene-clusters that underlie behavioral plasticity (caste differentiation) and the ones that are affected during *Ophiocordycpes*-induced behavioral manipulation are the same.

**In other words, to induce the characteristic manipulated biting behavior, the manipulating fungal parasite seems to be targeting the same genes and processes that otherwise allow ants to display behavioral plasticity.**



Annotated circadian GCN

## Step 4: Explore your clusters-of-interest

### 4.1 Cluster: module-9

#### 4.1.1 What are these overlapping genes?

* Let’s focus on the cluster module-9/module-9 that contains most Cflo genes that:
* are sig. higher expressed in foragers (v. nurses) and
* are sig. up-regulated in forager heads during behavioral manipulation (v. uninfected foragers)

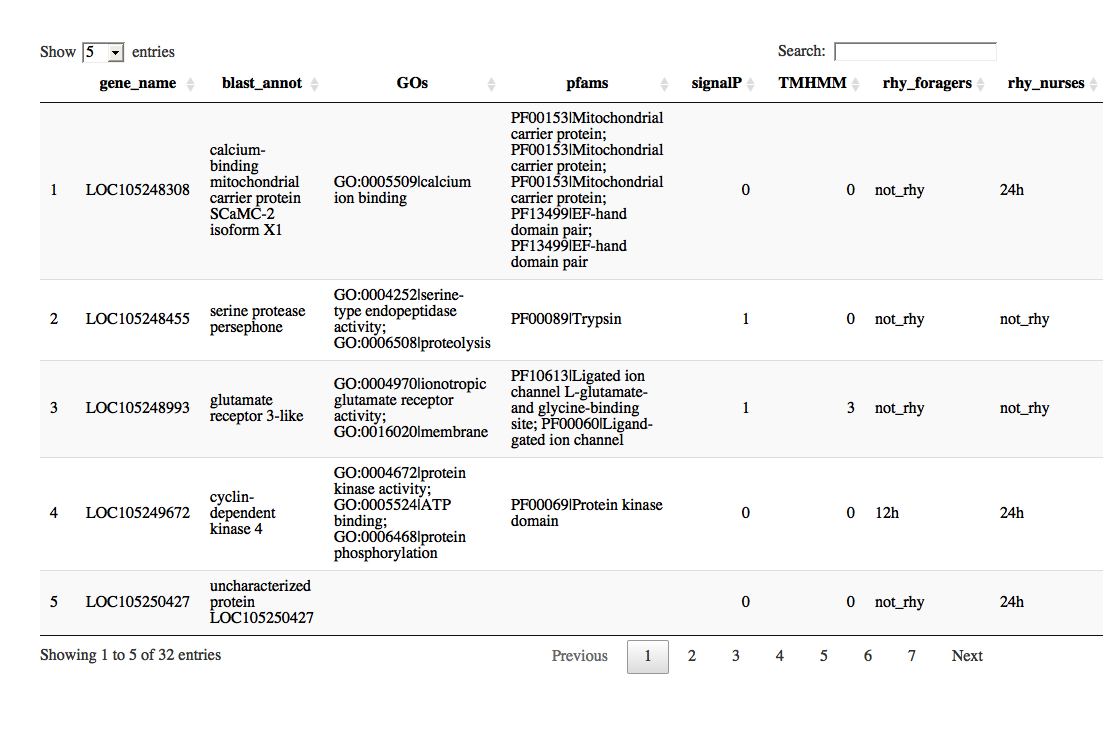
# specify our cluster of interest (coi)  
coi.1 <- "module-9"  
# How many genes are there in the cluster?  
module\_genes[[coi.1]] %>% length() # n = 209 genes

## [1] 209

# specify our genes of interest (goi)  
goi.1 <- ophio.up  
# how many genes are there in the gene-set?  
goi.1 %>% length() # n = 232 genes

