Eigengene Network Analysis of Human and Chimpanzee Microarray Data 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 two datasets. The two data sets correspond to gene expression measurements in human and chimpanzee brains. The R code allows to reproduce the Figures and tables reported in Langfelder 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 reference

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

The microarray data and processing steps are described in

- *Oldham M, Horvath S, Geschwind D (2006) Conservation and Evolution of Gene Co-expression Networks in Human and Chimpanzee Brains. PNAS. Nov 21;103(47):17973-8*

This tutorial and the data files can be found at the webpage
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/

The unprocessed data came originally from Khaitovich et al (2004). To facilitate comparison with the original analysis, we use the microarray normalization procedures and gene selection procedures described in Oldham et al (2006).

Analysis of microarray data

The following description is taken from Oldham et al 2006. The dataset used for network construction consisted of 36 Affymetrix HGU95Av2 microarrays surveying gene expression with 12,625 probe sets in three adult humans and three adult chimpanzees across six matched brain regions: Broca's area, anterior cingulate cortex, primary visual cortex, prefrontal cortex, caudate nucleus, and cerebellar vermis. Because the arrays used in this study were designed for human mRNA sequences, it is necessary to account for the effect of interspecies sequence differences on chimpanzee gene-expression values. All probes on the array were compared against the human genome (Build 34) and the chimpanzee draft genome using MegaBlast (http://www.ncbi.nlm.nih.gov/BLAST/). Any probe without a perfect match in both species (approximately 1/4) was masked during the calculation of expression values (GCOSv1.2, Affymetrix, Inc). In addition, only probe sets with six or more matching probes were retained for subsequent analyses (n=11,768/12,625). For each array, expression values were scaled to an average intensity of 200 (GCOSv1.2, Affymetrix, Inc.). Quantile normalization was then performed and inter-array correlations were calculated using R. Following quantile normalization, the average inter-array correlation was 0.924 among all 18 human arrays and 0.937 among all 18 chimpanzee arrays. In specific instances in which the composition of this dataset was altered, the removal of a brain region(s) took place prior to quantile normalization.

For computational reasons, network analysis was limited to 4000 probe sets. (Note: although some genes are represented by multiple probe sets and other probe sets are not fully annotated, for consistency we refer to probe sets as "genes" throughout the manuscript, unless otherwise noted.) In order to enrich this subset with genes likely to play important roles in the brain, a non-neural "filter" was applied to identify genes with greater variance in neural versus non-neural tissue. A publicly available microarray dataset was obtained for human lung (3). The human lung dataset consisted of 18 Affymetrix HGU95Av2 microarrays measuring gene expression in normal adult human lung tissue, and was normalized as described above. The same probe mask file that was used for the human and chimpanzee brain datasets was also applied to the human lung dataset. After scaling all arrays to the same average intensity (200), the mean inter-array correlation for one human lung array (CL2001032718AA) was 2.91 SD below the average and was removed from the dataset. Following quantile normalization in R, the average inter-array correlation for the human lung dataset was 0.925. For each probe set on the microarray, the variance was calculated in the non-neural (lung) and neural (brain) datasets for the human samples. All probe sets were ranked according to their variance in both the neural and non-neural datasets, and each probe set's rank in the neural dataset was subtracted from its rank in the non-neural dataset. These rank differences (sorted from high to low) were used to select the top 4000 probe sets among all probe sets showing greater variance in human brain than human lung. In cases where <4000 probe sets showed greater variance in brain than lung, the top 4000 probe sets were selected solely on the basis of their rank differences between the neural and non-neural datasets.

Weighted gene co-expression network construction.

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. 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^{−p
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 i =9 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).

Consensus module detection

Consensus module detection is similar to the individual dataset module detection in that it uses hierarchical clustering of genes according to a measure of gene dissimilarity. We use a gene-gene consensus dissimilarity measure Dissim(i,j) as input of average linkage hierarchical clustering. We define modules as branches of the dendrogram. To cut-off branches we use a fixed height cut-off. Modules must contains a minimum number n_0 of genes. Module detection proceeds along the following steps:
(1) perform a hierarchical clustering using the consensus dissimilarity measure;
(2) cut the clustering tree at a fixed height cut-off;
(3) each cut branch with at least n_0 genes is considered a separate module;

(4) all other genes are considered unassigned and are colored in ``grey''.
The resulting modules will depend to some degree on the cut height and minimum size.

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. This definition is equivalent to defining the module eigengene as the first principal component of cor(t(X)), i.e. the correlation matrix of the gene expression data.

Consensus module analysis

Our gene selection criteria were identical to those of Oldham et al 2006, described above. Specifically, of the 4000 genes (Affymetric microarray probesets) expressed in the brain samples, we selected the ones whose scaled connectivity (that is, connectivity normalized by the connectivity of the most connected gene) is at least 0.1 in both human and chimp samples. The Pearson correlation matrices for each dataset were turned into adjacencies by raising their absolute values to power e =9; the adjacencies were used to compute the TOM similarities in each dataset. The consensus dissimilarity was computed as minimum TOM across the two data sets. We used the consensus dissimilarity as input of average-linkage hierarchical clustering. Branches of the resulting dendrogram were identified using a fixed-height cut of 0.96 and minimum module size of 25. To determine whether some of the 9 detected initial modules were too similar, we calculated their eigengenes in each dataset, and formed their correlation matrices (one for each dataset). A consensus eigengene similarity matrix was calculated as the minimum of the data set specific 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 7 final consensus modules that are described in our main text.

References

- *Oldham M, Horvath S, Geschwind D (2006) Conservation and Evolution of Gene Co-expression Networks in Human and Chimpanzee Brains. PNAS. Nov 21;103(47):17973-8*
- *Khaitovich, P., Muetzel, B., She, X., Lachmann, M., Hellmann, I., Dietzsch, J., Steigele, S., Do, H. H., Weiss, G., Enard, W., Heissig, F., Arendt, T., Nieselt-Struwe, K., Eichler, E. E. & Paabo, S. (2004) Genome Res 14, 1462-73*
- *Zhang, B. & Horvath, S. (2005) Statistical Applications in Genetics and Molecular Biology 4, Article 17.*

- *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.

