

# A simulated gene co-expression network to illustrate the use of the topological overlap matrix for module detection

Steve Horvath, Mike Oldham

Correspondence to [shorvath@mednet.ucla.edu](mailto:shorvath@mednet.ucla.edu)

## Abstract

Here we simulate gene expression data that give rise 2 distinct network modules.

This simulation studies is used to show

- a) that a module detection method based on the topological overlap matrix is able to detect the true underlying module structure.
- b) how to use our custom made R code functions, which can be found here:

<http://www.genetics.ucla.edu/labs/horvath/GeneralFramework/NetworkFunctions.txt>

Specifically, using the simulated data, we construct 2 gene co-expression networks: a weighted network and an unweighted network. The weighted network is based on using the power adjacency function to define the connection strenghts (adjacencies) between pairs of genes. Thus, the connection strength equals the correlation raised to a certain power (soft thresholding). The unweighted network is based on hard thresholding the absolute value of the Pearson correlation matrix.

To group genes with coherent expression profiles into modules, we use average linkage hierarchical clustering, which uses the topological overlap measure as dissimilarity.

In an unweighted network, 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 2005].

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.

## Definition of Topological Overlap for an Unweighted Network

As cited in our article, the topological overlap matrix comes from the manuscript by Ravasz et al 2002. They report our form of the topological overlapmatrix in the methods supplement of their paper (there is a typo in the main paper!). Topological overlap of two nodes (genes) reflects their relative interconnectivity. For a network represented by an adjacency matrix

$A = [a_{ij}]$ ,  $a_{ij} \in \{0,1\}$  is given by

$$\omega_{ij} = \frac{l_{ij} + a_{ij}}{\min\{k_i, k_j\} + 1 - a_{ij}} \quad (1)$$

where,  $l_{ij} = \sum_{u \neq i, j} a_{iu} a_{uj}$  denotes the number of nodes to which both i and j are connected, and  $k$  is

the number of connections of a node, with  $k_i = \sum_{u \neq i} a_{iu}$  and  $k_j = \sum_{u \neq j} a_{ju}$ . Since  $a_{ij} \in [0,1]$ , we find

that  $l_{ij} = \sum_{u \neq i, j} a_{iu} a_{uj} \leq \min(\sum_{u \neq i, j} a_{iu}, \sum_{u \neq i, j} a_{uj}) = \min(\sum_{u \neq i} a_{iu}, \sum_{u \neq j} a_{ju}) - a_{ij}$ .

Thus,  $l_{ij} + a_{ij} \leq \min(\sum_{u \neq i} a_{iu}, \sum_{u \neq j} a_{ju})$ . Since  $1 - a_{ij} \geq 0$ , we find that  $\omega_{ij}$  is a number between 0 and 1.

There are 2 reasons for adding  $1 - a_{ij}$  to the denominator in the topological overlap matrix: 1) in this form, the denominator can never be 0 and 2) for an *unweighted* network, one can show that  $\omega_{ij}=1$  only if the node with fewer links satisfies two conditions: (a) all of its neighbors are also neighbors of the other node, i.e. it is connected to all of the neighbors of the other node and (b) it is linked to the other node. In contrast,  $\omega_{ij}=0$  if  $i$  and  $j$  are unlinked and the two nodes don't have common neighbors. Further, the topological overlap matrix is symmetric, i.e.  $\omega_{ij} = \omega_{ji}$ . and its diagonal elements are set to 0 (i.e.  $\omega_{ii}=0$ ). The rationale for considering this similarity measure is that nodes that are part of highly integrated modules are expected to have high topological overlap with their neighbors.

### **Definition of Topological Overlap for a *Weighted* Network**

Formula 1 for the topological overlap does not require that the adjacency matrix  $A=[a_{ij}]$  contain binary entries (1 or 0). Zhang and Horvath (2005) proposed to generalize the topological overlap matrix to weighted networks by simply using the real numbers  $a_{ij}$  in  $[0,1]$  in formula 1. Below, we illustrate that this definition is sensible since it retrieves the correct cluster structure in a simulated example.

### **Identifying Gene Modules**

Authors differ on how they define modules. Intuitively speaking, we assume that modules are groups of genes whose expression profiles are highly correlated across the samples. Modules are groups of nodes that have high topological overlap. Module identification is based on the topological overlap matrix  $\Omega=[\omega_{ij}]$  defined as (1). To use it in hierarchical clustering, it is turned into a dissimilarity measure by using the standard approach of subtracting it from one (i.e. the topological overlap based dissimilarity measure is defined by  $d_{ij}^{\omega} = 1 - \omega_{ij}$ ).

To group genes with coherent expression profiles into modules, we use average linkage hierarchical clustering coupled with the TOM-based dissimilarity. In this article, gene modules correspond to branches of the hierarchical clustering tree (dendrogram). The simplest (not necessarily best) method is to choose a height cutoff to cut branches off the tree. The resulting branches correspond to gene modules, i.e. sets of highly co-expressed genes.

## **REFERENCES**

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](http://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 (2006) *Connectivity, Module-Conformity, and Significance: Understanding Gene Co-Expression Network Methods. Technical Report and software code at* <http://www.genetics.ucla.edu/labs/horvath/ModuleConformity/>

#The user should copy and paste the following script into the R session.

#Text after "#" is a comment and is automatically ignored by R.

