WGCNA in Proteomic data. All with 0 norm

Contents

1 Data input, cleaning and pre-processing

Load protein abundance data, pre-process them into a format suitable for network analysis, and clean the data by removing obvious outlier samples as well as proteins and samples with excessive numbers of missing entries.

1.a Loading expression data

The expression data is contained in the files:

- 1-9_A1_A2_A3___precol2cm_col75cm_top15_70000_3e6_50_35000_5e4_100_iw4_excl40_2h_200nlmin_1ul_A1vs
  that comes with this tutorial.
- 10-14_C1_C2_C3___precol2cm_col75cm_top15_70000_3e6_50_35000_5e4_100_iw4_excl40_2h_200nlmin_1ul_C1

These files contains several quality parametres. We have used grouped protein abundance to compare samples.

Files not share same proteins, this is one important point to discuss. One aporximation is consider missing protein with 0 abundance. Also it can be considered only the shared proetins.

```
library(readxl)

# Load the WGCNA package
library(WGCNA);
# The following setting is important, do not omit.
options(stringsAsFactors = FALSE)
#Read in the female liver data set
pheno<-read_excel("./nomenclatura.xlsx")
```

```
# Take a quick look at what is in the data set:
dim(pheno)
```

```
## [1] 6 3
```

```
names(pheno)
```

```
## [1] "Nomenclatura Raw Data"         "Nomenclatura mostres de miRNA"
## [3] "Nomenclatura unificada"
```

```
load("RESULTATS/OBJECTES/data_abundance_0")
datExpr0 = as.data.frame(data_abundance_0)
rownames(datExpr0) = datExpr0$Row.names
datExpr0<-datExpr0[,-1]
datExpr0<-t(datExpr0)
```

The expression data set contains 6 samples. Note that each row corresponds to a protein and column to a sample.

1.b Checking data for excessive missing values and identification of outlier microarray samples

We first check for proteins and samples with too many missing values:

```
gsg = goodSamplesGenes(datExpr0, verbose = 3);
```

```
##  Flagging genes and samples with too many missing values...
##   ..step 1
##   ..Excluding 16 genes from the calculation due to too many missing samples or zero variance.
##   ..step 2
```

```
gsg$allOK
```

```
## [1] FALSE
```

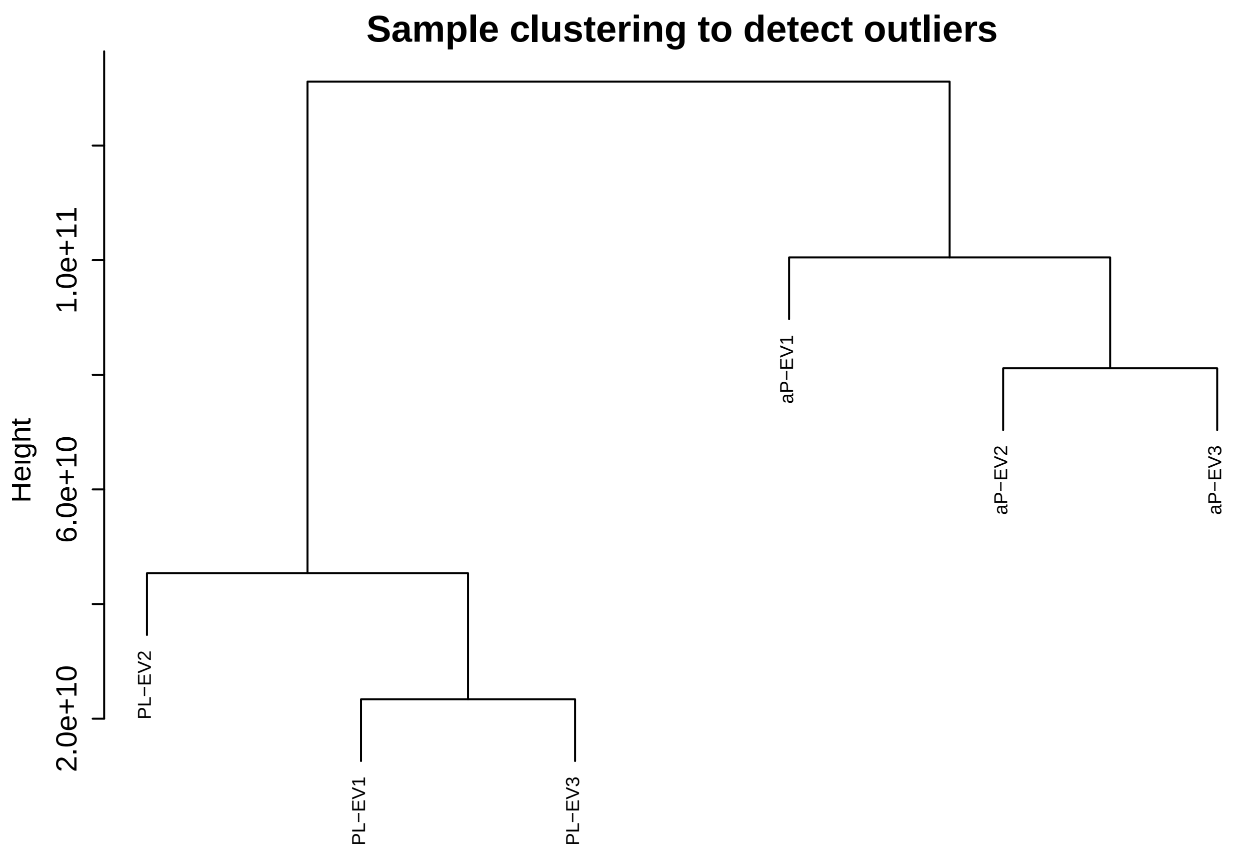
```
if (!gsg$allOK)
{
# Optionally, print the gene and sample names that were removed:
if (sum(!gsg$goodGenes)>0)
printFlush(paste("Removing genes:", paste(names(datExpr0)[!gsg$goodGenes], collapse = ", ")));
if (sum(!gsg$goodSamples)>0)
printFlush(paste("Removing samples:", paste(rownames(datExpr0)[!gsg$goodSamples], collapse = ", ")));
# Remove the offending genes and samples from the data:
datExpr0 = datExpr0[gsg$goodSamples, gsg$goodGenes]
}
```

```
## Removing genes:
```