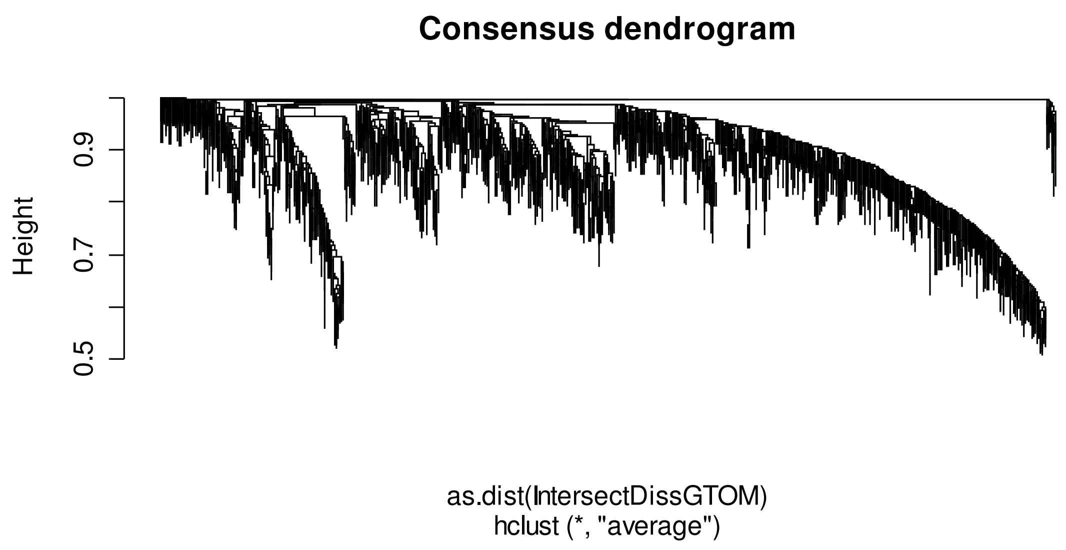
Unzip all the files into the same directory. The user should copy and paste the following script into the R session. Text after "#" is a comment and is automatically ignored by R.

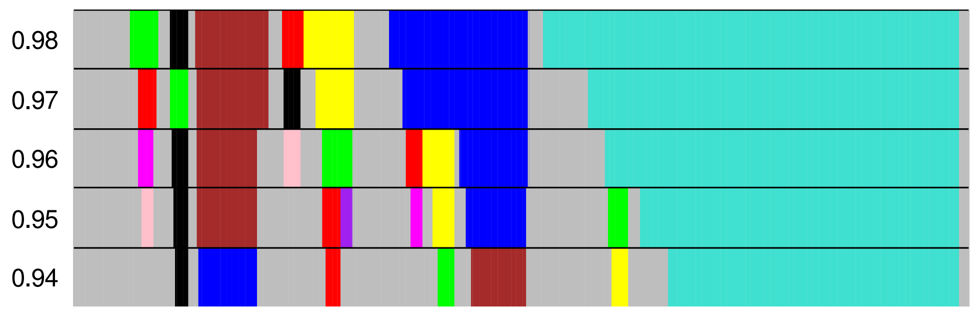
#Absolutely no warranty on the code. Please contact Peter Langfelder and Steve Horvath # with suggestions

# Set the working directory of the R session by using the following command:
```
setwd(“C:/Documents and Settings/HumanChimpNetwork”)
```
# Note that we use / instead of \ in the path.

```
source("NetworkFunctions-HumanChimp.R");

options(stringsAsFactors = FALSE);

data=read.csv("Dataset 1 (network construction).csv", header=T, as.is=T,
strip.white=TRUE)

ProbeNames = data[,1]
SampleNames = colnames(data)[-1]
ExprData = data.frame(t(data[data$Brain_variant_H>0,2:37]))
AuxData = data.frame(t(data[data$Brain_variant_H>0,38:39]))

names(ExprData) = ProbeNames[data$Brain_variant_H>0]
names(AuxData) = ProbeNames[data$Brain_variant_H>0]


# Note - in the data file, chimp is first and human second - we want  them the other
way around.

Subsets = c(rep(2, times=18), rep(1, times=18))
No.Sets = 2;
Set.Labels = c("Human", "Chimp");
SoftPower = 9;
ModuleMinSize = 25;
BranchHeightCutoff = 0.95;
No.Samples = length(Subsets);
No.SelectedGenes = 2500;

KeepOverlapSign = FALSE;

OutFileBase = "Plot-HC-";
PlotDir = ""
FuncAnnoDir = ""
NetworkFile = "HumanChimp-Network.RData";
StandardCex = 1.4;

# Calculate gene co-expression networks in each set

Connectivity = GetConnectivity(ExprData, Subsets, SoftPower=SoftPower, verbose=2);

ScaledConnectivity = Connectivity;
ScaledConnectivity[,1] = Connectivity[,1]/max(Connectivity[,1])
ScaledConnectivity[,2] = Connectivity[,2]/max(Connectivity[,2])
# Now repeating the same criteria that Mike Oldham and co. used:

minconnections=.1

SelectedGenes = ScaledConnectivity[,1]>minconnections | ScaledConnectivity[,
2]>minconnections

print(table(SelectedGenes));

Modules = GetModules(ExprData, Subsets, SelectedGenes, SoftPower,
                     BranchHeightCutoff, ModuleMinSize, KeepOverlapSign =
KeepOverlapSign, verbose = 2);

Network = list(Subsets = Subsets, SoftPower = SoftPower, Connectivity = Connectivity,
                 SelectedGenes = SelectedGenes, Dissimilarity = Modules
$Dissimilarity,
                 DissimilarityLevel = Modules$DissimilarityLevel,
                 ClusterTree = Modules$ClusterTree, Colors = Modules$Colors,
                 BranchHeightCutoff = Modules$BranchHeightCutoff,
                 ModuleMinSize = Modules$ModuleMinSize);

rm(Modules); collect_garbage();
```

