Eigengene Network Analysis:
Four Tissues Of Female Mice
R Tutorial

Peter Langfelder and Steve Horvath
Correspondence: shorvath@mednet.ucla.edu, Peter.Langfelder@gmail.com

This is a self-contained R software tutorial that illustrates how to carry out an eigengene network analysis across four corresponding to gene expression measurements in the brain, muscle, liver and adipose tissues of female mice of an F2 mouse cross. The R code shows how to perform the analysis reported in Langelder and Horvath (2007). Some familiarity with the R software is desirable but the document is fairly self-contained.

The methods and biological implications are described in the following refence

- *Langfelder P, Horvath S (2007) Eigengene networks for studying the relationships between co-expression modules. BMC Systems Biology*

This tutorial and the data files can be found at
http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/EigengeneNetwork.
More material on weighted network analysis can be found at
http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork.

For a detailed description of the mouse intercross, see Wang et al. (2006). A description of the microarray data can be found in Ghazalpour et al (2006). To facilitate comparison with the original analysis of Ghazalpour et al (2006), we used the same gene selection in our analysis.

Microarray Data
The Agilent microarrays measured gene expression profiles in female mice of an F2 mouse intercross described in Ghazalpour et al 2006. The F2 mouse intercross data set (referred to as B × H cross) involved 135 female mice derived from the F2 intercross between inbred strains C3H/HeJ and C57BL/6J (Ghazalpour et al. 2006; Wang et al. 2006). B×H mice are ApoE null (ApoE −/−) and thus hyperlipidemic and were fed a high-fat diet. The B×H mice were sacrificed at 24 weeks.

Microarray analysis

RNA preparation and array hybridizations were performed at Rosetta Inpharmatics (Seattle, Washington, United States). The custom ink-jet microarrays used in this study (Agilent Technologies [Palo Alto, California, United States], previously described) contain 2,186 control probes and 23,574 non-control oligonucleotides extracted from mouse Unigene clusters and combined with RefSeq sequences and RIKEN full-length clones. Mouse livers were homogenized and total RNA extracted using Trizol reagent (Invitrogen, Carlsbad, California, United States) according to manufacturer's protocol. Three r g of total RNA was reverse transcribed and labeled with either Cy3 or Cy5 fluorochromes. Purified Cy3 or Cy5 complementary RNA was hybridized to at least two microarray slides with fluor reversal for 24 h in a hybridization chamber, washed, and scanned using a laser confocal scanner. Arrays were quantified on the basis of spot intensity relative to background, adjusted for experimental variation between arrays using average intensity over multiple channels, and fit to an error model to determine significance (type I error). Gene expression is reported as the ratio of the mean log10 intensity (mlratio) relative to the pool derived from 150 mice randomly selected from the F2 population.

Microarray data reduction

In order to minimize noise in the gene expression dataset, several data-filtering steps were taken. First, preliminary evidence showed major differences in gene expression levels between sexes among the F2 mice used, and therefore only female mice were used for network construction. The construction and comparison of the male network will be reported elsewhere. Only those mice with complete phenotype, genotype, and array data were used. This gave a final experimental sample of 135 female mice used for network construction. To reduce the computational burden and to possibly enhance the signal in our data, we used only the 8,000 most-varying female liver genes in our preliminary network construction. For module detection, we limited our analysis to the 3,600 most-connected genes because our module construction method and visualization tools cannot handle larger datasets at this point. By definition, module genes are highly connected with the genes of their module (i.e., module genes tend to have relatively high connectivity). Thus, for the purpose of module detection, restricting the analysis to the most-connected genes should not lead to major information loss. Since the network nodes in our analysis correspond to genes as opposed to probesets, we eliminated multiple probes with similar expression patterns for the same gene. Specifically, the 3,600 genes were examined, and where appropriate, gene isoforms and genes containing duplicate probes were excluded by using only those with the highest expression among the redundant transcripts. This final filtering step yielded a count of 3,421 genes for the experimental network construction.

Weighted gene co-expression network construction.

Constructing a weighted co-expression network is critical for identifying modules and for defining the intramodular connectivity. In co-expression networks, nodes correspond to genes, and connection strengths are determined by the pairwise correlations between expression profiles. In contrast to unweighted networks, weighted networks use soft thresholding of the Pearson correlation matrix for determining the connection strengths between two genes. Soft thresholding of the Pearson correlation preserves the continuous nature of the gene co-expression information, and leads to results that are highly robust with respect to the weighted network construction method (Zhang and Horvath 2005). The theory of the network construction algorithm is described in detail elsewhere (Zhang and Horvath, 2005). Briefly, a gene co-expression similarity measure (absolute value of the Pearson product moment correlation) is used to relate every pairwise gene–gene relationship. An adjacency matrix is then constructed using a "soft" power adjacency function $a_{ij} = |cor(x_i, x_j)|^{)}$ where the absolute value of the Pearson correlation measures gene is the co-expression similarity, and a_{ij} represents the resulting adjacency that measures the connection strengths. The network connectivity (k_{all}) of the i-th gene is the sum of the connection strengths with the other genes. This summation performed over all genes in a particular module is the intramodular connectivity (k_{in}). The network satisfies scale-free topology if the connectivity distribution of the nodes follows an inverse power law, (frequency of connectivity p(k) follows an approximate inverse power law in k, i.e., p(k) ~ k^{−q). Zhang and Horvath (2005) proposed a scale-free topology criterion for choosing , which was applied here. In order to make meaningful comparisons across datasets, a power of w =6 was chosen for all analyses. This scale free topology criterion uses the fact that gene co-expression networks have been found to satisfy approximate scale-free topology. Since we are using a weighted network as opposed to an unweighted network, the biological findings are highly robust with respect to the choice of this power. Many co-expression networks satisfy the scale-free property only approximately.

Topological Overlap and Module Detection

A major goal of network analysis is to identify groups, or "modules", of densely interconnected genes. Such groups are often identified by searching for genes with similar patterns of connection strengths to other genes, or high "topological overlap". It is important to recognize that correlation and topological overlap are very different ways of describing the relationship between a pair of genes: while correlation considers each pair of genes in isolation, topological overlap considers each pair of genes in relation to all other genes in the network. More specifically, genes are said to have high topological overlap if they are both strongly connected to the same group of genes in the network (i.e. they share the same "neighborhood"). Topological overlap thus serves as a crucial filter to exclude spurious or isolated connections during network construction (Yip and Horvath 2007). To

calculate the topological overlap for a pair of genes, their connection strengths with all other genes in the network are compared. By calculating the topological overlap for all pairs of genes in the network, modules can be identified. The advantages and disadvantages of the topological overlap measure are reviewed in Yip and Horvath (2007) and Zhang and Horvath (2005).

Definition of the Eigengene

Denote by X the expression data of a given module (rows are genes, columns are microarray samples). First, the gene expression data X are scaled so that each gene expression profile has mean 0 and variance 1. Next, the gene-expression data X are decomposed via singular value decomposition ($X=UDV^T$) and the value of the first module eigengene, V_1, represents the module eigengene. Specifically, V_1 corresponds to the largest singular value.

