Final Report

Jennifer Chang

April 25, 2014

Motivation:

Organisms are robust systems that will try to compensate for different types of environmental stress effects, adjusting the expression of genes and their protein products accordingly. This project seeks to gain insight into what pathways are affected by each type of stress. The related pathways may give some idea on how the organism adapts to the environment.

The model organism we're looking at is E coli. The expression levels of 4495 genes were recorded under different stressful conditions.

Questions of Interest:

- Which genes have similar expression profiles among the different treatments?
- Are genes with similar expression profiles located near each other in biological pathways?

Data Available:

- Expression data for 4454 genes in E. coli (microarray experiment) under different stressful conditions.
  - From another student in my lab
- KEGG pathways data from kegg.jp

Load Libraries

```
library(ggplot2)
library(reshape)
library(GGally)
library(XML)
library(WGCNA)
```

E Coli Data

```
data <- read.csv("data/gene_median.csv", header = TRUE, stringsAsFactors = FALSE)
data <- subset(data, SD1 != -1)  # dropped anything that had -1 across all values
data <- data[, c(1:3, 5:18, 20, 21)]  # Drop C1 and PH2, (bubble on the array)
# names(data)<-c('Name','Minimal C-source 2','Minimal C & N Source
# 1','Minimal C & N Source 2','Cold Shock rep 1','Cold Shock rep 2','Heat
# Shock rep 1','Heat Shock rep 2','Minimal N Source 1','Minimal N Source
# 2','Osmotic Shock 1','Osmotic Shock 2','Oxidative Stress 1','Oxidative
# Stress 2','Low pH 1','Control 1','Control 2','UV Treatment 1','UV
# Treatment 2')
dim(data)
```

```
## [1] 4454   19
```

```
head(data)
```

```
##   Name  SD1  SD2  C.2 C.N.1 C.N.2  N.1  N.2  CS1  CS2  HS1  HS2 OSMO1
## 1 aaaD 3794 4669 4454  3959  5013 5114 2782 3875 4297 3594 5451  3259
## 2 aaaE 6006 8089 6045  7254  7540 7078 6873 4987 6720 7362 7564  7719
## 3 aaeA 3902 2599 3703  3410  2992 3871 3864 4079 3808 3170 3538  2481
## 4 aaeB 3623 3544 4213  4025  3411 4170 4454 3948 4061 3730 3023  4287
## 5 aaeR 4914 5957 6575  5096  5789 6017 6640 4355 5097 6729 5982  6825
## 6 aaeX 5030 3633 5731  2915  5076 4392 5255 5139 4524 5416 5358  4769
##   OSMO2 OXI1 OXI2  PH1  UV1  UV2
## 1  3707 4115 3864 3154 3690 4588
## 2  5649 7910 6834 5839 5754 6982
## 3  2968 3512 3045 3352 3811 3650
## 4  3625 3434 4158 2618 3985 3872
## 5  6197 6800 5486 4928 6927 4889
## 6  5143 6026 5126 5165 5155 4004
```

```
summary(data)
```

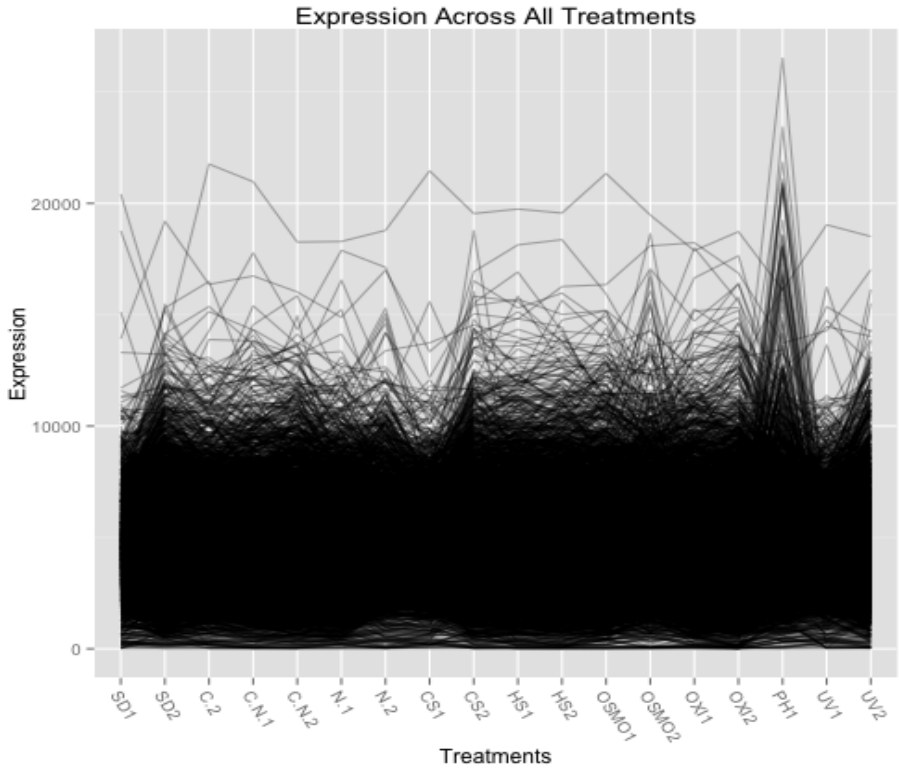
```
##      Name                SD1             SD2             C.2       
##  Length:4454        Min.   :   19   Min.   :   17   Min.   :   21  
##  Class :character   1st Qu.: 3371   1st Qu.: 3013   1st Qu.: 3164  
##  Mode  :character   Median : 4350   Median : 4420   Median : 4386  
##                     Mean   : 4478   Mean   : 4711   Mean   : 4626  
##                     3rd Qu.: 5423   3rd Qu.: 6114   3rd Qu.: 5768  
##                     Max.   :20421   Max.   :19200   Max.   :21762  
##      C.N.1           C.N.2           N.1             N.2       
##  Min.   :   15   Min.   :   17   Min.   :   18   Min.   :   20  
##  1st Qu.: 3014   1st Qu.: 2951   1st Qu.: 3066   1st Qu.: 3321
```

```
##  Median : 4328   Median : 4325   Median : 4371   Median : 4418
##  Mean   : 4608   Mean   : 4630   Mean   : 4563   Mean   : 4663
##  3rd Qu.: 5855   3rd Qu.: 5968   3rd Qu.: 5816   3rd Qu.: 5688
##  Max.   :20971   Max.   :18261   Max.   :18289   Max.   :18784
##       CS1             CS2             HS1             HS2
##  Min.   :   21   Min.   :   16   Min.   :   15   Min.   :   14
##  1st Qu.: 3362   1st Qu.: 2904   1st Qu.: 2964   1st Qu.: 2919
##  Median : 4330   Median : 4344   Median : 4356   Median : 4340
##  Mean   : 4510   Mean   : 4650   Mean   : 4672   Mean   : 4660
##  3rd Qu.: 5517   3rd Qu.: 6022   3rd Qu.: 6032   3rd Qu.: 6034
##  Max.   :21460   Max.   :19540   Max.   :19744   Max.   :19565
##      OSMO1           OSMO2           OXI1            OXI2
##  Min.   :   19   Min.   :   19   Min.   :   19   Min.   :   14
##  1st Qu.: 2906   1st Qu.: 3163   1st Qu.: 2929   1st Qu.: 2850
##  Median : 4358   Median : 4376   Median : 4352   Median : 4292
##  Mean   : 4664   Mean   : 4719   Mean   : 4654   Mean   : 4637
##  3rd Qu.: 6001   3rd Qu.: 5851   3rd Qu.: 6001   3rd Qu.: 6038
##  Max.   :21342   Max.   :19475   Max.   :18230   Max.   :18727
##       PH1             UV1             UV2
##  Min.   :   15   Min.   :   22   Min.   :   16
##  1st Qu.: 3170   1st Qu.: 3417   1st Qu.: 2972
##  Median : 4342   Median : 4373   Median : 4381
##  Mean   : 4711   Mean   : 4513   Mean   : 4635
##  3rd Qu.: 5791   3rd Qu.: 5457   3rd Qu.: 5904
##  Max.   :26516   Max.   :19037   Max.   :18512
```

