Supplemental Method 1. WGCNA script.

```
#Microarray and Trait data load

getwd()
workingDir = "C:/Users/..."
setwd(workingDir)
library(WGCNA)
library(flashClust)
options(stringsAsFactors = FALSE)
femData = read.csv("Veg_Data.csv")
traitData = read.csv("Traits1.csv")

femaleSamples1 = rownames(datExpr0);
traitRows1 = match(femaleSamples1, traitData$Sample)
datTraits1 = traitData[traitRows1, -1]
rownames(datTraits1) = traitData[traitRows1, 1]
collectGarbage()
ExprTraits = data.frame(datTraits1$Whorl, datTraits1$Month,
datExpr0)

#Unnecessary data removing and data transposing

datExpr0 = as.data.frame(t(femData[, -c(1)]))
names(datExpr0) = femData$gene;
rownames(datExpr0) = names(femData)[-c(1)]

#Identification of samples with too many missing data

NumberMissingByArray=apply( is.na(data.frame(datExpr0)),1,
sum)
NumberMissingByArray

#Verification and removal of genes with too many missing
values

gsg = goodSamplesGenes(datExpr0, verbose = 3)
gsg$allOK
if (!gsg$allOK)
{
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 = ",
")))
datExpr0 = datExpr0[gsg$goodSamples, gsg$goodGenes]
}
```

```
#Hierarchical clusterin of the samples

sampleTree = flashClust(dist(datExpr0), method = "average")
par(cex = 0.6);
par(mar = c(1,5,2,0.5))
plot(sampleTree, main = "Sample clustering to detect
outliers", sub="", xlab="", cex.lab = 1.5, cex.axis = 1.5,
cex.main = 2)
abline(h = 25, col = "red")


# Choosing the soft-thresholding power: analysis of network
topology

powers = c(c(1:10), seq(from = 12, to=20, by=2))
sft = pickSoftThreshold(datExpr, powerVector = powers,
verbose = 5)
write.csv(sft, file="SoftTrehold_Powers_for_Network.csv")
sizeGrWindow(9, 5)
par(mfrow = c(1,2))
cex1 = 0.9
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,c
ex=cex1,col="red")
abline(h=0.85,col="red")
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")
k=softConnectivity(datE=datExpr,power=14)
sizeGrWindow(10,5)
par(mfrow=c(1,2))
hist(k, col="darkgrey")
scaleFreePlot(k, main="Check scale free topology\n",
pch=19, col="black")

# Network constrcution

net = blockwiseModules(datExpr, maxBlockSize = 10000, power
= 4, minModuleSize = 5, reassignThreshold = 0,
mergeCutHeight = 0.25, numericLabels = TRUE,
pamRespectsDendro = FALSE, saveTOMs = TRUE, saveTOMFileBase
= "TOM", verbose = 3)
table(net$colors)
```

```
sizeGrWindow(12, 9)
mergedColors = labels2colors(net$colors)
plotDendroAndColors(net$dendrograms[[1]],
mergedColors[net$blockGenes[[1]]], "Module colors",
dendroLabels = FALSE, hang = 0.03, addGuide = TRUE,
guideHang = 0.05)
moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs
geneTree = net$dendrograms[[1]]

#Correlations between gene modules and traits

nGenes = ncol(datExpr)
nSamples = nrow(datExpr)
MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, datTraits, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor,
nSamples)
sizeGrWindow(10,6)
textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
signif(moduleTraitPvalue, 1), ")", sep = "")
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(9, 11, 3, 0))
labeledHeatmap(Matrix = moduleTraitCor,xLabels =
names(datTraits),yLabels = names(MEs),ySymbols =
names(MEs),colorLabels = FALSE, colors = blueWhiteRed(50),
textMatrix = textMatrix, setStdMargins = FALSE, cex.text =
0.5, cex.main = 2, cex.lab = 1.2, zlim = c(-1,1),main =
paste("Module-trait relationships"))

#Representing modules by eigengenes and relating eigengenes
to one another

signif(cor(MEs0, use="p"), 2)

#We define a dissimilarity measure between the module
eigengenes that keeps track of the sign of the correlation
#between the module eigengenes, and use it to cluster the
eigengene:
dissimME=(1-t(cor(MEs0, method="p")))/2
hclustdatME=hclust(as.dist(dissimME), method="average" )
par(mfrow=c(1,1))
plot(hclustdatME, main="Clustering tree based of the module
eigengenes")
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, datTraits, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor,
nSamples)
```

```
# Recalculate module eigengenes

MEas = moduleEigengenes(datExpr, moduleColors, excludeGrey
= FALSE, grey = ifelse(is.numeric(colors), 0,
"orange"))$eigengenes

# Plotting relationships of selected triats with eigengene
modules (it can made for any trait selecting the appropiate
trait in "datTraits$...")
