Chronic Fatigue Syndrome (CFS)
Gene Co-expression Network Analysis
R Tutorial

Angela Presson,Steve Horvath,

Correspondence: shorvath@mednet.ucla.edu, http://www.ph.ucla.edu/biostat/people/horvath.htm

This R tutorial describes how to carry out a gene co-expression network analysis using soft thresholding. The network construction is conceptually straightforward: nodes represent genes and nodes are connected if the corresponding genes are significantly co-expressed across appropriately chosen tissue samples. Here we study networks that can be specified with the following adjacency matrix: $A=[a_{ij}]$ is symmetric with entries in [0,1]. By convention, the diagonal elements are assumed to be zero. For unweighted networks, the adjacency matrix contains binary information (connected=1, unconnected=0). In weighted networks the adjacency matrix contains weights.

To cite this tutorial or the statistical methods please use

1. *Zhang B, Horvath S (2005) A General Framework for Weighted Gene Co-Expression Network Analysis. Statistical Applications in Genetics and Molecular Biology. In Press.Technical Report and software code at: www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork*.

For the generalized topological overlap matrix as applied to *unweighted* networks see

2. *Yip A, Horvath S (2005) Generalized Topological Overlap Matrix and its Applications in Gene Co-expression Networks. http://www.genetics.ucla.edu/labs/horvath/GTOM/*.

For some additional theoretical insights consider

3. *Horvath S, Dong J, Yip A (2005) Using the Factorizability Decomposition to Understand Connectivity in Modular Gene Co-Expression Networks. Technical Report and software code at http://www.genetics.ucla.edu/labs/horvath/ModuleConformity/*

Data

Subjects

We analyzed the phenotype, genotype and expression data of 164 samples from a four year longitudinal study conducted by the Centers for Disease Control (CDC, http://www.cdc.gov/). The phenotype data included several variables that measured different aspects of chronic fatigue syndrome. "Intake diagnosis" was a classification of CFS based on the 1994 case definition criteria (Fukuda et al. 1994). In addition to intake diagnosis, the data set included scores from established diagnostic procedures used to evaluate quality of life in people suffering from cancer and other illnesses: 1) Medical Outcomes Survey Short Form (SF-36), 2) Multidimensional Fatigue Inventory (MFI), and 3) CDC Symptom Inventory Case Definition scales (Kaasa et al. 1998; Reeves et al. 2005). The SF-36 scale assesses eight characteristics: physical function, role physical, bodily pain, general health, vitality, social function, role emotional, and mental health. The MFI scale assesses five characteristics: general fatigue, physical fatigue, mental fatigue, reduced motivation, and reduced activity. Each of these 14 characteristics is derived from several scores designed to evaluate the particular characteristic. Reeves et al. (2005) clustered these 14 scores from 118 patients and identified three clusters of chronic fatigue severity: high, moderate and low (where low severity patients were equivalent to the control population). The CDC symptom inventory scale assesses symptoms accompanying chronic fatigue.

Gene Expression Microarray Data

Peripheral blood mononuclear cells were assayed with approximately 20,000 probes from glass slide arrays (MWG Biotech). ArrayVision software read the slides and normalized the data by subtracting background intensity from the spot intensity values. When background intensity exceeded spot intensity, ArrayVision set the probe intensity values to zero.

Test and Second data set Subjects

A major difficulty in studying complex diseases is that cases can present multiple symptoms. To reduce genetic heterogeneity, we considered the 127 samples classified as ill according to the intake diagnosis. This group included some patients with a medical condition such as depression that is considered exclusionary in some CFS studies (Reeves et al. 2003). These subjects had the following CFS severity distribution: high (24), moderate (48), and low (15); and the majority were female (98).

Among these samples, we further restricted our analysis to the 87 samples that had severity scores (64 females and 23 males). We refer to these samples as the "test data set", and the remaining samples were considered for validating our analysis based on their empiric diagnosis. The empiric score was highly correlated with the severity trait (0.782, p-value = 2.2 x 10^{-16}), as it was based on half of the 14 scales used to define severity. Since the majority of the second data set samples were female, we avoided sex confounding by analyzing only the female subjects in the second data set. Furthermore, we discarded two samples in the second data set that had outlying mean gene expression values. The resulting second data data set consisted of empiric, gene expression and SNP data for 31 female samples.

Genetic Marker Data

We considered 36 autosomal SNPs that the CDC had previously selected from eight candidate CFS genes, *TPH2* (SNPs selected from locus 12q21), *POMC* (2p24), *NR3C1* (5q34), *CRHR2* (7p15), *TH* (11p15), *SLC6A4* (17q11.1), *CRHR1* (17q21), *COMT* (22q11.1) (Smith et al. 2006). We additively coded the SNPs as 0, 1, or 2, for genotypes AA, AB, and BB, respectively. While this additive coding method may be sub-optimal for dominant or recessively acting loci, it has been shown to be effective for many genetic models (Horvath et al. 2004).

Prediction of CFS genes

It has been reported independently by many research groups that genes with high connectivity are more likely to be involved in complex diseases. We measure the importance of CFS genes using "gene significance". Abstractly speaking, gene significance is any quantitative measure that specifies how biologically significant a gene is. One goal of network analysis is to relate the measure of gene significance (here chronic fatigue severity) to intramodular connectivity and genetic marker data.

Methods Outline

The network construction is conceptually straightforward: nodes represent genes and nodes are connected if the corresponding genes are significantly co-expressed across appropriately chosen tissue samples. Here we study networks that can be specified with the following adjacency matrix: $A=[a_{ij}]$ is symmetric with entries in [0,1]. By convention, the diagonal elements are assumed to be zero. For unweighted networks, the adjacency matrix contains binary information (connected=1, unconnected=0). In weighted networks the adjacency matrix contains weights.

The absolute value of the Pearson correlation between expression profiles of all pairs of genes was determined for 2677 transcripts. Then, pioneering the use of a novel approach to the generation of a weighted gene coexpression networks, the Pearson correlation measure was transformed into a connection strength measure by using a power function (connection $strength(i,j)=|correlation(i,j)|^{\beta})$ (Zhang and Horvath 2005). The connectivity measure for each gene is the sum of the connection strengths ($\text{correlation}^{\beta}$) between that gene and all the other genes in the network. Gene expression networks, like virtually all types of biological networks, exhibit an approximate scale free topology. A linear regression model fitting index R^2 between log p(k) and log(k) was used to determine how well a resulting network fit scale free topology for a range of values. The scale free topology criterion (Zhang and Horvath 2005) was used to determine the power. We will discuss the choice of this power in great detail below and provide several arguments that this choice of power results in a biologically meaningful network.

Weighted Network Construction

The absolute pair-wise Pearson correlation is computed between expression profiles of each pair of genes. We then calculate the adjacency matrix A, where $a_{ij} = cor(i,j)^{\beta}$ where β is chosen to create a scale-free network.

Module Construction

To group genes with coherent expression profiles into modules, we use average linkage hierarchical clustering, which uses the topological overlap dissimilarity measure.
The topological overlap of two nodes reflects their similarity in terms of the commonality of the nodes they connect to, see (Ravasz et al 2002; Yip and Horvath 2007).
Once a dendrogram is obtained from a hierarchical clustering method, we need to choose a height cutoff in order to arrive at a clustering. It is a judgement call where to cut the tree branches.
The height cut-off can be found by inspection: a height cutoff value is chosen in the dendrogram such that modules are distinct, which can be evaluated by a multi-dimensional scaling plot.

Detection of hub genes

To identify hub genes for the network, one may either consider the whole network connectivity (denoted by kTotal) or the intramodular connectivity (kWithin).
We find that intramodular connectivity is far more meaningful than whole network connectivity

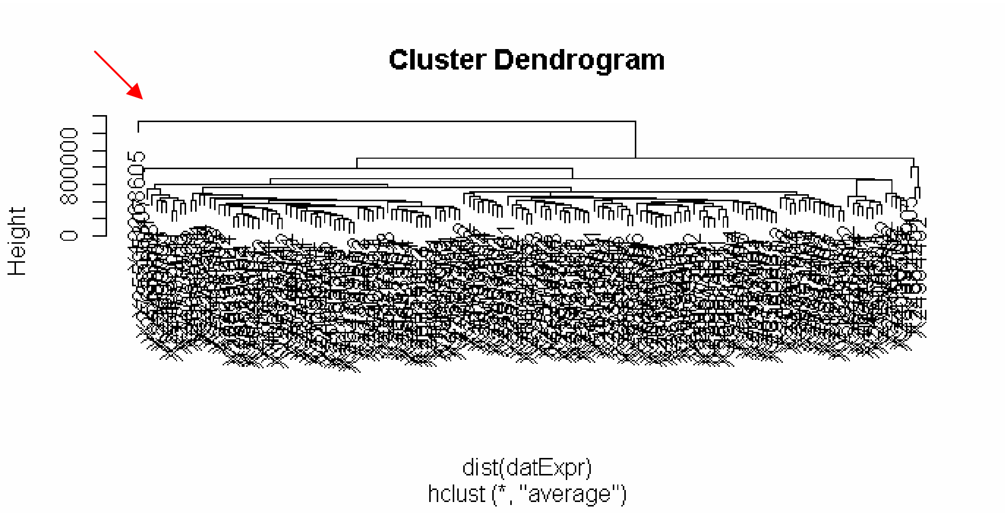
Relating hub status to gene (prognostic) significance