The data is E. coli microarry data recording the expression of genes (4472 genes in total) in each row under different stress conditions. (minimal C, N, Cold shock, etc)

The Following produces a parallel cordinate plot of gene expression across all treatments. Each line is a separate gene.

```
# scale options : std, robust, uniminmax, globalminmax, center, centerObs
ggparcoord(data, columns = c(2:19), scale = "globalminmax", alphaLines = 0.3) +
    labs(title = "Expression Across All Treatments", x = "Treatments", y = "Expression") +
    theme(axis.text.x = element_text(angle = 300, hjust = 0))
```



Create Clusters: WGCNA Analysis

Install WGCNA package (The following R code will not execute when knit because eval=FALSE)

```
install.packages(c("dynamicTreeCut", "cluster", "flashClust", "Hmisc", "reshape",
    "foreach", "doParallel"))
source("http://bioconductor.org/biocLite.R")
biocLite("impute")
install.packages("WGCNA")
```

Analyze the microarray data and cluster the genes into Modules

```
allowWGCNAThreads()
```

```
## Allowing multi-threading with up to 2 threads.
```

```
# Prepare data for input
input <- as.data.frame(t(data[, -1]))
names(input) = data$Name
head(input[, 1:6])
```

```
##       aaaD aaaE aaeA aaeB aaeR aaeX
## SD1   3794 6006 3902 3623 4914 5030
## SD2   4669 8089 2599 3544 5957 3633
## C.2   4454 6045 3703 4213 6575 5731
## C.N.1 3959 7254 3410 4025 5096 2915
## C.N.2 5013 7540 2992 3411 5789 5076
## N.1   5114 7078 3871 4170 6017 4392
```

```
# Choose soft threshold
powers = c(c(1:10), seq(from = 12, to = 30, by = 2))
sft = pickSoftThreshold(input, powerVector = powers, verbose = 5)
```

```
## pickSoftThreshold: will use block size 4454.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 4454 of  4454
##    Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
## 1      1    0.316  1.220          0.707 1500.00  1.50e+03 2290.0
## 2      2    0.131 -0.305          0.636  720.00  6.77e+02 1490.0
## 3      3    0.591 -0.788          0.797  406.00  3.49e+02 1060.0
## 4      4    0.698 -0.994          0.850  253.00  1.98e+02  802.0
## 5      5    0.755 -1.120          0.888  167.00  1.19e+02  626.0
## 6      6    0.742 -1.230          0.883  116.00  7.46e+01  501.0
## 7      7    0.780 -1.290          0.915   83.50  4.82e+01  409.0
## 8      8    0.779 -1.350          0.919   61.80  3.22e+01  338.0
## 9      9    0.774 -1.390          0.925   46.80  2.17e+01  283.0
## 10    10    0.751 -1.450          0.917   36.10  1.50e+01  240.0
## 11    12    0.768 -1.510          0.936   22.60  7.67e+00  176.0
## 12    14    0.792 -1.530          0.956   14.90  4.10e+00  133.0
## 13    16    0.804 -1.550          0.963   10.20  2.33e+00  102.0
## 14    18    0.810 -1.560          0.966    7.21  1.37e+00   80.5
## 15    20    0.795 -1.600          0.957    5.25  8.39e-01   64.3
## 16    22    0.804 -1.580          0.954    3.92  5.26e-01   52.0
## 17    24    0.823 -1.530          0.952    2.99  3.35e-01   42.6
## 18    26    0.864 -1.480          0.960    2.32  2.16e-01   35.2
## 19    28    0.932 -1.400          0.993    1.84  1.44e-01   29.7
## 20    30    0.927 -1.460          0.993    1.47  9.73e-02   27.7
```

```
sizeGrWindow(9, 5)
par(mfrow = c(1, 2))
cex1 = 0.9
```

```
plot(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
    xlab = "Soft Threshold (power)", ylab = "Scale Free Topology Model Fit, signed R^2",
    main = paste("Scale independence"))
text(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
    labels = powers, cex = cex1, col = "red")
abline(h = 0.9, col = "red")
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")
```

