```
#########################
######### INPUT #########
#########################

# load the package
library(WGCNA)
setwd("/users/nanette/")

# load the data
options(stringsAsFactors = FALSE);
noflagData = read.csv("log10_maize_expr_data.csv", row.names=1);
dim(noflagData);
names(noflagData);
datExpr0 = as.data.frame(t(noflagData));
dim(datExpr0)

# check for genes and samples with too many missing values:
gsg = goodSamplesGenes(datExpr0, verbose = 3);
gsg$allOK
#remove the offending genes and samples from the data
if (!gsg$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]
}
dim(datExpr0)

# cluster the samples to see if there are any obvious outliers
sampleTree = flashClust(dist(datExpr0), method = "average");

# draw the sample dendogram
pdf(file = "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)
dev.off()

# remove the sample outliers
clust = cutreeStatic(sampleTree, cutHeight = 1000000, minSize = 10)
table(clust)
keepSamples = (clust==1)
datExpr = datExpr0[keepSamples, ]
nGenes = ncol(datExpr)
nSamples = nrow(datExpr)

#####################################################

# load the trait data
traitData = read.csv("GLS_pheno.csv");
dim(traitData)
names(traitData)
allTraits = traitData[,];
```

```
# Form a data frame analogous to expression data that will hold the trait
data.
RILSamples = rownames(datExpr);
traitRows = match(RILSamples, allTraits$RIL);
datTraits = allTraits[traitRows,];
rownames(datTraits) = allTraits[traitRows, 1];
datTraits = datTraits[,-1]
collectGarbage();

# re-cluster samples
sampleTree2 = flashClust(dist(datExpr), method = "average")

# visualize how the traits relate to the sample dendrogram
traitColors = numbers2colors(datTraits, signed = FALSE);
pdf(file = "sampleClustering_traitHeatmap.pdf", width = 18, height = 10);
plotDendroAndColors(sampleTree2, traitColors,
                  groupLabels = "GLS score",
                  main = "Sample dendrogram and trait heatmap")
dev.off()

# save the relevant expression and trait data
save(datExpr, datTraits, file = "Maize_dataInput.RData")

############################################
######### NETWORK CONSTRUCTION #########
############################################

enableWGCNAThreads()
options(stringsAsFactors = FALSE);

# 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
maize_sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
maize_sft
# maize_sft$powerEstimate = 12

# Plot the results:
#sizeGrWindow(9, 5)
pdf(file = "Soft_threshold.pdf", width = 18, height = 10);
par(mfrow = c(1,2));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding
power
plot(maize_sft$fitIndices[,1], -sign(maize_sft$fitIndices[,3])*
maize_sft$fitIndices[,2],
     xlab="Soft Threshold (power)",ylab="Scale Free Topology Model
Fit,signed R^2",type="n",
    main = paste("Scale independence"));
text(maize_sft$fitIndices[,1], -sign(maize_sft$fitIndices[,3])*
maize_sft$fitIndices[,2],
    labels=powers,cex=cex1,col="red");
# this line corresponds to using an R^2 cut-off
abline(h=0.90,col="red")
# Mean connectivity as a function of the soft-thresholding power
plot(maize_sft$fitIndices[,1], maize_sft$fitIndices[,5],
    xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
    main = paste("Mean connectivity"))
text(maize_sft$fitIndices[,1], maize_sft$fitIndices[,5], labels=powers,
cex=cex1,col="red")
dev.off()
```

```
# identify modules
bwnet_block = blockwiseModules(datExpr, maxBlockSize = 20000,
                     power = 14, minModuleSize = 30,
                     reassignThreshold = 0, mergeCutHeight = 0.25,
                     numericLabels = TRUE,
                     saveTOMs = TRUE,
                     saveTOMFileBase = "TOM-blockwise",
                     verbose = 3)

table(bwnet_block$colors)
moduleLabels = bwnet_block$colors
# Convert labels to colors for plotting
bwModuleColors = labels2colors(moduleLabels)

pdf(file = "Gene_dendrogram_module_colors.pdf", width = 20, height = 10);
# Plot the dendrogram and the module colors underneath
plotDendroAndColors(bwnet_block$dendrograms[[1]],
bwModuleColors[bwnet_block$blockGenes[[1]]],
                  "Module colors", main = "Gene dendrogram and module
colors",
                  dendroLabels = FALSE, hang = 0.03,
                  addGuide = TRUE, guideHang = 0.05)
dev.off()


#######################################################
######### RELATE MODULES TO EXTERNAL DATA #########
#######################################################

# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr, bwModuleColors)$eigengenes
# number of samples (individuals) x number of eigengenes
MEs = orderMEs(MEs0)
# number of samples (individuals) x number of eigengenes
moduleTraitCor = cor(MEs, datTraits, use = "p");
# number of eigengenes x number of traits
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);
# number of eigengenes x number of traits

pdf(file = "Module-trait_relationships.pdf", width = 9, height = 15);
# 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 = colnames(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"))

dev.off()
```

```
######################################################

# Read in ALL traits
traitData = read.table("phenotypes_all_input.txt",header=T);
dim(traitData)
names(traitData)
TraitNames = traitData[,1]
allTraits = traitData[,2:146]
allTraits <- t(allTraits)
colnames(allTraits) <- TraitNames
data1 <- transform(allTraits, RIL= rownames(allTraits))

# Form a data frame analogous to expression data that will hold the
clinical traits.