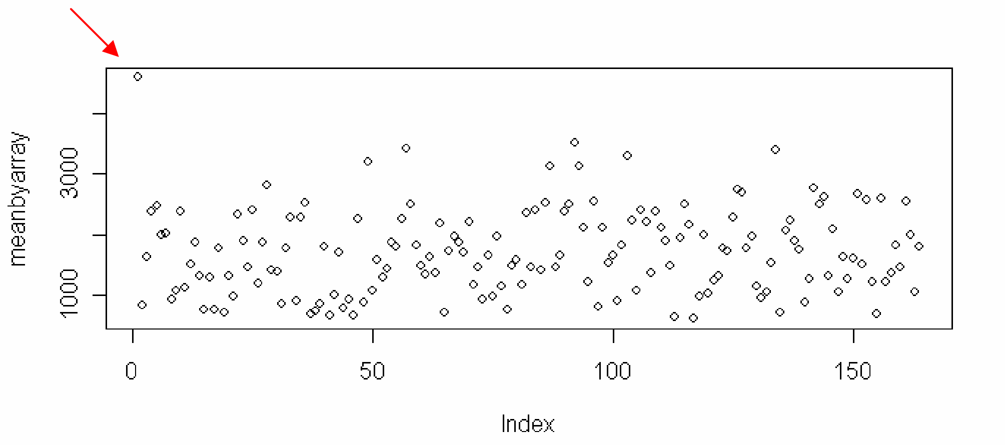
Abstractly speaking, gene significance (GS) is any quantitative measure that specifies biological significance. Typically GS is just the absolute value of the Pearson correlation between a trait T and a gene expression profile x_i , $GS_i=|cor(T, x_i)|$) One goal of network analysis is to relate gene significance to intramodular connectivity k_i.

Since the module identification is computationally intensive, only 2677 genes were considered in the following analysis. R software code, a tutorial, and a technical report for generating weighted gene co-expression networks can be found at the following webpage: www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork.

```
# Absolutely no warranty on the code. Please contact SH with suggestions.

# CONTENTS
# This document contains functions for carrying out the following tasks:
# A) Assessing scale free topology and choosing the parameters of the adjacency function using the
#      scale free topology criterion (Zhang and Horvath 05)
# B) Computing the topological overlap matrix
# C) Defining gene modules using clustering procedures
# D) Summing up modules using their first principal components (first eigengene)
# E) Relating a measure of gene significance to the modules
# F) Carrying out a within module analysis (computing intramodular connectivity)
#    and relating intramodular connectivity to gene significance.

# Download the R software at http://www.R-project.org.

# After installing R, you need to install several additional R library packages:
# For example to install Hmisc, open R,
# go to menu "Packages\Install package(s) from CRAN",
# then choose Hmisc. R will automatically install the package.
# When asked "Delete downloaded files (y/N)? ", answer "y".
# Do the same for some of the other libraries mentioned below. But note that
# several libraries are already present in the software so there is no need to re-install them.

# This tutorial and its referenced data files are provided on the following webpage
# www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/ChronicFatigue .
# Download the data file "CFS.full.data.zip" (9.75 MB) containing 3 separate data files:
# "Expression_data_CFS.txt", "SNP_data_CFS.txt", "Clinical_data_CFS.txt" and two R function
# files required for the network analysis: "RFunctions.txt" and "NetworkFunctions.txt".


# Unzip all the files into the same directory, for example
#"C:\Documents and Settings\apresson\My Documents\CFStutorial"
```

```
# Set the working directory of the R session by using the following command:
# setwd("Directory Path").  For example,
# setwd("C:/Documents and Settings/apresson/My Documents/CFStutorial")
# Note that R requires / instead of \ in the path name.
# The user should copy and paste the following script into the R session.
# Text after "#" is a comment and is automatically ignored by R.

#Comments and code shaded green indicate updates based on user comments 1/26/10.
setwd("C:/Documents and Settings/apresson/My Documents/CFStutorial")

library(MASS)     # standard, no need to install
library(class)    # standard, no need to install
library(cluster)
library(impute)   # install it for imputing missing value
source("sma_package.txt") # Code from the no longer supported sma package.

#Memory
# check the maximum memory that can be allocated
memory.size(TRUE)/1024
# increase the available memory
memory.limit(size=4000)

#Load network functions
source("NetworkFunctions_Jan2010.txt");
source("neo.txt");source("CausalityFunctions.txt")

#Read in the 3 full data files:
datSNP=read.delim("SNP_data_CFS.txt",header=TRUE,sep="\t",row.names=1,as.is=T)
datClinical=read.delim("Clinical_data_CFS.txt",header=TRUE,sep="\t",row.names=1)
datExpr=read.delim("Expression_data_CFS.txt",header=TRUE,row.names=1,sep="\t")

#The following data frame contains the gene expression data: columns are genes,
#rows are arrays (samples)

datExpr=t(datExpr);datExpr=data.frame(datExpr);
datSNP=data.frame(datSNP);
datClinical=data.frame(datClinical);

#Check that the data has been read in correctly:
dim(datExpr)
#[1]   164 19797
dim(datClinical)
#[1]  68 164
dim(datSNP)
#[1] 164 42

#1. Construction of a gene co-expression network
#     A. Data pre-processing
#           i.Code to remove outlying arrays
par(cex=1.25,cex.axis=1.25,cex.main=1.75,cex.lab=1.25,mfrow=c(2,1),
mar=c(2,2,2,1))
par(mfrow=c(2,1))
h1=hclust(dist(datExpr),method="average")
plot(h1)
mean1=function(x) mean(x,na.rm=T)
median1=function(x) median(x,na.rm=T)

meanbyarray=apply(datExpr,1,mean1)
plot(meanbyarray)
```

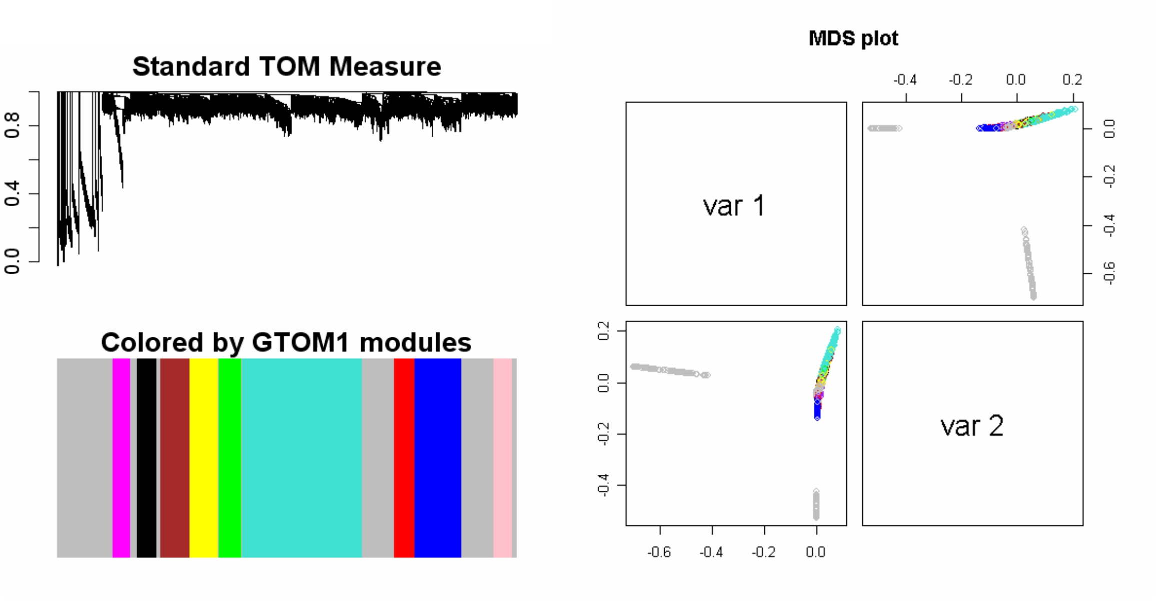




```
#Array X10268605 has an outlying gene expression profile and array X10043905
#has abnormally high average gene expression. Remove these two outliers arrays.

droparray=c(1,10)
datExpr=datExpr[-droparray,]; datSNP=datSNP[-droparray,];
datClinical=datClinical[,-droparray];

#1.A.ii. Code to remove outlying genes
#                               (1) 633 genes with ratios of mean:median
#                                      > 50 => 19164 genes
#                               (2) 811 genes with at least one intensity
#                                   reading greater than 50,000 => 18353 genes
#                               (3) Require the mean to be in the top 50% and
#                                   the variance to be in the 2/3rd's => 8966
#                                   genes

#(1)
mmratio=apply(datExpr,2,mean1)/apply(datExpr,2,median1)
datExpr=datExpr[,mmratio<50]
table(mmratio<50);
#FALSE  TRUE
#   633 19164

#(2)
mymax=function(x) {ifelse(max(x)<50000,TRUE,FALSE)};
```

