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

## 6. Relating modules and module eigengenes to external 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, set standard options, and load the results of previous sections. Note that as of October 19, 2009, the function `plot.mat` that was part of the now defunct package sma is now part of the WGCNA package and is now named `plotMat`. Please upgrade the package if you have an older version without the function `plotMat`.

```
# 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);
# Load the previously saved data
load("Simulated-NetworkConstruction.RData");
attach(ModuleEigengeneNetwork1)
```

# 6 Relating modules and module eigengenes to external data

## 6.a Representing modules by eigengenes and relating eigengenes to one another

To get a sense of how related the modules are one can summarize each module by its eigengene (first principal component).

```
datME=moduleEigengenes(datExpr,colorh1)$eigengenes
signif(cor(datME, use="p"), 2)
```

The result is

```
> signif(cor(datME, use="p"), 2)
            MEblue MEbrown MEgreen  MEgrey MEturquoise MEyellow
MEblue       1.000  0.0770 -0.2100  0.0980      0.6600   -0.120
MEbrown      0.077  1.0000 -0.2800 -0.0084      0.1200    0.091
MEgreen     -0.210 -0.2800  1.0000  0.0320     -0.0021   -0.120
MEgrey       0.098 -0.0084  0.0320  1.0000      0.4200    0.660
MEturquoise  0.660  0.1200 -0.0021  0.4200      1.0000    0.095
MEyellow    -0.120  0.0910 -0.1200  0.6600      0.0950    1.000
```

We define a dissimilarity measure between the module eigengenes that keeps track of the sign of the correlation between the module eigengenes, and use it to cluster the eigengene:

```
dissimME=(1-t(cor(datME, method="p")))/2
hclustdatME=hclust(as.dist(dissimME), method="average" )
# Plot the eigengene dendrogram
par(mfrow=c(1,1))
plot(hclustdatME, main="Clustering tree based of the module eigengenes")
```

The resulting eigengene dendrogram is shown in Fig. 1.

### 6.a.1 Pairwise scatter plots of the samples (arrays) along the module eigengenes

We create a pairwise scatter plots of the samples (arrays) along the module eigengenes:

```
sizeGrWindow(8,9)
plotMEpairs(datME,y=y)
```

The plots is shown in Fig. 2. The module eigengenes (first PC) of different modules may be highly correlated. WGCNA can be interpreted as a biologically motivated data reduction scheme that allows for dependency between the resulting components. In contrast, principal component analysis imposes orthogonality between the components. Since modules may represent biological pathways there is no biological reason why modules should be orthogonal to each other.

## 6.b Relating observed module eigengenes to simulated true module eigengenes

We now relate the observed module eigengenes to the simulated true module eigengenes:

```
signif(cor(datME, ModuleEigengeneNetwork1[,-1]),2)
```

The result is

```
> signif(cor(datME, ModuleEigengeneNetwork1[,-1]),2)
            MEturquoise MEblue MEbrown MEgreen MEyellow
MEblue            0.680  1.000   0.066  -0.190  -0.1100
MEbrown           0.100  0.071   0.990  -0.280   0.0840
MEgreen           0.015 -0.210  -0.270   0.990  -0.1100
MEgrey            0.270  0.077  -0.140   0.047   0.0083
MEturquoise       1.000  0.660   0.110   0.021   0.1200
MEyellow          0.099 -0.120   0.110  -0.110   0.9900
```