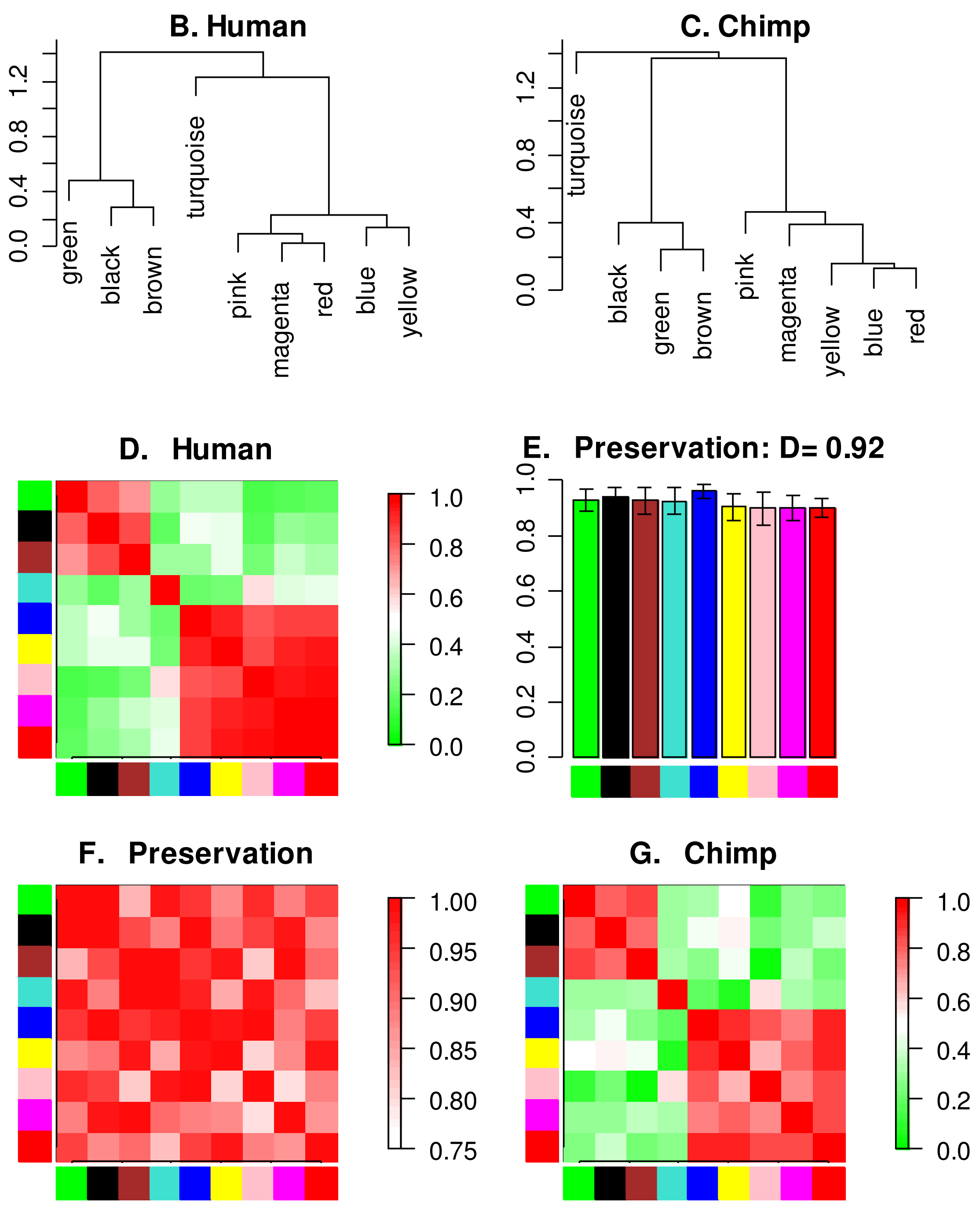
```
#save(Network, file=NetworkFile);
#load(file=NetworkFile);

#--------------------------------------------------------------------------------------
-----------
# Calculate consensus modules

ConsBranchHeightCut = 0.95; ConsModMinSize = ModuleMinSize;

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

# Detect modules in the calculated consensus dissimilarity for 6 different height
cuts

No.Genes = sum(Network$SelectedGenes);
ConsCutoffs = seq(from = 0.94, to = 0.98, by = 0.01);
No.Cutoffs = length(ConsCutoffs);
ConsensusColor = array(dim = c(No.Genes, No.Cutoffs));
for (cut in 1:No.Cutoffs)
{
  ConsensusColor[, cut] = as.character(modulecolor2(Consensus$ClustTree,
                                ConsCutoffs[cut],30));
}

# Plot a preliminary consensus dendrogram with the detected colors

par(mfrow = c(2,1));
plot(Consensus$ClustTree, labels = FALSE, ylim=c(0,2), main= "Consensus dendrogram");
str_cutoffs = paste(ConsCutoffs, collapse = ", ");
hclustplotn(Consensus$ClustTree, ConsensusColor,
RowLabels=as.character(ConsCutoffs));
```