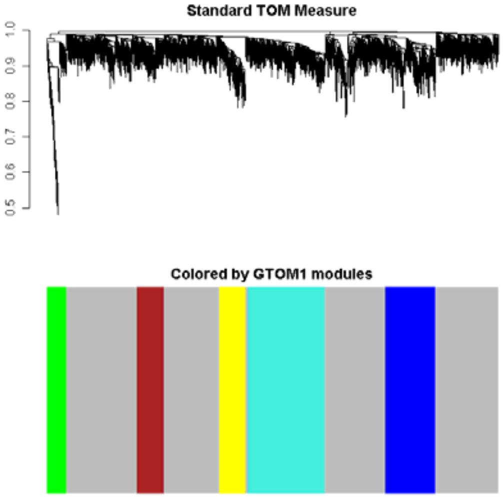
```
table(apply(datExpr,2,mymax))
datExpr=datExpr[,apply(datExpr,2,mymax)]
#FALSE  TRUE
#  811 18353

#(3)
varGenes=apply(datExpr,2,var)
meanGenes=apply(datExpr,2,mean)

selectGenes= meanGenes>quantile(meanGenes,0.5)& varGenes >
quantile(varGenes,0.33);
table(selectGenes);
#FALSE  TRUE
# 9387  8966

datExpr=datExpr[,selectGenes];dim(datExpr);
#[1]  162 8966

#1.A.iii. Remove all arrays/samples from datExpr and datClinical in the Intake
#classification control group (level 5), resulting in 127 arrays.

keeparrays=ifelse(as.numeric(datClinical[1,])<5,TRUE,FALSE)
table(keeparrays)

datClinical=datClinical[,keeparrays]
datExpr=datExpr[keeparrays,];
datSNP=datSNP[keeparrays,];
datExpr0 = datExpr; #Save this data frame for the standard analysis, Step 7.

dim(datExpr);dim(datClinical);dim(datSNP);
#[1]  127 8966
#[1]  68 127
#[1] 127  42

#1.B. SOFT THRESHOLDING
#Now we investigate soft thesholding with the power adjacency function.  Note
#that this code takes about 45 minutes to run and consequently it has been
#commented out. It produces the following table that was used to determine the
#power (beta) for transforming the correlation matrix.
# powers1=c(seq(1,10,by=1),seq(12,20,by=2))
# RpowerTable=PickSoftThreshold(datExpr, powervector=powers1)[[2]]
#   Power scale.law.R.2  slope truncated.R.2 mean.k. median.k. max.k.
# 1      1       -0.0525  0.246        -0.186 2560.00  2840.000 4340.0
# 2      2        0.0404 -0.456         0.197 1120.00  1210.000 2550.0
# 3      3        0.2290 -0.767         0.531  578.00   590.000 1610.0
# 4      4        0.3720 -0.980         0.716  324.00   309.000 1060.0
# 5      5        0.4470 -1.230         0.813  193.00   171.000  740.0
# 6      6        0.5120 -1.430         0.874  121.00    98.800  533.0
# 7      7        0.5780 -1.600         0.917   78.10    60.100  392.0
# 8      8        0.6360 -1.690         0.951   52.20    37.400  293.0
# 9      9        0.6930 -1.750         0.973   35.80    23.900  223.0
# 10    10        0.7430 -1.790         0.988   25.20    16.000  172.0
# 11    12        0.8020 -1.880         0.995   13.30     7.650  105.0
# 12    14        0.8260 -1.910         0.993    7.53     4.000   68.7
# 13    16        0.9100 -1.740         0.987    4.57     2.150   46.3
# 14    18        0.9240 -1.910         0.963    2.95     1.200   42.1
# 15    20        0.9410 -1.930         0.948    2.02     0.683   39.8

#Note: We would like the minimum Beta that: 1.) maximizes the scale.law.R.2
#(minimum of 0.8) and 2.) gives a dendrogram that can be divided into several
#modules (minimum of 3, excluding background "grey").  Furthermore, it is useful
#to acquire modules that are not defined by a data artifact (such as one outlier
#array).  Both beta=12 and beta=14 are reasonable choices, and the majority of
#these beta values should work well as WGCNA is relatively robust to the choice
```

```
#of beta (Zhang and Horvath 2005 [9]). In practice, Affymetrix data often
#results in good modules with low beta values ~4-8.

#1.C. Reduce the 8966 gene set to a more manageable number, ~3000 genes, by
#discarding genes with low connectivity.
windows()
beta1=14
datExpr=datExpr0;
kGenes=SoftConnectivity(datExpr,power=beta1)
quantile(kGenes)
table(kGenes>quantile(kGenes,0.66))

#Choose the genes with connectivities in the top 1/3rd.
datExpr=datExpr[,kGenes >quantile(kGenes,0.66)]
dim(datExpr)
#[1]  127 3049

#1.D. Create the adjacency and topological overlap matrices.
AdjMat1rest <- abs(cor(datExpr,use="p"))^beta1
collect_garbage()

#The following code computes the topological overlap matrix based on the
#adjacency matrix.

dissGTOM1=TOMdist1(AdjMat1rest)
collect_garbage()

#1.E. Now we carry out hierarchical clustering with the TOM matrix. Branches of
#the resulting clustering tree will be used to define gene modules.
hierGTOM1 <- hclust(as.dist(dissGTOM1),method="average");

colorh1=as.character(modulecolor2(hierGTOM1,h1=0.98, minsize1=100))
par(mfrow=c(2,1), mar=c(2,2,2,1))
plot(hierGTOM1, main="Standard TOM Measure", labels=F, xlab="", sub="");
hclustplot1(hierGTOM1,colorh1, title1="Colored by GTOM1 modules")
```

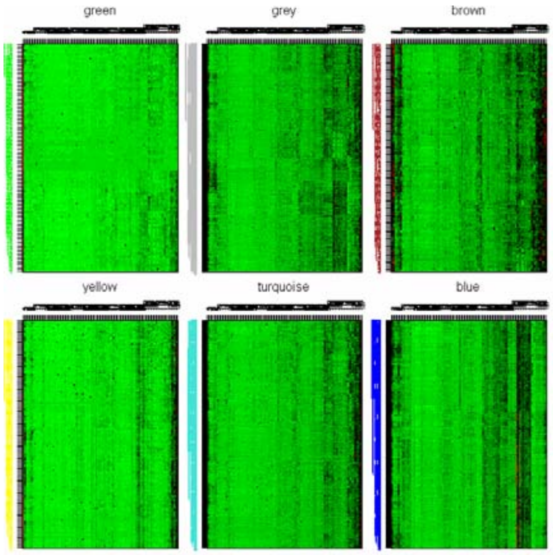


```
#The dendrogram and multi-dimensional scaling (MDS) plot show that some grey
#background genes are very different from the module genes. Note that the code
#for the MDS plot is not listed here, but it is given later in this tutorial.
#Try removing the grey genes to the left of the magenta module in the
#dendrogram.
```

```
ocolor=colorh1[hierGTOM1$order];length(ocolor);
mat=cbind(ocolor,hierGTOM1$order); # mat[1:400,]; #1st magenta is at index 373.

#Before removing these genes, check that they are not highly correlated with
#severity.
mean(abs(cor((datExpr[,hierGTOM1$order[1:372]]),t(datClinical[3,]),use="p")))
#[1] 0.1303418

selectind=hierGTOM1$order[373:3049]
datExpr=datExpr[,selectind];dim(datExpr); #127 x 2677
dissGTOM1=dissGTOM1[selectind, selectind]
hierGTOM1=hclust(as.dist(dissGTOM1),method="average");

colorh1=as.character(modulecolor2(hierGTOM1,h1=0.98, minsize1=100))
par(mfrow=c(2,1), mar=c(2,2,2,1))
plot(hierGTOM1, main="Clustered Gene Expression Values", labels=F, xlab="",
sub="");
hclustplot1(hierGTOM1,colorh1, title1="Colored by Modules")
#dev.copy2eps(file="figure2a.eps");dev.off()
```

