## **Table S1.** The used WGCNA commands.

```
#DEGs.txt is table S3.
# The data in trait_D.txt is as followed:
```

| ID | Cultivar | Coltime | Treatment |
|---|---|---|---|
| J01 | 1 | 1 | 0 |
| J02 | 1 | 1 | 0 |
| J03 | 1 | 1 | 0 |
| J31 | 1 | 2 | 1 |
| J32 | 1 | 2 | 1 |
| J33 | 1 | 2 | 1 |
| J71 | 1 | 3 | 1 |
| J72 | 1 | 3 | 1 |
| J73 | 1 | 3 | 1 |
| J111 | 1 | 4 | 1 |
| J112 | 1 | 4 | 1 |
| J113 | 1 | 4 | 1 |
| P01 | 0 | 1 | 0 |
| P02 | 0 | 1 | 0 |
| P03 | 0 | 1 | 0 |
| P31 | 0 | 2 | 1 |
| P32 | 0 | 2 | 1 |
| P33 | 0 | 2 | 1 |
| P71 | 0 | 3 | 1 |
| P72 | 0 | 3 | 1 |
| P73 | 0 | 3 | 1 |
| P111 | 0 | 4 | 1 |
| P112 | 0 | 4 | 1 |
| P113 | 0 | 4 | 1 |

```
#The following is the used WGCNA commands

setwd('F:/WGCNA/budding') #working directory is where the input file locates
library(WGCNA)
TPM=read.table('DEGs.txt',head=T,row.names=1,sep='\t')
dim(TPM)
names(TPM)
datExpr0= as.data.frame(t(TPM))
head(names(datExpr0))
rownames(datExpr0)
gsg = goodSamplesGenes(datExpr0, verbose = 3)
gsg$allOK

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]
}

traitData=read.table('trait_D.txt',head=T,row.names=1,sep='\t')
dim(traitData)
names(traitData)
allTraits = traitData[ , ]
dim(allTraits)
names(allTraits)
TPMSamples = rownames(datExpr0)
TPMSamples
traitSamples =rownames(allTraits)
traitSamples
traitRows = match(TPMSamples, traitSamples)
datTraits = allTraits[traitRows,]
rownames(datTraits)
collectGarbage()

enableWGCNAThreads()
powers=c(c(1:10),seq(from=12,to=20,by=2))
powers
sft=pickSoftThreshold(datExpr0,powerVector=powers, verbose=5)
sft
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",main = paste("Scale independence"))

text(sft$fitIndices[,1],-sign(sft$fitIndices[,3])*sft$fitIndices[,2],labels=powers,cex=cex1,col='red')
abline(h=0.80,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")
```

```
cor=WGCNA::cor
net=blockwiseModules(datExpr0, power= 12, networkType='unsigned', TOMType='unsigned',
minModuleSize = 100, mergeCutHeight=0.25, numericLabels=TRUE,
pamRespectsDendro=FALSE, saveTOMs=TRUE, saveTOMFileBase='graftingTOM',verbose=3)
cor=stats::cor

class(net)
names(net)
table(net$colors)

sizeGrWindow(3,4.5)
mergedColors=labels2colors(net$color)
plotDendroAndColors(net$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

nGenes= ncol(datExpr0)
nSamples= nrow(datExpr0)
MEs0 =moduleEigengenes(datExpr0, moduleColors)$eigengenes
MEs0[1:6, 1:6]
MEs =orderMEs(MEs0)
write.table(MEs, file = "epigengene_expression.xls",sep="\t")
moduleTraitCor= cor(MEs, datTraits, use = "p")
moduleTraitCor[1:10, ]
moduleTraitPvalue= corPvalueStudent(moduleTraitCor, nSamples)
moduleTraitPvalue[1:6, ]

pval=c(signif(moduleTraitPvalue, 2))
star=ifelse(pval>0.05,"",ifelse(pval>0.01,"*",ifelse(pval>0.001,"**",ifelse(pval>0.001,
"***","***"))))
textMatrix= paste(signif(moduleTraitCor, 2), star, sep = "")
dim(textMatrix)= dim(moduleTraitCor)
sizeGrWindow(10,16)
par(mar = c(9, 10, 3, 3))
labeledHeatmap(Matrix= moduleTraitCor,xLabels= names(datTraits),
yLabels= names(MEs), ySymbols= names(MEs),colorLabels= FALSE,colors= greenWhiteRed(50),
textMatrix= textMatrix,setStdMargins= FALSE,cex.text= 1,zlim =c(-1,1),main =paste("Module-
trait relationships"))

MEs=moduleEigengenes(datExpr0, moduleColors)$eigengenes
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="")
traitNames=names(datTraits)
geneTraitSignificance = as.data.frame(cor(datExpr0, datTraits, use = "p"))
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance), nSamples))
names(geneTraitSignificance) = paste("GS.", traitNames, sep="")
names(GSPvalue) = paste("p.GS.", traitNames, sep="")

for (trait in traitNames){
  traitColumn=match(trait,traitNames)

  for (module in modNames){
    column = match(module, modNames)
    moduleGenes = moduleColors==module

    if (nrow(geneModuleMembership[moduleGenes,]) > 1){
      #sizeGrWindow(7, 7)
      pdf(file=paste("9_", trait, "_", module,"_Module membership vs gene significance.pdf",sep=""),width=7,height=7)
      par(mfrow = c(1,1))
      verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                         abs(geneTraitSignificance[moduleGenes, traitColumn]),
                         xlab = paste("Module Membership in", module, "module"),
                         ylab = paste("Gene significance for ",trait),
                         main = paste("Module membership vs. gene significance\n"),
                         cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col = module)
      dev.off()
    }
  }
}

sizeGrWindow(5,7.5)
par(cex = 0.9)
plotEigengeneNetworks(MEs, "", marDendro = c(0,4,1,2), marHeatmap = c(3,4,1,2), cex.lab = 0.8,
xLabelsAngle= 90)

for (mod in 1:nrow(table(moduleColors)))
{
modules = names(table(moduleColors))[mod]
probes = names(datExpr0)
inModule = (moduleColors == modules)
modProbes = probes[inModule]
modGenes = modProbes
adjacency = adjacency(datExpr0[ , inModule], power = 12)
TOM = TOMsimilarity(adjacency)
modTOM = TOM
dimnames(modTOM) = list(modProbes, modProbes)
cyt = exportNetworkToCytoscape(modTOM,
edgeFile = paste("CytoscapeInput-edges-", modules , ".txt", sep=""),
nodeFile = paste("CytoscapeInput-nodes-", modules, ".txt", sep=""),
weighted = TRUE,
threshold = 0.02,
nodeNames = modProbes,
altNodeNames = modGenes,
nodeAttr = moduleColors[inModule])
}

datKME=signedKME(datExpr0, MEs, outputColumnName="kME_MM.")
write.csv(datKME, "all_kME_MM.csv")
probes = names(datExpr0)
for (mod in 1:nrow(table(moduleColors)))
{
module = names(table(moduleColors))[mod]
inModule = (moduleColors==module)
moduleGenes = as.data.frame(probes[inModule])
names(moduleGenes)= "genename"
column=match(module, modNames)
module_KME = as.data.frame(datKME[inModule,column])
names(module_KME)="KME"
rownames(module_KME)=moduleGenes$genename
FilterGenes = abs(module_KME$KME) > 0.9
table(FilterGenes)
module_hub = subset(module_KME, abs(module_KME$KME)>0.9)
write.csv(module_hub, file=paste(module,'_hub','.csv',sep=""))
}
```