Tutorial on studying module preservation:
IV. Preservation of KEGG pathways between human and chimp data

Peter Langfelder and Steve Horvath

February 21, 2013

Contents

1 Overview

In this document we provide the analysis code of our Application 3, Preservation of KEGG pathways between human and chimpanzee brains. In this application the module corresponds to the various signalling pathways contained ine the Kyoto Encyclopedia of Genes and Genomes [2]. Specifically, we focused the analysis on the following 8 signaling pathways: Hedgehog signaling pathway (12 genes in our data sets), apoptosis (24 genes in our data sets), TGF-beta signaling pathway (26 genes in our data sets), Phosphatidylinositol signaling system (39 genes in our data sets), Wnt signaling pathway (55 genes in our data sets), Endocytosis (59 genes in our data sets), Calcium signaling pathway (78 genes in our data sets), MAPK signaling pathway (93 genes in our data sets). 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 4 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);
# Load a custom function to produce circle network plot
source("circlePlot.R");
```

3 Data input and preprocessing

We load the expression data and split them into separate data sets for human and chimp samples. The data has been carefully pre-processed and cleaned, which allows us to use it as is.

```
# Set this the character variable below to the directory where you store the expression data files.
dataDir = "./"
# Load the data
file = bzfile("../../../Old/HumanChimpData/Dataset 1 (network construction).csv.bz2");
dat1 = read.csv(file, header=T)
# This data frame contains the gene expression data.
# By our convention, columns are genes and rows are samples.
datExpr=data.frame(t(dat1[dat1$Brain_variant_H>0,2:39]))
indexHuman=c(19:36)
indexChimp=c(1:18)
nSets = 2;
multiExpr = list();
multiExpr[[1]] = list(data = datExpr[indexHuman, ]);
multiExpr[[2]] = list(data = datExpr[indexChimp, ]);
colnames(multiExpr[[1]]$data) = dat1$Probe_set[dat1$Brain_variant_H>0]
colnames(multiExpr[[2]]$data) = dat1$Probe_set[dat1$Brain_variant_H>0]
probes = colnames(multiExpr[[1]]$data)
```

Next we load the `KEGG.db` package that contains information from KEGG, and isloate gene IDs for selected pathways.

```
library(KEGG.db)
intPaths = c("Wnt signaling pathway", "Hedgehog signaling pathway",
            "Phosphatidylinositol signaling system", "MAPK signaling pathway",
            "Calcium signaling pathway", "TGF-beta signaling pathway", "Apoptosis", "Endocytosis");
xx <- as.list(KEGGPATHNAME2ID)
IDs = sapply(xx, as.character)
intIDs = IDs[match(intPaths, names(IDs))]
xx <- as.list(KEGGPATHID2EXTID)
intIDs2 = spaste("hsa", intIDs);
ids2entrez = match(intIDs2, names(xx));
entrezPathways = xx[ids2entrez];
nPathways = length(intPaths);
```

We next match the probe sets contained in the data to their respective gene IDs to allow matching egainst the pathway information.

```
library(hgu95av2.db)
x <- hgu95av2ENTREZID
mapped_probes <- mappedkeys(x)
# Convert to a list
xx <- as.list(x[mapped_probes])
dbProbes = names(xx);
dbEntrez = sapply(xx, I)
```

```
# Restrict the expression data to probes that exist in the hgu95av2 data base
data2db = match(probes, dbProbes);
fin = is.finite(data2db);
probeEntrez = dbEntrez[data2db[fin]]
multiExpr[[1]]$data = multiExpr[[1]]$data[, fin];
multiExpr[[2]]$data = multiExpr[[2]]$data[, fin];
# get gene names...
x <- hgu95av2SYMBOL
mapped_probes <- mappedkeys(x)
# Convert to a list
xx <- as.list(x[mapped_probes])
dbProbes = names(xx);
dbGeneNames = sapply(xx, I)
data2db = match(probes, dbProbes);
fin = is.finite(data2db);
probeGeneNames = dbGeneNames[data2db[fin]]
entrez2name = cbind(probeEntrez, probeGeneNames);
```

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.

