PAH_GEO_replication

Deniz Ertuğrul

2025-09-21

Libraries

```
library(dplyr)
library(GEOquery)
library(Biobase)
library(affy)
library(limma)
library(sva)
library(tidyr)
library(ggplot2)
library(ComplexHeatmap)
library(circlize)
library(clusterProfiler)
library(enrichplot)
library(ggplot2)
library(ggrepel)
library(VennDiagram)
library(org.Hs.eg.db)
library(hugene10sttranscriptcluster.db)
library(STRINGdb)
library(igraph)
library(ggraph)
library(ggvenn)


library(conflicted)
conflicts_prefer(WGCNA::cor)
conflicts_prefer(base::as.factor)


library(WGCNA)
allowWGCNAThreads()
```

```
## Allowing multi-threading with up to 16 threads.
```

Data Acquisition and Pre-processing

```
# Download datasets
gse117261 <- getGEO("GSE117261", GSEMatrix =TRUE, destdir = "./geo_files/")
```

```
## Found 1 file(s)
```

```
## GSE117261_series_matrix.txt.gz
## Using locally cached version: ./geo_files//GSE117261_series_matrix.txt.gz
```

```
## Using locally cached version of GPL6244 found here:
## ./geo_files//GPL6244.soft.gz
```

```
gse113439 <- getGEO("GSE113439", GSEMatrix =TRUE, destdir = "./geo_files/")
```

```
## Found 1 file(s)
```

```
## GSE113439_series_matrix.txt.gz
```

```
## Using locally cached version: ./geo_files//GSE113439_series_matrix.txt.gz
```

```
## Using locally cached version of GPL6244 found here:
## ./geo_files//GPL6244.soft.gz
```

```
# Extract expression matrices
exprs_1 <- exprs(gse117261[[1]])
exprs_2 <- exprs(gse113439[[1]])

# Extract phenotype data (sample information)
pheno_1 <- pData(gse117261[[1]])
pheno_2 <- pData(gse113439[[1]])

# Let's look at the dimensions
cat("Dataset 1 (GSE117261):\n")
```

```
## Dataset 1 (GSE117261):
```

```
print(dim(exprs_1))
```

```
## [1] 33297    83
```

```
cat("\nDataset 2 (GSE113439):\n")
```

```
##
## Dataset 2 (GSE113439):
```

```
print(dim(exprs_2))
```

```
## [1] 33297    26
```

Harmonize the Data

Before we can combine the datasets, we have two issues to solve:

1. The gene lists (row names) must be identical.
2. The phenotype data (column information) needs to be cleaned and made consistent.

Find the common genes between the two datasets

```
common_genes <- base::intersect(rownames(exprs_1), rownames(exprs_2))

# Filter both expression matrices to only include common genes
exprs_1_common <- exprs_1[common_genes, ]
exprs_2_common <- exprs_2[common_genes, ]
```

Clean and standardize phenotype data

We need a column that clearly identifies “PAH” vs “Normal” and a column for the batch. We’ll use dplyr for this.

```
# For GSE117261
pheno_1_clean <- pheno_1 %>%
  # Use the correct column name `clinical_group:ch1`
  dplyr::select(disease.state = `clinical_group:ch1`) %>%
  mutate(
    group = ifelse(grepl("PAH", disease.state, ignore.case = TRUE), "PAH", "Normal"),
    batch = "GSE117261"
  )

# For GSE113439
# Use the `disease state:ch1` column, which exists in this dataset
pheno_2_clean <- pheno_2 %>%
  dplyr::select(disease.state = `disease state:ch1`) %>%
  mutate(
    group = ifelse(grepl("PAH", disease.state, ignore.case = TRUE), "PAH", "Normal"),
    batch = "GSE113439"
  )


cat("GSE117261 Group Counts:\n")
```

```
## GSE117261 Group Counts:
```

```
print(table(pheno_1_clean$group))
```

```
##
## Normal    PAH
##     25     58
```

```
cat("\nGSE113439 Group Counts:\n")
```

```
##
## GSE113439 Group Counts:
```

```
print(table(pheno_2_clean$group))
```

```
##
## Normal    PAH
##     12     14
```

```
# Remove the patients with CTD or CHD or CTEPH from GSE113439
pheno_2_clean <- pheno_2_clean %>%
  dplyr::filter(!grepl("CTD|CHD|CTEPH", disease.state, ignore.case = TRUE))
# Filter the expression matrix accordingly
exprs_2_common_filtered <- exprs_2_common[, rownames(pheno_2_clean)]



cat("\nGSE113439 Group Counts after filtering CTD/CHD/CTEPH:\n")
```

```
##
## GSE113439 Group Counts after filtering CTD/CHD/CTEPH:
```

```
print(table(pheno_2_clean$group))
```

```
##
## Normal    PAH
##     11      6
```

```
head(pheno_2_clean)
```

```
##                       disease.state group     batch
## GSM3106326 idiopathic PAH patient   PAH GSE113439
## GSM3106330 idiopathic PAH patient   PAH GSE113439
## GSM3106331 idiopathic PAH patient   PAH GSE113439
## GSM3106336 idiopathic PAH patient   PAH GSE113439
## GSM3106337 idiopathic PAH patient   PAH GSE113439
## GSM3106339 idiopathic PAH patient   PAH GSE113439
```

Now we can merge the harmonized data into single objects.

```
# Combine expression data
combined_expr <- cbind(exprs_1_common, exprs_2_common_filtered)

# Combine phenotype data
combined_pheno <- rbind(pheno_1_clean, pheno_2_clean)

# Verify that the initial combination is correct (100 samples)
stopifnot(all(colnames(combined_expr) == rownames(combined_pheno)))
cat("Combined expression matrix dimensions:", dim(combined_expr), "\n")
```

```
## Combined expression matrix dimensions: 33297 100
```

```
# Create mapping from probes to gene symbols
cat("Mapping probe IDs to gene symbols...\n")
```

```
## Mapping probe IDs to gene symbols...
```

```
probe_to_gene <- AnnotationDbi::select(hugene10sttranscriptcluster.db,
                                      keys = rownames(combined_expr),
                                      columns = c("PROBEID", "SYMBOL", "ENTREZID"),
                                      keytype = "PROBEID")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# Remove probes without gene symbols
probe_to_gene_clean <- probe_to_gene[!is.na(probe_to_gene$SYMBOL) & probe_to_gene$SYMBOL !=
"", ]

# For multiple probes mapping to the same gene, keep the most variable one
gene_variance <- apply(combined_expr, 1, var)
probe_to_gene_clean$variance <- gene_variance[probe_to_gene_clean$PROBEID]

# Keep the most variable probe per gene
probe_to_gene_final <- probe_to_gene_clean %>%
  group_by(SYMBOL) %>%
  slice_max(variance, n = 1, with_ties = FALSE) %>%
  ungroup()

# Filter the expression matrix to keep only the final list of probes
combined_expr_annotated <- combined_expr[probe_to_gene_final$PROBEID, ]
# Assign the gene symbols as the new row names
rownames(combined_expr_annotated) <- probe_to_gene_final$SYMBOL

# Overwrite the original combined_expr with this new, cleaned, and annotated matrix
combined_expr <- combined_expr_annotated

# --- Final Check ---
cat("Success! The expression and phenotype data are synchronized and annotated.\n")
```

```
## Success! The expression and phenotype data are synchronized and annotated.
```

```
cat("Final dimensions of combined expression matrix:", dim(combined_expr), "\n")
```

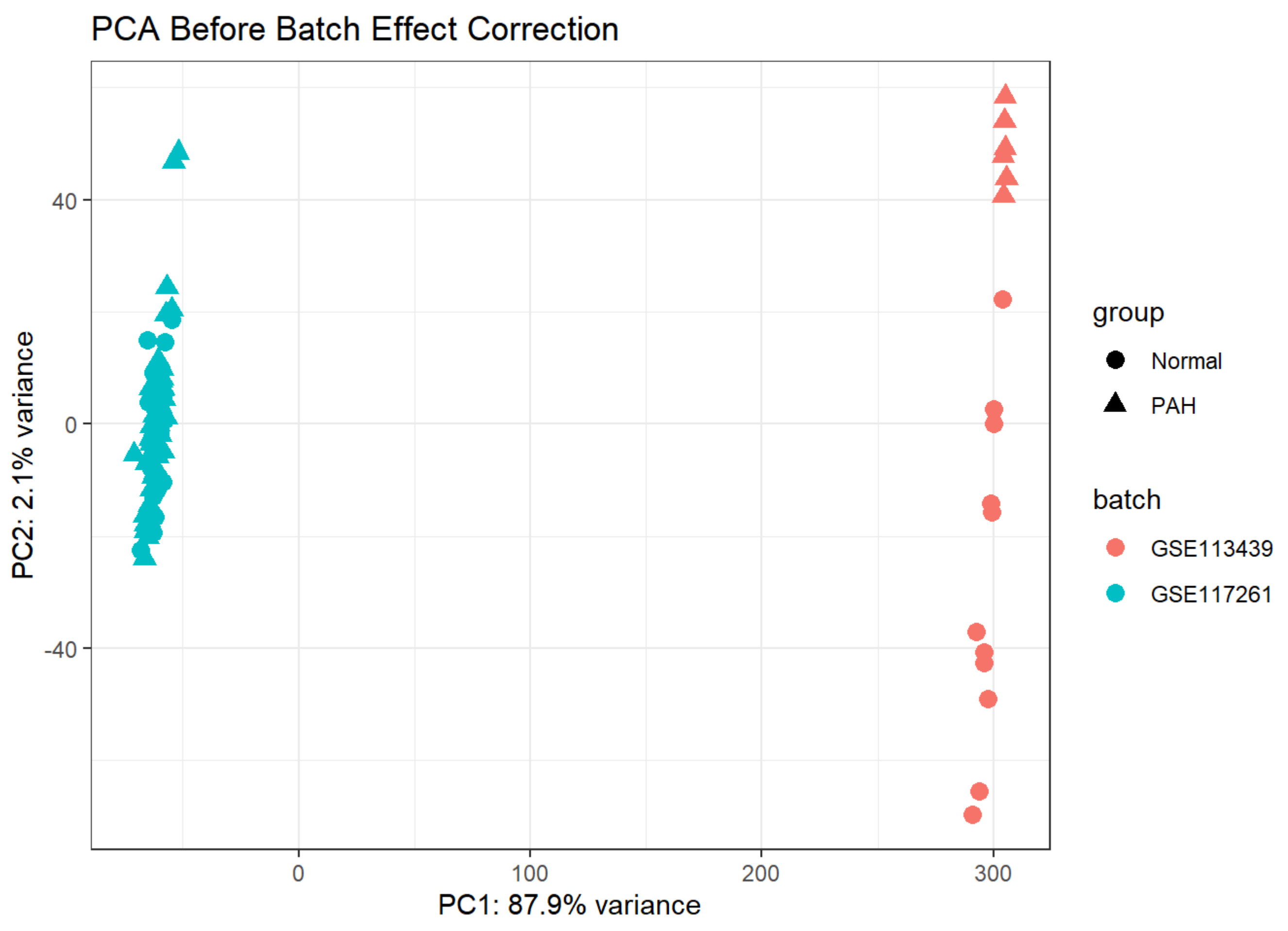
```
## Final dimensions of combined expression matrix: 21177 100
```

```
cat("Final dimensions of combined phenotype data:", dim(combined_pheno), "\n")
```

```
## Final dimensions of combined phenotype data: 100 3
```

