Tutorial on studying module preservation: II. Preservation of the cholesterol biosynthesis module among mouse tissues

Peter Langfelder and Steve Horvath

October 23, 2010

Contents

1 Overview

In this document we provide the analysis code of our Application 1, Preservation of the cholesterol biosynthesis module among mouse tissues. In this application we illustrate that modules need not correspond to clusters; here the module corresponds to the GO term "Cholesterol biosynthetic process" (CBP, GO id GO:0006695 and its GO offspring). We provide the full R code that we used in the analysis described in the main paper. The data were first published in [1].
We encourage readers unfamiliar with any of the functions used in this tutorial to open an R session and type

```
help(functionName)
```

(replace `functionName` with the actual name of the function) to get a detailed description of what the functions does, what the input arguments mean, and what is the output.

Execution time

We advise the reader that the actual calculation of preservation statistics in Section 5 is rather long. Calculation of network preservation statistics may take several hours to a a few days, and the calculation of IGP using the `clusterRepro` package may take several days, perhaps even weeks.

2 Setting up the R session

After starting R we execute a few commands to set the working directory and load the requisite packages:

```
# Display the current working directory
getwd();
# If necessary, change the path below to the directory where the data files are stored.
# "." means current directory. On Windows use a forward slash / instead of the usual \.
workingDir = ".";
setwd(workingDir);
# Load the package
library(WGCNA);
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
```

3 Data input and preprocessing

Here we load and pre-process the data from 8 different combinations of tissues and genders. Since the pre-processing takes some time, at the end we save the results so subsequent runs can be done faster.
The expression data is contained in 8 files that are included in the data zip bundle that comes with this tutorial. The reader should change the directory setting below to where he/she downloaded and stored the expression data files.

```
# Set this the character variable below to the directory where you store the expression data files.
dataDir = "../../../Data-AllMouse/";
# Set up basic constants
nGenders = 2;
nTissues = 4;
dataFiles = matrix(
    spaste(dataDir, "ApoE_", c("Adipose", "Brain", "Liver", "Muscle"), "_F2_",
           c(rep("Female", nTissues), rep("Male", nTissues)), "_mlratio.csv.bz2"),
    nTissues, nGenders);
setNames = c("Adipose Female", "Brain Female", "Liver Female", "Muscle Female",
             "Adipose Male", "Brain Male", "Liver Male", "Muscle Male");
expr0 = list();
nSets = nTissues * nGenders;
index = function(tissue, gender) { (gender - 1) * nTissues + tissue}
tissue = function(index) { (index-1)%%nTissues + 1}
gender = function(index) { as.integer((index-1)/nTissues) + 1}
```

The pre-processing part follows. This part only needs to be executed once. We load the raw data and restrict them to common probe sets.

```
stdProbes = NULL;
# Read in the data
for (g in 1:nGenders) for (t in 1:nTissues)
{
  printFlush(paste("Reading file ", dataFiles[t,g]));
  x = read.csv(bzfile(dataFiles[t,g]))
  probes = x$transcript_id
  if (!is.null(stdProbes))
  {
    stdProbes = intersect(probes, stdProbes);
  } else
    stdProbes = probes;
  exprCols = substring(colnames(x), 1, 3)=="F2_";
  expr0[[index(t,g)]] = list(data = t(as.matrix(x[, exprCols])));
  colnames(expr0[[index(t,g)]]$data) = probes;
  collectGarbage();
}
# Restrict data to common probe sets
for (set in 1:nSets)
  expr0[[set]]$data = expr0[[set]]$data[, match(stdProbes, colnames(expr0[[set]]$data))];
# Read in annotation data
annot = read.csv(bzfile("../../../Data-AllMouse/ApoE_GeneAnnotation.csv.bz2"));
geneSymbols = annot$gene_symbol[match(stdProbes, annot$substanceBXH)];
# Keep only probes that are associated to a known gene
for (set in 1:nSets)
  expr0[[set]]$data = expr0[[set]]$data[, !is.na(geneSymbols)];
geneSymbols = geneSymbols[!is.na(geneSymbols)]
stdProbes = colnames(expr0[[set]]$data);
geneSymbols = annot$gene_symbol[match(stdProbes, annot$substanceBXH)];
collectGarbage();
# Remove +/- 2 from expressions. These values are truncations of more extreme values
# and hence are not reliable.
for (set in 1:nSets)
  expr0[[set]]$data[abs(expr0[[set]]$data)==2] = NA;
```

