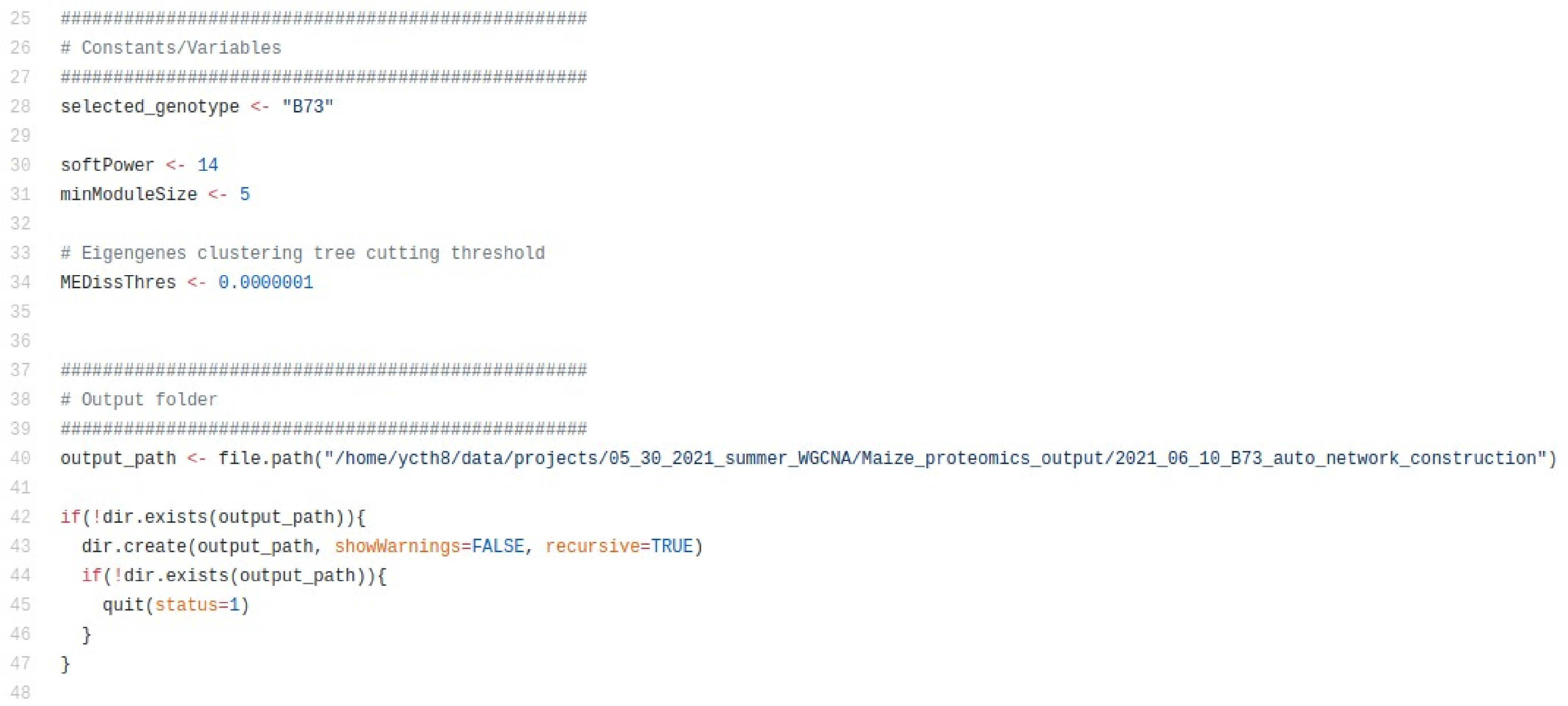
WGCNA – Auto Network Construction for B73