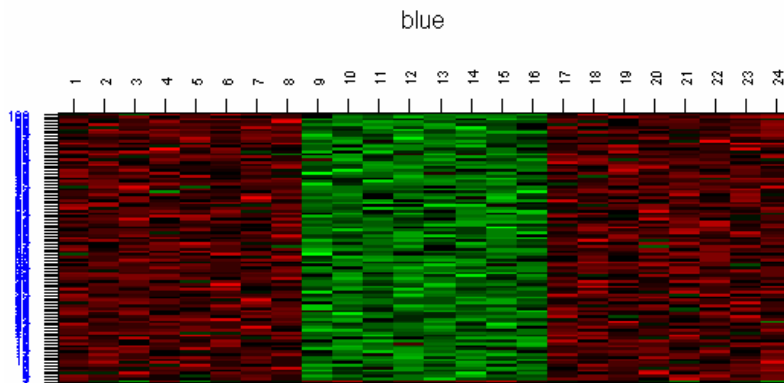
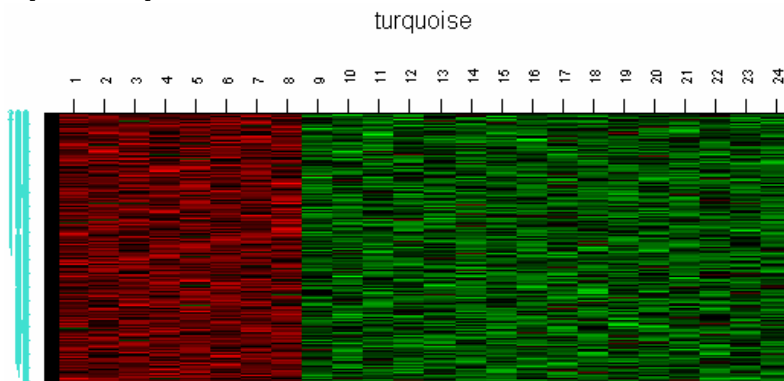
```
# read in the R libraries
library(MASS) # standard, no need to install
library(class) # standard, no need to install
library(cluster)
library(sma) # install it for the function plot.mat
library(impute)# install it for imputing missing value
library(Hmisc) # install it for the C-index calculations

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

# read in the custom network functions (see our webpage:
http://www.genetics.ucla.edu/labs/horvath/GeneralFramework/NetworkFunctions.txt)

source("NetworkFunctions.txt")

# Simulated Model
# Here we simulate the data. For simplicity, we look at 600 genes and 24 microarray samples.
# The following plot (described below) visualizes our simulation model: the turquoise module genes
are over expressed in the first 8 samples, the blue module genes are over-expressed in the first 8
and the last 8 samples. The grey genes are not in any module and they do not have a distinct
expression pattern
```



```
# This model is simulated as follows:
no.genes=600
no.samples=24
```

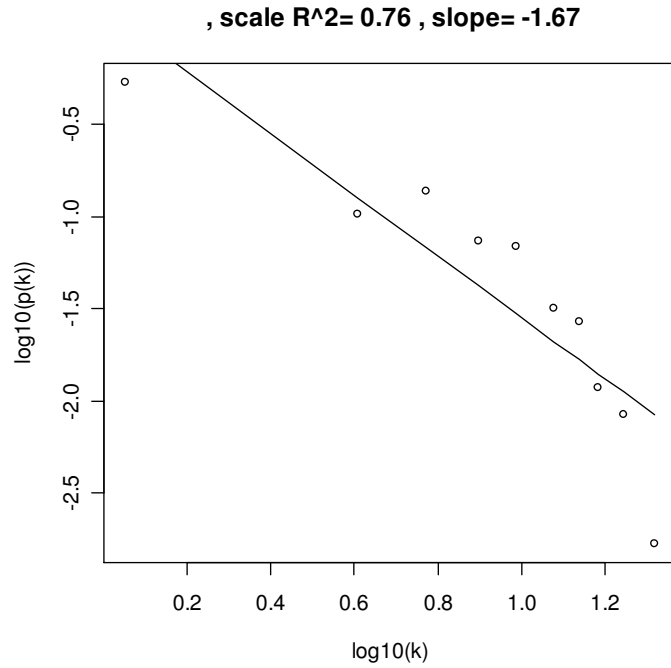
```

# We assume that 1/3 of the genes are in module 1 and 1/6 of the genes are in module 2
#Proportion of genes in modules 1 and 2, respectively.
prop1=c(1/3,1/6)
# This vector code the true simulated cluster structure.
# Turquoise denotes the biggest module. Blue the second module.
# Grey denotes non-module genes
truecluster=rep(c("turquoise","blue","grey"),c(as.integer(no.genes*prop1[1]),as.integer(
no.genes*prop1[2]),no.genes*(1-prop1[1]-prop1[2])))

# This matrix contains the simulated expression values (rows are genes, columns samples)
# Each simulated cluster has a distinct mean expression across the samples
dat1=matrix(rnorm(no.genes*no.samples),nrow=no.genes,ncol=no.samples)
clusterpattern1=rep(c(3,0,0), c(no.samples/3,no.samples/3,no.samples/3))
clusterpattern2=rep(c(3,0,3), c(no.samples/3,no.samples/3,no.samples/3))
clusterpattern3=rep(c(0,0,0), c(no.samples/3,no.samples/3,no.samples/3))
restrict1= truecluster=="turquoise"
dat1[restrict1,]=
dat1[restrict1,]+matrix(clusterpattern1,nrow=sum(restrict1),ncol=no.samples,byrow=T)
restrict2= truecluster=="blue"
dat1[restrict2,]=dat1[restrict2,]+matrix(clusterpattern2,nrow=sum(restrict2),ncol=no.samples,byrow=T
)
restrict3= truecluster=="grey"
dat1[restrict3,]=dat1[restrict3,]+matrix(clusterpattern3,nrow=sum(restrict3),ncol=no.samples,byrow=T
)
# The following matrix of gene expression values is defined such that
# rows correspond to microarrays and columns to genes.
datExpr=t(dat1)
# Based on the gene expression data, we define the weighted network (=adjacency matrix)
# using soft thresholding with a power parameter of 10.
ADJ=abs(cor(datExpr))^10

```

```
# Then the connectivity is defined as the sum of connection strengths
k=apply(ADJ,2,sum)
# This plot shows that the connectivity follows approximately
# scale free topology with gamma=2.13
par(mfrow=c(1,1))
ScaleFreePlot1(k)
```



---

Comment: The network satisfies scale free topology approximately. Our emphasis here is to illustrate module detection and the relationship between module structure and scale free topology is another topic...