```
colMedians = function(x) { apply(x, 2, median, na.rm = TRUE) }
geneExpr = list();
for (set in 1:nSets)
{
  x = collapseRows(t(multiExpr[[set]]$data), probeEntrez, probes[fin],
                                          method = "function", methodFunction = colMedians,
                                          connectivityPower = 9);
  geneExpr[[set]] = list(data = t(x$datETcollapsed));
}
```

4 Calculation of module preservation

The next chunk of code sets up basic variables for the calculation of module membership as well as print out the numbers of genes in each pathway.

```
nGenes = checkSets(geneExpr)$nGenes;
setNames = c("Human", "Chimp");
mp = list();
cr = list();
doModulePreservation = TRUE;
doClusterRepro = TRUE;
library(clusterRepro)
entrezCodes = colnames(geneExpr[[1]]$data);
pathSizes = rep(0, nPathways);
for (p in 1:nPathways)
{

  path2all = match(as.numeric(entrezPathways[[p]]), colnames(geneExpr[[1]]$data))
  printFlush("=====================================");
  printFlush(intPaths[p]);
  printFlush("=====================================");
  print(table(is.na(path2all)));
  pathSizes[p] = sum(!is.na(path2all));
}
```

We next calculate network module preservation statistics. This part may take several hours. At the end we save the results so this part need not be re-run upon subsequent analysis.

```
set.seed(20);
for (p in 1:nPathways)
{
  printFlush(paste("============================== Pathway", p, " ============================="));
  path2all = match(as.numeric(entrezPathways[[p]]), entrezCodes)
  fin = is.finite(path2all);
  pathwayLabels = sample(c(0,1), nGenes, replace = TRUE);
  pathwayLabels[path2all[fin]] = 2;
  multiColor = list();
  for (set in 1:nSets) multiColor[[set]] = pathwayLabels;
  names(multiColor) = setNames;
  names(geneExpr) = setNames;

  if (doModulePreservation)
  {

    print(system.time( {
      mp[[p]] = modulePreservation(geneExpr, multiColor, networkType = "unsigned", corFnc = "cor",
                            referenceNetworks = c(1:nSets), verbose = 4, maxModuleSize = 500,
                            maxGoldModuleSize = 500, nPermutations = 200)
    }));

    save(mp, file = "individualPathways-02-mp.RData");
  }
}
```

In the next step we use the `clusterRepro` function to calculate an alternative statistic of module preservation. We included this statistic for comparison but since `clusterRepro` measures preservation of clusters, it is not really suited for this application. As we will see in the results, it fails in this application in the sense that it does not discriminate between preserved and non-preserved pathways.

```
for (p in 1:nPathways)
{
  printFlush(paste("============================== Pathway", p, " ============================="));
  path2all = match(as.numeric(entrezPathways[[p]]), entrezCodes)
  fin = is.finite(path2all);
  pathwayLabels = sample(c(0,1), nGenes, replace = TRUE);
  pathwayLabels[path2all[fin]] = 2;
  multiColor = list();
  for (set in 1:nSets) multiColor[[set]] = pathwayLabels;
  names(multiColor) = setNames;
  names(geneExpr) = setNames;
  if (doClusterRepro)
  {
    multiMEs = multiSetMEs(geneExpr, universalColors = pathwayLabels);
    cr[[p]] = list();
    for (set in 1:nSets)
    {
      printFlush(paste("Working on ", setNames[set]));
      centr = as.matrix(multiMEs[[set]]$data[, c(2:3)]);
      rownames(centr) = rownames(geneExpr[[set]]$data);
      colnames(centr) = c("Random", "Pathway");
      print(system.time( {
         cr[[p]][[set]] = clusterRepro(Centroids = centr, New.data = geneExpr[[set]]$data,
                          Number.of.permutations = 5000);
      }));
      save(cr, file = "individualPathways-02-cr.RData")
    }
  }
}
```

5 Analysis of results and plots for the main article

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

```
load(file = "individualPathways-02-mp.RData");
load(file = "individualPathways-02-cr.RData");

