WGCNA – Step-by-step Network Construction for O2

```r
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 <- "O2"
softPower <- 22
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)
  }
}
```
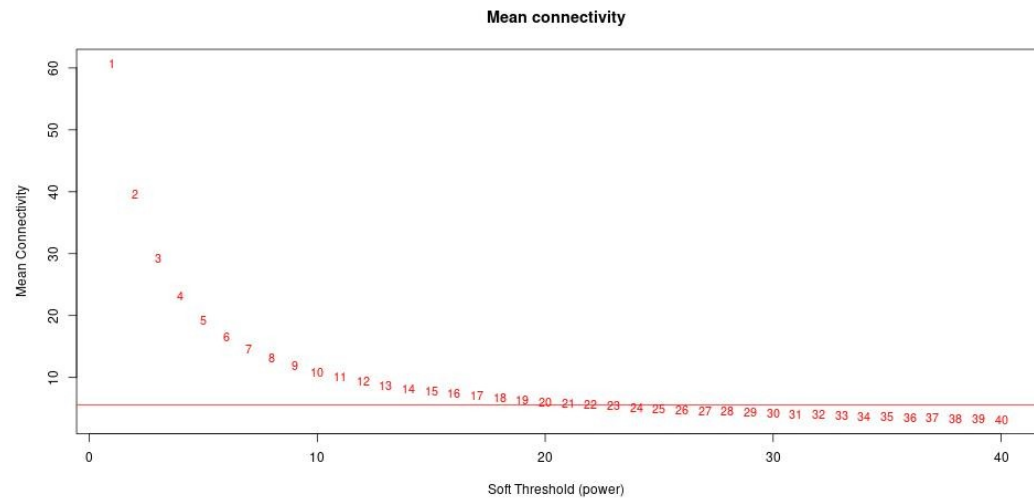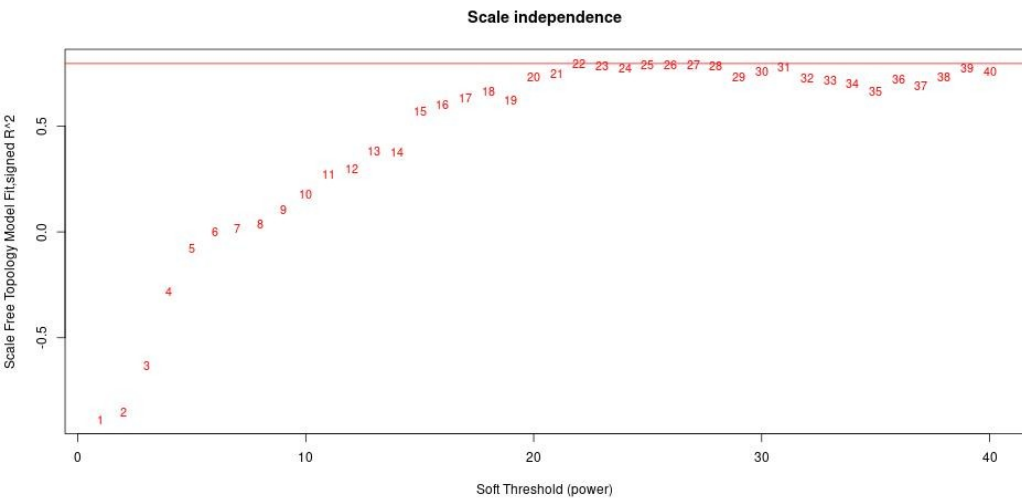
```
54   #################################################
55   # Read in input file
56   #################################################
57
58   folder_path = file.path("/home/ycth8/data/projects/2021_05_30_summer_WGCNA/Maize_proteomics_output/")
59
60   datExpr = read.csv(
61     file = file.path(folder_path, "datExpr.csv"),
62     header = TRUE,
63     row.names = 1,
64     check.names = FALSE,
65     stringsAsFactors = FALSE
66   )
67
68   datExpr = datExpr[startsWith(rownames(datExpr), selected_genotype),]
```
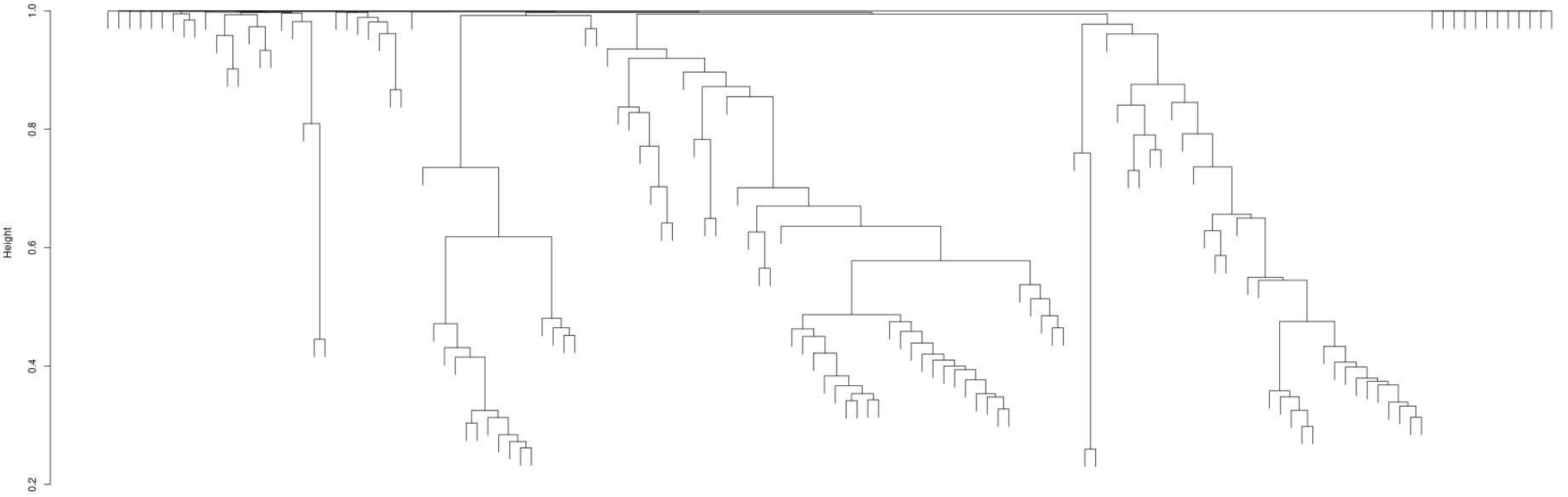
```r
 94    ####################################################
 95    # Choose a set of soft-thresholding powers
 96    ####################################################
 97
 98    powers = 1:40
 99    # Call the network topology analysis function
100    sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
101
102    # Plot tree
103    cat(rep("\n", 2))
104    jpeg(filename = file.path(output_path, "softThreshold.jpeg"), width = 1920, height = 480)