Lines = as.data.frame(datTraits$Lines)
names(Lines) = "Lines"
Origin = as.data.frame(datTraits$Origin)
names(Origin) = "Origin"
Yield = as.data.frame(datTraits$Yield);
names(Yield) = "Yield"
KN = as.data.frame(datTraits$KN)
names(KN) = "KN"
TKW = as.data.frame(datTraits$TKW)
names(TKW) = "TKW"
DW.FW_Veg = as.data.frame(datTraits$DW.FW_Veg);
names(DW.FW_Veg) = "DW.FW_Veg"
WC_Veg = as.data.frame(datTraits$WC_Veg)
names(WC_Veg) = "WC_Veg"
C.N_Veg = as.data.frame(datTraits$C.N_Veg)
names(C.N_Veg) = "C.N_Veg"
N_Veg = as.data.frame(datTraits$N_Veg)
names(N_Veg) = "N_Veg"
C_Veg = as.data.frame(datTraits$C_Veg)
names(C_Veg) = "C_Veg"
Nitrate_Veg = as.data.frame(datTraits$Nitrate_Veg);
names(Nitrate_Veg) = "Nitrate_Veg"
DW.FW_15DAS = as.data.frame(datTraits$DW.FW_15DAS)
names(DW.FW_15DAS) = "DW.FW_15DAS"
WC_15DAS = as.data.frame(datTraits$WC_15DAS)
names(WC_15DAS) = "WC_15DAS"
C.N_15DAS = as.data.frame(datTraits$C.N_15DAS);
names(C.N_15DAS) = "C.N_15DAS"
N_15DAS = as.data.frame(datTraits$N_15DAS)
names(N_15DAS) = "N_15DAS"
C_15DAS = as.data.frame(datTraits$C_15DAS)
names(C_15DAS) = "C_15DAS"
Nitrate_15DAS = as.data.frame(datTraits$Nitrate_15DAS)
names(Nitrate_15DAS) = "Nitrate_15DAS"
MET = orderMEs(cbind(MEs, Lines, Origin, Yield, KN, TKW,
DW.FW_Veg, WC_Veg, C.N_Veg, N_Veg, C_Veg, Nitrate_Veg,
DW.FW_15DAS, WC_15DAS, C.N_15DAS, N_15DAS, C_15DAS,
Nitrate_15DAS))
plotEigengeneNetworks(MET, heatmapColors=blueWhiteRed(50),
excludeGrey = FALSE, greyLabel = "orange", "", marDendro =
c(0,4,1,2), marHeatmap = c(7,7,1,2), cex.lab = 0.8,
xLabelsAngle = 90)

# Plot the network

dissTOM = 1-TOMsimilarityFromExpr(datExpr, power = 14)
plotTOM = dissTOM^7;
diag(plotTOM) = NA;
sizeGrWindow(9,9)
TOMplot(plotTOM, geneTree, moduleColors, main = "Network
heatmap plot, all genes")

#Multi-dimensional scaling plot of the network

cmd1=cmdscale(as.dist(dissTOM),2)
sizeGrWindow(7, 6)
par(mfrow=c(1,1))
plot(cmd1, col=as.character(moduleColors), main="MDS plot",
xlab="Scaling Dimension 1", ylab="Scaling Dimension 2")

# Recalculate module eigengenes

datME=moduleEigengenes(datExpr,moduleColors)$eigengenes

# Measure of module significance as average gene
significance (it can be made for al the traits changing the
"datTraits~$..." value)

GS1=as.numeric(cor(datTraits$Yield,datExpr, use="p"))
GeneSignificance1=abs(GS0)
ModuleSignificanceYield=tapply(GeneSignificance0,
moduleColors, mean, na.rm=T)
sizeGrWindow(8,7)
par(mar=c(8.3,5,2,2))
plotModuleSignificance(GeneSignificance1,moduleColors,
cex=1.5, cex.main=2, cex.lab=1.5, cex.axis=1.5,
cex.sub=1.5,axis.lty=1, las=3, ylim=c(0,0.7))

# Write the module significance data frame into a file

moduleInfo0 = data.frame(ModuleSignificanceLines,
ModuleSignificanceOrigin, ModuleSignificanceYield,
ModuleSignificanceKN, ModuleSignificanceTKW,
ModuleSignificanceDW.FW_Veg, ModuleSignificanceWC_Veg,
ModuleSignificanceC.N_Veg, ModuleSignificanceN_Veg,
ModuleSignificanceC_Veg, ModuleSignificanceNitrate_Veg,
