`CoSync`: an R package for Co-Synchronization network analysis of pseudo-temporally ordered single cell data

Ye Chen, Shouguo Gao, Nathaniel Wolanyk, Xujing Wang
Systems Biology Center
National Heart, Lung, and Blood Institute
National Institutes of Health
yechent@gamil.com

Feburary 2016
Version 1.1 updated: April 2016

Contents

1 Background

The `CoSync` package is designed to construct co-synchronization networks for pseudo-temporally ordered single cell data. The metrics included in this package are: phase synchronization index, Granger causality, and coherence.

Recently we have seen a rapid blossom of single cell proteomic and transcriptomic technologies. In particular, it has be observed that cells obtained during a biological process, such as differentiation, activation and development, represent a continuum of intermediate cellular states through the process trajectory, and algorithms have been developed to pseudo-temporally ordered them along the virtual trajectory. Data of pseudo-temporally ordered single cells offer unique resources for dissection of intracellular signaling hierarchy. The sample size (i.e. the number of pseudo time points) is much larger than typical bulk sample studies; more importantly, the single-cell level measurements preserve the critical intercellular variation information that is lost in bulk assays.

Here, we assume the single cells are pseudo-temporally ordered with data properly normalized. CoSync takes the ordered data as input, and return the co-synchronization modules and GO enrichment analysis results.

2 Summary of method

CoSync takes the expression matrix with columns ordered by pseudo times as input. The data is first filtered by the variance, then the synchronization indices between each gene pair are calculated. CoSync provides three options in returning the synchronization index matrix: phase locking, Granger causality and coherence. For phase locking index, three measures are provided: Shannon's entropy rho, gamma, strobo lambda [Rosenblum, 2001], and the angle as well. With these matrices, CoSync can output the synchronization graph with user defined cutoff; the gene pairs with high synchronization indices will be connected by edges in the graph

CoSync treats the synchronization indices as the edge weights between genes. Based on the scale free topology fit, user defines the best power to transform the data. If none of the values meet the scale-free index line (which is 0.9), the power is set to 1. With the power determined, the co-synchronized genes are then clustered into the same module with the distances defined by the topological overlapping matrices. The synchronization graph and the heat map for each module are provided for visualization purposes.

To further investigate each module, CoSync provides the table with module assignment, GO annotation and GO enrichment score. If more sophisticated GO enrichment is of interest, CoSync calls the topGO functions and return results for each module with the p-value highlighted tree figure. If the data input has more than one condition, then the overlaps between modules under different conditions can be exported and visualized in a heat map.

Implementing CoSync involves the following major steps:

1. Pick the most informative n (suggested $n < 5000$ for computational demand issue) genes with highest variation to build the network.
2. Calculate the pair-wise synchronization index.
3. Pick the appropriate soft-threshold for the synchronization index matrix.

4. Define the co-synchronization modules by WGCNA.
5. Visualize highly synchronized pairs.
6. Export and plot the results.

3 Detailed implementation

The complete analysis is performed in steps, users will need to build the synchronization index matrix, and then pick the soft-threshold before you obtain the modules. In this document, we use the single cell dataset from [Trapnell, 2014] as an example to illustrate the application of CoSync. The dataset includes quantified gene expression levels in differentiating human myoblast at 0, 24, 48 and 72h after switching to low-serum medium. The single cells at proliferating path (path 1) and differentiating myoblast path (path 2) were pseudo-temporally ordered spearately. Using CoSync, the gene clusters and co-synchronization module were constructed for cells at each path.

3.1 Determine the co-synchronization index

The library depends on

Loading the package

```
library(Cosync)
library(RSEIS)
library(MSBVAR)
library(igraph)
library(WGCNA)
library(biomaRt)
library(gplots)
library(RColorBrewer)
library(topGO)
```

The advantage of considering co-synchronization index other than correlation can be depicted as the following example. Using the `phaseLocking` function to estimate the phase locking index.

```
x=sin(1:100)
y=cos(1:100)
cor(x,y)


## [1] -0.002733781


phaseLocking(x,y)


## $`entropy rho`
```

```
## [1] -0.8491037
##
## $gamma
## [1] 0.999224
##
## $`strobo lmb`
## [1] 0.9941078
##
## $`strobo ang`
## [1] -1.572809
```

```
z=t(matrix(c(x,y),ncol=2))
grangerTest(z)
```

```
## $F_statistics_matrix
##          [,1]     [,2]
## [1,]     0.00 67615.53
## [2,] 67615.53     0.00
##
## $p_value_matrix
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

```
coherenceTest(z)
```

```
## [1] 0.9999764
```

As we can see, the correlation is low while the co-synchronization indices are high. Let's use the pseudo-temporally ordered RNA-seq data from the library monocle as a display example. Load the monocle pseudo-temporally ordered expression data from the data folder of the package. The file contains the necessary data of the HSMM dataset from monocle, and the precomputed phase locking index matrices.