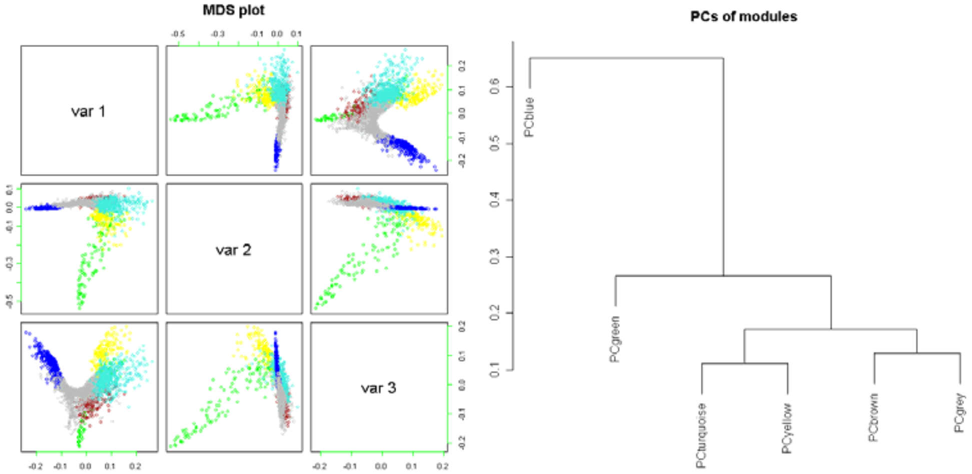


```
#1.F. Check that these modules are legitimate, for example not a result of an
#outlier array, using heat map and multi-dimensional scaling plots.
length(unique(colorh1))
#Choose a 2x3 layout "mfrow=c(2,3)" to include all 6 modules.

par(mfrow=c(2,3), mar=c(1, 2, 4, 1)) ;
ClusterSamples=hclust(dist(datExpr),method="average")

allcolors=unique(colorh1)
for(i in 1:length(allcolors)){
which.module=allcolors[i]
plot.mat(t(scale(datExpr[ClusterSamples$order,][,colorh1==which.module ])
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
title=which.module )}
```



```
#If an outlier array is present, it may be detected in the heatmap plots as a
#bright red line contrasted against a pure green background (or bright green
#line against a pure red background).

# We also check that the modules are distinct by using classical multi-
# dimensional scaling. Here we use 3 scaling dimensions.
# This may take several minutes.
cmd1=cmdscale(as.dist(dissGTOM1),3)
pairs(cmd1, col=as.character(colorh1),  main="MDS plot")

#pchf=rep(19,2677)
#pchf[colorh1=="grey"]=21
#plot(cmd1[,c(1,3)], col=as.character(colorh1), pch=pchf,
#cex=1.2,main="Multidimensional Scaling Plot of Hub Genes",xlab=list("Scaling
#Dimension 1",font=2), ylab=list("Scaling Dimension 2",font=2));
#dev.copy2eps(file="figure2b.eps")
#dev.off()

attach(data.frame(t(datClinical)))
table(sex)
#1  2
#98 29
#Recode CFS variables so that higher values = more severe.
severity = 4 - Cluster; table(severity);
sevEmpiric = 6 - Empiric; table(sevEmpiric);
snp12=as.vector(datSNP[,14]);table(snp12);
#0  1  2
#32 67 28

#PC1 code moved ahead of grey-shaded code below.
PC1=ModulePrinComps1(datExpr,colorh1)[[1]]
hclustPC1=hclust(as.dist( 1-abs(t(cor(PC1, method="p")))), method="average" )
par(mfrow=c(1,1))
plot(hclustPC1, main="PCs of modules")
PCblue=PC1[,1]

##The following code shaded grey is optional in case you plan to run this
##tutorial more than once (as it takes ~ 30 minutes or more to get to this
##point). The data sets created via the code below correspond to the 8966 and
##2677 gene expression sets discussed in the BMC Systems Biology article.
##Write data to file, so that next time it can be accessed via these files:
```

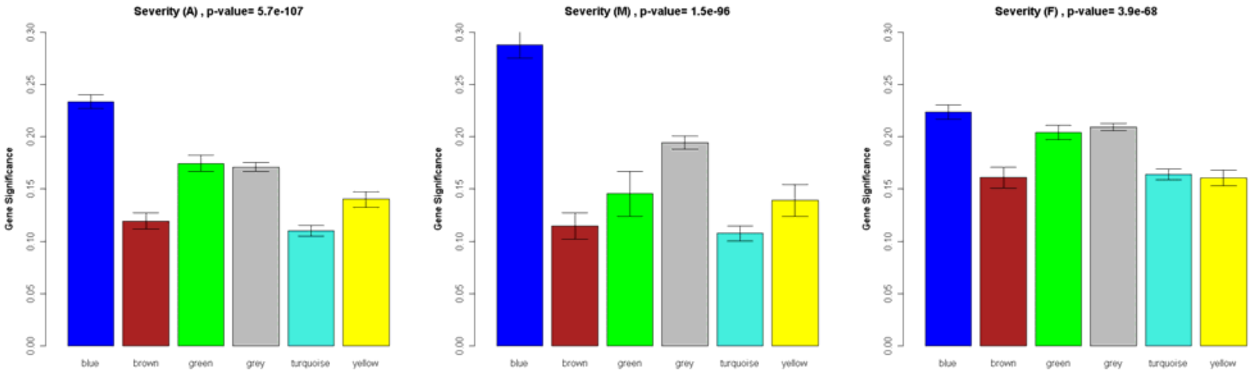
```
write.table(cbind(severity,sevEmpiric,snp12,PCblue,sex),"CFS_trait_data_127x5.tx
t",quote=F,sep="\t")
#write.table(datExpr0,"CFS expr data 127x8966.txt",quote=F,sep="\t")
#write.table(datExpr,"CFS expr data 127x2677.txt",quote=F,sep="\t")
#write.table(colorh1,"CFS module defs 2677.txt",quote=F,sep="\t")
#If you have created the files above, then you can start the tutorial from
#here (but you must reload the libraries and source code from page 5):
datTrait=read.table("CFS_trait_data_127x5.txt",header=TRUE,sep="\t",row.names=1#
,as.is=T)
datExpr0 = read.table("CFS_expr_data_127x8966.txt",header=TRUE,sep="\t",
row.names=1,as.is=T)
datExpr = read.table("CFS_expr_data_127x2677.txt",header=TRUE,sep="\t",
row.names=1,as.is=T)
colorh1 = read.table("CFS_module_defs_2677.txt"); colorh1 =
as.vector(colorh1[,2]);
#Check that the data has been read in correctly and attach the trait data:
dim(datExpr0)
#[1]  127 8966
dim(datExpr)
#[1]  127 2677
dim(datTrait);attach(datTrait);
#[1]  127 5
table(sex)
#1  2
#98 29
table(severity);
#severity
# 1  2  3
#15 48 24
table(sevEmpiric);
#sevEmpiric
# 1  2  3  4  5
#21 12 35 20 38
table(snp12);
#snp12
# 0  1  2
#32 67 28
table(colorh1);
#     blue     brown      green       grey  turquoise    yellow
#      299       157        115       1496        458       152
```