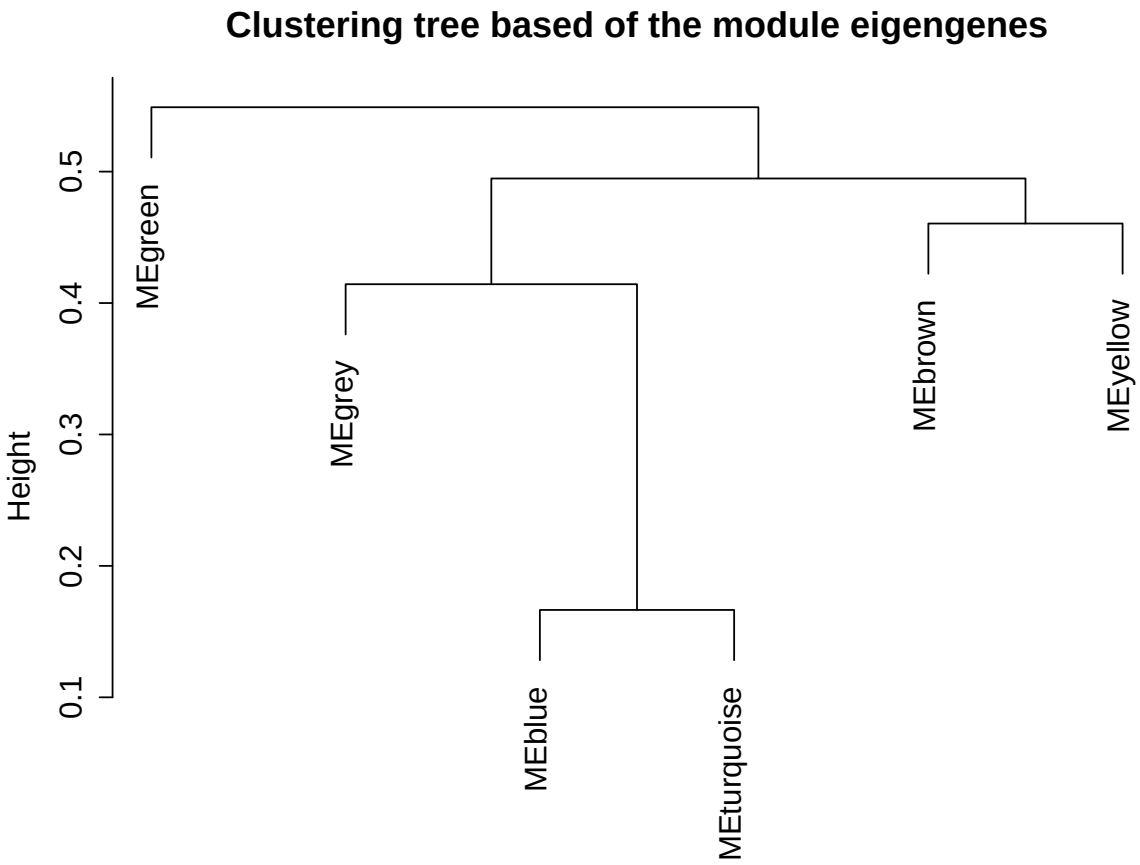


Figure 1: Hierarchical clustering dendrogram of module eigengenes. The blue and turquise eigengenes are highly related, as evidenced by their low merging height.

Here the rows correspond observed eigengenes, and the columns correspond to the simulated eigengenes. We find that the observed module eigengenes (MEblue, etc) are highly correlated ($r \geq 0.98$) with their true simulated counterparts. Although module detection may include noise genes, the module eigengene retrieval is highly robust [1]. This is because the module eigengenes definition is primarily based on the most highly connected intramodular hub genes. Genes at the fringe of the module barely affect the module definition.

## 6.c Diagnostics: heatmap plots of module expression

We now create a heatmap plots of module expressions. This may help identify modules that are "held together" by spurious correlations caused by outlying arrays.

```
sizeGrWindow(8,9)
par(mfrow=c(3,1), mar=c(1, 2, 4, 1))
which.module="turquoise";
plotMat(t(scale(datExpr[,colorh1==which.module ]) ),nrgcols=30,rlabels=T,
        clabels=T,rcols=which.module,
        title=which.module )
```