Zsummary = array(NA, dim = c(nSets, nSets, nPathways));
Zdensity = array(NA, dim = c(nSets, nSets, nPathways));
Zconn = array(NA, dim = c(nSets, nSets, nPathways));
Zquality = array(NA, dim = c(nSets, nSets, nPathways));

psummary = array(NA, dim = c(nSets, nSets, nPathways));
pdensity = array(NA, dim = c(nSets, nSets, nPathways));
pconn = array(NA, dim = c(nSets, nSets, nPathways));
pquality = array(NA, dim = c(nSets, nSets, nPathways));
for (p in 1:nPathways) for (ref in 1:nSets) for (test in 1:nSets) if (ref!=test)
{
  Z = c(as.matrix(mp[[p]]$preservation$Z[[ref]][[test]][-c(1:3), -1]));
  Zsummary[ref, test, p] = Z[1];
  Zdensity[ref, test, p] = Z[2];
  Zconn[ref, test, p] = Z[3];
  Zquality[ref, test, p] = mp[[p]]$quality$Z[[ref]][[test]][4, 2];
  pmat = c(as.matrix(mp[[p]]$preservation$log.p[[ref]][[test]][-c(1:3), -1]));
  psummary[ref, test, p] = 10^(pmat[1]);
  pdensity[ref, test, p] = 10^(pmat[2]);
  pconn[ref, test, p] = 10^(pmat[3]);
}
Znames = c("Zsummary.quality", "Zsummary",
          "Zdensity", "Zconnectivity");

order = c(1:nSets)

setNames2 = sub("Male", "M", setNames, fixed = TRUE);
setNames2 = sub("Female", "F", setNames2, fixed = TRUE);

Zs = list();
Zs[[1]] = Zquality;
Zs[[2]] = Zsummary;
Zs[[3]] = Zdensity;
Zs[[4]] = Zconn;

maxy = max(c(unlist(Zs)), na.rm = TRUE);
miny = min(c(unlist(Zs)), na.rm = TRUE);
split = as.matrix(as.data.frame(strsplit(intPaths, split = " ", fixed = TRUE)));
shortPaths = apply(split[-3, ], 2, paste, collapse = " ");
split[split=="Phosphatidylinositol"] = "Phosph"
split[split=="Phosphat"] = "Phos"
split[split=="Endocytosis"] = "Endoc"
split[split=="Apoptosis"] = "Apop"
split[split=="Calcium"] = "Ca"
split[split=="Hedgehog"] = "Hdhog"
split[split=="TGF-beta"] = "TGFb"

pathCR = matrix(0, nPathways, nSets);
randCR = matrix(0, nPathways, nSets);
```

```
pCR = matrix(0, nPathways, nSets);
for (p in 1:nPathways) for (set in 1:nSets)
{
  pathCR[p, set] = cr[[p]][[set]]$Actual.IGP[2];
  randCR[p, set] = cr[[p]][[set]]$Actual.IGP[1];
  pCR[p, set] = cr[[p]][[set]]$p.value[2];
}
# For better plotting:
for (p in 1:nPathways) for (set in 1:nSets)
  pCR[p, set] = pCR[p, set] + (1.5-set) * 0.001;
```

Next we calculate kME (that is, eigengene-based intramodular connectivity) for each pathway.

```
intPathways = c(1:8);
split = strsplit(intPaths, split = " ", fixed = TRUE);
kME = list();
for (p in intPathways)
{
  path2all = match(as.numeric(entrezPathways[[p]]), entrezCodes)
  fin = is.finite(path2all);
  pathwayLabels = sample(c(0,1), nGenes, replace = TRUE);

  pathwayLabels[path2all[fin]] = 2;

  multiColor = list();
  for (set in 1:nSets) multiColor[[set]] = pathwayLabels;
  names(multiColor) = setNames;
  names(geneExpr) = setNames;
  multiMEs = multiSetMEs(geneExpr, universalColors = pathwayLabels);

  nPG = sum(pathwayLabels==2);
  kME[[p]] = matrix(0, nPG, nSets);
  for (set in 1:nSets)
    kME[[p]][, set] = c(cor(geneExpr[[set]]$data[, pathwayLabels==2], multiMEs[[set]]$data[, 3]));

  if (cor(kME[[p]][, 1], kME[[p]][, 2]) < 0) kME[[p]][, 2] = -kME[[p]][, 2]
}
```

We next produce Figure 7 in the main article. We start by preparing plot ranges, labels etc.