Next we cluster the samples (in contrast to clustering proteins that will come later) to see if there are any obvious outliers.

```
sampleTree = hclust(dist(datExpr0), method = "average");
# Plot the sample tree: Open a graphic output window of size 12 by 9 inches
# The user should change the dimensions if the window is too large or too small.
# sizeGrWindow(12,9)
#pdf(file = "Plots/sampleClustering.pdf", width = 12, height = 9);
par(cex = 0.6);
par(mar = c(0,4,2,0))
```

```r
plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="", cex.lab = 1.5,
cex.axis = 1.5, cex.main = 2)
```



1.c Loading clinical trait data

```r
pheno<-read_excel("./nomenclatura.xlsx")
dim(pheno)
```

```
## [1] 6 3
```

```r
names(pheno)
```

```
## [1] "Nomenclatura Raw Data"         "Nomenclatura mostres de miRNA"
## [3] "Nomenclatura unificada"
```

```r
pheno$grup<-as.factor(c("NaCl","NaCl","NaCl","PR","PR","PR"))
# remove columns that hold information we do not need.
allTraits = data.frame(pheno)

dim(allTraits)
```

```
## [1] 6 4
```

```r
names(allTraits)
```

```
## [1] "Nomenclatura.Raw.Data"         "Nomenclatura.mostres.de.miRNA"
## [3] "Nomenclatura.unificada"        "grup"
```

```
# Form a data frame analogous to expression data that will hold the clinical traits.
nameSamples = rownames(datExpr0);
traitRows = match(nameSamples, allTraits$Nomenclatura.unificada);
datTraits = allTraits[traitRows,];
rownames(datTraits) = allTraits[traitRows, 1];
collectGarbage()
```

We now have the expression data in the variable datExpr, and the corresponding clinical traits in the variable datTraits.

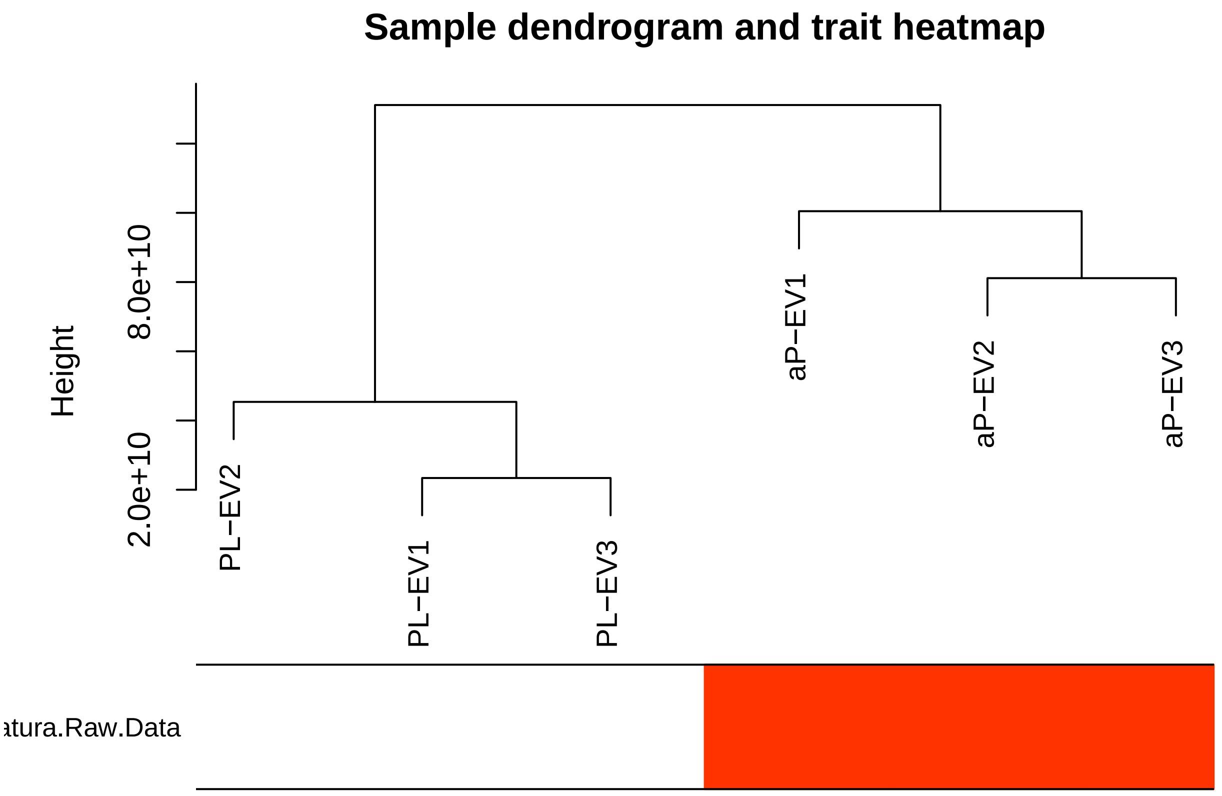
Sample dendrogram

Before we continue with network construction and module detection, we visualize how the clinical traits relate to the sample dendrogram

```
# Re-cluster samples
sampleTree2 = hclust(dist(datExpr0), method = "average")
# Convert traits to a color representation: white means low, red means high, grey means missing entry
datTraits
```

```
##    Nomenclatura.Raw.Data Nomenclatura.mostres.de.miRNA Nomenclatura.unificada
## C1                    C1                          PRP1                 aP-EV1
## C2                    C2                          PRP2                 aP-EV2
## C3                    C3                          PRP3                 aP-EV3
## A1                    A1                         NaCl1                 PL-EV1
## A2                    A2                         NaCl2                 PL-EV2
## A3                    A3                         NaCL3                 PL-EV3
##    grup
## C1   PR
## C2   PR
## C3   PR
## A1 NaCl
## A2 NaCl
## A3 NaCl
```

```
traitColors = numbers2colors(as.numeric(datTraits$grup), signed = FALSE);
# Plot the sample dendrogram and the colors underneath.
plotDendroAndColors(sampleTree2, traitColors,
groupLabels = names(datTraits),
main = "Sample dendrogram and trait heatmap")
```



Batch effect?

Groups are very different (between other things, are not share proteins).

