YEAST
Gene Co-expression Network Analysis
R Tutorial

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 with the R software. We show how to construct unweighted networks using hard thresholding and how to construct weighted networks 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. *Bin Zhang and Steve Horvath (2005) "A General Framework for Weighted Gene Co-Expression Network Analysis", Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17* Technical Report and software code at: www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork

More details on the biological results can be found in the following reference.

- *Carlson MRJ, Zhang B, Fang Z, Mischel P, Horvath S, Nelson SF (2006) Gene Connectivity, Function, and Sequence Conservation: Predictions from Modular Yeast Co-Expression Networks. BMC Genomics 2006, 7:40 (3).*

The following theoretical reference explores the meaning of coexpression network analysis

- *Horvath S, Dong J (2008) Geometric Interpretation of Gene Co-Expression Network Analysis. PloS Computational Biology. 4(8): e1000117. PMID: 18704157*

The WGCNA R package is described in

- *Langfelder P, Horvath S (2008) WGCNA: an R package for Weighted Correlation Network Analysis. BMC Bioinformatics. 2008 Dec 29;9(1):559. PMID: 19114008*

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

- *Yip A, Horvath S (2007) Gene network interconnectedness and the generalized topological overlap measure. BMC Bioinformatics 8:22*

Microarray DATA

To demonstrate our methods, we apply them to gene co-expression networks constructed from a yeast microarray data set. These yeast microarray data record gene expression levels during different stages of cell cycles in yeasts.
The yeast data are described in the following reference.

Eisen, M. and Spellman, P. and Brown, P. and Botstein, D. (1998)
Cluster Analysis and Display of Genome-wide Expression Patterns,
PNAS, vol 95 (25) 14863-14868.

Actually our data set represents a subset of genes and samples from the original data set.
We omitted genes with a lot of missing values and with low variance across the 44 yeast samples.
The resulting data set contains 4000 different genes.

Prediction of "Essential" yeast genes.
For each yeast gene, it is known whether or not is is essential for yeast survival based on gene knock-out experiments. This binary information is encoded in the variable "essentiality" which equals 1 if the gene is essential otherwise 0.
It has been reported independently by many research groups, genes with high connectivity (are more likely to be essential. We will confirm this finding as part of this tutorial.
We refer to essentiality as a measure of "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 essentiality) to intramodular connectivity.

Module Construction
To group genes with coherent expression profiles into modules, we use average linkage hierarchical clustering, which uses the topological overlap measure as dissimilarity.
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 some of the resulting branches correspond to the discrete diagonal blocks (modules) in the TOM plot. In general, we recommend to use the function cutreeDynamic for branch cutting (Langfelder and Horvath 2008). But here we will use the static (fixed) cutree method.

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

```
# CONTENTS
# This document contains function 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.
# G) Miscellaneous other functions, e.g. for computing the cluster coefficient.
```

Downloading the R software
1) Go to http://www.R-project.org, download R and install it on your computer

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.

To get this tutorial and data files, go to the following webpage
www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork
Download the zip file containing:
1) R function file: "NetworkFunctions.txt", which contains several R functions
needed for Network Analysis.
2) A test data file: "YEASTCellCycle4000.csv"
3) Of course, this file: "YeastTutorialHorvath.txt"

Unzip all the files into the same directory

Please copy and paste the following script into the R session.
Text after "#" is a comment and is automatically ignored by R.

read in the R libraries
library(MASS) # standard, no need to install
library(class) # standard, no need to install
library(cluster)
library(impute)# install it for imputing missing value

 # Download the WGCNA library as a .zip file from
http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/Rpackages/WGCNA/
and choose "Install package(s) from local zip file" in the packages tab

library(WGCNA)
options(stringsAsFactors=F)

```
# note the backslashes / in the following path

setwd("C:/Documents and Settings/Steve Horvath/My
Documents/ADAG/LinSong/NetworkScreening/")
# read in the microarray data
dat0 = read.csv("YEASTCellCycle4000.csv",header=T, row.names=1)

# the following gene summary file contains information on the yeast genes
datSummary=dat0[,1:7]
# the column essentiality indicates which gene is essential for yeast survival
table(datSummary$essentiality)
# message: there are 645 essential genes

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

datExpr = t(dat0[,8:51])

# This following code allow us to restrict our analysis
```

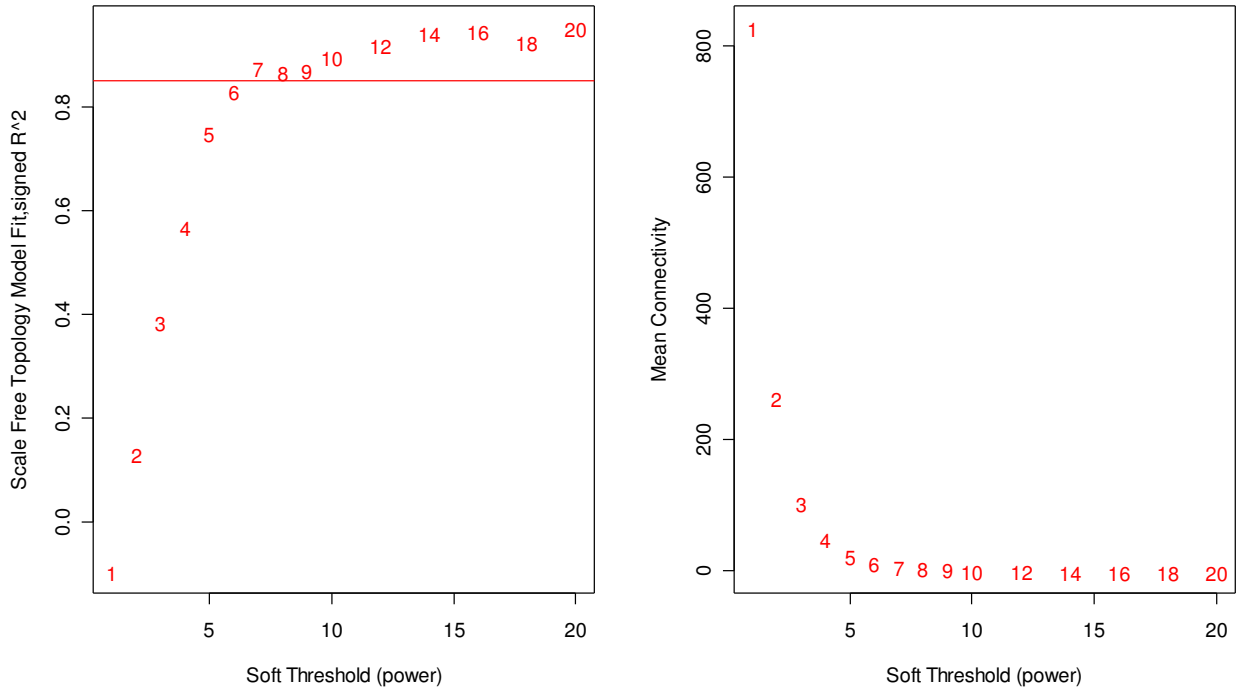
```
# to the most varying genes
var1=function(x) var(x,na.rm=T)
vargenes=apply(datExpr,2,var1)
rankvar=rank(-vargenes)
restVariance= rankvar< 4001 #i.e. we keep all genes.
sum(restVariance)
datExpr=datExpr[,restVariance]
datSummary=datSummary[restVariance,]

no.samples = dim(datExpr)[[1]]
dim(datExpr)
rm(dat0);gc()
```

To choose a cut-off value, we propose to use the Scale-free Topology Criterion (Zhang and
Horvath 2005). Here the focus is on the linear regression model fitting index
(denoted below by scale.law.R.2) that quantify the extent of how well a network
satisfies a scale-free topology.
The function PickSoftThreshold can help one to estimate the cut-off value
when using hard thresholding with the step adjacency function.
The first column (different from the row numbers) lists the soft threshold Power
The second column reports the resulting scale free topology fitting index R^2 (scale.law.R.2)
The third column reports the slope of the fitting line.
The fourth column reports the fitting index for the truncated exponential scale free model.
Usually we ignore it.
The remaining columns list the mean, median and maximum connectivity.
To a soft threshold (power) with the scale free topology criterion:
aim for reasonably high scale free R^2 (column 2), higher than say .80
and negative slope (around -1, col 4).
In practice, we pick the threshold by looking for a "kink" in the
relationship between R^2 and power, see below.

# Now we produce the scale free topology fitting indices for different
# soft thresholds (powers). It will be used to pick a soft threshold.
powers1=c(seq(1,10,by=1),seq(12,20,by=2))
RpowerTable=pickSoftThreshold(datExpr, powerVector=powers1)[[2]]
gc()

Power	scale.law.R.2	slope	truncated.R.2	mean.k.	median.k.	max.k.
1	1	-0.096	-0.981	0.965	826.766	826.977	1309.071
2	2	0.130	-1.706	0.977	263.707	257.186	594.227
3	3	0.391	-1.906	0.989	105.166	98.339	313.387
4	4	0.578	-2.010	0.982	48.521	42.651	181.108
5	5	0.754	-1.864	0.998	24.883	20.178	111.531
6	6	0.834	-1.855	0.996	13.843	10.224	72.876
7	7	0.877	-1.889	0.994	8.218	5.471	52.074
8	8	0.874	-1.923	0.986	5.145	3.113	38.754
9	9	0.878	-1.924	0.984	3.367	1.816	29.504
10	10	0.903	-1.874	0.994	2.288	1.091	22.878
11	12	0.924	-1.742	0.993	1.157	0.418	14.355
12	14	0.946	-1.638	0.996	0.644	0.176	9.822
13	16	0.950	-1.604	0.995	0.386	0.080	7.453
14	18	0.931	-1.584	0.968	0.245	0.037	5.792
15	20	0.956	-1.534	0.982	0.163	0.018	4.658

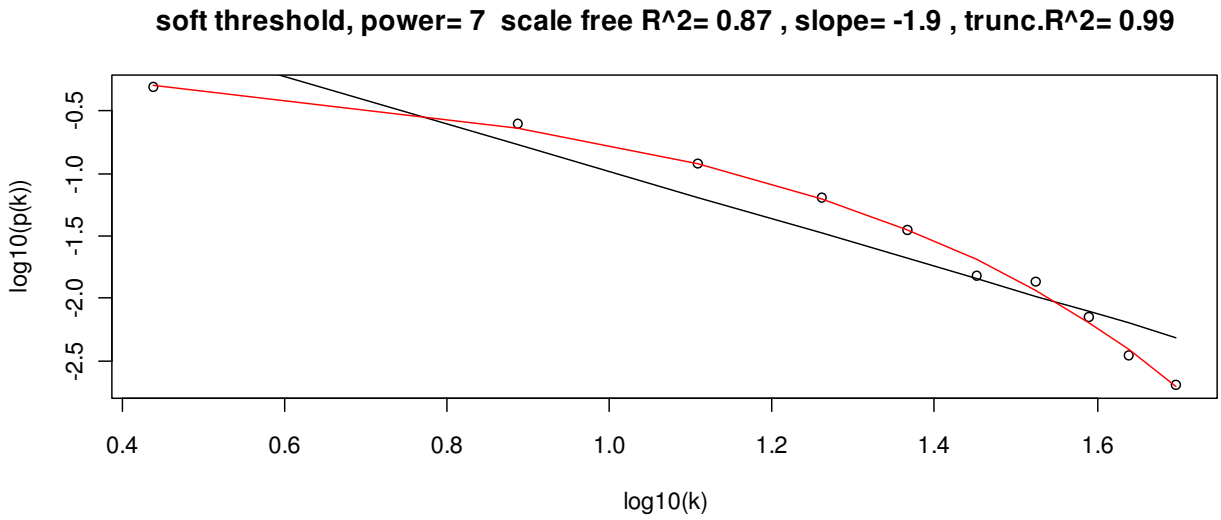
```
gc()
cex1=1
par(mfrow=c(1,2))
plot(RpowerTable[,1], -sign(RpowerTable[,3])*RpowerTable[,2],xlab="
Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n")
text(RpowerTable[,1], -sign(RpowerTable[,3])*RpowerTable[,2],
labels=powers1,cex=cex1,col="red")
# this line corresponds to using an R^2 cut-off of h
abline(h=0.85,col="red")
plot(RpowerTable[,1], RpowerTable[,5],xlab="Soft Threshold (power)",ylab="Mean
Connectivity", type="n")
text(RpowerTable[,1], RpowerTable[,5], labels=powers1, cex=cex1,col="red")
```




```
# Note that at power=7, the curve has an elbow or kink, i.e. for this power the scale free topology
# fit does not improve after increasing the power. This is why we choose beta1=7
# Also the scale free topology criterion with a R^2 threshold of 0.85 would lead us to pick a power
# of 7.
```

Note that there is a natural trade-off between maximizing scale-free topology model fit (R^2) and maintaining a high mean number of connections: parameter values that lead to an R^2 value close to 1 may lead to networks with very few connections. Actually, we consider a signed version of the scale free topology fitting index. Since it is biologically implausible that a networks contains more hub genes than non-hub genes, we multiply R^2 with -1 if the slope of the regression line between log_{10}(p(k)) and log_{10}(k) is positive.

```
# We use the following power for the power adjacency function.
beta1=7
Connectivity=softConnectivity(datExpr,power=beta1)
# Let's create a scale free topology plot.
# The black curve corresponds to scale free topology and
# the red curve corresponds to truncated scale free topology.
par(mfrow=c(1,1))
scaleFreePlot(Connectivity, main=paste("soft threshold, power=",beta1), truncated=T);
```



#Module Detection
An important step in network analysis is module detetion.
Here we use methods that use clustering in combination with the topological
overlap matrix, see Yip and Horvath (2007) and Ravasz et al 2001.