Consensus module analysis

For this analysis, we started with the 3421 genes (probesets) used in Ghazalpour et al 2006. The genes were the most connected genes among the 8000 most varying genes of the female liver dataset. We filtered out 6 genes that had zero variance in at least one of the four tissue datasets, leaving 3415 genes. For each of the data sets (corresponding to the 4 tissues), the Pearson correlation matrix of the genes was calculated and turned into adjacencies by raising the absolute value to power e =6. From the adjacency matrices, we calculated the TOM similarities which were then used to calculate the consensus dissimilarity. The dissimilarity was used as input in average-linkage hierarchical clustering. Branches of the resulting dendrogram were identified using the Dynamic Tree Cut algorithm (Langfelder et al. 2007). The maximum merging height for the cutting was set to 0.97, and minimum module size to 25. This procedure resulted in 12 initial consensus modules. To determine whether some of the initial consensus modules should be merged, we calculated their eigengenes in each dataset, and formed their correlation matrices (one for each dataset). A ``minimum consensus similarity'' matrix was calculated as the minimum of the dataset eigengene correlation matrices; this matrix was turned into dissimilarity by subtracting it from one and used as input of average-linkage hierarchical clustering again. In the resulting dendrogram of consensus modules, branches with merging height less than 0.25 were identified and modules on these branches were merged. Such branches correspond to modules whose eigengenes have a correlation of 0.75 or higher, which we judge to be close enough to be merged. This module-merging procedure resulted in 11 final consensus modules that are described in the main text.

References

The microarray data and processing steps are described in

- *Ghazalpour A, Doss S, Zhang B, Wang S, Plaisier C, Castellanos R, Brozell A, Schadt EE, Drake TA, Lusis AJ, Horvath S (2006) "Integrating Genetic and Network Analysis to Characterize Genes Related to Mouse Weight". PLoS Genetics. Volume 2 | Issue 8 | AUGUST 2006*

The mouse cross is described in

- *Wang S, Yehya N, Schadt EE, Wang H, Drake TA, et al. (2006) Genetic and genomic analysis of a fat mass trait with complex inheritance reveals marked sex specificity. PLoS Genet 2:e15*

Weighted gene co-expression network analysis is described in

- *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.*

The Dynamic Tree Cut algorithm is described in

- *Peter Langfelder, Bin Zhang and Steve Horvath (2007) Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R, Bioinformatics.*

Other references

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

R code performing the statistical analysis

Downloading the R software:
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 the following packages: `MASS`, `cluster`, `sma`, `impute`, `survival`, and `fields`. Note that some of these libraries may already present in R so there is no need to re-install them.

Download the files:
1) R function file: "NetworkFunctions-Mouse.R", which contains several R functions needed for Network Analysis.
2) The zipped data files and this tutorial.

Unzip all the files into the same directory. The user should copy and paste the following

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

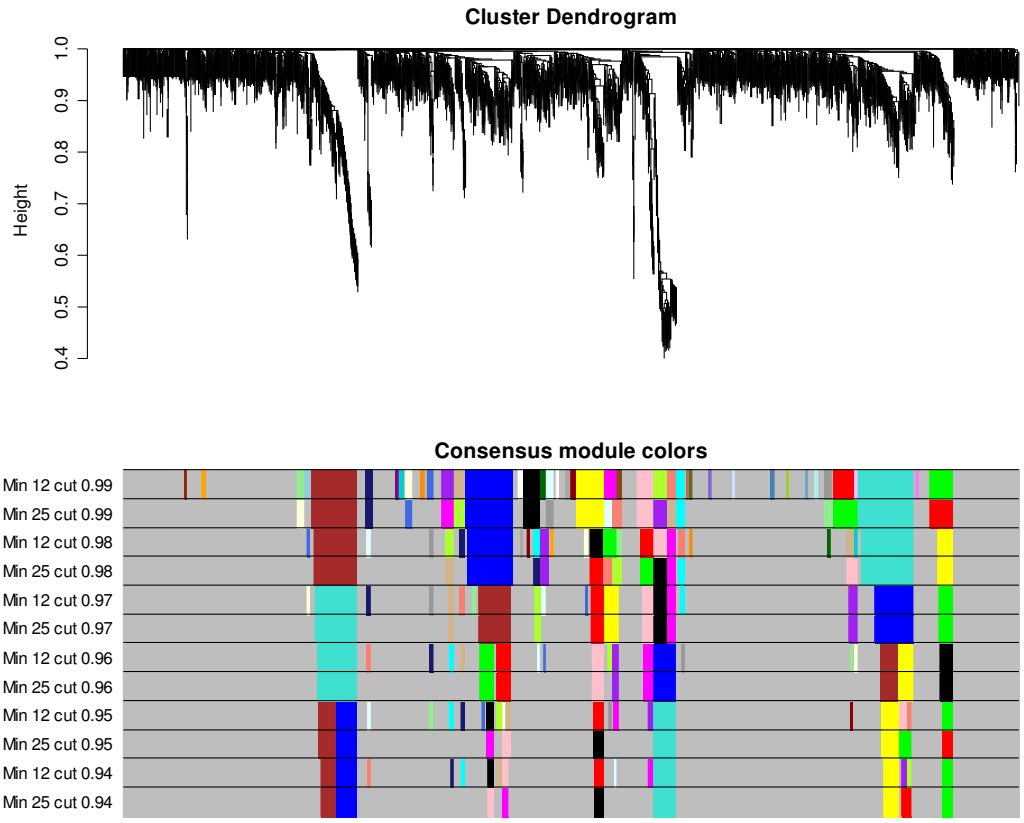
#Absolutely no warranty on the code. Please contact Peter Langfelder and Steve Horvath # with suggestions
Set the working directory of the R session by using the following command (replace the directory # below by your working directory):
`setwd("C:/Documents and Settings/MouseNetwork")`
Note that we use / instead of \ in the path.

```
source("NetworkFunctions-Mouse.R");
set.seed(1);          #needed for .Random.seed to be defined; used in imputation.
options(stringsAsFactors = FALSE);

# Read in the four datasets
data = read.table("cnew_liver_bxh_f2female_8000mvgenes_p3600_UNIQUE_tommodules.xls",
            header=T, strip.white=T, comment.char="")
AllLiverColors = data$module;
data2 = read.csv("BrainFemaleFromLiverFemale3600.csv",
            header=T, strip.white=T, comment.char="")
data2_expr = data2[, c(9:(dim(data2)[2]))];
data3 = read.csv("AdiposeFemaleFromLiverFemale3600.csv",
            header=T, strip.white=T, comment.char="")
data3_expr = data3[, c(9:(dim(data3)[2]))];
data4 = read.csv("MuscleFemaleFromLiverFemale3600.csv",
            header=T, strip.white=T, comment.char="")
data4_expr = data4[, c(9:(dim(data4)[2]))];

# Remove auxiliary data

AuxData = data[,c(1:8,144:150)]
names(AuxData) = colnames(data)[c(1:8,144:150)]
ProbeNames = data[,1];
SampleNames = colnames(data)[9:143];

# General descriptive parameters

No.Sets = 4;
Set.Labels = c("Liver", "Brain", "Adipose", "Muscle");
ModuleMinSize = 25;

PlotOrder = c(2,4,1,3);
PlotRank = c(3,1,4,2);

Set.Labels = Set.Labels[PlotOrder]

# Put the data into a standard "multi set" structure: vector of lists
```

```
ExprData = vector(mode = "list", length = No.Sets);
ExprData[[PlotRank[1]]] = list(data = data.frame(t(data[,-c(1:8,144:150)])));
ExprData[[PlotRank[2]]] = list(data = data.frame(t(data2_expr)));
ExprData[[PlotRank[3]]] = list(data = data.frame(t(data3_expr)));
ExprData[[PlotRank[4]]] = list(data = data.frame(t(data4_expr)));

names(ExprData[[PlotRank[1]]]$data) = ProbeNames;
names(ExprData[[PlotRank[2]]]$data) = data2[, 1];
names(ExprData[[PlotRank[3]]]$data) = data3[, 1];
names(ExprData[[PlotRank[4]]]$data) = data4[, 1];

ExprData = KeepCommonProbes(ExprData, OrderBy = PlotRank[1]);

#Impute zeros into normalized ExprData for NAs

for (set in 1:No.Sets)
{
  ExprData[[set]]$data = scale(ExprData[[set]]$data);
  ExprData[[set]]$data[is.na(ExprData[[set]]$data)] = 0;
}

rm(data, data2, data3, data4, data2_expr, data3_expr, data4_expr);
collect_garbage();

# Read in the clinical trait data

trait_data = read.csv("ClinicalTraits.csv",
                      header=T, strip.white=T, comment.char="");

Gender = trait_data$sex;

AllTraits = trait_data[, -c(31, 16)];
AllTraits = AllTraits[, c(2, 11:36) ];

# Put the traits into a structure resembling the structure of expression data
# For each set only keep traits for the samples that also have expression data

Traits = vector(mode="list", length = No.Sets);
for (set in 1:No.Sets)
{
  SetSampleNames = data.frame(Names = row.names(ExprData[[set]]$data));
  SetTraits = merge(SetSampleNames, AllTraits, by.y = "Mice", by.x = "Names", all = FALSE, sort = FALSE);
  Traits[[set]] = list(data = SetTraits[, -1]);
  row.names(Traits[[set]]$data) = SetTraits[,1 ];
}

rm(AllTraits); rm(SetTraits); rm(SetSampleNames); rm(trait_data);
```

