Short tutorial on studying module preservation:
Preservation of female mouse liver modules in male data

Peter Langfelder and Steve Horvath

October 21, 2010

Contents

1 Overview

This tutorial presents an introductoryi, minimal analysis of module preservation between two independent data sets. The data we use here are expression measurements in livers of a mouse F2 cross described in [1]. We use the data set of female mice as the reference set, and the male mice as the test set. Because male and female mice are clearly very similar, one would expect that most modules will be preserved. In this sense, module preservation servers as validation for the modules found in the female data. However, metabolism is known to be a sexually dimorphic trait, and it is possible that some of the modules found in the female samples are female-specific and will not be found in the male data. Thus. module preservation can also serve as a tool for differential network analysis.

Here we simply load the data, calculate module preservation using the function `modulePreservation`, and briefly study the results.

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.

1.a 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);
```

2   Data input

We assume the user has downloaded the data for this tutorial and saved them in a local directory. The expression data is contained in two files `cnew_liver_bxh_f2female_8000mvgenes_p3600_UNIQUE_tommodules.xls.bz2` and `LiverMaleFromLiverFemale3600.csv.bz2` that come with this tutorial.

```
#Read in the female liver data set
# The following 3421 probe set were arrived at using the following steps
#1) reduce to the 8000 most varying, 2) 3600 most connected, 3) focus on unique genes
file = bzfile("cnew_liver_bxh_f2female_8000mvgenes_p3600_UNIQUE_tommodules.xls.bz2");
dat0=read.table(file, header=TRUE)
names(dat0)
```

In addition to expression data, the data files contain extra information about the surveyed genes we do not need. One can inspect larger data frames such as `femData` by invoking R data editor via `fix(femData)`. The expression data set contains 135 samples. Note that each row corresponds to a gene and column to a sample or auxiliary information. We now remove the auxiliary data and transpose the expression data for further analysis.

```
# this contains information on the genes
datSummary=dat0[,c(1:8,144:150)]
# the following data frame contains
# the gene expression data: columns are genes, rows are arrays (samples)
datExprFemale <- t(dat0[,9:143])
no.samples <- dim(datExprFemale)[[1]]
dim(datExprFemale)
# Set the columns names to probe names
colnames(datExprFemale) = datSummary$substanceBXH
# This module assignment was obtained by Ghazalpour et al
colorsFemale = dat0$module
```

We now similarly read in the male data and put the data in a suitable form:

```
file = bzfile("LiverMaleFromLiverFemale3600.csv.bz2");
data = read.csv(file, header = TRUE);
datExprMale = t(data[, substring(colnames(data), 1, 3)=="F2_"]);
colnames(datExprMale) = data$substanceBXH
```

In this case the gene names (column names) of the two data sets agree, but that need not always be the case. The calculation below will automatically ignore gene names that cannot be matched across data sets; the user should make sure that there are enough matching gene names that the calculation makes sense (e.g., at least half the genes in each module should be common between the reference and test data sets).

3   Calculation of module preservation

In this section we use the function `modulePreservation` to study module preservation of female modules in the male data. The function call is simple and most of this section will be devoted to the analysis of the results. We now set up the multi-set expression data and corresponding module colors:

```
setLabels = c("Female", "Male");
multiExpr = list(Female = list(data = datExprFemale), Male = list(data = datExprMale));
multiColor = list(Female = colorsFemale);
```

Roll the drums, here comes the calculation of module preservation. This calculation takes about 45 minutes on a fast desktop PC. If this code has been already executed once and the result is saved, it can be loaded from disk, thus saving time.

```
system.time( {
  mp = modulePreservation(multiExpr, multiColor,
                          referenceNetworks = 1,
                          nPermutations = 200,
                          randomSeed = 1,
                          quickCor = 0,
                          verbose = 3)
          } );
# Save the results
save(mp, file = "modulePreservation.RData");
```

Alternatively, if the data has already been calculated before, load them from disk:

```
load(file = "modulePreservation.RData")
```

4 Analysis and display of module preservation results

We now analyze the data. Isolate the observed statistics and their Z scores:

```
ref = 1
test = 2
statsObs = cbind(mp$quality$observed[[ref]][[test]][, -1], mp$preservation$observed[[ref]][[test]][, -1])
statsZ = cbind(mp$quality$Z[[ref]][[test]][, -1], mp$preservation$Z[[ref]][[test]][, -1]);
```

We look at the main output: the preservation *medianRank* and $Z_{summary}$ statistics.