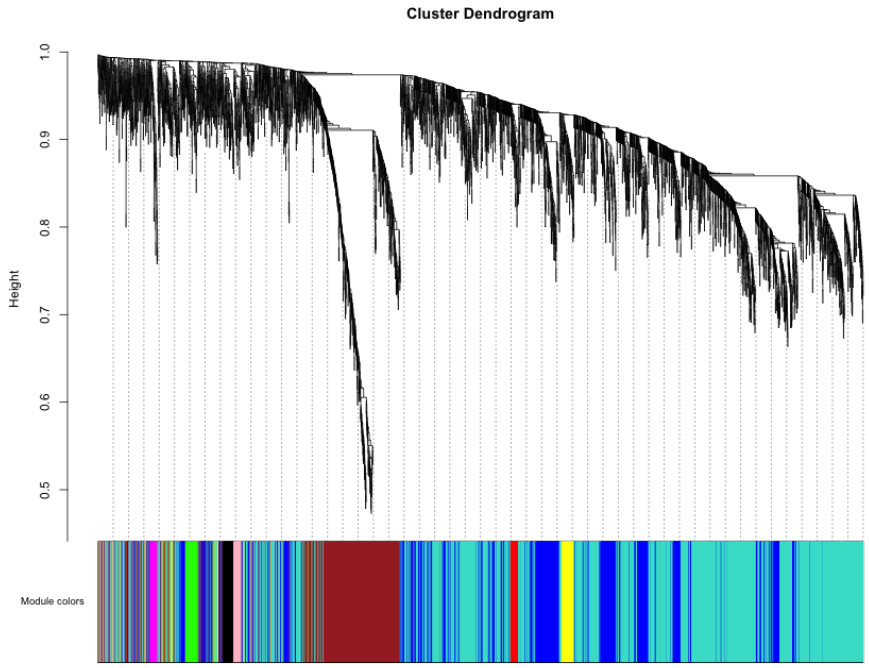
For some reason plot does not produce an image in knitr. I've included a screen shot of the image here.

I chose a soft threshold of 7

```
# Cluster into Modules
net = blockwiseModules(input, power = 7, minModuleSize = 30, reassignThreshold = 0,
    mergeCutHeight = 0.25, numericLabels = TRUE, pamRespectsDendro = FALSE,
    saveTOMs = TRUE, saveTOMFileBase = "geneTOM", verbose = 3)
```

```
##  Calculating module eigengenes block-wise from all genes
##    Flagging genes and samples with too many missing values...
##     ..step 1
## Cluster size 4454 broken into 2351 2103
## Cluster size 2351 broken into 975 1376
## Done cluster 975
## Done cluster 1376
## Done cluster 2351
## Cluster size 2103 broken into 851 1252
## Done cluster 851
## Done cluster 1252
## Done cluster 2103
##  ..Working on block 1 .
##     TOM calculation: adjacency..
##     ..will use 2 parallel threads.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication..
##     ..normalization..
##     ..done.
##    ..saving TOM for block 1 into file geneTOM-block.1.RData
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking modules for statistical meaningfulness..
```

```
##      ..removing 18 genes from module 1 because their KME is too low.
##      ..removing 14 genes from module 2 because their KME is too low.
##      ..removing 29 genes from module 3 because their KME is too low.
##      ..removing 3 genes from module 4 because their KME is too low.
##      ..removing 3 genes from module 5 because their KME is too low.
##      ..removing 7 genes from module 6 because their KME is too low.
##      ..removing 2 genes from module 8 because their KME is too low.
##      ..removing 1 genes from module 10 because their KME is too low.
##      ..removing 7 genes from module 11 because their KME is too low.
##      ..removing 3 genes from module 12 because their KME is too low.
##      ..removing 3 genes from module 14 because their KME is too low.
##      ..removing 1 genes from module 15 because their KME is too low.
##      ..removing 1 genes from module 16 because their KME is too low.
##      ..removing 3 genes from module 18 because their KME is too low.
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.25
## Cluster size 2268 broken into 1543 725
## Cluster size 1543 broken into 669 874
## Done cluster 669
## Done cluster 874
## Done cluster 1543
## Done cluster 725
##        Calculating new MEs...
## Cluster size 2268 broken into 1543 725
## Cluster size 1543 broken into 669 874
## Done cluster 669
## Done cluster 874
## Done cluster 1543
## Done cluster 725

# create dendrogram
sizeGrWindow(12, 9)
mergedColors = labels2colors(net$colors)
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
    "Module colors", dendroLabels = FALSE, hang = 0.03, addGuide = TRUE,
    guideHang = 0.05)
```




```
data$cluster = mergedColors
table(net$colors)
```

```
##
##    0    1    2    3    4    5    6    7    8    9
##  111 2268  953  663  133   85   72   65   57   47
```

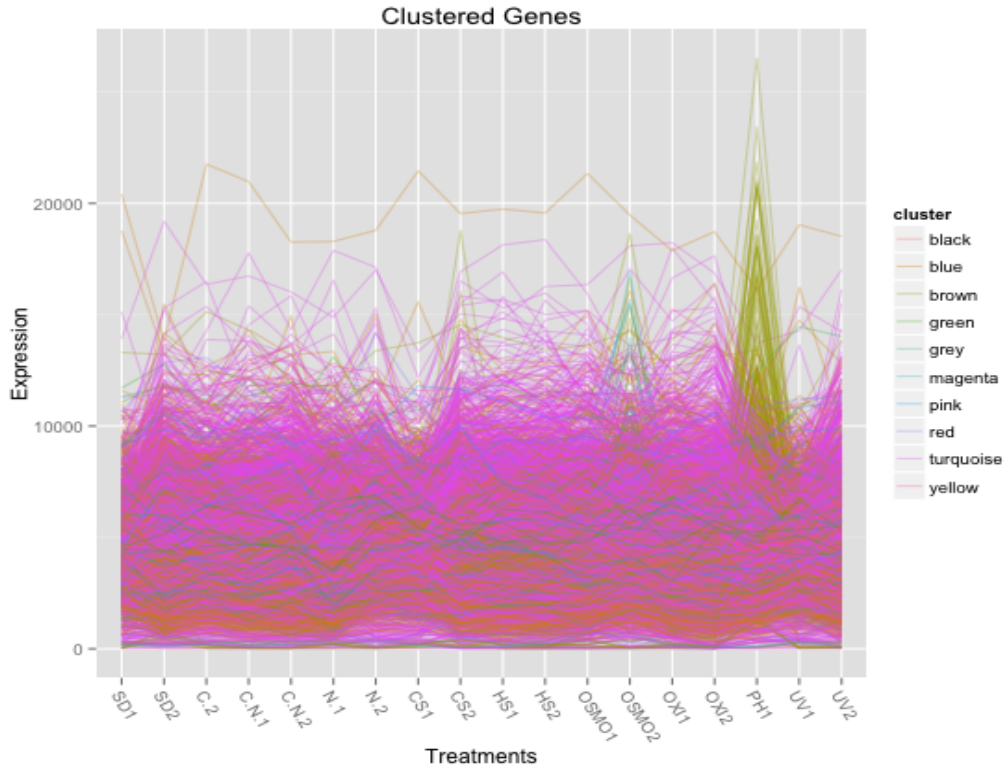
```
table(data$cluster)  # grey contains the unclustered genes
```

```
##
##      black       blue      brown      green       grey    magenta       pink
##         65        953        663         85        111         47         57
##        red turquoise     yellow
##         72       2268        133
```

