```
source("http://bioconductor.org/biocLite.R")
biocLite(c("AnnotationDbi", "impute", "GO.db", "preprocessCore"))
install.packages("WGCNA")

orgCodes = c("Hs", "Mm", "Rn", "Pf", "Sc", "Dm", "Bt", "Ce", "Cf", "Dr", "Gg");
orgExtensions = c(rep(".eg", 4), ".sgd", rep(".eg", 6));
packageNames = paste("org.", orgCodes, orgExtensions, ".db", sep="");

biocLite(c("KEGG.db", "topGO", packageNames, "hgu133a.db", "hgu95av2.db", "annotate",
"hgu133plus2.db", "SNPlocs.Hsapiens.dbSNP.20100427", "minet", "OrderedList"))

#getwd()
workingDir = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data";
setwd(workingdir)
library(WGCNA)
options(stringsAsFactors = FALSE)
femData =
read.csv("/Users/leerkesm/Library/R/3.2/library/WGCNA/data/LiverFemale3600.csv")

#> dim(femData)
#[1] 3600  143


#=====================================================================================

#
#  Code chunk 1
#
#=====================================================================================



# 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 WGCNA 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");
# Take a quick look at what is in the data set:
dim(femData);
names(femData);


#=====================================================================================

#
#  Code chunk 2
#
#=====================================================================================
datExpr0 = as.data.frame(t(femData[, -c(1:8)]));
names(datExpr0) = femData$substanceBXH;
rownames(datExpr0) = names(femData)[-c(1:8)];


#=====================================================================================

#
#  Code chunk 3
#
#=====================================================================================



gsg = goodSamplesGenes(datExpr0, verbose = 3);
gsg$allOK


#=====================================================================================

#
#  Code chunk 4
#
#=====================================================================================



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]
}


#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



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

#=====================================================================================

#
#  Code chunk 6
#
#=====================================================================================



# Plot a line to show the cut
abline(h = 15, col = "red");
# Determine cluster under the line
clust = cutreeStatic(sampleTree, cutHeight = 15, minSize = 10)
table(clust)
# clust 1 contains the samples we want to keep.
keepSamples = (clust==1)
datExpr = datExpr0[keepSamples, ]
nGenes = ncol(datExpr)
nSamples = nrow(datExpr)


#=====================================================================================

#
#  Code chunk 7
#
#=====================================================================================



traitData =
read.csv("/Users/leerkesm/Library/R/3.2/library/WGCNA/data/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) ];
dim(allTraits)
names(allTraits)

# Form a data frame analogous to expression data that will hold the clinical traits.

femaleSamples = rownames(datExpr);
traitRows = match(femaleSamples, allTraits$Mice);
datTraits = allTraits[traitRows, -1];
rownames(datTraits) = allTraits[traitRows, 1];

collectGarbage();


#=====================================================================================

#
#  Code chunk 8
#
#=====================================================================================
# Re-cluster samples
sampleTree2 = hclust(dist(datExpr), method = "average")
# Convert traits to a color representation: white means low, red means high, grey
means missing entry
traitColors = numbers2colors(datTraits, signed = FALSE);
# Plot the sample dendrogram and the colors underneath.
plotDendroAndColors(sampleTree2, traitColors,
                    groupLabels = names(datTraits),
                    main = "Sample dendrogram and trait heatmap")


#=====================================================================================

#
#  Code chunk 9
#
#=====================================================================================



save(datExpr, datTraits, file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/test_FemaleLiver-01-
dataInput.RData")



#deel2

#=====================================================================================

#
#  Code chunk 1
#
#=====================================================================================



# 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 WGCNA package
#library(WGCNA)
# The following setting is important, do not omit.
#options(stringsAsFactors = FALSE);
# Allow multi-threading within WGCNA. This helps speed up certain calculations.
# At present this call is necessary for the code to work.
# Any error here may be ignored but you may want to update WGCNA if you see one.
# Caution: skip this line if you run RStudio or other third-party R environments.
# See note above.
#enableWGCNAThreads()
# Load the data saved in the first part
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
01-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames


#=====================================================================================

#
#  Code chunk 2
#
#=====================================================================================



# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
# 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")


#=====================================================================================

#
#  Code chunk 3
#
#=====================================================================================



net = blockwiseModules(datExpr, power = 6,
                       TOMType = "unsigned", minModuleSize = 30,
                       reassignThreshold = 0, mergeCutHeight = 0.25,
                       numericLabels = TRUE, pamRespectsDendro = FALSE,
                       saveTOMs = TRUE,
                       saveTOMFileBase = "femaleMouseTOM",
                       verbose = 3)


#=====================================================================================

#
#  Code chunk 4
```

```
#
#=====================================================================================



# 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)


#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
geneTree = net$dendrograms[[1]];
save(MEs, moduleLabels, moduleColors, geneTree,
     file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/test_FemaleLiver-02-
networkConstruction-auto.RData")



#deel 2b


#=====================================================================================

#
#  Code chunk 1
#
#=====================================================================================



# 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 WGCNA package
#library(WGCNA)