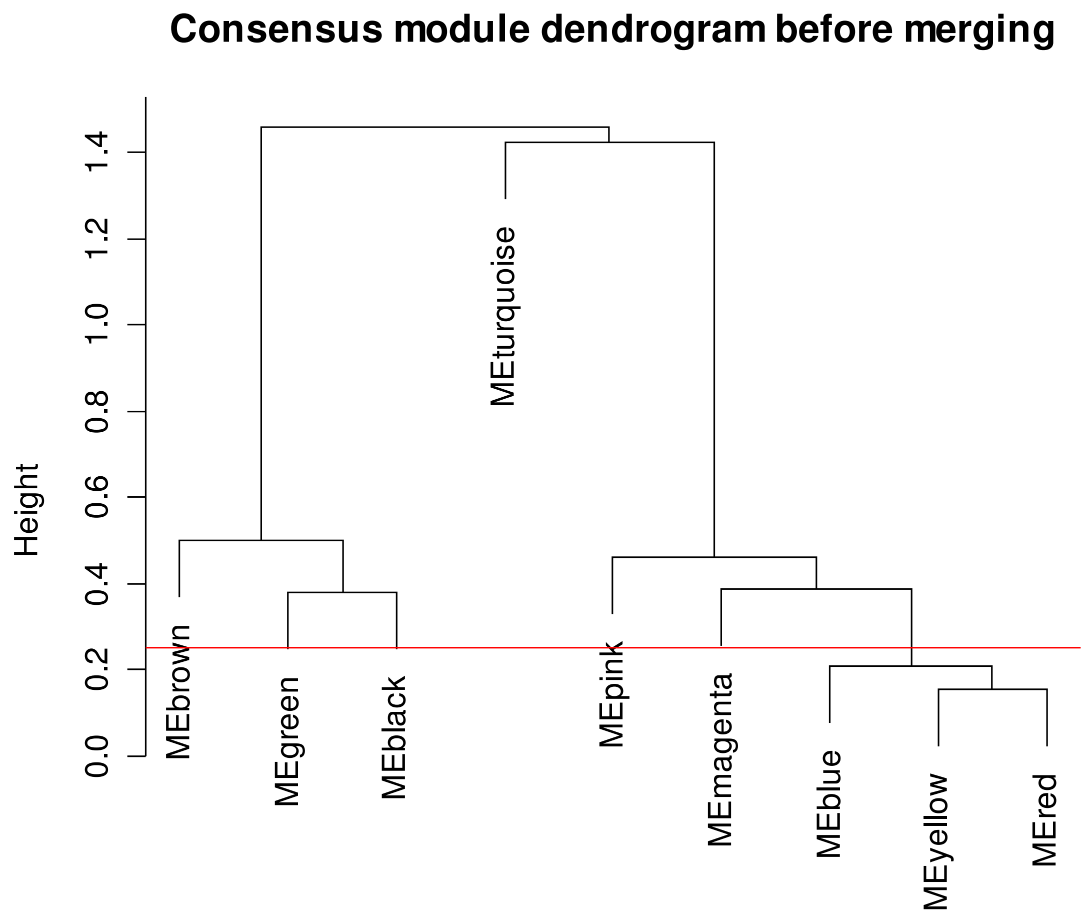


```
#dev.copy2eps(file=paste(PlotDir, OutFileBase, "ConsDendr-raw.eps",sep=""));

# We pick the dendrogram cut height to be 0.96.

ChosenCut = 3;
Consensus$Colors = as.character(modulecolor2(Consensus$ClustTree,
                                ConsCutoffs[ChosenCut],30));

#Calculate consensus module principal components (consensus eigengenes) in both sets

PCs = NetworkModulePCs(ExprData, Network, UniversalModuleColors = Consensus$Colors,
                       verbose=3)

# Order module eigengenes (MEs, also referred as Principal Components, PCs)

OrderedPCs = OrderPCs(PCs, GreyLast=TRUE, GreyName = "MEgrey");
```

```
# Plot a matrix of network and preservation plots

SizeWindow(7.5,9);
par(cex = StandardCex/1.4);
par(mar = c(2,2.0,2.2,5));
PlotCorPCsAndDendros(OrderedPCs, Titles = Set.Labels, ColorLabels = TRUE, IncludeSign
= TRUE,
           IncludeGrey = FALSE, setMargins = FALSE, plotCPMeasure = FALSE,
            plotMeans = TRUE, CPzlim = c(0.75,1), plotErrors = TRUE, marHeatmap =
c(2,3,2.5,5),
            marDendro = c(1,3,1.2,2), LetterSubPlots = TRUE, Letters = "BCDEFGHIJKL",
            PlotDiagAdj = TRUE);
```

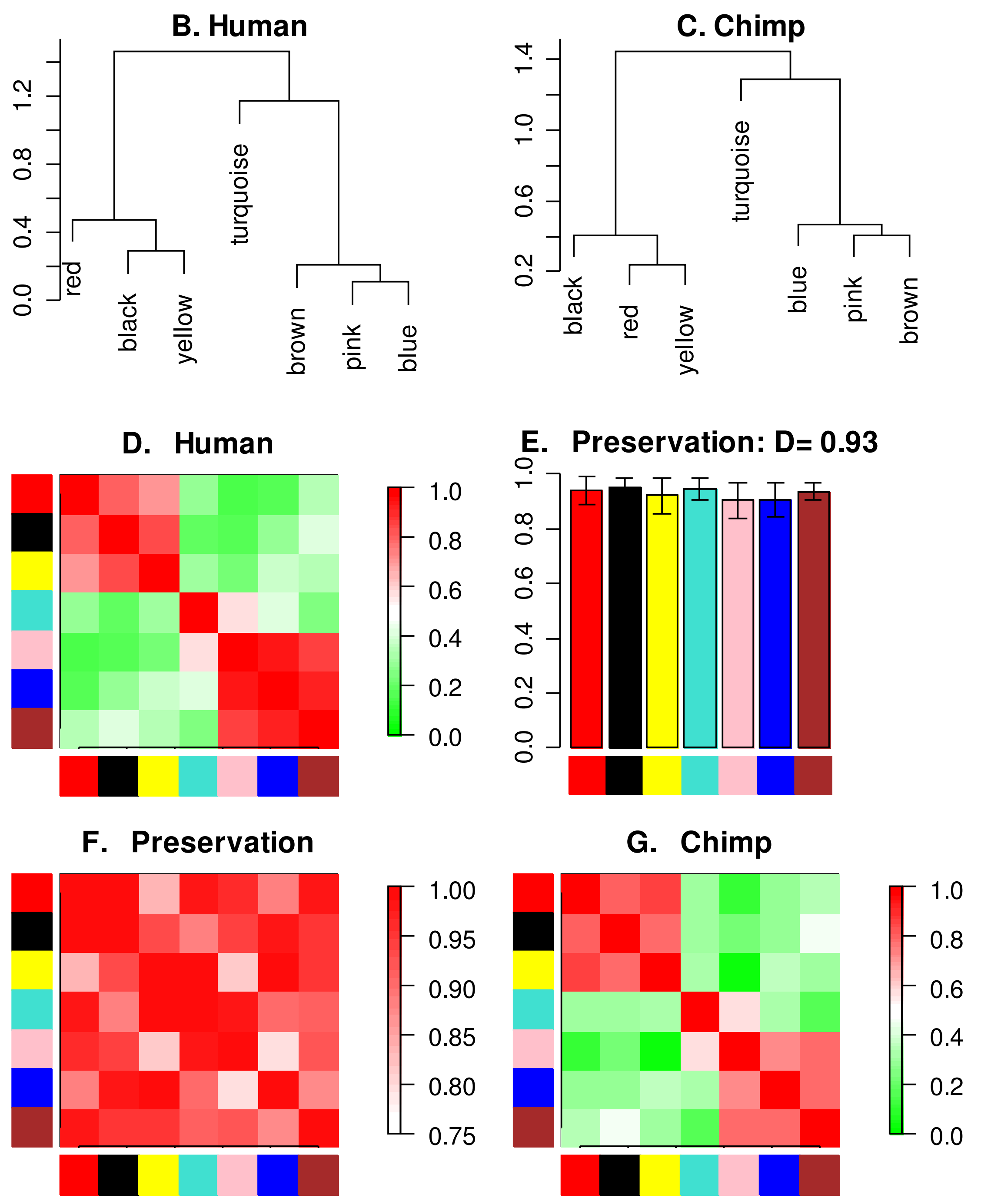


```
#dev.copy2eps(file=paste(PlotDir, OutFileBase, "PCCor-raw.eps",sep=""));

#--------------
# Merge modules whose eigengenes are too close:
# Cluster the found module eigengenes and merge ones that are too close to one
another _in both sets_.

ModuleMergeCut = 0.25;
MergedColors = MergeCloseModules(ExprData, Network, Consensus$Colors, CutHeight =
ModuleMergeCut,
                        OrderedPCs = OrderedPCs, UseAbs = 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");
```