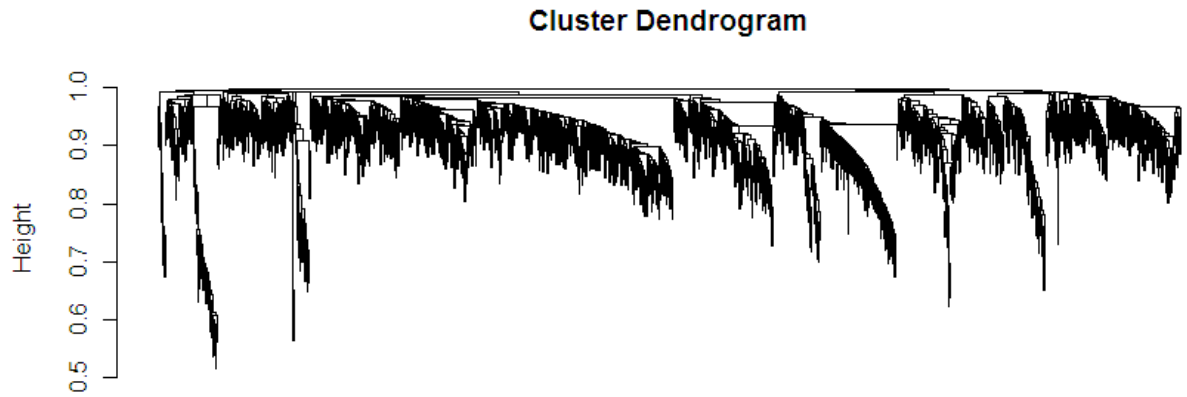
```
# This code allows one to restrict the analysis to the most connected genes,
# which may speed up calculations when it comes to module detection.
ConnectivityCut = 2001 # number of most connected genes that will be considered
ConnectivityRank = rank(-Connectivity)
restConnectivity = ConnectivityRank <= ConnectivityCut
# thus our module detection uses the following number of genes
sum(restConnectivity)
```

Now we restrict the adjacency matrix to the most connected genes

```
ADJrest = adjacency(datExpr[,restConnectivity], power=beta1)

# The following code computes the topological overlap matrix based on the
# adjacency matrix.
dissTOM=TOMdist(ADJrest)
gc()
```

Now we carry out hierarchical clustering with the TOM matrix. Branches of the
resulting clustering tree will be used to define gene modules.

```
hierTOM = hclust(as.dist(dissTOM),method="average");
par(mfrow=c(1,1))
plot(hierTOM,labels=F,sub="",xlab="")
```



According to our definition, modules correspond to branches of the tree.
The question is what height cut-off should be used? This depends on the
biology. Large heigth values lead to big modules, small values lead to small
but tight modules.
In reality, the user should use different thresholds to see how robust the findings are.

The function cutreeStaticColor colors each gene by the branches that
result from choosing a particular height cut-off.

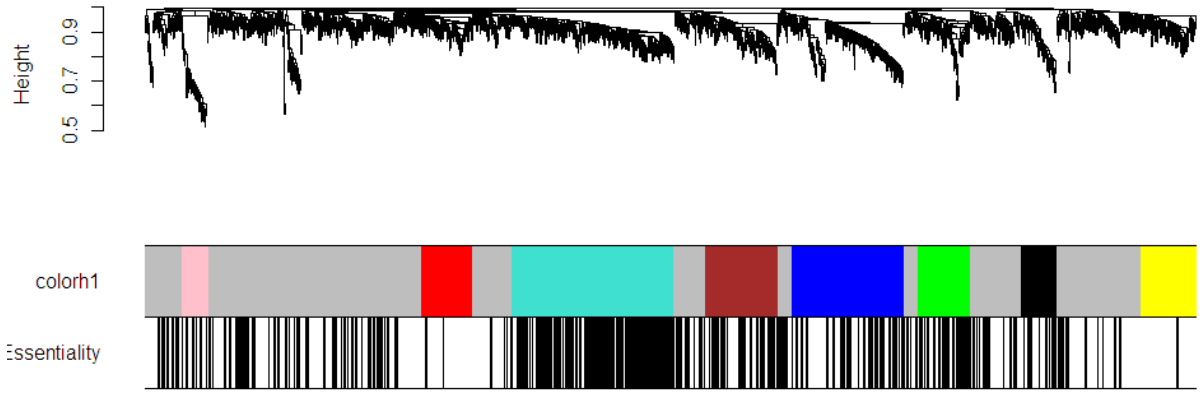
GREY IS RESERVED to color genes that are not part of any module.
We only consider modules that contain at least 125 genes.

colorh1= cutreeStaticColor(hierTOM,cutHeight = 0.97)

```
table(colorh1)
```

```
colorh1
    black      blue     brown     green      grey    purple       red turquoise    yellow
       67       215       139        98       920        53        96       308       105
```

#LET US NOW DEFINE THE "GENE SIGNIFICANCE",
which indicates whether or not the gene is essential for yeast survival based on knock-out
#experiments.
GeneSignificance=datSummary$essentiality[restConnectivity]
par(mfrow=c(2,1), mar=c(2,4,2,2))
plot(hierTOM, main="", labels=F, xlab="", sub="");
plotColorUnderTree(hierTOM,colors=data.frame(colorh1,Essentiality=
GeneSignificance))

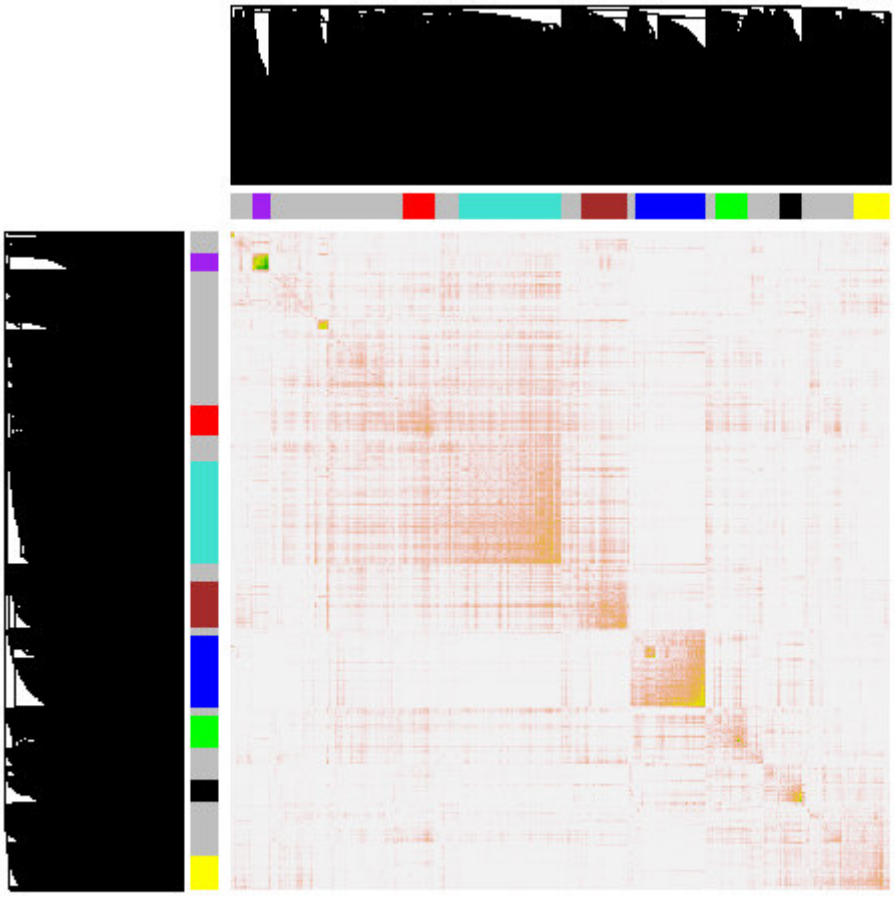


COMMENTS:

1) The colors are assigned based on module size. Turquoise (others refer to it as cyan) colors the largest module, next comes blue, next brown, etc. Just type table(colorh1) to figure out which color corresponds to what module size.
2) The color band called essentiality indicates whether a gene is known to be essential (black) or not (white).
3) The minimum module size is determine by the input parameter minSize of cutreeStaticColors. The default size is minSize=50.
4) Here we choose a fixed height cut-off for cutting off branches. But we have also developed a more flexible method for cutting off branches that adaptively choose a different height for each branch. The resulting dynamic tree cutting algorithm (cutreeDynamic) is desccribed in Langfelder Zhang and Horvath (2008).

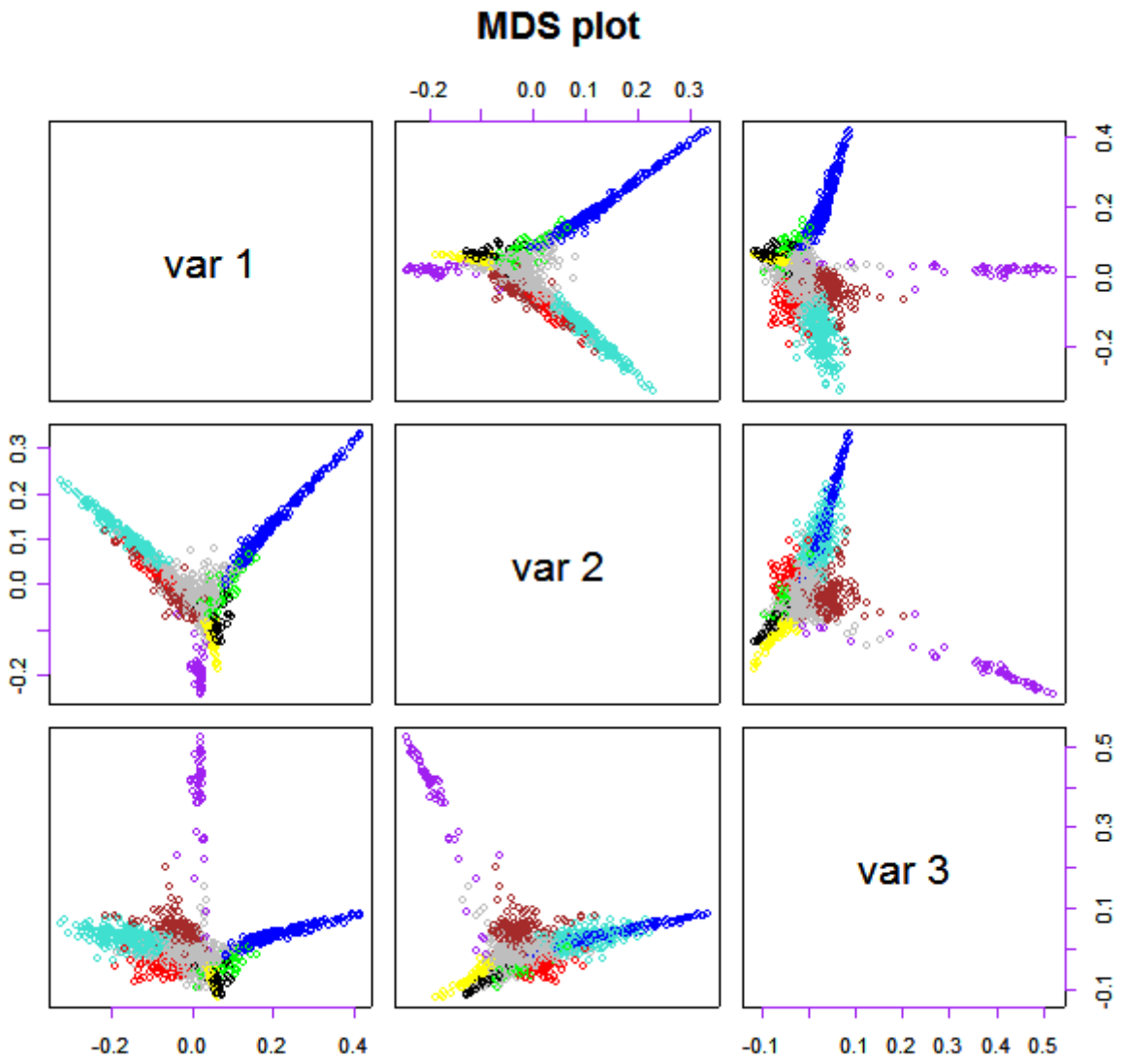
An alternative view of this is the so called TOM plot that is generated by the
function TOMplot
Inputs: TOM distance measure, hierarchical (hclust) object, color

```
# Warning: for large gene sets, this may take a while...
TOMplot(dissTOM , hierTOM, colorh1, terrainColors=TRUE)
```