## #Module Detection

# An important step in network analysis is module detection.

# Here we use methods that use clustering in combination with the topological

# overlap matrix (TOM)

# For the definition of the weighted network TOM matrix see Horvath and Zhang 04.

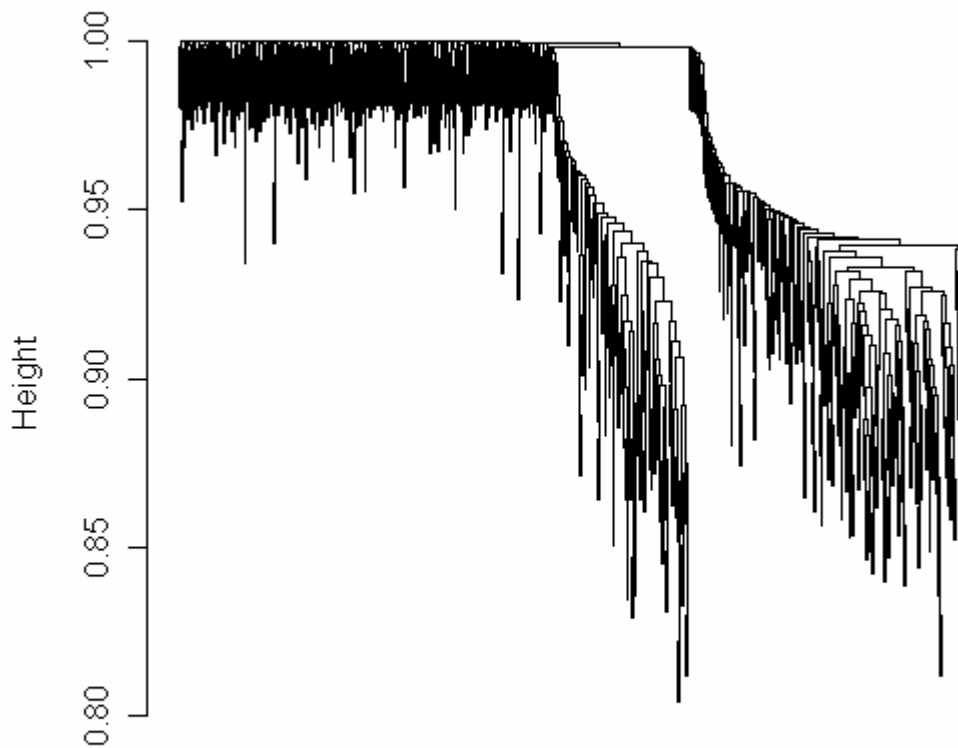
# For the unweighted network TOM matrix see Ravasz et al (2001) and Yip and Horvath (2005)

```
dissGTOM1=TOMdist1(ADJ)
collect_garbage()
```

# 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");
par(mfrow=c(1,1))
plot(hierGTOM1,labels=F)
```

### Cluster Dendrogram



```
as.dist(dissGTOM1)
hclust(*, "average")
```

```

# By our definition, modules correspond to branches of the tree.
# The question is what height cut-off h1 should be used? This depends on the
# biology. Large height values lead to big modules, small values lead to small
# but very cohesive modules, see also Horvath, Dong, Yip (2005).
# In reality, the user should use different thresholds to see how robust the findings are.

```

```

# 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.
# We only consider modules that contain at least 10 genes.

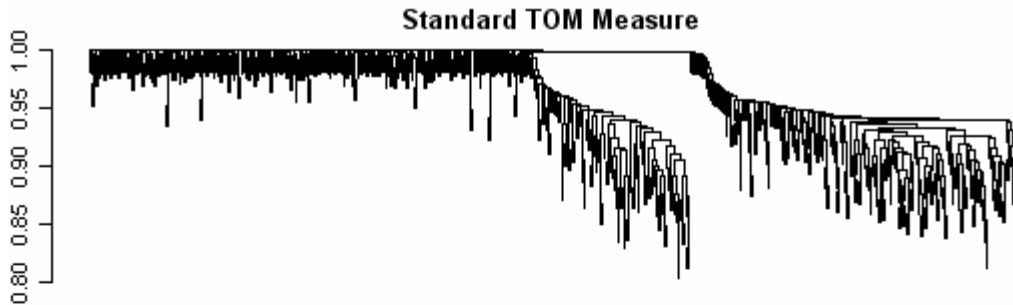
```

```

colorh1=as.character(modulecolor2(hierGTOM1,h1=.98, minsize1=10))