```
load("./data/expr_sync.RData")
```

For challenges in the computational time and the detection of true signal, we suggest to filter genes at the first step. By monocle, the single cell samples are fitted to 3 paths, wherein one path is considered to be the noise. Here, we consider the path 1 and 2. We will build up the synchronization modules for each path, and find the intersection of them.

```
n_genes = 2000
sd_all = apply(expr_all,1,function(x) sd(x))
expr1 = as.matrix(expr_all[names(sort(sd_all,decreasing = T)[1:n_genes]),
                           rownames(subset(HSMM_pheno,State=="1"))])
rownames(expr1)=HSMM_feature[rownames(expr1),1]
expr2 = as.matrix(expr_all[names(sort(sd_all,decreasing = T)[1:n_genes]),
```

```
                                rownames(subset(HSMM_pheno,State=="2"))])
rownames(expr2)=HSMM_feature[rownames(expr2),1]
```

Using phase locking index as an example, we calculate the phase locking index matrices as the following. This step may be time consumingg. (Note that the functions phaseLockingMatrix and coherenceTest provide the progression bar, but grangerTest does not. Approximate run time is, 4 hr phase locking, 4 hr Granger, 2 hr coherence, with 2000 genes.)

```
#Here we skip this step and they are precomputed and saved in expr_sync.RData
#phase1=phaseLockingMatrix(expr1)
#phase2=phaseLockingMatrix(expr2)
```

Throughout this vignette, we use phase locking index gamma for the synchronization index [Rosenblum, 2001].

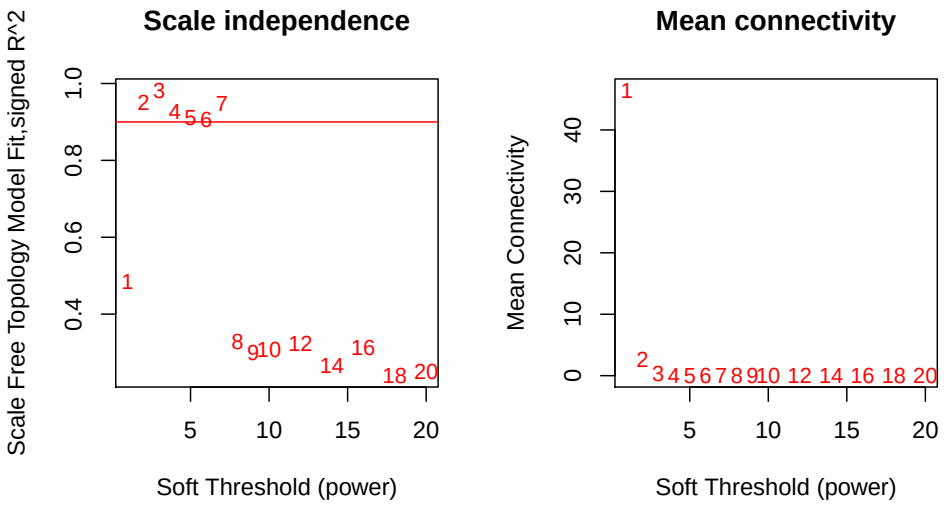
3.2 Buidling modules

We assume that the whole graph contains the genes as vertices, and synchronization index as the edge weight between vertices. To make the weighted graph closer to be a scale free graph, we raised different powers from 1 to 20 to the weighted adjacency matrix and use WGCNA and to find a reasonable power. This power is called soft-threshold.

```
power1=wgcnaPower(phase1$gamma)
```

Check the result in the plot. The smallest power that passes 0.9 should be picked as the soft-threshold. If none, then use 1 as the soft-threshold for next step.