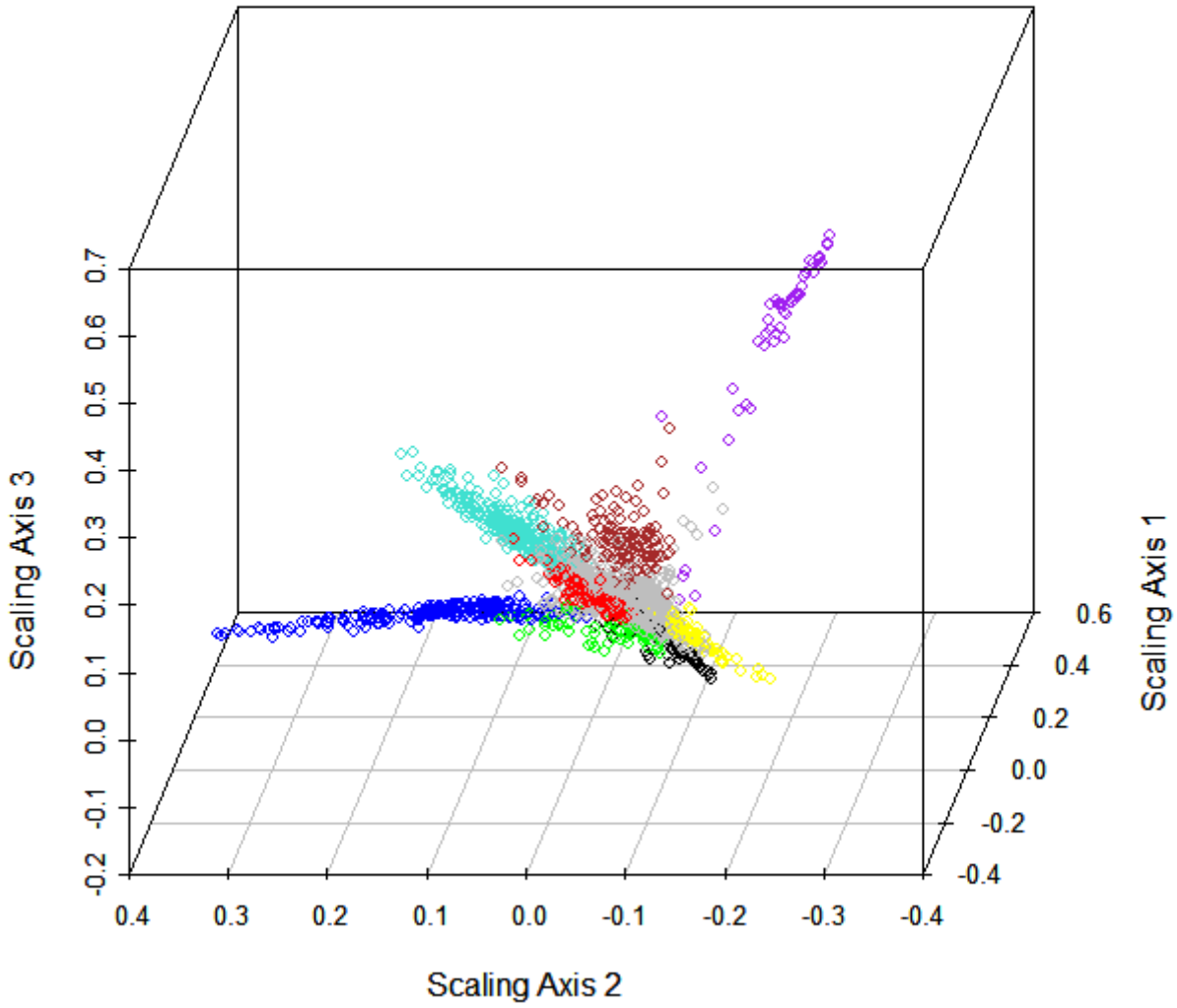


#In the above, note that rows and columns are genes colored by module membership.
#Note that modules correspond to squares along the diagonal.

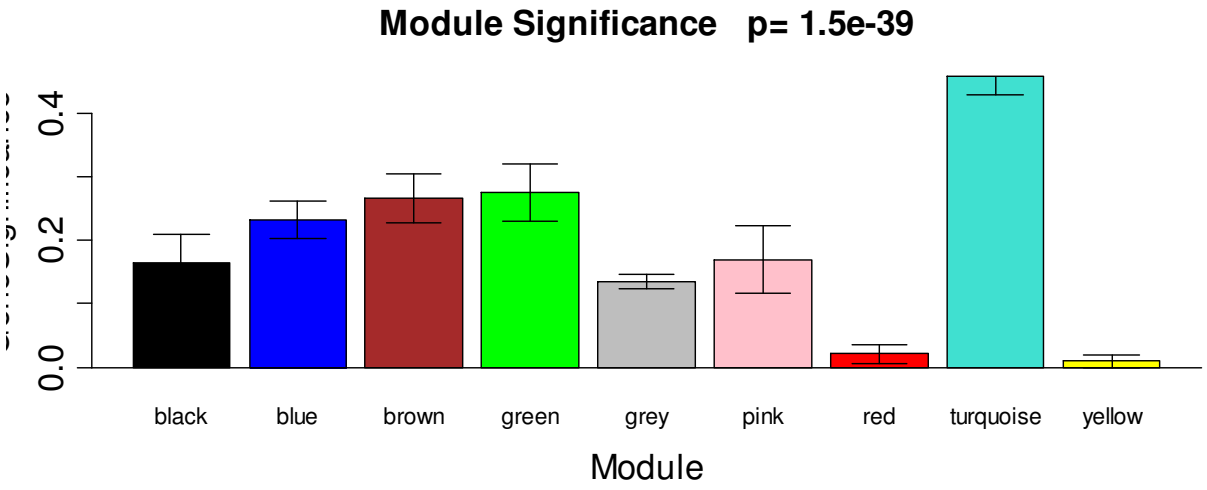
We also propose to use classical multi-dimensional scaling plots for visualizing the network.
#Here we chose 3 scaling dimensions
cmd1=cmdscale(as.dist(dissTOM),3)
pairs(cmd1, col=as.character(colorh1), main="MDS plot")


MDS plot
var 1
var 2
var 3
-0.2
0.0
0.1
0.2
0.3
0.4
-0.2
0.0
0.2
0.4
-0.1
0.1
0.2
0.3
0.4
0.5
-0.1
0.1
0.3
0.5

```
# Here is a 3 D plot of the same scaling coordinates.
par(mfrow=c(1,1), mar=c(4,3,2,3)+0.1)
library(scatterplot3d)
scatterplot3d(cmd1,color=colorh1,angle=250,
xlab="Scaling Axis 1", ylab="Scaling Axis 2", zlab="Scaling Axis 3")
```



# The function verboseBarplot creates a bar plot
# that shows whether modules are enriched with essential genes.
# It also reports a Kruskal Wallis P-value.
# The gene significance can be a binary variable or a quantitative variable.
# also plots the 95% confidence interval of the mean
par(mfrow=c(1,1))
verboseBarplot(GeneSignificance,colorh1,main="Module Significance ",
col=levels(factor(colorh1)) ,xlab="Module")



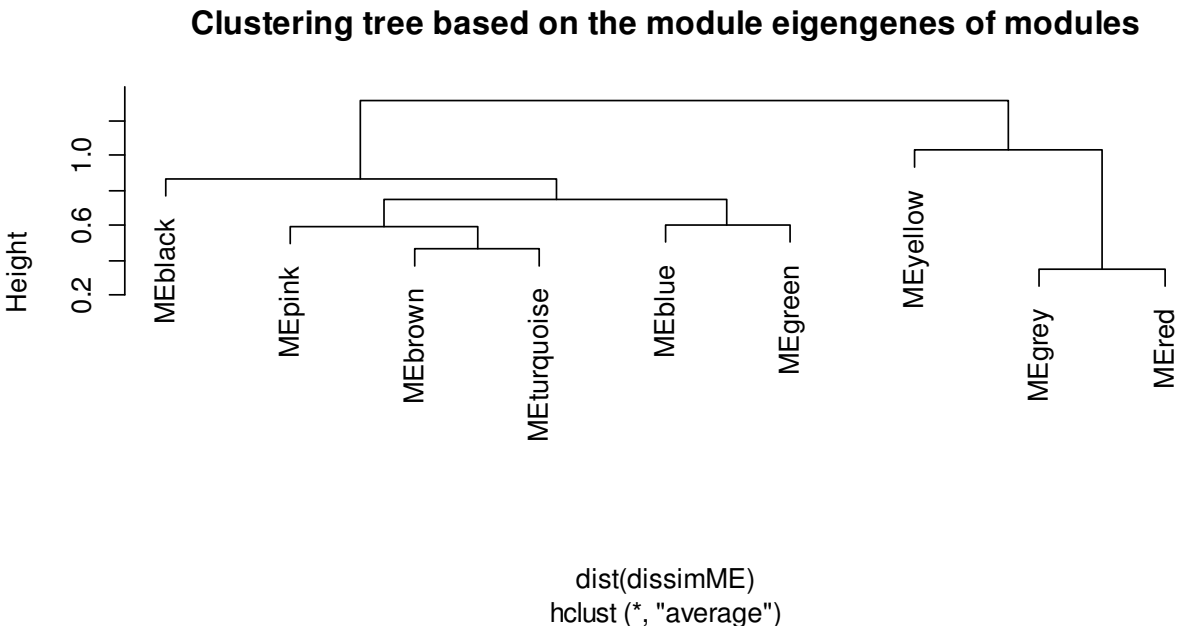
Note that one module is highly enriched with essential genes.

To get a sense of how related the modules are one can summarize each module
by its first eigengene (referred to as principal components).
and then correlate these module eigengenes with each other.

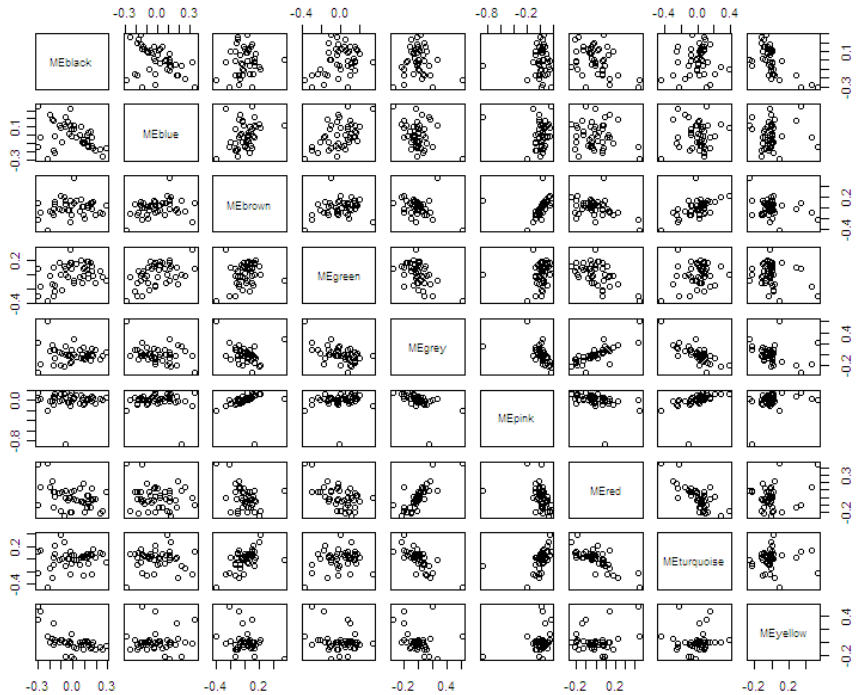
datME=moduleEigengenes(datExpr[,restConnectivity],colorh1)[[1]]