## [1] 232

# Identify overlapping genes  
overlapping.genes.1 <- intersect(module\_genes[[coi.1]], goi.1) # n = 32 genes  
# what are these genes?  
overlapping.genes.1.annot <-  
 db %>%   
 tbl(., "annot\_fpkm") %>%   
 filter(gene\_name %in% overlapping.genes.1) %>%   
 select(gene\_name,   
 blast\_annot=old\_annotation,   
 GOs, pfams, signalP, TMHMM) %>%   
 collect() %>%   
 # add a column that indicates if the gene is rhythmic or not  
 ## in forager brains   
 mutate(rhy\_foragers = ifelse(gene\_name %in% for.rhy, "24h",  
 ifelse(gene\_name %in% for.rhy.12, "12h",  
 ifelse(gene\_name %in% for.rhy.8, "8h", "not\_rhy")))) %>%   
 ## in nurse brains  
 mutate(rhy\_nurses = ifelse(gene\_name %in% nur.rhy, "24h",  
 ifelse(gene\_name %in% nur.rhy.12, "12h",  
 ifelse(gene\_name %in% nur.rhy.8, "8h", "not\_rhy"))))  
  
# Visualize the results  
DT::datatable(overlapping.genes.1.annot, options = list(  
 pageLength = 5,  
 lengthMenu = c(5, 10, 15, 20)  
))

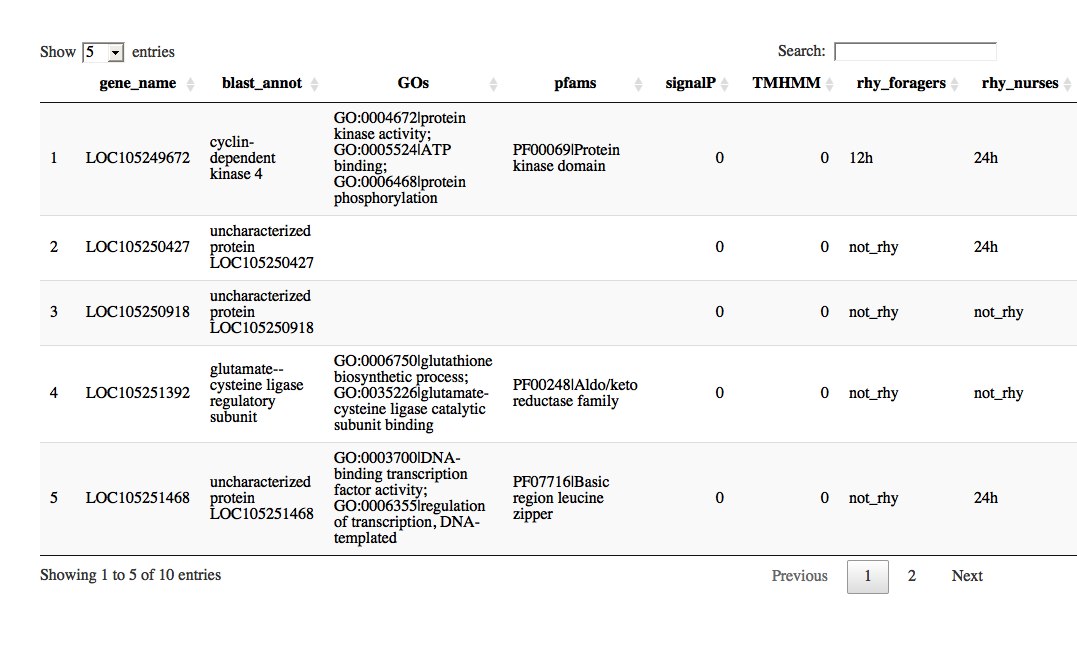


Which genes in module-9 show overlap with both genes of interests?

# define the second gene-set of interst  
goi.2 <- for.up  
# how many genes are up-regulated during manipulation?  
goi.2 %>% length() # n = 34 genes

## [1] 34

# Identify overlapping.genes  
overlapping.genes.2 <- intersect(overlapping.genes.1, goi.2) # n = 10 genes  
# what are these genes?  
overlapping.genes.2.annot <-   
 db %>%   
 tbl(., "annot\_fpkm") %>%   
 filter(gene\_name %in% overlapping.genes.2) %>%   
 select(gene\_name,   
 blast\_annot=old\_annotation,   
 GOs, pfams, signalP, TMHMM) %>%   
 collect() %>%   
 # add a column that indicates if the gene is rhythmic or not  
 ## in forager brains   
 mutate(rhy\_foragers = ifelse(gene\_name %in% for.rhy, "24h",  
 ifelse(gene\_name %in% for.rhy.12, "12h",  
 ifelse(gene\_name %in% for.rhy.8, "8h", "not\_rhy")))) %>%   
 ## in nurse brains  
 mutate(rhy\_nurses = ifelse(gene\_name %in% nur.rhy, "24h",  
 ifelse(gene\_name %in% nur.rhy.12, "12h",  
 ifelse(gene\_name %in% nur.rhy.8, "8h", "not\_rhy"))))  
  
