Tutorial for the WGCNA package for R:
III. Using simulated data to evaluate different module detection methods and gene screening approaches

2. Loading of expression data

Steve Horvath and Peter Langfelder

December 7, 2011

Contents

0 Setting up the R session

Before starting, the user should choose a working directory, preferably a directory devoted exclusively for this tutorial. After starting an R session, change working directory, load the requisite packages and set standard options:

```
# 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 WGCNA package
library(WGCNA)
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
```

2 Loading of expression and trait data

In this section we illustrate loading of expression data and clinical trait. The files holding the information are provided on the main tutorial page.

```
datGeneSummary=read.csv("GeneSummaryTutorial.csv")
datTraits=read.csv("TraitsTutorial.csv")
datMicroarrays=read.csv("MicroarrayDataTutorial.csv")
```

A quick look at the content of `datMicroarrays`:

```
> datMicroarrays[1:5,1:5]
      X GeneName     Sample1    Sample2    Sample3
1 Gene1    Gene1 -0.76903180  0.2010896 -0.8929247
```

```
2 Gene2    Gene2 -0.25281902  0.4293141 -1.1007358
3 Gene3    Gene3  0.33073345 -0.2741976  0.7029428
4 Gene4    Gene4  0.01335525  0.1155758 -0.4554590
5 Gene5    Gene5  0.30580959  0.1004434 -0.6086998
```

We now reformat the data and set appropriate gene and sample names:

```
# This vector contains the microarray sample names
ArrayName= names(data.frame(datMicroarrays[,-1]))
 # This vector contains the gene names
GeneName= datMicroarrays$GeneName
# We transpose the data so that the rows correspond to samples and the columns correspond to genes
# Since the first column contains the gene names, we exclude it.
datExpr=data.frame(t(datMicroarrays[,-1]))
names(datExpr)=datMicroarrays[,1]
dimnames(datExpr)[[1]]=names(data.frame(datMicroarrays[,-1]))
#Also, since we simulated the data, we know the true module color:
truemodule= datGeneSummary$truemodule
rm(datMicroarrays)
collectGarbage()
```

The first few entries in `datExpr` are now

```
> datExpr[1:5,1:5]
             Gene1       Gene2       Gene3       Gene4      Gene5
Sample1 -0.7690318 -0.25281902  0.33073345  0.01335525  0.3058096
Sample2  0.2010896  0.42931409 -0.27419755  0.11557575  0.1004434
Sample3 -0.8929247 -1.10073581  0.70294278 -0.45545899 -0.6086998
Sample4  0.2861690  0.05983439  0.02233195  0.04184443 -0.3252222
Sample5 -0.6224324 -0.61522751  0.80939447 -0.46023446 -1.0474164
```

The input gene expression data should have the above format where column names correspond to gene (or probe) names, row names correspond to array names. We now isolate the microarray trait y from the read data.

```
# First, make sure that the array names in the file datTraits line up with those in the microarray data
table( dimnames(datExpr)[[1]]==datTraits$ArrayName)
# Next, define the microarray sample trait
y=datTraits$y
```

Because the loaded data are identical to the simulated ones, we do not need to save the results here. Subsequent sections of the tutorial use the results of the data simulation section (Section 1).