par(mfrow=c(3,1), mar=c(2,2,2,1))
plot(hierGTOM1, main="Standard TOM Measure", labels=F, xlab="", sub="");
hclustplot1(hierGTOM1,colorh1, title1="Colored by TOM modules")
hclustplot1(hierGTOM1, truecluster, title1="Colored by TRUE module")

```



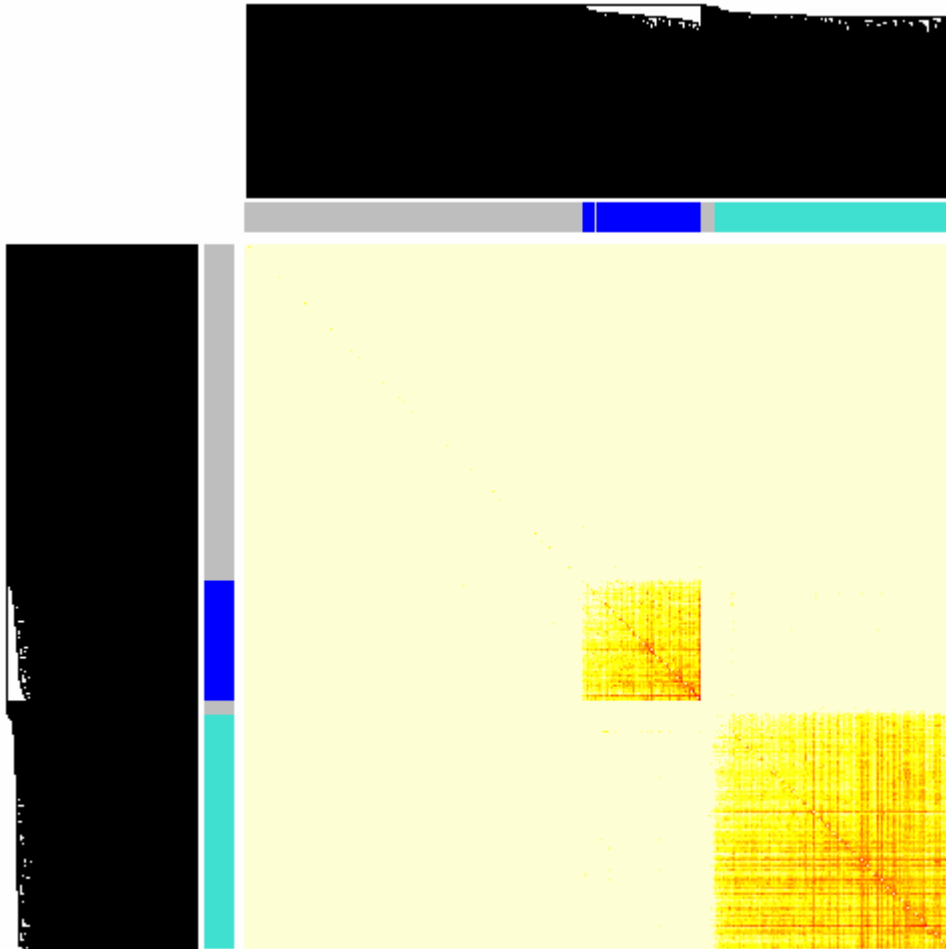
COMMENT: The colors are assigned based on module size. Turquoise (others refer to it as cyan) colors the largest module, next comes blue, etc. Just type `table(colorh1)` to figure out which color corresponds to what module size.

The following table shows that the module detection retrieved the true module structure. Only 2 gene was misclassified.

```
table(colorh1,truecluster)
```

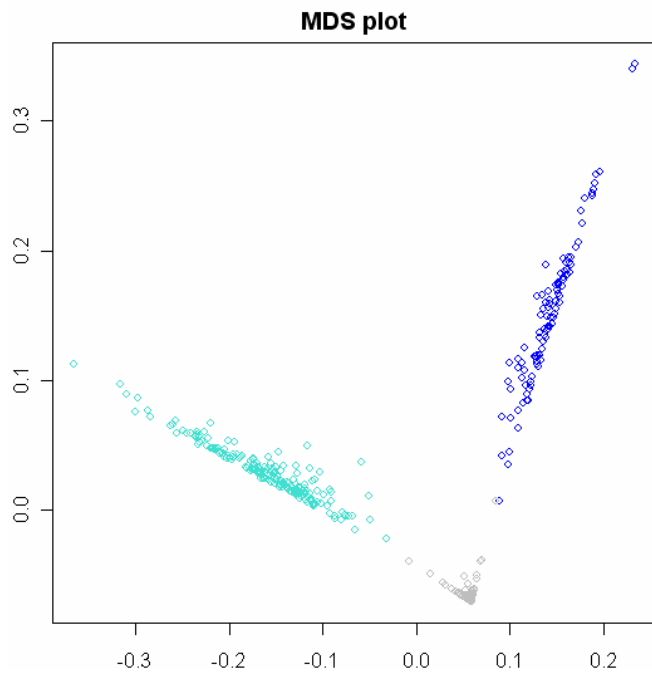
```
# An alternative view of this is the so called TOM plot that is generated by the  
# function TOMplot1  
# Inputs: TOM distance measure, hierarchical (hclust) object, color
```

