Tutorial for the WGCNA package for R
II. Consensus network analysis of liver expression data, female and male mice

1. Data input and cleaning

Peter Langfelder and Steve Horvath

February 13, 2016

Contents

1 Data input, cleaning and pre-processing

1.a Loading expression data

The expression data is contained in two files that come with this tutorial, `LiverFemale3600.csv` and `LiverMale3600.csv`. After starting an R session, we check that the current directory is appropriately set, and load the requisite packages and the data. In addition to expression data, the data files contain extra information about the surveyed probes we do not need.

```
# 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);
#Read in the female liver data set
femData = read.csv("LiverFemale3600.csv");
# Read in the male liver data set
maleData = read.csv("LiverMale3600.csv");
# Take a quick look at what is in the data sets (caution, longish output):
dim(femData)
names(femData)
dim(maleData)
names(maleData)
```

The data sets contain roughly 130 samples each. Note that each row corresponds to a gene and column to a sample or auxiliary information. We now remove the auxiliary data and put the expression data into a multi-set format suitable for consensus analysis.

```
# We work with two sets:
nSets = 2;
# For easier labeling of plots, create a vector holding descriptive names of the two sets.
setLabels = c("Female liver", "Male liver")
shortLabels = c("Female", "Male")
# Form multi-set expression data: columns starting from 9 contain actual expression data.
multiExpr = vector(mode = "list", length = nSets)

multiExpr[[1]] = list(data = as.data.frame(t(femData[-c(1:8)])));
names(multiExpr[[1]]$data) = femData$substanceBXH;
rownames(multiExpr[[1]]$data) = names(femData)[-c(1:8)];
multiExpr[[2]] = list(data = as.data.frame(t(maleData[-c(1:8)])));
names(multiExpr[[2]]$data) = maleData$substanceBXH;
rownames(multiExpr[[2]]$data) = names(maleData)[-c(1:8)];
# Check that the data has the correct format for many functions operating on multiple sets:
exprSize = checkSets(multiExpr)
```

The variable `exprSize` contains useful information about the sizes of all of the data sets:

```
>exprSize

$nSets
[1] 2

$nGenes
[1] 3600

$nSamples
[1] 135 124

$structureOK
[1] TRUE
```

1.b Rudimentary data cleaning and outlier removal

We first identify genes and samples with excessive numbers of missing samples. These can be identified using the function `goodSamplesGenesMS`.

```
# Check that all genes and samples have sufficiently low numbers of missing values.
gsg = goodSamplesGenesMS(multiExpr, verbose = 3);
gsg$allOK
```

If the last statement returns `TRUE`, all genes and samples have passed the cuts. If it returns `FALSE`, the following code removes the offending samples and genes:

```
if (!gsg$allOK)
{
  # Print information about the removed genes:
  if (sum(!gsg$goodGenes) > 0)
    printFlush(paste("Removing genes:", paste(names(multiExpr[[1]]$data)[!gsg$goodGenes],
                                              collapse = ", ")))
  for (set in 1:exprSize$nSets)
  {
```

```
    if (sum(!gsg$goodSamples[[set]]))
      printFlush(paste("In set", setLabels[set], "removing samples",
                       paste(rownames(multiExpr[[set]]$data)[!gsg$goodSamples[[set]]], collapse = ", ")))
    # Remove the offending genes and samples
    multiExpr[[set]]$data = multiExpr[[set]]$data[gsg$goodSamples[[set]], gsg$goodGenes];
  }
  # Update exprSize
  exprSize = checkSets(multiExpr)
}
```

We now cluster the samples on their Euclidean distance, separately in each set.

```
sampleTrees = list()
for (set in 1:nSets)
{
  sampleTrees[[set]] = hclust(dist(multiExpr[[set]]$data), method = "average")
}
```

The easiest way to see the two dendrograms at the same time is to plot both into a pdf file that can be viewed using standard pdf viewers.

```
pdf(file = "Plots/SampleClustering.pdf", width = 12, height = 12);
par(mfrow=c(2,1))
par(mar = c(0, 4, 2, 0))
for (set in 1:nSets)
  plot(sampleTrees[[set]], main = paste("Sample clustering on all genes in", setLabels[set]),
       xlab="", sub="", cex = 0.7);
dev.off();
```

By inspection, there seems to be one outlier in the female data set, and no obvious outliers in the male set. We now remove the female outlier using a semi-automatic code that only requires a choice of a height cut. We first re-plot the two sample trees with the cut lines included (see Fig. 1):

```
# Choose the "base" cut height for the female data set
baseHeight = 16
# Adjust the cut height for the male data set for the number of samples
cutHeights = c(16, 16*exprSize$nSamples[2]/exprSize$nSamples[1]);
# Re-plot the dendrograms including the cut lines
pdf(file = "Plots/SampleClustering.pdf", width = 12, height = 12);
par(mfrow=c(2,1))
par(mar = c(0, 4, 2, 0))
for (set in 1:nSets)
{
  plot(sampleTrees[[set]], main = paste("Sample clustering on all genes in", setLabels[set]),
       xlab="", sub="", cex = 0.7);
  abline(h=cutHeights[set], col = "red");
}
dev.off();
```

Now comes the actual outlier removal:

```
for (set in 1:nSets)
{
  # Find clusters cut by the line
  labels = cutreeStatic(sampleTrees[[set]], cutHeight = cutHeights[set])
  # Keep the largest one (labeled by the number 1)
  keep = (labels==1)
  multiExpr[[set]]$data = multiExpr[[set]]$data[keep, ]
}
```

```
collectGarbage();
# Check the size of the leftover data
exprSize = checkSets(multiExpr)
exprSize
```