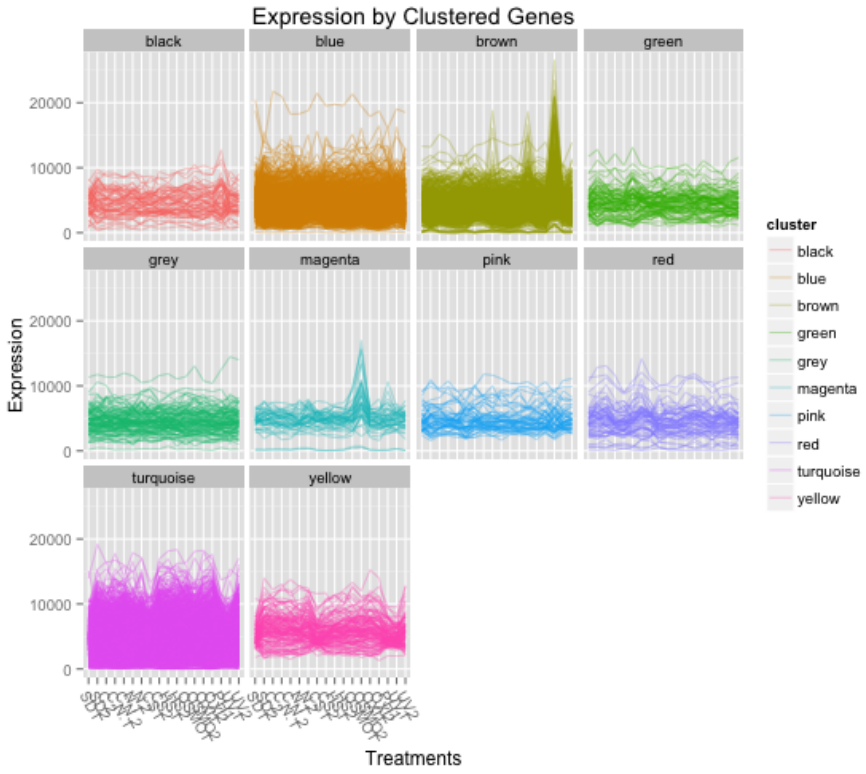
Each module is given a color name. The table above shows the number of genes clustered into each module. The cluster 0 contain genes actually contains genes that did not fit in any other cluster. So really there are 9 clusters and 125 unclustered genes

Viewing the transcription levels in each cluster

```
# can't get facets to work with ggparcoord, must go back to ggplot
ggparcoord(data, columns = c(2:19), groupColumn = 20, scale = "globalminmax",
    alphaLines = 0.3) + labs(title = "Clustered Genes", x = "Treatments",
    y = "Expression") +  theme(axis.text.x = element_text(angle = 300, hjust = 0))
```




```
data.m <- melt.data.frame(data, id.vars = c("Name", "cluster"))

ggplot(data.m, aes(x = variable, y = value, colour = cluster, group = Name)) +
    geom_line(alpha = 0.3) + facet_wrap(~cluster) +
    theme(axis.text.x = element_text(angle = 300,  hjust = 0)) +
    labs(title = "Expression by Clustered Genes", x = "Treatments",
    y = "Expression")
```



Export Co-expression Matrix as a graph

```
# Creates the adjacency matrix for the genes
adjacency = adjacency(input, power = 7)
TOM = TOMsimilarity(adjacency)  # takes some time
```

```
## ..connectivity..
## ..matrix multiplication..
## ..normalization..
## ..done.
```

```
dimnames(TOM) = list(data$Name, data$Name)
head(TOM[1:6, 1:6])
```

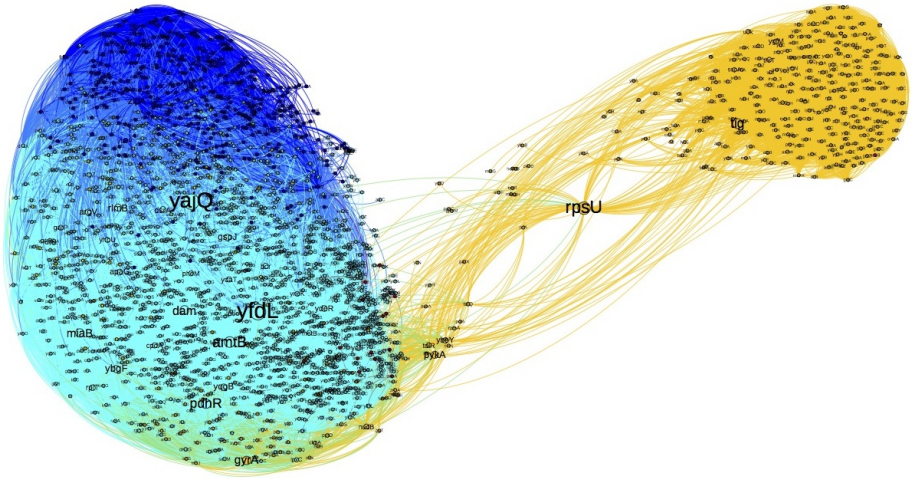
```
##            aaaD     aaaE     aaeA      aaeB      aaeR      aaeX
## aaaD 1.0000000 0.047651 0.001912 0.0007478 0.0004678 0.0024274
## aaaE 0.0476510 1.000000 0.052926 0.0030266 0.0156704 0.0113879
## aaeA 0.0019120 0.052926 1.000000 0.0011428 0.0027852 0.0021497
## aaeB 0.0007478 0.003027 0.001143 1.0000000 0.0010335 0.0003515
```

```
## aaeR 0.0004678 0.015670 0.002785 0.0010335 1.0000000 0.0018222
## aaeX 0.0024274 0.011388 0.002150 0.0003515 0.0018222 1.0000000
```