```
TOMplot1(dissGTOM1 , hierGTOM1, truecluster)
```



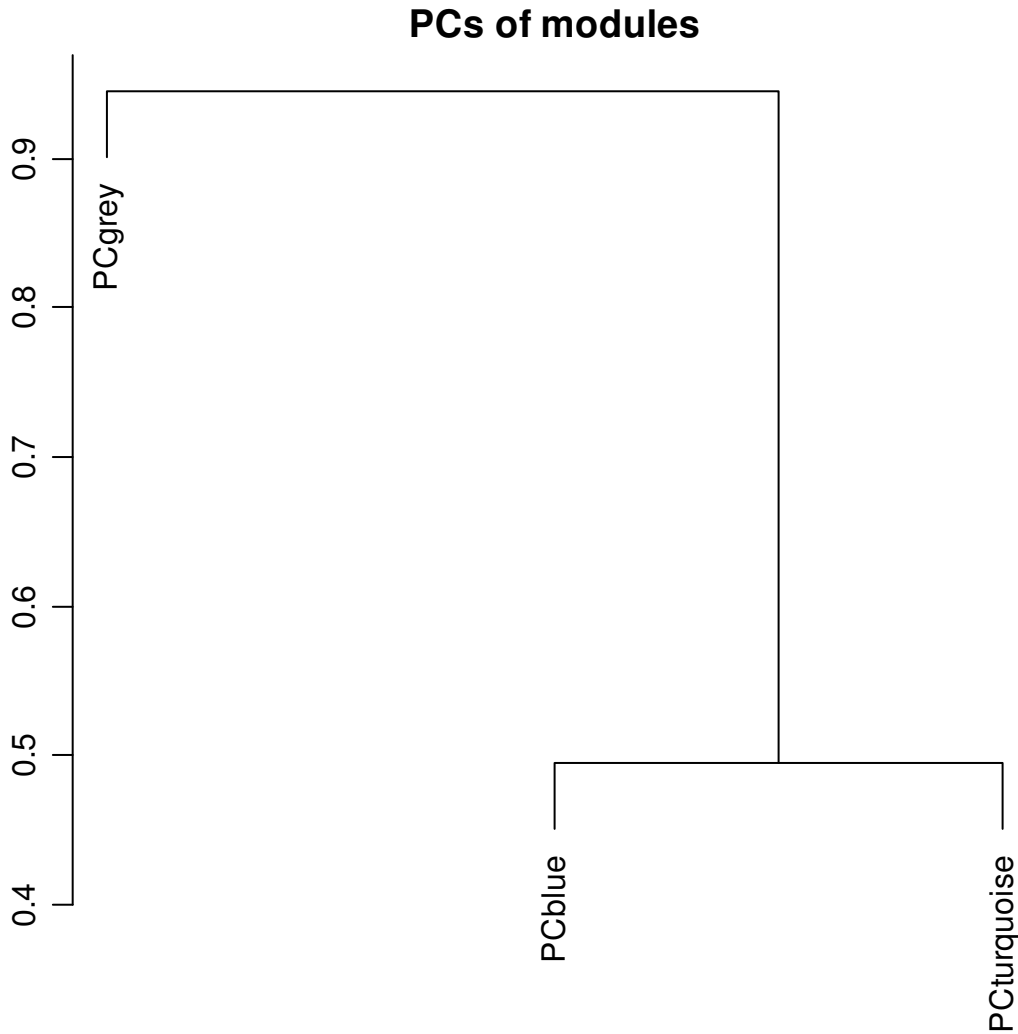
```
collect_garbage()

# We also propose to use classical multi-dimensional scaling plots
# for visualizing the network. Here we chose 2 scaling dimensions
cmd1=cmdscale(as.dist(dissGTOM1),2)
par(mfrow=c(1,1))
plot(cmd1, col=as.character(colorh1), main="MDS plot")
```



Notice that the turquoise module genes are very distinct from the blue and the grey genes. This is in part due to using a high power of 10 for constructed the weighted network. A lower power leads to less distinct results, see below.

```
# To get a sense of how related the modules are one can summarize each module
# by its first eigengene (referred to as principal components).
# For more theoretical details consult Horvath, Dong and Yip (2005)
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")
```

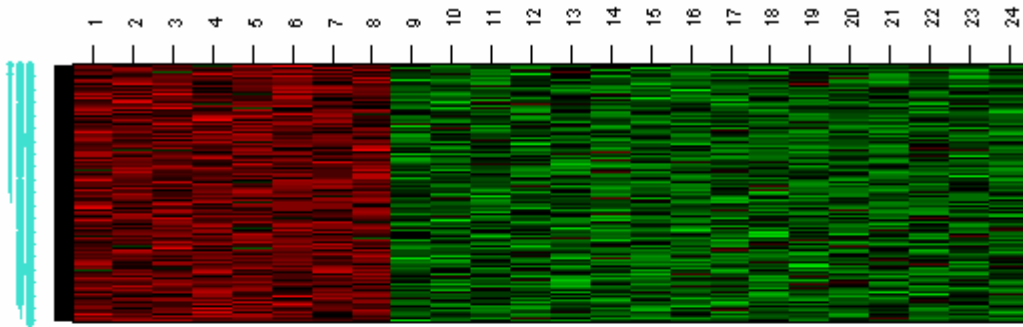


#Comments: the module eigengenes of the blue and the turquoise module have some correlation

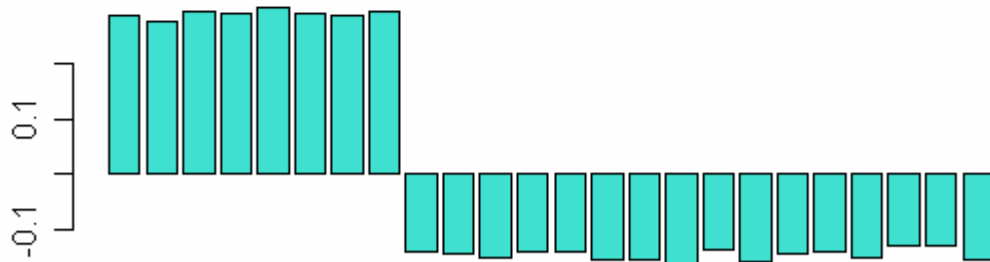
# The following produces heatmap plots for each module.  
 # Here the rows are genes and the columns are samples.  
 # Well defined modules results in characteristic band structures since the corresponding genes are  
 # highly correlated.

```
par(mfrow=c(2,1))
ClusterSamples=hclust(dist(datExpr),method="average")
# for the first (turquoise) module we use
which.module="turquoise"
plot.mat(t(scale(datExpr[ClusterSamples$order,]),colorhl==which.module))
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
title=paste("heatmap of the expressions of the ",which.module," module."))
# Now we plot the expression values of the module eigengene
barplot(PC1$PCTurquoise[ClusterSamples$order],col=which.module, main="Expression
of the turquoise module eigengene")
```

heatmap of the expressions of the turquoise module.



**Expression of the turquoise module eigengene**

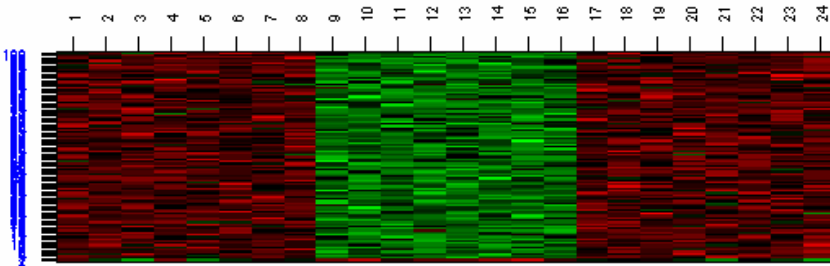


