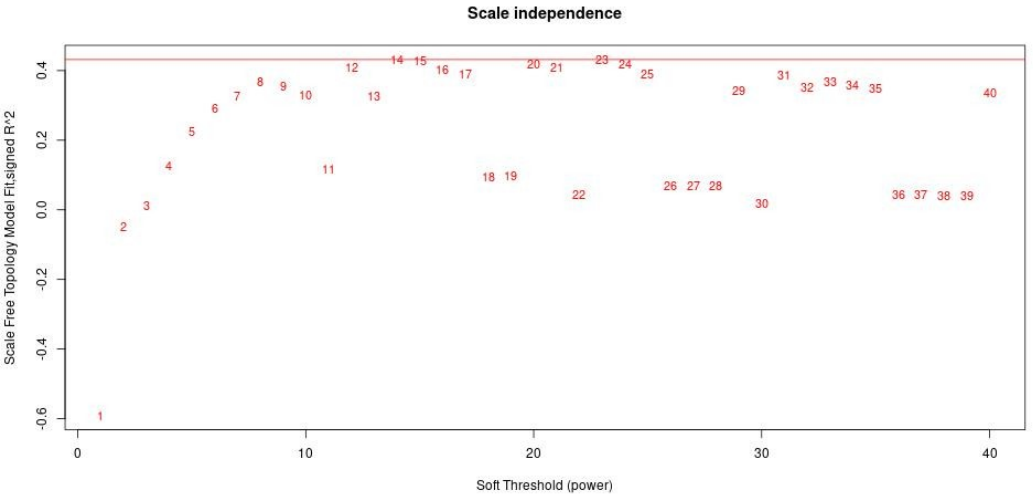
WGCNA – Step-by-step Network Construction for B73

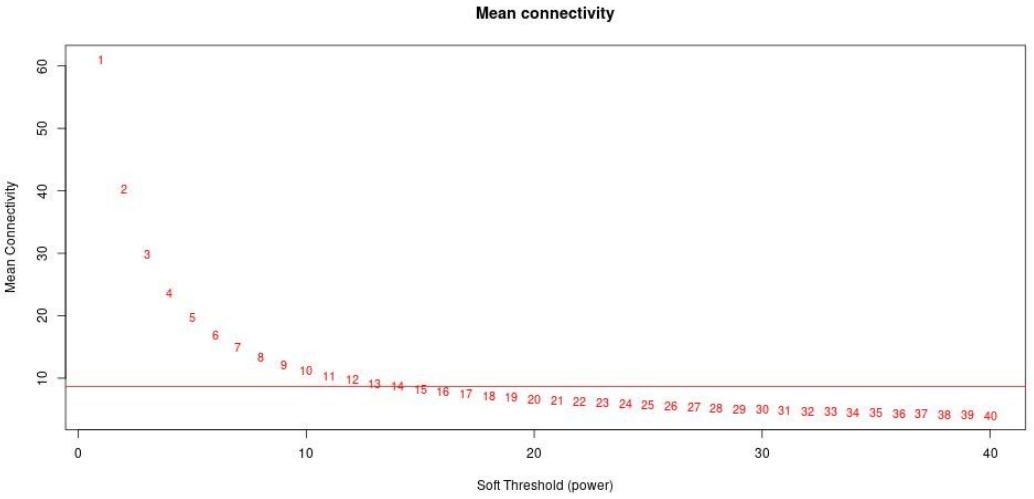
```r
rm(list=ls())

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 <- "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)
  }
}
```