105    par(mfrow = c(1,2))
106    cex1 = 0.9
107
108    # Scale-free topology fit index as a function of the soft-thresholding power
109    plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
110        xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",
111        main = paste("Scale independence"))
112    text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
113        labels=powers,cex=cex1,col="red")
114    # this line corresponds to using an R^2 cut-off of h
115    abline(h=sft$fitIndices[sft$fitIndices$Power == softPower, "SFT.R.sq"], col="red")
116
117    # Mean connectivity as a function of the soft-thresholding power
118    plot(sft$fitIndices[,1], sft$fitIndices[,5],
119        xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
120        main = paste("Mean connectivity"))
121    text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")
122    abline(h=sft$fitIndices[sft$fitIndices$Power == softPower, "mean.k."], col="red")
123    dev.off()
124
125
126    # Collect garbage
127    collectGarbage()
```
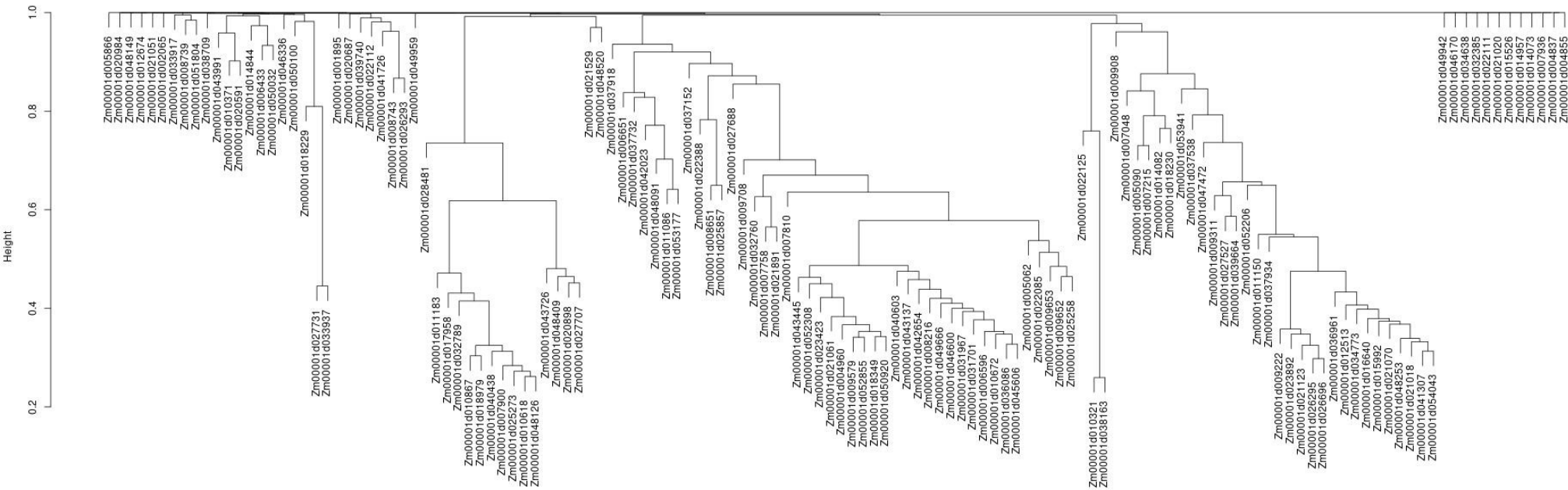
Scale independence

Mean connectivity

```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()
```

Gene clustering on TOM-based dissimilarity