  
# Visualize the results  
DT::datatable(overlapping.genes.2.annot, options = list(  
 pageLength = 5,  
 lengthMenu = c(5, 10, 15, 20)  
))



#### 4.1.2 What’s special about my cluster?

Now that we know that the module **module-9** contains most of our genes of interest, we can infer its function (enriched GOs and PFAMs) and also identify the genes that are important for the cluster to be functional (i.e., hub genes).

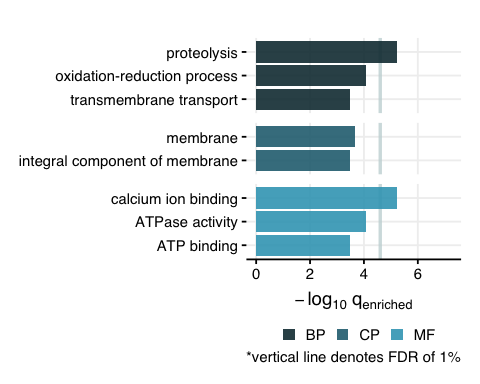
This is the primary advantage of systems-level analysis: - Use different sources of evidence to identify the clusters in the network that are of interest, - Analyze the cluster-of-interest

#### 4.1.3 Enriched GO terms

First up, let’s see which processes are overrepresented in the cluster.

# To run a functional enrichment analyis, we first need to define the set of background genes; for our purpose, we will use the 9139 genes that we used to build our circadian GCN  
bg.genes <- dat %>% pull(gene\_name)  
  
# Run the enrichment function (note, GO HERE TO READ MORE ABOUT THIS FUNCTION)  
# png(paste0(path\_to\_repo, "/results/figures/module\_9\_enrichments.png"),   
# width = 16, height = 10, units = "cm", res = 400)  
go\_enrichment(geneset = module\_genes[[coi.1]],  
 function.dir = path\_to\_repo,  
 org = "cflo",   
 bg = bg.genes) %>% #view()  
   
 # visualize the results  
 go\_enrichment\_plot(function.dir = path\_to\_repo, clean = "yes")

## [1] "Loading annotation file for Camponotus floridanus"  
## [1] "Done."  
## [1] "Number of genes in background geneset: 9139"  
## [1] "Number of genes in the test set: 209"  
## [1] "--------------------------------"  
## [1] "Number of GO terms in background geneset: 1968"  
## [1] "Number of GO terms (at least 5genes) in background geneset: 373"  
## [1] "Number of GO terms (at least 5genes) in test set: 13"  
## [1] "Testing for enrichment..."

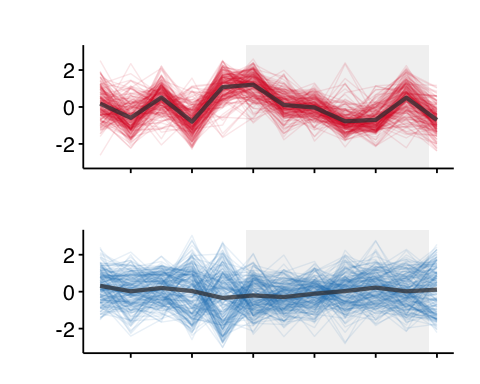


# trash <- dev.off()

#### 4.1.4 Daily rhythms?

Second, let’s plot the daily expression patterns of all genes in the cluster, for nurses and foragers.