We next remove outlier samples based on hierarchical clustering.

```
# Look at sample clustering and remove samples
dists = list();
trees = list();
for (set in 1:nSets)
{
  dists[[set]] = dist(scale(expr0[[set]]$data));
  trees[[set]] = hclust(dists[[set]], method = "average");
}
cutHeights = rep(250, nSets);
# Plot the trees if desired
if (FALSE)
{
  sizeGrWindow(12,9);
  par(mfrow = c(2,4));
  for (set in 1:nSets)
  {
    plot(trees[[set]], main = setNames[set], xlab = "", sub = "", labels = FALSE);
    abline(h = cutHeights[set], col = "red");
  }
}
# Cut the trees:
expr1 = list();
for (set in 1:nSets)
{
  labs = cutreeStatic(trees[[set]], cutHeight = cutHeights[set], minSize = 30);
  expr1[[set]] = list(data = expr0[[set]]$data[labs==1, ]);
}
```

The next step is to remove probe sets and/or samples that contain an excessive amount of missing data.

```
ggs = goodSamplesGenesMS(expr1)
for (set in 1:nSets)
   expr1[[set]]$data = expr1[[set]]$data[, ggs$goodGenes];
collectGarbage();
```

Lastly, we convert probe-level expression data to gene-level expression data. For each gene, we use the median expression among the probe sets that represent the gene. We save the result so it can be used for future re-runs of the following code.

```
geneSymbols = geneSymbols[ggs$goodGenes];
stdProbes = stdProbes[ggs$goodGenes];
expr = list();
for (set in 1:nSets)
{
  printFlush(paste("Working on set", setNames[set]))
  print(system.time( {
    expr[[set]] = list(data = t(as.matrix(apply(expr1[[set]]$data, 1, tapply, geneSymbols, median,
                                                na.rm = TRUE))))
                  }));
}
# Save the results
save(expr, annot, file = "individualPathways-allGene-expr-annot.RData");
```

If the expression data have been pre-processed previously, they can be loaded using

```
load(file = "individualPathways-allGene-expr-annot.RData");
```

4 Identifying genes that belog to the selected GO term

We next identify the module we are interested in, namely Cholesterol Biosynthetic Process (CBP). We use the packages GO.db and org.Mm.egGO2EG available from Bioconductor, `http://www.bioconductor.org/`, to obtain the list of genes present in CBP.

```
collectGarbage();
geneSymbols = colnames(expr[[1]]$data);
LLIDs = annot$LocusLinkID[match(geneSymbols, annot$gene_symbol)]
# Load the GO.db package
library(GO.db)
xx <- as.list(GOTERM)
# Identify the term "Cholesterol Biosynthetic Process"
terms = lapply(xx, Term)
terms2 = sapply(terms, I)
candidates = grep("cholesterol bio", terms2, fixed = TRUE)
terms2[candidates]
candidates = match("cholesterol biosynthetic process", terms2)
GOid = names(terms2)[candidates]
# Get also all GO children:
# Max 1000 children..
oxx = as.list(GOBPOFFSPRING);
ind = match(GOid, names(oxx));
GOid = c(GOid, as.character(oxx[[ind]]));
# Load the mouse GO database to get the entrez codes for this category
library("org.Mm.eg.db")
go2eg = as.list(org.Mm.egGO2EG)
ecInTerms = rep("", 100000);
ind = 1;
for (t in GOid)
{
  go2egInd = match(t, names(go2eg));
  if (is.finite(go2egInd))
  {
    gs = unique(as.character(go2eg[[match(t, names(go2eg))]]))
    l = length(gs);
    ecInTerms[ind:(ind + l-1)] = gs;
    ind = ind + l;
  }
}
ecInTerms = unique(ecInTerms);
# Look for the CBP genes in the expression data
term2expr = match(ecInTerms, LLIDs);
fin = is.finite(term2expr);
symbolsInTerm = geneSymbols[term2expr[fin]]
# Which genes do we find?
sort(symbolsInTerm)
# Number of genes:
length(symbolsInTerm);
```