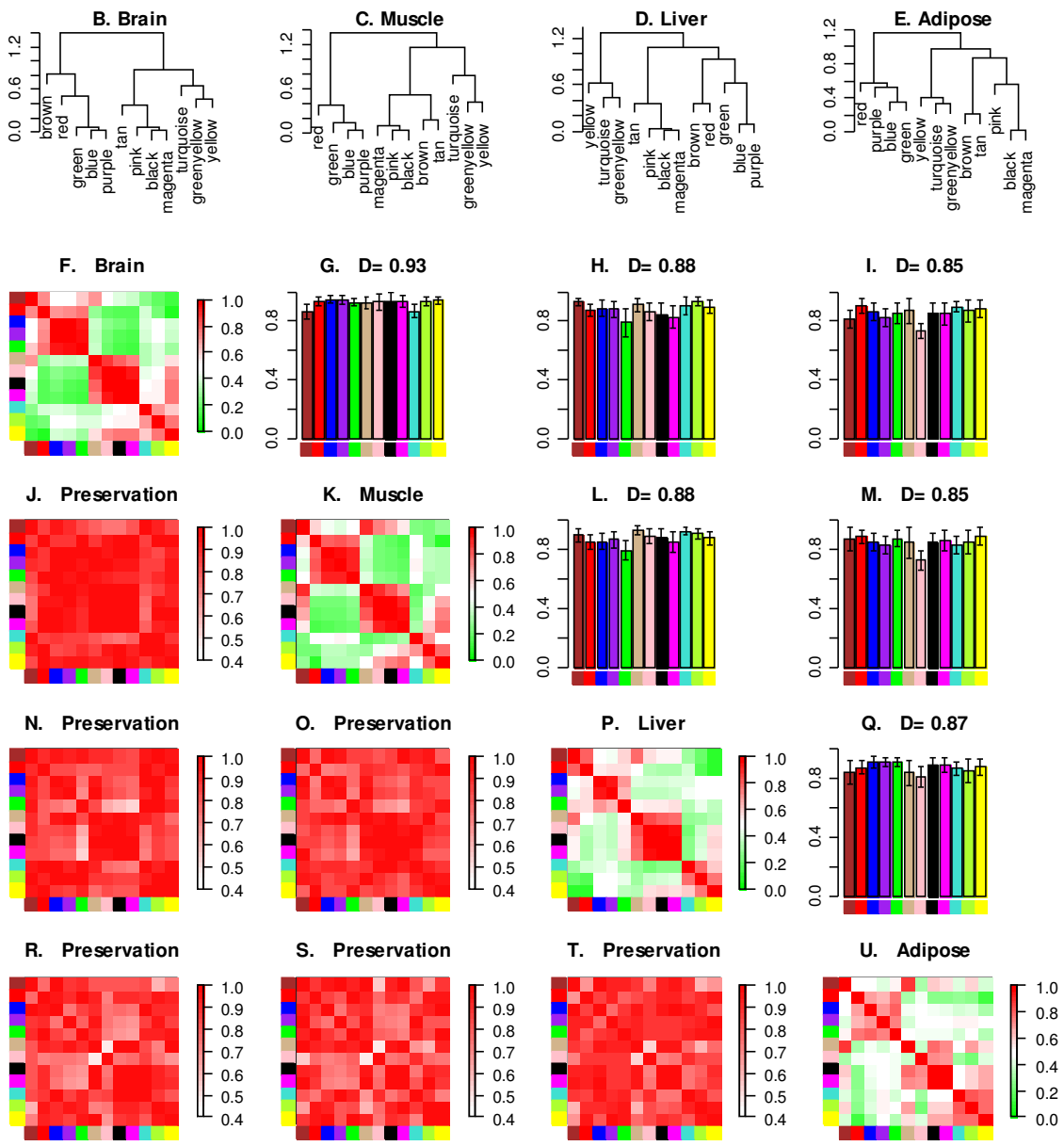
```
collect_garbage();

OutFileBase = "Mouse4Tissue-";
OutDir = ""
PlotDir = ""
FuncAnnoDir = ""
NetworkFile = "Mouse4Tissue-Network.RData";
StandardCex = 1.4;

# Calculate standard weighted gene coexpression networks in all sets

Network = GetNetwork(ExprData = ExprData, DegreeCut = 0,
                BranchHeightCutoff = 0.985, ModuleMinSize = 80, verbose = 4);

#save(Network, file=NetworkFile);
#load(file=NetworkFile);

# The following are the module colors in female of Ghazalpour et al (2006):

LiverColors = AllLiverColors[Network$SelectedGenes];

# Keep only selected the genes (probes) that were selected for the analysis

for (set in 1:No.Sets)
{
  ExprData[[set]]$data = ExprData[[set]]$data[, Network$SelectedGenes];
}

Network$SelectedGenes = rep(TRUE, times = sum(Network$SelectedGenes));


#-----------------------------------------------------------------------------------------
# Calculate and cluster consensus gene dissimilarity

ConsBranchHeightCut = 0.95; ConsModMinSize = ModuleMinSize;
ConsModMinSize2 = 12;

Consensus = IntersectModules(Network = Network,
                        ConsBranchHeightCut = ConsBranchHeightCut, ConsModMinSize = ConsModMinSize,
                        verbose = 4)

# Detect modules in the dissimilarity for several cut heights

No.Genes = sum(Network$SelectedGenes);
ConsCutoffs = seq(from = 0.94, to = 0.99, by = 0.01);
No.Cutoffs = length(ConsCutoffs);
ConsensusColor = array(dim = c(No.Genes, 2*No.Cutoffs));
```

```
for (cut in 1:No.Cutoffs)
{
  ConsensusColor[ ,2*cut-1] = labels2colors(cutreeDynamic(dendro = Consensus$ClustTree,
                              deepSplit = TRUE,
                              cutHeight = ConsCutoffs[cut], minClusterSize=ConsModMinSize,
                              method = "tree"));
  ConsensusColor[ ,2*cut] = labels2colors(cutreeDynamic(dendro = Consensus$ClustTree,
                              deepSplit = TRUE,
                              cutHeight = ConsCutoffs[cut], minClusterSize=ConsModMinSize2,
                              method = "tree"));
}

# Plot found module colors with the consensus dendrogram

SizeWindow(12,9);
par(mfrow = c(2,1));
par(mar=(c(2,7,2,2)+0.1));
plot(Consensus$ClustTree, labels = FALSE);
labels1 = paste("Min", ConsModMinSize, "cut", ConsCutoffs);
labels2 = paste("Min", ConsModMinSize2, "cut", ConsCutoffs);
labels = as.vector(rbind(labels1, labels2));
hclustplotn(Consensus$ClustTree, ConsensusColor,
          main=paste("Consensus module colors"), RowLabels = labels);
```