# Obtain the stacked z-plots for nurses (blue) and foragers (red)  
zplots.module <-   
 module\_genes[[coi.1]] %>%   
 stacked.zplot()  
  
# Plot them side by side  
zplots.module[[1]] / zplots.module[[2]]



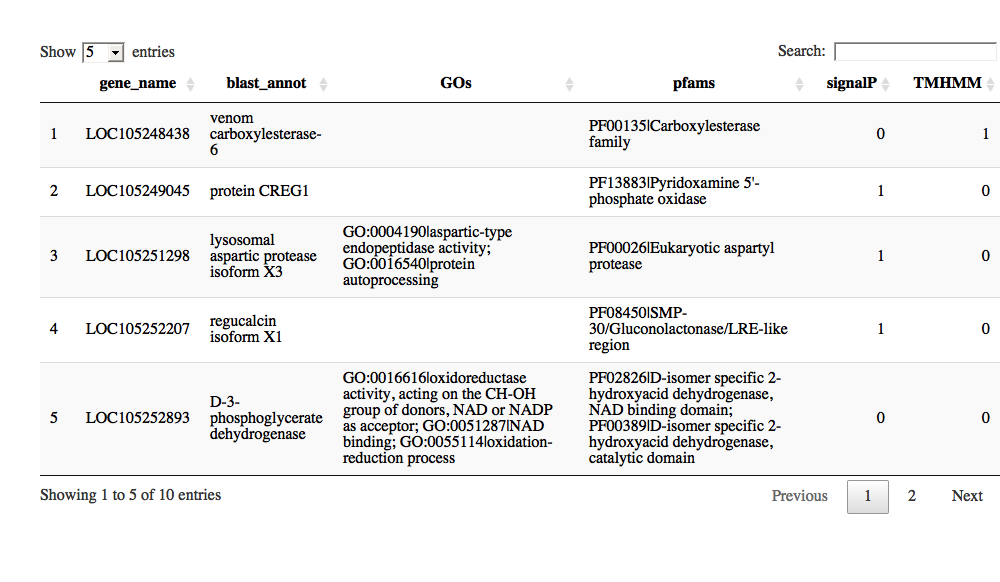
LEGEND: **RED** = Forager brains, **BLUE** = Nurse brains

#### 4.1.5 HUB genes?

Need to: - identify the hub genes in the module-9 cluster - other genes of interest based on their location in the network?

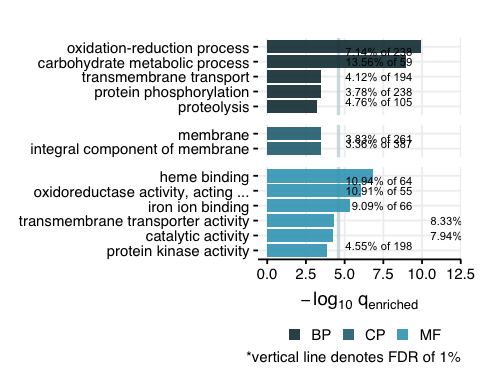
### 4.2 Cluster: module-6

#### 4.2.1 Overlapping genes

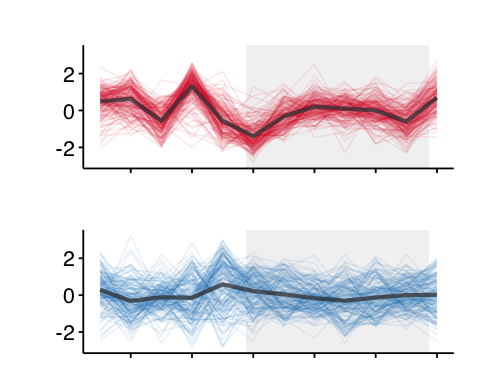


#### 4.2.2 Enriched GO terms

## [1] "Loading annotation file for Camponotus floridanus"  
## [1] "Done."  
## [1] "Number of genes in background geneset: 9139"  
## [1] "Number of genes in the test set: 179"  
## [1] "--------------------------------"  
## [1] "Number of GO terms in background geneset: 1968"  
## [1] "Number of GO terms (at least 5genes) in background geneset: 373"  
## [1] "Number of GO terms (at least 5genes) in test set: 16"  
## [1] "Testing for enrichment..."