We now set up module labels that reflect the pathway membership. The function `modulePreservation` requires that there be at least 2 proper modules. Thus, in addition to the pathway module, we also create an auxiliary module with randomly sampled membership.

```
size = checkSets(expr);
nGenes = size$nGenes;
pathGenes = term2expr[fin];
nPathGenes = length(pathGenes);
pathwayLabels = sample(c(0,1), nGenes, replace = TRUE);
pathwayLabels[term2expr[fin]] = 2;
```

We next calculate module eigengenes of the pathway module in each of the data sets and calculate module membership (KME).

```
multiMEs = multiSetMEs(expr, universalColors = pathwayLabels)
KMEpathway = matrix(0, nPathGenes, nSets);
for (set in 1:nSets)
{
  KMEpathway[, set] = bicor(expr[[set]]$data[, pathGenes], multiMEs[[set]]$data[, 3], use = "p")
}
# If necessary, can plot a scatterplot of the module membeships
if (FALSE)
{
  sizeGrWindow(9,9);
  verbosePairs(KMEpathway, names = setNames, abline = TRUE)
}
```

5 Calculation of module preservation

Here we call the function `modulePreservation` to calculate a multitude of module preservation statistics.

```
multiColor = list();
for (set in 1:nSets) multiColor[[set]] = pathwayLabels;
names(multiColor) = setNames;
names(expr) = setNames;
mp = modulePreservation(expr, multiColor, networkType = "unsigned", corFnc = "bicor",
                        referenceNetworks = c(1:nSets), verbose = 4, maxModuleSize = 500,
                        maxGoldModuleSize = 500, nPermutations = 200)
# Save the results
save(mp, file = "individualPathways-allgenes-mp.RData");
```

Calculation of In-Group Proportion

To compare network module preservation statistics to existing methods of measuring cluster preservation, we calculate the In-Group Proportion (IGP) [2]. The reader should be aware that because of the large number of genes in our data sets, this calculation can take several days. The results are saved to disk.

```
library(clusterRepro)
cr = list();
set.seed(10);
for (set in 1:nSets)
{
  printFlush(paste("Working on ", setNames[set]));
  # Leave out
  centr = as.matrix(multiMEs[[set]]$data[, c(2:3)]);
  rownames(centr) = rownames(expr[[set]]$data);
  colnames(centr) = c("Random", "CholSynth");
  print(system.time({
    cr[[set]] = clusterRepro(Centroids = centr, New.data = expr[[set]]$data,
                     Number.of.permutations = 2000);
     } ));
  save(cr, file = "individualPathways-allgenes-cr.RData")
}
```

6 Analysis and graphical representation of results

We load the saved results of the calculations. This step is only necessary if the results were calculated in a separate R session.

```
load(file = "individualPathways-allgenes-mp.RData");
load(file = "individualPathways-allgenes-cr.RData")
```

We next format the results to make subsequent analysis easier. We isolate the summary Z statistics.

```
Zsummary = matrix(NA, nSets, nSets);
Zdensity = matrix(NA, nSets, nSets);
Zconn = matrix(NA, nSets, nSets);
Zquality = matrix(NA, nSets, nSets);
for (ref in 1:nSets) for (test in 1:nSets) if (ref!=test)
{
  Z = c(as.matrix(mp$preservation$Z[[ref]][[test]][-c(1:3), -1]));
  Zsummary[ref, test] = Z[1];
  Zdensity[ref, test] = Z[2];
  Zconn[ref, test] = Z[3];
  Zquality[ref, test] = median(c(as.matrix(mp$quality$Z[[ref]][[test]][4, 2])));
}
Zs = list();
Zs[[1]] = Zquality
Zs[[2]] = Zsummary
Zs[[3]] = Zdensity
Zs[[4]] = Zconn;
Znames = c("Zsummary.quality", "Zsummary.preservation",
          "Zdensity.preservation", "Zconnectivity.preservation");
order = c(1,5,3,7,2,6,4,8);
setNames2 = sub("Female", "F", setNames, fixed = TRUE);
setNames2 = sub("Male", "M", setNames2, fixed = TRUE);
```