Visualize the Batch Effect (Before Correction)

```
# Perform PCA
pca_before <- prcomp(t(combined_expr), scale. = TRUE)
pca_before_df <- as.data.frame(pca_before$x)
pca_before_df <- cbind(pca_before_df, combined_pheno)

# Plot PCA
ggplot(pca_before_df, aes(x = PC1, y = PC2, color = batch, shape = group)) +
  geom_point(size = 3) +
  ggtitle("PCA Before Batch Effect Correction") +
  theme_bw() +
  labs(
    x = paste0("PC1: ", round(summary(pca_before)$importance[2,1]*100, 1), "% variance"),
    y = paste0("PC2: ", round(summary(pca_before)$importance[2,2]*100, 1), "% variance")
  )
```



Correct for Batch Effects using ComBat

We need to provide `ComBat` with:

- `dat` : The combined, uncorrected expression matrix.
- `batch` : The vector indicating which batch each sample belongs to.
- `modcombat` : A model matrix of the biological variables we want to *preserve*. This is crucial. We tell `ComBat` to protect the “PAH vs. Normal” signal.

```
# Create the batch vector
batch <- combined_pheno$batch

# Create the model matrix to preserve the group differences. This tells ComBat NOT to remove
the differences between PAH and Normal samples
modcombat <- model.matrix(~group, data = combined_pheno)

# Apply ComBat
combat_expr <- ComBat(dat = as.matrix(combined_expr), batch = batch, mod = modcombat, par.pri
or = TRUE, prior.plots = FALSE)
```

```
## Found2batches
```

```
## Adjusting for1covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

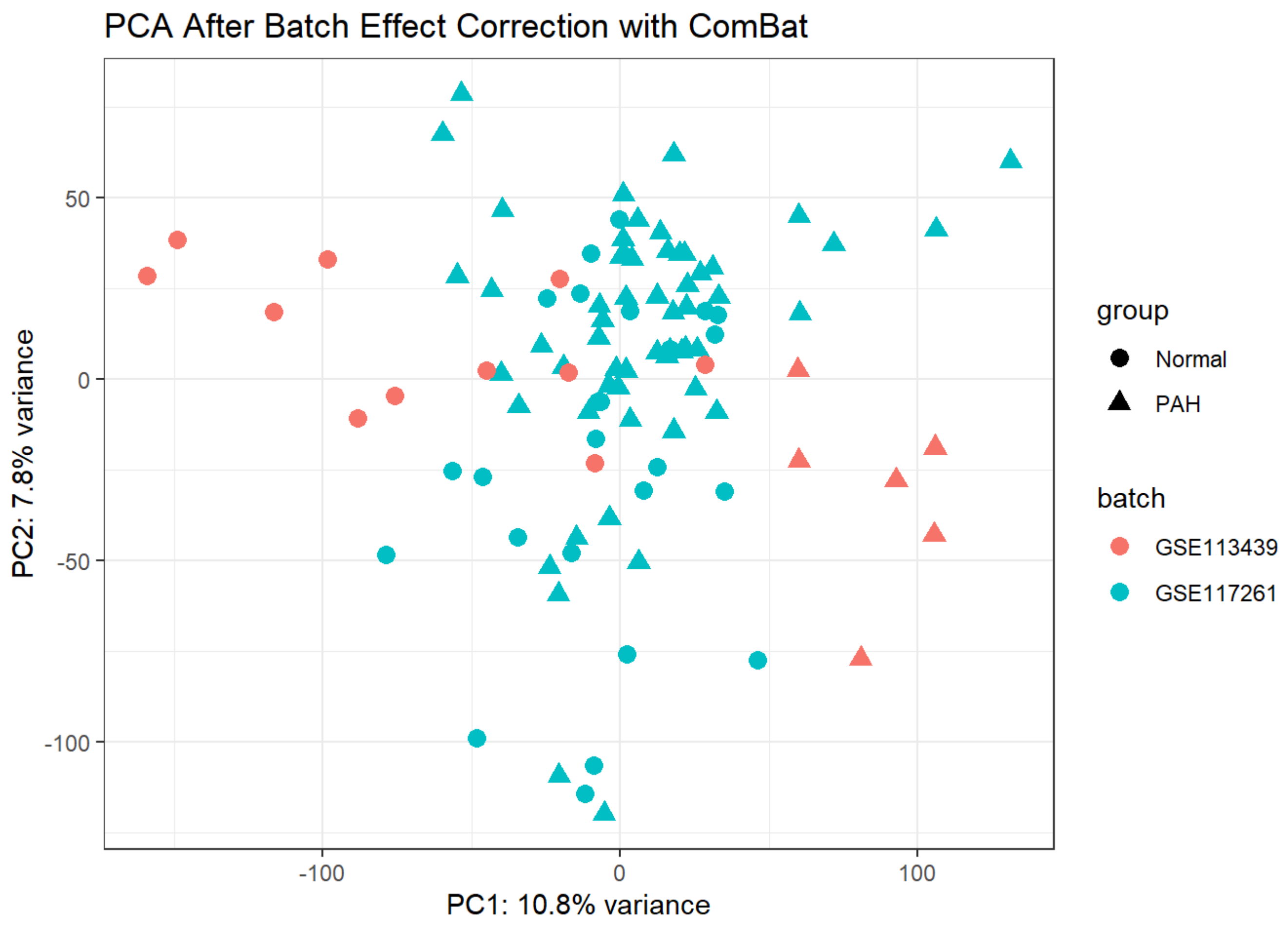
```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

Visualize the Results (After Correction)

```
# Perform PCA on the corrected data
pca_after <- prcomp(t(combat_expr), scale. = TRUE)
pca_after_df <- as.data.frame(pca_after$x)
pca_after_df <- cbind(pca_after_df, combined_pheno)

# Plot PCA
ggplot(pca_after_df, aes(x = PC1, y = PC2, color = batch, shape = group)) +
  geom_point(size = 3) +
  ggtitle("PCA After Batch Effect Correction with ComBat") +
  theme_bw() +
  labs(
    x = paste0("PC1: ", round(summary(pca_after)$importance[2,1]*100, 1), "% variance"),
    y = paste0("PC2: ", round(summary(pca_after)$importance[2,2]*100, 1), "% variance")
  )
```



Differential Gene Expression and WGCNA

Identify Differentially Expressed Genes (DEGs)

- The study uses the `limma` package to find DEGs (Section 2.2).
- We will create a design matrix that distinguishes between PAH and normal samples and then fit a linear model.

```
group <- factor(combined_pheno$group, levels = c("Normal", "PAH"))

design <- model.matrix(~group)


fit <- lmFit(combat_expr, design)
fit <- eBayes(fit)
top_genes <- topTable(fit, coef="groupPAH", number = Inf, p.value = 0.05, adjust.method = "B
H", lfc = 0.5)
cat("Number of DEGs (adj.P.Val < 0.05 & |log2FC| > 0.5):", nrow(top_genes), "\n")
```

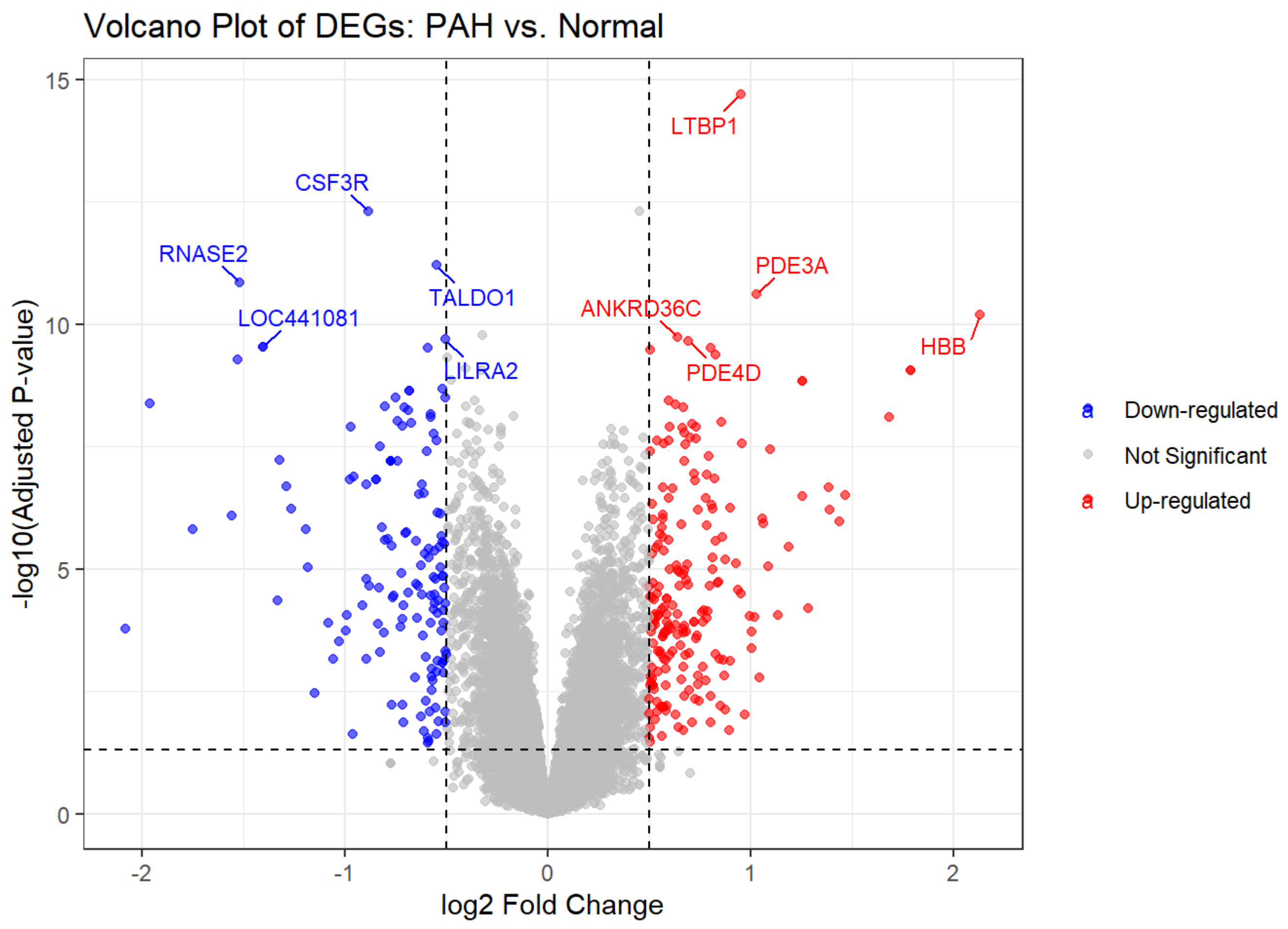
```
## Number of DEGs (adj.P.Val < 0.05 & |log2FC| > 0.5): 343
```