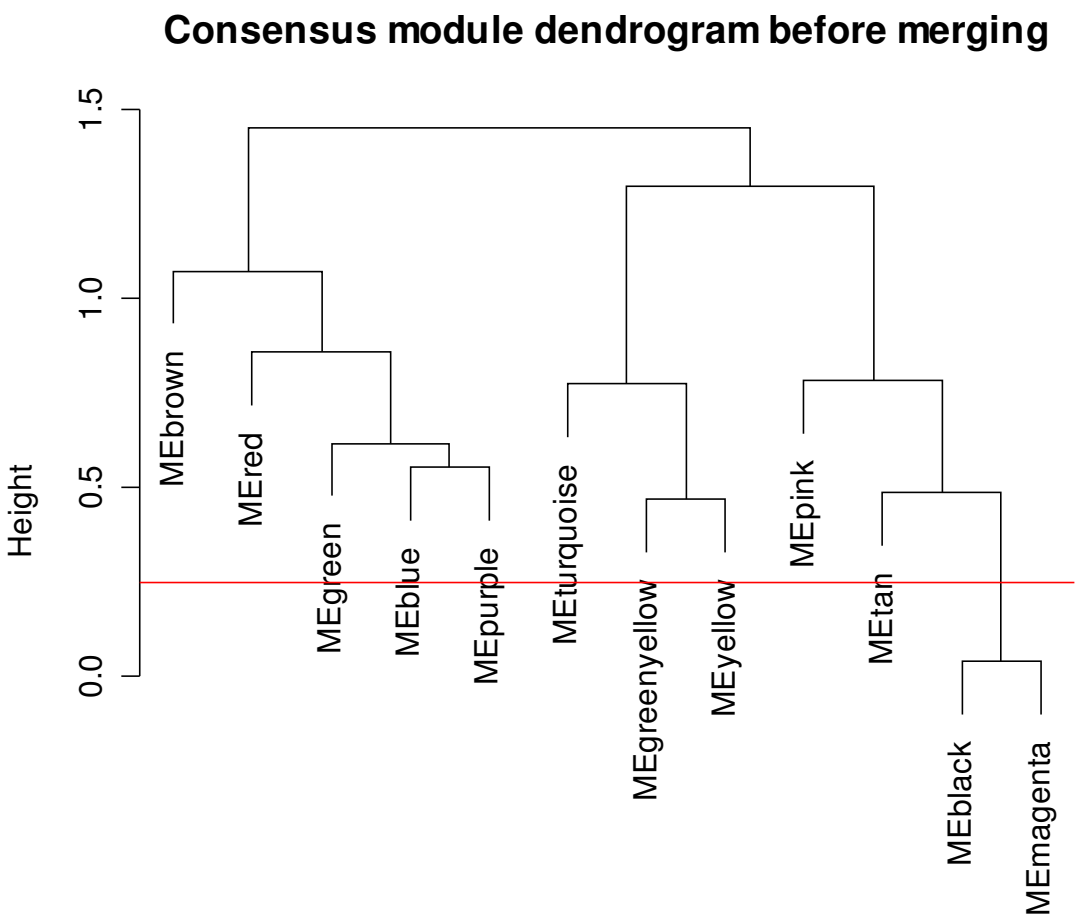



```
# We choose the cutoff height to be 0.97.
```

