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

7. Module membership, intramodular connectivity and screening for intramodular hub genes

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:

```
# 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)
```

```
library(cluster)
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE);
# Load the previously saved data
load("Simulated-RelatingToExt.RData");
```

7 Intramodular connectivity, module membership, and screening for intramodular hub genes

7.a Intramodular connectivity

We begin by calculating the intramodular connectivity for each gene. (In network literature, connectivity is often referred to as "degree".) The function intramodularConnectivity computes the whole network connectivity `kTotal`, the within module connectivity `kWithin`, `kOut=kTotal-kWithin`, and `kDiff=kIn-kOut=2*kIN-kTotal`

```
ADJ1=abs(cor(datExpr,use="p"))^6
Alldegrees1=intramodularConnectivity(ADJ1, colorh1)
head(Alldegrees1)
```

An example output is shown here:

```
> head(Alldegrees1)
        kTotal  kWithin     kOut    kDiff
Gene1 31.80186 28.37595 3.425906 24.95005
Gene2 28.88249 26.47896 2.403522 24.07544
Gene3 25.38600 23.11852 2.267486 20.85103
Gene4 24.01574 22.12962 1.886122 20.24350
Gene5 24.93663 21.69175 3.244881 18.44687
Gene6 25.91260 23.92613 1.986469 21.93966
```

7.b Relationship between gene significance and intramodular connectivity

We plot the gene significance against intramodular connectivity:

```
colorlevels=unique(colorh1)
sizeGrWindow(9,6)
par(mfrow=c(2,as.integer(0.5+length(colorlevels)/2)))
par(mar = c(4,5,3,1))
for (i in c(1:length(colorlevels)))
{
  whichmodule=colorlevels[[i]];
  restrict1 = (colorh1==whichmodule);
  verboseScatterplot(Alldegrees1$kWithin[restrict1],
                     GeneSignificance[restrict1], col=colorh1[restrict1],
                     main=whichmodule,
                     xlab = "Connectivity", ylab = "Gene Significance", abline = TRUE)
}
```

The resulting plot is shown in Fig. 1. For the green and the brown module we observe that intramodular hub genes tend to have high gene significance. The opposite is true in the turquoise module.

7.c Generalizing intramodular connectivity for all genes on the array

The intramodular connectivity measure is only defined for the genes inside a given module. But in practice it can be very important to measure how connected a given genes is to biologically interesting modules. Toward this end, we

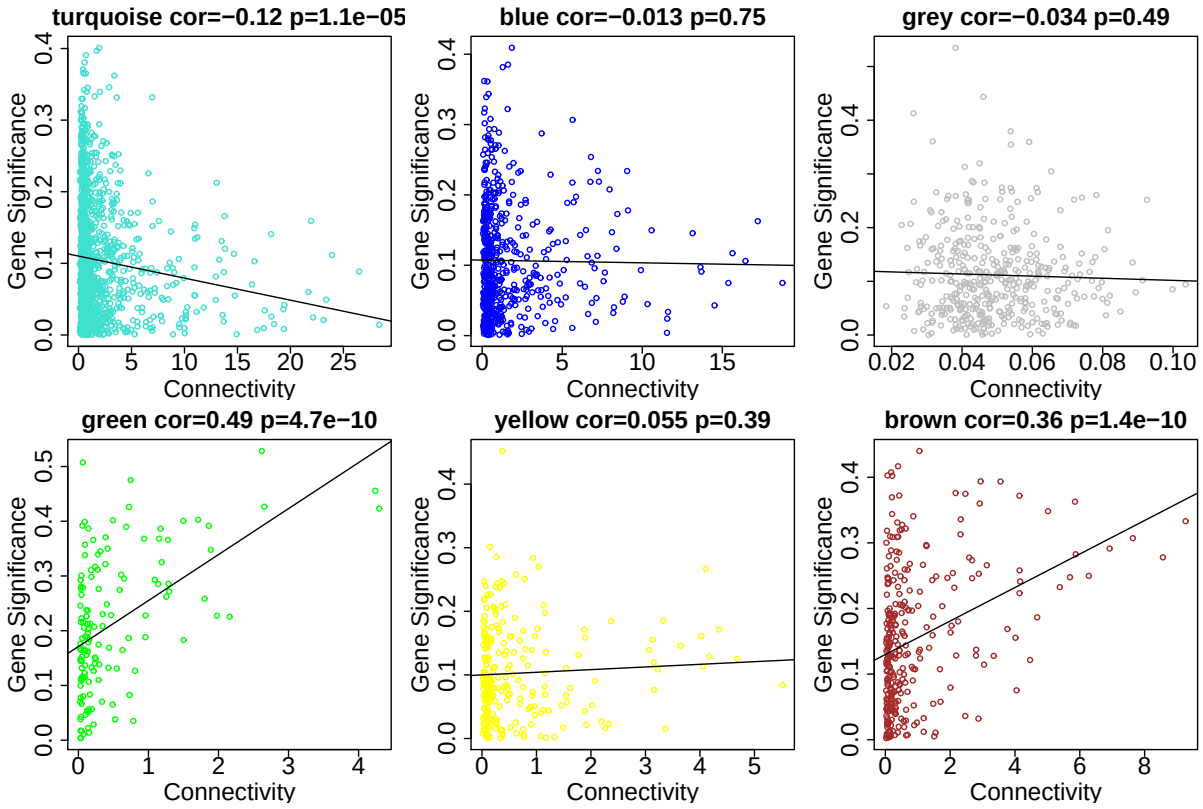

Figure 1: Gene significance (y-axis) vs. intramodular connectivity (x-axis) plotted separately for each module in the simulated data set. For the green and the brown module we observe that intramodular hub genes tend to have high gene significance. The opposite is true in the turquoise module.