```
# Compare preservation to quality:
print( cbind(statsObs[, c("medianRank.pres", "medianRank.qual")],
             signif(statsZ[, c("Zsummary.pres", "Zsummary.qual")], 2)) )
```

The numbers are nice, but (to paraphrase a saying) a picture is worth a thousand numbers. We plot the preservation *medianRank* and $Z_{summary}$ for the female modules as a function of module size. The plotting code may seem a bit involved, but it is worth going through.

```r
# Module labels and module sizes are also contained in the results
modColors = rownames(mp$preservation$observed[[ref]][[test]])
moduleSizes = mp$preservation$Z[[ref]][[test]][, 1];
# leave grey and gold modules out
plotMods = !(modColors %in% c("grey", "gold"));
# Text labels for points
text = modColors[plotMods];
# Auxiliary convenience variable
plotData = cbind(mp$preservation$observed[[ref]][[test]][, 2], mp$preservation$Z[[ref]][[test]][, 2])
# Main titles for the plot
mains = c("Preservation Median rank", "Preservation Zsummary");
# Start the plot
sizeGrWindow(10, 5);
#pdf(fi="Plots/BxHLiverFemaleOnly-modulePreservation-Zsummary-medianRank.pdf", wi=10, h=5)
par(mfrow = c(1,2))
par(mar = c(4.5,4.5,2.5,1))
for (p in 1:2)
{
  min = min(plotData[, p], na.rm = TRUE);
  max = max(plotData[, p], na.rm = TRUE);
  # Adjust ploting ranges appropriately
  if (p==2)
  {
    if (min > -max/10) min = -max/10
    ylim = c(min - 0.1 * (max-min), max + 0.1 * (max-min))
  } else
    ylim = c(max + 0.1 * (max-min), min - 0.1 * (max-min))
  plot(moduleSizes[plotMods], plotData[plotMods, p], col = 1, bg = modColors[plotMods], pch = 21,
       main = mains[p],
       cex = 2.4,
       ylab = mains[p], xlab = "Module size", log = "x",
       ylim = ylim,
       xlim = c(10, 2000), cex.lab = 1.2, cex.axis = 1.2, cex.main =1.4)
  labelPoints(moduleSizes[plotMods], plotData[plotMods, p], text, cex = 1, offs = 0.08);
  # For Zsummary, add threshold lines
  if (p==2)
  {
    abline(h=0)
    abline(h=2, col = "blue", lty = 2)
    abline(h=10, col = "darkgreen", lty = 2)
  }
}
# If plotting into a file, close it
dev.off();
```

The resulting plot is shown in Figure 1. We find that most modules show strong evidence of preservation. Th exceptions are the lightyellow and salmon modules whose $Z_{summary}$ statistics are 2.0 (borderline between no preservation and very weak preservation) and 7.6 (moderate preservation), respectively. The lightyellow and salmon modules also have to lowest median rank statistic, meaning their observed preservation statistics tend to be lowest among all modules. In Supplementary Text S5 published alongside the main module preservation paper we analyze the weak to non-existent preservation of these two modules and show that their presence in the female data is mainly due to an outlier sample. Thus, in this example module preservation statistics flag modules that are driven by technical artefacts.
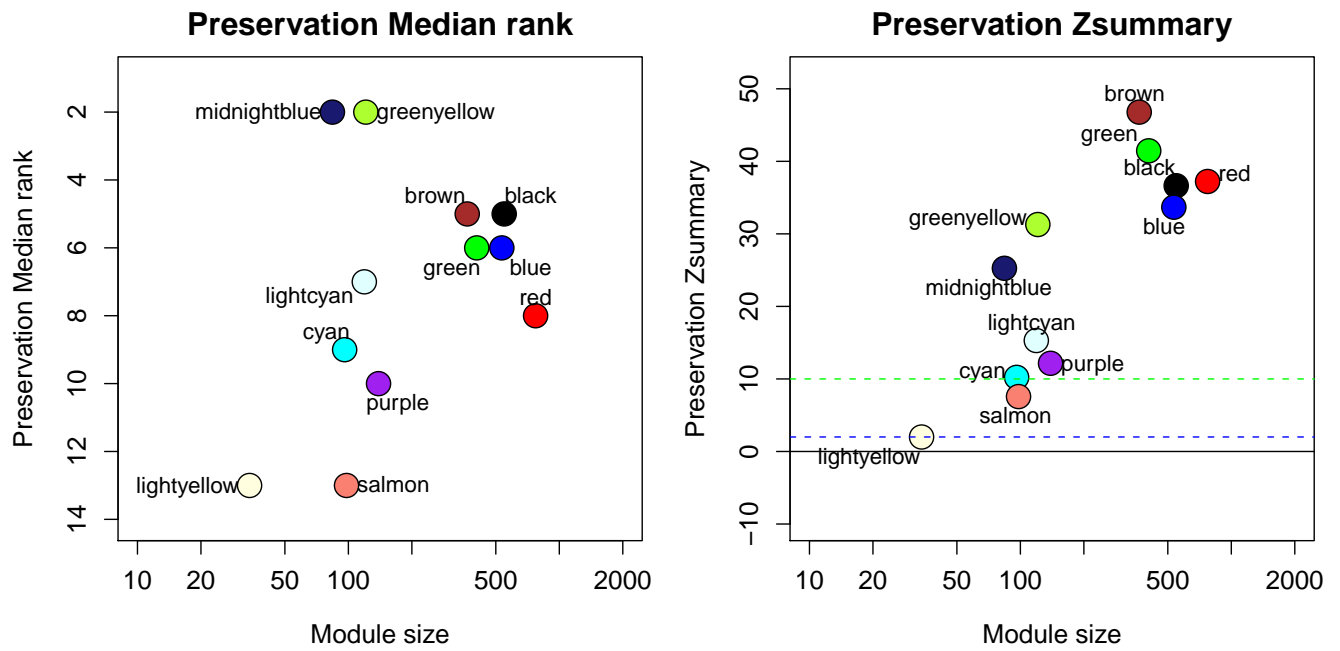
Figure 1: The *medianRank* and $Z_{summary}$ statistics of module preservation of female modules in male modules (y-axis) vs. module size (x-axis).

We now plot the density and connectivity statistics all in one plot. We include the module quality measures for comparison:

```r
# Re-initialize module color labels and sizes
modColors = rownames(statsZ)
moduleSizes = mp$quality$Z[[ref]][[test]][, 1];
# Exclude improper modules
plotMods = !(modColors %in% c("grey", "gold"));
# Create numeric labels for each module
labs = match(modColors[plotMods], standardColors(50));
# Start the plot: open a suitably sized graphical window and set sectioning and margins.
sizeGrWindow(12, 9);
par(mfrow = c(3,5))
par(mar = c(3,3,2,1))
par(mgp = c(1.6, 0.4, 0));
# Plot each Z statistic in a separate plot.
for (s in 1:ncol(statsZ))
{
  min = min(statsZ[plotMods, s], na.rm = TRUE);
  max = max(statsZ[plotMods, s], na.rm = TRUE);
  if (min > -max/5) min = -max/5
  plot(moduleSizes[plotMods], statsZ[plotMods, s], col = 1, bg = modColors[plotMods], pch = 21,
       main = colnames(statsZ)[s],
       cex = 1.7,
       ylab = colnames(statsZ)[s], xlab = "Module size", log = "x",
       ylim = c(min - 0.1 * (max-min), max + 0.1 * (max-min)),
       xlim = c(20, 1000))
  labelPoints(moduleSizes[plotMods], statsZ[plotMods, s], labs, cex = 0.7, offs = 0.04);
  abline(h=0)
  abline(h=2, col = "blue", lty = 2)
  abline(h=10, col = "darkgreen", lty = 2)
}
```

The result is shown in Figure 2. The translation table between module colors and the numeric labels can be obtained for example as follows:

```r
data.frame(color = modColors[plotMods], label = labs)
```

The result is

```
> data.frame(color = modColors[plotMods], label = labs)
          color label
1         black     7
2          blue     2
3         brown     3
4          cyan    14
5         green     5
6   greenyellow    11
7     lightcyan    16
8   lightyellow    19
9   midnightblue    15
10       purple    10
11          red     6
12       salmon    13
```

Of note is the fact the the quality (for example, Zsummary.qual) of the lightyellow and salmon modules is quite high, definitely on par with other modules, while their preservation is much lower. The all quality statistics point to bradly the same conclusion, but they measure different aspects of quality and hence are all somewhat different from

each other. Hence the usefulness of the Zsummary statistic. The same can be said about the preservation statistics - they are broadly similar but in details they differ, and summary is very useful to form an overall conclusion.

As a programming/graphics side note, some of the plots illustrate graphically the limitations of the function `labelPoints`. There are no miracles, not even in WGCNA.
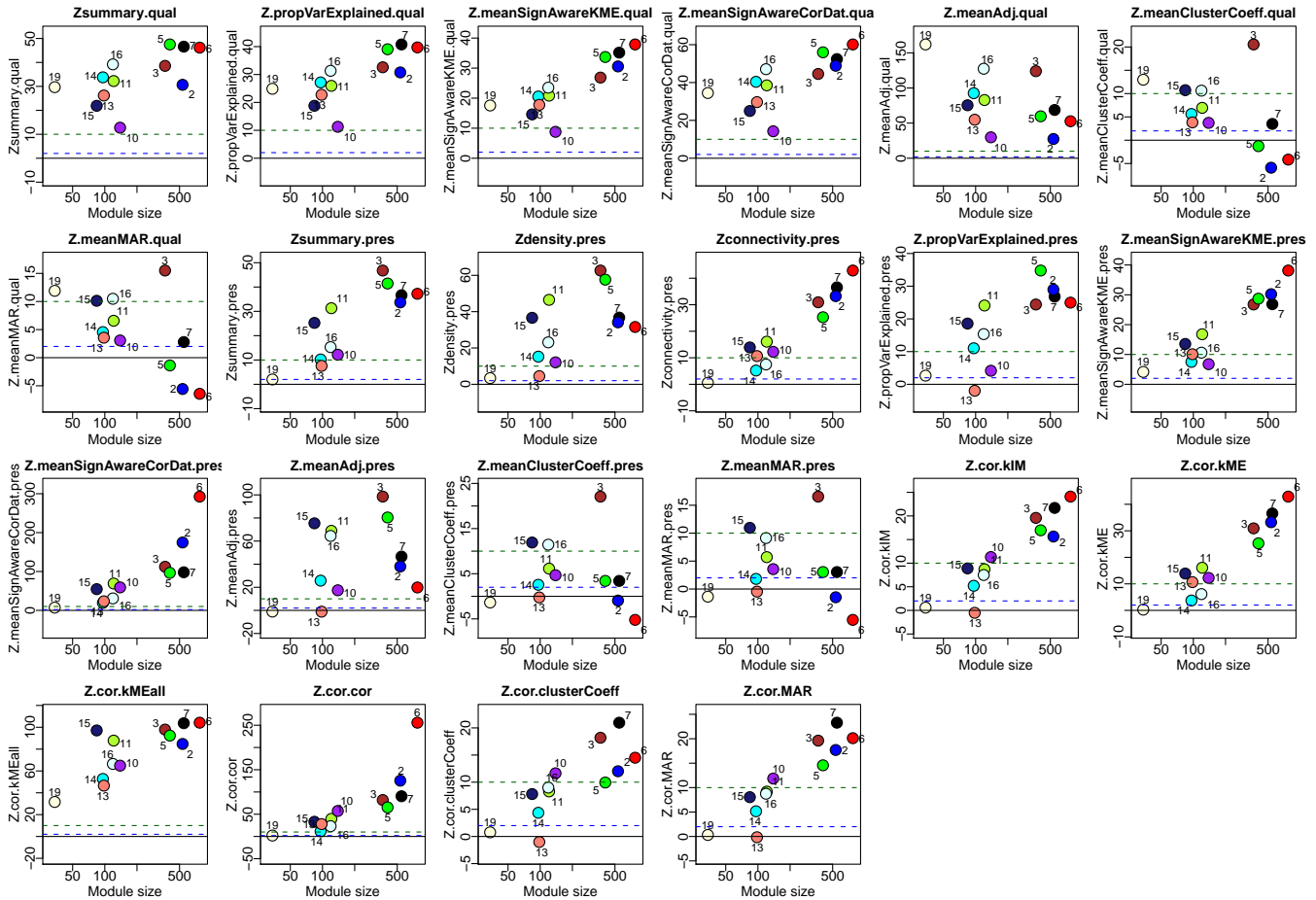

Figure 2: Statistics of module quality (first row; names end with suffix .qual) and module preservation (second and third row) of female modules. Modules are labeled by their color and by numeric labels (see text).

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.