We normalize the data with combat in other report

```
library(sva)
row.names(datExpr0)
```

```
## [1] "aP-EV1" "aP-EV2" "aP-EV3" "PL-EV1" "PL-EV2" "PL-EV3"
```

```
datExpr0<-
ComBat(
  t(datExpr0),batch = c(rep("ap",3),rep("PL",3)),
  mod = NULL,
  par.prior = TRUE,
  prior.plots = FALSE,
  mean.only = FALSE,
  ref.batch = NULL,
  BPPARAM = bpparam("SerialParam")
)
```

```
## Found 726 genes with uniform expression within a single batch (all zeros); these will not be adjuste
```

```
datExpr0<-t(datExpr0)
```

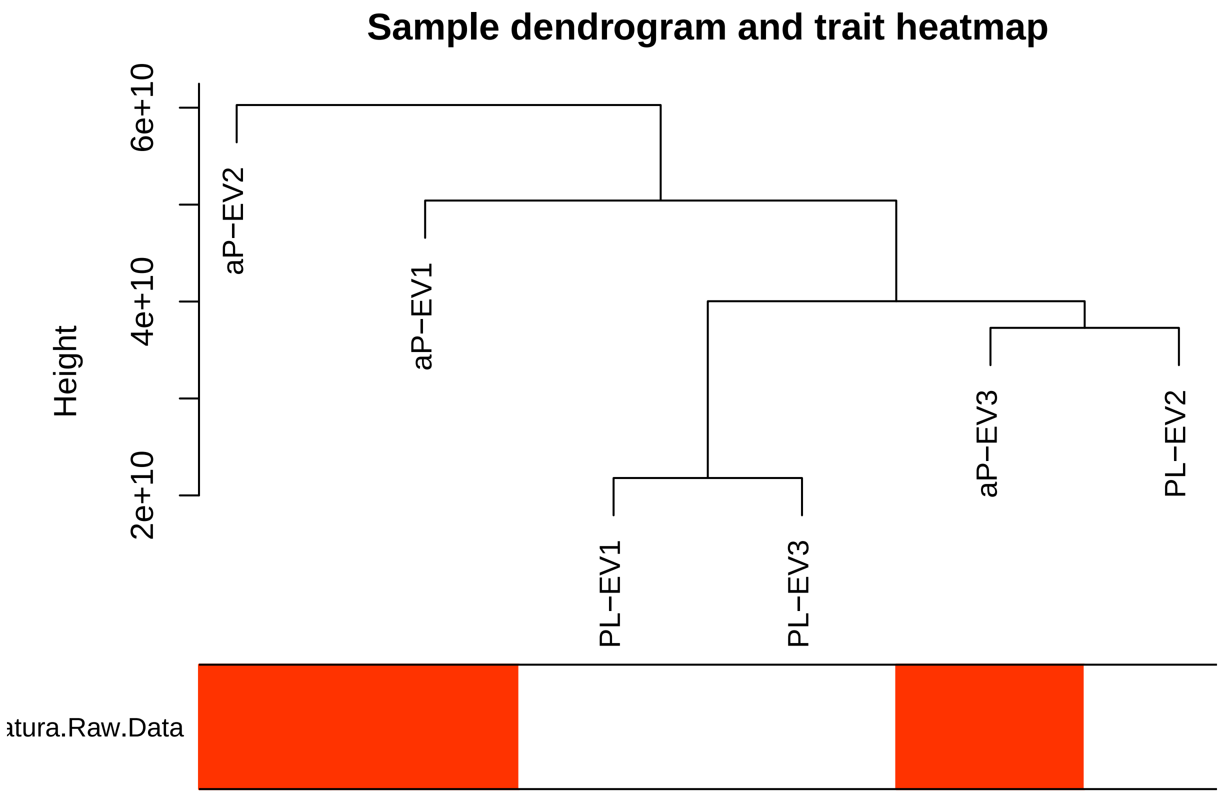
Sample dendrogram

Before we continue with network construction and module detection, we visualize how the clinical traits relate to the sample dendrogram

```
# Re-cluster samples
sampleTree2 = hclust(dist(datExpr0), method = "average")
# Convert traits to a color representation: white means low, red means high, grey means missing entry
datTraits
```

```
##    Nomenclatura.Raw.Data Nomenclatura.mostres.de.miRNA Nomenclatura.unificada
## C1                    C1                          PRP1                 aP-EV1
## C2                    C2                          PRP2                 aP-EV2
## C3                    C3                          PRP3                 aP-EV3
## A1                    A1                         NaCl1                 PL-EV1
## A2                    A2                         NaCl2                 PL-EV2
## A3                    A3                         NaCL3                 PL-EV3
##    grup
## C1   PR
## C2   PR
## C3   PR
## A1 NaCl
## A2 NaCl
## A3 NaCl
```

```
traitColors = numbers2colors(as.numeric(datTraits$grup), signed = FALSE);
# Plot the sample dendrogram and the colors underneath.
plotDendroAndColors(sampleTree2, traitColors,
groupLabels = names(datTraits),
main = "Sample dendrogram and trait heatmap")
```