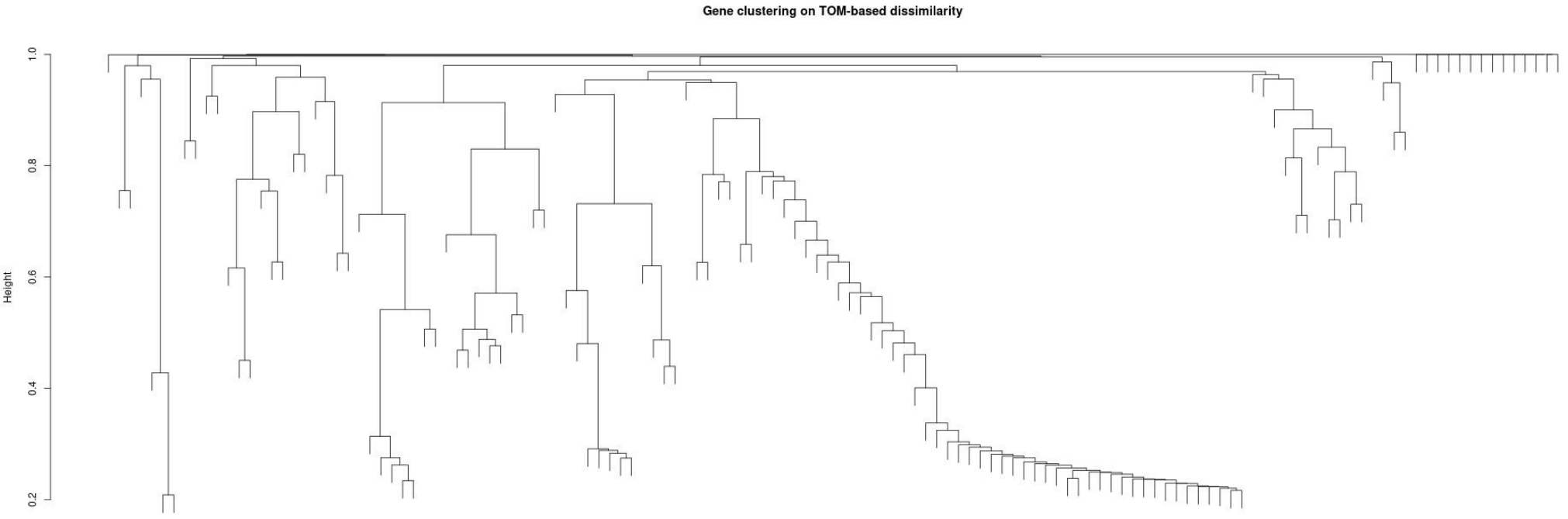
```
54  ###################################################
55  # Read in input file
56  ###################################################
57
58  folder_path = file.path("/home/ycth8/data/projects/2021_05_30_summer_WGCNA/Maize_proteomics_output/")
59
60  datExpr = read.csv(
61    file = file.path(folder_path, "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),]
```

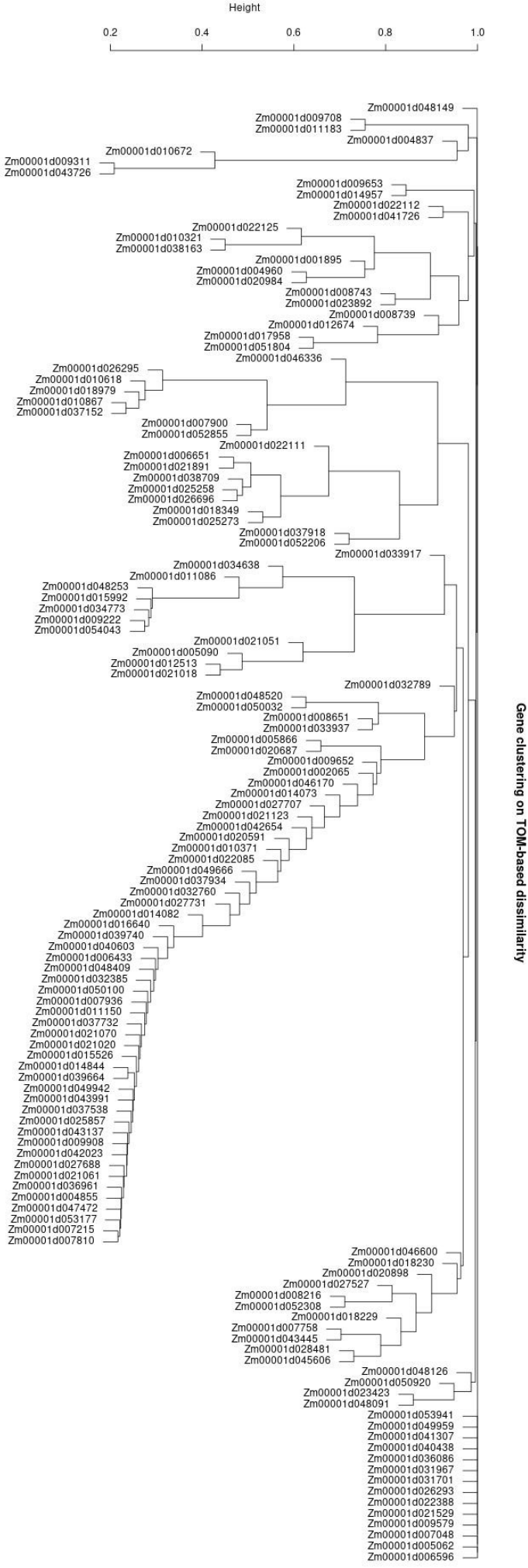
```r
################################################
# Choose a set of soft-thresholding powers
################################################

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()


# Collect garbage
collectGarbage()
```