Gene clustering on TOM-based dissimilarity
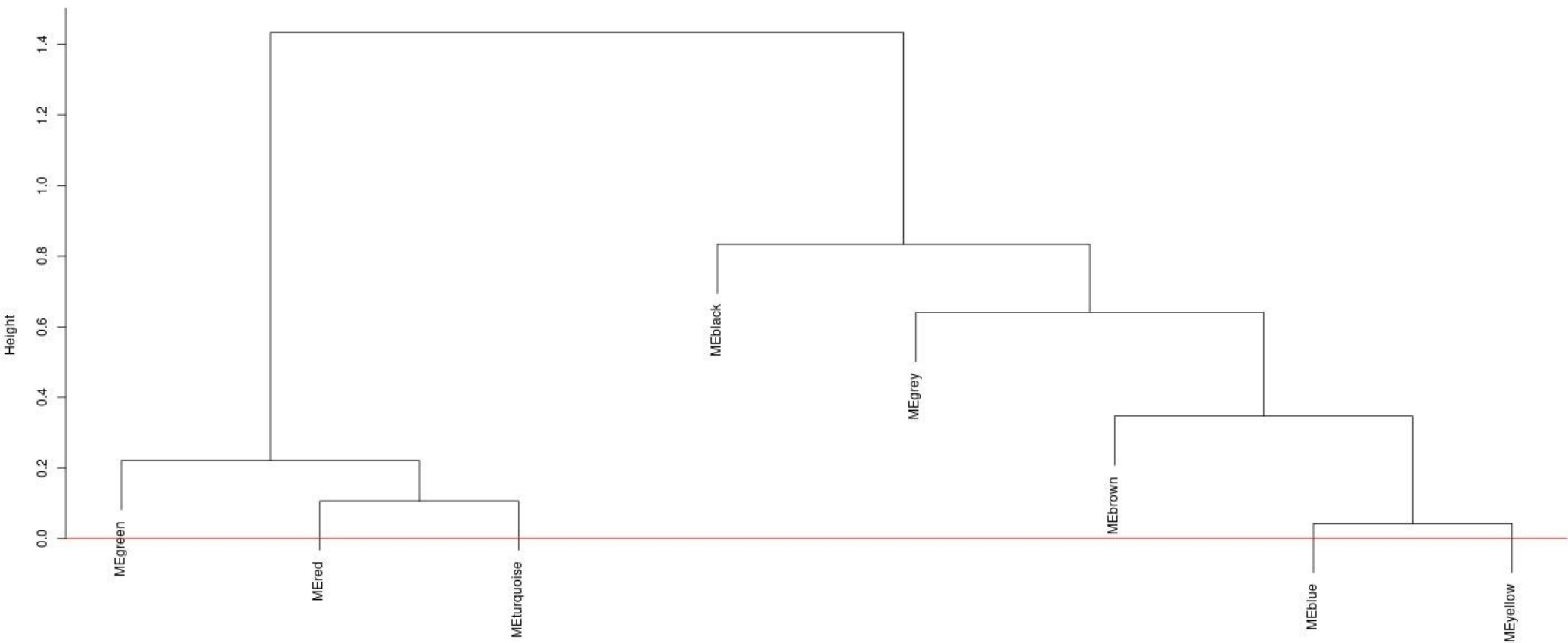
```r
158    # Module identification using dynamic tree cut:
159    dynamicMods = cutreeDynamic(
160        dendro = geneTree,
161        distM = dissTOM,
162        deepSplit = 2,
163        pamRespectsDendro = FALSE,
164        minClusterSize = minModuleSize
165    )
166    print(table(dynamicMods))
167
168
169    # Convert numeric lables into colors
170    dynamicColors = labels2colors(dynamicMods)
171
172    print(table(dynamicColors))
173
174    # Plot the dendrogram and colors underneath
175    cat(rep("\n", 2))
176    jpeg(filename = file.path(output_path, "networkConstruction_dynamic_colors.jpeg"), width = 1920, height = 720)
177    plotDendroAndColors(
178        geneTree,
179        dynamicColors,
180        "Dynamic Tree Cut",
181        dendroLabels = FALSE,
182        hang = 0.03,
183        addGuide = TRUE,
184        guideHang = 0.05,
185        main = "Gene dendrogram and module colors"
186    )
187    dev.off()
```

```
190    ####################################################
191    # Modules merging by clustering module eigengenes
192    ####################################################
193    # Calculate eigengenes