We next produce Figure 2 in the main article. This figure combines the summary Z statistics, clusterRepro results, and scatterplots of module membership in the CBP pathway in female liver vs. all other tissues.

```
order2 = c(1,3,2,4, 5,7,6,8);
sizeGrWindow(10, 8);
#pdf(file = "Plots/indivPathway-allgenes-GOCholesterolBiosynthesis-summaryForPaper.pdf", w = 10, h=7.7)
layout(matrix(c(1:12), 3, 4, byrow = TRUE));
par(mar = c(5.3, 3.3, 4, 0.5));
par(mgp = c(1.9, 0.6, 0))
# Plot of summary Z statistics
for (i in 2:4)
{
   pres = Zs[[i]][order, order][ref, -ref];
   labeledBarplot2(pres, names = setNames2[order][-ref],
              main = spaste(LETTERS[i-1], ". ", Znames[i], "\nReference set: liver female"),
              cex.names = 1.2,
              cex.axis = 1.2, ylab = Znames[i], cex.main = 1.4, cex.lab = 1.2);
}
# Plot of IGP calculated by clusterRepro
labeledBarplot2(crPathway[order], names = setNames2[order],
              main = "D. Observed IGP",
              ylab = "Actual.IGP", cex.names = 1.2,
              cex.axis = 1.2, cex.main = 1.4, cex.lab = 1.2);
# KME scatterplots
par(mar = c(3.3, 3.3, 4, 0.5));
par(mgp = c(1.9, 0.6, 0))
ref = 3;
ind = 5
for (set in 1:nSets)
if (order2[set]!=ref)
{
  verboseScatterplot(KMEpathway[, ref], KMEpathway[, order2[set]],
                  xlab = spaste("KME in ", setNames[ref]),
                  ylab = spaste("KME in ", setNames[order2[set]]),
                  main = spaste(LETTERS[ind], ". KME in ", setNames[order2[set]], "\nvs. ",
                              setNames[ref], "\n"),
                  cex.main = 1.4, cex.lab = 1.2, cex.axis = 1.2, abline = TRUE
                 )
  ind = ind + 1;
} else
  plot(c(0,1), type ="n", axes = FALSE, xlab = "", ylab ="");
# If plotting into a file, close it
dev.off();
```

The result is shown in Figure 1.