Scale independence
Scale Free Topology Model Fit,signed R^2
Soft Threshold (power)


Mean connectivity
Mean Connectivity
Soft Threshold (power)

```
####################################################
# 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()
```

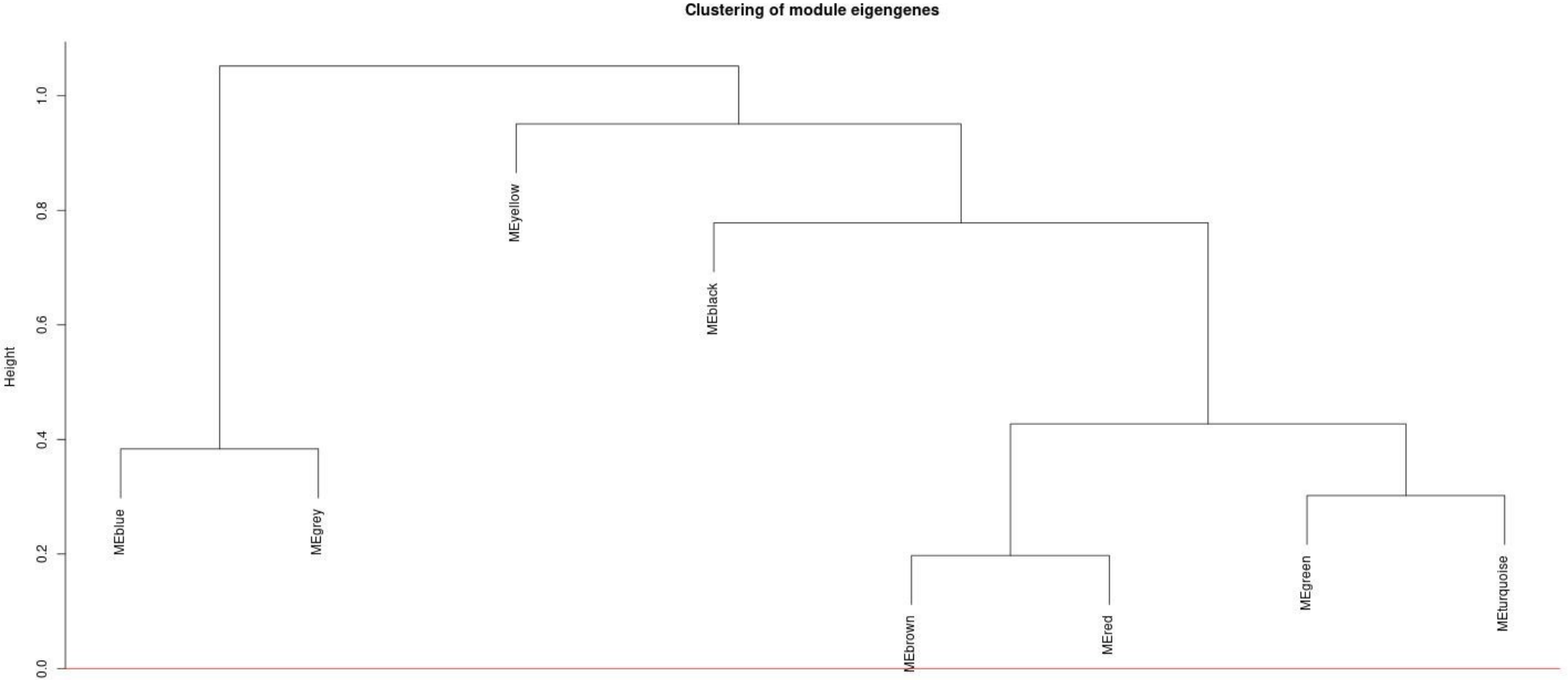

Gene clustering on TOM-based dissimilarity
Height
1.0
0.8
0.6
0.4
0.2


Height
0.2
0.4
0.6
0.8
1.0
Zm00001d048149
Zm00001d009708
Zm00001d011183
Zm00001d004837
Zm00001d010672
Zm00001d009311
Zm00001d043726
Zm00001d009653
Zm00001d014957
Zm00001d022112
Zm00001d041726
Zm00001d022125
Zm00001d010321
Zm00001d038163
Zm00001d001895
Zm00001d004960
Zm00001d020984
Zm00001d008743
Zm00001d023892
Zm00001d008739
Zm00001d012674
Zm00001d017958
Zm00001d051804
Zm00001d046336
Zm00001d026295
Zm00001d010618
Zm00001d018979
Zm00001d010867
Zm00001d037152
Zm00001d007900
Zm00001d052855
Zm00001d022111
Zm00001d006651
Zm00001d021891
Zm00001d038709
Zm00001d025258
Zm00001d026696
Zm00001d018349
Zm00001d025273
Zm00001d037918
Zm00001d052206
Zm00001d033917
Zm00001d034638
Zm00001d011086
Zm00001d048253
Zm00001d015992
Zm00001d034773
Zm00001d009222
Zm00001d054043
Zm00001d021051
Zm00001d005090
Zm00001d012513
Zm00001d021018
Zm00001d032789
Zm00001d048520