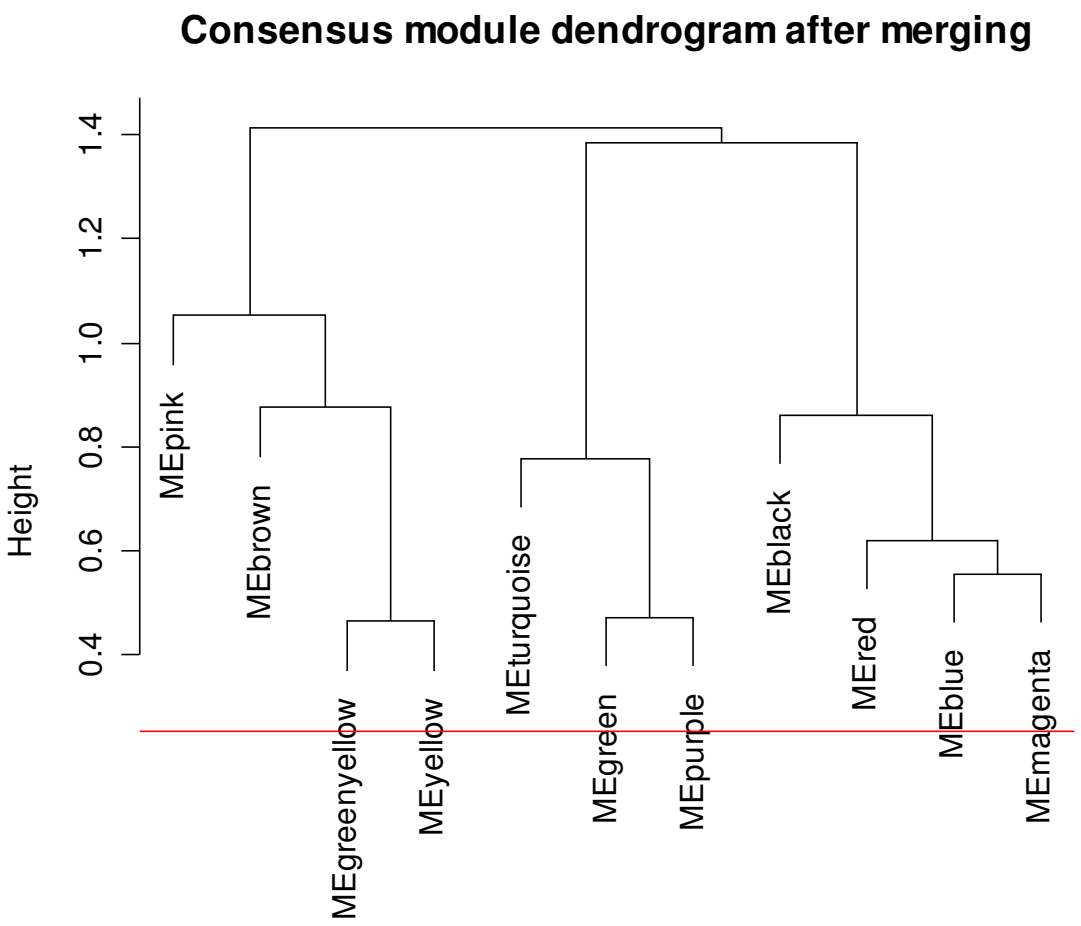
```
ChosenCut = 4;
Consensus$Colors = ConsensusColor[, 2*ChosenCut-1];

# Redo the set module detection using dynamic colors and the same parameters as the consensus, so
# they are comparable.

ModuleMergeCut = 0.25;

MergedSetColors = Network$Colors;
for (set in 1:No.Sets)
{
  Network$Colors[, set] = labels2colors(cutreeDynamic(dendro = Network$ClusterTree[[set]]$data,
                                deepSplit = FALSE,
                                cutHeight = ConsCutoffs[ChosenCut],
                                minClusterSize =1.6 * ConsModMinSize, method = "tree"));
  MergedSetCols = mergeCloseModules(ExprData, colors = Network$Colors[, set], cutHeight =
ModuleMergeCut,
                               useSets = set, useAbs = FALSE, iterate = FALSE, relabel = TRUE,
                               MEs = NULL, getNewMEs = FALSE, verbose = 4, indent = 0);
  MergedSetColors[, set] = MergedSetCols$colors;
  collect_garbage();
}

# Calculate "raw" consensus MEs and plot them

PCs = NetworkModulePCs(ExprData, Network, UniversalModuleColors = Consensus$Colors,
                    verbose=3)
OrderedPCs = OrderPCs(PCs, GreyLast=TRUE, GreyName = "MEgrey");

SizeWindow(9, 9.5);
par(cex = StandardCex/2.0);
PlotCorPCsAndDendros(OrderedPCs, Titles = Set.Labels, ColorLabels = TRUE, IncludeSign = TRUE,
          IncludeGrey = FALSE, plotCPMeasure = FALSE,
          plotMeans = T, CPzlim = c(0.4,1), plotErrors = TRUE, marHeatmap = c(1.6,2.4,2.5,5),
           marDendro = c(1,3,1.2,2), LetterSubPlots = TRUE, Letters = "BCDEFGHIJKLMNOPQRSTUV",
           PlotDiagAdj = TRUE);
```



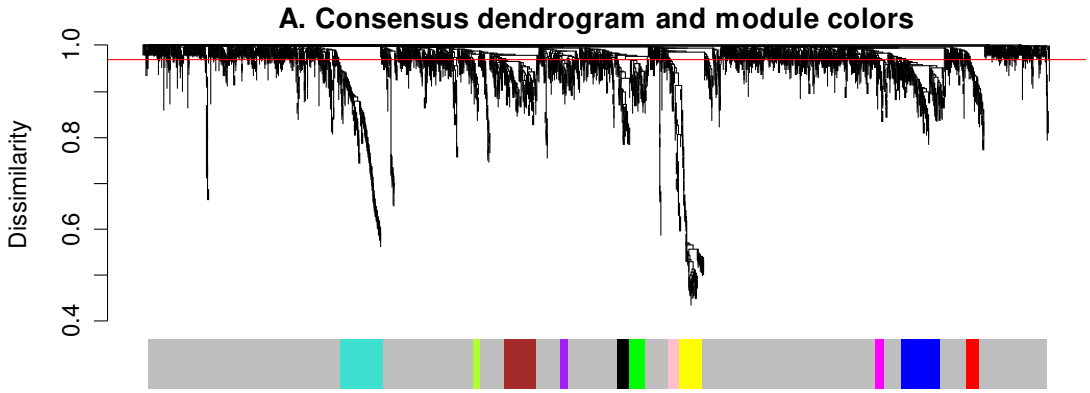

```
MergedColors = MergeCloseModules(ExprData, Network, Consensus$Colors, CutHeight = ModuleMergeCut,
                OrderedPCs = OrderedPCs, IncludeGrey = FALSE, verbose = 4, print.level = 0);

# Plot the detailed consensus dendrogram with the module clustering

SizeWindow(7,7);
par(mfrow = c(1,1));
par(cex = StandardCex/1.2);
plot(MergedColors$ClustTree, main = "Consensus module dendrogram before merging", xlab = "", sub = "");
abline(ModuleMergeCut, 0, col="red");
```