2.1 Automatic network construction and module detection

2.a.1 Choosing the soft-thresholding power: analysis of network topology

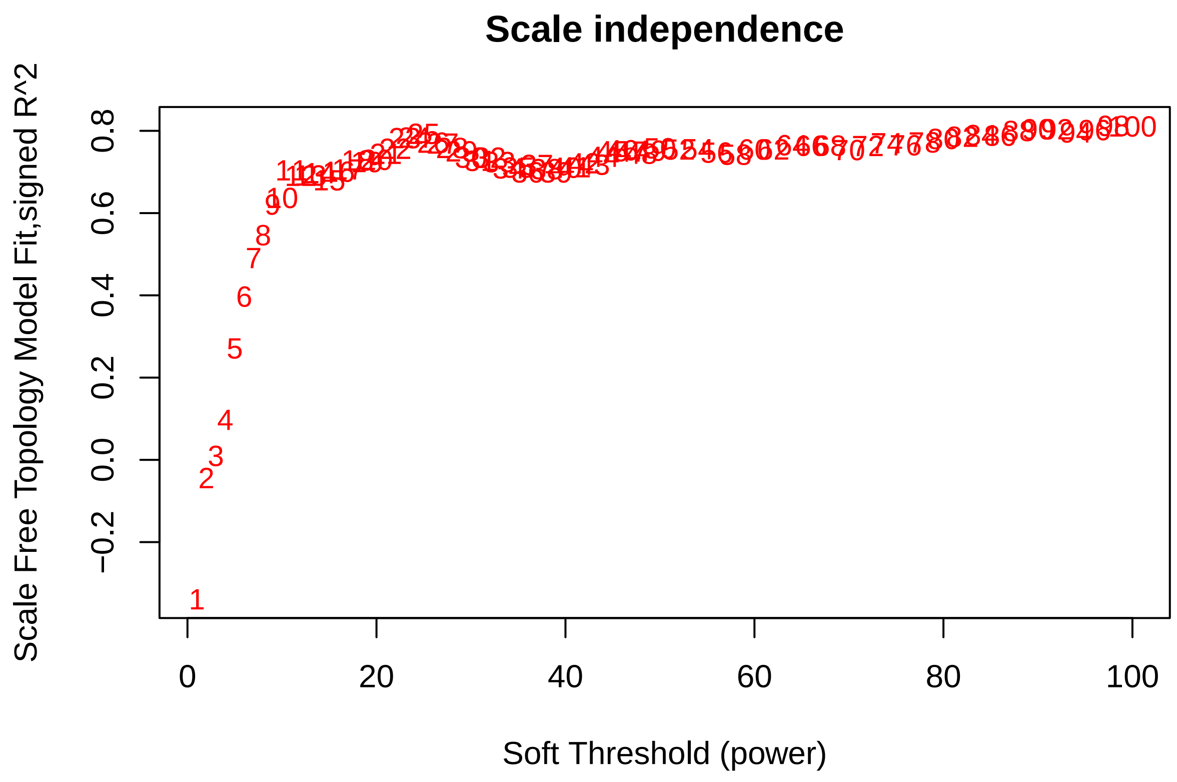
Constructing a weighted gene network entails the choice of the soft thresholding power beta to which co-expression similarity is raised to calculate adjacency [1]. The authors of [1] have proposed to choose the soft thresholding power based on the criterion of approximate scale-free topology. We refer the reader to that work for more details; here we illustrate the use of the function pickSoftThreshold that performs the analysis of network topology and aids the user in choosing a proper soft-thresholding power.

```
# Choose a set of soft-thresholding powers
powers = c(c(1:50), seq(from = 52, to=100, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr0,
                         networkType = "signed hybrid",
                        powerVector = powers, verbose = 5)
```