194    MEList = moduleEigengenes(datExpr, colors = dynamicColors)
195    MEs = MEList$eigengenes
```
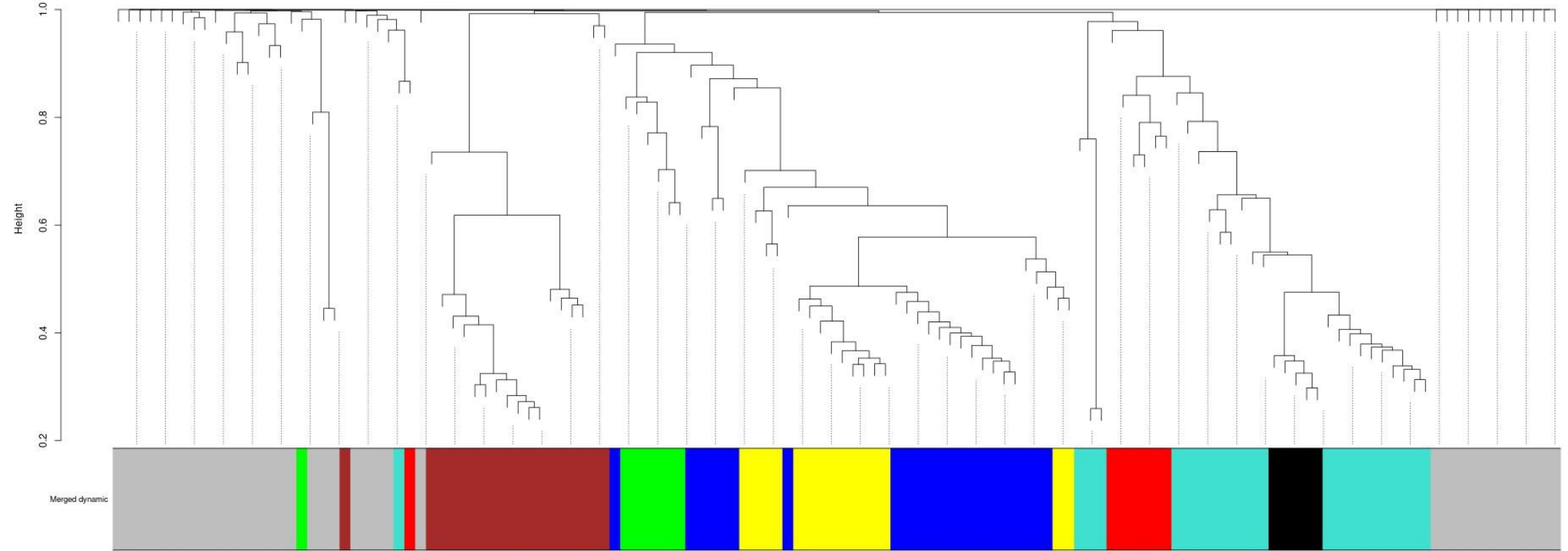
```r
266    # Calculate dissimilarity of module eigengenes
267    MEDiss = 1-cor(MEs)
268
269    # Cluster module eigengenes
270    METree = hclust(as.dist(MEDiss), method = "average")
271
272    # Plot the result
273    cat(rep("\n", 2))
274    jpeg(filename = file.path(output_path, "moduleEigengenesClustering.jpeg"), width = 1440, height = 720)
275    plot(
276      METree,
277      main = "Clustering of module eigengenes",
278      xlab = "",
279      sub = ""
280    )
281
282    # Plot the cut line into the dendrogram
283    abline(h=mergeCutHeight, col = "red")
284    dev.off()
285
```

Clustering of module eigengenes

```r
287    # Call an automatic merging function
288    merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = mergeCutHeight, verbose = 3)
289
290    # The merged module colors
291    mergedColors = merge$colors
292
293    print(table(mergedColors))
294
295    # Eigengenes of the new merged modules:
296    mergedMEs = merge$newMEs
297
298    cat(rep("\n", 2))
299    jpeg(filename = file.path(output_path, "networkConstruction_dynamic_and_merged_colors.jpeg"), width = 1920, height = 720)