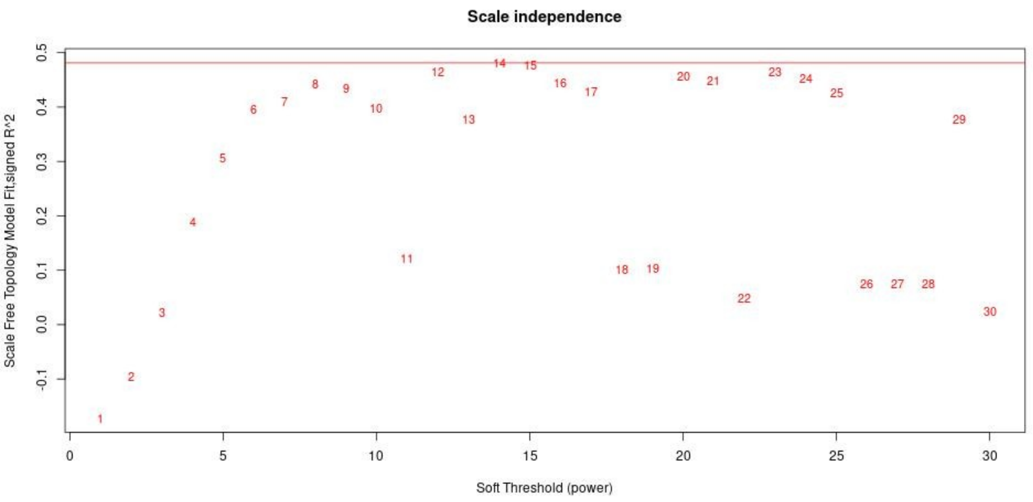
```r
rm(list=ls())

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

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()


```

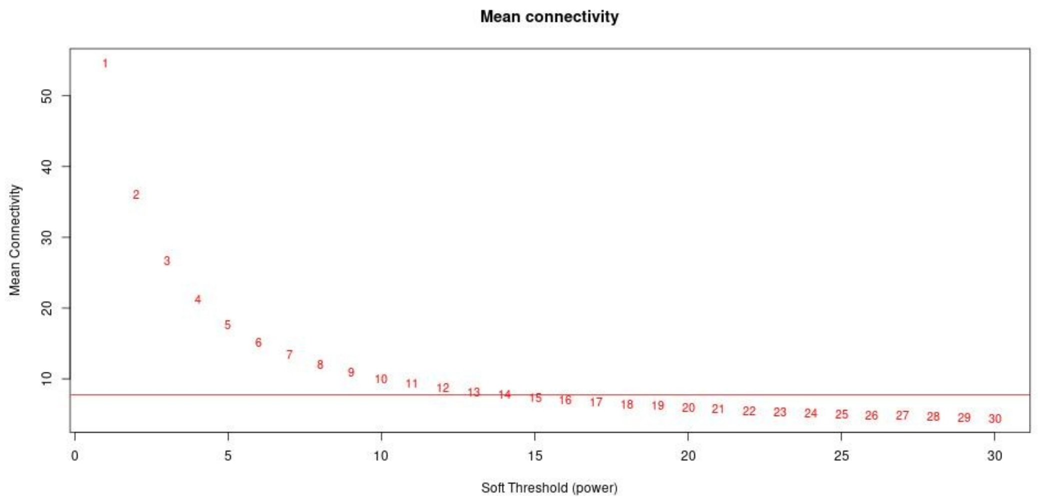


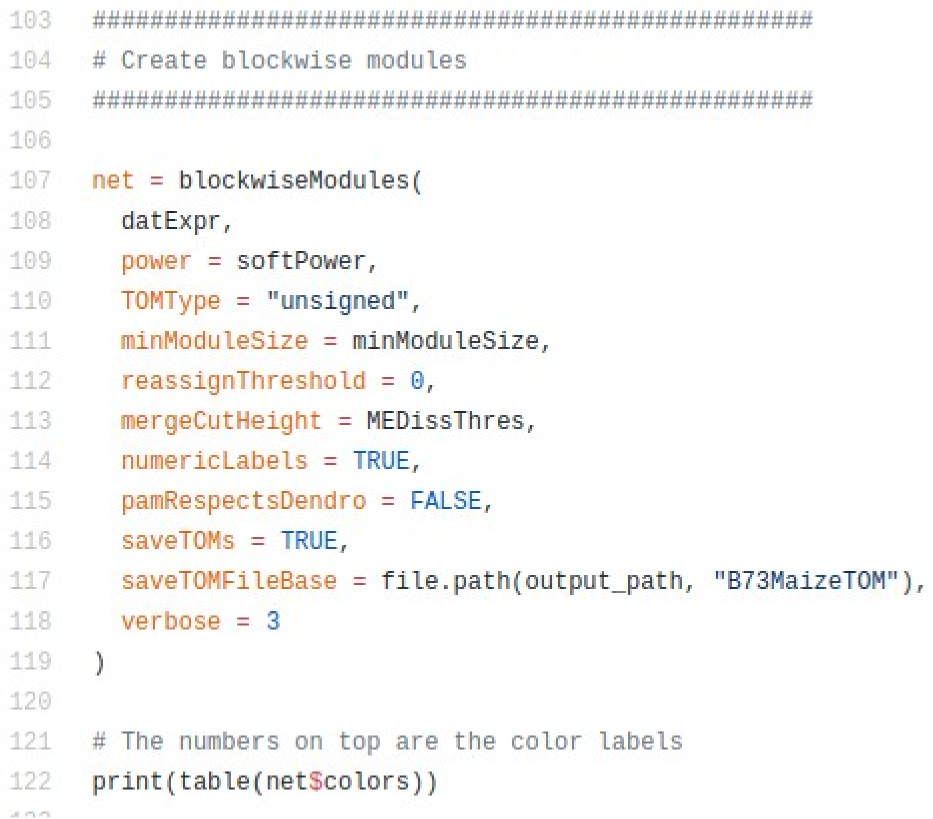
```r
##################################################
# Constants/Variables
##################################################
selected_genotype <- "B73"

softPower <- 14
minModuleSize <- 5

# Eigengenes clustering tree cutting threshold
MEDissThres <- 0.0000001


##################################################
# Output folder
##################################################
output_path <- file.path("/home/ycth8/data/projects/05_30_2021_summer_WGCNA/Maize_proteomics_output/2021_06_10_B73_auto_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/05_30_2021_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),]