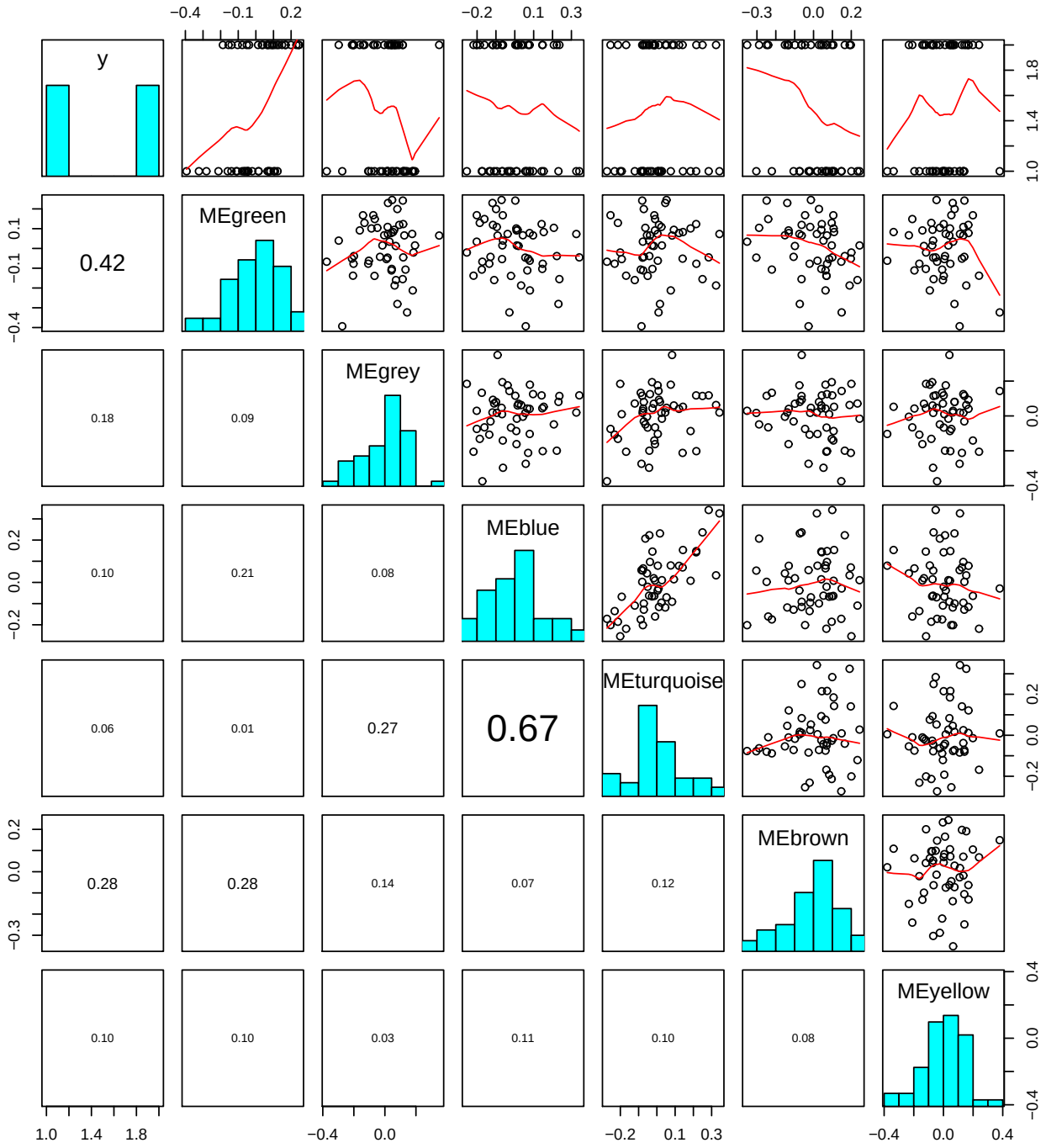


Figure 2: Pairwise scatterplots of module eigengenes and the trait $y$.

```
# for the second (blue) module we use
which.module="blue";
plotMat(t(scale(datExpr[,colorh1==which.module ]) ),nrgcols=30,rlabels=T,
        clabels=T,rcols=which.module,
        title=which.module )
which.module="brown";
plotMat(t(scale(datExpr[,colorh1==which.module ]) ),nrgcols=30,rlabels=T,
        clabels=T,rcols=which.module,
        title=which.module )
```