define a module eigengene-based connectivity measure for each gene as the correlation between a the gene expression and the module eigengene. Specifically,

$$kME_{brown}(i) = \text{cor}(x_i, ME_{brown}) \tag{1}$$

where x_i is the gene expression profile of gene i and ME_{brown} is the module eigengene of the brown module. Note that the definition does not require that gene i is a member of the brown module. We define a data frame containing he module membership (MM) values for each module. In the past, we called the module membership values kME.

```
datKME=signedKME(datExpr, datME, outputColumnName="MM.")
# Display the first few rows of the data frame
head(datKME)
```

The output is

```
> head(datKME)
         MM.blue    MM.brown      MM.green    MM.grey MM.turquoise   MM.yellow
Gene1  0.6830511  0.11547756 -0.007124794  0.2840109    0.9481457  0.09588170
Gene2  0.6342657  0.02257975  0.080277091  0.3029967    0.9356343  0.06889483
Gene3 -0.6198067 -0.12531203  0.008372054 -0.2776929   -0.9121710 -0.17852211
Gene4  0.5966736  0.06469079  0.049862112  0.2671967    0.9052030  0.11707603
Gene5  0.6642214  0.14369720 -0.017975774  0.2442237    0.9017972 -0.01038067
Gene6 -0.6018161 -0.15167072  0.006667131 -0.2053897   -0.9192597 -0.17138960
```

We have a module membership value for each gene in each module.

7.d Finding genes with high gene significance and high intramodular connectivity in interesting modules

Our previous analysis has shown that the brown module is an "interesting"module in that its module significance is high. Here we show how to find genes with high gene significance and high intramodular connectivity in the brown module.

```
FilterGenes= abs(GS1)> .2 & abs(datKME$MM.brown)>.8
table(FilterGenes)
```

```
> table(FilterGenes)
FilterGenes
FALSE  TRUE
 2989    11
```

Which genes were filtered in?

```
dimnames(data.frame(datExpr))[[2]][FilterGenes]
```

```
> dimnames(data.frame(datExpr))[[2]][FilterGenes]
 [1] "Gene1051" "Gene1052" "Gene1053" "Gene1054" "Gene1056" "Gene1057"
 [7] "Gene1059" "Gene1060" "Gene1061" "Gene1063" "Gene1075"
```

7.e Relationship between the module membership measures (e.g. MM.turquoise) and intramodular connectivity

We now explore the relationship between the module membership measures (e.g. `MM.turquoise`) and intramodular connectivity.

```
sizeGrWindow(8,6)
par(mfrow=c(2,2))
# We choose 4 modules to plot: turquoise, blue, brown, green.
# For simplicity we write the code out explicitly for each module.
which.color="turquoise";
restrictGenes=colorh1==which.color
verboseScatterplot(Alldegrees1$kWithin[ restrictGenes],
                   (datKME[restrictGenes, paste("MM.", which.color, sep="")])^6,
                   col=which.color,
                   xlab="Intramodular Connectivity",
                   ylab="(Module Membership)^6")

which.color="blue";
restrictGenes=colorh1==which.color
verboseScatterplot(Alldegrees1$kWithin[ restrictGenes],
                   (datKME[restrictGenes, paste("MM.", which.color, sep="")])^6,
                   col=which.color,
                   xlab="Intramodular Connectivity",
                   ylab="(Module Membership)^6")

which.color="brown";
restrictGenes=colorh1==which.color
verboseScatterplot(Alldegrees1$kWithin[ restrictGenes],
                   (datKME[restrictGenes, paste("MM.", which.color, sep="")])^6,
                   col=which.color,
                   xlab="Intramodular Connectivity",
                   ylab="(Module Membership)^6")
```

```
which.color="green";
restrictGenes=colorh1==which.color
verboseScatterplot(Alldegrees1$kWithin[ restrictGenes],
                   (datKME[restrictGenes, paste("MM.", which.color, sep="")])^6,
                   col=which.color,
                   xlab="Intramodular Connectivity",
                   ylab="(Module Membership)^6")
```

The resulting plot is shown in Fig. 2. Note that after raising the module membership to a power of 6, it is highly correlated with the intramodular connectivity (`kWithin`).