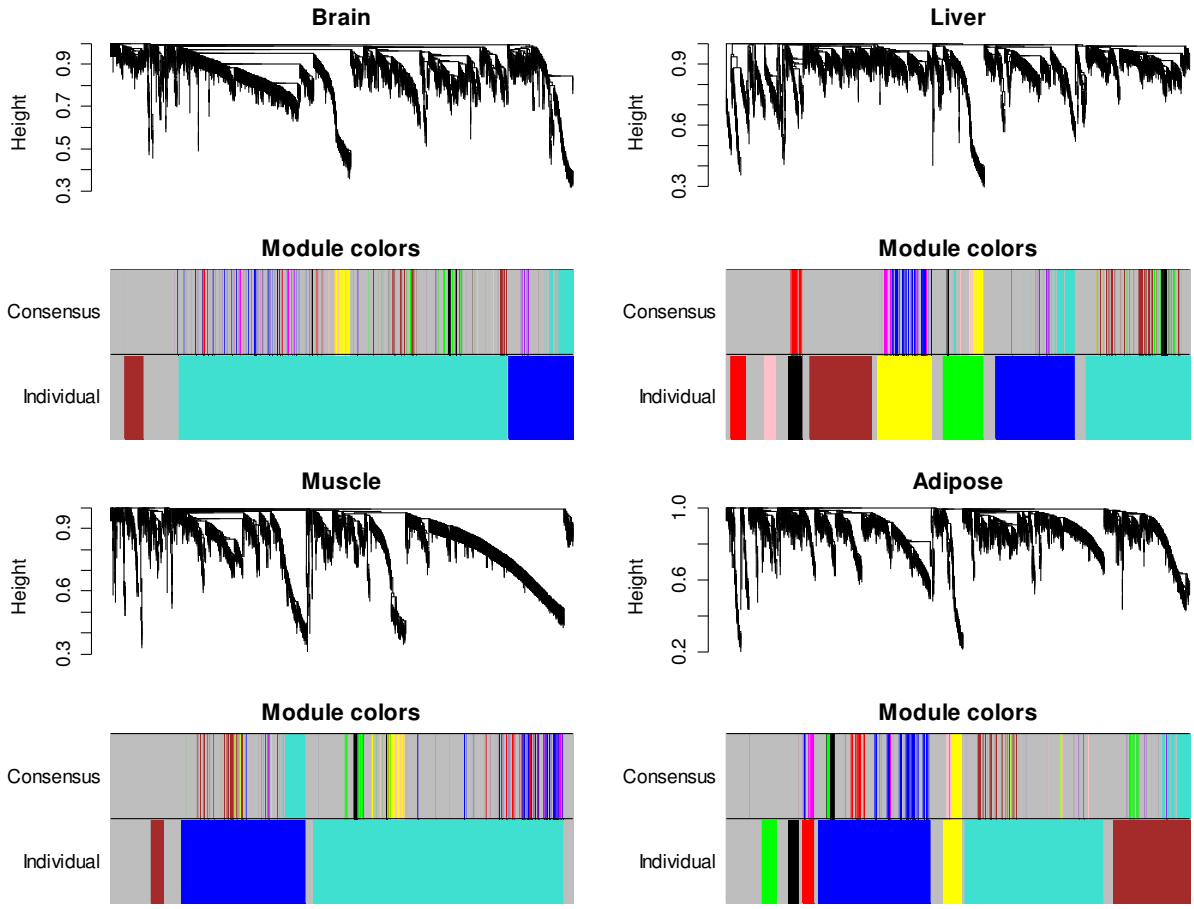

```
MergedColors = MergeCloseModules(ExprData, Network, MergedColors$Colors, CutHeight =
ModuleMergeCut,
                OrderedPCs = NULL, IncludeGrey = FALSE, verbose = 4, print.level = 0);

SizeWindow(7,7);
par(mfrow = c(1,1));
par(cex = StandardCex/1.2);
plot(MergedColors$ClustTree, main = "Consensus module dendrogram after merging", xlab = "", sub = "");
abline(ModuleMergeCut, 0, col="red");
```



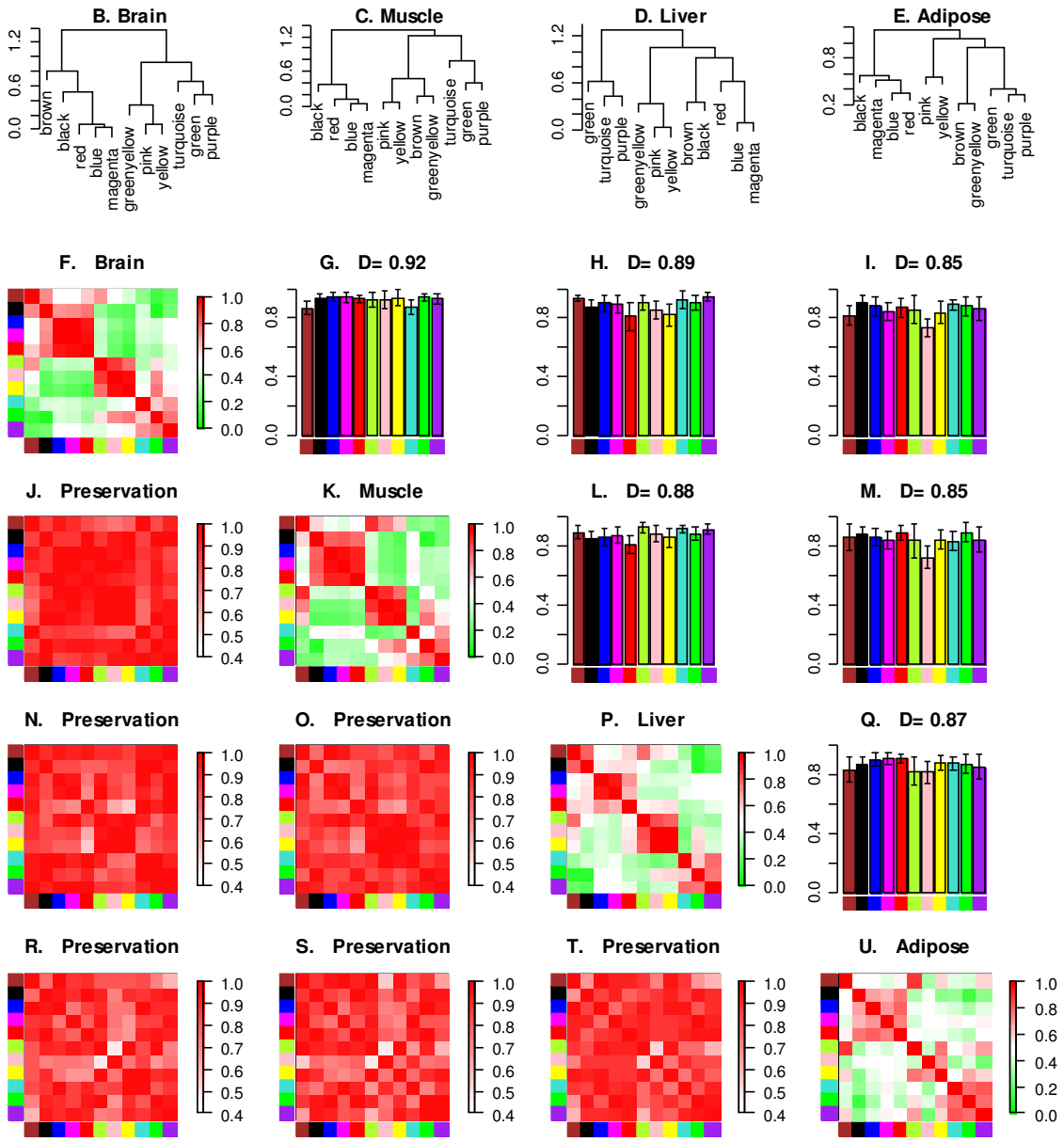

```
SizeWindow(12,4);
lo = layout(matrix(c(1,2), 2, 1, byrow = TRUE), heights = c(0.85, 0.15));
layout.show(lo);
par(cex = StandardCex);

par(mar=c(0,5.2,1.4,1.4));
plot(Consensus$ClustTree, labels = FALSE, main = "A. Consensus dendrogram and module colors", xlab="",
sub="", hang = 0.04, ylab = "Dissimilarity");
abline(ConsCutoffs[ChosenCut], 0, col="red");
par(mar=c(0,5.2,0,1.4)+0.2);

hclustplot1(Consensus$ClustTree, MergedColors$Colors, title1="")
```




```
SizeWindow(12,9);
par(mfcol=c(4,2));
par(mar=c(0.6,4,2,1.2)+0.1);
par(cex = StandardCex/1.4);
for (i in (1:No.Sets))
{
  plot(Network$ClusterTree[[i]]$data,labels=F,xlab="",main=Set.Labels[i],
       ylim=c(0,1), sub="")
  #SCColor = cbind(Network$Colors[, i], MergedSetColors[, i], MergedColors$Colors);   # Compare it to the set
colors
  #RowLabels = c("Set", "Merged set", "Consensus");
  SCColor = cbind(MergedSetColors[, i], MergedColors$Colors);   # Compare it to the set colors
  RowLabels = c("Individual", "Consensus");
  hclustplotn(Network$ClusterTree[[i]]$data, SCColor, RowLabels = RowLabels, cex.RowLabels = 1,
            main="Module colors")
}
```




```
PCs = NetworkModulePCs(ExprData, Network, UniversalModuleColors = MergedColors$Colors,
                       verbose=3)
OrderedPCs = OrderPCs(PCs, GreyLast=TRUE, GreyName = "MEgrey");

# Plot a standard network differential analysis plot

SizeWindow(9, 9.5);
par(cex = StandardCex/2.0);
PlotCorPCsAndDendros(OrderedPCs, Titles = Set.Labels, ColorLabels = TRUE, IncludeSign = TRUE,
          IncludeGrey = FALSE, plotCPMeasure = FALSE,
          plotMeans = T, CPzlim = c(0.4,1), plotErrors = TRUE, marHeatmap = c(1.6,2.4,2.5,5),
           marDendro = c(1,3,1.2,2), LetterSubPlots = TRUE, Letters = "BCDEFGHIJKLMNOPQRSTUV",
           PlotDiagAdj = TRUE);
```