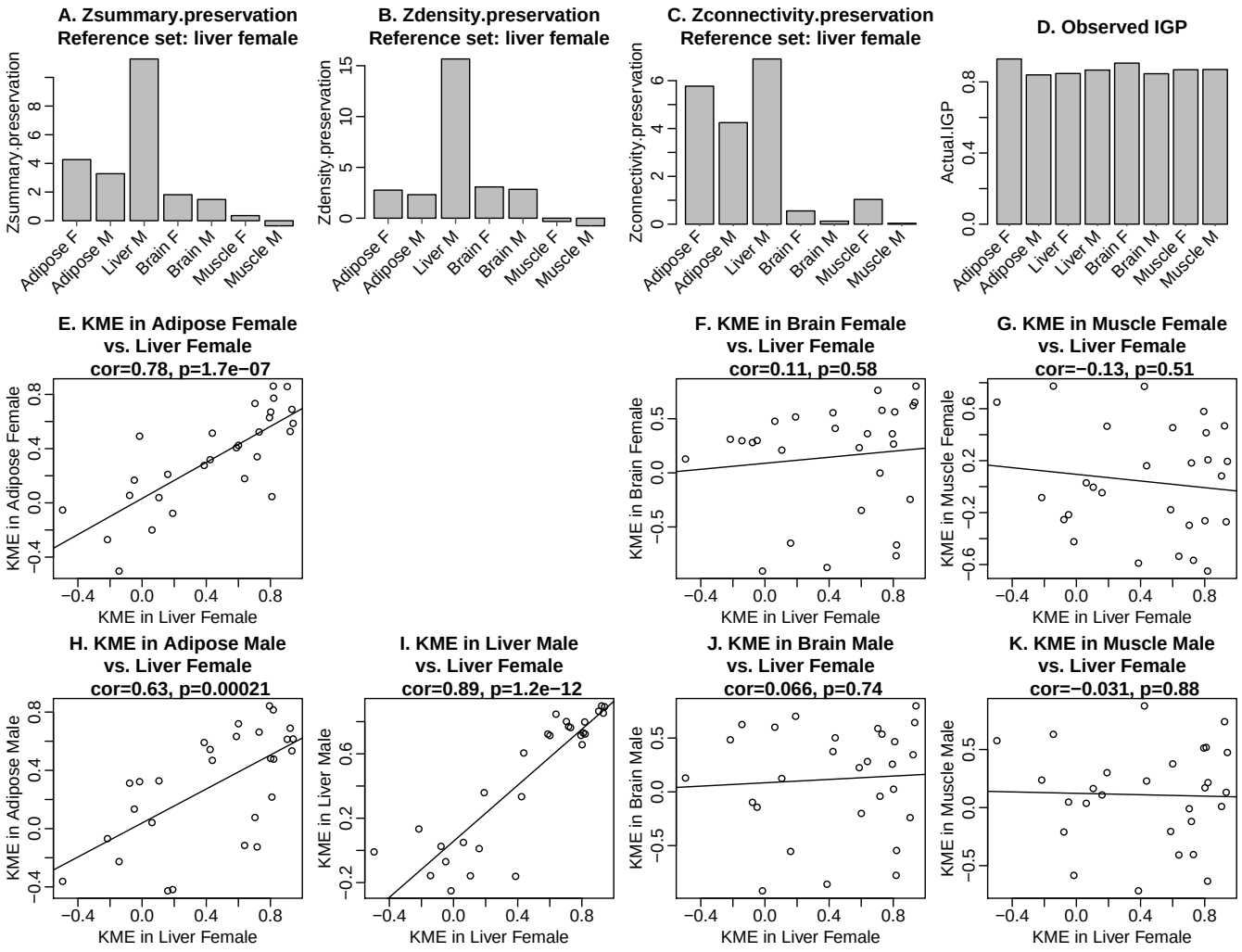

Figure 1: This figure reproduces Figure 2 in our main article and presents quantitative evaluation of the similarities among the networks depicted in Figures 2 and 3. Panels A–C show summary preservation statistics in other tissue and sex combinations. Panel A shows the composite preservation statistic $Z_{summary}$. The CBP module in the female liver network is highly preserved in the male liver network ($Z_{summary} > 10$) and moderately preserved in adipose networks. There is no evidence of preservation in brain or muscle tissue networks. Panels B and C show the density and connectivity statistics, respectively. Panel D shows the results of the in group proportion analysis [2]. According to the IGP analysis, the CBP module is equally preserved in all networks. E-K show the scatter plots of *kME* in one test data set (indicated in the title) vs. the liver female reference set. Each point corresponds to a gene; Pearson correlations and the corresponding p-values are displayed in the title of each scatter plot. The eigengene-based connectivity *kME* is strongly preserved between adipose and liver tissues; it is not preserved between female liver and the muscle and brain tissues.

We next output all statistics into a CSV table that can be opened in spreadsheet software such as Microsoft Excel or OpenOffice Calc.

```
ref = 3
stats = list()
ind = 1;
refNames = NULL;
testNames = NULL;
dropRows = c(1,3)
for (test in 1:nSets) if (ref!=test)
{
  stats[[ind]] = cbind(mp$quality$observed[[ref]][[test]][-dropRows,, drop = FALSE],
                 mp$preservation$observed[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$referenceSeparability$observed[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$testSeparability$observed[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$quality$Z[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$quality$log.p[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$quality$log.pBonf[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$preservation$Z[[ref]][[test]][-dropRows,-1, drop = FALSE],
                 mp$preservation$log.p[[ref]][[test]][-dropRows,-1, drop = FALSE],
                 mp$preservation$log.pBonf[[ref]][[test]][-dropRows,-1, drop = FALSE],
                 mp$referenceSeparability$Z[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$referenceSeparability$log.p[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$referenceSeparability$log.pBonf[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$testSeparability$Z[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$testSeparability$log.p[[ref]][[test]][-dropRows, -1, drop = FALSE],
                 mp$testSeparability$log.pBonf[[ref]][[test]][-dropRows, -1, drop = FALSE]);

  ind = ind + 1;
  refNames = c(refNames, setNames[ref]);
  testNames = c(testNames, setNames[test]);
}
table = NULL;
nPlots = ind-1;
for (p in 1:nPlots)
{
  nMods = nrow(stats[[p]]);
  modNames = rownames(stats[[p]]);
  modStatus = rep("GO term CBP", nMods);
  modStatus[modNames=="grey"] = "improper module"
  modStatus[modNames=="0.1"] = "all-network sample"
  description = cbind( referenceSet = rep(refNames[p], nMods),
                       testSet = rep(testNames[p], nMods),
                       module = rownames(stats[[p]]),
                       moduleType = modStatus);

  if (is.null(table))
  {
    table = cbind(description, stats[[p]])
  } else
    table = rbind(table, cbind(description, stats[[p]]));
}
x = as.matrix(table)
write.table(table, file = "preservationOfCholesterolBiosynthesis-allResults.csv", sep = ",", quote = FALSE,
          row.names = FALSE);
```