# The following setting is important, do not omit.
#options(stringsAsFactors = FALSE);
# Allow multi-threading within WGCNA. At present this call is necessary.
# Any error here may be ignored but you may want to update WGCNA if you see one.
# Caution: skip this line if you run RStudio or other third-party R environments.
# See note above.
#enableWGCNAThreads()
# Load the data saved in the first part
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
01-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames


#=====================================================================================

#
#  Code chunk 2
#
#=====================================================================================



# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
# 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")

#=====================================================================================

#
#  Code chunk 3
#
#=====================================================================================



softPower = 6;
adjacency = adjacency(datExpr, power = softPower);


#=====================================================================================

#
#  Code chunk 4
```

```
#
#=====================================================================================


# Turn adjacency into topological overlap
TOM = TOMsimilarity(adjacency);
dissTOM = 1-TOM

#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



# Call the hierarchical clustering function
geneTree = hclust(as.dist(dissTOM), method = "average");
# Plot the resulting clustering tree (dendrogram)
sizeGrWindow(12,9)
plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based dissimilarity",
     labels = FALSE, hang = 0.04);


#=====================================================================================

#
#  Code chunk 6
#
#=====================================================================================



# We like large modules, so we set the minimum module size relatively high:
minModuleSize = 30;
# Module identification using dynamic tree cut:
dynamicMods = cutreeDynamic(dendro = geneTree, distM = dissTOM,
                deepSplit = 2, pamRespectsDendro = FALSE,
                minClusterSize = minModuleSize);
table(dynamicMods)


#=====================================================================================

#
#  Code chunk 7
#
#=====================================================================================



# Convert numeric lables into colors
dynamicColors = labels2colors(dynamicMods)
table(dynamicColors)
# Plot the dendrogram and colors underneath
sizeGrWindow(8,6)
plotDendroAndColors(geneTree, dynamicColors, "Dynamic Tree Cut",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05,
                    main = "Gene dendrogram and module colors")


#=====================================================================================
#
#  Code chunk 8
#
#=====================================================================================



# Calculate eigengenes
MEList = moduleEigengenes(datExpr, colors = dynamicColors)
MEs = MEList$eigengenes
# Calculate dissimilarity of module eigengenes
MEDiss = 1-cor(MEs);
# Cluster module eigengenes
METree = hclust(as.dist(MEDiss), method = "average");
# Plot the result
sizeGrWindow(7, 6)
plot(METree, main = "Clustering of module eigengenes",
     xlab = "", sub = "")


#=====================================================================================
#
#  Code chunk 9
#
#=====================================================================================



MEDissThres = 0.25
# Plot the cut line into the dendrogram
abline(h=MEDissThres, col = "red")
# Call an automatic merging function
merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = MEDissThres, verbose =
3)
# The merged module colors
mergedColors = merge$colors;
# Eigengenes of the new merged modules:
mergedMEs = merge$newMEs;


#=====================================================================================
#
#  Code chunk 10
#
#=====================================================================================



sizeGrWindow(12, 9)
#pdf(file = "Plots/geneDendro-3.pdf", wi = 9, he = 6)
plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),
                    c("Dynamic Tree Cut", "Merged dynamic"),
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
#dev.off()


#=====================================================================================
#
#  Code chunk 11
#
#=====================================================================================



# Rename to moduleColors
moduleColors = mergedColors
# Construct numerical labels corresponding to the colors
colorOrder = c("grey", standardColors(50));
moduleLabels = match(moduleColors, colorOrder)-1;
MEs = mergedMEs;
# Save module colors and labels for use in subsequent parts
save(MEs, moduleLabels, moduleColors, geneTree, file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/test_FemaleLiver-02-
networkConstruction-stepByStep.RData")




deel 2c

#=====================================================================================
#
#  Code chunk 1
#
#=====================================================================================



# 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 WGCNA package
#library(WGCNA)
# The following setting is important, do not omit.
#options(stringsAsFactors = FALSE);
# Allow multi-threading within WGCNA. This helps speed up certain calculations.
# At present this call is necessary.
# Any error here may be ignored but you may want to update WGCNA if you see one.
# Caution: skip this line if you run RStudio or other third-party R environments.
# See note above.
#enableWGCNAThreads()
# Load the data saved in the first part
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
01-dataInput.RData");
```

```
#The variable lnames contains the names of loaded variables.