```
par(mfrow = c(1,2));
  cex1 = 0.9;
  # Scale-free topology fit index as a function of the soft-thresholding power
  plot(power1$fitIndices[,1],-sign(power1$fitIndices[,3]) * power1$fitIndices[,2],
    xlab = "Soft Threshold (power)",ylab = "Scale Free Topology Model Fit,signed R^2",type =
      "n",main = paste("Scale independence"));
  text(power1$fitIndices[,1],-sign(power1$fitIndices[,3]) * power1$fitIndices[,2],
    labels = c(c(1:10), seq(from = 12, to=20, by=2)),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(power1$fitIndices[,1], power1$fitIndices[,5],
    xlab = "Soft Threshold (power)",ylab = "Mean Connectivity", type =
      "n",main = paste("Mean connectivity"))
  text(power1$fitIndices[,1], power1$fitIndices[,5], labels = c(c(1:10), seq(from = 12,
  to=20, by=2)), cex = cex1,col ="red")
```

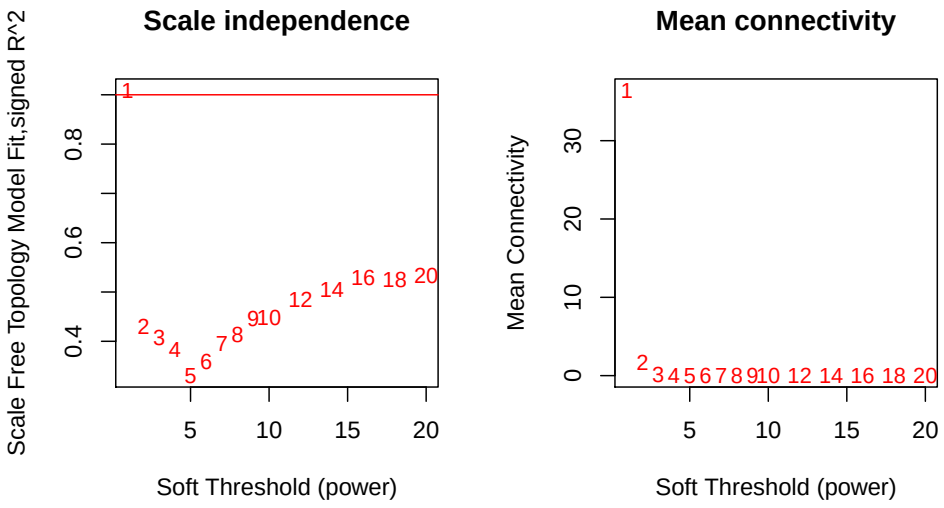


And we do the same for the synchronization matrix of path 2.

```
power2=wgcnaPower(phase2$gamma)
```

Check the result in the plot.

```
par(mfrow = c(1,2));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding power
plot(power2$fitIndices[,1],-sign(power2$fitIndices[,3]) * power2$fitIndices[,2],
xlab = "Soft Threshold (power)",ylab = "Scale Free Topology Model Fit,signed R^2",type =
"n",main = paste("Scale independence"));
text(power2$fitIndices[,1],-sign(power2$fitIndices[,3]) * power2$fitIndices[,2],
labels = c(c(1:10), seq(from = 12, to=20, by=2)),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(power2$fitIndices[,1], power2$fitIndices[,5],
xlab = "Soft Threshold (power)",ylab = "Mean Connectivity", type =
"n",main = paste("Mean connectivity"))
text(power2$fitIndices[,1], power2$fitIndices[,5], labels = c(c(1:10), seq(from = 12,
to=20, by=2)), cex = cex1,col ="red")
```



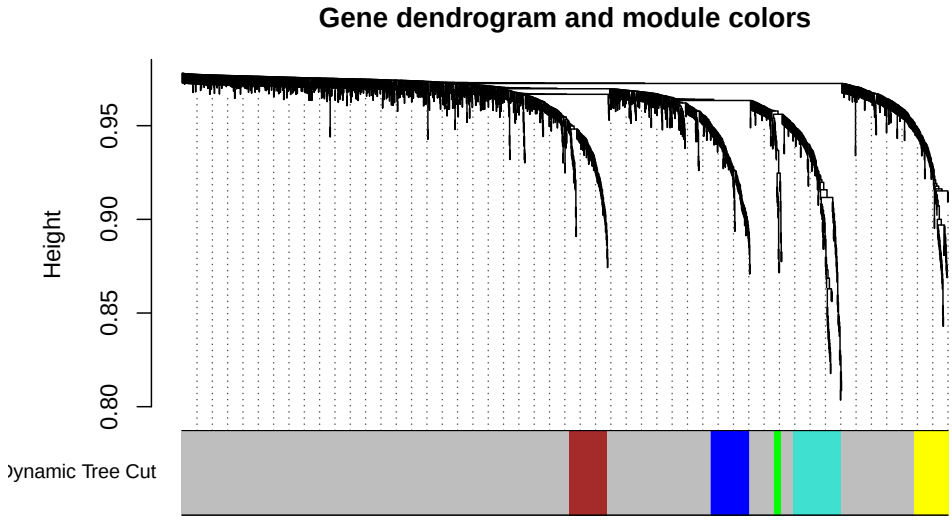
Base on the figures, we pick 1 as the soft-threshold power for the synchronization matrices, which is the original phase index. Although the discussion on different sychronization index matrices is still needed, to carry out the analysis, if none of the power enables distribution to be scale free, we suggest to keep the original matrix, which is the soft-threshold of 1. We compile WGCNA functions into the one which returns the modules and the dendrogram. We set the default minimum module size to 10.