Lastly, we produce the motivation figure in which we show the network of the CBP module in the 8 tissue/sex combinations. We use a custom R function to display the networks. The function is included in the file `circlePlot.R`.

```
# Calculate adjacencies within the module
pathwayAdjs = list();
for (set in 1:nSets)
{
  printFlush(paste("Working on set", setNames[set]));
  bc = bicor(expr[[set]]$data[, pathGenes], use = "p");
  #bc[bc<0] = 0;
  pathwayAdjs[[set]] = abs(bc)^4 * sign(bc);
}
# We order the genes by a weighted average connectivity
conn = matrix(0, nPathGenes, nSets);
for (set in 1:nSets)
  conn[, set] = apply(abs(pathwayAdjs[[set]]), 2, sum)-1
weights = c(3,1,5,1, 3,1,5,1);
wMat = matrix(weights, nPathGenes, nSets, byrow = TRUE);
wconn = apply(conn * wMat, 1, sum);
order = order(-wconn);
# use the gene names as lables
labels = colnames(pathwayAdjs[[ref]]);
# Ordering of pathways:
CPorder = c(3,1,2,4,7,5,6,8);
# Input the R function circlePlot
source("circlePlot.R");
sizeGrWindow(12,6)
#pdf(file = "Plots/indivPathway-allgenes-GOCholesterolBiosynthesis-circlePlots.pdf", wi=12, h=6.5)
par(mfrow =c(2,4));
par(mar = c(0.3, 0.2, 1.5, 0.2))
for (set in 1:nSets)
{
  circlePlot(pathwayAdjs[[CPorder[set]]], labels, order, main = setNames[CPorder[set]],
             variable.cex.labels = TRUE,
             radii = c(0.56, 0.62), center = c(0.1, 0.04),
             min.cex.labels = 1.2, max.cex.labels = 1.4, cex.main = 1.4);
}
# If plotting into a file, close it
dev.off();
```

The result is shown in Figure 2. The female and male liver networks appear very similar. The adipose networks also show some similarity to the liver networks, while brain and muscle networks appear different. These results are quantified more precisely in Figure 1.