```
maxy = max(c(unlist(Zs)), na.rm = TRUE);
miny = min(c(unlist(Zs)), na.rm = TRUE);
yrange = maxy - miny
x = rep(pathSizes, rep(2, nPathways));
minx = min(x); maxx = max(x); xrange = maxx - minx
xlim = c( minx - 0.17 * xrange, maxx + 0.24*xrange)
ylim = c( miny - 0.01 * yrange, maxy + 0.04 *yrange)
#ylim = c( miny, maxy)

intPathways = c(1:8);
split = strsplit(intPaths, split = " ", fixed = TRUE);
#ssPathNames = split[1, ]
ssPathNames = c("Wnt", "Hed", "Phos", "MAPK", "Ca", "TGF", "Apo", "End");

pathway = "Apoptosis"
ip = match(pathway, intPaths);
path2all = match(as.numeric(entrezPathways[[ip]]), entrezCodes)
fin = is.finite(path2all);
pathAdj = list();
kIM = list();
```

```
for (set in 1:nSets)
{
  c = cor(geneExpr[[set]]$data[, path2all[fin]]);
  pathAdj[[set]] = c * abs(c);
  kIM[[set]] = apply(abs(pathAdj[[set]]), 2, sum, na.rm = TRUE) - 1;
}
geneOrder = order(-kIM[[1]]);
pathEntrez = colnames(geneExpr[[set]]$data)[path2all[fin]];
pathLabels = entrez2name[match(pathEntrez, entrez2name[, 1]), 2];
```

The actual plotting is next.

```
sizeGrWindow(6.83,9.19)
#pdf(file = "Plots/KEGGpathways-SummaryForPaper.pdf", w=6.83, h=9.19);
layout(matrix( c( 1,1,2,2,3,3,12, 4,5,5,6,6,7,12, 4,5,5,6,6,7,13, 8,9,9,10,10,11,13), 7,4),
       heights = c(3,1,2,2,1,3, 4), widths = c(3, 0.5, 1.5, 2));
par(mar = c(3.3, 3.3, 1.5, 0.3));
par(mgp = c(1.7, 0.6, 0))
ind = 1;
for (i in 2:4)
{
  if (i < 4)
  {
    x = pathSizes;
    y = c(Zs[[i]][1,2,]);
    colors = rep("black", nPathways);
    paths = ssPathNames;
    labels = paths
    plot(x,y, main = spaste(LETTERS[ind], ". ",
                           Znames[i]), pch = 21, bg = colors,
         ylab = Znames[i], ylim = ylim, cex.main = 1.2, cex.lab = 1.1,
         cex.axis = 1.1, cex = 1.5, xlim = xlim, xlab = "Number of genes in pathway")
    addGrid(linesPerTick = 1);
    labelPoints(x,y,labels, offs = 0.05, cex = 1.0, col = colors, jiggle = 0.06)
  } else {
    y = apply(cbind(Zs[[i]][2,1,], Zs[[i]][1,2,]), 1, mean);
    x = pathSizes;
    colors = rep("black", nPathways);
    labels = ssPathNames
    plot(x,y, main = spaste(LETTERS[ind], ". ", Znames[i]), pch = 21, bg = colors,
         ylab = Znames[i], ylim = ylim, cex.main = 1.2, cex.lab = 1.1,
         cex.axis = 1.1, cex = 1.5, xlim = xlim, xlab = "Number of genes in pathway")
    addGrid(linesPerTick = 1);
    labelPoints(x,y,labels, offs = 0.05, cex = 1.0, jiggle = 0.1)
  }
  ind = ind + 1;
}
par(mar = c(3.1, 3.3, 2.2, 1.7));
par(mgp = c(1.7, 0.6, 0))
order = order(pathSizes)
for (p in intPathways)
{
  verboseScatterplot(kME[[order[p]]][,1], kME[[order[p]]][,2],
                     main = spaste(LETTERS[ind], ". ",
                                   split[[order[p]]][1], "\n"),
                     abline = TRUE,
                     xlab = "kME in Human", ylab = "kME in Chimp", cex.main = 1.1, cex.lab = 1.1,
                     cex.axis = 1.1);
  ind = ind + 1;
}
```