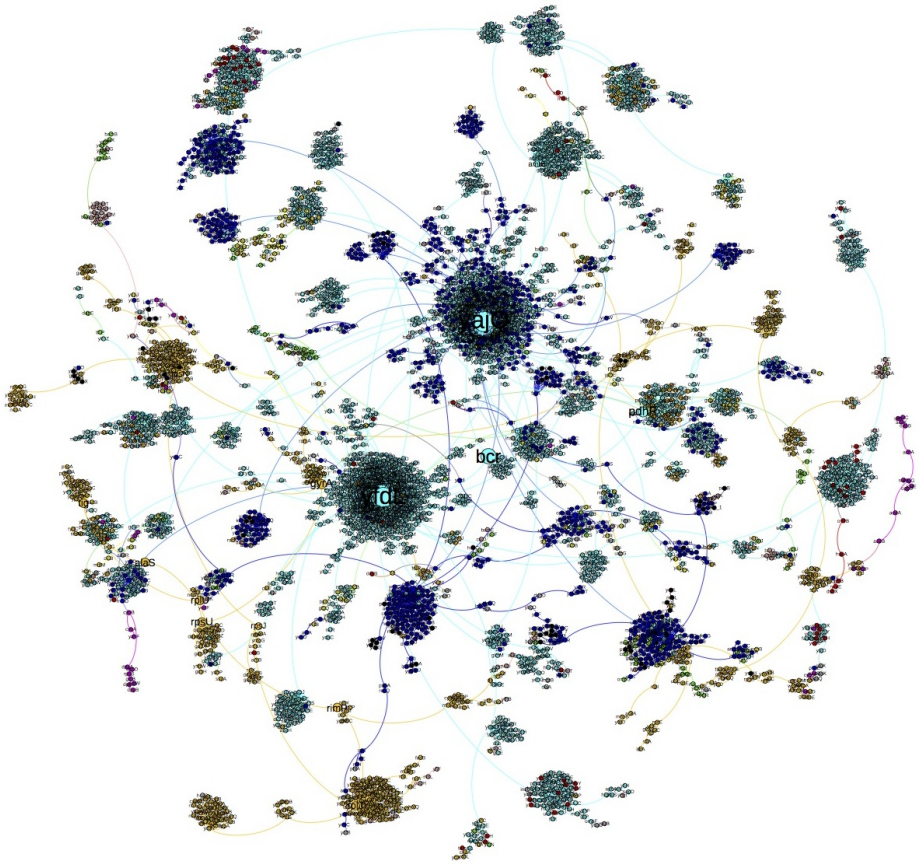
The following creates 2 files. A node file and an edge file. This can be read into an outside graph drawing software. I used gephi. (This will not evaluate during knitr, eval=FALSE)

```
module = c(unique(mergedColors))
genes = names(input)
cyt = exportNetworkToCytoscape(TOM, edgeFile = paste("CytoscapeInput-edges-",
    paste(module, collapse = "-"), ".txt", sep = ""),
    nodeFile = paste("CytoscapeInput-nodes-",
    paste(module, collapse = "-"), ".txt", sep = ""), weighted = TRUE,
    threshold = 0.02,  nodeNames = genes, nodeAttr = mergedColors)
```

There were around 4 Million interactions. I sorted the interactions by weight (co-expression value between two genes) and then arbitrarily visualized the top 50,001 interactions in gephi. Image created here. Nodes are colored by cluster. Size of node by number of degrees (connecting edges). As can be seen 3 clusters stand out. (yellow, turquoise, and blue) There is actually more. Some red, magenta, and one green gene is present. The highly connected yellow node between the yellow and blue+turquoise groups is rpsU (30S ribosomal subunit). The ribosome is responsible for genes being translated into protein. (VER IMPORTANT) The 30S subunit is the small subunit, acting like the thumb to hold the mRNA in the ribosomal active site.



Then I found the spanning graph of the interactions. (Least number of edges to include all nodes in the graph. Keep the highest weighted edge.) I'm still working on comparing the spanning graph with the biological pathways graph in KEGG.



Incorporating KEGG Biological Pathway Data

Each gene (eg. thrA) is associated with a particular KEGG gene id (eco:b0002).
In order to find the KEGG id, you can use the following website:

http://www.kegg.jp/dbget-bin/www_bfind_sub?max_hit=1000&dbkey=eco&mode=bfind&keywords=thrA

Replace the keywords=thrA with whichever gene you are searching for. I focused on E coli pathways (dbkey=eco) because that is the organism where our expression data comes from. I used a perl script to get the KEGG id, gene description and pathway ids. 356 genes did not have a corresponding KEGG id.

I was able to write an R function to retrieve KEGG ids.(originally done in perl)

```
# Function: Returns the KEGG id and gene description id and desc will be NA
# if KEGG id is not found
get_KEGG_ecoid_desc <- function(gene) {
    kegg.gene <- data.frame(name = gene, id = "NA", desc = "NA")
```

```
    # will return gene,eco_id,and description

     url <- paste("http://www.kegg.jp/dbget-bin/www_bfind_sub?max_hit=1000&dbkey=",
         "eco&mode=bfind&keywords=", gene, collapse = "", sep = "")
     doc <- htmlParse(url)
     root <- xmlRoot(doc)
     divs <- getNodeSet(root, "//div[@style]")
     t <- sapply(divs, xmlValue)

     if (length(t) < 2)
         return(kegg.gene)  # Gene does not have a KEGG id

     t <- t[grep(gene, t)][2]  # assumes it's the second entry.
     t <- unlist(strsplit(t, " "))

     if (t[2] != paste(gene, ";", sep = ""))
         return(kegg.gene)  # does not match the gene

     kegg.gene$id = t[1]
     kegg.gene$desc = paste(t[c(-1, -2)], collapse = " ", sep = " ")
     return(kegg.gene)
}
thrB.gene <- get_KEGG_ecoid_desc("thrB")  # example
thrB.gene
```

```
##   name    id
## 1 thrB b0003
##                                                                        desc
## 1 homoserine kinase (EC:2.7.1.39); K00872 homoserine kinase [EC:2.7.1.39]
```

The pathways data can be collected from the following website:

- http://www.kegg.jp/dbget-bin/www_bget?eco:b0002

Where "eco:b0002" can be replaced by whichever KEGG gene id you're looking at. Then pull out the Pathways entry in the table. The Pathways entry contains the KEGG pathway id and the Pathway name.