# We define a dissimilarity measure between the module eigengenes that keeps track of the sign of the correlation between the module eigengenes.
dissimME=1-(t(cor(datME, method="p")))/2

hclustdatME=hclust(dist(dissimME), method="average")
par(mfrow=c(1,1))
plot(hclustdatME, main="Clustering tree based on the module eigengenes of modules")



# Now we create scatter plots of the samples (arrays) along the module eigengenes.
pairs(datME)



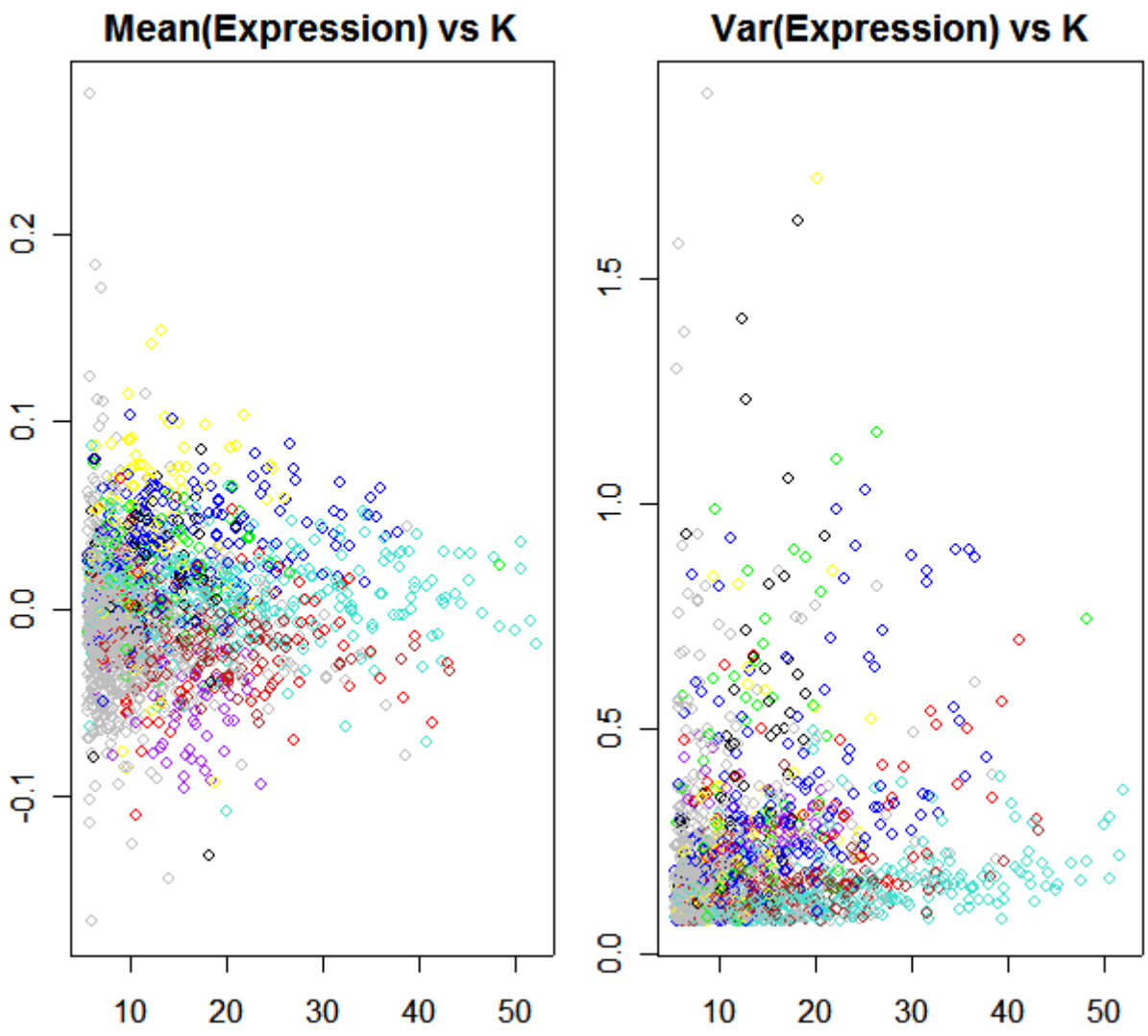
Compare this pairwise plot with the following correlation table between eigengenes.
signif(cor(datME, use="p"), 2)

	MEblack	MEblue	MEbrown	MEgreen	MEgrey	MEpink	MEred	MEturquoise	MEyellow
MEblack	1.000	-0.390	0.12	0.36	-0.056	0.042	-0.42	0.130	-0.610
MEblue	-0.390	1.000	0.30	0.50	-0.340	-0.085	-0.15	0.036	0.045
MEbrown	0.120	0.300	1.00	0.23	-0.330	0.220	-0.41	0.590	-0.300
MEgreen	0.360	0.500	0.23	1.00	-0.520	0.140	-0.58	0.230	-0.240
MEgrey	-0.056	-0.340	-0.33	-0.52	1.000	-0.470	0.80	-0.770	-0.240
MEpink	0.042	-0.085	0.22	0.14	-0.470	1.000	-0.34	0.440	0.280
MEred	-0.420	-0.150	-0.41	-0.58	0.800	-0.340	1.00	-0.750	0.220
MEturquoise	0.130	0.036	0.59	0.23	-0.770	0.440	-0.75	1.000	0.026
MEyellow	-0.610	0.045	-0.30	-0.24	-0.240	0.280	0.22	0.026	1.000

Message: the module eigengenes (first PC) of different modules may be highly correlated. WGCNA can be interpreted as a biologically motivated data reduction scheme that allows for dependency between the resulting components. Compare this to principal component analysis that would impose orthogonality between the components.
Since modules may represent biological pathways there is no biological reason why modules should be orthogonal to each other.
Aside: If you are interested in networks comprised of module eigengenes, the following article may be of interest:
Langfelder P, Horvath S (2007) Eigengene networks for studying the relationships between co-expression modules. BMC Systems Biology 2007, 1:54

#To study how connectivity is related to mean gene expression or variance of gene expression

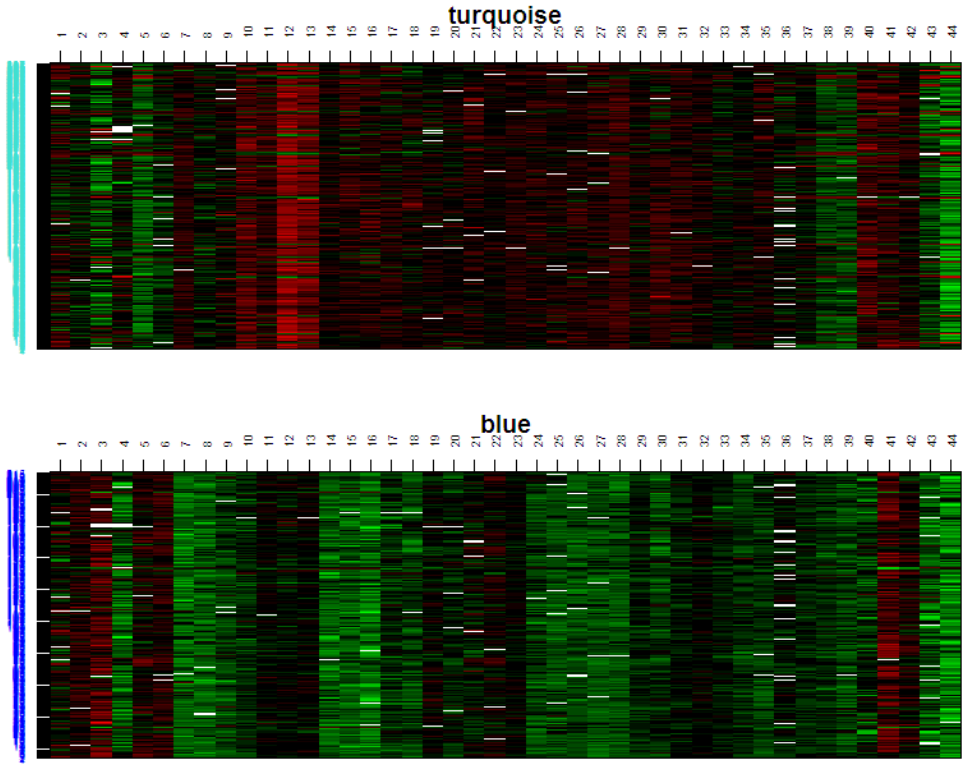
```
# we create the following plot.
mean1=function(x) mean(x,na.rm=T)
var1=function(x) var(x,na.rm=T)
meanExpr=apply( datExpr[,restConnectivity],2,mean1)
varExpr=apply( datExpr[,restConnectivity],2,var1)
par(mfrow=c(1,2))
plot(Connectivity[restConnectivity],meanExpr, col=colorh1,
main="Mean(Expression) vs K")
plot (Connectivity[restConnectivity],varExpr, col=colorh1, main="Var(Expression)
vs K")
```


Mean(Expression) vs K
Var(Expression) vs K
0.2
0.1
0.0
-0.1
1.5
1.0
0.5
0.0
10
20
30
40
50

```
# The following produces heatmap plots for each module.
# Here the rows are genes and the columns are samples.
# Well defined modules results in characteristic band structures since the corresponding genes are
# highly correlated.
```


```
# The following produces heatmap plots for each module.
# Here the rows are genes and the columns are samples.
# Well defined modules results in characteristic band structures since the corresponding genes are
# highly correlated.
```


```
par(mfrow=c(2,1), mar=c(1, 2, 4, 1))
ClusterSamples=hclust(dist(datExpr[,restConnectivity] ),method="average")
# for the first (turquoise) module we use
which.module="turquoise"
plot.mat(t(scale(datExpr[ClusterSamples$order,restConnectivity][,colorh1==which.module ])
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
main=which.module )
# for the second (blue) module we use
which.module="blue"
plot.mat(t(scale(datExpr[ClusterSamples$order,restConnectivity][,colorh1==which.module ])
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
main=which.module )
```




```
#For each module there should be a clear band structure, i.e. genes
#corresponding to a given array (column) should have the same color.
# The white bard represent missing values.
```

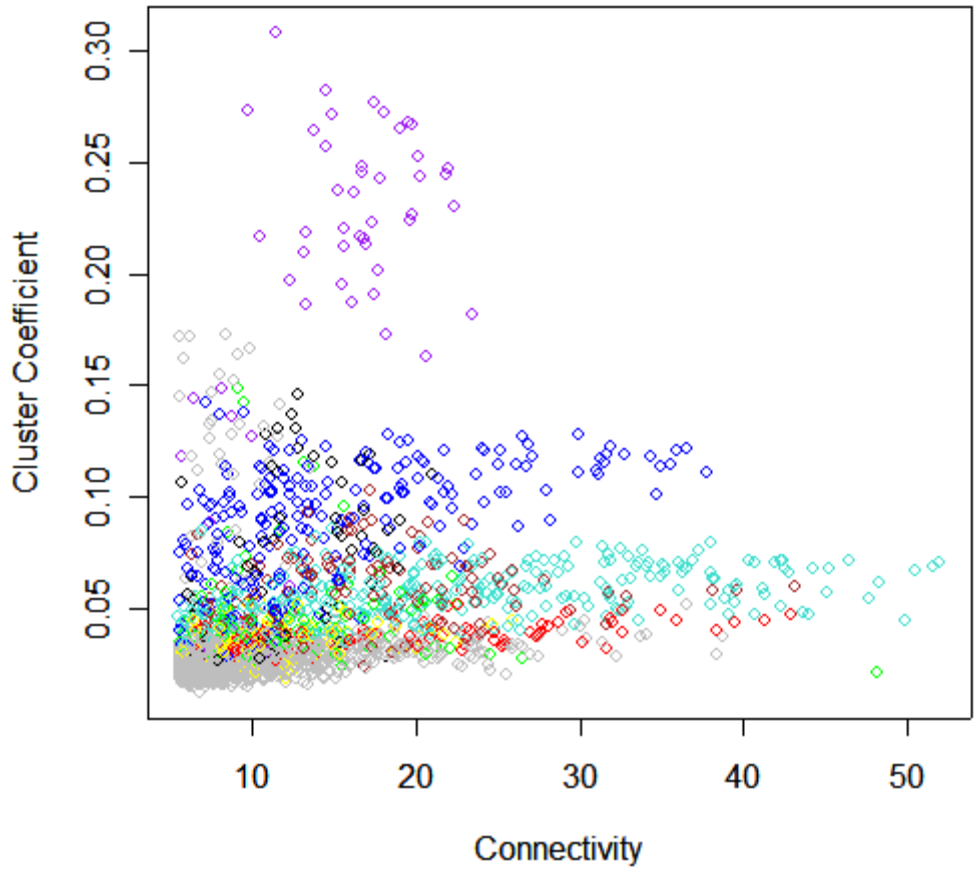
#Now we extend the color definition to all genes by coloring all non-module
genes grey.

```
color1=rep("grey",sum(restVariance))
color1[restConnectivity]=as.character(colorh1)
```