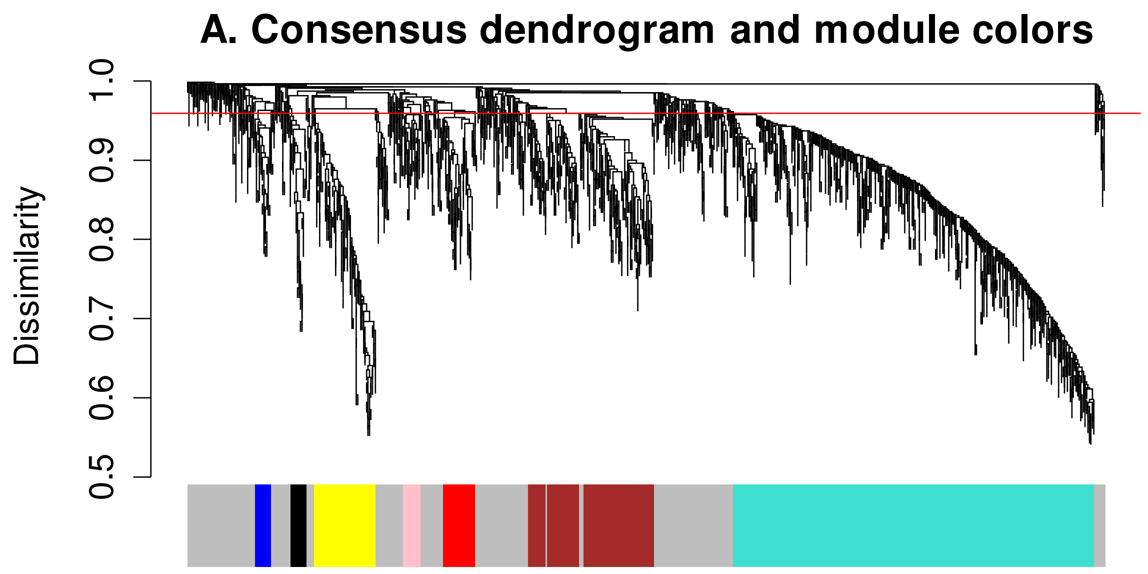


```
# dev.copy2eps(file=paste(PlotDir, OutFileBase, "ModuleDendr-raw.eps",sep=""));

# Cluster again? Copy - paste this until the function reports no merging of branches.
```