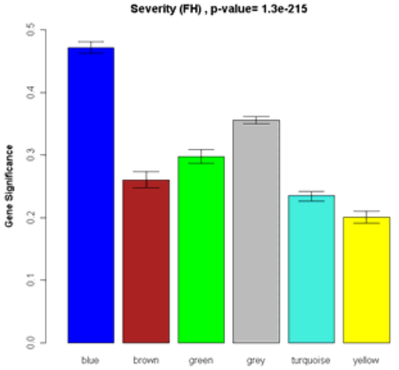


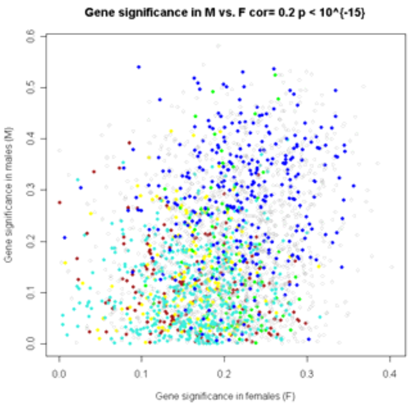
#2. Examining Network Properties

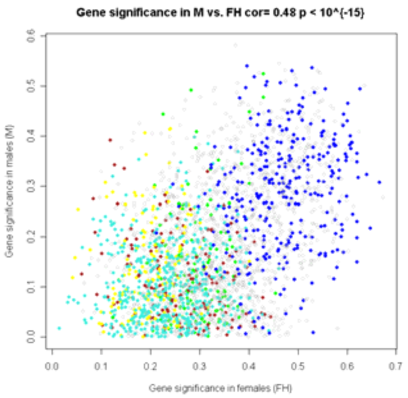
```
#A. Now investigate the relationship between the module-based gene expression
#and the trait data.  We explore this relationship in the following sample
#sets: males, females, homogenized females and second data set females.

#Define variables to stratify analyses based on sex, whether there is a
#severity score available, and homogenized and second data set samples.
allMales= sex==2;table(allMales);
#FALSE  TRUE
#   98    29
allFemales= sex==1;table(allFemales);
#FALSE  TRUE
#   29    98
sevScoreMales= sex==2 & !is.na(severity);table(sevScoreMales);
#FALSE  TRUE
#  104    23
sevScoreFemales= sex==1 & !is.na(severity);table(sevScoreFemales);
#FALSE  TRUE
#   63    64
sevScoreAll=!is.na(severity); table(sevScoreAll);
#FALSE  TRUE
#   40    87

#Homogenizing the female samples makes use of the fact that the blue module
#eigengene is related to CFS severity -- samples with high severity and low
#expression are discarded, and samples with low severity and high severity are
#discarded.
homFemales= sex==1 & (PCblue>median(PCblue[sex==1]) & severity>1 |  PCblue<=
median(PCblue[sex==1]) & severity<3);
homFemales[is.na(homFemales)]=FALSE
table(homFemales);
#FALSE  TRUE
#   74    53

#Define gene significance, SNP significance and connectivity measures for
#all, males, females, homogenized females and second data set samples.
GSseverity=as.numeric((cor(severity, datExpr, use="p",
method="p")));absGSseverity=abs(GSseverity);
GSseveritym=as.numeric((cor(severity[allMales], datExpr[allMales,], use="p",
method="p")));absGSseveritym=abs(GSseveritym);
GSseverityf=as.numeric((cor(severity[allFemales], datExpr[allFemales,], use="p",
method="p")));absGSseverityf=abs(GSseverityf);
GSseverityfh=as.numeric((cor(severity[homFemales], datExpr[homFemales,],
use="p", method="p")));absGSseverityfh =abs(GSseverityfh);


#2.B. Compute the SNP significance measure for each subgroup.
GSsnp= as.numeric((cor(snp12, datExpr, use="p",
method="p")));absGSsnp=abs(GSsnp)
GSsnpm= as.numeric((cor(snp12[allMales], datExpr[allMales,], use="p",
method="p")));absGSsnpm=abs(GSsnpm)
GSsnpf= as.numeric((cor(snp12[allFemales], datExpr[allFemales,], use="p",
method="p")));absGSsnpf=abs(GSsnpf)
GSsnpfh= as.numeric((cor(snp12[homFemales], datExpr[homFemales,], use="p",
method="p")));absGSsnpfh=abs(GSsnpfh)

#2.C. Compute the connectivity for each subgroup.
kme = cor(PC1,datExpr,use="p",method="p")
kmeM  = cor(PC1[allMales,],datExpr[allMales,],use="p",method="p")
kmeF  = cor(PC1[allFemales,],datExpr[allFemales,],use="p",method="p")
kmeFH  = cor(PC1[homFemales,],datExpr[homFemales,],use="p",method="p")


#2.D. Construct GSseverity bar plots for all, male and female samples. Also
#compare gene significance and connectivities between male and female samples.
```

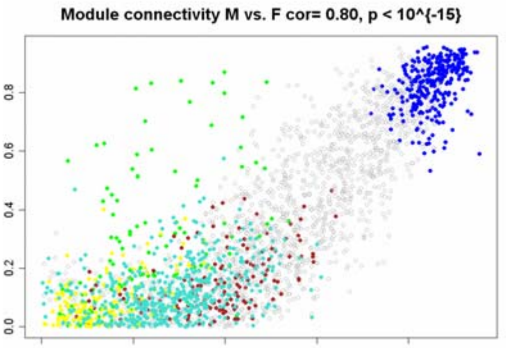
```
pchf=rep(19,length(colorh1));pchf[colorh1=="grey"]=21;
ModuleEnrichment1(abs(as.numeric(GSseverity)),colorh1,title1="a.) Severity
(A)",label=list("Gene Significance",font=2)) #, limit=c(0,0.3));
#dev.copy2eps(file="figure3a.eps")
#dev.off()
ModuleEnrichment1(abs(as.numeric(GSseveritym)),colorh1,title1="Severity
(M)",label=list("Gene Significance",font=2)) # , limity=c(0,0.3));
#dev.copy2eps(file="figure3b.eps")
#dev.off()
ModuleEnrichment1(abs(as.numeric(GSseverityf)),colorh1,title1="Severity
(F)",label=list("Gene Significance",font=2)) #, limity=c(0,0.3));
#dev.copy2eps(file="figure3c.eps")
#dev.off()
ModuleEnrichment1(abs(as.numeric(GSseverityfh)),colorh1,title1="Severity
(FH)",label=list("Gene Significance",font=2)) #,limit=c(0,0.5));
#dev.copy2eps(file="figure3d.eps")
#dev.off()
par(mfrow=c(2,2));
plot(abs(GSseverityf), abs(GSseveritym),col=colorh1,xlab="Gene significance in
females (F)", pch=pchf,ylab="Gene significance in males (M)",main="Gene
significance in M vs. F cor= 0.2 p < 10^{-15}")
plot(abs(GSseverityfh), abs(GSseveritym),col=colorh1,xlab="Gene significance in
females (FH)", pch=pchf,ylab="Gene significance in males (M)",main="Gene
significance in M vs. FH cor= 0.48 p < 10^{-15}")
plot(abs(kmeF[1,]), abs(kmeM[1,]),col=colorh1,xlab="Module connectivity in
females (F)", pch=pchf,ylab="Module connectivity in males (M)", main="Module
connectivity M vs. F cor= 0.80, p < 10^{-15}")
plot(abs(kmeFH[1,]), abs(kmeM[1,]),col=colorh1,xlab="Module connectivity in
homogenized females (F)", pch=pchf,ylab="Module connectivity in males (M)",
main="Module connectivity M vs. F cor= 0.78, p < 10^{-15}")
```