#### 4.2.3 Daily rhythms?



LEGEND: **RED** = Forager brains, **BLUE** = Nurse brains

#### 4.3.4 HUB genes?

coming soon…

### 4.3 Cluster: module-12

Connected to the forager-cluster (module-9)

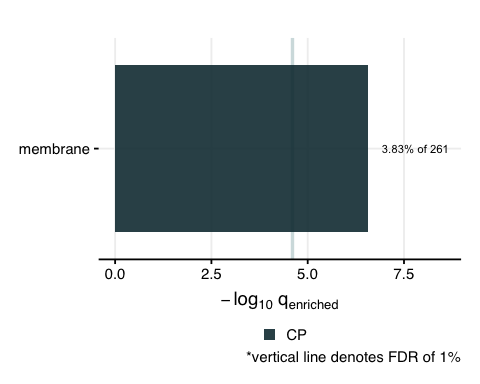
#### 4.3.1 Explore cluster

## How many genes are there in the module-12 cluster?

## [1] 95

#### 4.3.2 Enriched GO terms

## [1] "Loading annotation file for Camponotus floridanus"  
## [1] "Done."  
## [1] "Number of genes in background geneset: 9139"  
## [1] "Number of genes in the test set: 95"  
## [1] "--------------------------------"  
## [1] "Number of GO terms in background geneset: 1968"  
## [1] "Number of GO terms (at least 5genes) in background geneset: 373"  
## [1] "Number of GO terms (at least 5genes) in test set: 5"  
## [1] "Testing for enrichment..."



### 4.4 Cluster: module-4

Connected to the nurse-cluster (module-6)

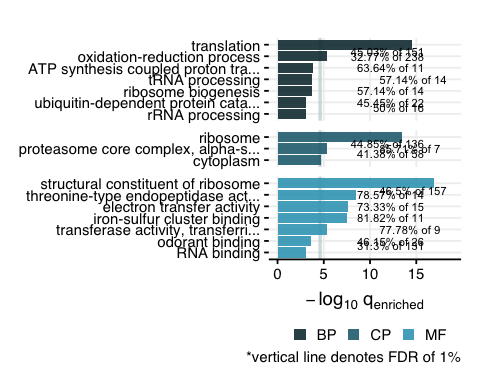
#### 4.4.1 Explore cluster

## How many genes are there in the module-4 cluster?

## [1] 2269

#### 4.4.2 Enriched GO terms

## [1] "Loading annotation file for Camponotus floridanus"  
## [1] "Done."  
## [1] "Number of genes in background geneset: 9139"  
## [1] "Number of genes in the test set: 2269"  
## [1] "--------------------------------"  
## [1] "Number of GO terms in background geneset: 1968"  
## [1] "Number of GO terms (at least 5genes) in background geneset: 373"  
## [1] "Number of GO terms (at least 5genes) in test set: 92"  
## [1] "Testing for enrichment..."



### 4.5 Cluster: module-11

Indirectly connected to forager- and nurse-cluster (and highly correlated to module-12)

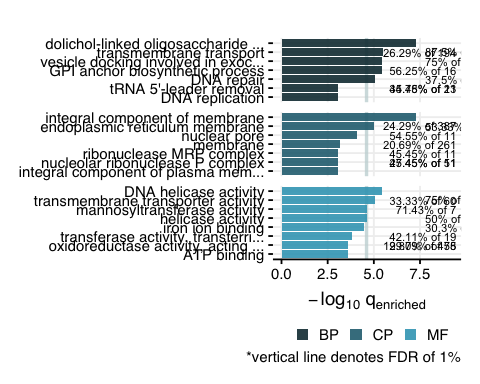
#### 4.5.1 Explore cluster

## How many genes are there in the module-11 cluster?

## [1] 1533

#### 4.4.2 Enriched GO terms

## [1] "Loading annotation file for Camponotus floridanus"  
## [1] "Done."  
## [1] "Number of genes in background geneset: 9139"  
## [1] "Number of genes in the test set: 1533"  
## [1] "--------------------------------"  
## [1] "Number of GO terms in background geneset: 1968"  
## [1] "Number of GO terms (at least 5genes) in background geneset: 373"  
## [1] "Number of GO terms (at least 5genes) in test set: 72"  
## [1] "Testing for enrichment..."