```
modules1=wgcnaAnalysisStaticCut(x=phase1$gamma,softPower=1,cutHeight=0.95)


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

Check the tree and the modules in the plot.

```
WGCNA::plotDendroAndColors(
    modules1$geneTree, labels2colors(modules1$modules), "Dynamic Tree Cut", dendroLabels = FALSE, hang =
  )
```



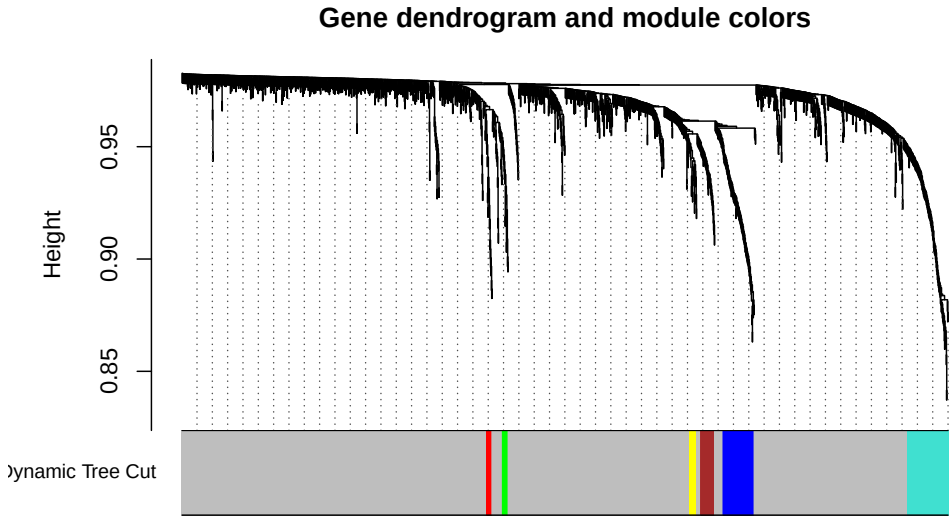
We do the same for the path 2 samples.

```
modules2=wgcnaAnalysisStaticCut(phase2$gamma,softPower=1,cutHeight=0.95)


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

Check the tree and the modules in the plot.

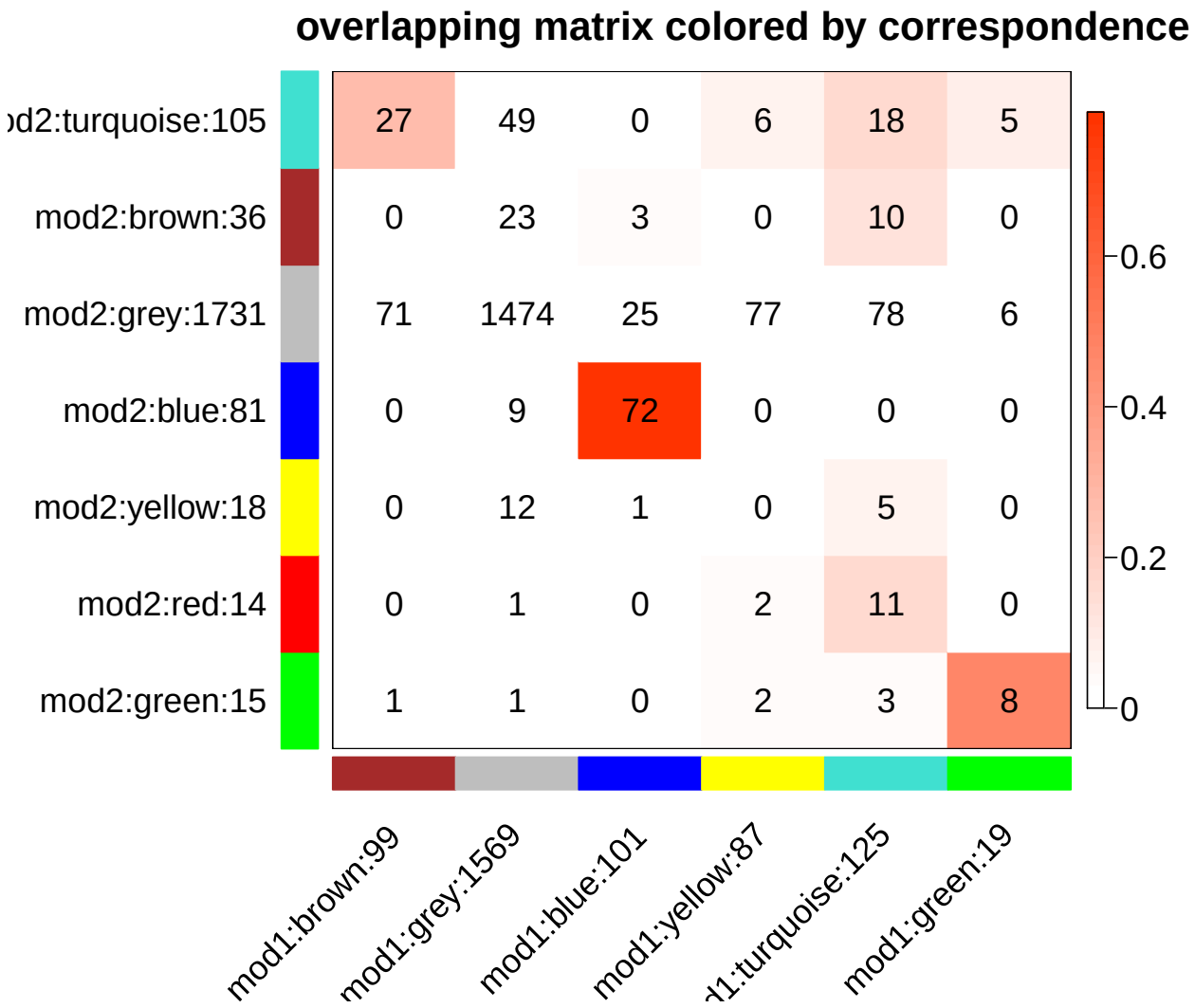
```
WGCNA::plotDendroAndColors(
    modules2$geneTree, labels2colors(modules2$modules), "Dynamic Tree Cut", dendroLabels = FALSE, hang =
  )
```



3.3 Visualize highly synchronized genes