Visualize DEGs with a Volcano Plot and Heatmap

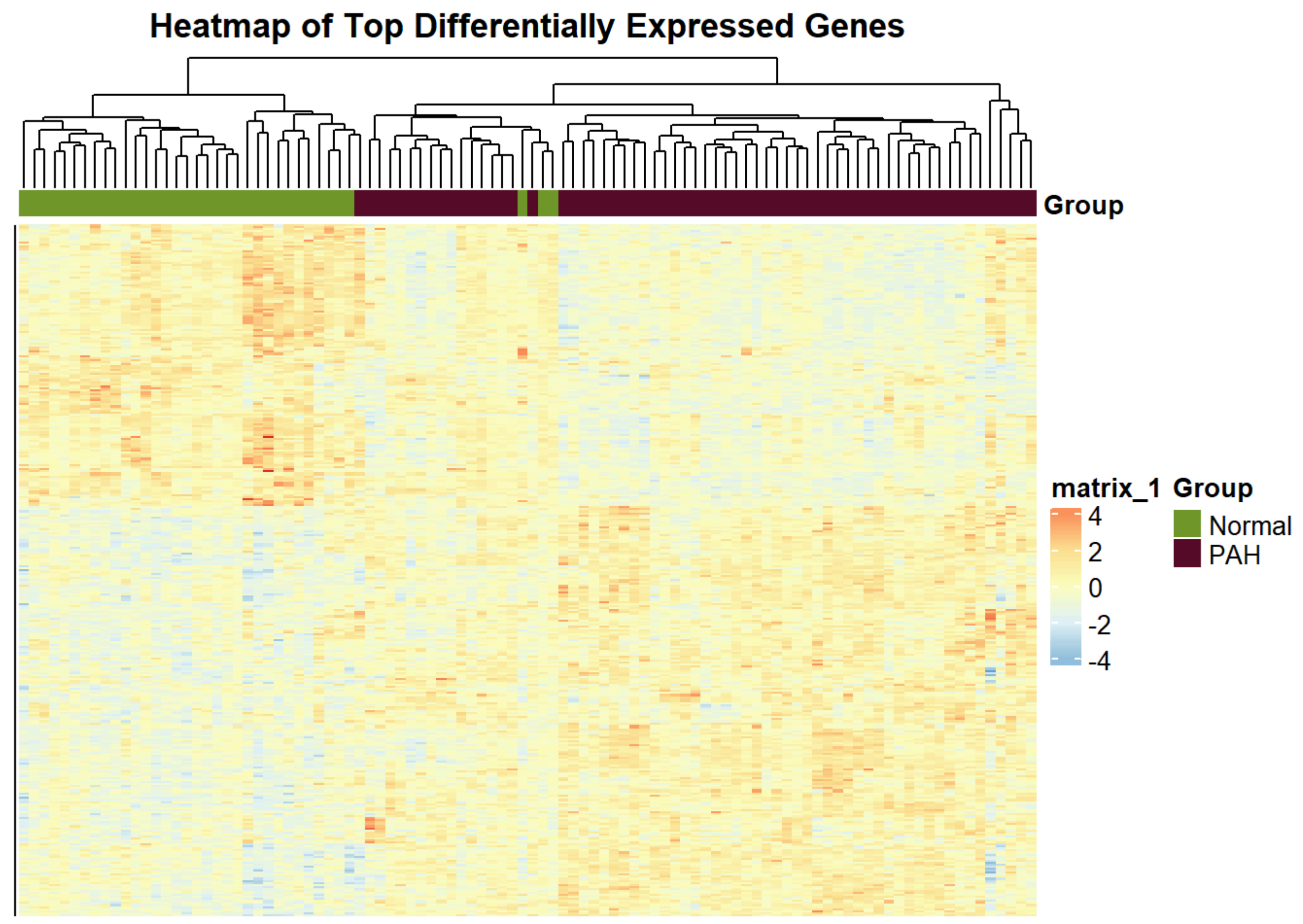
```
# We run topTable again, but without p-value or lfc cutoffs to get the full list.
all_genes <- topTable(fit, coef = "groupPAH", number = Inf, sort.by = "none")



# --- Part 1: Create the Volcano Plot with ggplot2 ---
# 1. Add a column to the 'all_genes' dataframe to identify significant genes
all_genes$significance <- "Not Significant"
# Mark up-regulated genes
all_genes$significance[all_genes$adj.P.Val < 0.05 & all_genes$logFC > 0.5] <- "Up-regulated"
# Mark down-regulated genes
all_genes$significance[all_genes$adj.P.Val < 0.05 & all_genes$logFC < -0.5] <- "Down-regulate
d"

# Check how many genes fall into each category
print(table(all_genes$significance))
```

```
##
##  Down-regulated Not Significant    Up-regulated
##             140           20834             203
```

```
# 2. Create the plot
volcano_plot <- ggplot(all_genes, aes(x = logFC, y = -log10(adj.P.Val), color = significanc
e)) +
  geom_point(alpha = 0.6, size = 1.5) +
  # Set custom colors
  scale_color_manual(values = c("Down-regulated" = "blue",
                                "Not Significant" = "grey",
                                "Up-regulated" = "red")) +
  # Add threshold lines
  geom_vline(xintercept = c(-0.5, 0.5), linetype = "dashed", color = "black") +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "black") +
  # Add titles and labels
  ggtitle("Volcano Plot of DEGs: PAH vs. Normal") +
  xlab("log2 Fold Change") +
  ylab("-log10(Adjusted P-value)") +
  # Apply a clean theme
  theme_bw() +
  theme(legend.title = element_blank()) # Remove legend title for a cleaner look

# Optional: Add labels for the top 10 most significant genes
top_labels <- subset(all_genes, adj.P.Val < 0.05 & abs(logFC) > 0.5)
top_labels <- top_labels[order(top_labels$adj.P.Val), ][1:10, ]

volcano_plot +
  geom_text_repel(data = top_labels, aes(label = rownames(top_labels)),
                  size = 3, box.padding = 0.5, max.overlaps = Inf)
```



```
# --- Part 2: Create the Heatmap with pheatmap ---


# 1. Get the expression data for only the significant DEGs
# The 'top_genes' object from the previous step already contains our DEGs
deg_expr_matrix <- combat_expr[rownames(top_genes), ]

# 2. Create an annotation data frame for the columns (samples)
# This tells pheatmap which samples belong to which group
annotation_col <- data.frame(
  Group = combined_pheno$group,
  row.names = rownames(combined_pheno)
)

# 3. Generate the heatmap
pheatmap(deg_expr_matrix,
         main = "Heatmap of Top Differentially Expressed Genes",
         annotation_col = annotation_col,
         scale = "row",          # This is VERY important - it scales each gene's expression
to have a mean of 0 and SD of 1 (Z-score)
         show_rownames = FALSE,  # Hides gene names, which would be too crow  ded
         show_colnames = FALSE,  # Hides sample names for a cleaner look
         cluster_cols = TRUE,    # Clusters the samples (columns) based on similarity
         cluster_rows = TRUE,    # Clusters the genes (rows) based on similarity
         treeheight_row = 0,     # Hides the row dendrogram for a cleaner look if desired
         legend = TRUE
)
```



Weighted Gene Co-expression Network Analysis (WGCNA)

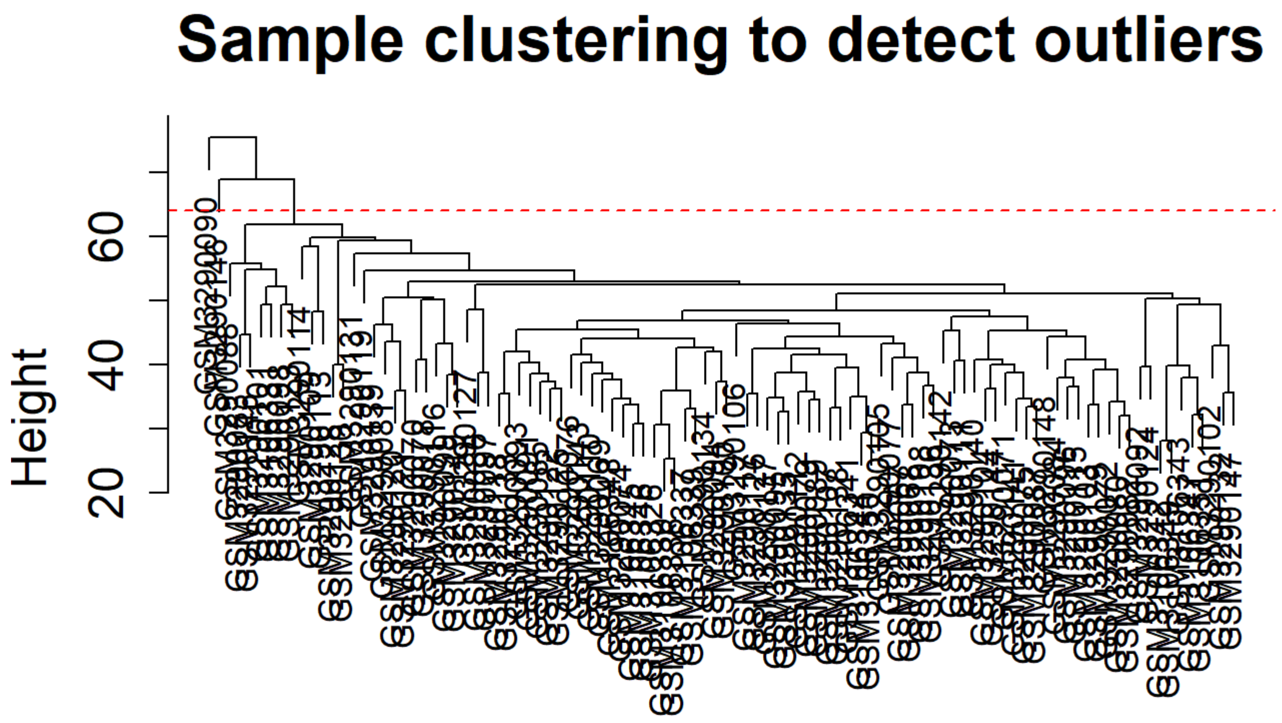
The goal is to find modules of co-expressed genes. - the WGCNA workflow: 1. **Input data preparation:** Select the top 25% most variable genes. 2. **Soft-thresholding power selection:** To achieve a scale-free topology. 3. **Network construction:** Create a topological overlap matrix (TOM). 4. **Module detection:** Identify gene modules using hierarchical clustering. 5. **Relate modules to traits:** Correlate module eigengenes with the disease state (PAH vs. normal).

Data Prep

```r
# WGCNA works with genes as columns and samples as rows,
datExpr0 <- as.data.frame(t(combat_expr))

# Calculate the variance of each gene
gene_variances <- apply(combat_expr, 1, var)

# Sort genes by variance in descending order and get the names of the top 25%
n_top_genes <- nrow(combat_expr) * 0.25
top_genes_wgcna <- names(sort(gene_variances, decreasing = TRUE))[1:n_top_genes]



# Filter the expression data to include only these top genes
datExpr <- datExpr0[, top_genes_wgcna]

#TODO burdan gsg diye bişey sildin sorun olursa bak

# WGCNA requires a check for genes and samples with too many missing values.
gsg <- goodSamplesGenes(datExpr, verbose = 3)
```

```
##  Flagging genes and samples with too many missing values...
##   ..step 1
```

```r
# If FALSE, remove the offending genes and samples.
if (!gsg$allOK) {
  datExpr <- datExpr[gsg$goodSamples, gsg$goodGenes]
}



#Check for sample outliers by clustering them
sampleTree <- hclust(dist(datExpr), method = "average")

plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="", cex.lab = 1.
5, cex.axis = 1.5, cex.main = 2)

# Set the height cut-off
cut_height <- 64
abline(h = cut_height, col = "red", lty = 2)
```



```
# Identify outlier samples
clust <- cutreeStatic(sampleTree, cutHeight = cut_height, minSize = 10)
table(clust)
```

```
## clust
##  0  1
##  2 98
```

```
# The outlier cluster is typically '0'
# Remove the outlier samples
keepSamples <- (clust != 0)
datExpr <- datExpr[keepSamples, ]
```