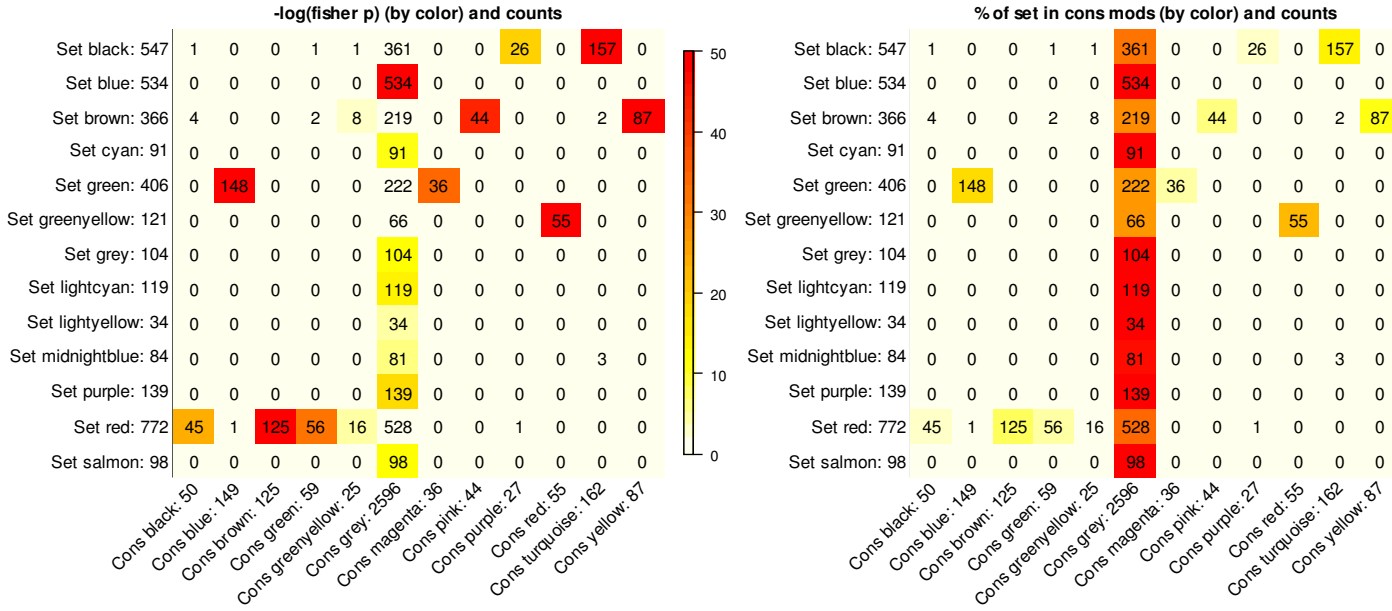

```
for (set in 1:No.Sets)
{
  cr = abs(cor(OrderedPCs[[set]]$data));
  print(paste(Set.Labels[set], ": Density(abs(cor(PCs))) =", mean(cr[upper.tri(cr)])));
}
```

```
[1] "Brain : Density(abs(cor(PCs))) = 0.43262752237385"
[1] "Muscle : Density(abs(cor(PCs))) = 0.435800047878673"
[1] "Liver : Density(abs(cor(PCs))) = 0.341731485173187"
[1] "Adipose : Density(abs(cor(PCs))) = 0.306419452946620"
```

```
#-------------------------------------------------------------------------------
# Attempt to assign consensus and liver set modules to one another
```