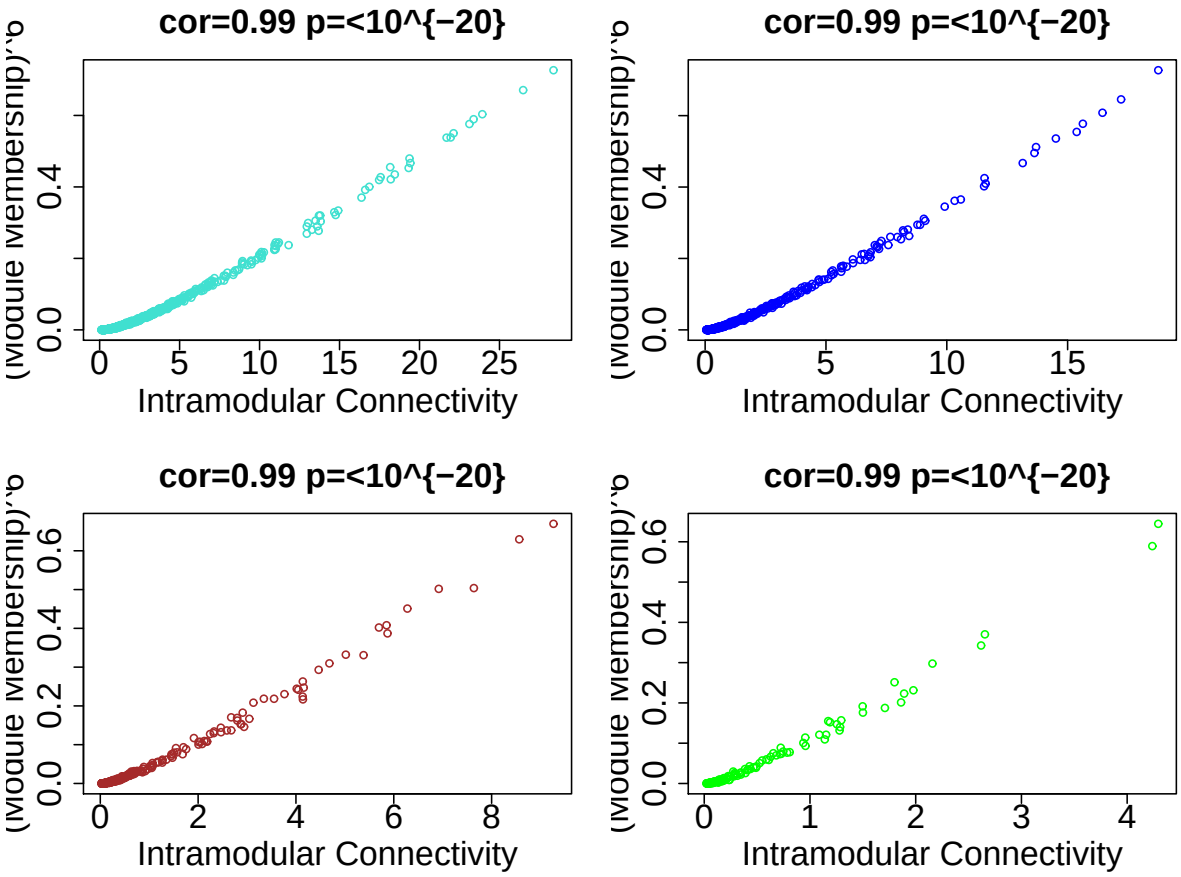

Figure 2: Module membership raised to power 6 (y-axis) vs. intramodular connectivity (x-axis) plotted separately for selected modules in the simulated data set. After raising the module membership to a power of 6, it is highly correlated with the intramodular connectivity (`kWithin`).

7.f Gene screening method based on a detailed definition module membership

The package contains a function for gene screening based on gene significance and module membership:

```
NS1=networkScreening(y=y, datME=datME, datExpr=datExpr,
        oddPower=3, blockSize=1000, minimumSampleSize=4,
        addMEy=TRUE, removeDiag=FALSE, weightESy=0.5)
```

We now compare the results of this detailed network screening analysis with that of the standard screening. How many genes in the returned list are noise genes?

```
# network screening analysis
mean(NoiseGeneIndicator[rank(NS1$p.Weighted,ties.method="first")<=100])
# standard analysis based on the correlation p-values (or Student T test)
mean(NoiseGeneIndicator[rank(NS1$p.Standard,ties.method="first")<=100])
```

```
> mean(NoiseGeneIndicator[rank(NS1$p.Weighted,ties.method="first")<=100])
[1] 0.2
> mean(NoiseGeneIndicator[rank(NS1$p.Standard,ties.method="first")<=100])
[1] 0.48
```

Here the network screening analysis leads to a top 100 list with the fewer number of noise genes. We now compare WGCNA based screening (based on p.Weighted) with standard screening by assessing the proportion of noise genes in gene lists created by ranking the p-values.

```
topNumbers=c(10,20,50,100)
for (i in c(1:length(topNumbers)) )
{
  print(paste("Proportion of noise genes in the top", topNumbers[i], "list"))
  WGCNApropNoise=mean(NoiseGeneIndicator[rank(NS1$p.Weighted,ties.method="first")<=topNumbers[i]])
  StandardpropNoise=mean(NoiseGeneIndicator[rank(NS1$p.Standard,ties.method="first")<=topNumbers[i]])
  print(paste("WGCNA, proportion of noise=", WGCNApropNoise,
        ", Standard, prop. noise=", StandardpropNoise))
  if (WGCNApropNoise< StandardpropNoise) print("WGCNA wins")
  if (WGCNApropNoise==StandardpropNoise) print("both methods tie")
  if (WGCNApropNoise>StandardpropNoise) print("standard screening wins")
}
```

```
[1] "Proportion of noise genes in the top 10 list"
[1] "WGCNA, proportion of noise= 0 , Standard, prop. noise= 0.4"
[1] "WGCNA wins"
[1] "Proportion of noise genes in the top 20 list"
[1] "WGCNA, proportion of noise= 0.1 , Standard, prop. noise= 0.4"
[1] "WGCNA wins"
[1] "Proportion of noise genes in the top 50 list"
[1] "WGCNA, proportion of noise= 0.14 , Standard, prop. noise= 0.4"
[1] "WGCNA wins"
[1] "Proportion of noise genes in the top 100 list"
[1] "WGCNA, proportion of noise= 0.2 , Standard, prop. noise= 0.48"
[1] "WGCNA wins"
```

In this data set, WGCNA-based screening leads to gene lists that contain fewer numbers of noise genes. Before we proceed, we remove some of the large data frames from the R session

```
rm(dissTOM); collectGarbage()
```

7.g Comparing the weighted correlation with the standard Pearson correlation

We now compare the weighted correlation with the standard Pearson correlation.

```
#Form a data frame containing standard and network screening results
CorPrediction1=data.frame(GS1,NS1$cor.Weighted)
cor.Weighted=NS1$cor.Weighted
# Plot the comparison
sizeGrWindow(8, 6)
verboseScatterplot(cor.Weighted, GS1,
```

```
        main="Network-based weighted correlation versus Pearson correlation\n",
        col=truemodule, cex.main = 1.2)
abline(0,1)
```

The plot is shown in Fig. 3. This example illustrates the central idea of network screening: WGCNA amplifies correlations of genes that are members of outcome related modules. The standard method (simple Pearson correlation) ignores module membership information.