#Computation of the cluster coefficient

The cluster coefficient measures the cliquishness of a gene,
consult Zhang and Horvath, Horvath et al (2005)

```
CC= clusterCoef(ADJrest)

gc()

# Now we plot cluster coefficient versus connectivity
# for all genes
par(mfrow=c(1,1))
plot(Connectivity[restConnectivity],CC,col=as.character(colorh1),
,xlab="Connectivity",ylab="Cluster Coefficient" )
```



This compute the correlation between cluster coefficient and connectivity within each module.

```
by(data.frame(CC=CC, k=Connectivity[restConnectivity]), INDICES=colorh1,FUN=cor)

colorh1: black
          CC         k
CC 1.0000000 0.4384517
k  0.4384517 1.0000000
------------------------------------------------------------------------
colorh1: blue
          CC         k
CC 1.0000000 0.5432612
k  0.5432612 1.0000000
------------------------------------------------------------------------
colorh1: brown
          CC         k
CC 1.0000000 0.2301026
k  0.2301026 1.0000000
------------------------------------------------------------------------
colorh1: green
            CC           k
CC  1.00000000 -0.04967428
k  -0.04967428  1.00000000
------------------------------------------------------------------------
colorh1: grey
           CC          k
CC 1.00000000 0.05520535
k  0.05520535 1.00000000
------------------------------------------------------------------------
colorh1: purple
          CC         k
CC 1.0000000 0.6078187
k  0.6078187 1.0000000
------------------------------------------------------------------------
colorh1: red
          CC         k
CC 1.0000000 0.5742447
k  0.5742447 1.0000000
------------------------------------------------------------------------
colorh1: turquoise
          CC         k
CC 1.0000000 0.4474475
k  0.4474475 1.0000000
------------------------------------------------------------------------
colorh1: yellow
          CC         k
CC 1.0000000 0.2852322
k  0.2852322 1.0000000
```

Comment: In most modules, we find a positive correlation between k and CC.

Constructing an unweighted network with HARD THRESHOLDING

Based on the expression data, the absolute pair-wise (Pearson) correlation coefficient between the expression profiles of each pair of genes is calculated. Then, a network with each node representing one gene is constructed. An edge between two nodes is present if their absolute correlation coefficient exceeds a threshold tau=0.7. We obtain the threshold tau by using the scale-free criterion proposed by Zhang and Horvath (2005).

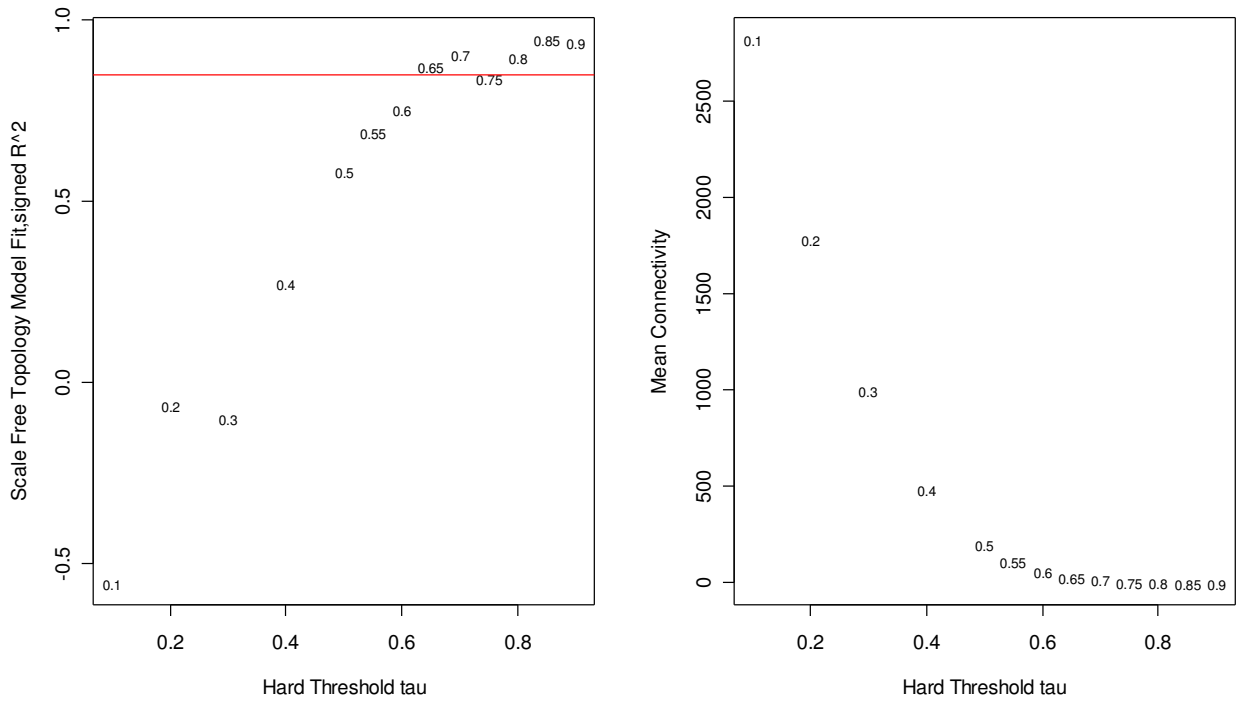
```
# To construct an unweighted network (hard thresholding),
# we consider the following vector of potential thresholds.

thresholds1= c(seq(.1,.5, by=.1), seq(.55,.9, by=.05) )

# To choose a cut-off value, we propose to use the Scale-free Topology Criterion (Zhang and
# Horvath 2005). Here the focus is on the linear regression model fitting index
# (denoted below by scale.law.R.2) that quantify the extent of how well a network
# satisfies a scale-free topology.
# The function PickHardThreshold can help one to estimate the cut-off value
# when using hard thresholding with the step adjacency function.
# The first column lists the threshold ("cut"),
# the second column lists the corresponding p-value based on the Fisher transform.
# The third column reports the resulting scale free topology fitting index R^2.
# The fourth column reports the slope of the fitting line.
# The fifth column reports the fitting index for the truncated exponential scale free model.
# Usually we ignore it.
# The remaining columns list the mean, median and maximum connectivity.
# To pick a hard threshold (cut) with the scale free topology criterion:
# aim for high scale free R^2 (column 3), high connectivity (col 6)
# and negative slope (around -1, col 4).

RdichotTable=pickHardThreshold(datExpr, cutVector= thresholds1)[[2]]
```

	Cut	p.value	scale.law.R.2	slope.	truncated.R.2	mean.k.	median.k.	max.k.
1	0.10	5.13e-01	0.5520	8.05	0.824	2820.000	2850	3340
2	0.20	1.88e-01	0.0616	1.62	0.705	1780.000	1820	2720
3	0.30	4.53e-02	-0.0980	-0.42	0.809	1000.000	1010	2090
4	0.40	6.48e-03	0.2740	-1.44	0.901	486.000	463	1440
5	0.50	4.70e-04	0.5850	-1.90	0.944	199.000	160	898
6	0.55	9.09e-05	0.6910	-1.86	0.962	117.000	83	663
7	0.60	1.32e-05	0.7560	-1.79	0.958	65.500	37	466
8	0.65	1.35e-06	0.8760	-1.62	0.988	34.500	14	314
9	0.70	8.73e-08	0.9060	-1.52	0.993	16.600	4	196
10	0.75	3.03e-09	0.8410	-1.56	0.968	7.250	1	123
11	0.80	4.31e-11	0.8980	-1.36	0.982	2.680	0	64
12	0.85	1.51e-13	0.9470	-1.23	0.961	0.797	0	32
13	0.90	0.00e+00	0.9420	-1.25	0.964	0.146	0	17

```
# Comment: Note that for a cut-off value (tau)=0.10 the scaling law R^2 equals 0.55, which seems
#to be pretty high. However, the slope is positive, i.e. this would predict there are more genes with
#high connectivity than there are genes with low connectivity. This is unbiological!
```

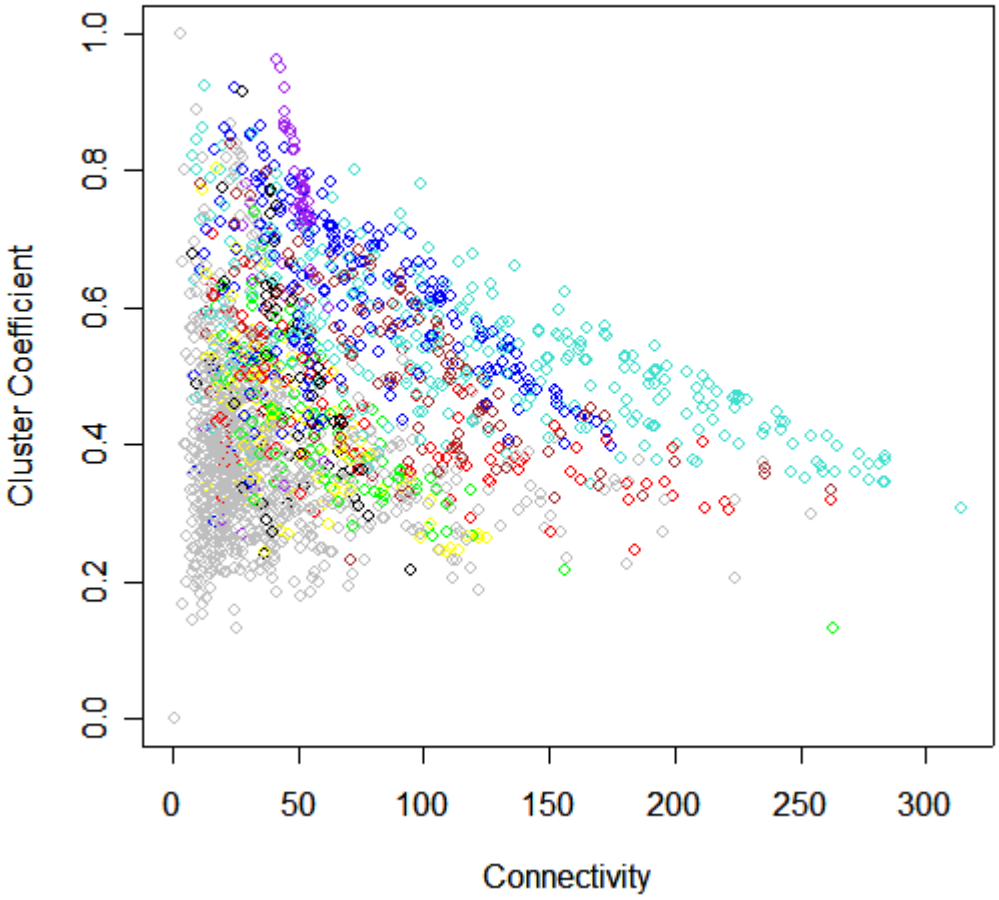
This is the reason why we look at the signed R^2 value:
#Signed R^2=`-sign(RdichotTable[,4])*RdichotTable[,3]`
#Let's plot the scale free topology model fitting index (R^2) versus the cut-off tau. However, the
#R^2 values of those cut-offs that lead to a negative slope have been pre-multiplied by -1.

```
cex1=0.7
gc()
par(mfrow=c(1,2))
plot(RdichotTable[,1], -sign(RdichotTable[,4])*RdichotTable[,3],xlab="Hard
Threshold tau",ylab="Scale Free Topology Model Fit,signed R^2", type="n")
text(RdichotTable[,1], -sign(RdichotTable[,4])*RdichotTable[,3] ,
labels=thresholds1,cex=cex1)
# this line corresponds to using an R^2 cut-off of h
abline(h=0.85,col="red")
plot(RdichotTable[,1], RdichotTable[,6],xlab="Hard Threshold tau",ylab="Mean
Connectivity", type="n")
text(RdichotTable[,1], RdichotTable[,6] , labels=thresholds1, cex=cex1)
```



Based on this analysis, we would choose the cut value (tau) of 0.65 for the correlation matrix
#since a) the scale law R^2 crosses 0.85, b) the slope looks OK (negative and around -1), and
c) the mean number of connections is reasonably high.
Later we discuss how robust our findings are with respect to this choice.

```
AdjMatHARD=abs(corhelp[restConnectivity,restConnectivity])>0.65+0.0
diag(AdjMatHARD)=0
cluster.coefrestHARD= ClusterCoef.fun(AdjMatHARD)
ConnectivityHARD= apply(AdjMatHARD,2,sum)
par(mfrow=c(1,1))
plot(ConnectivityHARD,cluster.coefrestHARD,col=as.character(colorh1),xlab="Conne
ctivity",ylab="Cluster Coefficient" )
```