For the modules in two different paths, we calculate the number of genes in the intersections and the concordance = (# of genes in the intersection/(# of genes in a module in path1 + # of genes in a module in path2). We also plot out the labeled heatmap. The number in each rectangle is the number of genes in the intersections between two corresponding modules. The color is scaled by the correspondence (in the range [0,1]). Number of genes in each module is given after the color name.

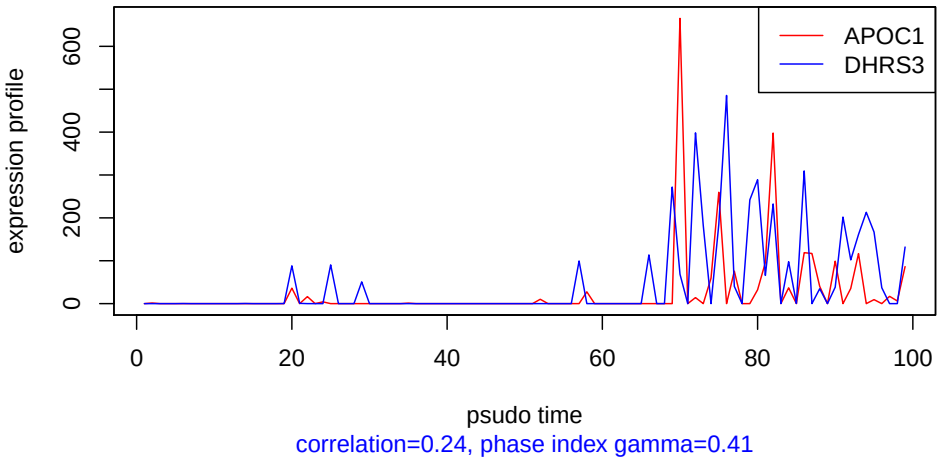
```
overlap_matrix=wgcnaOverlap(modules1 = modules1$modules,modules2 = modules2$modules)
```



Based on the concordance, module blue is highly preserved between the two different paths.

Here we plot a gene pair in path 1 that has high phase locking index but low correlation.

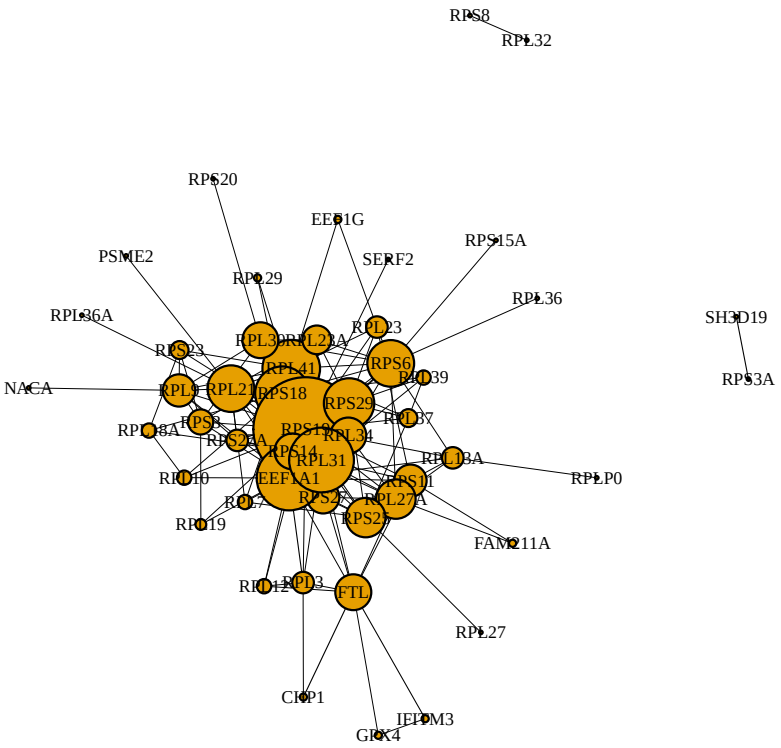
```
s1 = expr1["APOC1",]
s2 = expr1["DHRS3",]