```

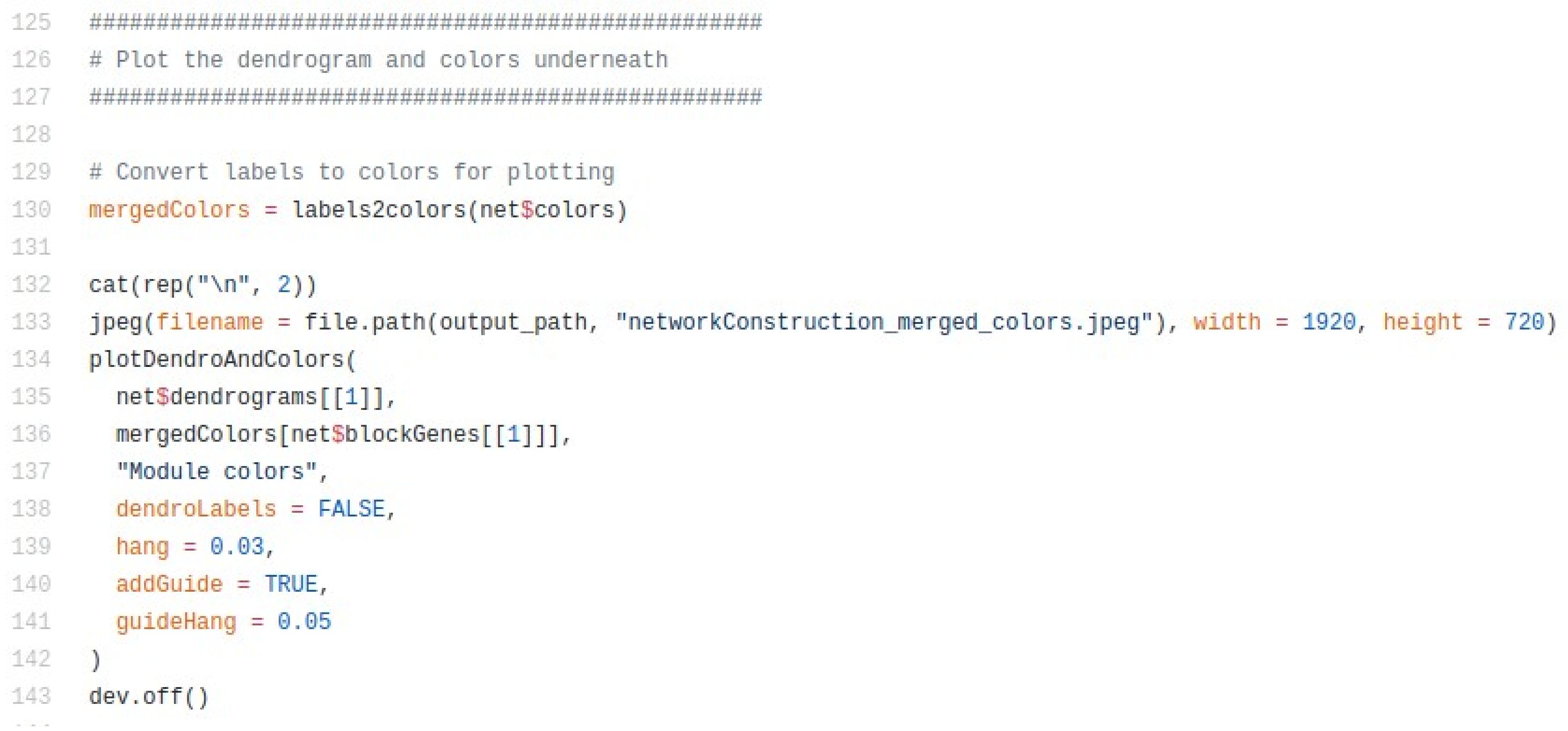
```r
##################################################
# Choose a set of soft-thresholding powers
##################################################

powers = 1:30
# 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)

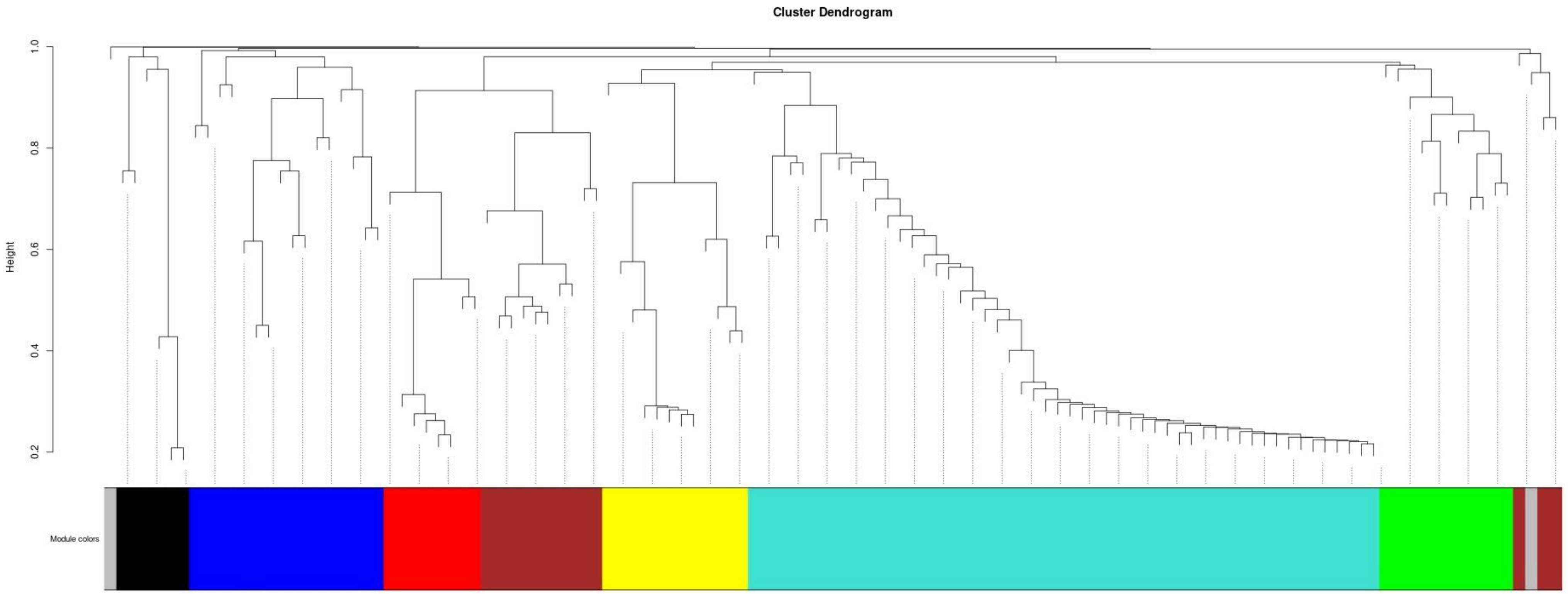

Mean connectivity
Mean Connectivity
Soft Threshold (power)



```r
##################################################
# Create blockwise modules
##################################################

net = blockwiseModules(
  datExpr,
  power = softPower,
  TOMType = "unsigned",
  minModuleSize = minModuleSize,
  reassignThreshold = 0,
  mergeCutHeight = MEDissThres,
  numericLabels = TRUE,
  pamRespectsDendro = FALSE,
  saveTOMs = TRUE,
  saveTOMFileBase = file.path(output_path, "B73MaizeTOM"),
  verbose = 3
)