Now we correlate the cluster coefficient with connectivity by module in the unweighted network

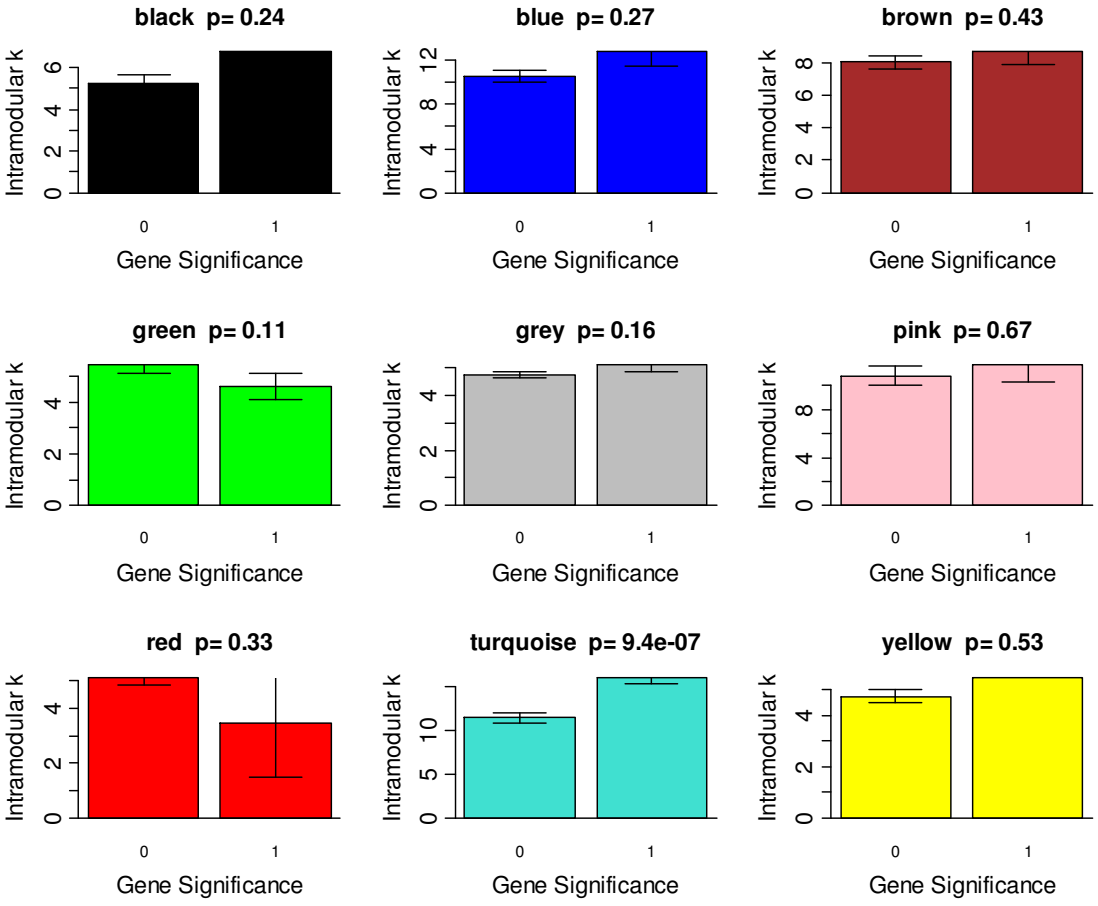
```
restHub=ConnectivityHARD>50
by(data.frame(CC= cluster.coefrestHARD[restHub], k=ConnectivityHARD[restHub]),
INDICES=colorh1[restHub],FUN=cor)
```

```
colorh1[restHub]: black
           CC          k
CC  1.0000000 -0.7565848
k  -0.7565848  1.0000000
------------------------------------------------------------
colorh1[restHub]: blue
           CC          k
CC  1.0000000 -0.5948222
k  -0.5948222  1.0000000
------------------------------------------------------------
colorh1[restHub]: brown
           CC          k
CC  1.0000000 -0.5767991
k  -0.5767991  1.0000000
------------------------------------------------------------
colorh1[restHub]: green
          CC         k
CC  1.000000 -0.736881
k  -0.736881  1.000000
------------------------------------------------------------
colorh1[restHub]: grey
           CC          k
CC  1.0000000 -0.1226995
k  -0.1226995  1.0000000
------------------------------------------------------------
colorh1[restHub]: purple
           CC          k
CC  1.0000000 -0.7748994
k  -0.7748994  1.0000000
------------------------------------------------------------
colorh1[restHub]: red
           CC          k
CC  1.0000000 -0.5555478
k  -0.5555478  1.0000000
------------------------------------------------------------
colorh1[restHub]: turquoise
           CC          k
CC  1.0000000 -0.6737847
k  -0.6737847  1.0000000
------------------------------------------------------------
colorh1[restHub]: yellow
          CC         k
CC  1.000000 -0.790202
k  -0.790202  1.000000
```

Comments: for unweighted networks there is an inverse relationship between cluster coefficient and connectivity. This is can be quite different for weighted networks Zhang and Horvath (2005).

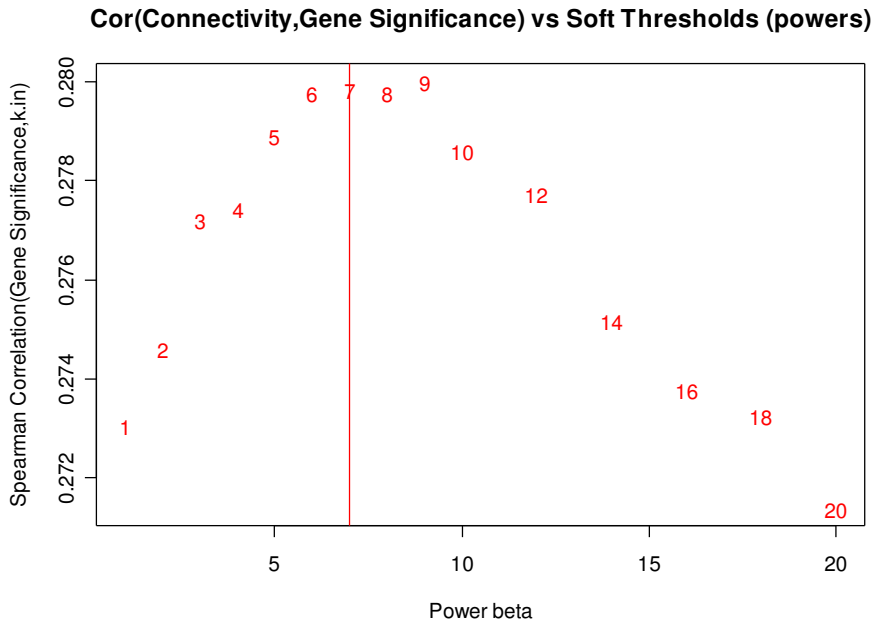
The function intramodularConnectivity computes the whole network connectivity kTotal,
the within module connectivity (kWithin). kOut=kTotal-kWithin and
and kDiff=kIn-kOut=2*kIN-kTotal

```
ConnectivityMeasures=intramodularConnectivity(ADJrest,colorh1)

names(ConnectivityMeasures)
[1] "kTotal"   "kWithin" "kOut"    "kDiff"

# The following plots who the gene significance (essentiality)
# vs intramodular connectivity


colorlevels=levels(factor(colorh1))
par(mfrow=c(3,3),mar=c(5, 4, 4, 2) + 0.1)
for (i in c(1:length(colorlevels) ) ) {
whichmodule=colorlevels[[i]];restrict1=colorh1==whichmodule
verboseBarplot(ConnectivityMeasures$kWithin[restrict1],
GeneSignificance[restrict1],col=colorh1[restrict1],main=
paste(whichmodule),xlab="Gene Significance",ylab="Intramodular k")
}
```



Now we want to see how the correlation between kWithin and gene essentiality changes for #different SOFT thresholds (powers). Restricted to turquoise module genes.
Also we compare the 2 different connectivity measures.

```
par(mfrow=c(1,1))
datconnectivitiesSoft=data.frame(matrix(666,nrow=sum(colorh1==whichmodule),ncol=
length(powers1)))
names(datconnectivitiesSoft)=paste("kWithinPower",powers1,sep="")
for (i in c(1:length(powers1)) ) {
datconnectivitiesSoft[,i]=apply(abs(corhelp[colorh1==whichmodule,
colorh1==whichmodule])^powers1[i],1,sum)}
SpearmanCorrelationsSoft=signif(cor(GeneSignificance[ colorh1==whichmodule],
datconnectivitiesSoft, method="s",use="p"))

plot(powers1, SpearmanCorrelationsSoft, main="Cor(Connectivity,Gene
Significance) vs Soft Thresholds (powers)",ylab="Spearman Correlation(Gene
Significance,k.in)",xlab="Power
beta",type="n",ylim=range(c(SpearmanCorrelationsSoft),na.rm=T)
)
text(powers1, SpearmanCorrelationsSoft,labels=powers1,col="red")
# this draws a vertical line at the tau that was chosen by the
# scale free topology criterion.
abline(v=7,col="red")
```



#Comment: Very high and very low values of the soft threshold lead seem to decrease the #biological signal. It is remarkable that the scale free topology criterion picked a good power.

Comments/Results

- We find that soft thresholding leads to results that are far more robust with respect to the choice of the adjacency function parameter.

- We find that soft thresholding is superior to hard thresholding especially for low values of the scale free topology R^2.
- In our opinion, soft centrality (connectivity) measures are better than hard measures because they are relatively robust with respect to the parameter of the adjacency function. For soft thresholding even choosing a power of beta=1 leads to a fairly good correlation. In contrast, choosing a hard threshold of tau=0.2 leads to a much reduced biological signal. Robustness is a very attractive property in this type of analysis since picking parameters of the adjacency function is rather ad-hoc.
- The Scale free topology criterion leads to estimates of the adjacency function that often have good biological signal.

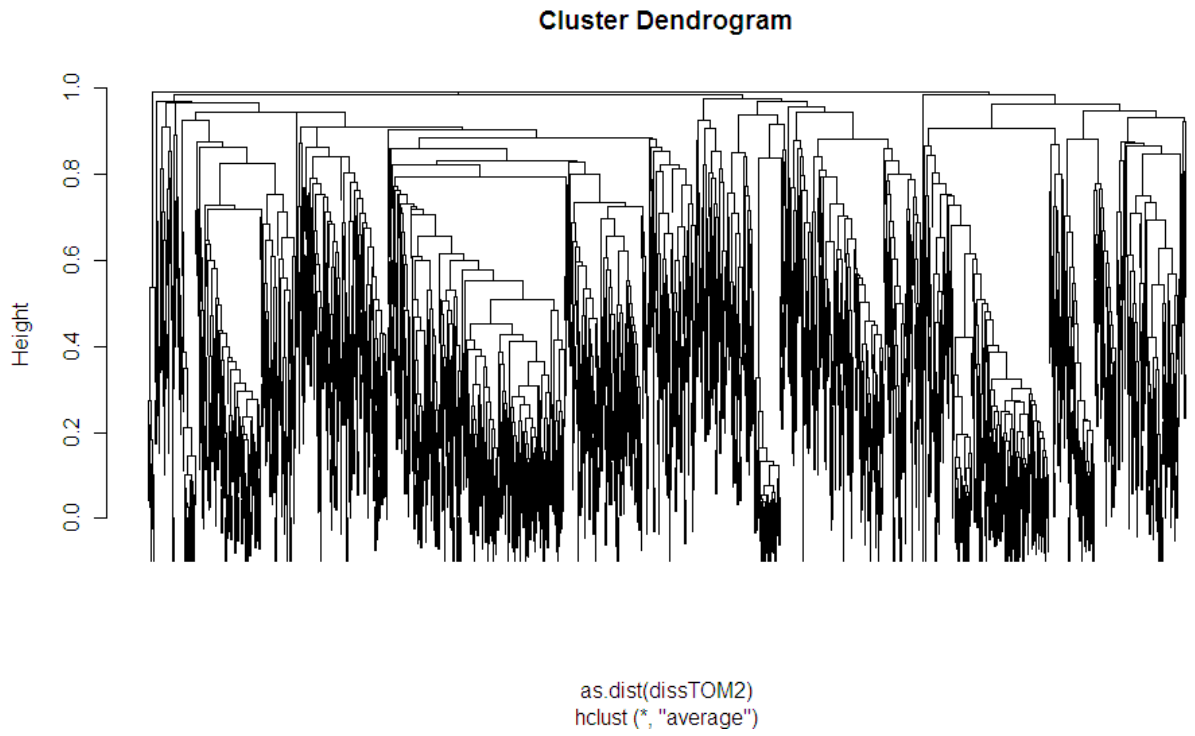
APPENDIX: HARD THRESHOLDING and Unweighted Networks