RILSamples = rownames(datExpr);
traitRows = match(RILSamples, data1$RIL);
datTraits = allTraits[traitRows,];
collectGarbage();

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

pdf(file = "Module-trait_relationships_ALL_traits.pdf", width = 21, height
= 12);
# 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 = colnames(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 (all)"))
dev.off()


#####################################################
######### WRITE RESULTS TO AN OUTPUT FILE #########
#####################################################

# Define variable weight containing the weight column of datTrait
gls = as.data.frame(datTraits);
names(gls) = "GLS"
# names (colors) of the modules
modNames = substring(names(MEs), 3)
geneModuleMembership = as.data.frame(cor(datExpr, MEs, use = "p"));
```

```
# number of genes x number of Module Eigengenes (MEs)
# for each gene it is possible to see the Module Membership (MM) to each
Module
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership),
nSamples));
# number of genes x number of Module Eigengenes (MEs)
names(geneModuleMembership) = paste("MM", modNames, sep="");
names(MMPvalue) = paste("p.MM", modNames, sep="");

geneTraitSignificance = as.data.frame(cor(datExpr, gls, use = "p"));
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance),
nSamples));
names(geneTraitSignificance) = paste("GS.", names(gls), sep="");
names(GSPvalue) = paste("p.GS.", names(gls), sep="");


annot = read.csv(file = "annot_file.csv");
dim(annot)
names(annot)
probes = names(datExpr)
probes2annot = match(probes, annot$Agilent.ID)
# The following is the number or probes without annotation:
sum(is.na(probes2annot))
# Should return 0.

# Create the starting data frame
geneInfo0 = data.frame(Agilent.ID = probes,
                    annot.group = annot$annot.group[probes2annot],
                    ZM.ID = annot$ZM.ID[probes2annot],
                    AT.ID = annot$AT.ID[probes2annot],
                    AT.annot = annot$AT.annot[probes2annot],
                    Rice.ID = annot$Rice.ID[probes2annot],
                    Rice.annot = annot$Rice.annot[probes2annot],
                    B2G.description = annot$B2G.description[probes2annot],
                                          B2G.GO.terms =
annot$B2G.GO.terms[probes2annot],
                    B2G.enzyme = annot$B2G.enzyme[probes2annot],
                    moduleColor = bwModuleColors,
                    geneTraitSignificance,
                    GSPvalue)
# Order modules by their significance for weight
modOrder = order(-abs(cor(MEs, gls, use = "p")));
# Add module membership information in the chosen order
for (mod in 1:ncol(geneModuleMembership))
{
  oldNames = names(geneInfo0)
  geneInfo0 = data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                       MMPvalue[, modOrder[mod]]);
  names(geneInfo0) = c(oldNames, paste("MM.", modNames[modOrder[mod]],
sep=""),
                     paste("p.MM.", modNames[modOrder[mod]], sep=""))
}
# Order the genes in the geneInfo variable first by module color, then by
geneTraitSignificance
geneOrder = order(geneInfo0$moduleColor, -abs(geneInfo0$GS.GLS));
geneInfo = geneInfo0[geneOrder, ]


write.csv(geneInfo, file = "geneInfo_maize.csv")

#################################
```

```
######### VISUALIZATION #########
#################################

# Recalculate module eigengenes
MEs = moduleEigengenes(datExpr, bwModuleColors)$eigengenes
gls = as.data.frame(datTraits);
names(gls) = "GLS"

gls2 = abs(gls-9)
names(gls2) = "GLS.swop"

MET = orderMEs(cbind(MEs, gls, gls2))

# Plot the dendrogram
pdf(file = "Eigengene_network_A.pdf", width = 10, height = 8);
#par(cex = 1)
par(cex = 0.9)
plotEigengeneNetworks(MET, "Eigengene dendrogram", marDendro = c(0,4,2,0),
                     plotHeatmaps = FALSE)
dev.off()
# Plot the heatmap matrix (note: this plot will overwrite the dendrogram
plot)

pdf(file = "Eigengene_network_B.pdf", width = 10, height = 8);
#par(cex = 1)
par(cex = 0.9)
plotEigengeneNetworks(MET, "Eigengene adjacency heatmap", marHeatmap =
c(5,5,2,2),
                     plotDendrograms = FALSE, xLabelsAngle = 90)
dev.off()


##########################################
######### EXPORT TO CYTOSCAPE #########
##########################################

# Read in the probe annotation
annot = read.csv(file = "annot_file.csv");
dim(annot)

# calculate the adjacency and TOM matrices --> TAKES LONG
adjacency = adjacency(datExpr, power = 14);
dim(adjacency)
TOM = TOMsimilarity(adjacency);
save(TOM, file = "Maize_TOM.RData")

# Select modules
modules = c("greenyellow", "paleturquoise", "blue", "yellowgreen",
"magenta","turquoise","darkred","yellow")

# Select module probes
probes = names(datExpr)
inModule = is.finite(match(bwModuleColors, modules));
modProbes = probes[inModule];
modGenes = annot$AT.ID[match(modProbes, annot$Agilent.ID)];
# Select the corresponding Topological Overlap
modTOM = TOM[inModule, inModule];

dimnames(modTOM) = list(modProbes, modProbes)
# Export the network into edge and node list files Cytoscape can read
cyt = exportNetworkToCytoscape(modTOM,
  edgeFile = paste("CytoscapeInput-edges-", paste(modules, collapse="-"),
".txt", sep=""),
  nodeFile = paste("CytoscapeInput-nodes-", paste(modules, collapse="-"),
".txt", sep=""),
  weighted = TRUE,
  threshold = 0.02,
  nodeNames = modProbes,
  altNodeNames = modGenes,
  nodeAttr = bwModuleColors[inModule]);


##########################################
```