I also wrote an R function to retrieve the pathway ids for a particular gene (previously implemented in perl)

```
get_KEGG_ecopath_desc <- function(kegg_gene) {
    kegg.path <- data.frame(name = kegg_gene$name, path_id = "NA", path_desc = "NA")
    # will return gene,eco_id,and description
```

```
url <- paste("http://www.kegg.jp/dbget-bin/www_bget?eco:", kegg_gene$id,
    collapse = "", sep = "")
doc <- htmlParse(url)
root <- xmlRoot(doc)

tds <- getNodeSet(root, "//div")
t <- sapply(tds, xmlValue)
t <- unlist(strsplit(t, "\n"))
t <- t[grep("eco\\d", t)][1]

if (is.na(t))
    return(kegg.path)  # No pathways data

t <- gsub("eco", "|eco", t)
t <- unlist(strsplit(t, "\\|"))  # also splits where eco is in description
good <- grep("eco\\d", t)
reconnect <- c(1:length(t))
notgood <- reconnect[-good]

for (i in length(notgood):1) {
    j = notgood[i]
    if (j > 1) {
        t[j - 1] = paste(t[j - 1], t[j], collapse = "", sep = "")
    }
}

good <- t[grep("eco\\d", t)]
good <- gsub("\\s+", " ", good)
good <- sub(" ", "|", good)

pathid = c()
desc = c()
for (i in 1:length(good)) {
    temp <- unlist(strsplit(good[i], "\\|"))
    pathid <- c(pathid, temp[1])
    desc <- c(desc, temp[2])
}

# really inelegant solution which gets rid of next line title
s <- length(desc)
cpy <- desc[s]
cpy <- unlist(strsplit(gsub("[A-Z]", "|", cpy), "\\|"))
stop <- nchar(cpy[length(cpy)]) + 1
desc[s] <- substr(desc[s], 1, nchar(desc[s]) - stop)

kegg.path = data.frame(name = rep(kegg_gene$name, length(desc)), path_id = pathid,
```

```
        path_desc = desc)
    return(kegg.path)
}
get_KEGG_ecopath_desc(thrB.gene)  # example
```

```
##   name  path_id                                     path_desc
## 1 thrB eco00260     Glycine, serine and threonine metabolism
## 2 thrB eco01100                           Metabolic pathways
## 3 thrB eco01120 Microbial metabolism in diverse environments
## 4 thrB eco01230                  Biosynthesis of amino acids
```

The gene thrB is involved in 4 pathways. If a gene is not in any pathways, the function still returns one row but with the path ids and description set to NA.

Getting information for over 4000 genes takes a few hours. Must put in 1 second delays between each html page reteival or can be cut off as a denial-of-service attack. The following will not run when knit. (eval=false) The gene fetching method (get_KEGG_ecoid_desc) worked fine except for some genes whose names appeared in the gene description. Sometimes the gene name is part of another word or it was another name for the gene. I had to manually inspect to determine to keep or throw out the record. Then I would continue the for loop for the rest of the genes.

special cases: gene(i)[replaced name] = keep ; gene(i) = throw out

ccmG(319)[dsbE], dgsA(535),fcl(711), galF(905),gcP(929),matA(1520),matC(1522),rfac(2199),rfc(2218),sohA(254
spr(2569)[rspR],ycgE(3198),yeeU(3479),yeeV(3480)[cbtA],yraM(4392)

```
eco.genes <- data.frame(name = character(), id = character(), desc = character())

end <- nrow(data)
for (i in 1:end) {
    print(paste("...fetching gene", i, " of", end))
    temp_gene <- get_KEGG_ecoid_desc(data$Name[i])
    eco.genes <- rbind(eco.genes, temp_gene)
    Sys.sleep(0.5)
}

eco.paths <- data.frame(name = character(), path_id = character(), path_desc = character())

end <- nrow(eco.genes)
for (i in 1:end) {
    print(paste("...fetching paths for gene ", i, "of", end))
    temp <- get_KEGG_ecopath_desc(gene_ecoid_desc[i, ])
    eco.paths <- rbind(eco.paths, temp)
    Sys.sleep(0.5)
}
```

```
# Save KEGG data to a file
write.table(eco.genes, file = "data/gene_ecoid_desc.csv", sep = ",", row.names = FALSE)
write.table(eco.paths, file = "data/gene_pathid_pathdesc.csv", sep = ",", row.names = FALSE)
```

You can look at the retrieved kegg data here.

```
eco.genes <- read.table("data/gene_ecoid_desc.csv", sep = ",", header = TRUE,
    stringsAsFactors = FALSE)
head(eco.genes)
```

```
##   name    id
## 1 aaaD b4634
## 2 aaaE b4693
## 3 aaeA b3241
## 4 aaeB b3240
## 5 aaeR b3243
## 6 aaeX b3242
##
## 1
## 2
## 3 p-hydroxybenzoic acid efflux system component; K15548 p-hydroxybenzoic acid efflux pump
## 4 p-hydroxybenzoic acid efflux system component; K03468 p-hydroxybenzoic acid efflux pump
## 5                                                    transcriptional regulator for
## 6                                                              membrane protein of
```

```
eco.paths <- read.table("data/gene_pathid_pathdesc.csv", sep = ",", header = TRUE,
    stringsAsFactors = FALSE)
head(eco.paths, 20)
```

```
##    name  path_id                                  path_desc
## 1  aaaD     <NA>                                       <NA>
## 2  aaaE     <NA>                                       <NA>
## 3  aaeA     <NA>                                       <NA>
## 4  aaeB     <NA>                                       <NA>
## 5  aaeR     <NA>                                       <NA>
## 6  aaeX     <NA>                                       <NA>
## 7   aas eco00071                      Fatty acid degradation
## 8   aas eco00564             Glycerophospholipid metabolism
## 9   aat     <NA>                                       <NA>
## 10 abgA     <NA>                                       <NA>
## 11 abgB     <NA>                                       <NA>
## 12 abgR     <NA>                                       <NA>
## 13 abgT     <NA>                                       <NA>
```

```
## 14 abrB     <NA>                                              <NA>
## 15 accA eco00061                        Fatty acid biosynthesis
## 16 accA eco00620                            Pyruvate metabolism
## 17 accA eco00640                          Propanoate metabolism
## 18 accA eco01100                             Metabolic pathways
## 19 accA eco01110        Biosynthesis of secondary metabolites
## 20 accA eco01120 Microbial metabolism in diverse environments
```

```
nrow(subset(eco.paths, is.na(path_id)))  # genes with no KEGG pathway information
```

```
## [1] 3023
```

```
uniq_paths <- unique(na.omit(eco.paths, path_id)$path_id)  # get uniq path ids
length(uniq_paths)
```

```
## [1] 111
```

```
head(uniq_paths)
```