lnames


#=====================================================================================

#
#  Code chunk 2
#
#=====================================================================================



# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
# 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")


#=====================================================================================

#
#  Code chunk 3
#
#=====================================================================================



bwnet = blockwiseModules(datExpr, maxBlockSize = 2000,
                       power = 6, TOMType = "unsigned", minModuleSize = 30,
                       reassignThreshold = 0, mergeCutHeight = 0.25,
                       numericLabels = TRUE,
                       saveTOMs = TRUE,
                       saveTOMFileBase =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/femaleMouseTOM-blockwise",
                       verbose = 3)


#=====================================================================================

#
#  Code chunk 4
```

```
#
#=====================================================================================



# Load the results of single-block analysis
load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-02-
networkConstruction-auto.RData");
# Relabel blockwise modules
bwLabels = matchLabels(bwnet$colors, moduleLabels);
# Convert labels to colors for plotting
bwModuleColors = labels2colors(bwLabels)


#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



# open a graphics window
sizeGrWindow(6,6)
# Plot the dendrogram and the module colors underneath for block 1
plotDendroAndColors(bwnet$dendrograms[[1]], bwModuleColors[bwnet$blockGenes[[1]]],
                    "Module colors", main = "Gene dendrogram and module colors in
block 1",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
# Plot the dendrogram and the module colors underneath for block 2
plotDendroAndColors(bwnet$dendrograms[[2]], bwModuleColors[bwnet$blockGenes[[2]]],
                    "Module colors", main = "Gene dendrogram and module colors in
block 2",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)


#=====================================================================================

#
#  Code chunk 6
#
#=====================================================================================



sizeGrWindow(12,9)
plotDendroAndColors(geneTree,
                    cbind(moduleColors, bwModuleColors),
                    c("Single block", "2 blocks"),
                    main = "Single block gene dendrogram and module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)


#=====================================================================================

#
```

```
#  Code chunk 7
#
#=====================================================================================


singleBlockMEs = moduleEigengenes(datExpr, moduleColors)$eigengenes;
blockwiseMEs = moduleEigengenes(datExpr, bwModuleColors)$eigengenes;


#=====================================================================================
#
#  Code chunk 8
#
#=====================================================================================


single2blockwise = match(names(singleBlockMEs), names(blockwiseMEs))
signif(diag(cor(blockwiseMEs[, single2blockwise], singleBlockMEs)), 3)


#deel 3

#=====================================================================================
#
#  Code chunk 1
#
#=====================================================================================


# 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 WGCNA package
#library(WGCNA)
# The following setting is important, do not omit.
#options(stringsAsFactors = FALSE);
# Load the expression and trait data saved in the first part
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
01-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames
# Load network data saved in the second part.
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
02-networkConstruction-auto.RData");
lnames


#=====================================================================================
#
#  Code chunk 2
#
#=====================================================================================



# Define numbers of genes and samples
nGenes = ncol(datExpr);
nSamples = nrow(datExpr);
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, datTraits, use = "p");
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);


#=====================================================================================
#
#  Code chunk 3
#
#=====================================================================================



sizeGrWindow(10,6)
# 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),
               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"))


#=====================================================================================
#
#  Code chunk 4
#
#=====================================================================================



# Define variable weight containing the weight column of datTrait
weight = as.data.frame(datTraits$weight_g);
names(weight) = "weight"
# names (colors) of the modules
modNames = substring(names(MEs), 3)
```

```
geneModuleMembership = as.data.frame(cor(datExpr, 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(datExpr, weight, use = "p"));
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance),
nSamples));

names(geneTraitSignificance) = paste("GS.", names(weight), sep="");
names(GSPvalue) = paste("p.GS.", names(weight), sep="");


#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



module = "brown"
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)


#=====================================================================================

#
#  Code chunk 6
#
#=====================================================================================



names(datExpr)


#=====================================================================================

#
#  Code chunk 7
#
#=====================================================================================
names(datExpr)[moduleColors=="brown"]


#=====================================================================================

#
#  Code chunk 8
#
#=====================================================================================



annot = read.csv(file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/GeneAnnotation.csv");
dim(annot)
names(annot)
probes = names(datExpr)
probes2annot = match(probes, annot$substanceBXH)
# The following is the number or probes without annotation:
sum(is.na(probes2annot))