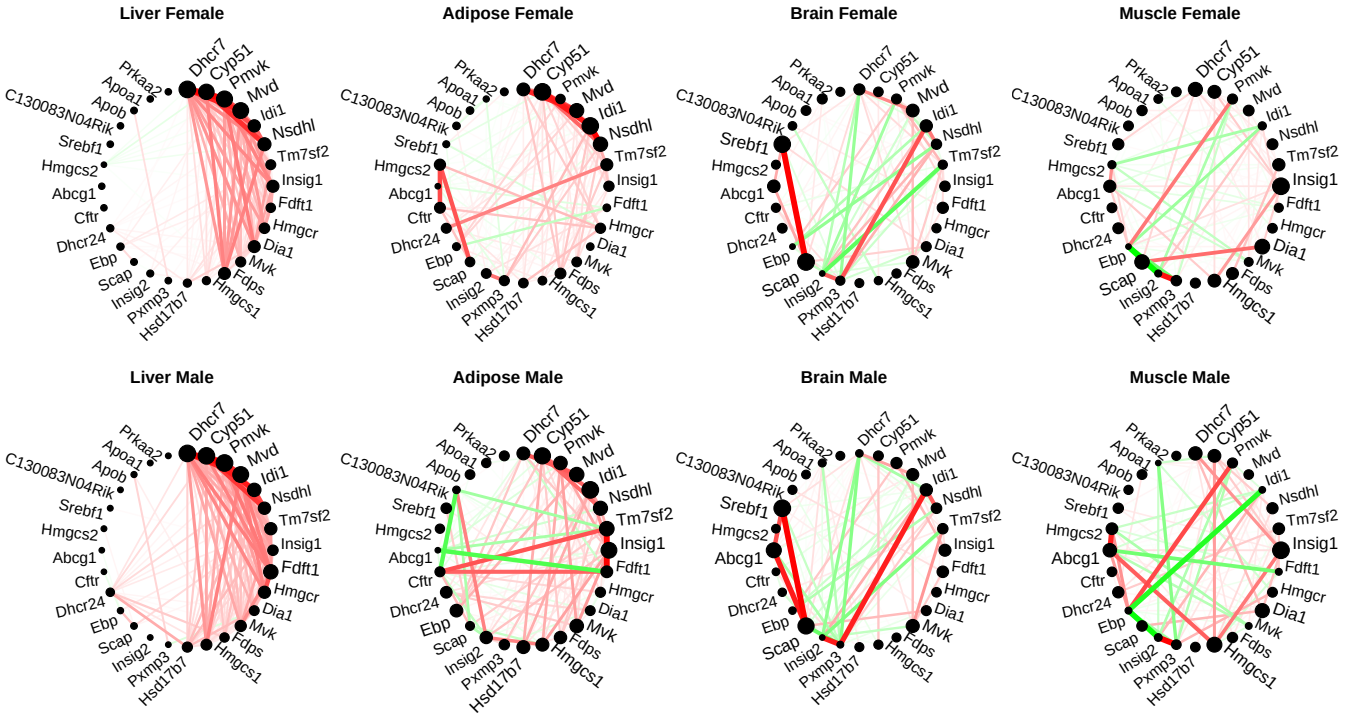

Figure 2: Network plot of the module of cholesterol biosynthesis genes in different mouse tissues in a rectangular layout. Positive correlations are represented by red lines, while negative correlations are represented by green lines. Correlation strength is represented by thickness and color saturations of the line. Intramodular hub genes are represented by larger points and their names are typeset in larger font. Note the similarity between the female and male liver networks. The adipose networks also show some similarity to the liver networks, while brain and muscle networks appear different.

The second version of the same plot has the individual circles positioned in a more circular pattern around the center in which we place the female liver network.

```
# Shorten long gene labels
shortLabels = substring(labels, 1, 9);
# Open a graphics window...
sizeGrWindow(9,9);
# ...or a file for plotting
#pdf(file="Plots/indivPathway-allgenes-GOCholesterolBiosynthesis-circlePlots-circle.pdf",wi=9,he=9);
# centers of the circles
centers = matrix(c(0.7,0.55, -0.37,-0.73, 0,0, -0.7,0.55,
                   0.72,-0.15, 0.37,-0.73, 0,0.7, -0.72,-0.15),
                   2, nSets)
# radii
radii = matrix(c(rep(c(0.22, 0.24), 2), 0.25, 0.28, rep(c(0.22, 0.24), 5)), 2, nSets) * 0.9;
# margins
par(mar = c(0,0,0,0));
# Plot the circles
for (set in 1:nSets)
{
  circlePlot(pathwayAdjs[[CPorder[set]]], shortLabels, order, main = "",
             center = centers[, CPorder[set]],
             startNewPlot = set==1,
             radii = radii[, CPorder[set]],
             min.cex.points = 0.5, max.cex.points = 2.0,
             variable.cex.labels = TRUE,
             min.cex.labels = 0.7, max.cex.labels = 1.0, cex.main = 1.4,
             xLabelOffset = 0, yLabelOffset = 0);

}
setNames2 = sub(" ", "\n", setNames, fixed = TRUE);
text(centers[1,], centers[2, ], setNames2, cex = 2.0, col = "#777777");
# If plotting into a file, close it.
dev.off();
```

The result is shown in Figure 3.

References

[1] A. Ghazalpour, S. Doss, B. Zhang, C. Plaisier, S. Wang, E.E. Schadt, A. Thomas, T.A. Drake, A.J. Lusis, and S. Horvath. Integrating genetics and network analysis to characterize genes related to mouse weight. *PloS Genetics*, 2(2):8, 2006.

