APPENDIX A

Building Weighted Gene Co-Expression Networks

Network construction: The nodes of a co-expression network correspond to gene expressions, and edges between genes are determined by the pairwise Pearson correlations between gene expressions. By raising the absolute value of the Pearson correlation to a power $\beta \geq 1$ (soft thresholding), the weighted gene co-expression network construction emphasizes large correlations at the expense of low correlations. Specifically, $a_{ij} = \left| cor\left(x_i, x_j\right) \right|^{\beta}$ represents the adjacency (connection strength) between genes *i* and *j*. In contrast to unweighted networks based on using a "hard" threshold to dichotomize the correlation matrix, weighted networks preserve the continuous nature of co-expression information and lead to highly robust results. A detailed discussion of the network construction algorithm and specific benefits of soft thresholding may be found in (Zhang and Horvath 2005).

The connectivity k_i of the i^{th} gene is defined as the sum of the adjacency measures with the given gene *i*: $k_i = \sum_{u \neq i} a_{iu}$. Analogously, the *intramodular* connectivity (kIN) is found by summation of adjacencies over all genes in a particular module (more details follow below).

Network module identification: In weighted gene co-expression networks, modules correspond to clusters of highly correlated genes. Similar to the term 'cluster', no consensus on the meaning of the term 'module' seems to exist in the literature. In our applications, we use a clustering procedure to identify modules (clusters) of genes with high topological overlap. The topological overlap between genes *i* and *j* reflects their relative interconnectedness (Ravasz et al. 2002; Yip and Horvath 2007). The topological overlap measure is defined by:

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

where $l_{ij} = \sum_{u \neq i,j} a_{iu} a_{uj}$. In an unweighted network, l_{ij} equals the number of genes to which both *i* and *j* are connected. In this case, the topological overlap measure $\omega_{ij} = 1$ if the gene with fewer connections satisfies two conditions: (a) all of its neighbors are also neighbors of the other gene, and (b) it is connected to the other gene. In contrast, $\omega_{ij} = 0$ if *i* and *j* are not connected and the two genes do not share any neighbors.

We follow the suggestion of Ravasz *et al.* 2002 to turn the topological overlap matrix Ω into a dissimilarity measure by subtracting its members from 1, i.e. $d_{ij} = 1 - \omega_{ij}$ (Ravasz et al. 2002; Yip and Horvath 2007). We use d_{ij} as input of average linkage hierarchical clustering to arrive at a dendrogram (clustering tree) (Kaufman and Rousseeuw 1990). Modules are defined as the branches of the dendrogram. For example, in Fig. 2a we show the dendrogram of BxD expression data. Genes or proteins of proper modules are assigned a color (e.g. turquoise, blue, etc.). Genes outside any proper module are colored grey. Our module definition depends on how the branches are cut from the dendrogram. Several methods and criteria for identifying branches in a dendrogram have been proposed (Ghazalpour et al. 2006; Ravasz et al. 2002; Zhang and Horvath 2005). In practice, it is advisable to study how robust the results are with respect to alternative module detection methods. In our online R software tutorial (found at http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/DifferentialNetworkAnalysis/), we show that our findings are highly robust with respect to alternative module definitions. In addition, we use a functional enrichment analysis of the resulting modules to provide indirect evidence that the modules are biologically meaningful. Our module detection approach has led

to biologically meaningful modules in several applications (Carlson et al. 2006; Gargalovic et al. 2006; Ghazalpour et al. 2006; Mischel and Cloughesy 2006; Oldham et al. 2006; Ravasz et al. 2002; Ye and Godzik 2004), but we make no claim that it is optimal.

Tissue contamination, technical artifacts, and microarray sample outliers can give rise to spurious, biologically uninteresting modules. Since our module detection method does not make explicit use of gene ontology information, it is advisable to relate modules to public gene ontology information. Toward this end, many functional enrichment software packages are available including the DAVID Database for Annotation, Visualization, and Integrated Discovery (Dennis et al. 2003).

Module eigengene: The module eigengene (ME) can be considered the best representative of the module expressions. Statistically, it is defined as the first principal component of the expression data. It can also be considered as the most highly connected gene inside the module. A trait-related module eigengene can be used in multiple ways, e.g. as an intermediate phenotype or to find chromosomal locations (module QTLs) that affect the module expressions (Ghazalpour et al. 2006).

Intramodular connectivity: Intramodular connectivity is an important concept for identifying clinically relevant genes (Ghazalpour et al. 2006; Horvath et al. 2006; Oldham et al. 2006). We utilize two highly related measures of intramodular connectivity. The standard intramodular connectivity measure kIN of a gene *i* in a module is defined as the row sum of all adjacencies between gene *i* and all other genes in the module: $kIN_i = \sum_{u \neq i} a_{iu}$. A high kIN indicates a gene is

highly connected to all other genes in the module, whereas a low kIN indicates a gene is only loosely connected to the module.

Similarly, we utilize a module eigengene-based intramodular connectivity measure, kME. Because the module eigengene can be considered both the best representative of module expressions and the most highly connected gene inside the module, it is intuitive to correlate gene expressions with this measure to obtain a surrogate connectivity measure. We determine kME via the following equation, where *i* represents the gene of interest: $kME_i = |cor(x_i, ME)|$. In practice we find that kME roughly approximates kIN; Suppl. Fig. 1 depicts the relationship between kIN and kME in the four largest modules in the BxH data. In the Red, Blue, Green and Black modules, we find Spearman correlations of 0.95 ($p \leq 10^{-20}$), 0.85 ($p \leq 10^{-20}$), 0.94 ($p \leq 10^{-20}$) and 0.92 ($p \leq 10^{-20}$), respectively.