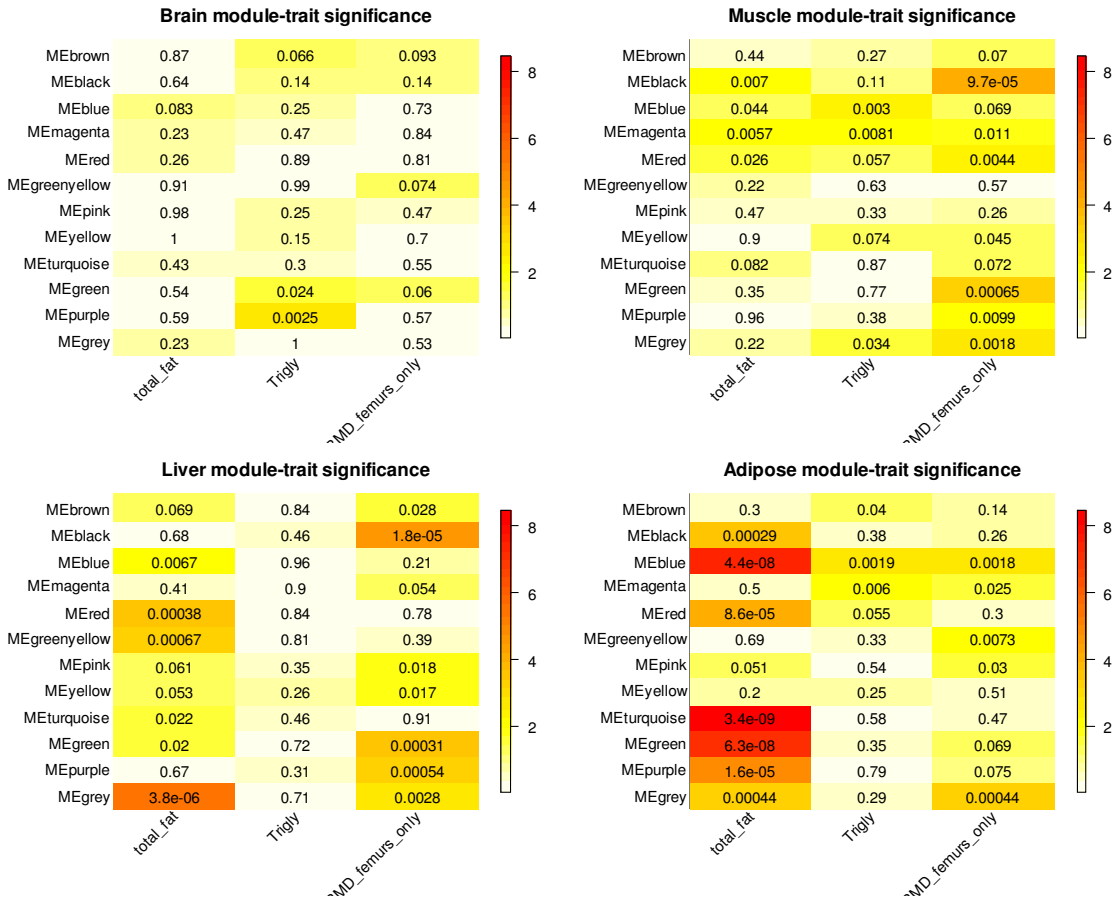
```
#SetModColors = as.factor(Network$Colors[, 1]);
SetModColors = as.factor(LiverColors);
No.SetMods = nlevels(SetModColors);
ConsModColors = as.factor(MergedColors$Colors);
No.ConsMods = nlevels(ConsModColors);

pTable = matrix(0, nrow = No.SetMods, ncol = No.ConsMods);
CountTbl = matrix(0, nrow = No.SetMods, ncol = No.ConsMods);

for (smod in 1:No.SetMods)
  for (cmod in 1:No.ConsMods)
  {
    SetMembers = (SetModColors == levels(SetModColors)[smod]);
    ConsMembers = (ConsModColors == levels(ConsModColors)[cmod]);
    pTable[smod, cmod] = -log10(fisher.test(SetMembers, ConsMembers, alternative = "greater")$p.value);
    CountTbl[smod, cmod] = sum(SetModColors == levels(SetModColors)[smod] & ConsModColors ==
                levels(ConsModColors)[cmod])
  }

pTable[is.infinite(pTable)] = 1.3*max(pTable[is.finite(pTable)]);
pTable[pTable>50 ] = 50 ;

PercentageTbl = CountTbl;
for (smod in 1:No.SetMods)
  PercentageTbl[smod, ] = as.integer(PercentageTbl[smod, ]/sum(PercentageTbl[smod, ]) * 100);


SetModTotals = as.vector(table(SetModColors));
ConsModTotals = as.vector(table(ConsModColors));

SizeWindow(12,5);
par(mfrow=c(1,2));
par(cex = StandardCex/2.0);
par(mar=c(8,10,2,2)+0.3);

HeatmapWithTextLabels(Matrix = pTable,
                xLabels = paste("Cons ", levels(ConsModColors), ": ",
                        ConsModTotals, sep=""),
                yLabels = paste("Set ", levels(SetModColors), ": ", SetModTotals,
                        sep=""),
                NumMatrix = CountTbl,
                InvertColors = TRUE, SetMargins = FALSE,
                main = "-log(fisher p) (by color) and counts",
                cex.Num = 0.8, cex.lab = 0.8);

par(cex = StandardCex/2.0);
par(mar=c(8,10,2,2)+0.3);
```

```
HeatmapWithTextLabels(Matrix = PercentageTbl,
                      xLabels = paste("Cons ", levels(ConsModColors), ": ",
                                      ConsModTotals, sep=""),
                      yLabels = paste("Set ", levels(SetModColors), ": ", SetModTotals,
                                      sep=""),
                      NumMatrix = CountTbl,
                      InvertColors = TRUE, SetMargins = FALSE,
                      main = "% of set in cons mods (by color) and counts", zlim = c(0,100),
                      cex.Num = 0.8, cex.lab = 0.8);
```




```
#-----------------------------------------------------------------------------------
# Add traits; see whether there are any traits associated with any of the consensus modules.

# If necessary, recalculate PCs

PCs = NetworkModulePCs(ExprData, Network, UniversalModuleColors = MergedColors$Colors,
                       verbose=3)
OrderedPCs = OrderPCs(PCs, GreyLast=TRUE, GreyName = "MEgrey");

# Select interesting traits

SelTraits = SelectTraits(Traits, BranchCut = 0.25, Impute = TRUE,
                         SelectOnSignificance = TRUE, PCs = OrderedPCs,
                         SignifThres = 0.30, verbose=1);

# Calculate module--trait significance
No.SelTraits = SelTraits$No.SelectedTraits;
TraitSignif = vector(mode="list", length = No.Sets);
```

```
No.Mods = dim(OrderedPCs[[1]]$data)[2];

for (set in 1:No.Sets)
{
  TraitSignif[[set]] = list(data = matrix(0, nrow = No.Mods, ncol = No.SelTraits));
  for (mod in 1:No.Mods)
    for (trait in 1:No.SelTraits)
    {
      ct = cor.test(OrderedPCs[[set]]$data[, mod], SelTraits$Traits[[set]]$data[, trait]);
      TraitSignif[[set]]$data[mod, trait] = ct$p.value;
    }
}

# Plot the module--trait significance

SizeWindow(14,11);
par(mfrow = c(2,2));
par(mar = c(5,8,3,3));

minp = 1; maxp = 0;
for (set in 1:No.Sets)
{
  minp = min(minp, TraitSignif[[set]]$data);
  maxp = max(maxp, TraitSignif[[set]]$data);
}

for (set in 1:No.Sets)
{
  m = -log10(TraitSignif[[set]]$data);
  HeatmapWithTextLabels(Matrix = m, xLabels = names(SelTraits$Traits[[1]]$data),
                        yLabels = names(OrderedPCs[[1]]$data), ColorLabels = FALSE,
                        NumMatrix = signif(TraitSignif[[set]]$data,2),
                        cex.Num = StandardCex/1.4, cex.lab = StandardCex/1.4,
                        InvertColors = TRUE, SetMargins = FALSE,
                        main = paste(Set.Labels[set], "module-trait significance"),
                        zlim = c(-log10(maxp), -log10(minp)));
}
```



None of the traits is related significantly to any of the module eigengenes in all four tissues.