[2] Amy V. Kapp and Robert Tibshirani. Are clusters found in one dataset present in another dataset? *Biostat*, 8(1):9–31, 2007.

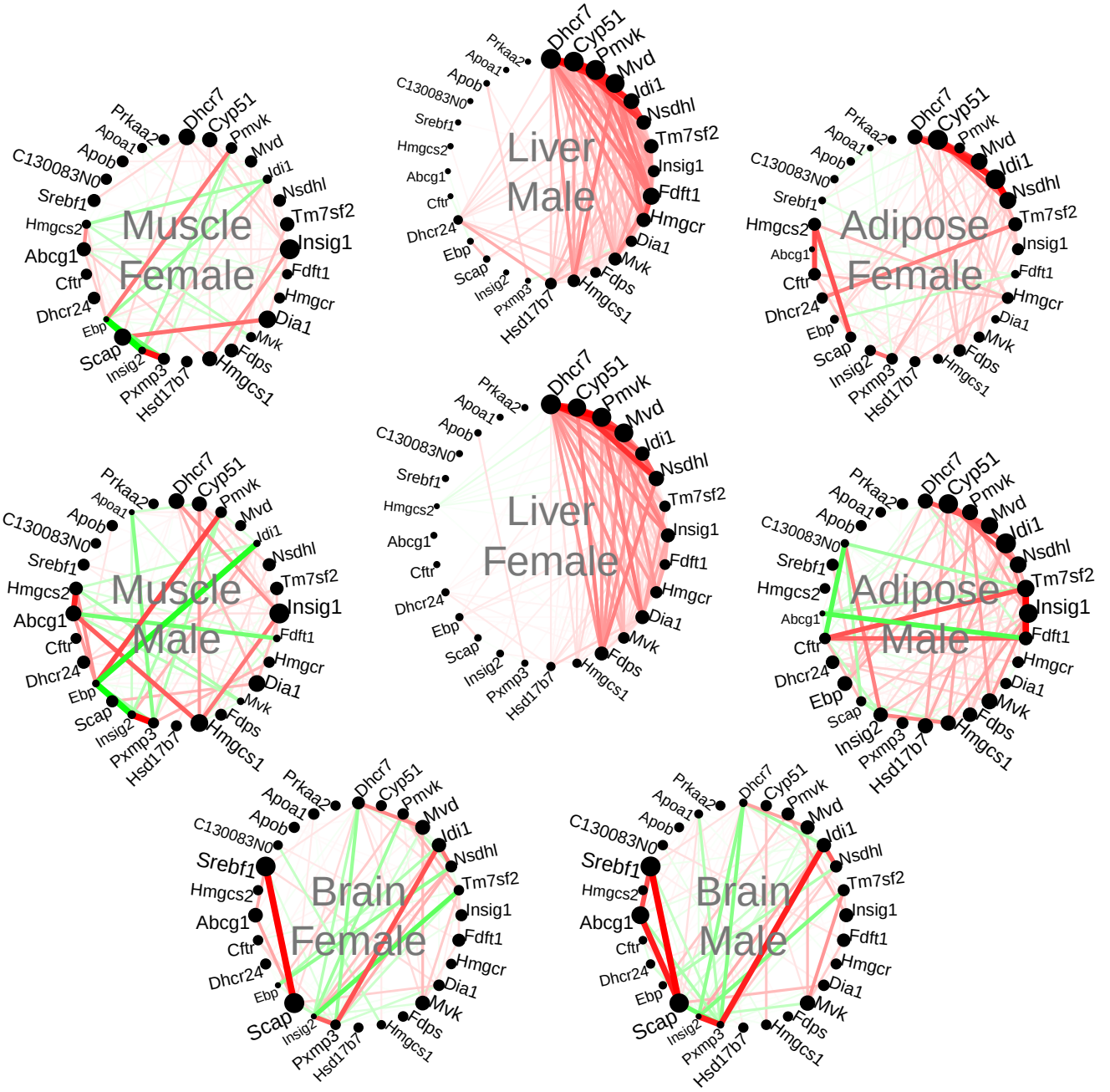

Figure 3: Network plot of the module of cholesterol biosynthesis genes in different mouse tissues in a circular layout.