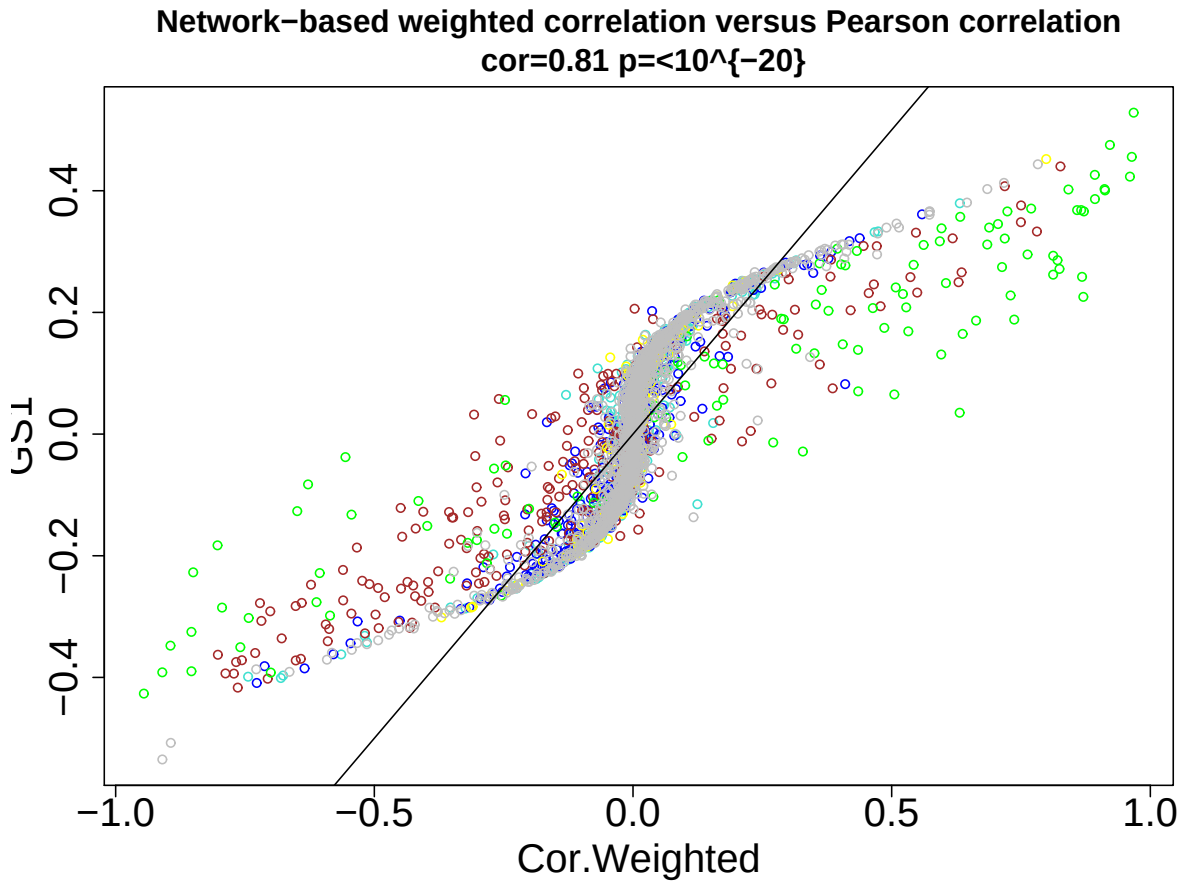

Figure 3: Trait-based gene significance in the training set (y-axis) plotted against the network-based significance (x-axis). WGCNA amplifies correlations of genes that are members of outcome related modules. The standard method (simple Pearson correlation) ignores module membership information.

7.h Simulating a large test set

We now simulate a *test set* comprised of thousands of observations. We choose a very large number of observation so that the test set will approximate the truth. But heed the words of Agent Scully (TV show X-files) "the truth is out there but so are the lies".

```
set.seed(2)
nSamples2=2000
MEgreen=rnorm(nSamples2)
scaledy2=MEgreen*ESgreen+sqrt(1-ESgreen^2)*rnorm(nSamples2)
y2=ifelse( scaledy2>median(scaledy2),2,1)
MEturquoise= ESturquoise*scaledy2+sqrt(1-ESturquoise^2)*rnorm(nSamples2)
```

```
# we simulate a strong dependence between MEblue and MEturquoise
MEblue= .6*MEturquoise+ sqrt(1-.6^2) *rnorm(nSamples2)
MEbrown= ESbrown*scaledy2+sqrt(1-ESbrown^2)*rnorm(nSamples2)
MEyellow= ESyellow*scaledy2+sqrt(1-ESyellow^2)*rnorm(nSamples2)
# Put together a data frame of eigengenes
ModuleEigengeneNetwork2=data.frame(y=y2,MEturquoise,MEblue,MEbrown,MEgreen, MEyellow)
# Simulate the expression data
dat2=simulateDatExpr5Modules(MEturquoise=ModuleEigengeneNetwork2$MEturquoise,
   MEblue=ModuleEigengeneNetwork2$MEblue,MEbrown=ModuleEigengeneNetwork2$MEbrown,
   MEyellow=ModuleEigengeneNetwork2$MEyellow,
   MEgreen=ModuleEigengeneNetwork2$MEgreen,simulateProportions=simulateProportions1,
   nGenes=nGenes1)
# recall that this is the signed gene significance in the training data
GS1= as.numeric(cor(y, datExpr, use="p"))
# the following is the signed gene significance in the test data
GS2=as.numeric( cor(ModuleEigengeneNetwork2$y, dat2$datExpr, use="p"))
```

7.i Preservation of signed gene significance between training and test data

We now evaluate whether the signed gene significance measure `GS=cor(y,datExpr)` is preserved between training and test data:

```
sizeGrWindow(8,6)
par(mfrow=c(1,1))
verboseScatterplot(GS1,GS2,
      main="Trait-based gene significance in test set vs. training set\n",
      xlab = "Training set gene significance",
      ylab = "Test set gene significance",
      col=truemodule, cex.main = 1.4)
```

The resulting plot is shown in Fig. 4. The gene significance of the training set is correlated with that of the test set. Note that the reproducible signal comes from the brown and green module genes. Recall that these modules were simulated to be related to the outcome y.