```
par(mar = c(0,0.5,3,0.5));
for (set in 1:nSets)
{
  circlePlot(pathAdj[[set]], labels = pathLabels, order = geneOrder,
             main = spaste(LETTERS[ind], ". ", pathway, " pathway\n", setNames[set], " brain data"),
             radii = c(1.2, 0.55), lineColors = greenWhiteRed(50),
             min.cex.labels = 1.1, max.cex.labels = 1.1, xLabelOffset = 0, yLabelOffset = 0,
             plotBox = c(-1.7, 1.7, -1, 1));
  ind = ind + 1
}
dev.off();
```

The result is shown in Figure 1.

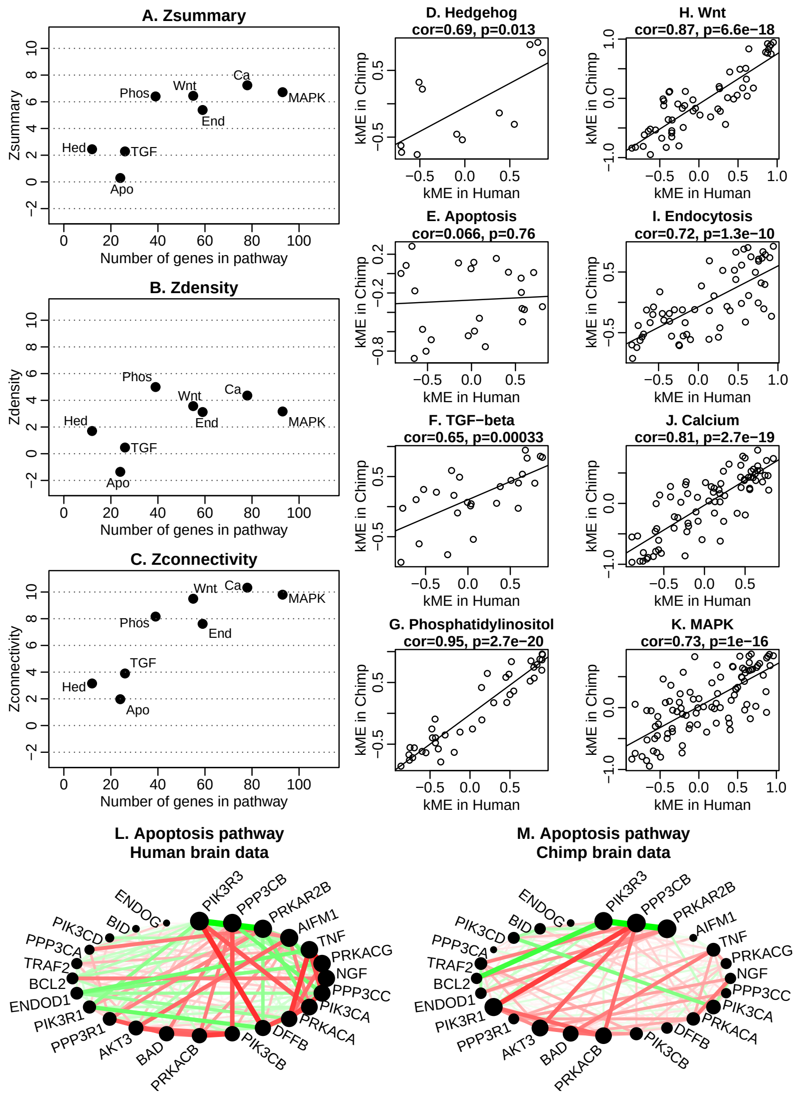

Figure 1: This figure reproduces Figure 7 in our main article. preserved between female liver and the muscle and brain tissues. In the upper row we present summary preservation Z statistics (y-axis) for selected KEGG pathways (interpreted as modules) versus the number of genes in the pathway (x-axis). Panel A shows $Z_{summary}$, panel B shows the density summary statistic $Z_{density}$, and panel C shows the connectivity summary statistic $Z_{connectivity}$. Pathway names are shortened for readability. Panel A shows that MAPK, Calcium, Endocytosis, Wnt, and Phosphatidylinositol show moderate to strong evidence of preservation ($Z_{summary}$ between 6 and 8) while the apoptosis pathway is not preserved as a network module. Panel C shows that this preservation signal mainly reflects connectivity preservation $Z_{connectivity}$ while panel B reveals that most modules have weak to moderate density preservation ($Z_{density} \lesssim 5$). Note that the apoptosis pathway shows no evidence of preservation in neither density nor connectivity. Panels D–H display scatter plots of eigengene-based connectivities in the chimpanzee data (y-axis) vs. in the human data (x-axis). Each point represents a gene in the pathway. Higher correlation means that the internal co-expression structure of the pathway is more strongly preserved. The apoptosis pathway has the lowest $cor.kME$ statistic, while the Phosphatidylinositol pathway has the highest.

We next create Figure 5 that compares network preservation statistics to the results of IGP. We first prepare all plotting variables.