Zm00001d050032
Zm00001d008651
Zm00001d033937
Zm00001d005866
Zm00001d020687
Zm00001d009652
Zm00001d002065
Zm00001d046170
Zm00001d014073
Zm00001d027707
Zm00001d021123
Zm00001d042654
Zm00001d020591
Zm00001d010371
Zm00001d022085
Zm00001d049666
Zm00001d037934
Zm00001d032760
Zm00001d027731
Zm00001d014082
Zm00001d016640
Zm00001d039740
Zm00001d040603
Zm00001d006433
Zm00001d048409
Zm00001d032385
Zm00001d050100
Zm00001d007936
Zm00001d011150
Zm00001d037732
Zm00001d021070
Zm00001d021020
Zm00001d015526
Zm00001d014844
Zm00001d039664
Zm00001d049942
Zm00001d043991
Zm00001d037538
Zm00001d025857
Zm00001d043137
Zm00001d009908
Zm00001d042023
Zm00001d027688
Zm00001d021061
Zm00001d036961
Zm00001d004855
Zm00001d047472
Zm00001d053177
Zm00001d007215
Zm00001d007810
Zm00001d046600
Zm00001d018230
Zm00001d020898
Zm00001d027527
Zm00001d008216
Zm00001d052308
Zm00001d018229
Zm00001d007758
Zm00001d043445
Zm00001d028481
Zm00001d045606
Zm00001d048126
Zm00001d050920
Zm00001d023423
Zm00001d048091
Zm00001d053941
Zm00001d049959
Zm00001d041307
Zm00001d040438
Zm00001d036086
Zm00001d031967
Zm00001d031701
Zm00001d026293
Zm00001d022388
Zm00001d021529
Zm00001d009579
Zm00001d007048
Zm00001d005062
Zm00001d006596
Gene clustering on TOM-based dissimilarity