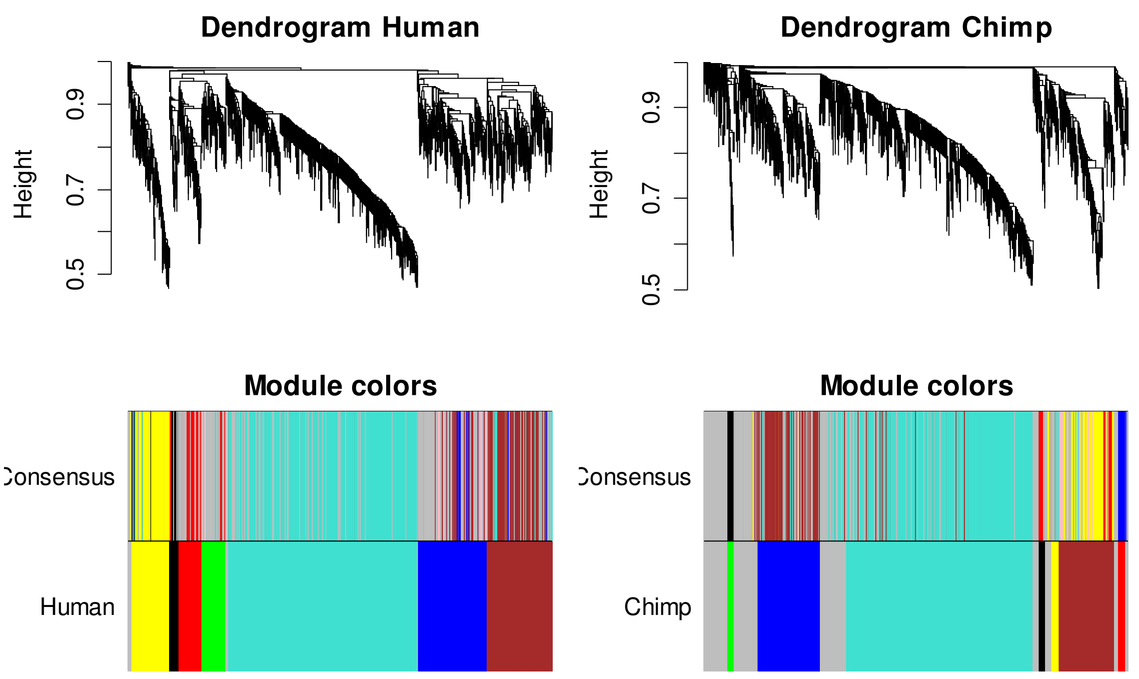
```
MergedColors = MergeCloseModules(ExprData, Network, MergedColors$Colors, CutHeight =
ModuleMergeCut,
                       OrderedPCs = OrderedPCs, UseAbs = FALSE, verbose = 4,
print.level = 0);

# Check the results...

No.Mods = nlevels(as.factor(MergedColors$Colors));
ModColors = levels(as.factor(MergedColors$Colors));

table(as.factor(MergedColors$Colors));

# Relabel consensus colors to match Oldham et al's human colors...

Colors2 = MergedColors$Colors;
MergedColors$Colors[Colors2=="black"] = "blue";
MergedColors$Colors[Colors2=="blue"] = "brown";
MergedColors$Colors[Colors2=="brown"] = "yellow";
MergedColors$Colors[Colors2=="green"] = "black";
MergedColors$Colors[Colors2=="red"] = "pink";
MergedColors$Colors[Colors2=="yellow"] = "red";

# Recalculate module principal components

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

# Plot a differential analysis plot of networks
SizeWindow(7.5,9);
par(cex = StandardCex/1.4);
par(mar = c(2,2.0,2.2,5));
PlotCorPCsAndDendros(OrderedPCs, Titles = Set.Labels, ColorLabels = TRUE, IncludeSign
= TRUE,
           IncludeGrey = FALSE, setMargins = FALSE, plotCPMeasure = FALSE,
            plotMeans = TRUE, CPzlim = c(0.75,1), plotErrors = TRUE, marHeatmap =
c(2,3,2.5,5),
            marDendro = c(1,3,1.2,2), LetterSubPlots = TRUE, Letters = "BCDEFGHIJKL",
            PlotDiagAdj = TRUE);
```



```
#dev.copy2eps(file=paste(PlotDir, OutFileBase, "PCCor.eps",sep=""));

# Plot the new module membership vs. the consensus dendrogram. This figure is used in
the
# paper.

SizeWindow(8,4);
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(0,4.2,2.2,0.2));
plot(Consensus$ClustTree, labels = FALSE, main = "A. Consensus dendrogram and module
colors", xlab="",
sub="", ylab = "Dissimilarity", hang = 0.03);
abline(ConsCutoffs[ChosenCut], 0, col="red");
par(mar=c(0.2,4.2,0,0.2));
hclustplot1(Consensus$ClustTree, MergedColors$Colors, title1 = "");
```



```
#dev.copy2eps(file=paste(PlotDir, OutFileBase, "ConsDendr-ConsOnly.eps",sep=""));
#dev2bitmap(file = paste(PlotDir, OutFileBase, "ConsDendr-ConsOnly.pdf",sep=""),
width = 8, type =
#"pdfwrite");

# Plot the new module membership vs. set dendrograms

SizeWindow(13,8);
par(mfcol=c(2,2));
par(mar=c(1.4,4,2,0.3)+0.1);
par(cex = StandardCex);
for (i in (1:No.Sets))
{
  plot(Network$ClusterTree[[i]]$data,labels=F,xlab="",main=paste("Dendrogram",
Set.Labels[i]),
          ylim=c(0,1), sub="")
  SCColor = cbind(Network$Colors[, i], MergedColors$Colors);
  RowLabels = c(Set.Labels[i], "Consensus");
  hclustplotn(Network$ClusterTree[[i]]$data, SCColor, RowLabels = RowLabels,
cex.RowLabels = 1,
               main="Module colors")
}
```