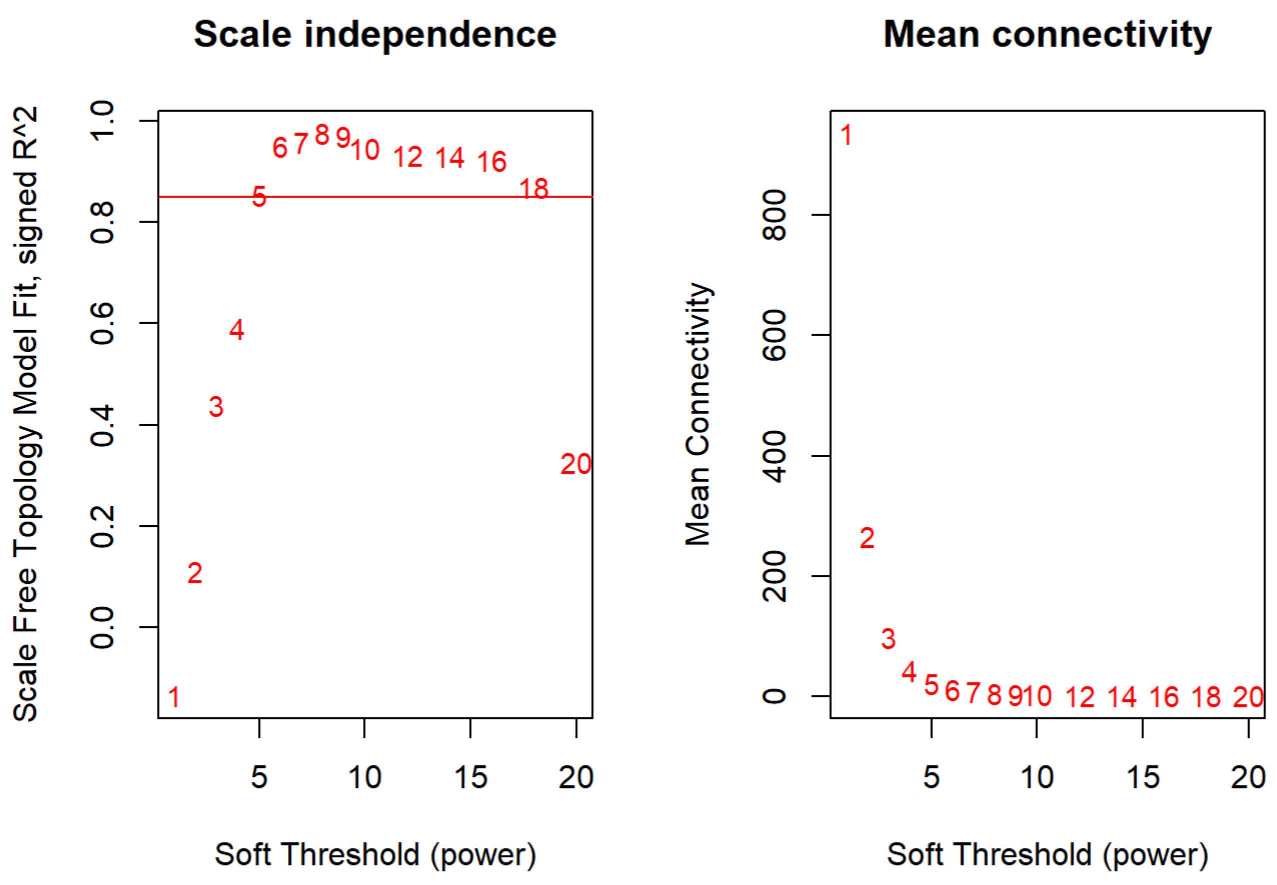
Soft-Thresholding Power Selection

This step helps us choose a power parameter (beta) to achieve a scale-free network topology.

```
powers <- c(c(1:10), seq(from = 12, to = 20, by = 2))

# Call the network topology analysis function
sft <- pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
```

```
## pickSoftThreshold: will use block size 5294.
##  pickSoftThreshold: calculating connectivity for given powers...
##    ..working on genes 1 through 5294 of 5294
##    Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
## 1      1    0.135  1.270          0.944 937.000  933.0000 1470.0
## 2      2    0.112 -0.694          0.946 266.000  255.0000  596.0
## 3      3    0.439 -1.370          0.958  97.200   88.8000  288.0
## 4      4    0.592 -1.500          0.944  42.400   35.7000  154.0
## 5      5    0.854 -1.670          0.992  21.200   16.0000  106.0
## 6      6    0.950 -1.890          0.975  11.800    7.8000   90.9
## 7      7    0.959 -1.870          0.951   7.240    4.0800   79.1
## 8      8    0.976 -1.750          0.969   4.800    2.3100   69.5
## 9      9    0.971 -1.650          0.964   3.410    1.3500   61.6
## 10    10    0.947 -1.560          0.936   2.560    0.8670   54.9
## 11    12    0.933 -1.430          0.915   1.640    0.3250   44.3
## 12    14    0.931 -1.340          0.914   1.200    0.1320   36.2
## 13    16    0.923 -1.270          0.912   0.944    0.0582   29.9
## 14    18    0.870 -1.330          0.834   0.789    0.0267   29.7
## 15    20    0.327 -2.030          0.235   0.686    0.0125   29.6
```

```
# Plot the results to help us choose the power
par(mfrow = c(1, 2))
cex1 <- 0.9

# Plot 1: Scale-free topology fit index as a function of the soft-thresholding power
plot(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
     xlab = "Soft Threshold (power)", ylab = "Scale Free Topology Model Fit, signed R^2",
     type = "n", main = "Scale independence")
text(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
     labels = powers, cex = cex1, col = "red")
# Add a horizontal line at the recommended R^2 cutoff of 0.85
abline(h = 0.85, col = "red")

# Plot 2: 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 = "Mean connectivity")
text(sft$fitIndices[, 1], sft$fitIndices[, 5], labels = powers, cex = cex1, col = "red")
```



Look at the “Scale independence” plot. We want to pick the LOWEST power where the red line crosses the R^2 threshold of 0.85.

```
chosen_power <- 6
```

Network Construction and Module Detection

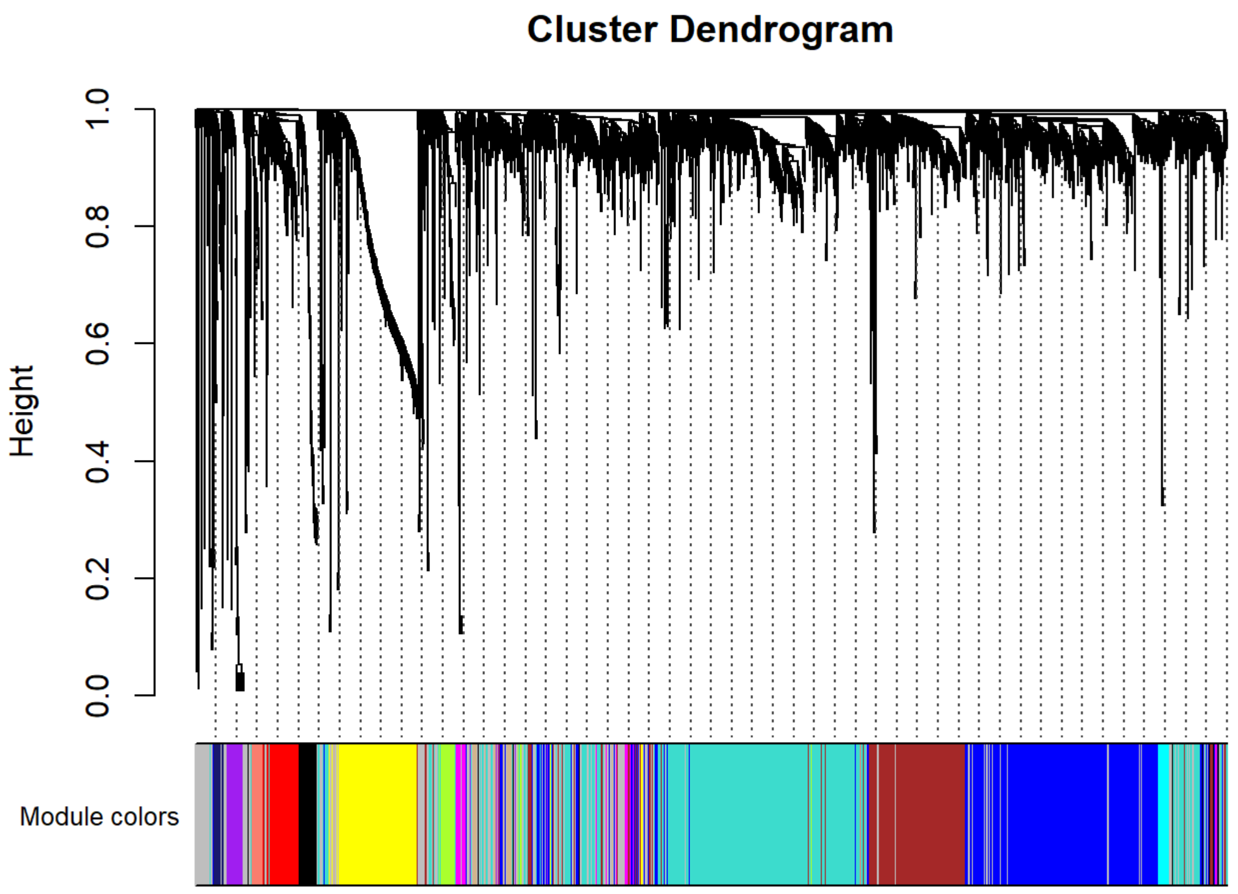
```
# We will use the one-step blockwiseModules function.

set.seed(123)
options(cor = WGCNA::cor)

net <- blockwiseModules(
  datExpr,
  power = chosen_power,
  TOMType = "signed",      # Used "signed" to distinguish between positive and negative
  minModuleSize = 20,      # Test this one
  reassignThreshold = 0,
  mergeCutHeight = 0.4,
  numericLabels = TRUE,      # Returns numeric labels for modules
  pamRespectsDendro = FALSE,
  saveTOMs = TRUE,
  saveTOMFileBase = "TOM",
  verbose = 3
)
```

```
##  Calculating module eigengenes block-wise from all genes
##    Flagging genes and samples with too many missing values...
##     ..step 1
##  ....pre-clustering genes to determine blocks..
##    Projective K-means:
##    ..k-means clustering..
##    ..merging smaller clusters...
## Block sizes:
## gBlocks
##    1    2
## 4633  661
##  ..Working on block 1 .
##     TOM calculation: adjacency..
##     ..will not use multithreading.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##    ..saving TOM for block 1 into file TOM-block.1.RData
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##      ..removing 145 genes from module 1 because their KME is too low.
##      ..removing 20 genes from module 2 because their KME is too low.
##      ..removing 14 genes from module 3 because their KME is too low.
##      ..removing 20 genes from module 4 because their KME is too low.
##      ..removing 36 genes from module 5 because their KME is too low.
##      ..removing 2 genes from module 6 because their KME is too low.
##      ..removing 1 genes from module 7 because their KME is too low.
##      ..removing 9 genes from module 8 because their KME is too low.
##      ..removing 1 genes from module 9 because their KME is too low.
##      ..removing 5 genes from module 14 because their KME is too low.
##  ..Working on block 2 .
##     TOM calculation: adjacency..
##     ..will not use multithreading.
##      Fraction of slow calculations: 0.000000
##     ..connectivity..
##     ..matrix multiplication (system BLAS)..
##     ..normalization..
##     ..done.
##    ..saving TOM for block 2 into file TOM-block.2.RData
##  ....clustering..
##  ....detecting modules..
##  ....calculating module eigengenes..
##  ....checking kME in modules..
##  ..merging modules that are too close..
##      mergeCloseModules: Merging modules whose distance is less than 0.4
##        Calculating new MEs...
```

```
table(net$colors)
```

```
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 1052 1183 1015  584  389  191  153  121  102   99   94   83   74   56   55   43
```

```
# Convert the numeric labels to colors
mergedColors <- labels2colors(net$colors)
# Plot the dendrogram and the module colors underneath
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                    "Module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
```



Relate Modules to Clinical Traits

The goal is to find which modules are significantly associated with PAH.