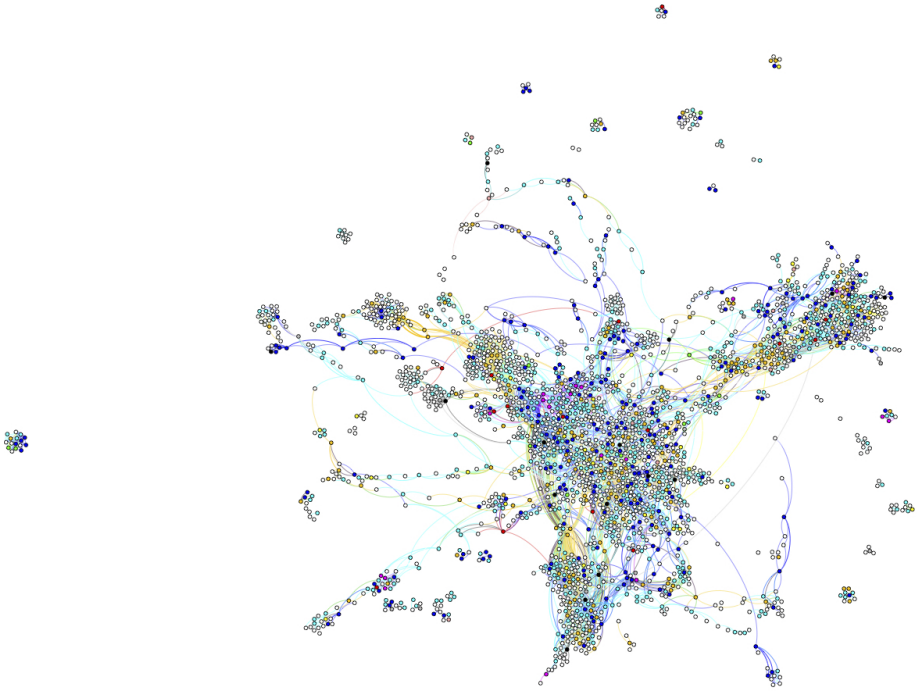
```
## [1] "eco00071" "eco00564" "eco00061" "eco00620" "eco00640" "eco01100"
```

Get the KGML files for each pathway. This will not execute when knit. (eval=false)

```
out_dir = "data/eco_path/"  # make sure this directory exists first
end <- length(uniq_paths)
for (i in 1:length(eco.path.uniq)) {
    print(paste("...fetching kgml files", i, "of", end))
    url <- paste("http://www.kegg.jp/kegg-bin/download?entry=", eco.path.uniq[i],
        "&format=kgml", collapse = "", sep = "")
    doc <- xmlTreeParse(url)
    root <- xmlRoot(doc)
    saveXML(root, paste(out_dir, eco.path.uniq[i], ".xml", collapse = "", sep = ""))
}
```

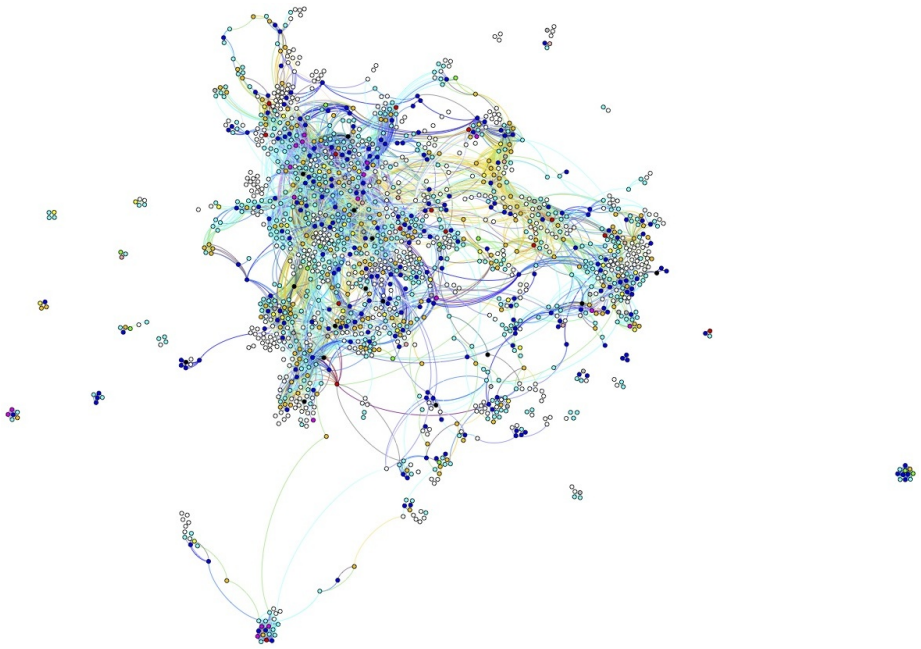
I combined each kegg pathway file into one graph. The pathways graph is different from the WGCNA generated graph because it includes compounds. (e.g. glucose, ATP, water, etc). An example of a biological pathway is Glycolysis. It would have a node for glucose (compound) which would connect to hexakinase (enzyme) which would connect to glucose-6-phosphate (compound) which would connect to phosphoglucose isomerase (enzyme) etc. Compounds and E coli genes not in the original microarray data are colored white. All the other nodes are colored by cluster. I filtered out any nodes with no edges. (Sometimes KEGG

merely groups some genes together as components of an enzyme but then doesn't connect it with any other enzymes/compounds. I'm trying to figure out why. Still exploring the data.) I was hoping that genes in the same cluster would be near each other in the pathway but that wasn't the case.



Part of the problem is the fact that many of the genes in the microarray data was not represented in the pathways data. (336 genes had no KEGG id. 2604 genes with KEGG ids had no KEGG pathways data. There were 4454 genes total. Therefore at max, there could be 1514 genes in the KEGG pathways graph.) It's possible that many of the genes-products are gene regulatory elements. They do not deal with compounds but act as transcription factors. I would need to incorporate gene regulatory networks. The majority of the biological pathways data on KEGG focuses on metabolism.

I tried removing any of the compounds (chemical intermediates) and connecting its neighbors. Therefore the network will be from enzyme to enzyme (gene to gene), instead of enzyme to compound to enzyme.



The white nodes are E coli enzymes in KEGG that were not in our microarray. The blue, turquoise, and yellow clusters have many genes and seem to be everywhere. Of the clusters that have few genes, magenta does seem to have a group of five genes that connect to one another. If you look at the other clusters with few genes; red, green, and pink are spread out with no two connecting the same color. I decided to focus on magenta to try to locate the pathway that the 5 genes participate in. The five genes are aceA, acnA, gltA, glcB, aceB.

```
eco:b4015 aceA isocitrate_lyase_(EC:4.1.3.1);_K01637_isocitrate_lyase_[EC:4.1.3.1]
eco:b1276 acnA aconitate_hydratase_1_(EC:4.2.1.3);_K01681_aconitate_hydratase_[EC:4.2.1.3]
eco:b0720 gltA citrate_synthase_(EC:2.3.3.1);_K01647_citrate_synthase_[EC:2.3.3.1]
eco:b2976 glcB malate_synthase_G_(EC:2.3.3.9);_K01638_malate_synthase_[EC:2.3.3.9]
eco:b4014 aceB malate_synthase_A_(EC:2.3.3.9);_K01638_malate_synthase_[EC:2.3.3.9]
```

All of them shared the following pathways:

- eco01200 Carbon metabolism
- eco00630 Glyoxylate and dicarboxylate metabolism
- eco01100 Metabolic pathways
- eco01120 Microbial metabolism in diverse environments (only these 5 from magenta in this pathway)

Go to following link to see the five highlighted in Red on the Microbial metabolism in diverse environments pathway. http://www.kegg.jp/kegg-bin/show_pathway?eco01120+b0720+b1276+b2976+b4014+b4015

The enzymes are just part of the cytric acid cycle.