ModuleSignificanceDW.FW_15DAS, ModuleSignificanceWC_15DAS,
ModuleSignificanceC.N_15DAS, ModuleSignificanceN_15DAS,
ModuleSignificanceC_15DAS, ModuleSignificanceNitrate_15DAS)
names(moduleInfo0) = names(datTraits)
# Write the data frame into a file
```

```
write.table(moduleInfo0,
file="ModuleSignificanceTraits.csv", row.names=T,sep=",")

# Calculate the intramodular connectivity

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


#Generalizing intramodular connectivity for all genes on
the array

datKME=signedKME(datExpr, datME, outputColumnName="MM.")


# Gene screening method based on a detailed definition
module membership (it can be made for any trait changing to
the appropiate "y" value)

NS1=networkScreening(y=datTraits$Yield, datME=datME,
datExpr=datExpr, oddPower=3, blockSize=2000,
minimumSampleSize=4,addMEy=TRUE, removeDiag=FALSE,
weightESy=0.5, getQValues=TRUE)


# Output of the results of network screening analysis

GeneResultsNetworkScreening3=data.frame(GeneName=row.names(
NS3), NS3)
probes = names(datExpr)
geneInfo3 = data.frame(name = probes, moduleColor =
moduleColors, GeneResultsNetworkScreening3)
write.table(geneInfo3,
file="GeneResultsNetworkScreening_Veg_Yield.csv",
row.names=F,sep=",")
datMEy = data.frame(datTraits$Whorl, datME)
eigengeneSignificance1 = cor(datMEy, datTraits$Whorl)
eigengeneSignificance1[1,1] =
(1+max(eigengeneSignificance1[-1, 1]))/2
eigengeneSignificance1.pvalue =
corPvalueStudent(eigengeneSignificance1, nSamples =
length(datTraits$Whorl))
namesME=names(datMEy)
out1=data.frame(t(data.frame(eigengeneSignificance1,
eigengeneSignificance1.pvalue, namesME, t(datMEy))))
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.table(out1,
file="MEResultsNetworkScreeningWhorl.csv", row.names=TRUE,
col.names = TRUE, sep=",")

# Network output file for Cytoscape (it can be made for any
module cahnging the "module=..." value)

module = "black"
probes = names(datExpr)
inModule = is.finite(match(moduleColors, module))
modProbes = probes[inModule]
modGenes = annot$Name[match(modProbes, annot$SustainPine3)]
modTOM = TOM[inModule, inModule]
dimnames(modTOM) = list(modProbes, modProbes)
cyt = exportNetworkToCytoscape(modTOM, edgeFile =
paste("CytoscapeInput-edges-", paste(module, collapse="-"),
".txt", sep=""), nodeFile = paste("CytoscapeInput-nodes-",
paste(module, collapse="-"), ".txt", sep=""), weighted =
TRUE, threshold = 0, nodeNames = modProbes, altNodeNames =
modGenes, nodeAttr = moduleColors[inModule])


# Network output file for Cytoscape only for 100 genes (it
can be made for any module cahnging the "module=..." value)

TOM = TOMsimilarityFromExpr(datExpr, power = 5);
probes = names(datExpr)

cyt = exportNetworkToCytoscape(TOM, edgeFile =
paste("CytoscapeInput-edges-all-05.txt", sep=""), nodeFile
= paste("CytoscapeInput-nodes-all-05.txt", sep=""),
weighted = TRUE, threshold = 0.05, nodeNames = probes,
nodeAttr = moduleColors);
```