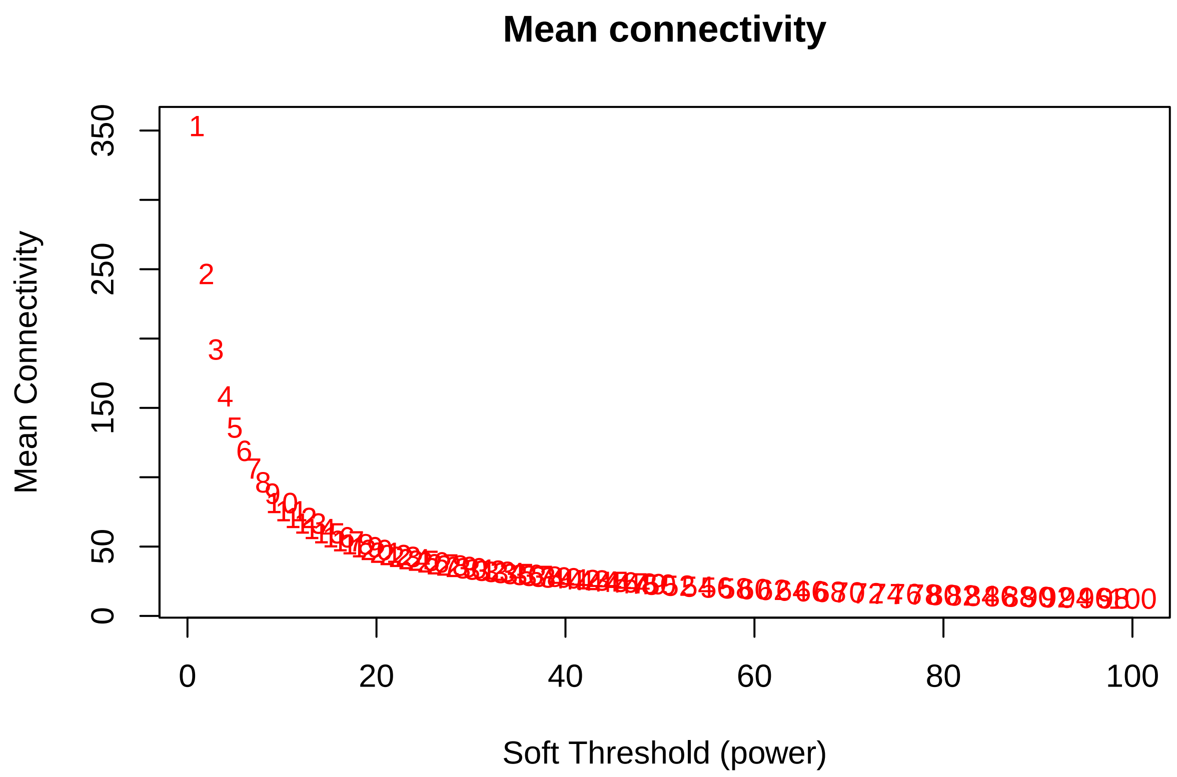
```
## pickSoftThreshold: will use block size 1154.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 1154 of 1154
##    Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
## 1      1  0.33900  0.479         0.2080   353.0    366.00  561.0
## 2      2  0.04390  0.130        -0.1220   247.0    241.00  441.0
## 3      3  0.00942 -0.051        -0.0673   192.0    180.00  370.0
## 4      4  0.09780 -0.152         0.1150   158.0    147.00  325.0
## 5      5  0.27000 -0.237         0.4280   136.0    123.00  292.0
## 6       6  0.39700 -0.285          0.6030   119.0    107.00  266.0
## 7       7  0.49100 -0.335          0.7230   106.0     95.00  245.0
## 8       8  0.54600 -0.366          0.7150    96.2     85.20  229.0
## 9       9  0.62100 -0.389          0.7630    88.1     78.00  215.0
## 10     10  0.63700 -0.403          0.7490    81.3     72.20  203.0
## 11     11  0.70400 -0.422          0.7740    75.6     66.50  192.0
## 12     12  0.69100 -0.436          0.7470    70.8     61.60  183.0
## 13     13  0.69100 -0.450          0.7120    66.5     57.20  175.0
## 14     14  0.69800 -0.457          0.7190    62.8     53.80  167.0
## 15     15  0.68000 -0.467          0.6820    59.5     50.50  160.0
## 16     16  0.70000 -0.478          0.6910    56.6     47.70  154.0
## 17     17  0.70700 -0.489          0.6890    54.0     45.10  148.0
## 18     18  0.72800 -0.492          0.7070    51.6     42.90  143.0
## 19     19  0.72400 -0.496          0.7050    49.4     41.30  138.0
## 20     20  0.72900 -0.501          0.7080    47.5     39.40  133.0
## 21     21  0.74500 -0.503          0.7250    45.7     37.80  129.0
## 22     22  0.75700 -0.505          0.7340    44.0     36.10  125.0
## 23     23  0.78200 -0.507          0.7550    42.5     34.60  122.0
## 24     24  0.78400 -0.509          0.7530    41.0     33.60  119.0
## 25     25  0.79300 -0.513          0.7600    39.7     32.90  116.0
## 26     26  0.77100 -0.516          0.7340    38.5     32.00  113.0
## 27     27  0.77000 -0.518          0.7280    37.3     31.20  110.0
## 28     28  0.75800 -0.520          0.7090    36.2     30.50  108.0
## 29     29  0.75000 -0.524          0.6950    35.2     29.70  105.0
## 30     30  0.73500 -0.529          0.6750    34.3     29.00  103.0
## 31     31  0.72700 -0.539          0.6640    33.4     28.40  101.0
## 32     32  0.73400 -0.542          0.6730    32.5     27.90   98.6
## 33     33  0.72400 -0.546          0.6570    31.7     27.40   96.6
## 34     34  0.70900 -0.548          0.6350    30.9     26.80   94.7
## 35     35  0.71100 -0.550          0.6370    30.2     26.10   93.0
## 36     36  0.69900 -0.554          0.6210    29.5     25.60   91.3
## 37     37  0.71700 -0.557          0.6420    28.9     24.90   89.7
## 38     38  0.70700 -0.558          0.6300    28.2     24.30   88.2
## 39     39  0.69900 -0.561          0.6190    27.6     23.80   86.7
## 40     40  0.71000 -0.564          0.6320    27.1     23.40   85.3
## 41     41  0.71200 -0.569          0.6330    26.5     23.00   83.9
## 42     42  0.72200 -0.568          0.6450    26.0     22.40   82.6
## 43     43  0.71700 -0.569          0.6390    25.5     21.90   81.3
## 44     44  0.73700 -0.571          0.6640    25.0     21.40   80.1
## 45     45  0.75000 -0.571          0.6800    24.5     21.00   78.9
## 46     46  0.75300 -0.572          0.6830    24.1     20.50   77.8
## 47     47  0.75000 -0.575          0.6790    23.7     20.10   76.8
## 48     48  0.74400 -0.580          0.6710    23.2     19.70   75.7
## 49     49  0.75100 -0.586          0.6800    22.8     19.40   74.8
## 50     50  0.75700 -0.592          0.6880    22.5     19.20   73.8
## 51     52  0.75400 -0.601          0.6850    21.7     18.40   72.0
## 52     54  0.75500 -0.605          0.6870    21.0     17.60   70.3
## 53     56  0.74700 -0.615          0.6780    20.4     16.90   68.7
## 54     58  0.74200 -0.620          0.6750    19.8     16.30   67.1
## 55     60  0.75500 -0.627          0.6930    19.2     15.80   65.7
## 56     62  0.75400 -0.633          0.6950    18.7     15.30   64.3
## 57     64  0.76500 -0.639          0.7100    18.2     14.80   63.0
## 58     66  0.76300 -0.643          0.7050    17.7     14.40   61.7
## 59     68  0.76400 -0.644          0.7070    17.3     14.00   60.5
## 60    70  0.75300 -0.650          0.6940    16.9     13.60   59.3
## 61    72  0.76400 -0.652          0.7100    16.5     13.20   58.2
## 62    74  0.77000 -0.654          0.7200    16.1     12.80   57.2
## 63    76  0.76500 -0.659          0.7140    15.7     12.50   56.2
## 64    78  0.77200 -0.665          0.7270    15.4     12.20   55.2
## 65    80  0.78000 -0.668          0.7400    15.0     11.90   54.3
## 66    82  0.78500 -0.668          0.7490    14.7     11.60   53.4
## 67    84  0.78900 -0.670          0.7570    14.4     11.30   52.6
## 68    86  0.78800 -0.673          0.7560    14.1     11.00   51.8
## 69    88  0.79900 -0.675          0.7740    13.8     10.80   51.0
## 70    90  0.80400 -0.679          0.7810    13.6     10.50   50.3
## 71    92  0.80400 -0.682          0.7810    13.3     10.20   49.6
## 72    94  0.79600 -0.685          0.7750    13.1     10.00   48.9
## 73    96  0.80200 -0.690          0.7840    12.8      9.81   48.2
## 74    98  0.81200 -0.694          0.8020    12.6      9.63   47.5
## 75   100  0.81000 -0.699          0.8040    12.4      9.45   46.9
```

```
# Plot the results:
sizeGrWindow(9, 5)
par(mfrow = c(1,2));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding power
```

```
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",
main = paste("Scale independence"));
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
labels=powers,cex=cex1,col="red");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.90,col="red")
```



```
# Mean connectivity as a function of the soft-thresholding power
plot(sft$fitIndices[,1], sft$fitIndices[,5],
xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
main = paste("Mean connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")
```



2.a.2 One-step network construction and module detection

Constructing the gene network and identifying modules is now a simple function call:

```r
pw<-11
net = blockwiseModules(datExpr0, power = pw,
TOMType = "unsigned", minModuleSize = 30,
reassignThreshold = 0, mergeCutHeight = 0.25,
numericLabels = TRUE, pamRespectsDendro = FALSE,
saveTOMs = TRUE,
saveTOMFileBase = "femaleMouseTOM",
verbose = 3)
```

```
##  Calculating module eigengenes block-wise from all genes
##    Flagging genes and samples with too many missing values...
##     ..step 1
##  ..Working on block 1 .
##     TOM calculation: adjacency..
##     ..will not use multithreading.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##    ..saving TOM for block 1 into file femaleMouseTOM-block.1.RData
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.25
##        Calculating new MEs...
```

We have chosen the soft thresholding power 11 , a relatively large minimum module size of 30, and a medium sensitivity (deepSplit=2) to cluster splitting. The parameter mergeCutHeight is the threshold for merging of modules. We have also instructed the function to return numeric, rather than color, labels for modules, and to save the Topological Overlap Matrix. The output of the function may seem somewhat cryptic, but it is easy to use. For example, net$colorscontainsthemoduleassignment,andnet$MEs contains the module eigengenes of the modules.