```
ref = 1
test = 2
for (p in 1:nPathways) for (set in 1:nSets)
{
  pathCR[p, set] = cr[[p]][[set]]$Actual.IGP[2];
  randCR[p, set] = cr[[p]][[set]]$Actual.IGP[1];
  pCR[p, set] = cr[[p]][[set]]$p.value[2];
}
densCols = match(c("propVarExplained.pres", "meanSignAwareKME.pres", "meanSignAwareCorDat.pres",
                   "meanAdj.pres"), names(mp[[1]]$preservation$observed[[ref]][[test]]));
connCols = match(c("cor.kIM", "cor.kME", "cor.cor"),
                 names(mp[[1]]$preservation$observed[[ref]][[test]]));
nCols = length(densCols) + length(connCols);
obsData = matrix(0, nPathways, nCols);
for (p in 1:nPathways)
  obsData[p, ] = as.numeric(mp[[p]]$preservation$observed[[ref]][[test]][4, c(densCols, connCols)]);

ranks = apply(-obsData, 2, rank);
densRank = apply(ranks[, c(1:4)], 1, median);
connRank = apply(ranks[, c(5:7)], 1, median);
medianRank = pmedian(densRank, connRank);

nPlots = 6;
plotData = list(matrix(0, nPathways, nPlots), matrix(0, nPathways, nPlots));

plotData[[1]][, c(1:4)] = pathSizes;
plotData[[2]][, 1] = Zsummary[1,2, ];
plotData[[2]][, 2] = medianRank;
plotData[[2]][, 3] = pathCR[, 2];
plotData[[2]][, 4] = -log10(pCR[, 2]+1e-4);

plotData[[1]][, c(5,6)] = plotData[[2]][, c(1:2)];
plotData[[2]][, c(5,6)] = plotData[[2]][, 3];

xLabs = c(rep("Module size", 4), "Zsummary", "Median rank");
yLabs = c("Zsummary", "Median rank", "Observed IGP", "-log10(IGP perm p)", "Observed IGP", "Observed IGP");
mains = spaste(LETTERS[1:nPlots], ". ", #rep("Ref: Human, Test: Chimp\n", nPlots),
               c(yLabs[1:4], paste(yLabs[5:6], "vs.", xLabs[5:6])),
               c("", "", "", "", "\n", "\n"));

textLabels = ssPathNames = c("Wnt", "Hed", "Phos", "MAPK", "Ca", "TGF", "Apo", "End");
verbose = c(rep(FALSE, 4), rep(TRUE, 2));
ablines = list(c(0, 2, 10), NA, NA, c(-log10(0.05), -log10(0.05/nPathways)), NA, NA);
abColors = list(c("black", "blue", "darkgreen"), NA, NA, c("blue", "red"), NA, NA);
logs = c("x", "x", "x", "x", "", "");
invertY = c(FALSE, TRUE, rep(FALSE, 4));
verSP = function(...) { verboseScatterplot(..., abline = TRUE) }
```

Next comes the actual plotting code.

```
cexLabels = 1.4
sizeGrWindow(6.83,9.19);
#pdf(file = "Plots/KEGGpathways-NetworkAndIGPStatistics.pdf", w=6.83, h=9.19, onefile = FALSE);
par(mfrow = c(3,2));
par(mar = c(3.7, 3.7, 3.2, 0.5));
par(mgp = c(2, 0.6, 0))
set.seed(2)
```

```
for (p in 1:nPlots)
{
  x = plotData[[1]][, p];
  y = plotData[[2]][, p]
  miny = min(y, ablines[[p]], na.rm = TRUE);
  maxy = max(y, ablines[[p]], na.rm = TRUE);
  miny = miny - (maxy-miny)*0.1;
  maxy = maxy + (maxy-miny)*0.1;
  minx = min(x);
  maxx = max(x);
  minx = minx - (maxx-minx)*0.18;
  maxx = maxx + (maxx-minx)*0.18;