The result is shown in Fig. 3. Well defined modules results in characteristic band structures since the corresponding genes are highly correlated. The plots show no apparent outlier samples. The modules look fine.

## 6.d Diagnostics: displaying module heatmap and the eigengene

We now create a plot that explains the relationships between modules (heatmap) and the corresponding module eigengene (barplot):

```
sizeGrWindow(8,7);
which.module="green"
ME=datME[, paste("ME",which.module, sep="")]
par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2))
plotMat(t(scale(datExpr[,colorh1==which.module ]) ),
        nrgcols=30,rlabels=F,rcols=which.module,
        main=which.module, cex.main=2)
par(mar=c(5, 4.2, 0, 0.7))
barplot(ME, col=which.module, main="", cex.main=2,
       ylab="eigengene expression",xlab="array sample")
```

The plot is shown in Fig. 4. Note that the module eigengene takes on low values in arrays where a lot of module genes are under-expressed (green color in the heatmap). The ME takes on high values for arrays where a lot of module genes are over-expressed (red in the heatmap). ME can be considered the most representative gene expression profile of the module.

## 6.e Finding modules that relate to a clinical trait

There are two approaches one can follow. We discuss each below.

### 6.e.1 Correlate the module eigengenes with the trait

```
signif(cor(y,datME, use="p"),2)
```

```
> signif(cor(y,datME, use="p"),2)
     MEblue MEbrown MEgreen MEgrey MEturquoise MEyellow
[1,]   -0.1   -0.28    0.42  -0.18       0.056    0.099
```

The green and the brown module look most interesting. An advantage of this approach is that it allows ut to compute p-values using

```
cor.test(y, datME$MEbrown)
```

```
> cor.test(y, datME$MEbrown)

Pearson's product-moment correlation

data:  y and datME$MEbrown
```