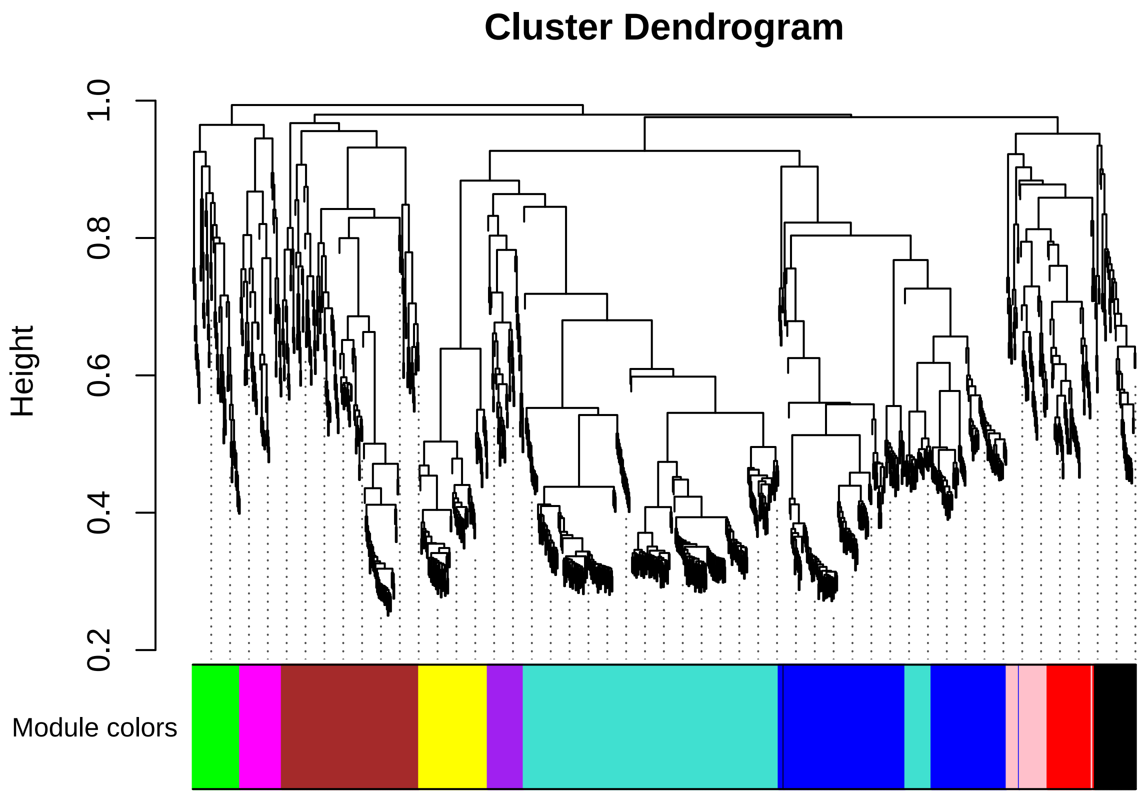
```
table(net$colors)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 344 247 168  84  58  56  51  51  51  44
```

The dendrogram can be displayed together with the color assignment using the following code:

```
# open a graphics window
sizeGrWindow(12, 9)
# Convert labels to colors for plotting
mergedColors = labels2colors(net$colors)
# Plot the dendrogram and the module colors underneath
```

```
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
"Module colors",
dendroLabels = FALSE, hang = 0.03,
addGuide = TRUE, guideHang = 0.05)
```



We note that if the user would like to change some of the tree cut, module membership, and module merging criteria, the package provides the function recutBlockwiseTrees that can apply modified criteria without having to recompute the network and the clustering dendrogram. This may save a sub-stantial amount of time. We now save the module assignment and module eigengene information necessary for subsequent analysis.

```
moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
geneTree = net$dendrograms[[1]];
save(MEs, moduleLabels, moduleColors, geneTree,
file = "FemaleLiver-02-networkConstruction-auto.RData")
```

3 Relating modules to external clinical trait

3a Quantifying module–trait associations

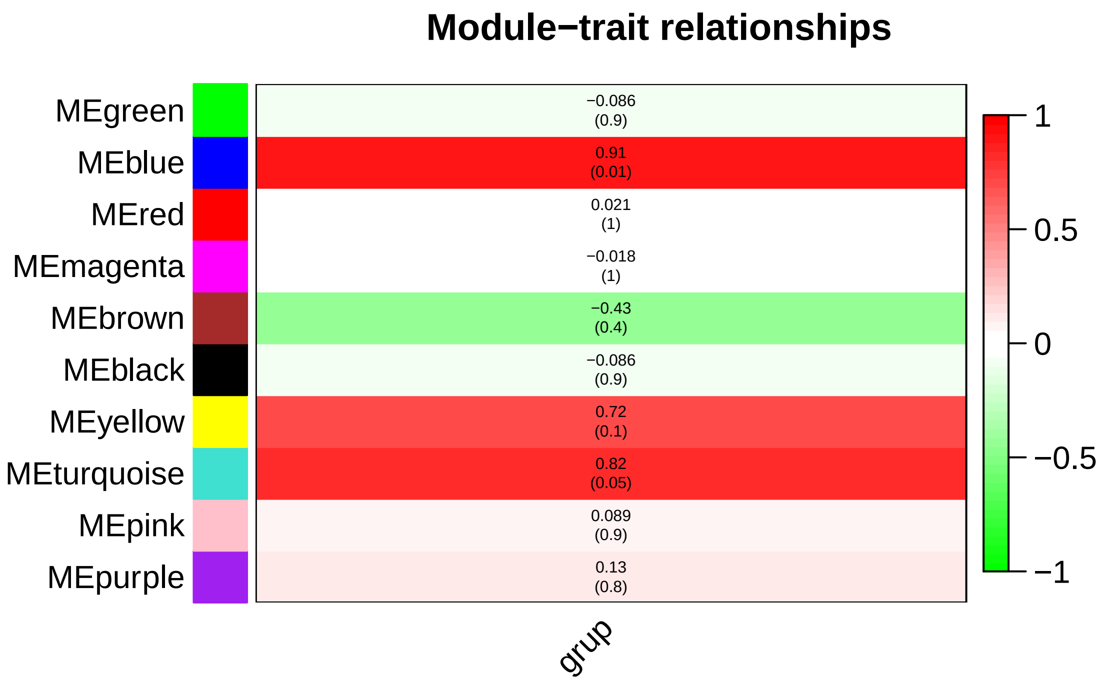
In this analysis we would like to identify modules that are significantly associated with the measured clinical traits. Since we already have a summary profile (eigengene) for each module, we simply correlate eigengenes with external traits and look for the most significant associations

```
# Define numbers of genes and samples
nGenes = ncol(datExpr0)
nSamples = nrow(datExpr0)
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr0, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
```

```
moduleTraitCor = cor(MEs, as.numeric(datTraits$grup), use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples)
```