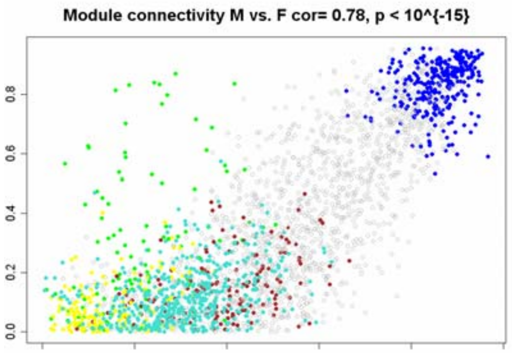






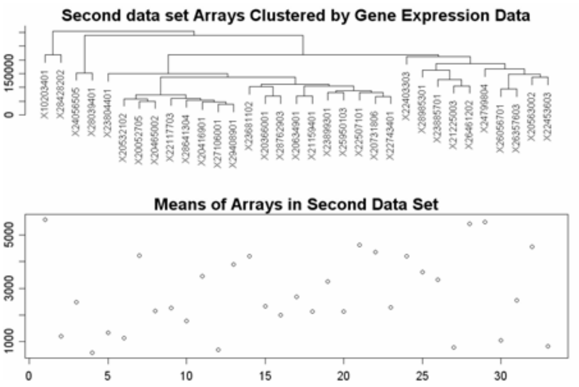




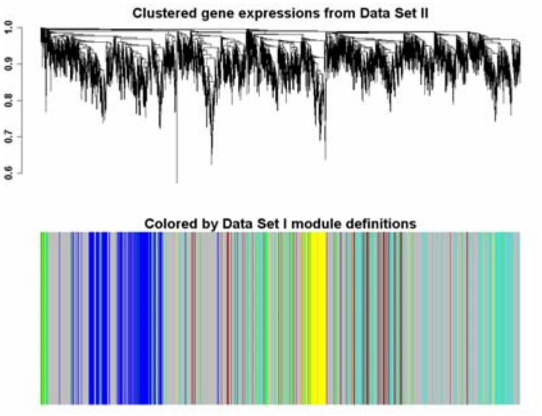


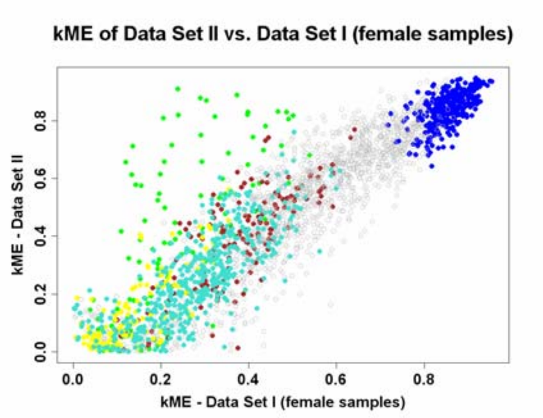
```
#The blue module genes have the most significant association with CFS severity
#in all samples and in both males and females considered separately. The
#homogenized female sample set (53) has a stronger association with CFS
#severity than the combined female sample set (64).

#There is no relationship between genes associated with CFS in males vs. genes
#associated with CFS in females, based on all female samples (above).  As a
#result, it may be useful to homogenize female samples to study those with
#similar genetic properties to the male samples.  Also, it will be useful to
#consider gender in the gene screening process. Note that when we compare gene
#significance between males and homogenized female samples, there is some
#association.  The blue module contains genes that are highly connected in both
#sexes, while the other modules contain genes with lower connectivities.

#3. Gene Screening Strategy
#A. Examine quantiles of GSseverity and GSsnp among subgroups.
quantile(absGSseverityym[colorh1=="blue"])
#          0%         25%         50%         75%        100%
#0.006756411 0.208796666 0.295572158 0.365277029 0.540251202

quantile(absGSseverityf[colorh1=="blue"])
#          0%         25%         50%         75%        100%
#0.007263834 0.185594089 0.223098624 0.272583149 0.356835798

quantile(absGSseverityfh[colorh1=="blue"])
#        0%       25%       50%       75%      100%
#0.2120275 0.4175653 0.4703933 0.5315133 0.6666089

quantile(absGSsnpm[colorh1=="blue"])
#          0%         25%         50%         75%        100%
#0.001336947 0.162282719 0.237828380 0.315607261 0.476327309

quantile(absGSsnpf[colorh1=="blue"])
#          0%         25%         50%         75%        100%
#0.001312125 0.061739063 0.102307557 0.145276626 0.244598357
```

```
quantile(absGSsnpfh[colorh1=="blue"])
#          0%          25%          50%          75%         100%
#0.001295996 0.081906219 0.142802430 0.196723305 0.348869071

#B. Screen for genes based on GSseverity, GSSNP, and connectivity in both males
#and homogenized females.
selM=absGSseveritym[colorh1=="blue"]>.2&(absGSsnpm[colorh1=="blue"])>.2&kmeM[1,
colorh1=="blue"]>quantile(kmeM[1,colorh1=="blue"],probs=.2)
selFH=absGSseverityfh[colorh1=="blue"]>.35&(absGSsnpfh[colorh1=="blue"])>.2&kmeF
H[1,colorh1=="blue"]>quantile(kmeFH[1,colorh1=="blue"],probs=.2)

table(selM)
#FALSE  TRUE
#  148   151
table(selFH)
#FALSE  TRUE
#  237    62

#Select only genes with same GS & SNP directions in M & F
selposSNP =
GSsnpm[colorh1=="blue"]>0&GSsnpfh[colorh1=="blue"]>0|GSsnpm[colorh1=="blue"]<0&G
Ssnpfh[colorh1=="blue"]<0
selposSev =
GSseveritym[colorh1=="blue"]>0&GSseverityfh[colorh1=="blue"]>0|GSseveritym[color
h1=="blue"]<0&GSseverityfh[colorh1=="blue"]<0
selCriteria1=selFH&selM&selposSNP&selposSev

#C. The screening strategy results in 20 candidate genes.
blueNames=names(datExpr[,colorh1=="blue"])
bgenes1= blueNames[selCriteria1]
length(bgenes1)
#[1] 20

bgenes1
#[1] "AF407165"  "AF077197"  "AK027815"  "AK024017"  "AK026314"
#[6] "AK027673"  "AF090939"  "AF118073"  "BC010019"  "AB062432"
#[11] "AK001458"  "BC001000"  "AF081282"  "BC020646"  "NM_002958"
#[16] "NM_003593" "XM13557"   "AF101044"  "NM_006400" "AF111802"

#4. Second data set Results
valFemales= sex==1&is.na(severity)&!is.na(sevEmpiric); table(valFemales);
#FALSE  TRUE
#   94    33
valhomFemales = valFemales & (PCblue>median(PCblue[sex==1],na.rm=T) &
sevEmpiric>1 |  PCblue<= median(PCblue[sex==1],na.rm=T) & sevEmpiric <5);
table(valhomFemales);
#FALSE  TRUE
#   97    30

#A. First check for outliers.
par(cex=1.25,cex.axis=1.25,cex.main=1.75,cex.lab=1.25,mfrow=c(2,1),
mar=c(2,2,2,1))
h1=hclust(dist(datExpr[valFemales,]),method="average")
plot(h1,main="Second data set Arrays Clustered by Gene Expression Data",xlab =
"",sub="")
meanbyarray=apply(datExpr[valFemales,],1,mean1)
plot(meanbyarray,main="Means of Arrays in Second Data Set",ylab="Array Mean")
```



```
#No substantial outliers.


#B. Compute the equivalent GSseverity in the second data set: GSsevEmpiric, the
#SNP significance GSsnpfv and the connectivities.
GSsevEmpiric=as.numeric((cor(sevEmpiric[valFemales], datExpr[valFemales,],
use="p", method="p")));absGSsevEmpiric=abs(GSsevEmpiric);
GSsevEmpiricH=as.numeric((cor(sevEmpiric[valhomFemales],
datExpr[valhomFemales,], use="p", method="p"))); absGSsevEmpiricH =
abs(GSsevEmpiricH);

GSsnpfv= as.numeric((cor(snp12[valFemales], datExpr[valFemales,], use="p",
method="p")));absGSsnpfv=abs(GSsnpfv)

GSsnpfvh= as.numeric((cor(snp12[valhomFemales], datExpr[valhomFemales,],
use="p", method="p")));absGSsnpfvh=abs(GSsnpfvh)

kmeFV = cor(PC1[valFemales,],datExpr[valFemales,],use="p",method="p")
kmeFVH = cor(PC1[valhomFemales,],datExpr[valhomFemales,],use="p",method="p")

#C. Create a gene co-expression network based on the second data set samples and
#color the clustered genes by their definitions in the original (127 sample)
#data set.
beta1=14
AdjMat1restfv <- abs(cor(datExpr[valFemales,],use="p"))^beta1
collect_garbage()
dissGTOM1fv=TOMdist1(AdjMat1restfv)
collect_garbage()
hierGTOM1fv <- hclust(as.dist(dissGTOM1fv),method="average");

par(cex=1.25,cex.axis=1.25,cex.main=1.75,cex.lab=1.25,mfrow=c(2,1),
mar=c(2,2,2,1))
plot(hierGTOM1fv, main=list("Clustered gene expressions from Data Set
II",font=2),
labels=F,xlab="", sub="");
hclustplot1(hierGTOM1fv,colorh1, title1=list("Colored by Data Set I module
definitions",font=2));
#dev.copy2eps(file="figure4a.eps")
#dev.off()
```