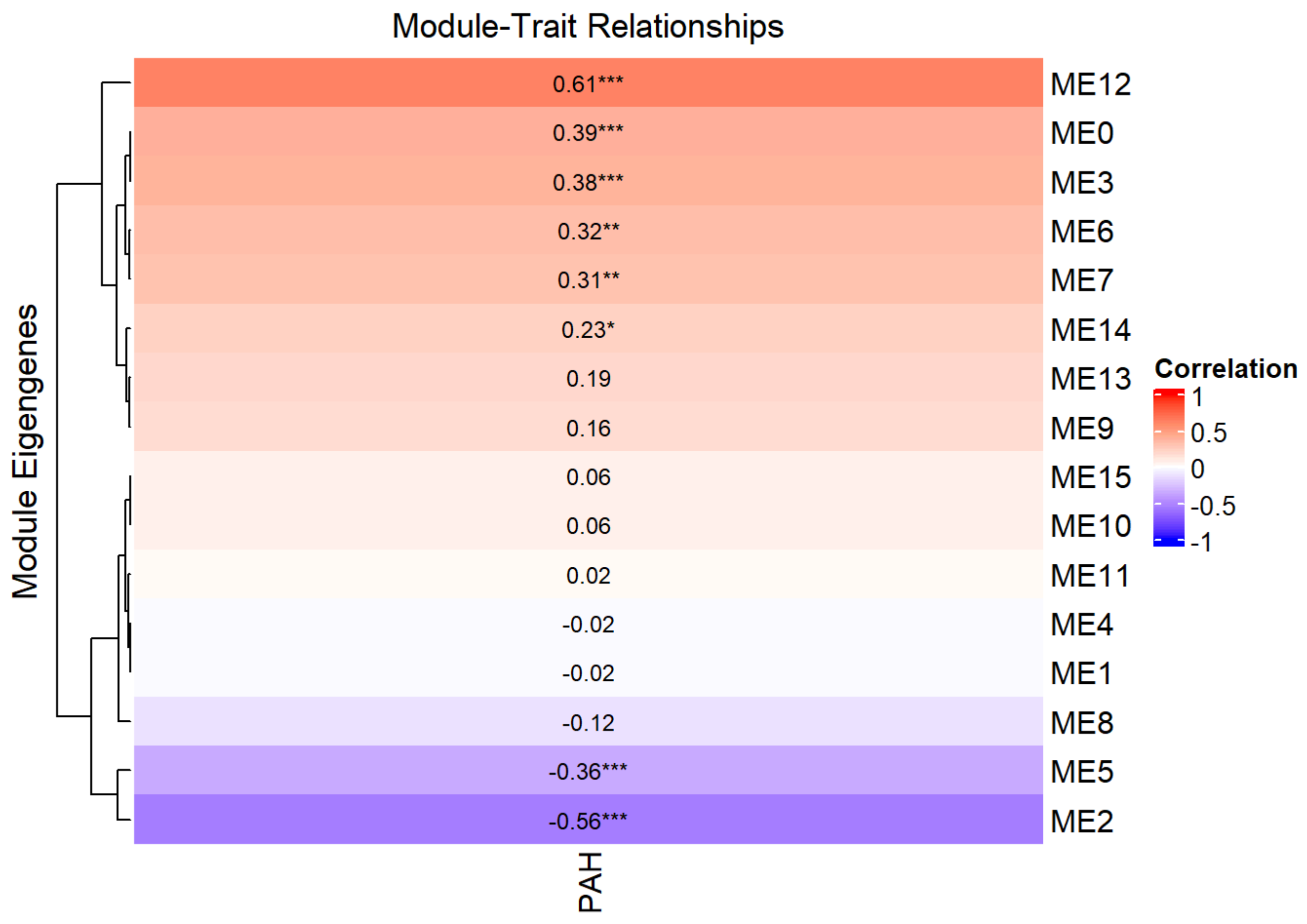
```r
# At this point, `datExpr` has already had outliers removed.
# We need to create a trait object that matches its dimensions.

# Start with the full phenotype data
allTraits <- combined_pheno

# Match the samples in the phenotype data to the samples that were kept in datExpr
# The rownames of datExpr contain the names of the samples we want to keep
traitRows <- match(rownames(datExpr), rownames(allTraits))
datTraits <- allTraits[traitRows, ]

# Sanity check: ensure the row names of expression and trait data are identical
stopifnot(all(rownames(datExpr) == rownames(datTraits)))
print(paste("Number of samples in datExpr:", nrow(datExpr)))
```

```
## [1] "Number of samples in datExpr: 98"
```

```r
print(paste("Number of samples in datTraits:", nrow(datTraits)))
```

```
## [1] "Number of samples in datTraits: 98"
```

```r
# Normal = 0, PAH = 1
# Create traitData from the filtered phenotype data
# Use datTraits, which is synchronized with datExpr
PAH_trait <- data.frame(PAH = ifelse(datTraits$group == "PAH", 1, 0))
rownames(PAH_trait) <- rownames(datTraits) # Assign the correct row names

# Module Eigengenes (MEs)
MEs0 <- moduleEigengenes(datExpr, net$colors)$eigengenes
MEs <- orderMEs(MEs0)

# Calculate Module-Trait Relationships
# This will now work because MEs (from datExpr) and PAH_trait (from datTraits) have the same
98 samples.
moduleTraitCor <- WGCNA::cor(MEs, PAH_trait, use = "pairwise.complete.obs")
moduleTraitPvalue <- corPvalueStudent(moduleTraitCor, nSamples = nrow(datExpr))



### Plotting
# Create a text matrix for annotations with significance stars
text_matrix <- ifelse(
  moduleTraitPvalue <= 0.001, paste0(round(moduleTraitCor, 2), "***"),
  ifelse(
    moduleTraitPvalue <= 0.01, paste0(round(moduleTraitCor, 2), "**"),
    ifelse(
      moduleTraitPvalue <= 0.05, paste0(round(moduleTraitCor, 2), "*"),
      round(moduleTraitCor, 2)
    )
  )
)

# Define the color palette
color_func <- colorRamp2(c(-1, 0, 1), c("blue", "white", "red"))

# Generate and display the heatmap
ht <- Heatmap(
  moduleTraitCor,
  name = "Correlation",
  col = color_func,
  cell_fun = function(j, i, x, y, width, height, fill) {
    grid.text(text_matrix[i, j], x, y, gp = gpar(fontsize = 9))
  },
  column_title = "Module-Trait Relationships",
  row_title = "Module Eigengenes",
  heatmap_legend_param = list(title = "Correlation")
)

draw(ht)
```



```
set.seed(NULL)
```

```
# --- Extract Most Significant Positive and Negative Module NUMBERS ---

# Find the row index of the maximum positive correlation
max_pos_cor_index <- which.max(moduleTraitCor[, "PAH"])
# Get the full module name (e.g., "ME4")
pos_mod_name <- rownames(moduleTraitCor)[max_pos_cor_index]

# Find the row index of the minimum (most negative) correlation
min_neg_cor_index <- which.min(moduleTraitCor[, "PAH"])
# Get the full module name (e.g., "ME17")
neg_mod_name <- rownames(moduleTraitCor)[min_neg_cor_index]

# Convert the module names to numeric variables by removing the "ME" prefix
positive_module <- as.numeric(gsub("ME", "", pos_mod_name))
negative_module <- as.numeric(gsub("ME", "", neg_mod_name))



# Print the final module numbers
cat("\n--- Most Significant Module Numbers ---\n",
    "Positive Module (highest positive correlation with PAH):", positive_module, "\n",
    "Negative Module (highest negative correlation with PAH):", negative_module, "\n")
```

```
##
## --- Most Significant Module Numbers ---
##  Positive Module (highest positive correlation with PAH): 12
##  Negative Module (highest negative correlation with PAH): 2
```

Identify Module DEGs

```
#top_genes
#
#module_assignments
#
##save module_assignments to txt
#write.table(module_assignments, file = "module_assignments.txt", sep = "\t", quote = FALSE,
col.names = NA)
```

```
degs <- rownames(top_genes)



# The module assignments for each gene from WGCNA.
# The gene names are the columns of datExpr, and the module numbers are in net$colors.
all_wgcna_genes <- colnames(datExpr)
module_assignments <- net$colors

# --- Step 2: Extract Genes from Your Significant Modules ---

# Get all genes belonging to the positively correlated module
genes_in_positive_module <- all_wgcna_genes[module_assignments == positive_module]

# Get all genes belonging to the negatively correlated module
genes_in_negative_module <- all_wgcna_genes[module_assignments == negative_module]

cat(
  "Total genes in Module", positive_module, "(positive correlation):", length(genes_in_positi
ve_module), "\n",
  "Total genes in Module", negative_module, "(negative correlation):", length(genes_in_negati
ve_module), "\n",
  "Total number of DEGs:", length(degs), "\n\n"
)
```

```
## Total genes in Module 12 (positive correlation): 74
##  Total genes in Module 2 (negative correlation): 1015
##  Total number of DEGs: 343
```