## Step 5: Random code chunks

# module-9 genes ----------------------------------------------------------  
  
db %>%   
 tbl(., "annot\_fpkm") %>%   
 filter(gene\_name %in% overlapping.genes.2) %>%   
 select(gene\_name,   
 blast\_annot=old\_annotation,   
 GOs, pfams, signalP, TMHMM) %>%   
 collect() %>%   
   
 # add a column that indicates if the gene is rhythmic or not  
 ## in forager brains   
 mutate(rhy\_foragers = ifelse(gene\_name %in% for.rhy, "24h",  
 ifelse(gene\_name %in% for.rhy.12, "12h",  
 ifelse(gene\_name %in% for.rhy.8, "8h", "not\_rhy")))) %>%   
 ## in nurse brains  
 mutate(rhy\_nurses = ifelse(gene\_name %in% nur.rhy, "24h",  
 ifelse(gene\_name %in% nur.rhy.12, "12h",  
 ifelse(gene\_name %in% nur.rhy.8, "8h", "not\_rhy")))) %>%   
   
 # add a column to indicate if the gene is for-up/down and up/down-regulated at manipulation  
 mutate(identity = "upregulated in foragers; upregulated at manipulation") %>%   
   
   
   
 write.csv(., "./results/genes\_of\_interest/genes\_for-up\_manipulation-up\_cluster9.csv", row.names = F)  
  
  
  
# module-6 genes ----------------------------------------------------------  
db %>%   
 tbl(., "annot\_fpkm") %>%   
 filter(gene\_name %in% overlapping.genes.4) %>%   
 select(gene\_name,   
 blast\_annot=old\_annotation,   
 GOs, pfams, signalP, TMHMM) %>%   
 collect() %>%   
   
 # add a column that indicates if the gene is rhythmic or not  
 ## in forager brains   
 mutate(rhy\_foragers = ifelse(gene\_name %in% for.rhy, "24h",  
 ifelse(gene\_name %in% for.rhy.12, "12h",  
 ifelse(gene\_name %in% for.rhy.8, "8h", "not\_rhy")))) %>%   
 ## in nurse brains  
 mutate(rhy\_nurses = ifelse(gene\_name %in% nur.rhy, "24h",  
 ifelse(gene\_name %in% nur.rhy.12, "12h",  
 ifelse(gene\_name %in% nur.rhy.8, "8h", "not\_rhy")))) %>%   
   
 # add a column to indicate if the gene is for-up/down and up/down-regulated at manipulation  
 mutate(identity = "upregulated in nurses; downregulated at manipulation") %>%   
   
 write.csv(., "./results/genes\_of\_interest/genes\_nur-up\_manipulation-down\_cluster6.csv", row.names = F)  
  
  
# Overlap between plasticity and manipulation genes ------------------------  
  
list.plasticity <- list(for.up, for.down)  
names(list.plasticity) <- c("for-UP", "for-DOWN")  
  
list.manipulation <- list(ophio.up, ophio.down)  
names(list.manipulation) <- c("ophio-UP", "ophio-DOWN")  
  
gom.plasticity\_v\_manipulation <- newGOM(list.plasticity, list.manipulation, genome.size = nGenes)  
png(paste0(path\_to\_repo, "/results/figures/gom\_plasticity\_v\_manipulation.png"),   
 width = 12, height = 12, units = "cm", res = 400)  
drawHeatmap(gom.plasticity\_v\_manipulation,  
 adj.p=T,  
 cutoff=0.05,  
 what="odds.ratio",  
 # what="Jaccard",  
 log.scale = T,  
 note.col = "grey80")  
dev.off()  
  
  
# Odds ratio and fisher’s exact test --------------------------------------  
  
# for-up and module-9  
test.table <-   
 data.frame(  
 in.module = c(20,189),  
 not.module = c(14,8916)  
 )  
  
rownames(test.table) <- c("in.for.up", "not.for.up")  
  
fisher.test(test.table)  
  
# for-down and module-6  
test.table <-   
 data.frame(  
 in.module = c(18,161),  
 not.module = c(29,8931)  
 )  
  
rownames(test.table) <- c("in.for.down", "not.for.down")  
  
fisher.test(test.table)  
  
# ophio-up and module-9  
test.table <-   
 data.frame(  
 in.module = c(32,177),  
 not.module = c(177,8730)  
 )  
  
rownames(test.table) <- c("in.ophio.up", "not.ophio.up")  
  
fisher.test(test.table)  
  
# ophio-down and module-6  
test.table <-   
 data.frame(  
 in.module = c(42,137),  
 not.module = c(137,8428)  
 )  
  
rownames(test.table) <- c("in.ophio.down", "not.ophio.down")  
  
fisher.test(test.table)$p.value