# Should return 0.


#=====================================================================================

#
#  Code chunk 9
#
#=====================================================================================



# Create the starting data frame
geneInfo0 = data.frame(substanceBXH = probes,
                      geneSymbol = annot$gene_symbol[probes2annot],
                      LocusLinkID = annot$LocusLinkID[probes2annot],
                      moduleColor = moduleColors,
                      geneTraitSignificance,
                      GSPvalue)
# Order modules by their significance for weight
modOrder = order(-abs(cor(MEs, weight, use = "p")));
# Add module membership information in the chosen order
for (mod in 1:ncol(geneModuleMembership))
{
  oldNames = names(geneInfo0)
  geneInfo0 = data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                         MMPvalue[, modOrder[mod]]);
  names(geneInfo0) = c(oldNames, paste("MM.", modNames[modOrder[mod]], sep=""),
                       paste("p.MM.", modNames[modOrder[mod]], sep=""))
}
# Order the genes in the geneInfo variable first by module color, then by
geneTraitSignificance
geneOrder = order(geneInfo0$moduleColor, -abs(geneInfo0$GS.weight));
geneInfo = geneInfo0[geneOrder, ]


#=====================================================================================

#
#  Code chunk 10
```

```
#
#=====================================================================================



write.csv(geneInfo, file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/geneInfo.csv")



#deel 4

#=====================================================================================

#
#  Code chunk 1
#
#=====================================================================================



# 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 WGCNA package
#library(WGCNA)
# The following setting is important, do not omit.
#options(stringsAsFactors = FALSE);
# Load the expression and trait data saved in the first part
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
01-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames
# Load network data saved in the second part.
lnames = load(file = "/Users/leerkesm/Library/R/3.2/library/WGCNA/data/FemaleLiver-
02-networkConstruction-auto.RData");
lnames


#=====================================================================================

#
#  Code chunk 2
#
#=====================================================================================



# Read in the probe annotation
annot = read.csv(file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/data/GeneAnnotation.csv");
# Match probes in the data set to the probe IDs in the annotation file
probes = names(datExpr)
probes2annot = match(probes, annot$substanceBXH)
# Get the corresponding Locuis Link IDs
```

```
allLLIDs = annot$LocusLinkID[probes2annot];
# $ Choose interesting modules
intModules = c("brown", "red", "salmon")
for (module in intModules)
{
  # Select module probes
  modGenes = (moduleColors==module)
  # Get their entrez ID codes
  modLLIDs = allLLIDs[modGenes];
  # Write them into a file
  fileName = paste("/Users/leerkesm/Library/R/3.2/library/WGCNA/data/LocusLinkIDs-",
module, ".txt", sep="");
  write.table(as.data.frame(modLLIDs), file = fileName,
              row.names = FALSE, col.names = FALSE)
}
# As background in the enrichment analysis, we will use all probes in the analysis.
#hiermeeverdergaan
fileName = paste("/Users/leerkesm/Library/R/3.2/library/WGCNA/data/LocusLinkIDs-
all.txt", sep="");
write.table(as.data.frame(allLLIDs), file = fileName,
            row.names = FALSE, col.names = FALSE)


#=====================================================================================

#
#  Code chunk 3
#
#=====================================================================================



GOenr = GOenrichmentAnalysis(moduleColors, allLLIDs, organism = "mouse", nBestP =
10);


#=====================================================================================

#
#  Code chunk 4
#
#=====================================================================================



tab = GOenr$bestPTerms[[4]]$enrichment


#=====================================================================================

#
#  Code chunk 5
#
#=====================================================================================



names(tab)
```

```
#=====================================================================================
#
#  Code chunk 6
#
#=====================================================================================



write.table(tab, file =
"/Users/leerkesm/Library/R/3.2/library/WGCNA/dataGOEnrichmentTable.csv", sep = ",",
quote = TRUE, row.names = FALSE)


#=====================================================================================
#
#  Code chunk 7
#
#=====================================================================================



keepCols = c(1, 2, 5, 6, 7, 12, 13);
screenTab = tab[, keepCols];
# Round the numeric columns to 2 decimal places:
numCols = c(3, 4);
screenTab[, numCols] = signif(apply(screenTab[, numCols], 2, as.numeric), 2)
# Truncate the the term name to at most 40 characters
screenTab[, 7] = substring(screenTab[, 7], 1, 40)
# Shorten the column names:
colnames(screenTab) = c("module", "size", "p-val", "Bonf", "nInTerm", "ont", "term
name");
rownames(screenTab) = NULL;
# Set the width of R's output. The reader should play with this number to obtain
satisfactory output.
options(width=95)
# Finally, display the enrichment table:
screenTab
```