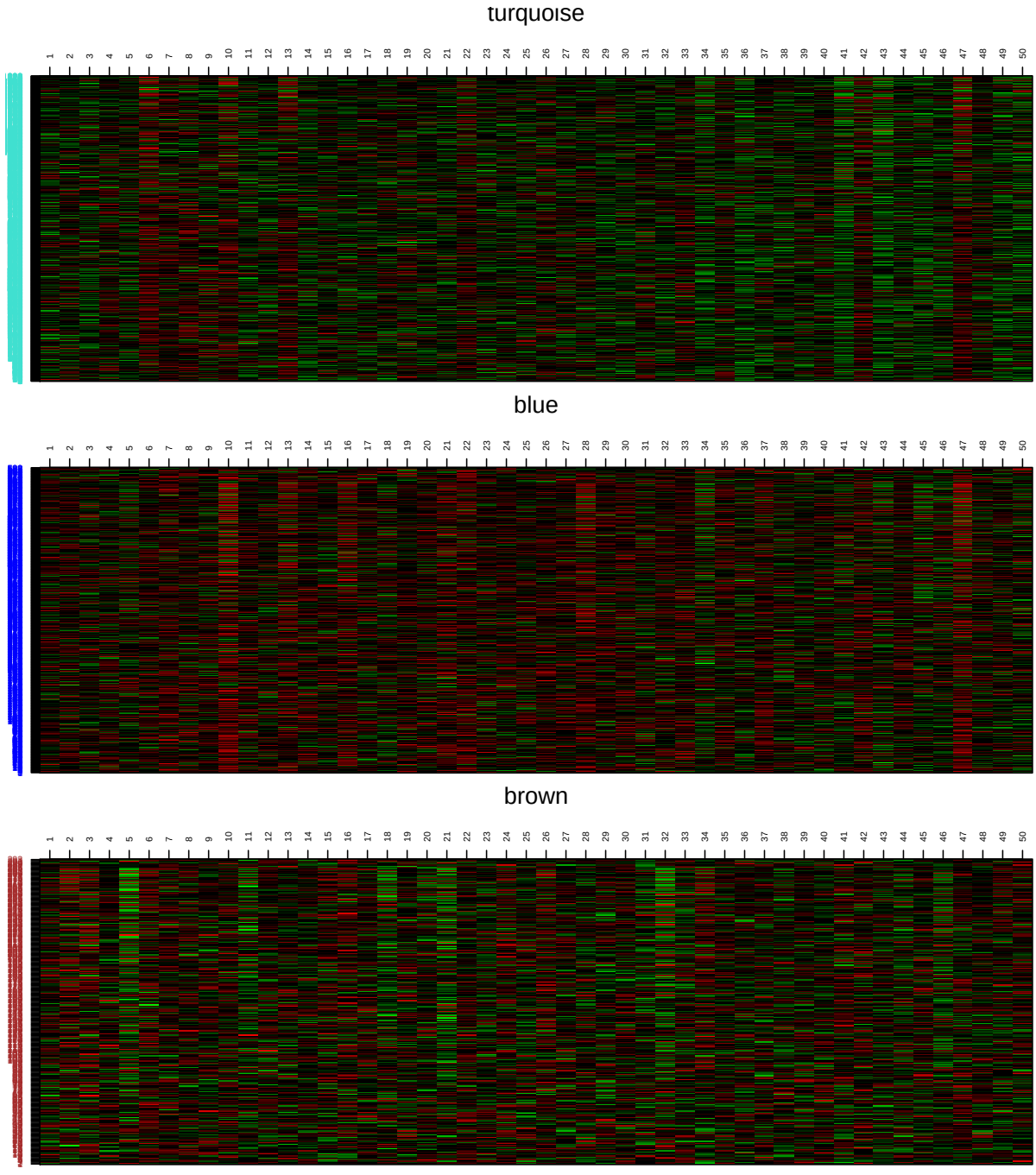


Figure 3: Module heatmaps. The rows correspond to genes and the columns to samples; green denotes under-expression and red over-expression. Well defined modules results in characteristic band structures since the corresponding genes are highly correlated. The plots show no apparent outlier samples. The modules look fine.