plot(1:length(s1),s1,ylim = range(min(s1,s2),max(s1,s2)),type = "l",col = "red", main =
"module turquoise",xlab = "psudo time",ylab = "expression profile",sub = paste(
"correlation=",toString(round(cor(s1,s2),2)),", phase index gamma=", toString(round(
phaseLocking(s1,s2)$gamma,2)), sep ="",col.sub = "blue")
par(new = TRUE)
plot(1:length(s1),s2,ylim = range(min(s1,s2),max(s1,s2)),type = "l",col = "blue",
xaxt = "n", yaxt = "n", ann = FALSE)
legend("topright",lty = 1,c("APOC1","DHRS3"), col = c("red","blue"))
```



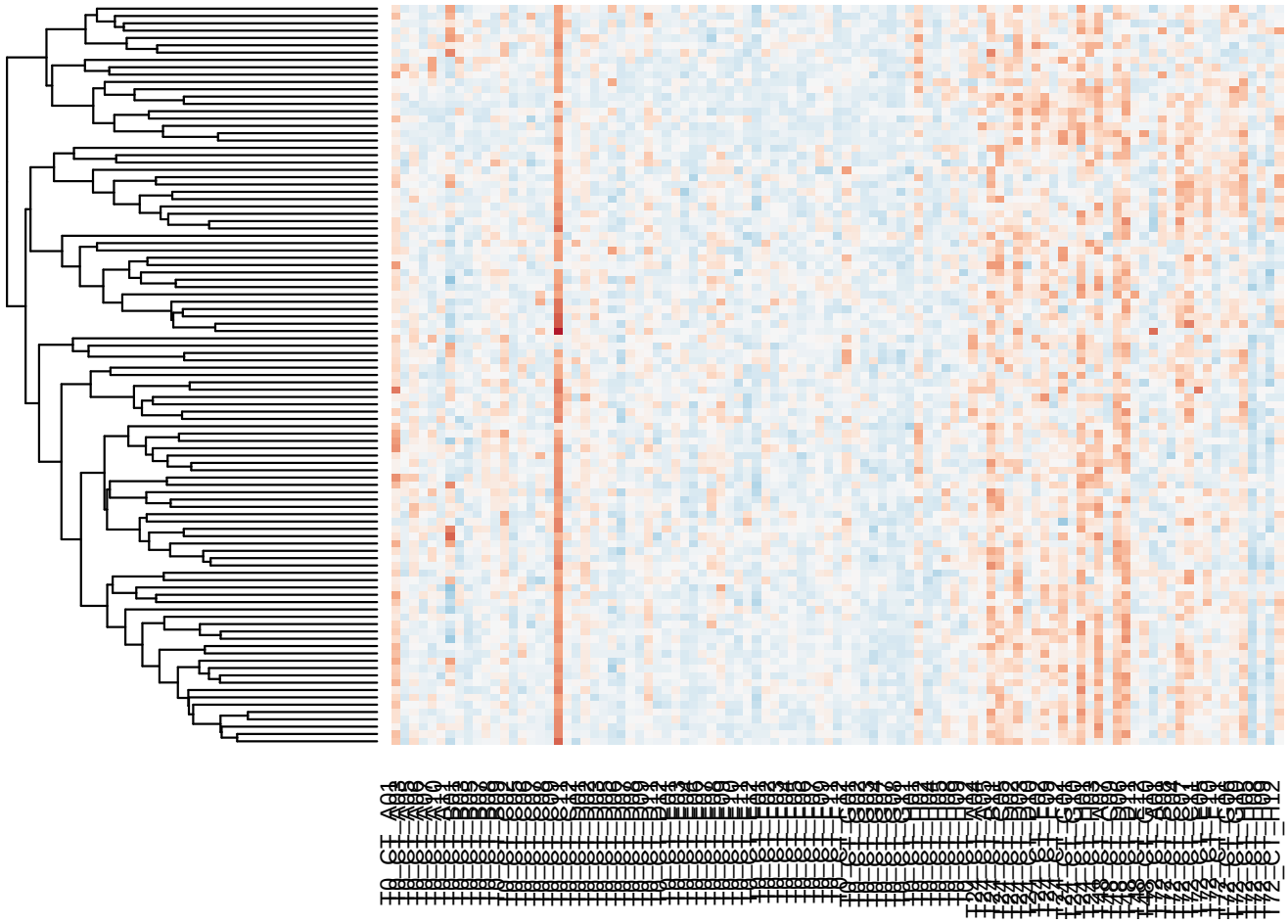
To examine genes and their expression in each module, the ModulePlots function of CoSync can plot out the graph for the indicated module and row-centered heatmap of each module. The edges are shown in the graph if their weights are over the threshold. Vertex size is proportional to its vertex degree.