7.j Evaluation of WGCNA based screening on test data

We now compare WGCNA based screening to standard gene screening on the test data set.

```
EvaluationGeneScreening1 = corPredictionSuccess(
          corPrediction = CorPrediction1,
          corTestSet=GS2,
          topNumber=seq(from=20, to=500, length=30) )
par(mfrow=c(2,2))
listcomp = EvaluationGeneScreening1$meancorTestSetOverall
matplot(x = listcomp$topNumber,
       y = listcomp[,-1],
       main="Predicting positive and negative correlations",
       ylab="mean cor, test data",
       xlab="top number of genes in the training data")
listcomp= EvaluationGeneScreening1$meancorTestSetPositive
matplot(x = listcomp$topNumber,
       y = listcomp[,-1],
       main="Predicting positive correlations",
       ylab="mean cor, test data",
       xlab="top number of genes in the training data")
listcomp= EvaluationGeneScreening1$meancorTestSetNegative
matplot(x = listcomp$topNumber,
       y = listcomp[,-1],
```

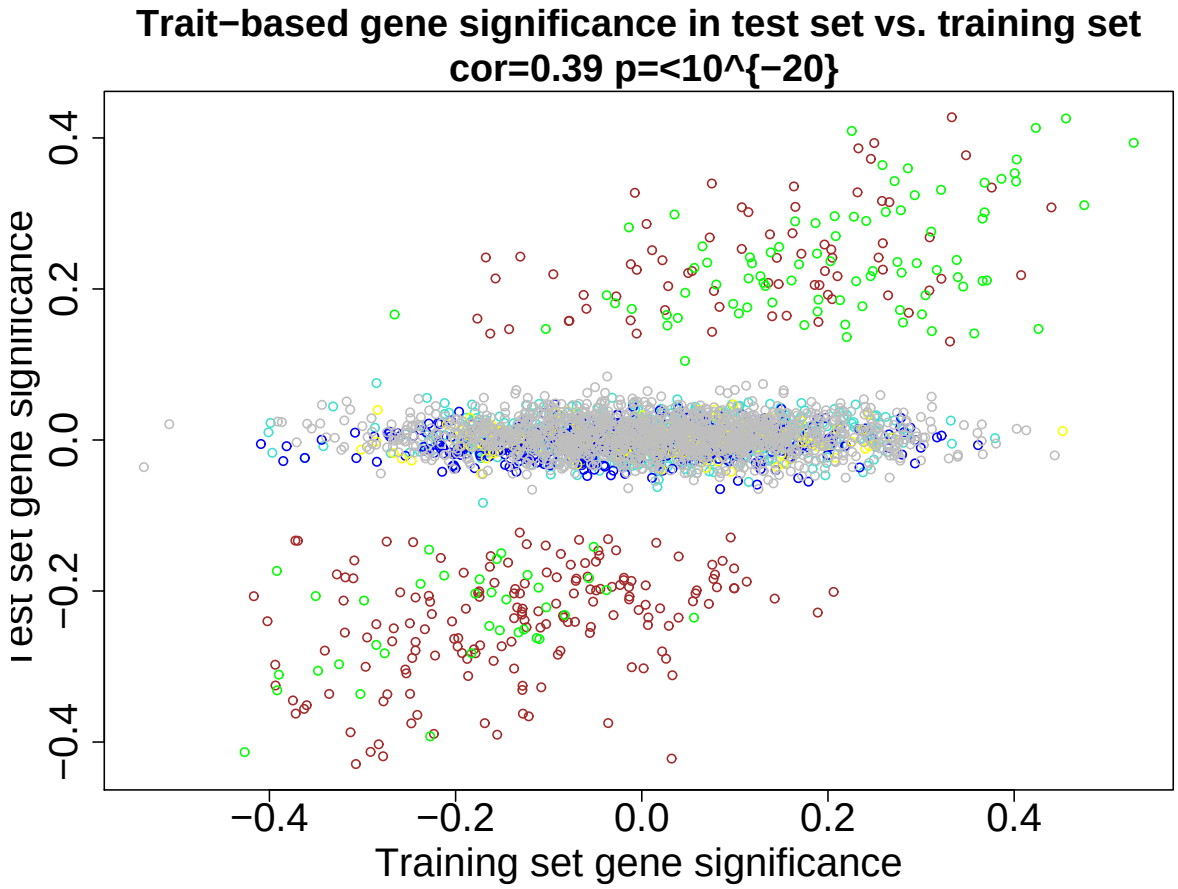

Figure 4: Trait-based gene significance in the test set (y-axis) plotted against the trait-based gene significance in the training set (x-axis). The gene significance of the training set is correlated with that of the test set. Note that the reproducible signal comes from the brown and green module genes. Recall that these modules were simulated to be related to the outcome y.

```
       main = "Predicting negative correlations",
       ylab = "mean cor, test data",
       xlab = "top number of genes in the training data")
```

The result is shown in Fig. 5. The figures show that in this simulated data set network screening is superior when it comes to predicting positive and negative correlations.
The following function allows us to test whether network screening is significantly better than standard screening when it comes to predicting correlations in a test set (`GS2`):

```
relativeCorPredictionSuccess(corPredictionNew = NS1$cor.Weighted,
                          corPredictionStandard = GS1,
                          corTestSet=GS2,
                          topNumber=c(10,20,50,100,200,500) )
```