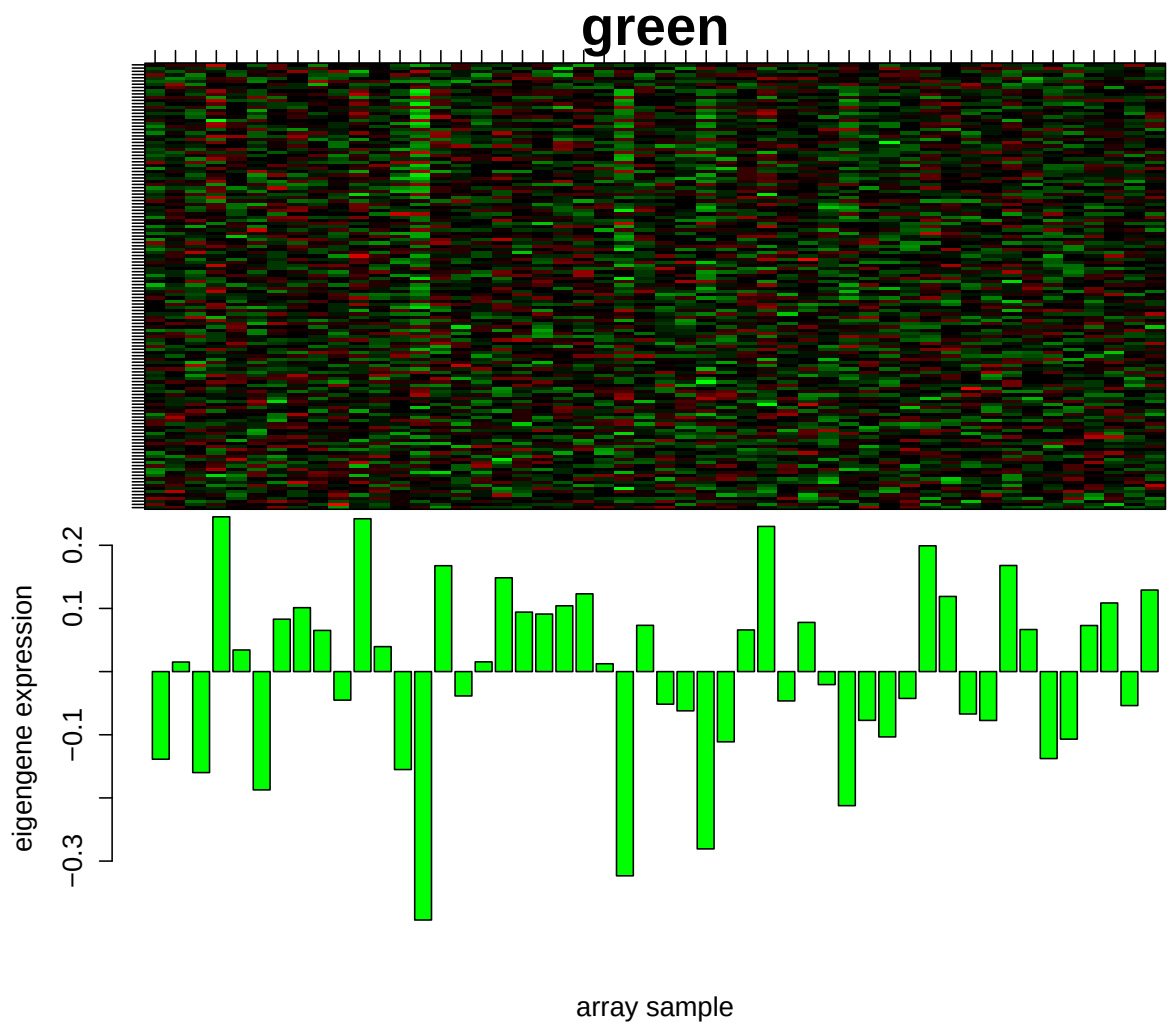


Figure 4: The top row shows theatmap of the green module genes (rows) across the microarrays (columns). The lower row shows the corresponding module eigengene expression values ($y$-axis) versus the same microarray samples. Note that the module eigengene takes on low values in arrays where a lot of module genes are under-expressed (green color in the heatmap). The ME takes on high values for arrays where a lot of module genes are over-expressed (red in the heatmap). ME can be considered the most representative gene expression profile of the module.

```
t = -1.9826, df = 48, p-value = 0.05315
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.514098581  0.003495354
sample estimates:
       cor
-0.2751200
```

The following code can be used to get all p-values:

```
p.values = corPvalueStudent(cor(y,datME, use="p"), nSamples = length(y))
```

### 6.e.2 Measure of module significance as average gene significance

One can also define a measure of module significance as the average gene significance of all genes in the module. We use the absolute value for defining a correlation based gene significance measure.

```
GS1=as.numeric(cor(y,datExpr, use="p"))
GeneSignificance=abs(GS1)
# Next module significance is defined as average gene significance.
ModuleSignificance=tapply(GeneSignificance, colorh1, mean, na.rm=T)
```

Here is the result:

```
> ModuleSignificance
     blue     brown     green      grey turquoise    yellow
0.1066630 0.1526828 0.2110672 0.1117847 0.1043860 0.1026779
```