```
subgraph1=ModulePlots(x=phase1$gamma,expr=expr1,modules=modules1$modules,thisColor = "blue",cut = 0.3)
```

Module blue


RPS8
RPL32
RPS20
EEF1G
RPS15A
PSME2
SERF2
RPS29
RPL36
RPL36A
RPL23
SH3D19
RPS23
RPL30
RPL24A
RPL41
RPS6
RPL39
RPS3A
NACA
RPL9
RPL21
RPS18
RPS29
RPS8
RPL37
RPL18A
RPS19
RPS28A
RPL34
RPS14
RPL31
RPL13A
RPL10
EEF1A1
RPS11
RPLP0
RPL7
RPS25
RPL27A
RPS27
RPL19
FAM211A
RPL12
RPL3
FTL
RPL27
CIAPIN1
IFITM3
GPX4

Module blue



The list of genes shown in the graph with the corresponding degree is also returned by this function.

```
head(subgraph1$subgraph_matrix)
```

```
##      subgraph ID of module blue gene ID degree
## [1,] "1"                        "FTL"   "10"
## [2,] "1"                        "RPL41" "16"
## [3,] "1"                        "RPS27" "9"
```

```
## [4,] "1"                          "RPS18" "15"
## [5,] "1"                          "RPL21" "13"
## [6,] "1"                          "RPS19" "29"
```

3.4 GO enrichment analysis

For all modules, CoSync compile the biomaRt id conversion function and WGCNA GO enrichment analysis function to investigate the dominant ontology. We also provide the GO annotation for each gene using biomaRt as well. These functions are compiled into `ModuleAnnotation` which returns a table.

```
path1_module=ModuleAnnotation(modules = modules1$modules)
```

Check the result.

```
path1_module[1:5,1:8]
```

```
##            module modSize bkgrModSize rank  enrichmentP   BonferoniP
## GO:0000184   blue      99          98    1 2.000137e-83 3.179218e-79
## GO:0022626   blue      99          98    2 9.829216e-83 1.562354e-78
## GO:0006614   blue      99          98    3 2.729063e-77 4.337846e-73
## GO:0019083   blue      99          98    4 2.729063e-77 4.337846e-73
## GO:0006613   blue      99          98    5 2.808820e-76 4.464619e-72
##            nModGenesInTerm fracOfBkgrModSize
## GO:0000184              68         0.6938776
## GO:0022626              68         0.6938776
## GO:0006614              69         0.7040816
## GO:0019083              69         0.7040816
## GO:0006613              69         0.7040816
```

So for the path 2 modules.