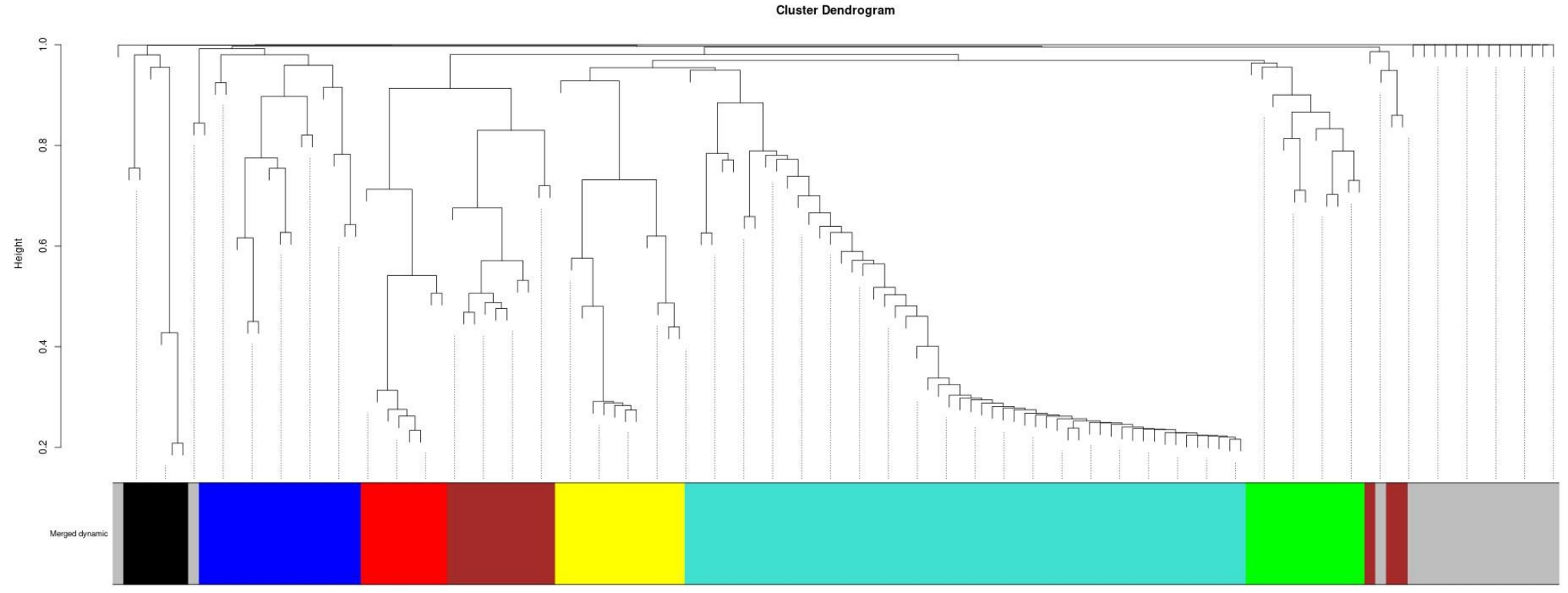
```r
# Module identification using dynamic tree cut:
dynamicMods = cutreeDynamic(
  dendro = geneTree,
  distM = dissTOM,
  deepSplit = 2,
  pamRespectsDendro = FALSE,
  minClusterSize = minModuleSize
)
print(table(dynamicMods))


# 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
##########################################################
# Modules merging by clustering module eigengenes
##########################################################
# Calculate eigengenes
MEList = moduleEigengenes(datExpr, colors = dynamicColors)
MEs = MEList$eigengenes
```

```r
# Calculate dissimilarity of module eigengenes
MEDiss = 1-cor(MEs)

# Cluster module eigengenes
METree = hclust(as.dist(MEDiss), method = "average")

# Plot the result
cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "moduleEigengenesClustering.jpeg"), width = 1440, height = 720)
plot(
  METree,
  main = "Clustering of module eigengenes",
  xlab = "",
  sub = ""
)

# Plot the cut line into the dendrogram
abline(h=mergeCutHeight, col = "red")
dev.off()