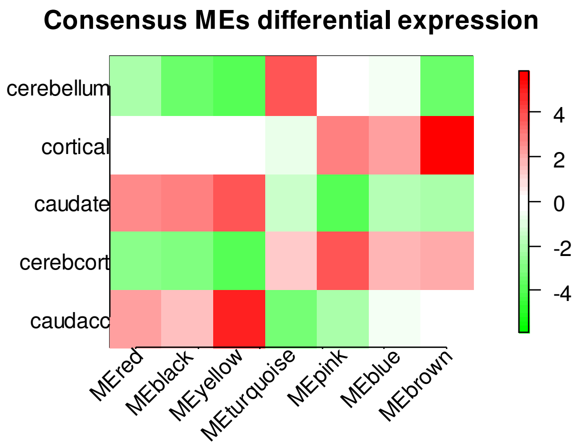


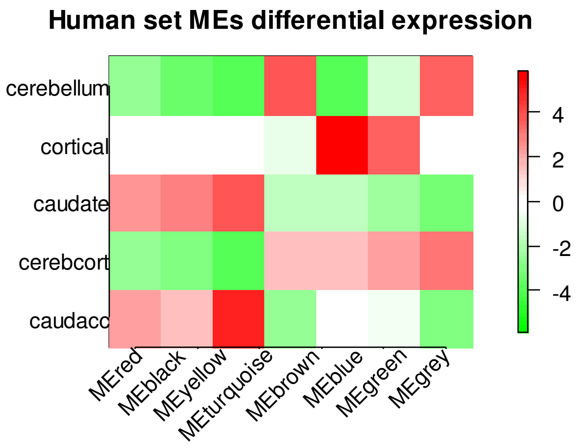
```
#dev.copy2eps(file=paste(PlotDir, OutFileBase, "SetDendr.eps",sep=""));

# Check module heatmap plots to make sure the modules make sense on the level of
expression
# data. Note: resulting figures are not reproduce here

ScaledExprData = scale(ExprData);

ModColors = levels(as.factor(MergedColors$Colors));
No.Mods = length(ModColors);
No.Samples = dim(ExprData)[1];
PlotSections = c(2,2);
PlotsPerScreen = prod(PlotSections);
No.SampleOrders = 2;
No.Screens = as.integer(No.Mods*No.SampleOrders/PlotsPerScreen);
if (No.Screens * PlotsPerScreen!=No.Mods) No.Screens = No.Screens + 1;

SampleOrder = matrix(0, nrow=No.Samples, ncol=No.SampleOrders);

BaseOrder = seq(from=1, by=6, length.out=6);
SampleOrder[1:6,1] = BaseOrder;
for (i in (2:6))
{
  SampleOrder[c( ((i-1)*6+1):(i*6)), 1] = BaseOrder + (i-1);
}
SampleOrder[,2] = c(1:No.Samples);

par(mar=c(2,3,5,3));
par(oma=c(0,0,2,0));
```

```
module = 1; sorder = 1;
for (screen in 1:No.Screens)
{
  par(mfrow = PlotSections);
  for (Plot in 1:PlotsPerScreen) if (module <= No.Mods)
  {
    ModExprData = ScaledExprData[, Network$SelectedGenes][
                                  SampleOrder[,sorder], MergedColors$Colors ==
ModColors[module]];
    plot.mat(t(ModExprData), title = ModColors[module], clabels = rownames(ExprData)
[SampleOrder[,sorder]]);
    if (sorder==2)
    {
      module = module + 1;
      sorder = 1;
    } else {
      sorder = sorder+1;
    }
  }
  print("Press enter to continue"); scan();
}

#-------------------------------------------------------------------------------------
----------------

#-------------------------------------------------------------------------------------
--
# Assign modules to brain regions; brain regions are based on Oldham et al, 2006.

No.Samples = length(Network$Subsets);
SetPCs = NetworkModulePCs(ExprData, Network, UniversalModuleColors = Network$Colors[,
1],
                       verbose=3)
OrderedSetPCs = OrderPCs(SetPCs, GreyLast=TRUE, GreyName = "MEgrey");
SetModColors = names(OrderedSetPCs[[1]]$data);
No.SetMods = length(SetModColors);
MergedSetPCs = matrix(0, nrow = No.Samples, ncol = No.SetMods);
for (set in 1:No.Sets)
{
  MergedSetPCs[Network$Subsets==set, ] = as.matrix(OrderedSetPCs[[set]]$data);
}

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

ConsModColors = names(OrderedConsPCs[[1]]$data);
No.ConsMods = length(ConsModColors);

MergedConsPCs = matrix(0, nrow = No.Samples, ncol = No.ConsMods);
for (set in 1:No.Sets)
```

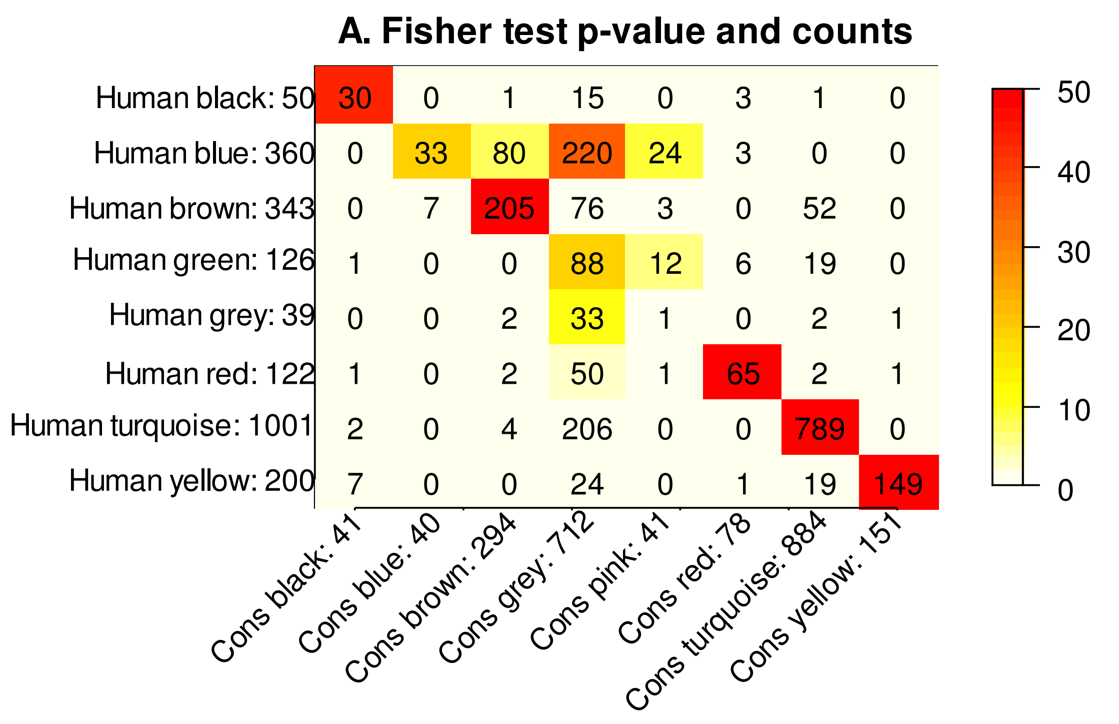
```
{
  MergedConsPCs[Network$Subsets==set, ] = as.matrix(OrderedConsPCs[[set]]$data);
}

Tissues = c("cerebellum", "cortical", "caudate", "cerebcort", "caudacc");
No.Tissues = length(Tissues);
ConsTissueAssignment = matrix(0, nrow=No.Tissues, ncol=No.ConsMods-1);
SetTissueAssignment = matrix(0, nrow=No.Tissues, ncol=No.SetMods-1);

# manually assign samples to tissue groups

Selector = matrix(0, nrow = No.Samples, ncol = No.Tissues);
Selector[, Tissues == "cerebellum"] = rep(c(0,0,0,0,0,1),6);
Selector[, Tissues == "cortical"] = rep(c(1,1,1,1,0,0),6);
Selector[, Tissues == "caudate"] = rep(c(0,0,0,0,1,0),6);
Selector[, Tissues == "cerebcort"] = rep(c(1,1,1,1,0,1),6);
Selector[, Tissues == "caudacc"] = rep(c(0,1,0,0,1,0),6);

for (tissue in 1:No.Tissues)
{
  print(paste("For tissue", Tissues[tissue], "selecting samples ",
         paste(rownames(ExprData)[Selector[, tissue]==1], collapse=", ")));

  for (mod in 1:No.SetMods) if (names(OrderedConsPCs[[1]]$data)[mod]!="MEgrey")
  {
     SetTissueAssignment[tissue, mod] = -log10(kruskal.test(MergedSetPCs[, mod],
Selector[, tissue])$p.value);
     gmean = tapply(MergedSetPCs[, mod], Selector[, tissue], mean);
     if (gmean[2]<gmean[1]) SetTissueAssignment[tissue, mod] =
-SetTissueAssignment[tissue, mod];
  }
  for (mod in 1:No.ConsMods) if (names(OrderedConsPCs[[1]]$data)[mod]!="MEgrey")
  {
     ConsTissueAssignment[tissue, mod] =
                -log10(kruskal.test(MergedConsPCs[, mod], Selector[, tissue])
$p.value);
     gmean = tapply(MergedConsPCs[, mod], Selector[, tissue], mean);
     if (gmean[2]<gmean[1]) ConsTissueAssignment[tissue, mod] =
-ConsTissueAssignment[tissue, mod];
  }
}

ConsModColorLabels = ConsModColors[ConsModColors!="MEgrey"];

SizeWindow(10,4);
par(mfrow=c(1,2));
par(cex = StandardCex/1.4);
par(mar = c(3,3,2,2)+0.2);

MaxPVal = max(abs(ConsTissueAssignment), abs(SetTissueAssignment));
```