300    plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),
301                        c("Dynamic Tree Cut", "Merged dynamic"),
302                        dendroLabels = FALSE, hang = 0.03,
303                        addGuide = TRUE, guideHang = 0.05)
304    dev.off()
305
306    cat(rep("\n", 2))
307    jpeg(filename = file.path(output_path, "networkConstruction_merged_colors.jpeg"), width = 1920, height = 720)
308    plotDendroAndColors(geneTree, mergedColors,
309                        "Merged dynamic",
310                        dendroLabels = FALSE, hang = 0.03,
311                        addGuide = TRUE, guideHang = 0.05)
312    dev.off()
```
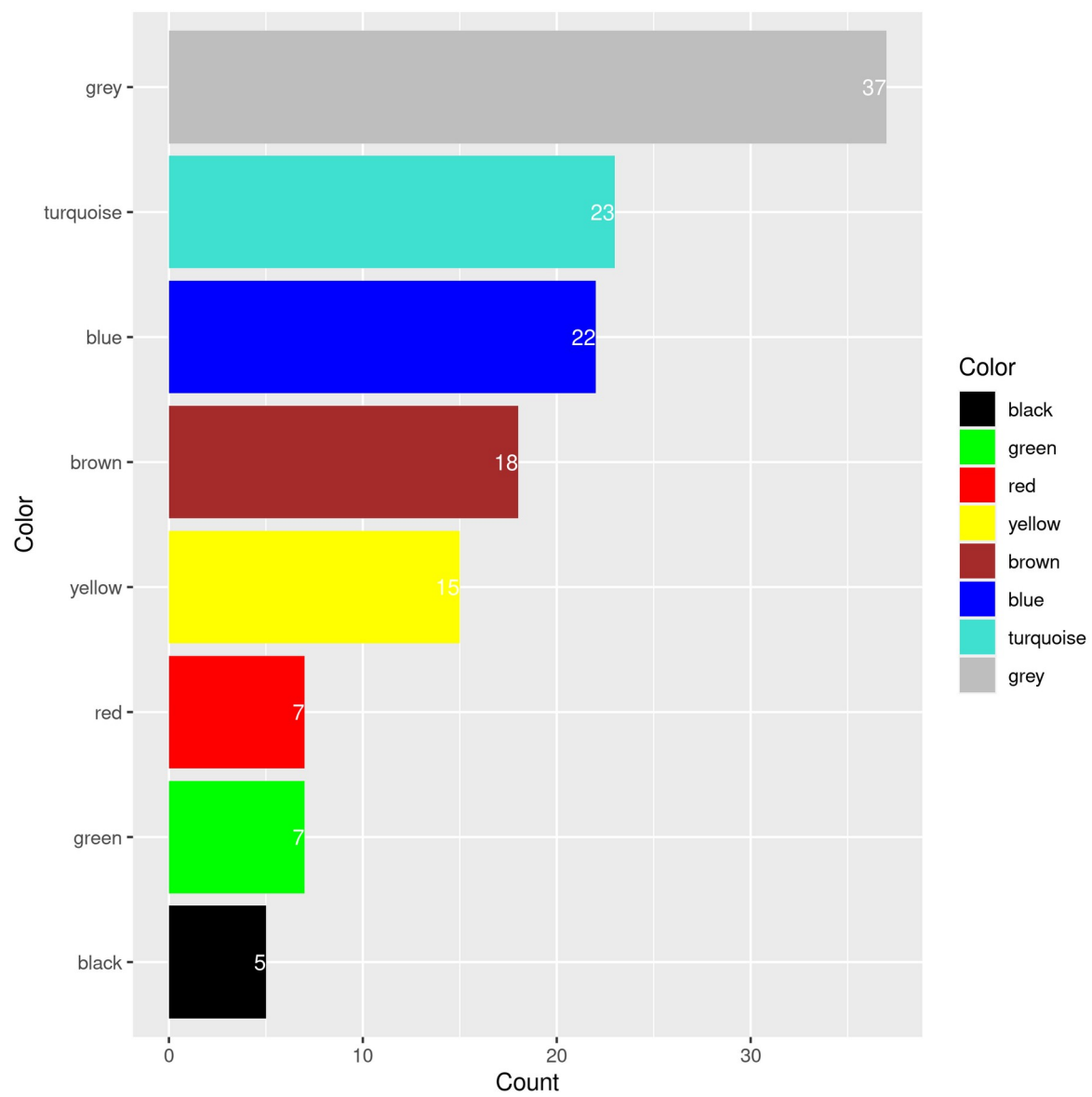
Cluster Dendrogram

Height

Merged dynamic

```r
###############################################
# Save genes and colors
###############################################
genes_colors_df <- data.frame(
  "Gene" = colnames(datExpr),
  "Color" = mergedColors,
  stringsAsFactors = FALSE
)

write.csv(
  x = genes_colors_df,
  file = file.path(output_path, "genes_colors_df.csv"),
  na = "",
  quote = FALSE,
  row.names = FALSE
)

```

```r
187   genes_colors_summary_df <- genes_colors_df %>%
188     group_by(Color) %>%
189     summarize(Count = n()) %>%