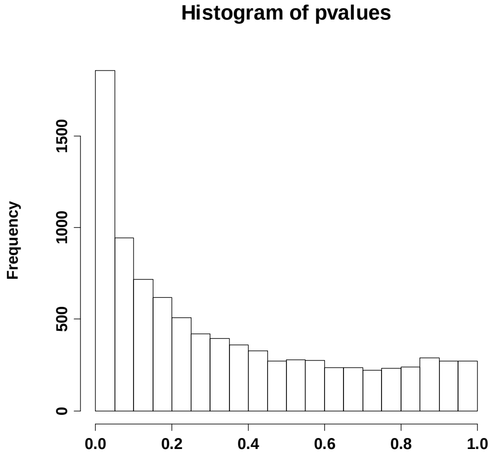


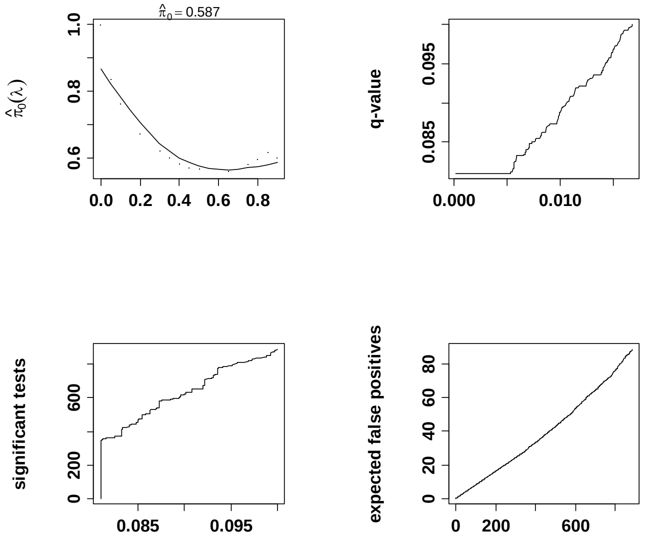


```
#This dendrogram of clustered gene expressions in the second data set colored by
#the original module definitions shows that the blue and yellow modules are the
#most preserved.

#It is also useful to check if the gene connectivities are similar between the
#original female sample set and the validation samples.

par(mfrow=c(1,1))
par(cex=1.25,cex.axis=1.5,cex.main=2,cex.lab=1.5,font=2,font.axis=2,font.lab=2,f
ont.main=2,mar=(c(5,5,6,3)+0.1),mgp=c(3,1,0))
plot(abs(kmeF[1,]),abs(kmeFV[1,]), col=as.character(colorh1), pch=pchf,
cex=1.2,main="kME of Data Set II vs. Data Set I (female samples)",ylab=list("kME –
Data Set II",font=2), xlab=list("kME – Data Set I (female samples)",font=2));
#dev.copy2eps(file="figure4b.eps")
#dev.off()

#The connectivities are well preserved between the two groups, and again the
#blue module is the most connected.


#D. Now check whether the candidate genes validate by applying the same
#screening criteria.

#Map the screening parameter values for the FH sample to the validation sample
#set by finding its percentile.
table(GSseverityfh[colorh1=="blue"]<0.35)/299
#     FALSE       TRUE
#0.94314381 0.05685619
table(GSsnpfh[colorh1=="blue"]<0.2)/299
#    FALSE      TRUE
#0.2474916 0.7525084

#Note: The absGSsevEmpiric was set to the 85th percentile (> 0.15 quantile)
#rather than the 94th percentile (> 0.06 quantile) which would conform to the
#screening procedure on the original data set (see commented out code, shaded
#grey, below).  This was possibly an error, but both values result in the same 6
#candidate genes.

selposVal = GSsnpfv[colorh1=="blue"]>0|GSsnpfv[colorh1=="blue"]<0 &
GSsevEmpiric[colorh1=="blue"]>0 | GSsevEmpiric[colorh1=="blue"]<0
selFV=absGSsevEmpiric[colorh1=="blue"]>quantile(absGSsevEmpiric[colorh1=="blue"]
,0.06)&absGSsnpfv[colorh1=="blue"]>quantile(absGSsnpfv[colorh1=="blue"],0.75)&
kmeFV[1,colorh1=="blue"]>quantile(kmeFV[1,colorh1=="blue"],probs=.2)


#selposVal = GSsnpfv[colorh1=="blue"]>0|GSsnpfv[colorh1=="blue"]<0 &
#GSsevEmpiric[colorh1=="blue"]>0 | GSsevEmpiric[colorh1=="blue"]<0
```

```
#selFV=absGSsevEmpiric[colorh1=="blue"]>quantile(absGSsevEmpiric[colorh1=="blue"
#],0.15)&absGSsnpfv[colorh1=="blue"]>quantile(absGSsnpfv[colorh1=="blue"],0.75)&
#kmeFV[1,colorh1=="blue"]>quantile(kmeFV[1,colorh1=="blue"],probs=.2)

selCriteria2= selFV&selposVal
selCriteria3= selCriteria1&selCriteria2

bgenes2= blueNames[selCriteria2]
length(bgenes2)  #61 genes
bgenes3= blueNames[selCriteria3]
length(bgenes3)  #6 genes
bgenes3
#[1] "AF407165"  "AF077197"  "AK026314"  "AF090939"  "NM_003593"
#[6] "NM_006400"
# PPP1R14C,VAMP5, TFB2M, (PRO0641 -withdrawn), FOXN1,DCTN2

#The FOXN1 (NM_003593) gene and 5 others validate.

#5. Summarizing the results in a table of correlations (Table 2 in
#BMC Systems Biology paper).
which(names(datExpr)=="NM_003593")
#[1] 2275
foxn1=datExpr[,2275];
#attach(datSNP);snp12=as.vector(hCV245410);
cor.test(PC1[,1],severity,use="p")
cor.test(PC1[allMales,1], severity[allMales],use="p")
cor.test(PC1[allFemales,1], severity[allFemales],use="p")
cor.test(PC1[valFemales,1], sevEmpiric[valFemales],use="p")
cor.test(PC1[homFemales,1], severity[homFemales],use="p")
#Since both SNP and severity measures are categorical, compute the Spearman
#correlations.
cor.test(snp12,severity,use="p",method="s")
cor.test(snp12[allMales], severity[allMales],use="p",method="s")
cor.test(snp12[allFemales], severity[allFemales],use="p",method="s")
cor.test(snp12[valFemales], sevEmpiric[valFemales],use="p",method="s")
cor.test(snp12[homFemales], severity[homFemales],use="p",method="s")

#Compute cor(expression profile,severity) and cor(expression profile,snp12) for
#the FOXN1 gene.
cor.test(foxn1,severity,use="p")
cor.test(foxn1[allMales], severity[allMales],use="p")
cor.test(foxn1[allFemales], severity[allFemales],use="p")
cor.test(foxn1[valFemales], sevEmpiric[valFemales],use="p")
cor.test(foxn1[homFemales], severity[homFemales],use="p")

cor.test(foxn1[!is.na(severity)],snp12[!is.na(severity)],use="p")
cor.test(foxn1[sevScoreMales], snp12[sevScoreMales],use="p")
cor.test(foxn1[sevScoreFemales], snp12[sevScoreFemales],use="p")
cor.test(foxn1[valFemales], snp12[valFemales],use="p")
cor.test(foxn1[homFemales], snp12[homFemales],use="p")

valhomFemales = valFemales & (PCblue>median(PCblue[sex==1],na.rm=T) &
sevEmpiric>1 |  PCblue<= median(PCblue[sex==1],na.rm=T) & sevEmpiric <5);
table(valhomFemales);
#FALSE  TRUE
#   97    30

#Compute correlations for homogenized validation samples.
cor.test(PC1[valhomFemales,1], sevEmpiric[valhomFemales],use="p")
cor.test(snp12[valhomFemales], sevEmpiric[valhomFemales],use="p",method="s")
cor.test(foxn1[valhomFemales], sevEmpiric[valhomFemales],use="p")
cor.test(foxn1[valhomFemales], snp12[valhomFemales],use="p")

#Also compute correlations for male and homogenized female samples combined.
```