  (if (verbose[p]) verSP else plot ) (plotData[[1]][, p], plotData[[2]][, p],
                      main = mains[p],
                      xlab = xLabs[p],
                      ylab = yLabs[p],
                      cex.main = cexLabels, cex.lab = cexLabels, cex.axis = cexLabels,
                      #bg = colorLabels,
                      #col = colorLabels,
                      pch = 21, col = "black", bg = "black",
                      cex = 1,
                      ylim = if (invertY[p]) c(maxy, miny) else c(miny, maxy),
                      xlim = c(minx, maxx)
                      );

  labelPoints(plotData[[1]][, p], plotData[[2]][, p], textLabels, cex = cexLabels, offs = 0.06,
              jiggle = 0.05);
  if (!is.na(ablines[[p]][[1]]))
    for (al in 1:length(ablines[[p]]))
      abline(h = ablines[[p]][[al]], col = abColors[[p]][[al]], lty = 2);
}
# If plotting into a file, close it
dev.off();
```

The result is shown in Figure 2.

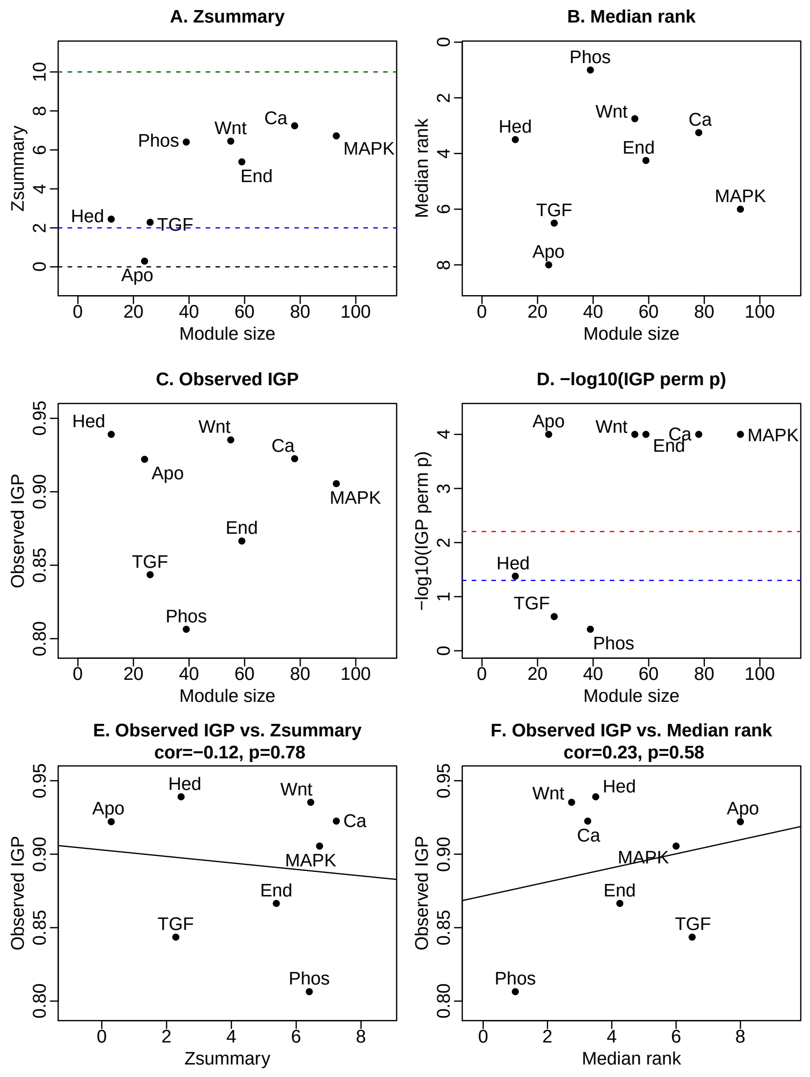

Figure 2: This figure reproduces Figure 6 in our main article. Here we compare the results of $Z_{summary}$ (panel A) to those of *medianRank* (panel B) and the IGP statistic (panels C and D). E. and F. show scatter plots between the observed IGP statistic and $Z_{summary}$ and *medianRank*, respectively. Here we find no significant relationship between the IGP statistic and the composite module preservation statistic. Since KEGG modules do not correspond to clusters, it is not clear whether cluster preservation statistics are useful in this example.

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] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl. Acids Res.*, 28(1):27–30, 2000.