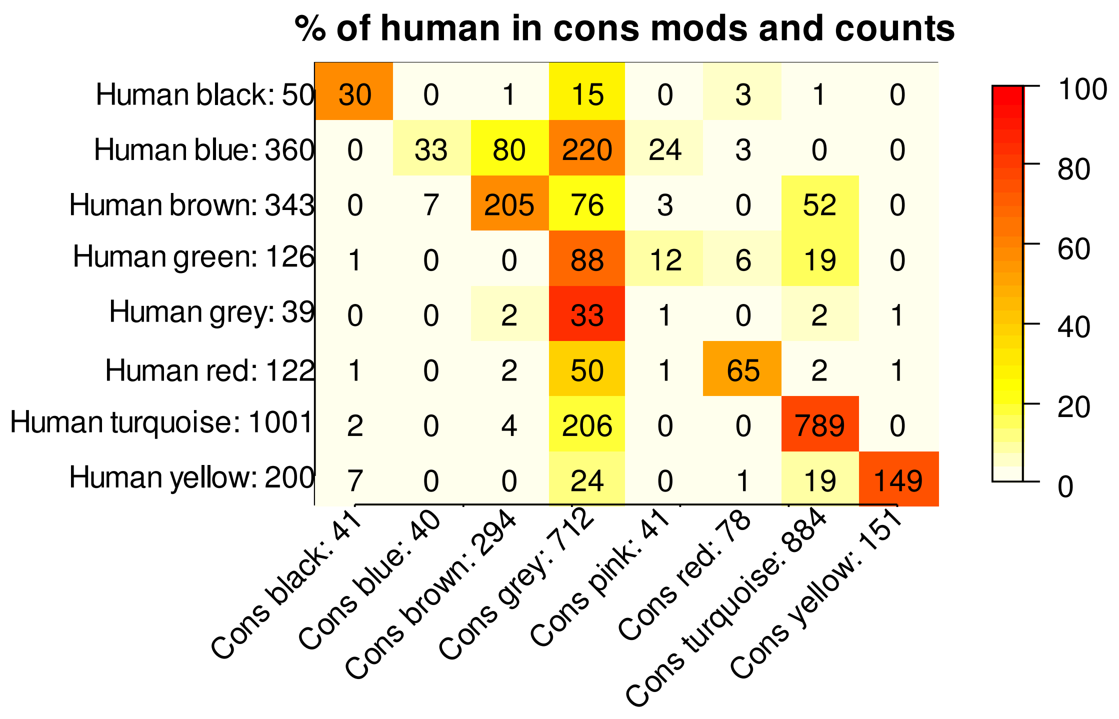
```
HeatmapWithTextLabels(Matrix = ConsTissueAssignment, xLabels = ConsModColorLabels,
yLabels = Tissues,
                      colors = GreenWhiteRed(50),
                      InvertColors=FALSE, main = "Consensus MEs differential
expression",
                      zlim = c(-MaxPVal, +MaxPVal));

par(cex = StandardCex/1.4);
par(mar = c(3,3,2,2)+0.2);
HeatmapWithTextLabels(Matrix = SetTissueAssignment, xLabels = SetModColors, yLabels =
Tissues,
                      colors = GreenWhiteRed(50),
                      InvertColors=FALSE, main = "Human set MEs differential
expression",
                      zlim = c(-MaxPVal, +MaxPVal));
```





```
#dev.copy2eps(file = paste(PlotDir, OutFileBase,
"ModuleTissueAssignment.eps",sep=""));

#-----------------------------------------------------------------------------------------
--
# Assign consensus and human set modules to one another

SetModColors = as.factor(Network$Colors[, 1]);
No.SetMods = nlevels(SetModColors);
ConsModColors = as.factor(MergedColors$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(7,9);
par(mfrow=c(2,1));
par(cex = StandardCex/1.4);
par(mar=c(6,9,2,2)+0.3);

HeatmapWithTextLabels(Matrix = pTable,
                      xLabels = paste("Cons ", levels(ConsModColors), ": ",
                                      ConsModTotals, sep=""),
                      yLabels = paste(Set.Labels[1], " ", levels(SetModColors), ": ",
SetModTotals,
                                      sep=""),
                      NumMatrix = CountTbl,
                      InvertColors = TRUE, SetMargins = FALSE,
                      main = "A. Fisher test p-value and counts");

par(cex = StandardCex/1.4);
par(mar=c(6,9,2,2)+0.3);

HeatmapWithTextLabels(Matrix = PercentageTbl,
                      xLabels = paste("Cons ", levels(ConsModColors), ": ",
                                      ConsModTotals, sep=""),
                      yLabels = paste(Set.Labels[1], " ", levels(SetModColors), ": ",
SetModTotals,
                                      sep=""),
                      NumMatrix = CountTbl,
                      InvertColors = TRUE, SetMargins = FALSE,
                      main = "% of human in cons mods and counts", zlim = c(0,100));
```





```
#dev.copy2eps(file = paste(PlotDir, OutFileBase, "SetVsConsModules.eps",sep=""));
```