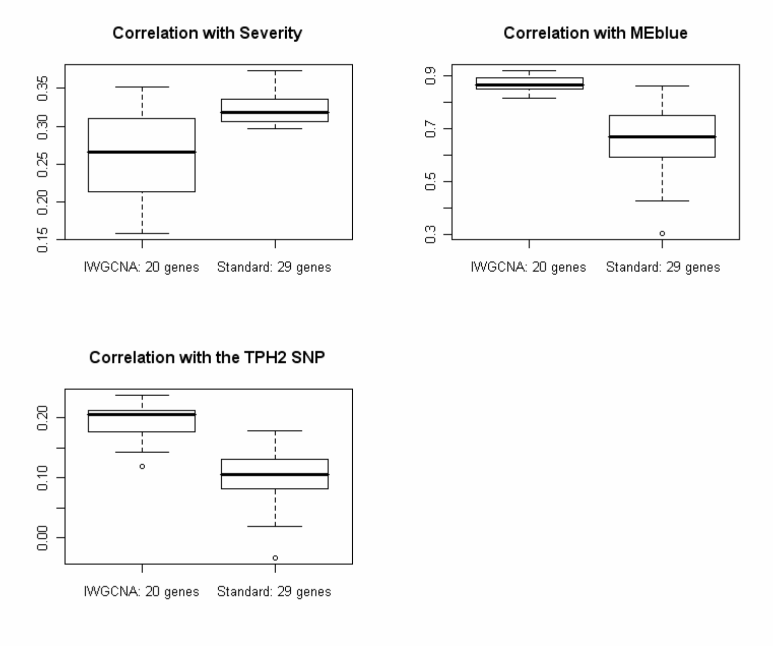
```
combMFH = homFemales | sevScoreMales; table(combMFH);
#FALSE  TRUE
#   51    76
cor.test(PC1[combMFH,1], severity[combMFH],use="p")
cor.test(snp12[combMFH], severity[combMFH],use="p",method="s")
cor.test(foxn1[combMFH], severity[combMFH],use="p")
cor.test(foxn1[combMFH], snp12[combMFH],use="p")


#6. Causality analysis using LEO (single.marker.analysis).

#A. Calculate LEO.NB.SingleMarker scores for all genes in the blue module using
all samples with severity scores (87).
alldat = data.frame(snp12[sevScoreAll],
PC1[sevScoreAll,1],datExpr[sevScoreAll,colorh1=="blue"])
SM.all=single.marker.analysis(alldat, snpcols=1, genecols=3:dim(alldat)[2],
traitcols=2)

#B. Calculate LEO.NB.SingleMarker scores for all genes in the blue module using
#male and homogenized females with severity scores (76).

alldat.mfh = data.frame(snp12[combMFH],
PC1[combMFH,1],datExpr[combMFH,colorh1=="blue"])
SM.mfh=single.marker.analysis(alldat.mfh, snpcols=1,
genecols=3:dim(alldat.mfh)[2], traitcols=2)

#We can then write our results to a file for later perusal with excel. First
#create an indicator for the 20 candidate genes.

candidate.genes = rep(0,length(blueNames))
candidate.genes [selCriteria1]=1;

#write.csv(cbind(candidate.genes,SM.all),"SingleMarkerAnalysis.ALL.CFS.csv")
#write.csv(cbind(candidate.genes,SM.mfh),"SingleMarkerAnalysis.MFH.CFS.csv")

allDat = data.frame(snp12,PCblue,datExpr[,colorh1=="blue"])
SM=single.marker.analysis(allDat, snpcols=1, genecols=3:dim(allDat)[2],
traitcols=2);


############# Saved in Workspace "R_5-2-07.Rdata" ################

#7. Standard analysis of trait and gene expression data (ignoring
#the SNP marker).
library(qvalue);

#7.A. Calculate p-values and q-values for the correlation between severity and
#datExpr, find the genes with the smallest q-value(s).

mypvals = function(x){cor.test(severity,datExpr0[,x],method="p",
use="p")$p.value}
pvalues = sapply(1:dim(datExpr0)[2],mypvals);hist(pvalues);
qobj = qvalue(pvalues)
qplot(qobj)
#qwrite(qobj, filename="myresults.txt")
table(qobj$qvalues==min(qobj$qvalues))
#FALSE  TRUE
# 8620   346
#There were 346 genes with the minimum q-value = 0.081
qvalues = qobj$qvalues
selq = qobj$qvalues==min(qobj$qvalues)
```





```
#Save the 346 genes to a file for IPA.
#write.table(names(datExpr0[1,selq]),"top346-qvalue-genes.txt" ,quote=F)
#cor(datExpr0[,selq],severity,use="p")

#7.B. These 346 genes were analyzed using Ingenuity Pathways Analysis (IPA)
#software (August 2008) and the top network was selected, which consisted of 29
#candidate genes.

#7.C. Calculate correlations for the 29 candidate genes selected using IPA.
#7.C.i. Calculate the correlations between these 29 candidate genes and
#severity.

c29.genes = read.table("std-analysis-29-candidate-genes-IPA.txt")
c29.genes = as.character(unlist(c29.genes))
table(!is.na(match(names(datExpr0[1,]),c29.genes)))
#FALSE  TRUE
# 8937    29
s.match = !is.na(match(names(datExpr0[1,]),c29.genes))
c29.GSseverity = cor(datExpr0[,s.match],severity,use="p"); #All 87 samples
c29.GSseveritym = cor(datExpr0[sex==2, s.match],severity[sex==2],use="p");
c29.GSseverityf = cor(datExpr0[sex==1, s.match],severity[sex==1],use="p");
cbind(c29.GSseverity,qvalues[s.match],pvalues[s.match], c29.GSseveritym,
c29.GSseverityf)
#                [,1]       [,2]         [,3]       [,4]      [,5]
#AB015292  0.3528525 0.08102779 0.0008020260 0.33003792 0.3788854
#AB019246  0.3493106 0.08102779 0.0009126929 0.36782653 0.3237809
#AF078165  0.3346708 0.08102779 0.0015331009 0.38921199 0.3179558
#AF087573  0.3326885 0.08102779 0.0016415146 0.25284583 0.3503530
#AF112880  0.3403612 0.08102779 0.0012569048 0.17281973 0.3553763
#

#7.C.ii Calculate their correlations with SNP12.
c29.GSsnp = cor(datExpr0[,s.match],snp12,use="p");
c29.GSsnpm = cor(datExpr0[sex==2, s.match],snp12[sex==2],use="p"); #Males
c29.GSsnpf = cor(datExpr0[sex==1, s.match],snp12[sex==1],use="p"); #Females
c29.datExpr = datExpr0[,s.match]
mypvals2 = function(x){cor.test(snp12, c29.datExpr[,x],method="p",
use="p")$p.value}
pvalues2 = sapply(1:dim(c29.datExpr)[2],mypvals2);
indc = 1:length(s.match);indc[s.match];
```

```
c29.indices = indc[s.match]

#7.C.iii Calculate the ranks of their PCblue correlations (out of 8966 genes).

c29.kme = cor(PCblue, c29.datExpr);dim(c29.kme);
network.cor = cor(PCblue,datExpr0);dim(network.cor);
#[1]    1 8966
ordercor = sort(abs(network.cor),decreasing=T,index.return=T)[[2]]
vk = match(c29.indices,ordercor)
cbind(names(c29.datExpr),as.vector(c29.kme),vk)
#                                      vk
# [1,] "AB015292"  "0.426738577002206" "3434"
# [2,] "AB019246"  "0.47467433186572"  "2945"
# [3,] "AF078165"  "0.799479824719028" "414"
# [4,] "AF087573"  "0.706619969601692" "948"
# [5,] "AF112880"  "0.631778630979381" "1517"
# …

#7.D. Extract these results for the 20 IWGCNA genes.

#Severity,SNP12 and PCblue correlations
c20.GSseverity = GSseverity[colorh1=="blue"][selCriteria1]
c20.GSseveritym = GSseveritym[colorh1=="blue"][selCriteria1]
c20.GSseverityf = GSseverityf[colorh1=="blue"][selCriteria1]

c20.GSsnp = GSsnp[colorh1=="blue"][selCriteria1]
c20.GSsnpm = GSsnpm[colorh1=="blue"][selCriteria1]
c20.GSsnpf = GSsnpf[colorh1=="blue"][selCriteria1]

c20.kme = kme[1,colorh1=="blue"][selCriteria1]
c20.kmeM = kmeM[1,colorh1=="blue"][selCriteria1]
c20.kmeF = kmeF[1,colorh1=="blue"][selCriteria1]

#Indices for 20 IWGCNA candidate genes in 8966 gene set.
c20.indices = match(bgenes1,names(datExpr0[1,]))
cbind(names(datExpr0[1, c20.indices]),bgenes1)

#Ranks for 20 IWGCNA candidate genes out of kme for all 8966 genes.
vk2 = match(c20.indices,ordercor);vk2;
cbind(bgenes1,as.vector(c20.kme),vk2)
#     bgenes1                         vk2
# [1,] "AF407165"  "0.86580913967316"  "114"
# [2,] "AF077197"  "0.863317298413618" "124"
# [3,] "AK027815"  "0.85596588593319"  "145"
# [4,] "AK024017"  "0.918508898096407" "10"
# [5,] "AK026314"  "0.898666181032471" "49"
# …

#7.E. Compare the 20 IWGCNA results with the 29 standard analysis candidates.
par(mfrow=c(2,2))
boxplot(c20.GSseverity, c29.GSseverity,names=c("IWGCNA: 20 genes","Standard: 29
genes"),main="Correlation with Severity")
boxplot(c20.kme, c29.kme,names= c("IWGCNA: 20 genes"," Standard: 29
genes"),main="Correlation with MEblue")
boxplot(c20.GSsnp, c29.GSsnp,names= c("IWGCNA: 20 genes","Standard: 29
genes"),main="Correlation with the TPH2 SNP")
dev.copy2pdf(file="figure6.pdf")
dev.off()
```



References

1. Fukuda K, Straus SE, Hickie I, Sharpe MC, Dobbins JG, Komaroff A (1994) The chronic fatigue syndrome: a comprehensive approach to its definition and study. International Chronic Fatigue Syndrome Study Group. Ann Intern Med 121:953-959
2. Horvath S, Xu X, Lake SL, Silverman EK, Weiss ST, Laird NM (2004) Family-based tests for associating haplotypes with general phenotype data: application to asthma genetics. Genet Epidemiol 26:61-69
3. Kaasa S, Knobel H, Loge JH, Hjermstad MJ (1998) Hodgkin's disease: quality of life in future trials. Ann Oncol 9 Suppl 5:137-145
4. Ravasz E, Somera AL, Mongru DA, Oltvai ZN, Barabasi AL (2002) Hierarchical organization of modularity in metabolic networks. Science 297:1551-1555
5. Reeves WC, Lloyd A, Vernon SD, Klimas N, Jason LA, Bleijenberg G, Evengard B, White PD, Nisenbaum R, Unger ER (2003) Identification of ambiguities in the 1994 chronic fatigue syndrome research case definition and recommendations for resolution. BMC Health Serv Res 3:25
6. Reeves WC, Wagner D, Nisenbaum R, Jones JF, Gurbaxani B, Solomon L, Papanicolaou DA, Unger ER, Vernon SD, Heim C (2005) Chronic fatigue syndrome--a clinically empirical approach to its definition and study. BMC Med 3:19
7. Smith AK, White PD, Aslakson E, Vollmer-Conna U, Rajeevan MS (2006) Polymorphisms in genes regulating the HPA axis associated with empirically delineated classes of unexplained chronic fatigue. Pharmacogenomics 7:387-394
8. Yip AM, Horvath S (2007) Gene network interconnectedness and the generalized topological overlap measure. BMC Bioinformatics 8:22
9. Zhang B, Horvath S (2005) A general framework for weighted gene co-expression network analysis. Stat Appl Genet Mol Biol 4:Article17