Now we carry out hard thresholding using tau=0.65

```
ADJhardrest = I(abs(cor(datExpr[,restConnectivity], use="p"))>0.65)+0.0

# The following code computes the topological overlap matrix based on the
# adjacency matrix.
dissTOM2=TOMdist(ADJhardrest)
gc()
```

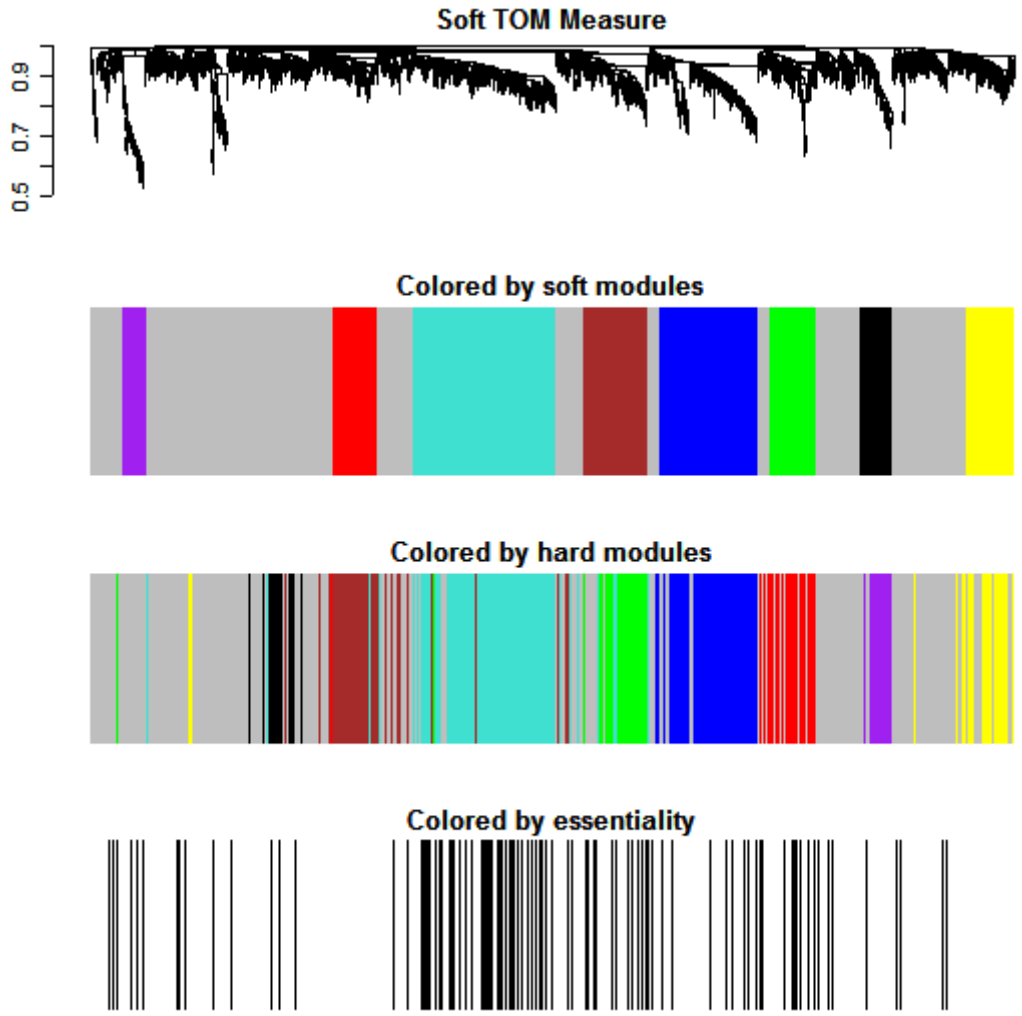
Now we carry out hierarchical clustering with the TOM matrix. Branches of the
resulting clustering tree will be used to define gene modules.

```
hierGTOM2 = hclust(as.dist(dissTOM2),method="average");
par(mfrow=c(1,1))
plot(hierGTOM2,labels=F)
```



By our definition, modules correspond to branches of the tree.

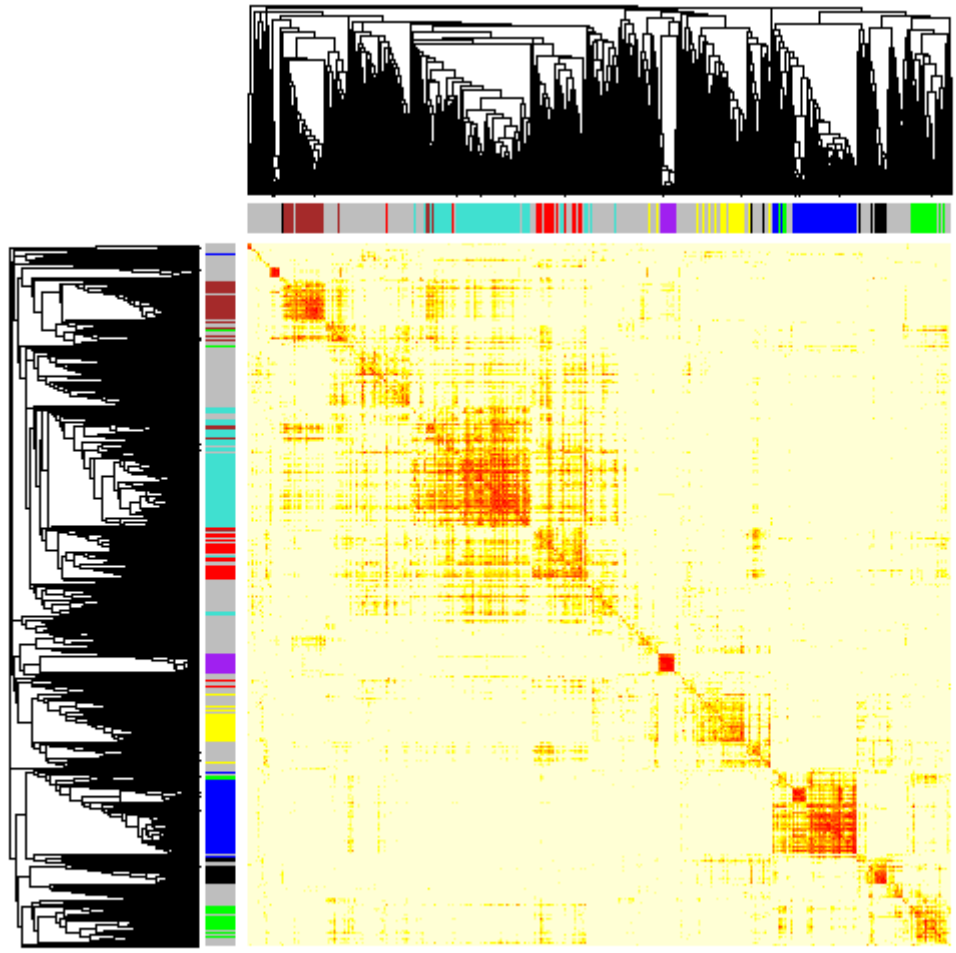
```
# The function modulecolor2 colors each gene by the branches that
# result from choosing a particular height cut-off.
# GREY IS RESERVED to color genes that are not part of any module.
colorh2= cutreeStaticColor(hierGTOM2, cutHeight=.75)

table(colorh2,colorh1)

par(mfrow=c(4,1), mar=c(2,2,2,1))
plot(hierTOM, main="Soft TOM Measure", labels=F, xlab="", sub="");
plotColorUnderTree(hierTOM,colors=colorh1)
title("Colored by soft modules")
plotColorUnderTree(hierTOM,colorh2)
title("Colored by hard modules")
plotColorUnderTree(hierTOM,GeneSignificance)
title("Colored by essentiality")
```



In the above, the first plot is the tree and the second plot shows the
corresponding branch colors.
Message: module assignment is highly preserved between hard and sof thresholding.
The third row indicates which genes are essential (black).

An alternative view of this is the so called TOM plot that is generated by the
function TOMplot
Inputs: TOM distance measure, hierarchical (hclust) object, color

```
# Warning: for large gene sets, say more than 2000 genes this may take a while...
TOMplot(dissTOM2 , hierGTOM2, colorh1)
```



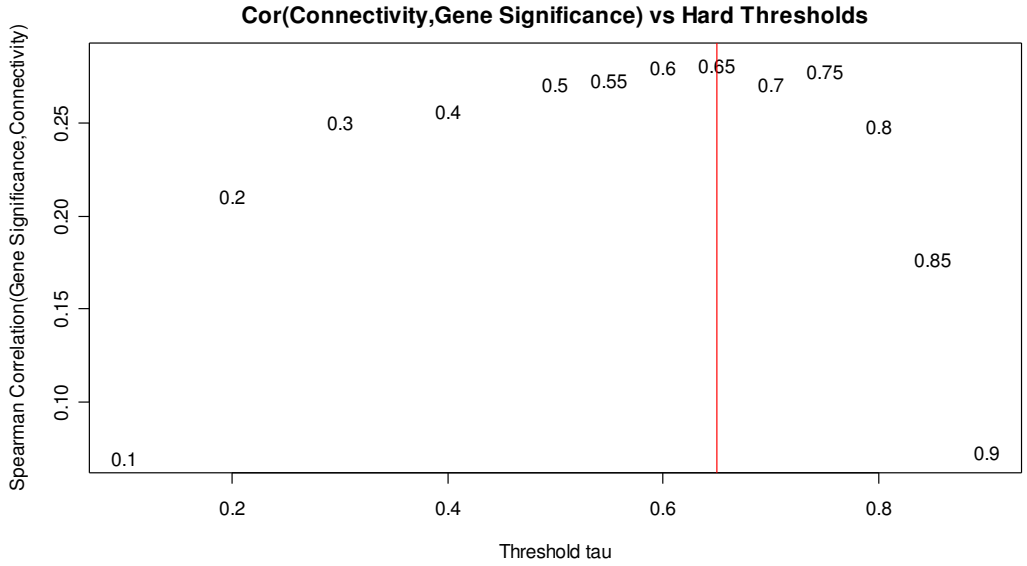
Comment: modules are highly preserved between soft and hard thresholding.

Now we want to see how the correlation between kWithin and essentiality
changes for different hard threshold values tau within the turquoise module.

```
corhelp=cor(datExpr[,restConnectivity],use="pairwise.complete.obs")
whichmodule="turquoise"
# the following data frame contains the intramodular connectivities
# corresponding to different hard thresholds
datconnectivitiesHard=data.frame(matrix(666,nrow=sum(colorh1==whichmodule),ncol=
length(thresholds1)))
names(datconnectivitiesHard)=paste("kWithinTau",thresholds1,sep="")
for (i in c(1:length(thresholds1)) ) {
datconnectivitiesHard[,i]=apply(abs(corhelp[colorh1==whichmodule,
colorh1==whichmodule])>=thresholds1[i],1,sum)}
SpearmanCorrelationsHard=signif(cor(GeneSignificance[ colorh1==whichmodule],
datconnectivitiesHard, method="s",use="p"))
```

Now we compare the performance of the connectivity measures k.in across different hard #thresholds when it comes to predicting prognostic genes in the turquoise module

```
par(mfrow=c(1,1), mar=c(5, 4, 4, 2) +0.1)
plot(thresholds1, SpearmanCorrelationsHard, main="
Cor(Connectivity,Gene Significance) vs Hard Thresholds",ylab="Spearman
Correlation(Gene Significance,Connectivity)",xlab="Threshold tau", type="n",
ylim=range(c(SpearmanCorrelationsHard),na.rm=T))
text(thresholds1, SpearmanCorrelationsHard,labels=thresholds1,col="black")
# this draws a vertical line at the tau that was chosen by the
# scale free topology criterion.
abline(v=0.65,col="red")
```

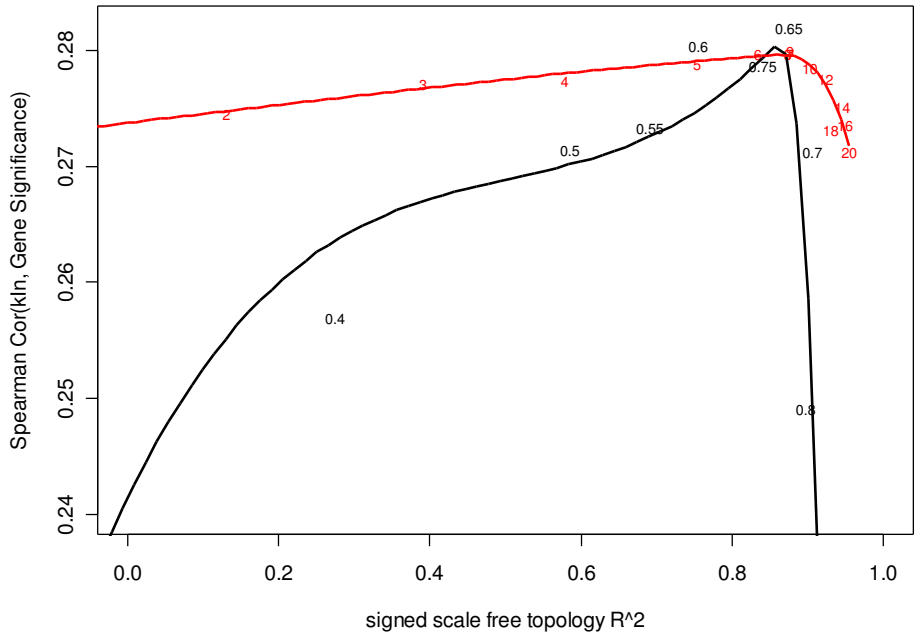


#Note that very high or very small threshold values lead to a small correlation,
#i.e. a diminished biological signal. The red line corresponds to the threshold that was picked using #the scale free topology criterion. It is remarkable that the scale free topology criterion picked the #threshold that leads to the most significant correlation between node #connectivity and gene #significance.