1.c Loading clinical trait data

We now read in the trait data and match the samples for which they were measured to the expression samples.

```
traitData = read.csv("ClinicalTraits.csv");
dim(traitData)
names(traitData)
# remove columns that hold information we do not need.
allTraits = traitData[, -c(31, 16)];
allTraits = allTraits[, c(2, 11:36) ];
# See how big the traits are and what are the trait and sample names
dim(allTraits)
names(allTraits)
allTraits$Mice
# Form a multi-set structure that will hold the clinical traits.
Traits = vector(mode="list", length = nSets);
for (set in 1:nSets)
{
  setSamples = rownames(multiExpr[[set]]$data);
  traitRows = match(setSamples, allTraits$Mice);
  Traits[[set]] = list(data = allTraits[traitRows, -1]);
  rownames(Traits[[set]]$data) = allTraits[traitRows, 1];
}
collectGarbage();
# Define data set dimensions
nGenes = exprSize$nGenes;
nSamples = exprSize$nSamples;
```

We now have the expression data for both sets in the variable `multiExpr`, and the corresponding clinical traits in the variable `Traits`. The last step is to save the relevant data for use in the subsequent analysis:

```
save(multiExpr, Traits, nGenes, nSamples, setLabels, shortLabels, exprSize,
    file = "Consensus-dataInput.RData");
```

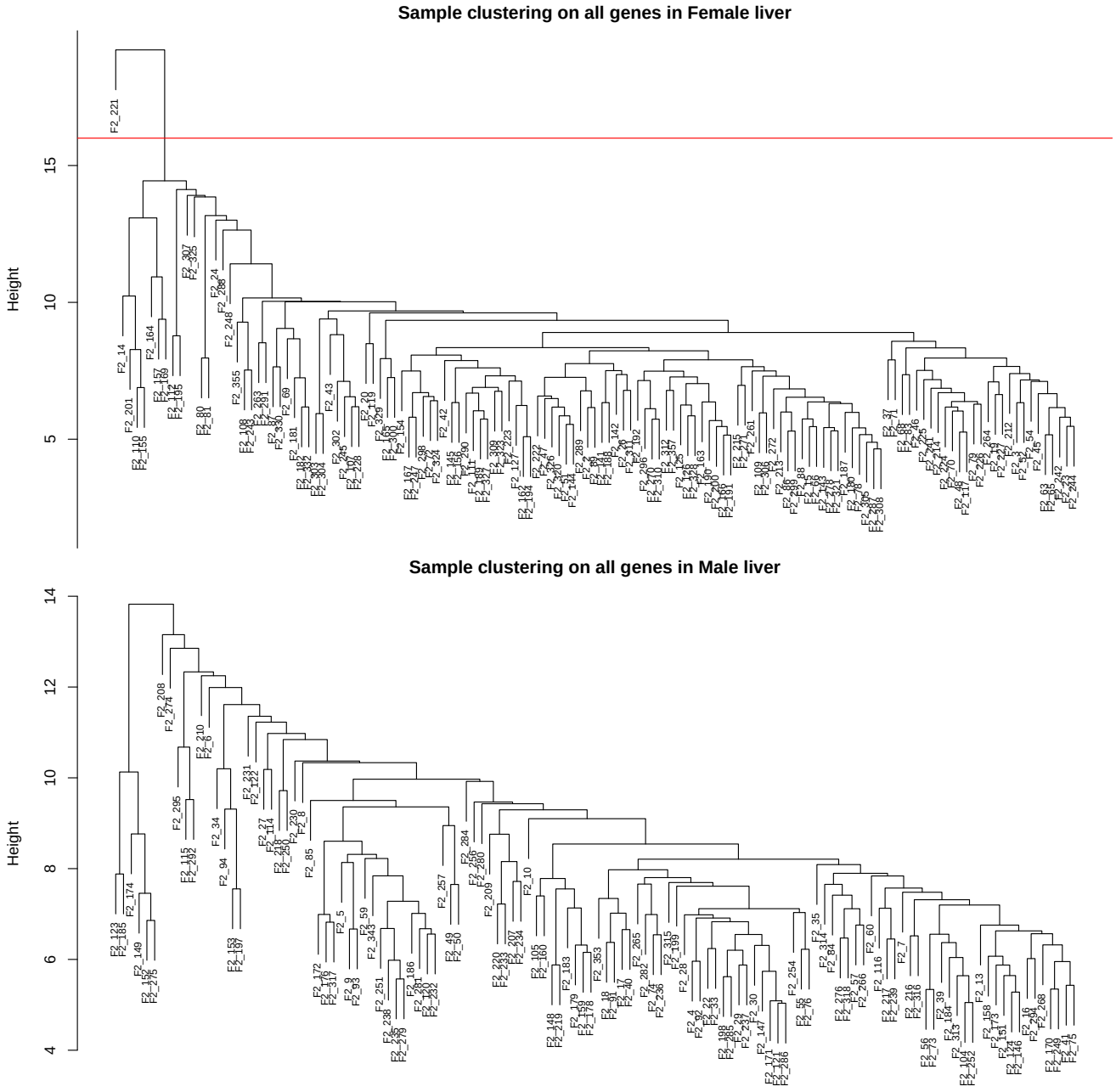

Figure 1: Clustering of samples in the two input sets, female and male liver expression data. The red line in the female dendrogram is the cut line for outlier detection; samples cut by this line are considered outliers. The corresponding cut line for the male samples lies outside of the dendrogram, indicating that no samples from the male set are considered outliers.