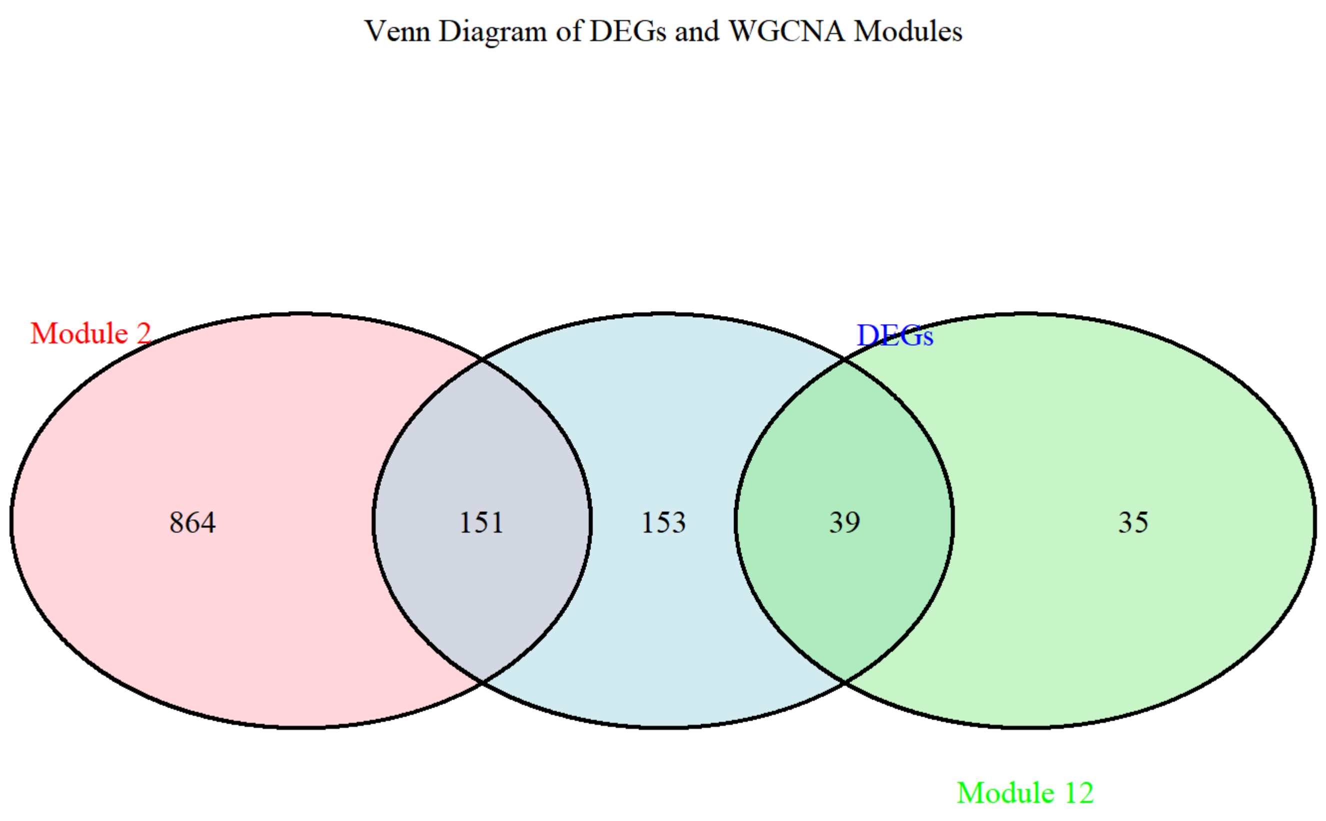
```r
# --- Step 3: Perform the Intersection ---

# Find the intersection of DEGs and the genes in the positive module
mdegs_positive_module <- base::intersect(genes_in_positive_module, degs)

# Find the intersection of DEGs and the genes in the negative module
mdegs_negative_module <- base::intersect(genes_in_negative_module, degs)

# Combine them into one final list of high-confidence genes
final_mdeg_list <- c(mdegs_positive_module, mdegs_negative_module)



# --- Step 4: Review the Results ---

cat(
  "Number of MDEGs from Module", positive_module, ":", length(mdegs_positive_module), "\n",
  "Number of MDEGs from Module", negative_module, ":", length(mdegs_negative_module), "\n",
  "Total number of final MDEGs for functional analysis:", length(final_mdeg_list), "\n\n",
  "First 10 MDEGs:\n"
)
```

```
## Number of MDEGs from Module 12 : 39
##  Number of MDEGs from Module 2 : 151
##  Total number of final MDEGs for functional analysis: 190
##
##  First 10 MDEGs:
```

```r
print(head(final_mdeg_list, 10))
```

```
##  [1] "THY1"         "CLEC4E"       "SLCO4A1"      "COL6A6"       "TM4SF18"
##  [6] "FCN3"         "CPA3"         "SFRP4"        "COL14A1"      "LOC105375730"
```

```r
venn.plot <- venn.diagram(
  x = list(
    DEGs = degs,
    Module_Positive = genes_in_positive_module,
    Module_Negative = genes_in_negative_module
  ),
  category.names = c("DEGs", paste("Module", positive_module), paste("Module", negative_modul
e)),
  filename = NULL, # Set to NULL to plot directly in R
  output = FALSE,
  fill = c("lightblue", "lightgreen", "lightpink"),
  alpha = 0.5,
  cat.col = c("blue", "green", "red"),
  main = "Venn Diagram of DEGs and WGCNA Modules"
)
grid.newpage()
grid.draw(venn.plot)
```



```
#filter the final_mdeg_list to only include genes of p<0.05 and |log2FC| > 1

final_mdeg_list_filtered <- final_mdeg_list[final_mdeg_list %in% rownames(top_genes[top_genes
$adj.P.Val < 0.05 & abs(top_genes$logFC) > 1, ])]

cat("Number of MDEGs with adj.P.Val < 0.05 and |log2FC| > 1:", length(final_mdeg_list_filtere
d), "\n")
```

```
## Number of MDEGs with adj.P.Val < 0.05 and |log2FC| > 1: 20
```

```
final_mdeg_list_filtered
```

```
##  [1] "THY1"         "COL14A1"      "LOC105375730" "IL1R2"        "SAA2"
##  [6] "S100A12"      "ASPN"         "WIF1"         "AQP9"         "S100A9"
## [11] "SAA1"         "S100A8"       "MGAM"         "LCN2"         "RNASE2"
## [16] "LOC441081"    "LOC728488"    "AGBL1"        "OGN"          "PDE3A"
```

Functional Analysis and Network Construction

GO analysis

```
# Load the required libraries


# 1. Convert Gene Symbols to Entrez IDs
gene_map <- bitr(final_mdeg_list_filtered,
                 fromType = "SYMBOL",
                 toType = "ENTREZID",
                 OrgDb = org.Hs.eg.db)
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
entrez_ids <- gene_map$ENTREZID
cat("Number of MDEGs successfully mapped to Entrez IDs:", length(entrez_ids), "\n")
```

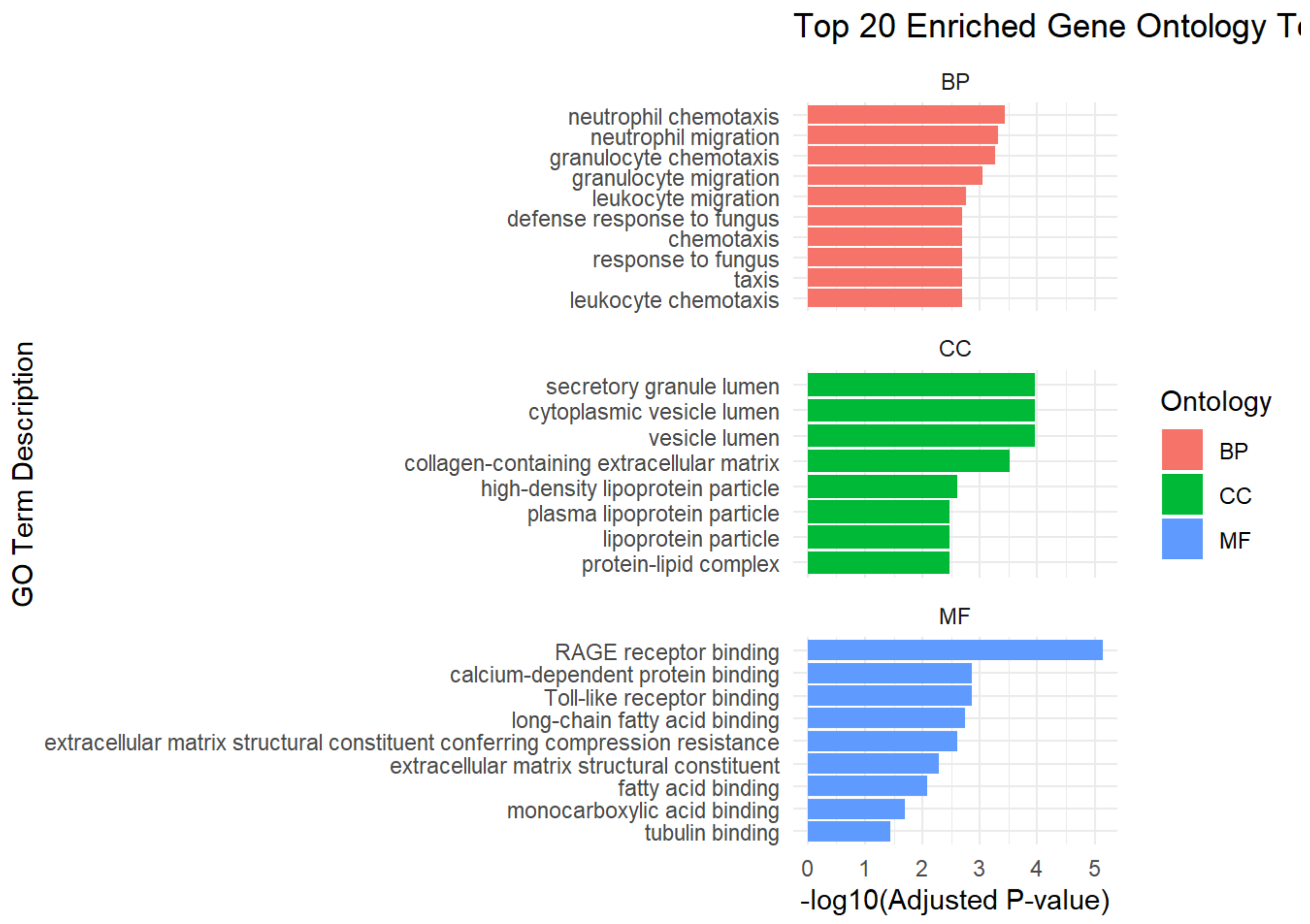
```
## Number of MDEGs successfully mapped to Entrez IDs: 20
```

```
mapped_gene_symbols <- final_mdeg_list_filtered[final_mdeg_list_filtered %in% gene_map$SYMBO
L]
cat("Number of MDEGs successfully mapped to Gene Symbols:", length(mapped_gene_symbols), "\n
\n")
```

```
## Number of MDEGs successfully mapped to Gene Symbols: 20
```

```
# 2. Perform GO Enrichment Analysis
# For the enrichment itself, we use the general Human OrgDb
go_results <- enrichGO(gene = entrez_ids,
                       OrgDb = org.Hs.eg.db,
                       keyType = "ENTREZID",
                       ont = "ALL",
                       pAdjustMethod = "BH",
                       pvalueCutoff = 0.05)
```

```r
go_df <- as.data.frame(go_results)

# Split the GO results by ontology
go_bp <- go_df[go_df$ONTOLOGY == "BP", ]
go_mf <- go_df[go_df$ONTOLOGY == "MF", ]
go_cc <- go_df[go_df$ONTOLOGY == "CC", ]

# Take the top N (e.g., 10) significant terms from each ontology
top_bp <- head(go_bp[order(go_bp$p.adjust), ], 10)
top_mf <- head(go_mf[order(go_mf$p.adjust), ], 10)
top_cc <- head(go_cc[order(go_cc$p.adjust), ], 10)

# Combine the results
top_go_combined <- rbind(top_bp, top_mf, top_cc)

# Reorder the descriptions for plotting
top_go_combined$Description <- factor(top_go_combined$Description, levels = rev(top_go_combin
ed$Description))

# Add a column for -log10 adjusted p-value
top_go_combined$negLogAdjP <- -log10(top_go_combined$p.adjust)


# Plot
p <- ggplot(top_go_combined, aes(x = negLogAdjP, y = Description, fill = ONTOLOGY)) +
  # Add the bars to the plot
  geom_col() +
  # Add a title and axis labels
  labs(
    title = "Top 20 Enriched Gene Ontology Terms by Ontology",
    x = "-log10(Adjusted P-value)",
    y = "GO Term Description",
    fill = "Ontology" # Label for the legend
  ) +
  # Group terms by ontology
  facet_wrap(~ ONTOLOGY, scales = "free_y", ncol = 1) +
  theme_minimal()

# Print the plot
print(p)
```



KEGG Pathway Analysis

```
# 4. Perform KEGG Pathway Analysis
kegg_results <- enrichKEGG(gene = entrez_ids,
                           organism = 'hsa', # 'hsa' is the code for Homo sapiens
                           pvalueCutoff = 0.05)
```