The output is

```
  topNumber corPredictionNew.kruskalP
1        10                1.725158e-02
2        20                3.437721e-03
3        50                2.042847e-05
```

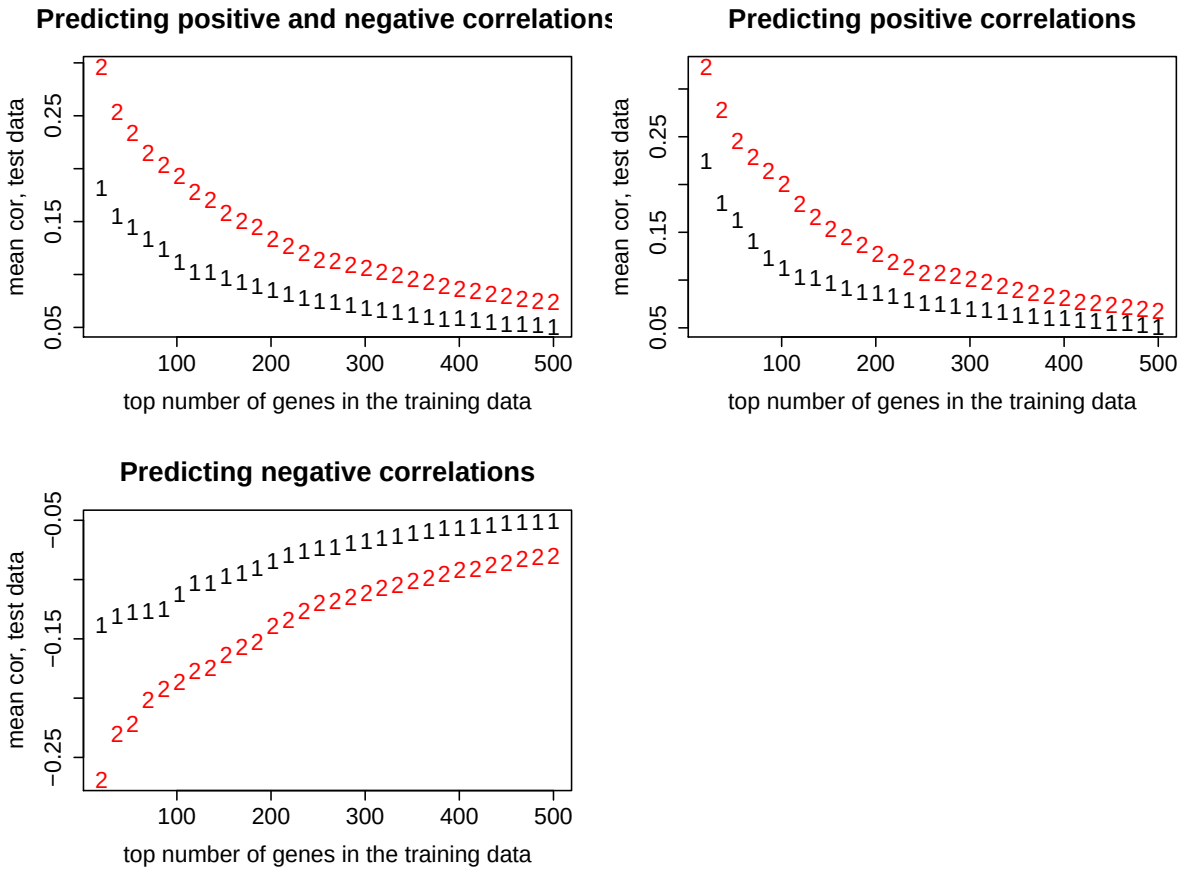

Figure 5: The red and black curve correspond to network screening and standard correlation based screening, respectively. The y-axis reports the mean correlations in the test set. In this simulated data set, network screening is superior when it comes to predicting positive and negative correlations.

4	100	6.861281e-07
5	200	1.451270e-05
6	500	1.350232e-04

7.k Output of the results of network screening analysis

We now illustrate output of the results of network screening into a csv file that can be opened by standard spreadsheet software and other applications.

```
# Create a data frame holding the results of gene screening
GeneResultsNetworkScreening=data.frame(GeneName=row.names(NS1), NS1)
# Write the data frame into a file
write.table(GeneResultsNetworkScreening, file="GeneResultsNetworkScreening.csv",
row.names=F,sep=",")
# Output of eigengene information:
datMEy = data.frame(y, datME)
eigengeneSignificance = cor(datMEy, y);
eigengeneSignificance[1,1] = (1+max(eigengeneSignificance[-1, 1]))/2
eigengeneSignificance.pvalue = corPvalueStudent(eigengeneSignificance, nSamples = length(y))
namesME=names(datMEy)
# Form a summary data frame
```

```
out1=data.frame(t(data.frame(eigengeneSignificance,
eigengeneSignificance.pvalue, namesME, t(datMEy))))
# Set appropriate row names
dimnames(out1)[[1]][1]="EigengeneSignificance"
dimnames(out1)[[1]][2]="EigengeneSignificancePvalue"
dimnames(out1)[[1]][3]="ModuleEigengeneName"
dimnames(out1)[[1]][-c(1:3)]=dimnames(datExpr)[[1]]
# Write the data frame into a file
write.table(out1, file="MEResultsNetworkScreening.csv", row.names=TRUE, col.names = TRUE, sep=",")
# Display the first few rows:
head(out1)
```