# The numbers on top are the color labels
print(table(net$colors))
```

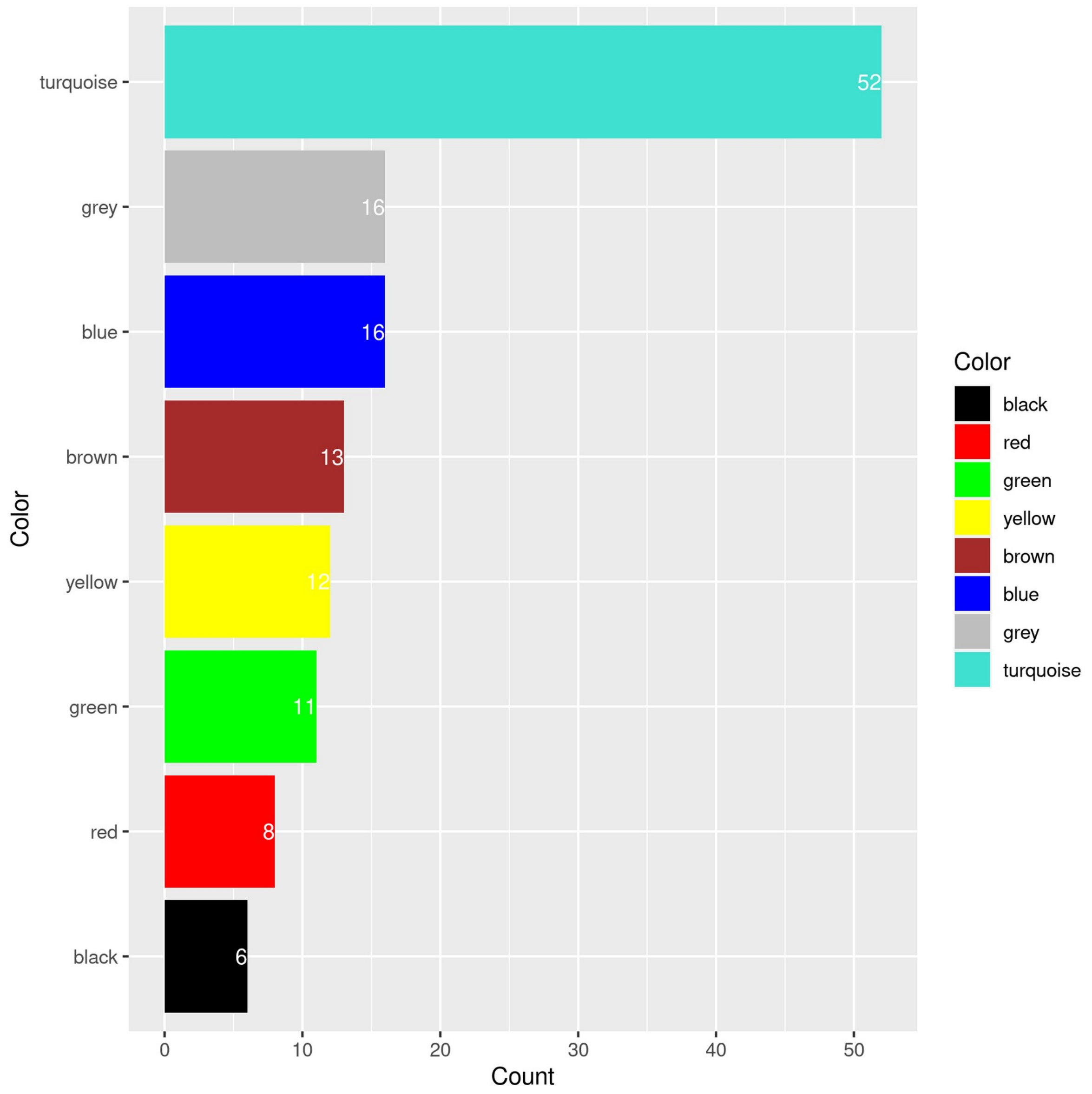


```
##################################################
# Plot the dendrogram and colors underneath
##################################################

# Convert labels to colors for plotting
mergedColors = labels2colors(net$colors)

cat(rep("\n", 2))
jpeg(filename = file.path(output_path, "networkConstruction_merged_colors.jpeg"), width = 1920, height = 720)
plotDendroAndColors(
  net$dendrograms[[1]],
  mergedColors[net$blockGenes[[1]]],
  "Module colors",
  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
Module colors

```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
16
16
13
12
11
8
6
Color
Count
0
10
20
30
40
50
Color
black
red
green
yellow
brown
blue
grey
turquoise

```r
##################################################
# Save important variables as RData
##################################################

moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
geneTree = net$dendrograms[[1]];
save(
  MEs, moduleLabels, moduleColors, geneTree,
  file = file.path(output_path, "B73-networkConstruction-auto.RData")
)
```