```
path2_module=ModuleAnnotation(modules = modules2$modules)
```

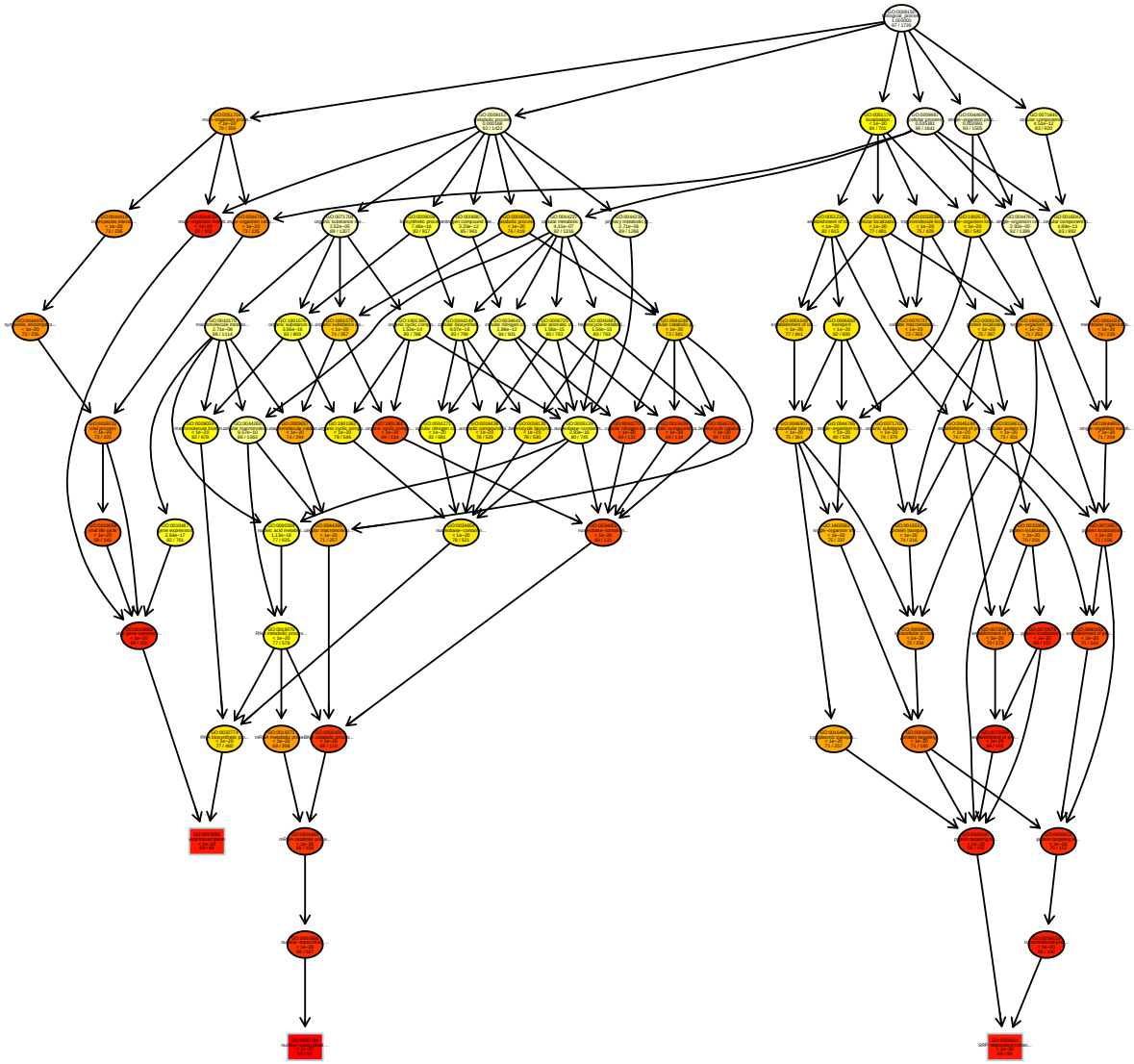
CoSync also provides the option to call topGO package to perform GO enrichment analysis on each individual module.

```
topgo1=topGOanalysis(modules1$modules, selectColor="blue")
```

```
##
## Building most specific GOs ..... ( 4295 GO terms found. )
##
## Build GO DAG topology .......... ( 7825 GO terms and 18437 relations. )
##
```

```
## Annotating nodes ..... ( 1729 genes annotated to the GO terms. )
##
##   -- Classic Algorithm --
##
##   the algorithm is scoring 1757 nontrivial nodes
##   parameters:
##   test statistic:  fisher
## TRUE_classic_3_all  --- no of nodes:  87


head(topgo1)
```



```
##       GOterm
## [1,] "GO:0000184"
## [2,] "GO:0006614"
## [3,] "GO:0019083"
## [4,] "GO:0006613"
## [5,] "GO:0019080"
## [6,] "GO:0044033"
##       GODesc
## [1,] "nuclear-transcribed mRNA catabolic process, nonsense-mediated decay"
## [2,] "SRP-dependent cotranslational protein targeting to membrane"
## [3,] "viral transcription"
```

```
## [4,] "cotranslational protein targeting to membrane"
## [5,] "viral gene expression"
## [6,] "multi-organism metabolic process"
##      pValue
## [1,] "1.35050270171158e-82"
## [2,] "1.86183062525915e-76"
## [3,] "1.86183062525915e-76"
## [4,] "1.91544252024205e-75"
## [5,] "1.80586582198444e-74"
## [6,] "1.80586582198444e-74"
```

By GO enrichment analysis, the genes in module blue of path 1 are closely related to catabolic process.

4 Conclusions

It is hoped that this package will facilitate analysis of psudo-temporally odered single cell expression data. It can distill co-synchronization information into graph clusters that outline the underlying dynamical structure.

5 Acknowledgments

This work was partially supported by the Intramural Research Program of the NIH, NHLBI. We thank Dr. Hu, Gangqing for insightful discussions and for allowing us to test CoSync on his T-cell differentiation time series data.

6 References

Rosenblum, M., A. Pikovsky, J. Kurths, C. Schafer and P.A. Tass, Phase syncrhonization: from theory to data analysis, in Handbook of Biological Physics, F. Moss and S. Gielen, Editors. 2001, Elsevier Science: Amsterdam. p. 279-321.

Gao, S., J. Hartman, J.L. Carter, M.J. Hessner and X. Wang, Global analysis of phase locking in gene expression during cell cycle: the potential in network modeling. BMC Syst Biol, (2010) 4: p. 167.

Thurman W.N. & Fisher M.E., Chickens, Eggs, and Causality, or Which Came First?, American Journal of Agricultural Economics, (1998) 237-238.

Langfelder, P. & Horvath, S. WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics, (2008) 9, 559

Trapnell, C. et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat. Biotechnol., (2014) 32, 381386.