The first few rows of the eigengene information table are

```
> head(out1)
                                     y       MEblue      MEbrown      MEgreen
EigengeneSignificance         0.70970248  -0.10142300  -0.27512004   0.41940495
EigengeneSignificancePvalue 7.909071e-09 4.834028e-01 5.315189e-02 2.431272e-03
ModuleEigengeneName                   y       MEblue      MEbrown      MEgreen
Sample1                      1.000000000 -0.020390790  0.069159851 -0.138749932
Sample2                       1.00000000   0.01062292   0.24403639   0.01513903
Sample3                      1.000000000 -0.109366048  0.232823387 -0.159918300
                                  MEgrey  MEturquoise     MEyellow
EigengeneSignificance        -0.17512633   0.05594979   0.09932761
EigengeneSignificancePvalue 2.238187e-01 6.995553e-01 4.925316e-01
ModuleEigengeneName               MEgrey  MEturquoise     MEyellow
Sample1                     -0.066877071 -0.050387559  0.002085968
Sample2                      -0.01593587   0.02762906   0.03332384
Sample3                      0.071309662 -0.121035286 -0.003678822
```

We now write out gene and trait information in format that is used as input in Section 2.

```
# Write out gene information
GeneName=dimnames(datExpr)[[2]]
GeneSummary=data.frame(GeneName, truemodule, SignalGeneIndicator, NS1)
write.table(GeneSummary, file="GeneSummaryTutorial.csv", row.names=F,sep=",")
# here we output the module eigengenes and trait y without eigengene significances
datTraits=data.frame(ArrayName, datMEy)
dimnames(datTraits)[[2]][2:length(namesME)]=paste("Trait",
                                    dimnames(datTraits)[[2]][2:length(namesME)],
                                    sep=".")
write.table(datTraits, file="TraitsTutorial.csv", row.names=F,sep=",")
rm(datTraits)
# here we output the simulated gene expression data
MicroarrayData=data.frame(GeneName, t(datExpr))
names(MicroarrayData)[-1]=ArrayName
write.table(MicroarrayData, file="MicroarrayDataTutorial.csv", row.names=F,sep=",")
rm(MicroarrayData)
```

7.1 Gene Screening based on a gene significance measure

One major advantage of network analysis is that it can carry out gene screening based on any gene significance measure. Such a measure could be defined without reference to an external sample outcome y. In our illustrative example we will pretend that the gene significance `GS1` was determined from external information. The appropriate network screening function to call is `networkScreeningGS`.

```
# Perform network screening
NS1GS=networkScreeningGS(datExpr=datExpr, datME = datME, GS=GS1)
```

```
# Organize its results for easier plotting
GSprediction1=data.frame(GS1,NS1GS$GS.Weighted)
GS.Weighted=NS1GS$GS.Weighted
# Plot a comparison between standard gene significance and network-weighted gene significance
sizeGrWindow(8, 6)
par(mfrow=c(1,1))
verboseScatterplot(GS1, GS.Weighted,
                   main="Weighted gene significance vs. the standard GS\n",
                   col=truemodule)
abline(0,1)
```

The result is shown in Fig. 6. Note that the weighted gene significance measure is amplified for genes that are members in modules with high module significance. In contrast, the gene significance of grey, non-module genes is downweighted.

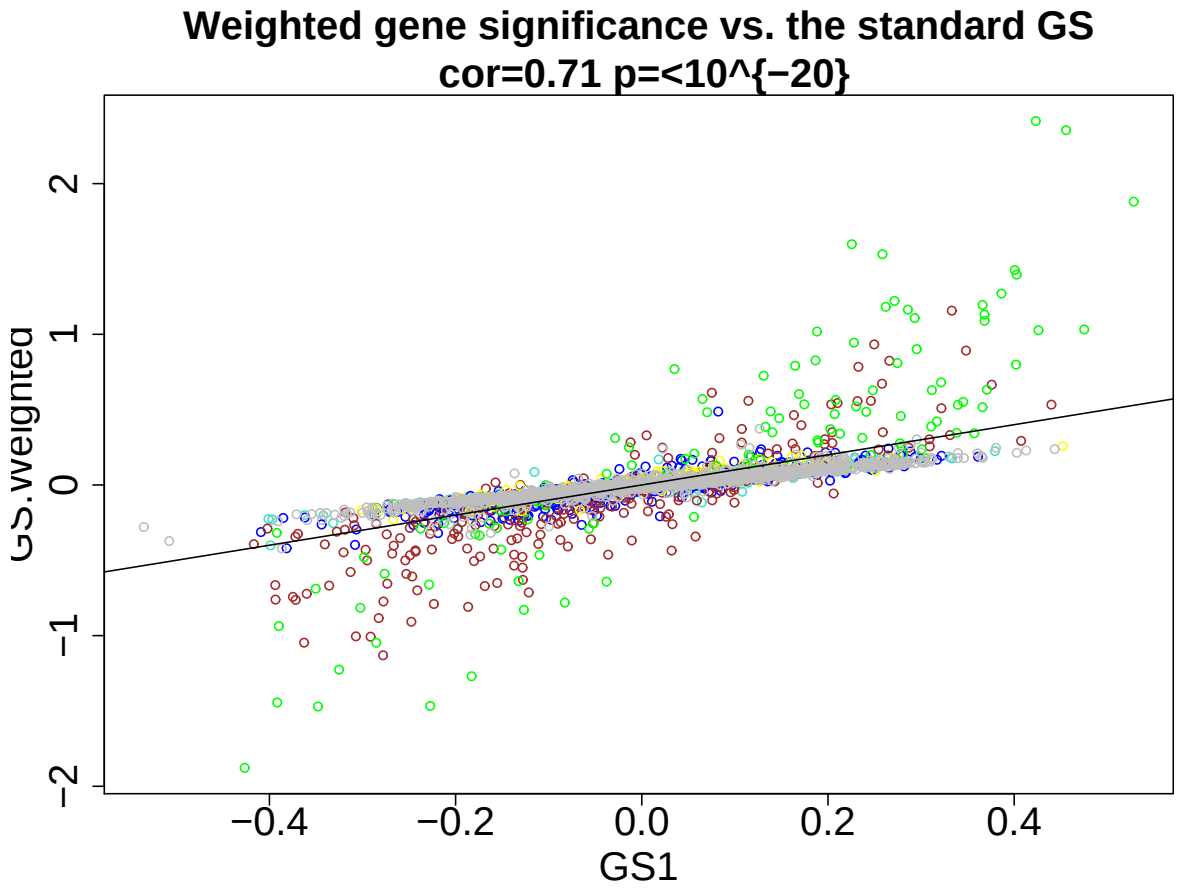

Figure 6: Network-weighted gene significance (y-axis) is shown against the input (standard) gene significance (x-axis). The weighted gene significance measure is amplified for genes that are members in modules with high module significance. In contrast, the gene significance of grey, non-module genes is downweighted.