Gene and module significance: Relating the expression data to clinical trait information represents an important step towards finding modules of physiological interest. To incorporate clinical trait information into the network, we make use of a gene significance measure. Abstractly speaking, the higher this value is, the more significant a gene is. In our application, the gene significance measures how correlated a gene expression is with a clinical trait. The gene significance with respect to a specific trait is referred to as GStrait, with GStrait of the i^{th} gene in the array = |cor(x(i), trait)|, where x(i) is the gene expression profile of the i^{th} gene. This gene significance measure may also be used to determine a composite, module significance value for each trait's correlation with a given module. The module significance is defined as the

average gene significance in a module of interest. In practice, it is highly related to the correlation between ME and trait.

A simple computational example: To allow the reader to better understand these network concepts, we provide a simple example in the following. We demonstrate a simple example of WGCNA by constructing a gene expression module consisting of 6 arrays and 10 genes from a module seed vector (Suppl. Fig. 2c). Starting with module seed values, a trait value, and SNP additive marker coding values for each of the 6 samples, we obtain connectivity and gene significance values for each gene of the simulated array. A power of $\beta = 10$ was used in this example.

A heat map plot of this module is depicted in Suppl. Fig. 2a. From this module, we recover the observed module eigengene (see Suppl. Fig. 2b) by finding the first principal component of the expression matrix. We find kIN as the row sum of adjacencies ($kIN_i = \sum_{u \neq i} a_{iu}$) and kME as the correlation of a given gene *i* with the module eigengene ($kME_i = |cor(x_i, ME)|$) for each gene in the simulated module. We relate each gene expression to the trait to find GStrait(i) = |cor(x(i), trait)|, where x(i) is the gene expression profile of the i^{th} gene. This measure tells us how enriched gene *i* is for the trait of interest. Similarly, we find the gene significance measure with respect to SNP genotypes as GS.SNP(i) = |cor(x(i), SNP)|. Gene expressions are shown in Suppl. Table 1, and the R code used to produce these results follows.

R CODE

R code starts here.
Read in the necessary 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
```

Set source to the appropriate location of the NetworkFunctions.txt file:

```
source("/Users/TovaFuller/Documents/HorvathLab2006/NetworkFunctions/NetworkFu
nctions.txt")

set.seed(1)
```

The following code simulates a module from a given module eigengene:

```
if(exists("SimulateModule") ) rm(SimulateModule);
SimulateModule=function(ME, size,minimumCor=.8) {
if (size<3) print("WARNING, error?: module size smaller than 3")
if(minimumCor==0) minimumCor=0.0001;
maxnoisevariance=var(ME,na.rm=T)*(1/sqrt(minimumCor)-1)
SDvector=sqrt(c(1:size)/size*maxnoisevariance)
datSignal=matrix(c(ME, ME ,-ME),nrow=size ,ncol=length(ME) ,byrow=T)
datNoise=SDvector* matrix(rnorm(size*length(ME)),nrow=size ,ncol=length(ME))
datModule=datSignal+datNoise
t(datModule)
}
```

We call our SimulateModule function to create our example module

```
module=SimulateModule(ME=rep(c(1,-1),c(3,3))  , size=10, minimumCor=.5)
```

We create a color vector that identifies this module as the Blue module (as the code for determining module eigengene will need this vector).

```
colorh1=rep("blue",10)
```

We find the observed module eigengene:

```
MEobserved=ModulePrinComps1(datexpr=as.matrix(data.frame(module)),
couleur=as.character(colorh1))[[1]]$PCblue
```

Our simulated module eigengene is:

```
moduleSeed=rep(c(1,-1),c(3,3))
```

The two intramodular connectivity measures are calculated below.

```
kME=as.numeric(abs(cor(MEobserved, module,use="p")))
kIN=SoftConnectivity(data.frame(module), power=10)
ks=DegreeInOut(abs(cor(module,use="p"))^6,colorh1)
kIN= ks$kWithin
```

We define the trait and SNP values.
```
trait=c(1,1,1,1,2,2)
SNP=c(2,2,1,1,0,0)
```

GS.trait and GS.SNP are the correlation of trait and SNP values with expressions,
respectively:
```
GS.trait=as.numeric(abs(cor(trait,module, use="p")))
GS.SNP=as.numeric(abs(cor(SNP,module, use="p")))
```

To create the heatmap plot shown in Suppl. Fig. 2a:
```
plot.mat(t(module))
par(mfrow=c(1,1),mar= c(5, 4, 4, 2) + 0.1)
```
To create the plot of ME shown in Suppl. Figure 2b:
```
plot(MEobserved,type="h")
abline(h=0)
```
To create the plot of the module seed shown in Suppl. Figure 2c:
```
plot(moduleSeed,type="h",ylab="Module Seed")
abline(h=0)
```

We write out data of interest:
```
expressions=signif(t(module),3)
write.csv(expressions,"expressions.csv")
rightcol=data.frame(signif(kME,3),signif(kIN,3), signif(GS.trait,3),
signif(GS.SNP,3))
bottomcol=t(data.frame(trait,SNP,signif(MEobserved,3)))
dimnames(rightcol)[[2]]=c("kME","kIN","GS.trait","GS.SNP")
write.csv(rightcol,"rightcol.csv")
write.csv(bottomcol,"bottomcol.csv")
```
End of code.