```

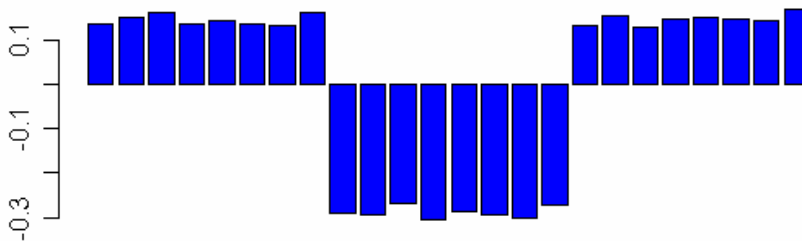
# Same for the blue module
par(mfrow=c(2,1))
ClusterSamples=hclust(dist(datExpr),method="average")
which.module="blue"
plot.mat(t(scale(datExpr[ClusterSamples$order,],[,colorh1==which.module ]))
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
title=paste("heatmap of the expressions of the ",which.module," module." ) )
# Now we plot the expression values of the module eigengene
barplot(PC1$PCblue[ClusterSamples$order],col=which.module, main="Expression of
the blue module eigengene")

```

heatmap of the expressions of the blue module.



**Expression of the blue module eigengene**

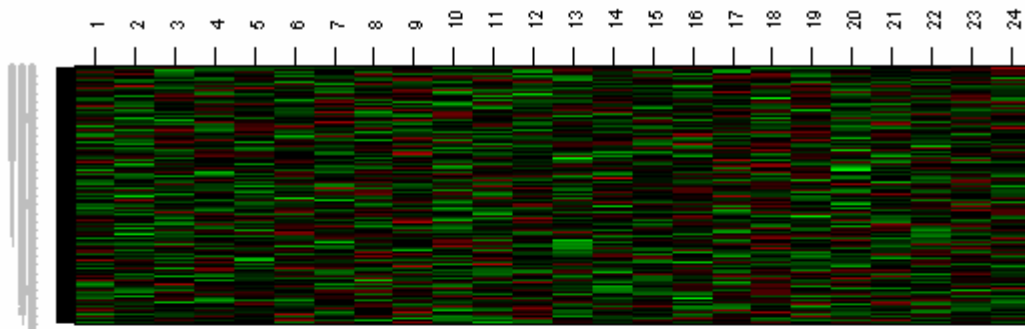


```

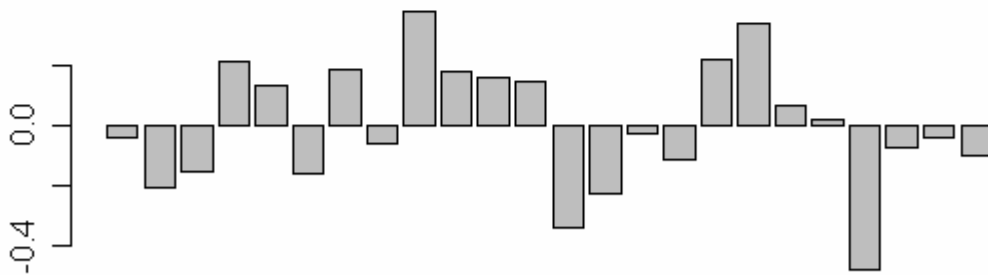
# Same for the grey non-module
par(mfrow=c(2,1))
ClusterSamples=hclust(dist(datExpr),method="average")
which.module="grey"
plot.mat(t(scale(datExpr[ClusterSamples$order,],[,colorh1==which.module ]))
),nrgcols=30,rlabels=T, clabels=T,rcols=which.module,
title=paste("heatmap of the expressions of the ",which.module," module." ) )
# Now we plot the expression values of the module eigengene
barplot(PC1$PCgrey[ClusterSamples$order],col=which.module, main="Expression of
the grey module eigengene")

```

heatmap of the expressions of the grey module.



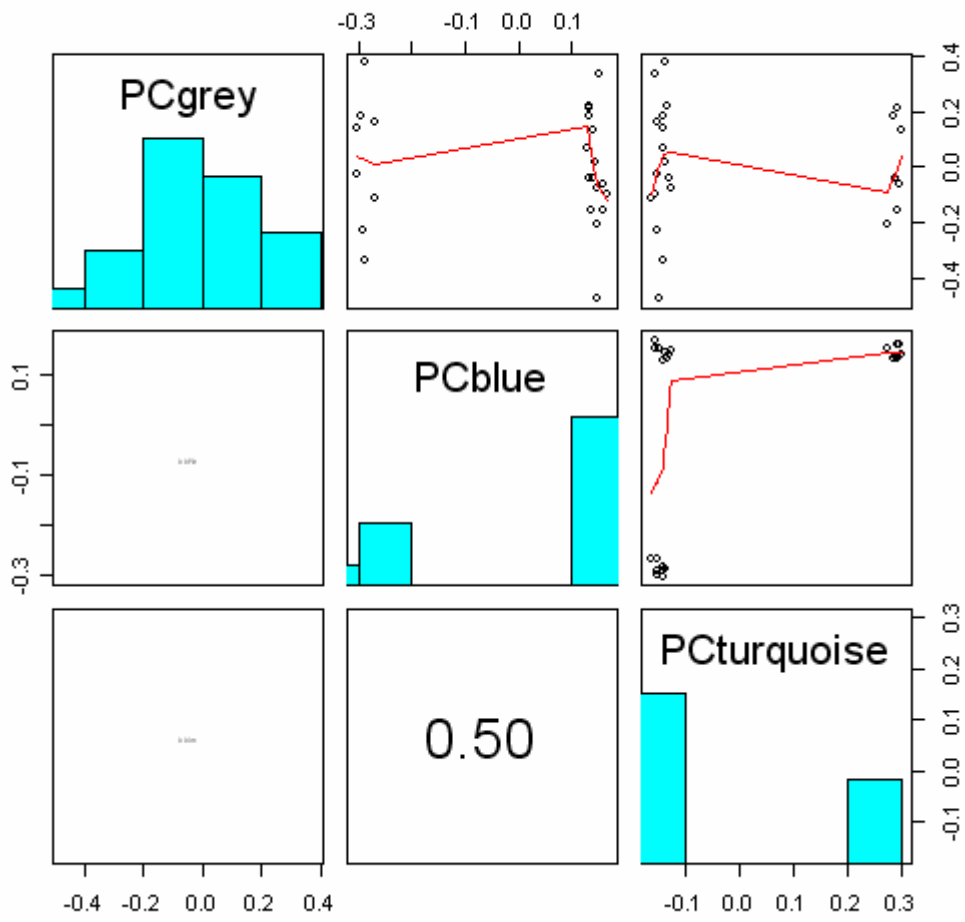
### Expression of the grey module eigengene



---