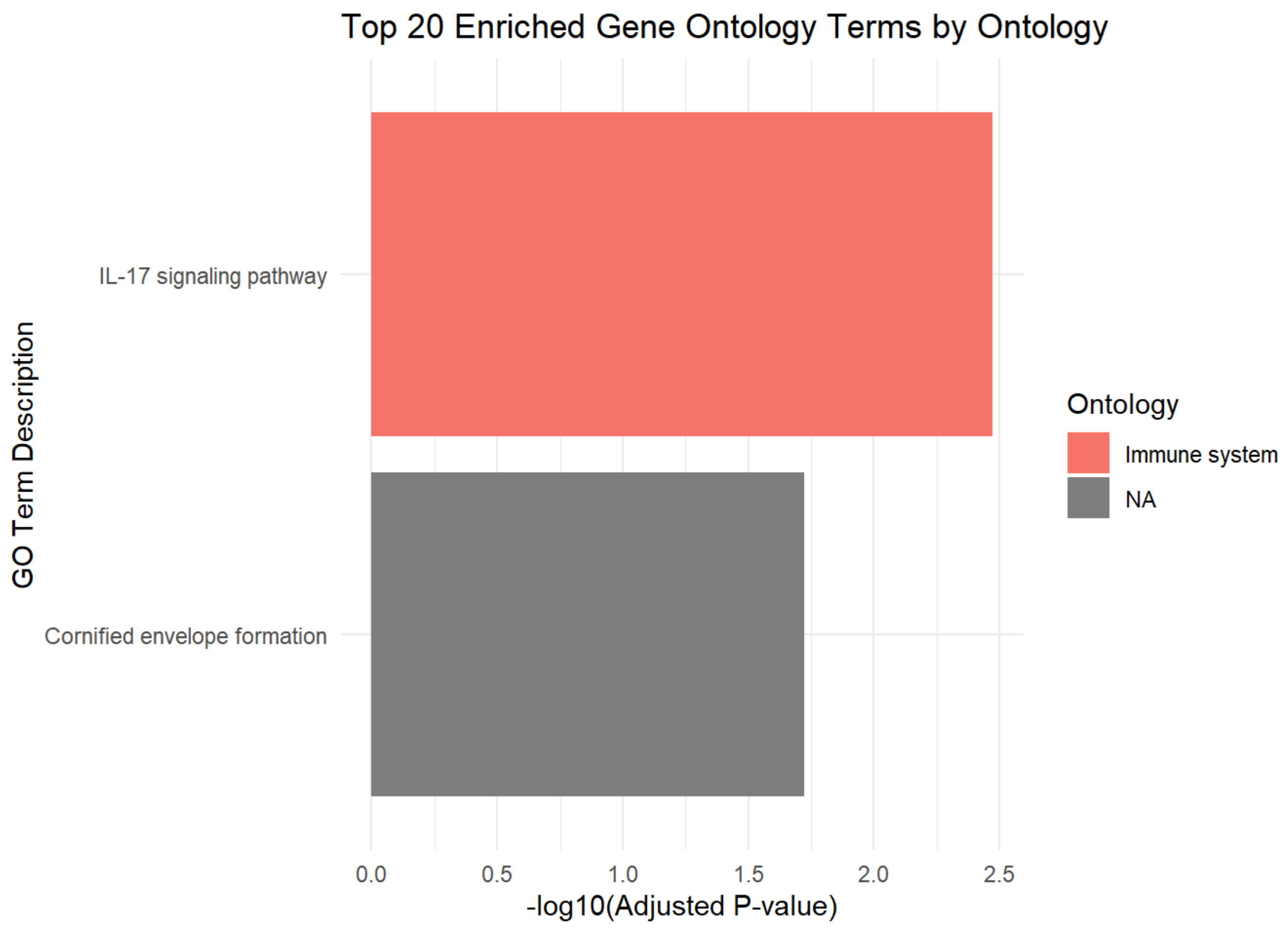
```
## Reading KEGG annotation online: "https://rest.kegg.jp/link/hsa/pathway"...
```

```
## Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/hsa"...
```

```
kegg_results_df <- as.data.frame(kegg_results)
kegg_results_df$negLogAdjP <- -log10(kegg_results_df$p.adjust)
kegg_results_df
```

```
##                      category    subcategory       ID                 Description
## hsa04657 Organismal Systems Immune system hsa04657      IL-17 signaling pathway
## hsa04382               <NA>          <NA> hsa04382 Cornified envelope formation
##          GeneRatio  BgRatio RichFactor FoldEnrichment   zScore      pvalue
## hsa04657      3/11  95/9446 0.03157895       27.11770 8.735628 0.000153380
## hsa04382      3/11 217/9446 0.01382488       11.87181 5.531980 0.001721458
##            p.adjust      qvalue         geneID Count negLogAdjP
## hsa04657 0.00337436 0.002906147 6280/6279/3934     3   2.471809
## hsa04382 0.01893604 0.016308550 6283/6280/6279     3   1.722711
```

```
# Plot
p <- ggplot(kegg_results_df, aes(x = negLogAdjP, y = reorder(Description, negLogAdjP), fill =
subcategory)) +
  # Add the bars to the plot
  geom_col() +
  # Add a title and axis labels
  labs(
    title = "Top 20 Enriched Gene Ontology Terms by Ontology",
    x = "-log10(Adjusted P-value)",
    y = "GO Term Description",
    fill = "Ontology" # Label for the legend
  ) +
  theme_minimal()

# Print the plot
print(p)
```



```
ggsave("KEGG_dotplot2.png", plot = p, width = 8, height = 6, dpi = 300)
```

```
#emap_kegg_results <- pairwise_termsim(kegg_results)
#kegg3 <- emapplot(emap_kegg_results, showCategory = 30, layout = "kk")
#kegg3
#ggsave("KEGG_emapplot.png", plot = kegg3, width = 8, height = 6, dpi = 300)
#
## 4. Cnetplot
#kegg4 <- cnetplot(kegg_results, categorySize = "pvalue", foldChange = NULL)
#kegg4
#ggsave("KEGG_cnetplot.png", plot = kegg4, width = 8, height = 6, dpi = 300)
```

Protein-Protein Interaction (PPI) Network with STRING

```
# 1. Initialize STRING database object
string_db <- STRINGdb$new(version = "11.5", species = 9606, score_threshold = 400) # 9606 is
the taxon ID for human

# 2. Map your gene symbols to STRING identifiers
mapped_genes <- string_db$map(data.frame(gene = mapped_gene_symbols), "gene", removeUnmappedR
ows = TRUE)
```

```
## Warning:  we couldn't map to STRING 15% of your identifiers
```

```
# --- Part 3: Identify Hub Genes using Network Centrality ---

# --- Part 3: Create Graph and Add All Necessary Attributes ---
conflicts_prefer(base::as.factor)
```

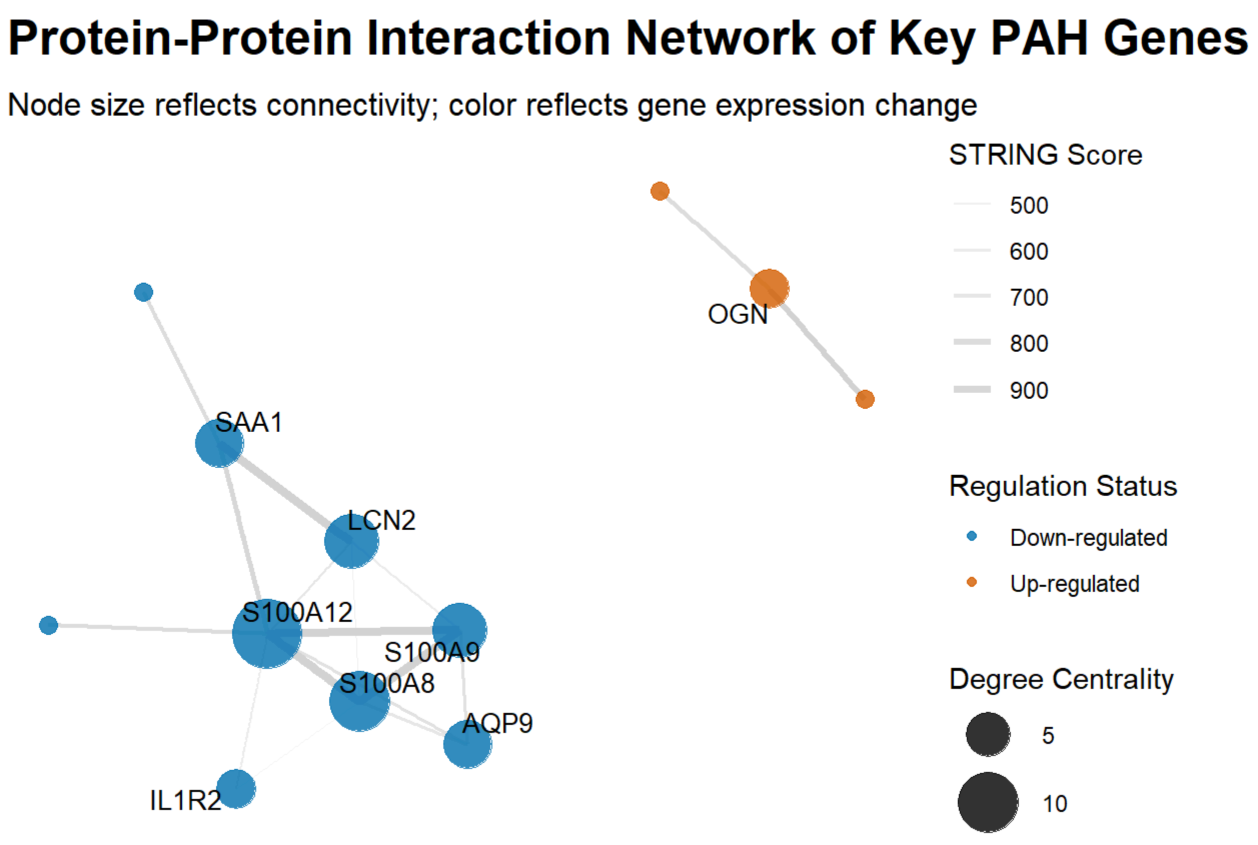
```
## [conflicted] Removing existing preference.
## [conflicted] Will prefer base::as.factor over any other package.
```

```r
if (nrow(mapped_genes) > 0) {

  # 1. Get interactions from STRING. This includes 'combined_score'.
  interactions <- string_db$get_interactions(mapped_genes$STRING_id)

  # 2. FIX: Create igraph object from the *full* interactions data frame.
  # This automatically adds 'combined_score' as an edge attribute.
  ppi_graph <- graph_from_data_frame(d = interactions, directed = FALSE)

  # 3. Calculate degree centrality for each node.
  degree_centrality <- igraph::degree(ppi_graph)

  # 4. Create a comprehensive data frame for node attributes.
  # This will also serve as the 'network_nodes' object for the bar chart.
  network_nodes <- data.frame(
    STRING_id = V(ppi_graph)$name,
    stringsAsFactors = FALSE
  )

  # 5. Map STRING IDs back to gene symbols and add degree.
  symbol_map <- setNames(mapped_genes$gene, mapped_genes$STRING_id)
  network_nodes$symbol <- symbol_map[network_nodes$STRING_id]
  network_nodes$degree <- degree_centrality[network_nodes$STRING_id]

  # 6. Add regulation status from your earlier DEG analysis ('all_genes' table).
  regulation_status <- all_genes %>%
    dplyr::select(logFC, adj.P.Val) %>%
    mutate(
      regulation = case_when(
        adj.P.Val < 0.05 & logFC > 0.5  ~ "Up-regulated",
        adj.P.Val < 0.05 & logFC < -0.5 ~ "Down-regulated",
        TRUE                           ~ "Neutral"
      ),
      symbol = rownames(.) # Get gene symbols from row names
    )

  # Merge regulation status into the nodes data frame.
  network_nodes <- merge(network_nodes, regulation_status[, c("symbol", "regulation")], by =
"symbol", all.x = TRUE)
  network_nodes$regulation[is.na(network_nodes$regulation)] <- "Neutral"

  # 7. Perform community detection for the advanced plot.
  communities <- cluster_walktrap(ppi_graph)
  network_nodes$community <- communities$membership[match(network_nodes$STRING_id, names(comm
unities$membership))]

  # 8. Add all created attributes directly to the graph object vertices.
  # 'ggraph' will now be able to find 'degree', 'regulation', 'symbol', and 'community'.
  V(ppi_graph)$symbol     <- network_nodes$symbol[match(V(ppi_graph)$name, network_nodes$STRI
NG_id)]
  V(ppi_graph)$degree      <- network_nodes$degree[match(V(ppi_graph)$name, network_nodes$STR
ING_id)]
  V(ppi_graph)$regulation  <- network_nodes$regulation[match(V(ppi_graph)$name, network_nodes
$STRING_id)]
  V(ppi_graph)$community   <- as.factor(network_nodes$community[match(V(ppi_graph)$name, netw
ork_nodes$STRING_id)])

  # 9. Sort by degree and print the top 10 hub genes.
  hub_genes <- network_nodes %>%
    arrange(desc(degree)) %>%
    head(10)

  cat("\n--- Top 10 Hub Genes (by Degree Centrality) ---\n")
  print(hub_genes[, c("symbol", "degree", "regulation")])
}
```

```
##
## --- Top 10 Hub Genes (by Degree Centrality) ---
##     symbol degree     regulation
## 1  S100A12     14 Down-regulated
## 2   S100A8     10 Down-regulated
## 3     LCN2      8 Down-regulated
## 4   S100A9      8 Down-regulated
## 5     AQP9      6 Down-regulated
## 6     SAA1      6 Down-regulated
## 7    IL1R2      4 Down-regulated
## 8      OGN      4   Up-regulated
## 9     ASPN      2   Up-regulated
## 10 COL14A1      2   Up-regulated
```

```r
# --- Visualization 1: Enhanced PPI Network ---


if (exists("ppi_graph") && vcount(ppi_graph) > 0) {

  ggraph(ppi_graph, layout = "fr") +
    # Edges: Thinner and more transparent for weaker interactions
    geom_edge_link(aes(alpha = combined_score, width = combined_score), colour = "lightgrey")
+

    # Nodes: Size mapped to degree, color mapped to regulation
    geom_node_point(aes(size = degree, color = regulation), alpha = 0.8) +

    # Labels: Add gene symbols, repelled to avoid overlap.
    # We only label the more important nodes (degree > X) to keep it clean.
    geom_node_text(aes(label = ifelse(degree > 3, symbol, NA)),
                   repel = TRUE,
                   size = 3.5,
                   color = "black") +

    # Aesthetics and Scales
    scale_edge_width_continuous(range = c(0.2, 1.5), name = "STRING Score") +
    scale_edge_alpha_continuous(range = c(0.1, 1), name = "STRING Score") +
    scale_size_continuous(range = c(3, 12), name = "Degree Centrality") +
    scale_color_manual(values = c("Up-regulated" = "#D55E00",
                                  "Down-regulated" = "#0072B2",
                                  "Neutral" = "grey80"),
                       name = "Regulation Status") +

    # Theme and Titles
    theme_graph(base_family = 'sans') +
    labs(
      title = "Protein-Protein Interaction Network of Key PAH Genes",
      subtitle = "Node size reflects connectivity; color reflects gene expression change"
    )
}
```