Since we have a moderately large number of modules and traits, a suitable graphical representation will help in reading the table. We color code each association by the correlation value:

```
# Will display correlations and their p-values
textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
signif(moduleTraitPvalue, 1), ")", sep = "");
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(6, 8.5, 3, 3));
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = moduleTraitCor,
xLabels = names(datTraits)[4],
yLabels = names(MEs),
ySymbols = names(MEs),
colorLabels = FALSE,
colors = greenWhiteRed(50),
textMatrix = textMatrix,
setStdMargins = FALSE,
cex.text = 0.5,
zlim = c(-1,1),
main = paste("Module-trait relationships"))
```



The analysis identifies the 2 significant module–trait associations. We will concentrate on **GRUP** as the trait of interest.

3.b Gene relationship to trait and important modules: Gene Significance and Module Membership

We quantify associations of individual genes with our trait of interest by defining Gene Significance GS as (the absolute value of) the correlation between the gene and the trait. For each module, we also define a quantitative measure of module membership MM as the correlation of the module eigengene and the gene expression profile. This allows us to quantify the similarity of all proteins on the array to every module

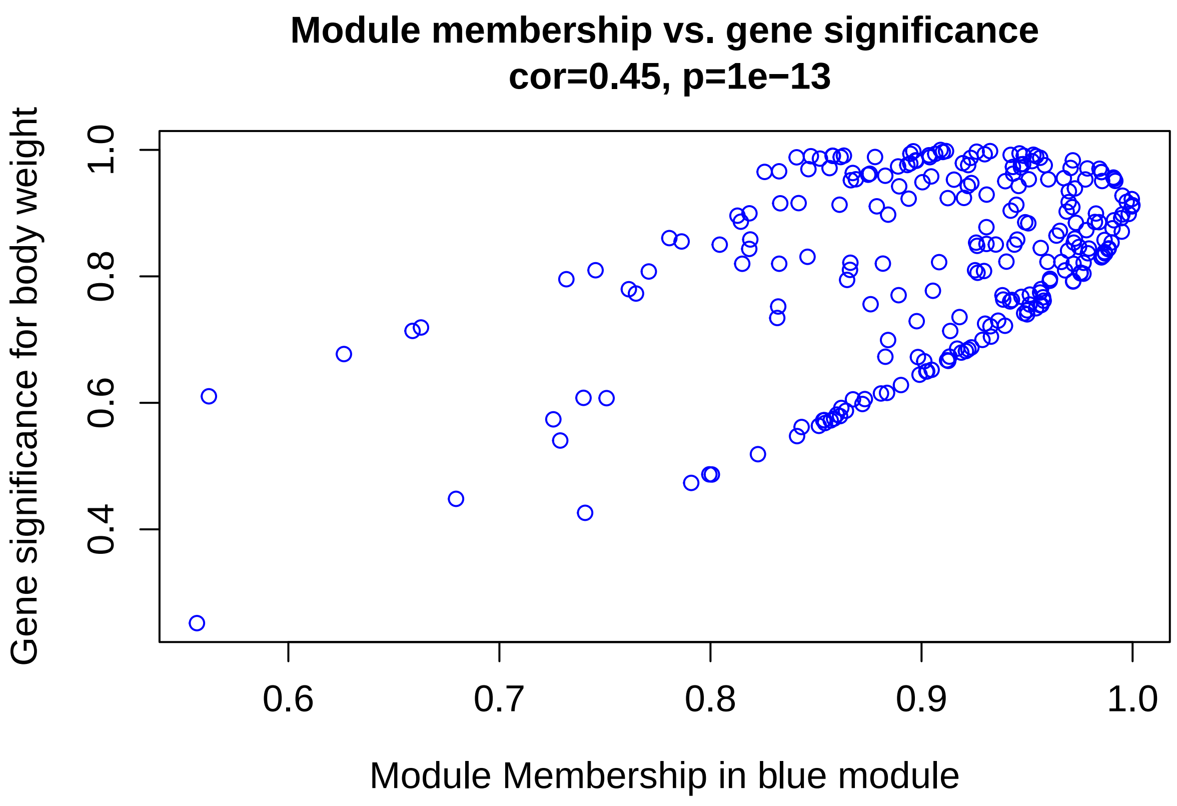
```
# Define variable weight containing the weight column of datTrait
grup = as.data.frame(as.numeric(datTraits$grup));
names(grup) = "grup"
# names (colors) of the modules
modNames = substring(names(MEs), 3)
geneModuleMembership = as.data.frame(cor(datExpr0, MEs, use = "p"));
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership), nSamples));


names(geneModuleMembership) = paste("MM", modNames, sep="")
names(MMPvalue) = paste("p.MM", modNames, sep="")
geneTraitSignificance = as.data.frame(cor(datExpr0, grup, use = "p"))
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance), nSamples))
names(geneTraitSignificance) = paste("GS.", names(grup), sep="")
names(GSPvalue) = paste("p.GS.", names(grup), sep="")
```

##3.c Intramodular analysis: identifying genes with high GS and MM

Using the GS and MM measures, we can identify genes that have a high significance for weight as well as high module membership in interesting modules. As an example, we look at the brown module that has the highest association with weight. We plot a scatterplot of Gene Significance vs. Module Membership in the significant modules