To plot module significance, one can use

```
sizeGrWindow(8,7)
par(mfrow = c(1,1))
plotModuleSignificance(GeneSignificance,colorh1)
```

The result is shown in Fig. 5. The advantage of this second approach is that it can be used for any gene significance measure. A gene significance measure could be defined without reference to a sample trait. For example, it could indicate pathway membership (1 or 0) or gene essentiality (1 or 0), etc.
The next logical step in an analysis of empirical data would be to carry out a functional enrichment analysis of each module, for example using the software EASE (David) at `http://david.abcc.ncifcrf.gov/summary.jsp`. We refer the reader to Tutorial I for an example of a functional enrichment analysis. Before we end, we again save calculated results for use in subsequent sections.

```
collectGarbage()
save.image("Simulated-RelatingToExt.RData")
```

# References


[1] P. Langfelder and S. Horvath. Eigengene networks for studying the relationships between co-expression modules. *BMC Systems Biology*, 1:54, 2007.

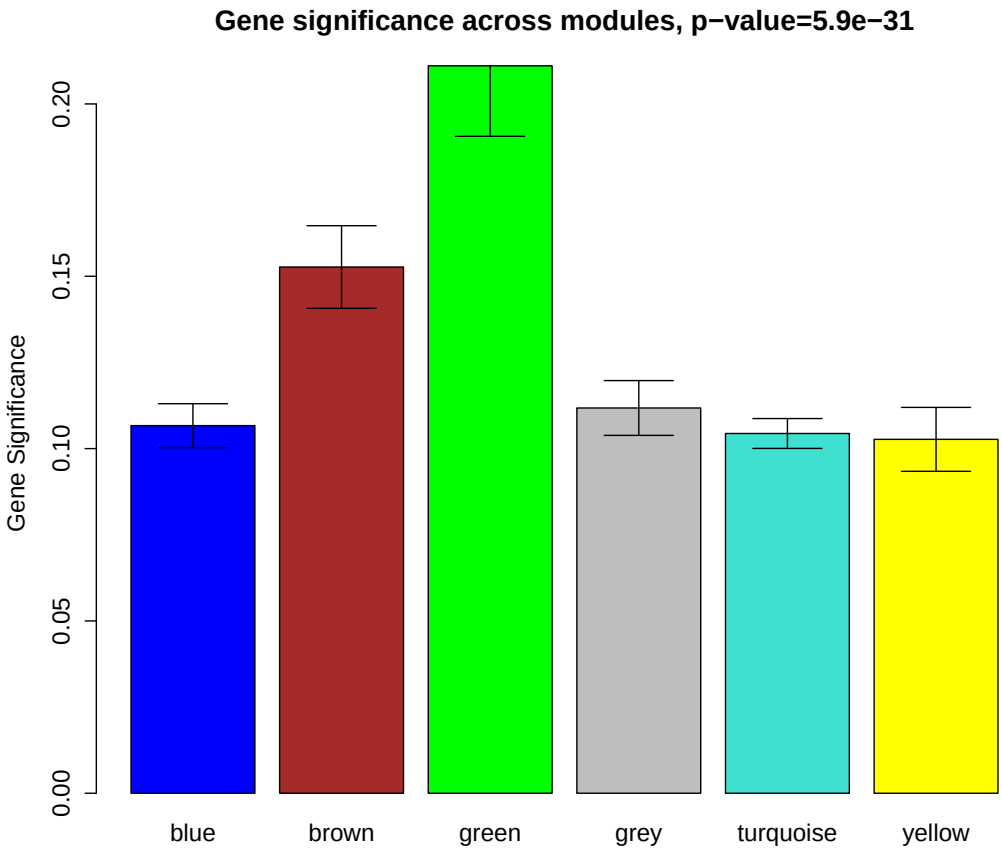


Figure 5: Barplot of module significance defined as the mean gene significance across all genes in the module. The green and brown modules are the most promising.