The following plot relates Spearman Correlation(kIn, Gene Significance) to scale free
topology R^2 for different hard and soft powers.

```
library(splines)
knots1=c(.85)
par(mfrow=c(1,1))
x1=-sign(RdichotTable[,4])*RdichotTable[,3]
y1= (as.vector(SpearmanCorrelationsHard))
# spline fit
bs1=lm(y1~bs(x1,knots=knots1,Connectivity=3))
xpred1=seq(min(x1),max(x1),length=100)
x2=-sign(RpowerTable[,3])*RpowerTable[,2]
y2= (as.vector(SpearmanCorrelationsSoft))
bs2=lm(y2~bs(x2, knots=knots1,Connectivity=3))
xpred2=seq(min(x2),max(x2),length=100)
plot(x1,y1,type="n",ylim=c(0.24,max(c(y1,y2))),xlim=c(0,1),
```

```
xlab="signed scale free topology R^2", ylab="Spearman Cor(kIn, Gene
Significance)")
text(x1, y1,col="black",label= as.character(thresholds1),cex=.7 )
points(x2,y2,type="n")
text(x2, y2,col="red",labels= as.character(powers1),cex=.7);
lines(xpred1,predict(bs1,data.frame(x1=xpred1 )  ),col="black",lwd=2)
lines(xpred2,predict(bs2,data.frame(x2=xpred2 )  ),col="red",lwd=2)
```



APPENDIX: Using "closeness" to an essential turquoise hub gene to show that soft thresholding is superior to hard thresholding.

#Let us now find the essential gene in turquoise module that has the highest connectivity

```
# Don't even try to understand this code, just trust it...
indexset=c(1:dim(ConnectivityMeasures)[[1]])[GeneSignificance==1 &
is.element(colorh1,c("turquoise")) ]
c(1:dim(ConnectivityMeasures)[[1]])[ GeneSignificance==1 &
is.element(colorh1,c("turquoise")) &
ConnectivityMeasures$kWithin==max(ConnectivityMeasures$kWithin[indexset]) ]

# This yields that gene 1422 is the most highly connected, essential gene in the
#turquoise module.

colorh1[1422]
ConnectivityMeasures[1422,]
```

IDEA: we will compare different closeness measure (adjacency matrices and
topological overlap matrices for different hard and soft thresholds) that relate

gene 1422 to other turquoise genes or other genes in the network.
We would expect that the closer a gene is to gene 1422, the more likely it is that it will be
#essential itself. Thus there should be a relationship between closeness and essentiality.
We measure this relationship with the C-index (area under the ROC curve)

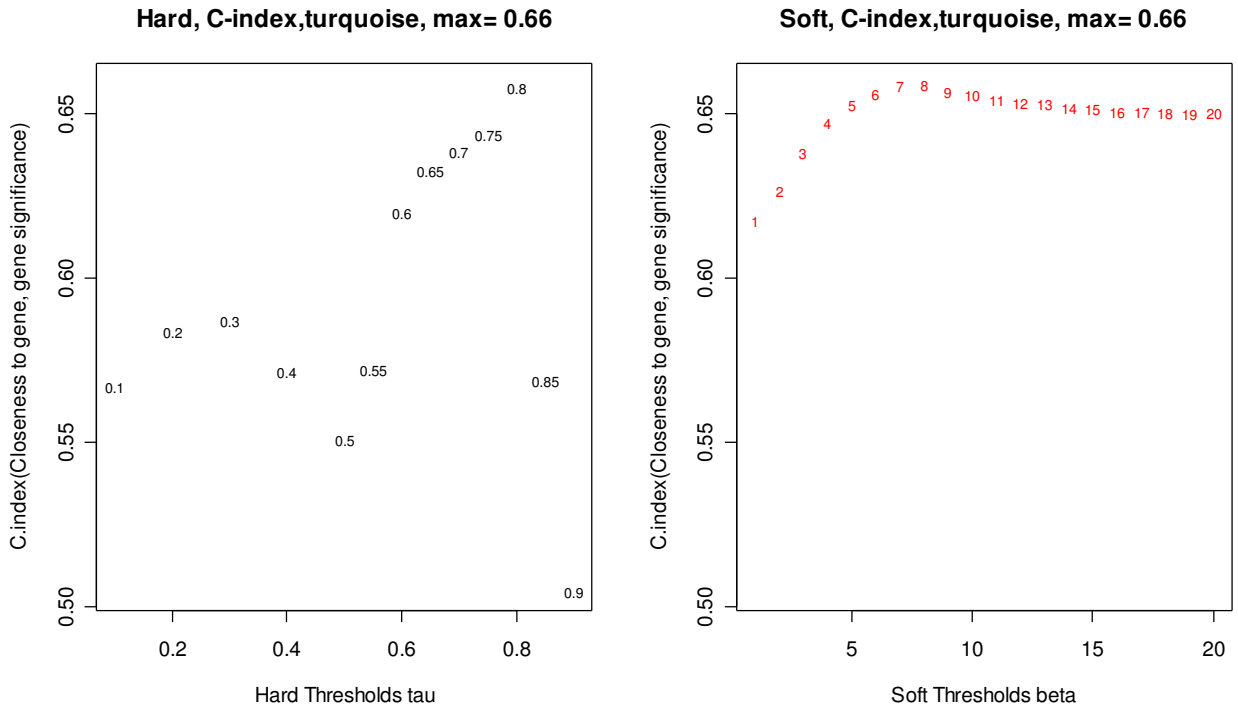
```
# The following vector will contain the C-index between closeness and
# essentiality for turquoise module genes only.
HardCindex=rep(666,length(thresholds1))
for (i in c(1:length(thresholds1)) ){

# The function TOMdistROW computes the TOM distance of a gene (node)
# with that of all other genes in the network.
# The first entry is an integer that specifies which row of the adjacency matrix
# corresponds to the gene of interest.
# Output=vector of TOM distances to the gene of interest.
TomVector=1-
TOMdistROW(1422,abs(corhelp[restConnectivity,restConnectivity])>thresholds1[i])
HardCindex[i]=rcorr.cens(c(TomVector[is.element(colorh1,c("turquoise"))]),
datSummary$essentiality
[restConnectivity][is.element(colorh1,c("turquoise"))])[[1]]
  }
```

Same for soft thresholding (weighted networks).

```
SoftCindex=rep(666,length(powers1))
for (i in c(1:length(powers1)) ){
TomVector=1-
TOMdistROW(1422,abs(corhelp[restConnectivity,restConnectivity])^powers1[i])
SoftCindex[i]=rcorr.cens(c(TomVector[is.element(colorh1,c("turquoise"))]),
datSummary$essentiality
[restConnectivity][is.element(colorh1,c("turquoise"))])[[1]]
  }
```

Plots that sum up our studies for hard and soft thresholds

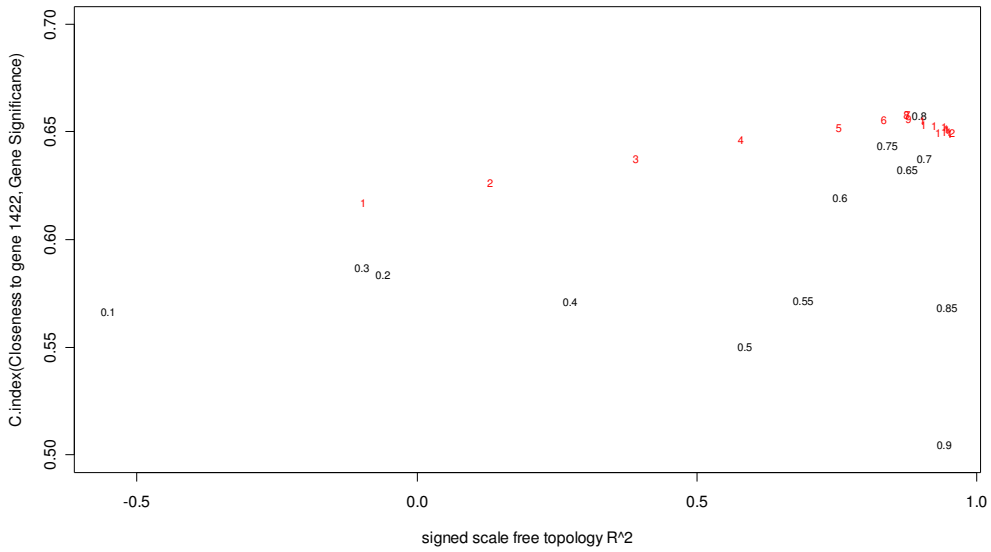
```
par(mfrow=c(1,2))
cex1=.7
range1=range(c(HardCindex, SoftCindex))
plot(thresholds1,HardCindex, main=paste("Hard, C-index,turquoise, max=",
signif(max(HardCindex),2)), type="n",ylab="C.index(Closeness to gene, gene
significance)",xlab="Hard Thresholds tau",ylim=range1)
text(thresholds1,HardCindex, labels=thresholds1, col="black",cex=cex1)
plot(powers1,SoftCindex, main=paste("Soft, C-index,turquoise, max=",
signif(max(SoftCindex),2)),type="n", ylab="C.index(Closeness to gene, gene
significance)",xlab="Soft Thresholds beta", ylim=range1)
text(powers1,SoftCindex,  labels=powers1, col="red" ,cex=cex1)
```



Comment: Again we find that the scale free topology criterion leads to networks whose "closeness" measure based on the TOM dissimilarity has approximately an optimal biological signal when considering the turquoise module genes. Note that on average the soft thresholding approach (right hand side) leads to a higher C-index, i.e. a larger biological signal.
The soft thresholding approach leads to findings that are much more robust with respect to the choice of the adjacency function parameter (here the power). In contrast, the hard thresholding approach is less robust with respect to the choice of the adjacency function parameter tau.

Let's compare hard thresholding to soft thresholding directly.
The following scatter plot relates the C-index to the scale free topology #fitting index (R^2) for different hard and soft thresholds.

```
par(mfrow=c(1,1))
cex1=.7
plot(-sign(RdichotTable[,4])*RdichotTable[,3],
HardCindex,type="n",ylim=c(0.5,.7),
xlab="signed scale free topology R^2", ylab="C.index(Closeness to gene 1422,
Gene Significance)")
text(-sign(RdichotTable[,4])*RdichotTable[,3], HardCindex,col="black",
label= as.character(RdichotTable[,1]),cex=cex1 )
points(-sign(RpowerTable[,3])*RpowerTable[,2], SoftCindex,
col="red",pch= as.character(RpowerTable[,1]) ,cex=cex1)
```



Comments

- overall soft thresholding leads to a "closeness" measure that contains more biological signal than that of hard thresholding.
- The findings for soft thresholding are relatively robust with respect ot the choice of the power. In contrast, the findings of hard thresholding are less robust.
- the scale free topology criterion leads to nearly optimal performance for both the weighted and unweighted networks.\

```
THE END:
To cite the code and methods in this manual, please use
```

Bin Zhang and Steve Horvath (2005) "A General Framework for Weighted Gene Co-Expression Network Analysis", Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17