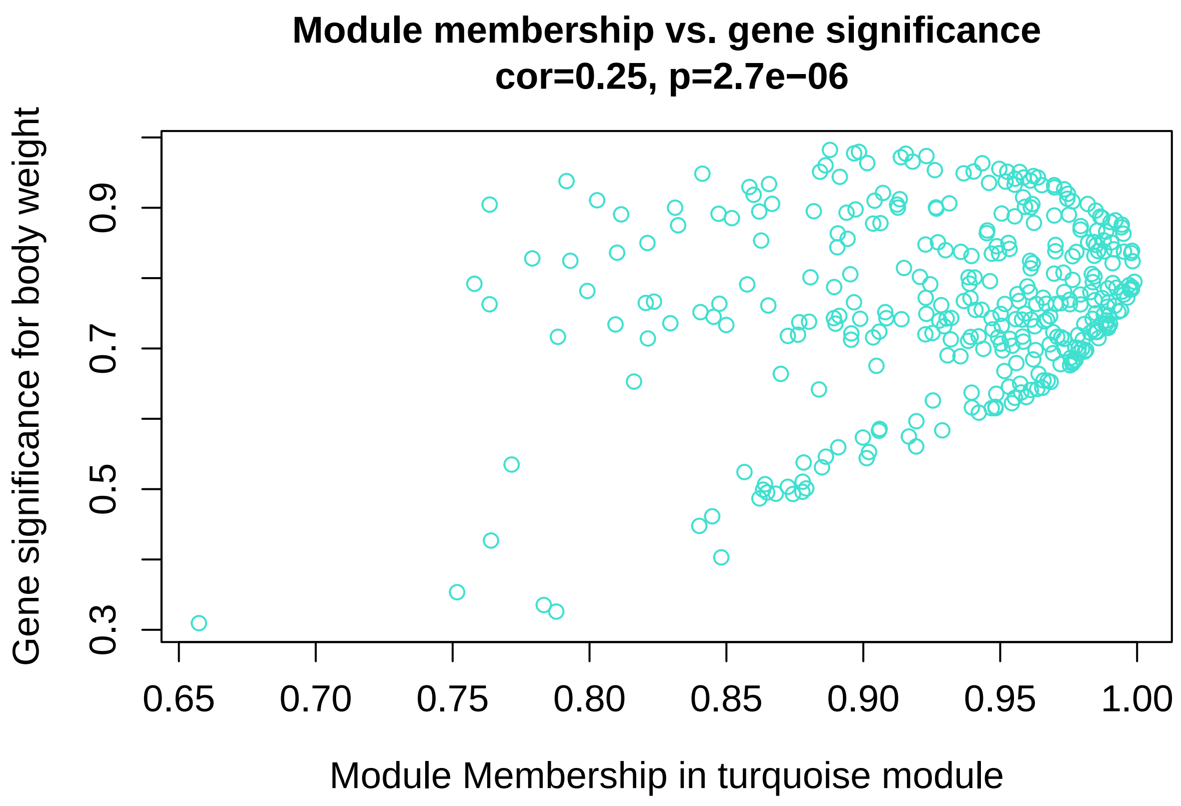
```
moduls_int<-gsub("ME","",rownames(moduleTraitPvalue)[moduleTraitPvalue<=0.05])
module = moduls_int[1]
column = match(module, modNames);
moduleGenes = moduleColors==module;
sizeGrWindow(7, 7)
par(mfrow = c(1,1))
```

```
verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                   abs(geneTraitSignificance[moduleGenes, 1]),
xlab = paste("Module Membership in", module, "module"),
ylab = "Gene significance for body weight",
main = paste("Module membership vs. gene significance\n"),
cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col = module)
```



```
module = moduls_int[2]
column = match(module, modNames);
moduleGenes = moduleColors==module;
sizeGrWindow(7, 7)
par(mfrow = c(1,1))
```

```
verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                   abs(geneTraitSignificance[moduleGenes, 1]),
xlab = paste("Module Membership in", module, "module"),
ylab = "Gene significance for body weight",
main = paste("Module membership vs. gene significance\n"),
cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col = module)
```



GS and MM are correlated, illustrating that proteins highly significantly associated with a trait are often also the most important (central) elements of modules associated with the trait. The reader is encouraged to try this code with other significance trait/module correlation.

3.d Summary output of network analysis results

We have found modules with high association with our trait of interest, and have identified their central players by the Module Membership measure. We now merge this statistical information with protein annotation.

```
# colnames(datExpr0)
# colnames(datExpr0)[moduleColors=="green"]
```

```
library(clusterProfiler)

library(dplyr)
library(DT)
```

```
gene<-colnames(datExpr0)[moduleColors==moduls_int[1]]
gene<-unlist(strsplit(gene,"\r\n"))
kegg_green<-
enrichKEGG(
gene,
organism = "hsa",
keyType = "uniprot",
pvalueCutoff = 0.05,

pAdjustMethod = "BH",
qvalueCutoff = 0.2,
use_internal_data = FALSE
)



gene<-colnames(datExpr0)[moduleColors==moduls_int[2]]
gene<-unlist(strsplit(gene,"\r\n"))
kegg_blue<-
enrichKEGG(
gene,
organism = "hsa",
keyType = "uniprot",
pvalueCutoff = 0.05,

pAdjustMethod = "BH",
qvalueCutoff = 0.2,
use_internal_data = FALSE
)
```

```
library(kableExtra)
knitr::kable(kegg_green@result[1:5,1:5])
```

	category	subcategory	ID	Description
hsa04720	Organismal Systems	Nervous system	hsa04720	Long-term potentiation
hsa04022	Environmental Information Processing	Signal transduction	hsa04022	cGMP-PKG signaling pathw
hsa04261	Organismal Systems	Circulatory system	hsa04261	Adrenergic signaling in card
hsa04713	Organismal Systems	Environmental adaptation	hsa04713	Circadian entrainment
hsa04921	Organismal Systems	Endocrine system	hsa04921	Oxytocin signaling pathway

```
kbl(kegg_green@result[1:5,1:5])
```

	category	subcategory	ID	Description
hsa04720	Organismal Systems	Nervous system	hsa04720	Long-term potentiation
hsa04022	Environmental Information Processing	Signal transduction	hsa04022	cGMP-PKG signaling pathw
hsa04261	Organismal Systems	Circulatory system	hsa04261	Adrenergic signaling in card
hsa04713	Organismal Systems	Environmental adaptation	hsa04713	Circadian entrainment
hsa04921	Organismal Systems	Endocrine system	hsa04921	Oxytocin signaling pathway