```
# Now we create scatter plots of the samples (arrays) along the module eigengenes.  
PC1=PC1[,hclustPC1$order]  
pairs( PC1, upper.panel = panel.smooth, lower.panel = panel.cor , diag.panel=panel.hist  
,main="Relation between modules")
```

## Relation between modules

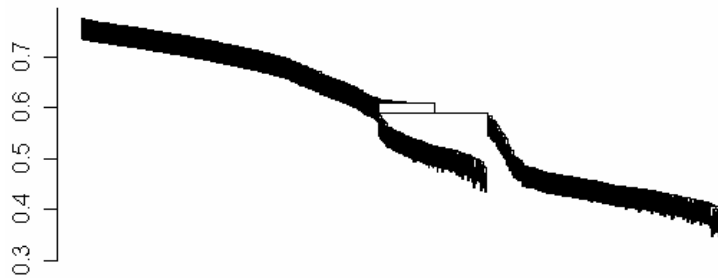


## **THE EFFECT OF CHOOSING A DIFFERENT soft thresholding power**

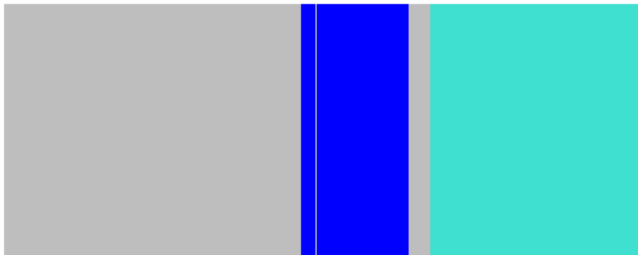
```
# Based on the gene expression data, we define the weighted network (=adjacency matrix)  
# using soft thresholding with a power parameter of 1.
```

```
dissGTOM2=TOMdist1(abs(cor(datExpr))^1)  
collect_garbage()  
# Now we carry out hierarchical clustering with the TOM matrix.  
hierGTOM2 <- hclust(as.dist(dissGTOM2),method="average");  
par(mfrow=c(2,1))  
plot(hierGTOM2,labels=F)  
hclustplot1(hierGTOM2, truecluster, title1="Colored by TRUE module")
```

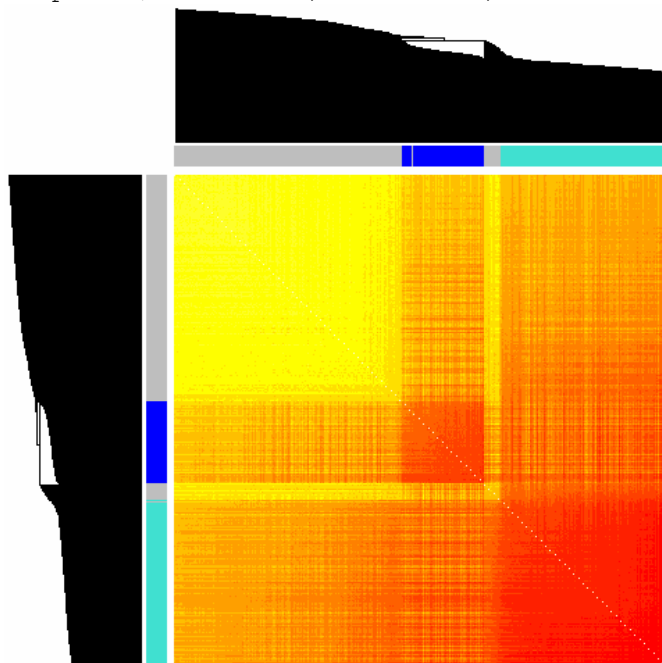
**Cluster Dendrogram**



**Colored by TRUE module**

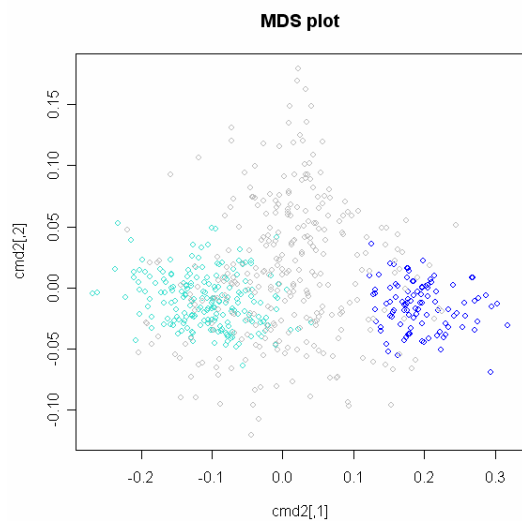


```
# An alternative view of this is the so called TOM plot that is generated by the  
TOMplot1(dissGTOM2 , hierGTOM2, truecluster)
```



Comment: Note that the modules are less pronounced: they have more topological overlap (red) with the grey (non-module) genes. Similarly, the following MDS plots shows that the module genes are less distinct from the grey genes.