7.1.1 Comparing network-weighted gene significance to standard gene significance in the test set

We now use the test set simulated earlier to compare the performance of the network-weighted gene significance screening to standard gene significance screening.

```
EvaluationGeneScreeningGS = corPredictionSuccess(corPrediction=GSprediction1, corTestSet=GS2,
```

```
                                    topNumber=seq(from=20, to=500, length=30) )
sizeGrWindow(8, 6)
par(mfrow=c(2,2))
listcomp= EvaluationGeneScreeningGS$meancorTestSetOverall
matplot(x=listcomp$topNumber,
       y=listcomp[,-1],
       main="Predicting positive and negative correlations",
       ylab="mean cor, test data",
       xlab="top number of genes in the training data")
listcomp= EvaluationGeneScreeningGS$meancorTestSetPositive
matplot(x=listcomp$topNumber,
       y=listcomp[,-1],
       main="Predicting positive correlations",
       ylab="mean cor, test data",
       xlab="top number of genes in the training data")
listcomp= EvaluationGeneScreeningGS$meancorTestSetNegative
matplot(x=listcomp$topNumber,
       y=listcomp[,-1],
       main="Predicting negative correlations",
       ylab="mean cor, test data",
       xlab="top number of genes in the training data")
```

The result is shown in Fig. 7. In this simulated data set, network screening is superior when it comes to predicting positive and negative correlations.

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

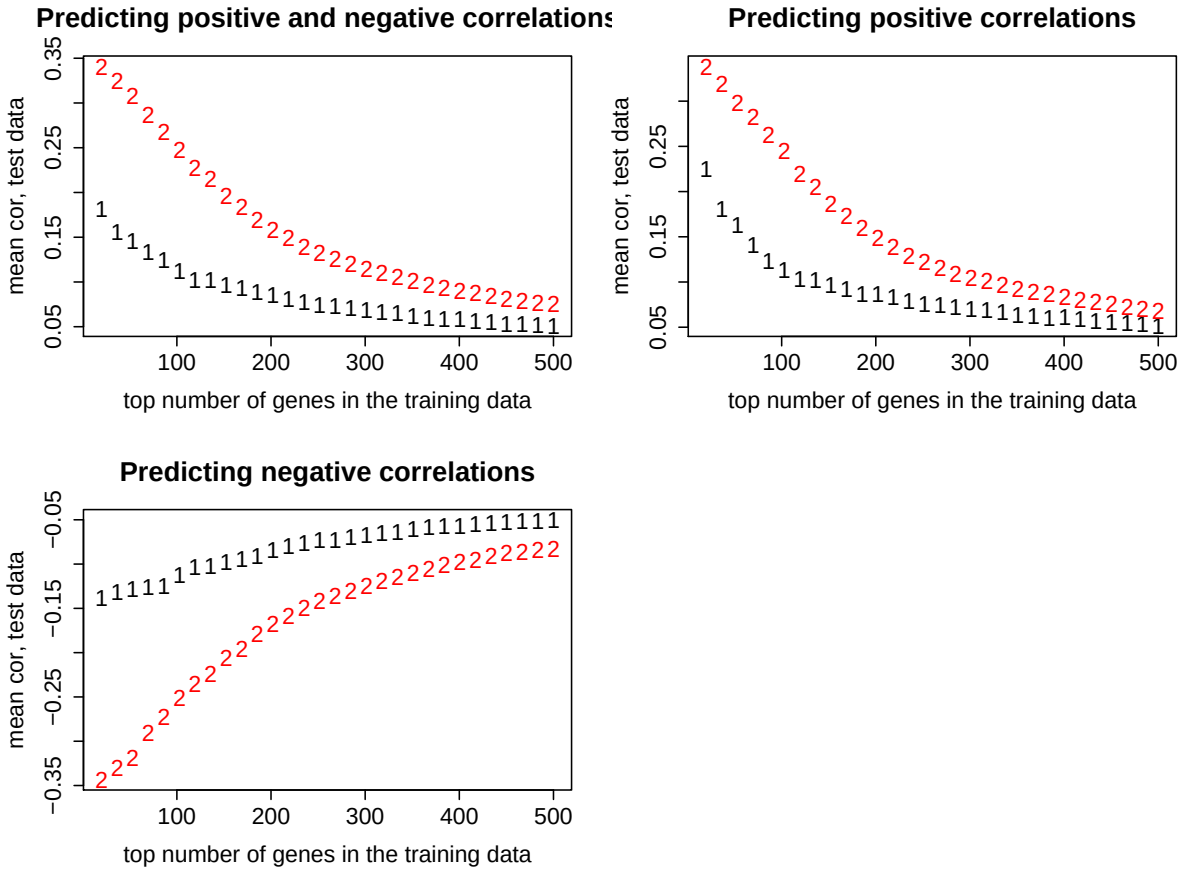

Figure 7: Comparing network-weighted gene significance (red) and standard significance (black). The y-axis reports the mean correlations in the test set as a function of the top number of genes included in training data. In this simulated data set, network screening is superior when it comes to predicting positive and negative correlations.