```

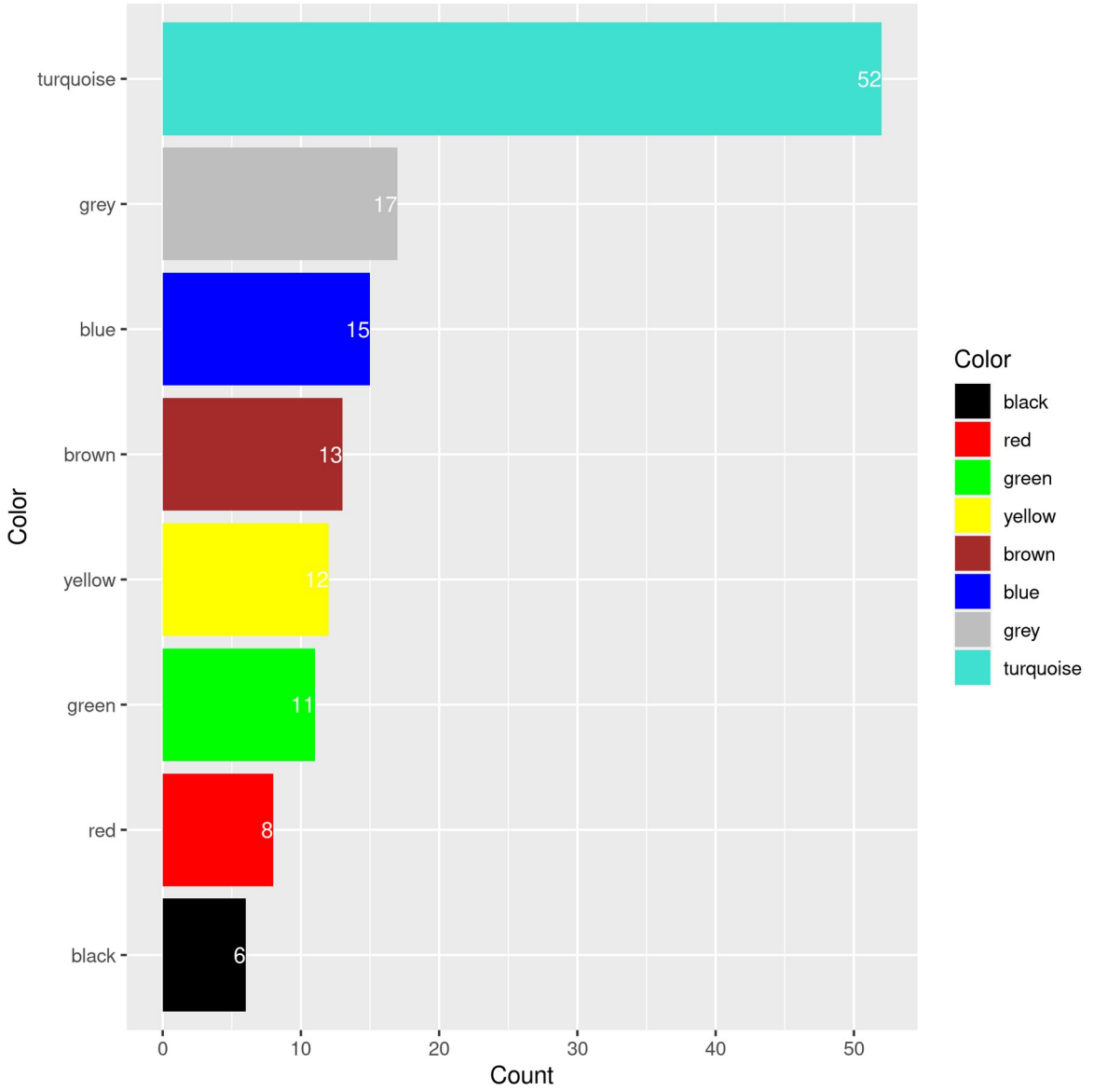

Clustering of module eigengenes
Height
0.0
0.2
0.4
0.6
0.8
1.0
MEblue
MEgrey
MEyellow
MEblack
MEbrown
MEred
MEgreen
MEturquoise

```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 = 720)
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()
```


Cluster Dendrogram
Height
1.0
0.8
0.6
0.4
0.2
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
genes_colors_summary_df <- genes_colors_df %>%
  group_by(Color) %>%
  summarize(Count = n()) %>%
  arrange(Count) %>%
  as.data.frame(stringsAsFactors = FALSE)

genes_colors_summary_df$Color <- factor(genes_colors_summary_df$Color, levels = unique(genes_colors_summary_df$Color))

p <- ggplot(data=genes_colors_summary_df, aes(x = Color, y=Count)) +
  geom_bar(mapping = aes(fill = Color), stat="identity") +
  geom_text(aes(label=Count), hjust=1, color="white", size=3.5) +
  coord_flip() +
  scale_fill_manual(values = levels(genes_colors_summary_df$Color))

ggsave(
  filename = "genes_colors_summary.png",
  plot = p,
  path = output_path
)

```


turquoise
grey
blue
brown
yellow
green
red
black
52
17
15
13
12
11
8
6
Color
0
10
20
30
40
50
Count
Color
black
red
green
yellow
brown
blue
grey
turquoise

```
###################################################
# Save as RData
###################################################

# Rename to moduleColors
moduleColors = mergedColors

# Construct numerical labels corresponding to the colors
colorOrder = c("grey", standardColors(50))
moduleLabels = match(moduleColors, colorOrder)-1

MEs = mergedMEs

save(
  datExpr, MEs, moduleLabels, moduleColors, geneTree,
  file = file.path(output_path, paste0(selected_genotype, "-networkConstruction-stepByStep.RData"))
)
```