```
cmd2=cmdscale(as.dist(dissGTOM2),2)  
par(mfrow=c(1,1))  
plot(cmd2, col=as.character(truecluster), main="MDS plot")
```



## Discussion of findings when choosing a power=1 to construct the network

- 1) When using a power of 1 to define the network, one arrives at a measure of topological overlap that still allows one to retrieve the true underlying cluster structure. Thus the module identification is highly robust with respect to the power. This is an advantage of the weighted network method. In contrast, module detection is less robust to the choice of a hard threshold used to dichotomize the correlation matrix when constructing an unweighted network.
- 2) While the network construction is very robust, there is an advantage in considering higher powers: The higher the power, the more distinct become the branches corresponding to the modules since the correlations with the grey noise genes become suppressed.

## Constructing an unweighted networks and comparing it to the weighted network.

Here we study whether the 'soft' modules of the weighted network described above can also be found in the unweighted network

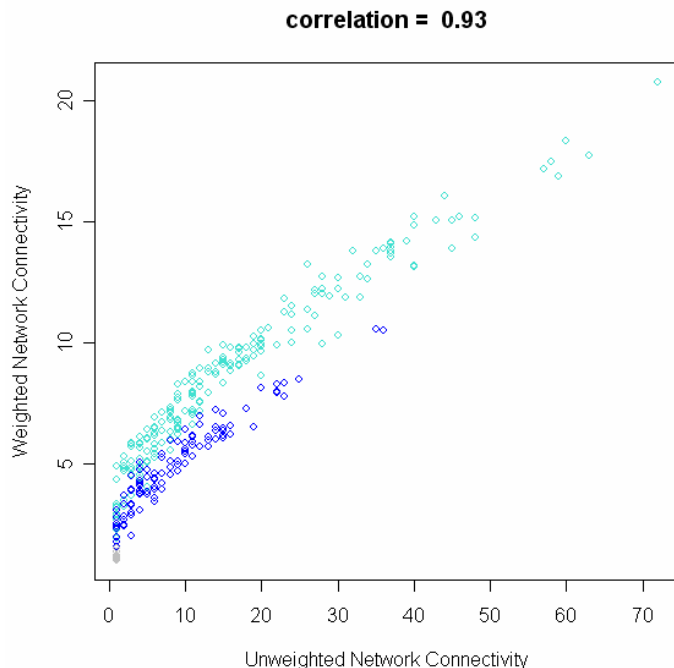
#Let's define the adjacency matrix of an unweighted network by

# using a hard threshold to dichotomize the correlation matrix.

```
ADJhard <- abs(cor(datExpr,use="p"))>0.8
collect_garbage()
# This is the unweighted connectivity
khard=as.vector(apply(ADJhard,2,sum))
```

# Let's compare weighted to unweighted connectivity in a scatter plot

```
par(mfrow=c(1,1))
plot(khard, k,xlab="Unweighted Network Connectivity",ylab="Weighted Network Connectivity",main=paste("correlation = ",signif(cor(khard,k), 2)),col=colrh1)
```

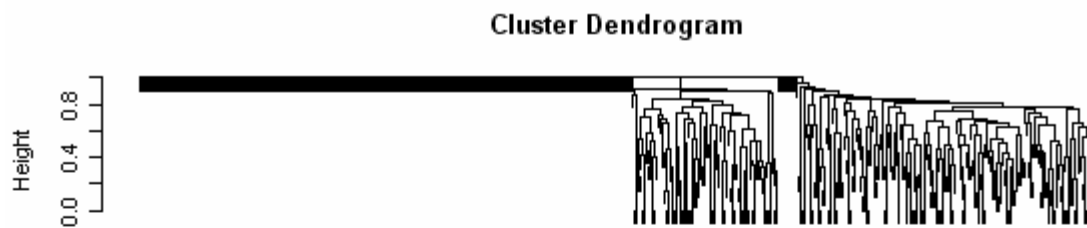


# Comments: weighted and unweighted network connectivity measures are highly correlated

```
# The following code computes the topological overlap matrix based on the
# adjacency matrix.
dissGTOM1hard=TOMdist1(ADJhard)
collect_garbage()
```

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

```
hierGTOM1hard <- hclust(as.dist(dissGTOM1hard),method="average");
par(mfrow=c(3,1))
plot(hierGTOM1hard,labels=F)
colorh2=as.character(modulecolor2(hierGTOM1hard,h1=.99, minsize1=10))
hclustplot1(hierGTOM1hard,colorh2, title="Colored by UNWEIGHTED TOM modules")
hclustplot1(hierGTOM1hard, truecluster, title="Colored by TRUE module")
```



```
as.dist(dissGTOM1hard)
hclust (*, "average")
```

**Colored by UNWEIGHTED TOM modules**



**Colored by TRUE module**



```
table(colorh2,truecluster)
```

```
      truecluster
colorh2  blue grey turquoise
  blue      91   0         0
  grey       9  300        12
  turquoise  0   0        188
```

Note that 21=9+12 observations get misclassified. Our previously discussed weighted network was slightly better at retrieving the module structure.

## Discussion

Our simulated example clearly illustrates that the topological overlap matrix can retrieve the simulated underlying module structure both in weighted and unweighted networks.

Using this code one can easily verify that module identification is highly robust with respect to a) whether a weighted or an unweighted network is used, b) what soft threshold is used, i.e. what power is used for computing the adjacency matrix c) what hard threshold is used.

The advantage of using a soft thresholding approach with the power adjacency function is that it preserves the continuous nature of the gene co-expression information (correlation) and its findings are more robust. There are also deeper theoretical reasons for using the power adjacency function, see Horvath, Dong, Yip (2006).