190     arrange(Count) %>%
191     as.data.frame(stringsAsFactors = FALSE)
192
193   genes_colors_summary_df$Color <- factor(genes_colors_summary_df$Color, levels = unique(genes_colors_summary_df$Color))
194
195   p <- ggplot(data=genes_colors_summary_df, aes(x = Color, y=Count)) +
196     geom_bar(mapping = aes(fill = Color), stat="identity") +
197     geom_text(aes(label=Count), hjust=1, color="white", size=3.5) +
198     coord_flip() +
199     scale_fill_manual(values = levels(genes_colors_summary_df$Color))
200
201   ggsave(
202     filename = "genes_colors_summary.png",
203     plot = p,
204     path = output_path
205   )
206
```

```
392    ################################################
393    # Save as RData
394    ################################################
395
396    # Rename to moduleColors
397    moduleColors = mergedColors
398
399    # Construct numerical labels corresponding to the colors
400    colorOrder = c("grey", standardColors(50))
401    moduleLabels = match(moduleColors, colorOrder)-1
402
403    MEs = mergedMEs
404
405    save(
406      datExpr, MEs, moduleLabels, moduleColors, geneTree,
407      file = file.path(output_path, paste0(selected_genotype, "-networkConstruction-stepByStep.RData"))
408    )
```