Maybe Go in a Different Direction...

While I'm still working with the KEGG pathway graph, I may explore a different direction. Maybe do a Gene Ontology Enrichment (WGCNA contains a function) in the modules then create a shiny app where you can select each module and it will display a word cloud of GO terms for the selected module. I don't think there's a fast or convienent way to display very large graphs in R. I may be wrong but I haven't run into anything that can deal with my dataset yet.

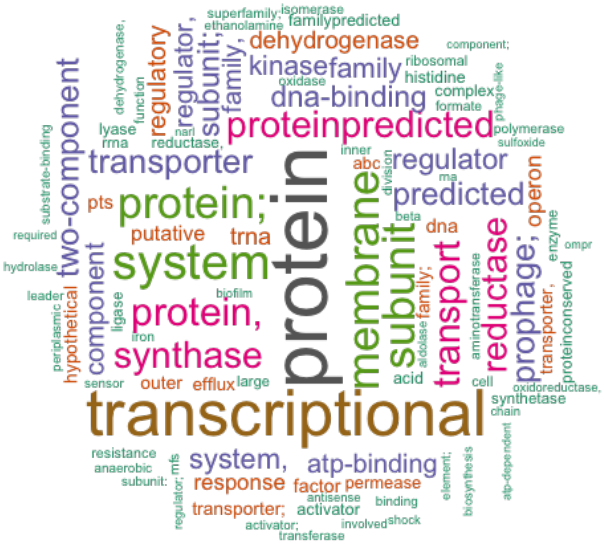
Example word cloud below.

```
# text mining example
library(tm)
library(wordcloud)
```

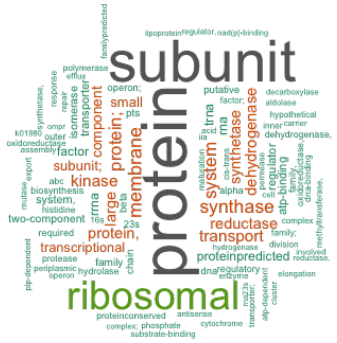
```
## Loading required package: Rcpp
## Loading required package: RColorBrewer
```

```
# prepare the data
m1 <- merge(data, eco.genes, by.x = "Name", by.y = "name")
g_c_d <- m1[, c(1, 20, 22)]
write.csv(g_c_d, file = "data/gen_cluster_des.txt", row.names = FALSE)

# function to plot wordcloud for a module
plot_mod_wordcloud <- function(mod) {
    current_mod <- subset(g_c_d, cluster == mod)
    wc.input <- paste(current_mod$desc, collapse = "")

    path_words <- Corpus(VectorSource(wc.input))
    path_words <- tm_map(path_words, stripWhitespace)
    path_words <- tm_map(path_words, tolower)
    path_words <- tm_map(path_words, removeWords, stopwords("english"))
    # path_words <- tm_map(path_words,stemDocument)

    par(mar = rep(2, 4))
    wordcloud(path_words, scale = c(5, 0.5), max.words = 100, random.order = FALSE,
        rot.per = 0.35, use.r.layout = FALSE, colors = brewer.pal(8, "Dark2"))
}

# turquoise, blue, brown, grey, green, magenta, red, pink, yellow, black
plot_mod_wordcloud("turquoise")
```

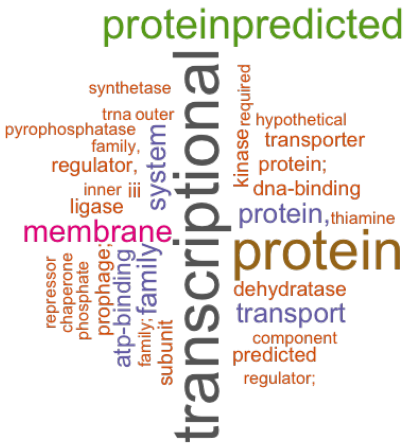

```
plot_mod_wordcloud("blue")
```



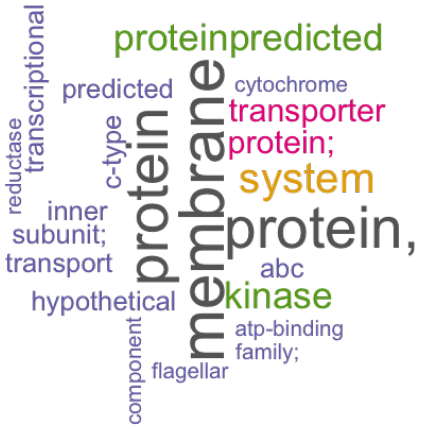
```
plot_mod_wordcloud("brown")
```



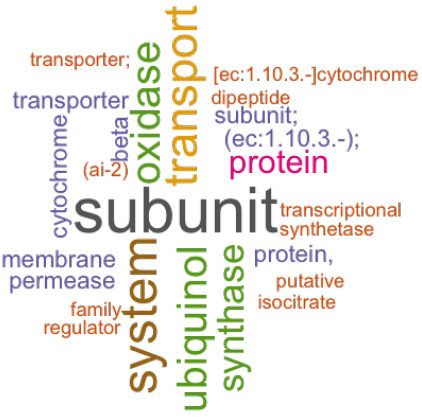

```
plot_mod_wordcloud("grey")  # this is suppose to be the unclustered genes
```




```
plot_mod_wordcloud("green")
```

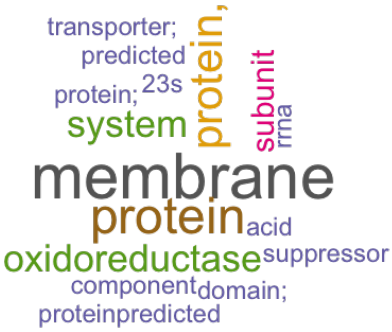



```
plot_mod_wordcloud("magenta")
```




```
plot_mod_wordcloud("red")
```


```
plot_mod_wordcloud("pink")
```




```
plot_mod_wordcloud("black")
```

Shiny App

Really basic Shiny app. User selects cluster(module) and shiny app displays word cloud. Also displays table of values below and the user can select how many records to show. This is extremely basic. The app works. It does not execute during knitr because eval=false

```
# Shiny app
library(shiny)
runApp("my_app")
```