WGCNA – Average Expression Plot for B73

```r
#!/usr/bin/Rscript --vanilla
rm(list=ls())

library(jpeg)
library(dplyr)
library(tidyr)
library(tibble)
library(stringr)
library(ggplot2)

library(argparse)

library(foreach)
library(iterators)
library(parallel)
library(doParallel)

library(WGCNA)
# library(KEGGREST)
# library(biomaRt)

set.seed(1)

# Enable WGCNA threads to speed up calculations
enableWGCNAThreads()
```

```r
#########################################################################
# Constants/Variables
#########################################################################
selected_genotype <- "B73"
softPower <- 14
minModuleSize <- 5
mergeCutHeight <- 0.0000001


##################################################
# Output folder
##################################################
output_path <- file.path(
  paste0(
    "/home/ycth8/data/projects/2021_05_30_summer_WGCNA/Maize_proteomics_output/",
    paste0("2021_06_10_", selected_genotype, "_step_by_step_network_construction")
  )
)

if(!dir.exists(output_path)){
  dir.create(output_path, showWarnings=FALSE, recursive=TRUE)
  if(!dir.exists(output_path)){
    quit(status=1)
  }
}
```

```r
##################################################
# Read in input file
##################################################

folder_path = file.path("/home/ycth8/data/projects/2021_05_30_summer_WGCNA/Maize_proteomics_output/")

datExpr = read.csv(
  file = file.path(folder_path, "datExpr.csv"),
  header = TRUE,
  row.names = 1,
  check.names = FALSE,
  stringsAsFactors = FALSE
)

datExpr = datExpr[startsWith(rownames(datExpr), selected_genotype),]
```

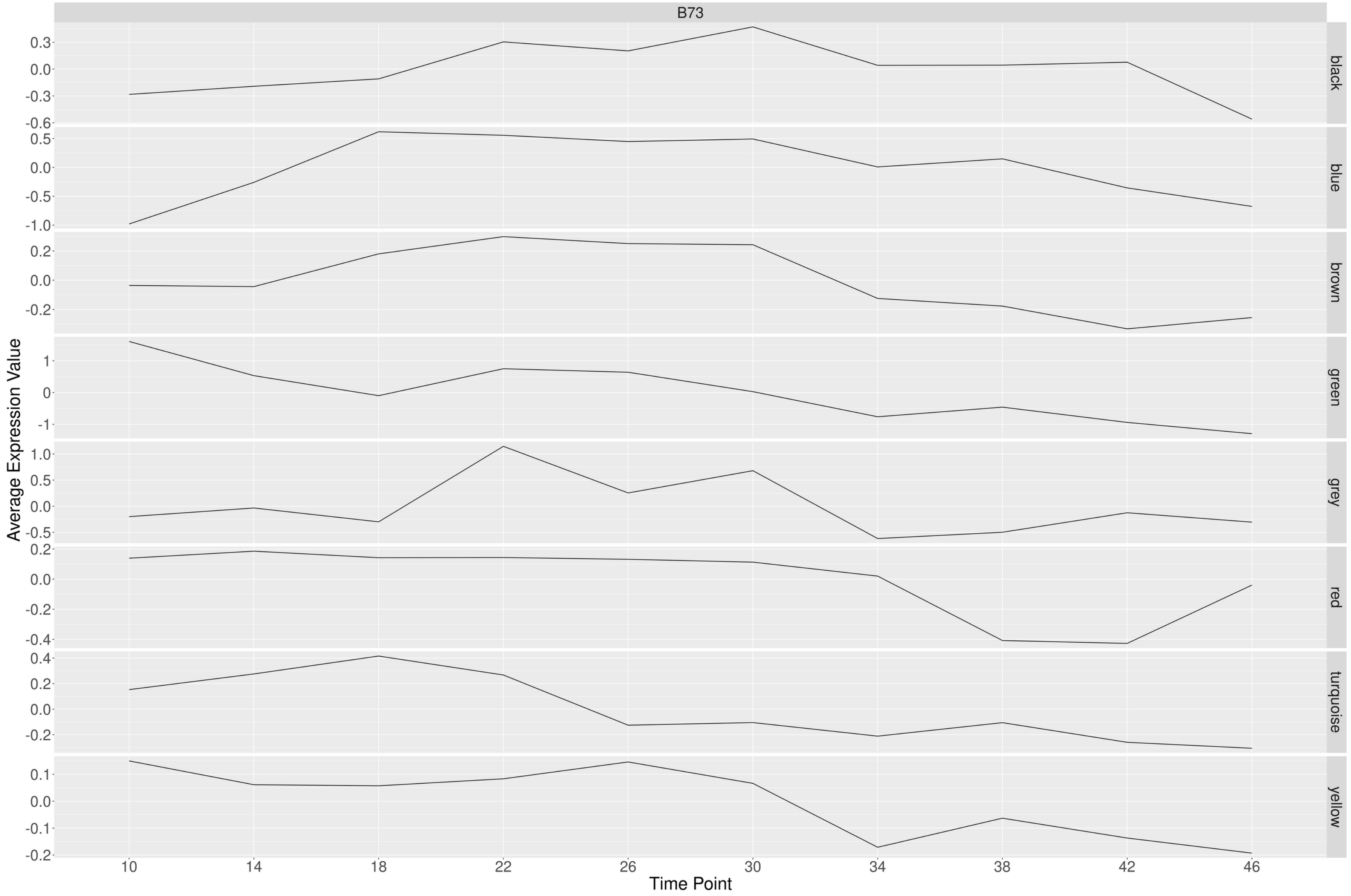
```r
##################################################
# Choose a set of soft-thresholding powers
##################################################

powers = 1:40
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)

# Plot tree
cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "softThreshold.jpeg"), width = 1920, height = 480)
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=sft$fitIndices[sft$fitIndices$Power == softPower, "SFT.R.sq"], 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")
abline(h=sft$fitIndices[sft$fitIndices$Power == softPower, "mean.k."], col="red")
dev.off()
```

```r
#################################################
# Make a gene tree and identify modules
#################################################
adjacency = adjacency(datExpr, power = softPower)

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

# Call the hierarchical clustering function
geneTree = hclust(as.dist(dissTOM), method = "average")

# Plot the resulting clustering tree (dendrogram)
cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "geneClustering.jpeg"), width = 1920, height = 720)
plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based dissimilarity",
     labels = FALSE, hang = 0.04)
dev.off()

jpeg(filename = file.path(output_path, "geneClustering_with_gene_id.jpeg"), width = 1920, height = 720)
plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based dissimilarity",
     labels = colnames(datExpr), hang = 0.04)
dev.off()
```

```r
dynamicMods = cutreeDynamic(
  dendro = geneTree,
  distM = dissTOM,
  deepSplit = 2,
  pamRespectsDendro = FALSE,
  minClusterSize = minModuleSize
)
print(table(dynamicMods))


# Convert numeric lables into colors
dynamicColors = labels2colors(dynamicMods)

print(table(dynamicColors))

# Plot the dendrogram and colors underneath
cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "networkConstruction_dynamic_colors.jpeg"), width = 1920, height = 720)
plotDendroAndColors(
  geneTree,
  dynamicColors,
  "Dynamic Tree Cut",
  dendroLabels = FALSE,
  hang = 0.03,
  addGuide = TRUE,
  guideHang = 0.05,
  main = "Gene dendrogram and module colors"
)
dev.off()
```

```r
# Call an automatic merging function
merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = mergeCutHeight, verbose = 3)

# The merged module colors
mergedColors = merge$colors

print(table(mergedColors))

# Eigengenes of the new merged modules:
mergedMEs = merge$newMEs

cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "networkConstruction_dynamic_and_merged_colors.jpeg"), width = 1920, height = 7
plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),
                    c("Dynamic Tree Cut", "Merged dynamic"),
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
dev.off()

cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "networkConstruction_merged_colors.jpeg"), width = 1920, height = 720)
plotDendroAndColors(geneTree, mergedColors,
                    "Merged dynamic",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
dev.off()
```

```r
#################################################
# Organize merged average expression table and plot average expression
#################################################
MEList2 = moduleEigengenes(datExpr, colors = mergedColors)

# Create average expression table
averageExpr <- MEList2$averageExpr %>%
  rownames_to_column(var = "Sample") %>%
  pivot_longer(!Sample, names_to = "Module", values_to = "Measurement") %>%
  separate(Sample, c("Sample", "Time")) %>%
  as.data.frame(stringsAsFactors = FALSE)

averageExpr$Sample <- factor(averageExpr$Sample, levels = unique(averageExpr$Sample))

averageExpr$Module <- sub("^AE", "", averageExpr$Module)
averageExpr$Module <- factor(averageExpr$Module, levels = unique(averageExpr$Module))

write.csv(
  x = averageExpr,
  file = file.path(output_path, "mergedAverageExpr.csv"),
  na = "",
  quote = FALSE
)

# Plot average expression plot
p <- ggplot(data = averageExpr, mapping = aes(x = Time, y = Measurement, group=1)) +
  geom_line() +
  facet_grid(Module ~ Sample, scales = "free") +
  labs(x = "Time Point", y = "Average Expression Value")
```


B73
black
blue
brown
green
grey
red
turquoise
yellow
Average Expression Value
Time Point
10
14
18
22
26
30
34
38
42
46
0.3
0.0
-0.3
-0.6
0.5
-0.5
-1.0
0.2
-0.2
1
0
-1
1.0
-0.4
0.4
0.1
-0.1