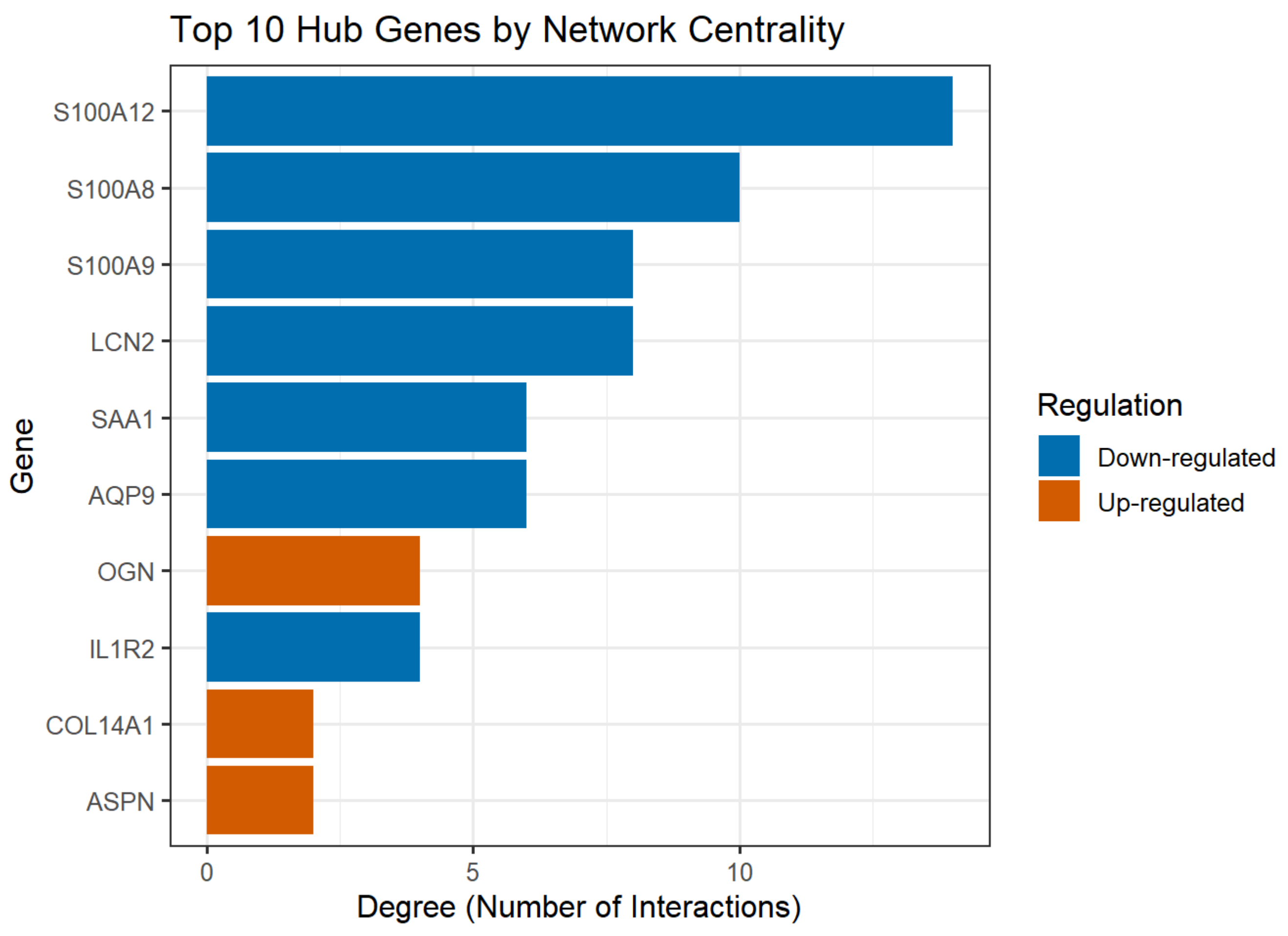
```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).
```



```
#save the plot
ggsave("PPI_Network_Enhanced.png", width = 12, height = 8, dpi = 300)
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_text_repel()`).
```

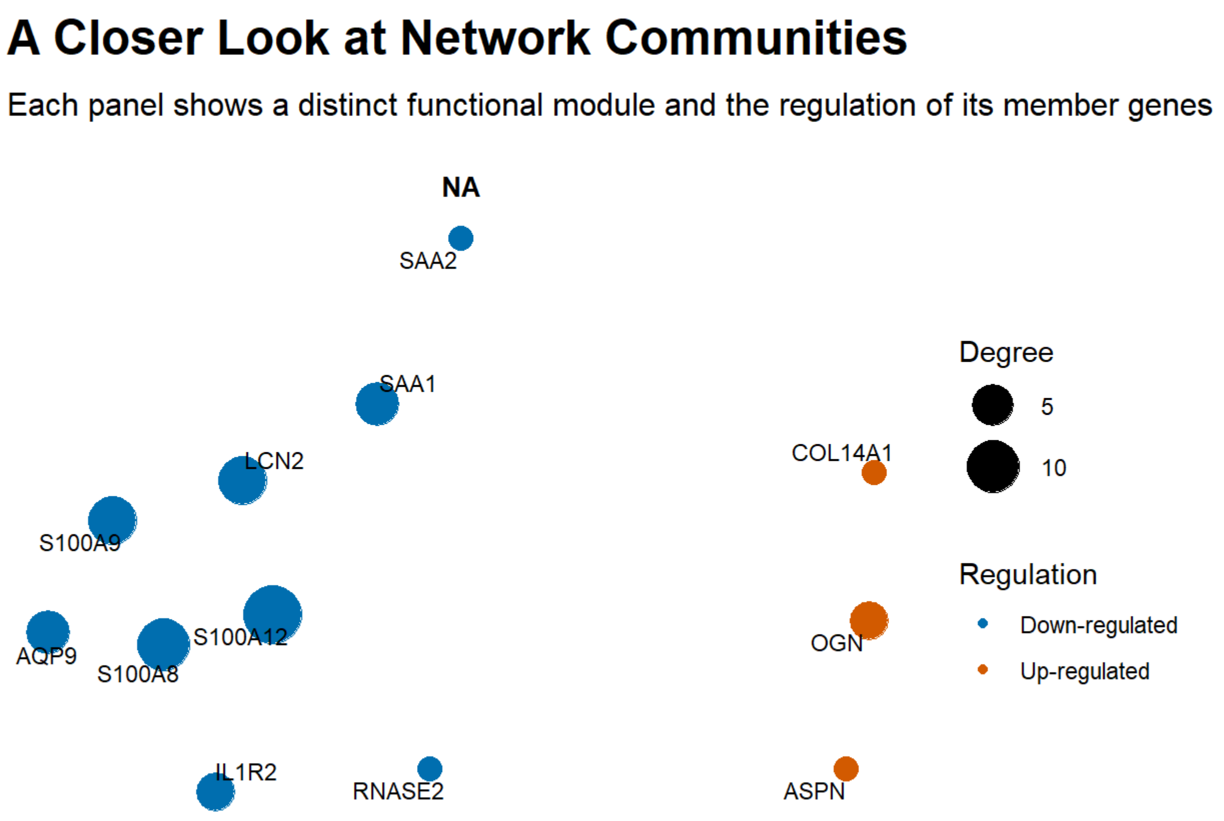
```
# --- Visualization 2: Bar Chart of Top Hub Genes ---

if (exists("network_nodes") && nrow(network_nodes) > 0) {

  # Get the top 10 genes by degree
  top_hubs <- network_nodes %>%
    arrange(desc(degree)) %>%
    head(10)

  ggplot(top_hubs, aes(x = reorder(symbol, degree), y = degree, fill = regulation)) +
    geom_col() +
    coord_flip() + # Flips axes to make labels readable
    scale_fill_manual(values = c("Up-regulated" = "#D55E00",
                                 "Down-regulated" = "#0072B2",
                                 "Neutral" = "grey80"),
                      name = "Regulation") +
    theme_bw(base_size = 12) +
    labs(
      title = "Top 10 Hub Genes by Network Centrality",
      x = "Gene",
      y = "Degree (Number of Interactions)"
    )
}
```



```
#save the plot
ggsave("Top10_Hub_Genes_Barplot.png", width = 8, height = 6, dpi = 300)
```

```r
# --- ADVANCED Hybrid Plot: Faceting by Community ---

if (exists("ppi_graph") && "community" %in% vertex_attr_names(ppi_graph)) {

  ggraph(ppi_graph, layout = 'fr') +
    # Use faceting to create a subplot for each community
    facet_nodes(~community, scales = "free") +

    # Edges within each subplot
    geom_edge_link(alpha = 0.5, width = 0.7, colour = "darkgrey") +

    # Nodes: Color is now back to Regulation, size is degree
    geom_node_point(aes(color = regulation, size = degree)) +

    # Labels for all nodes within their smaller plots
    geom_node_text(aes(label = symbol), repel = TRUE, size = 3) +

    # Aesthetics and scales
    scale_color_manual(values = c("Up-regulated" = "#D55E00",
                                  "Down-regulated" = "#0072B2",
                                  "Neutral" = "grey80"),
                       name = "Regulation") +
    scale_size_continuous(range = c(4, 10), name = "Degree") +
    theme_graph(base_family = 'sans', border = TRUE) +
    labs(
      title = "A Closer Look at Network Communities",
      subtitle = "Each panel shows a distinct functional module and the regulation of its mem
ber genes"
    )
}
```

```
## Warning: Unknown or uninitialised column: `ggraph_index`.
```



```
#save the plot
ggsave("PPI_Community_Facet_Plot.png", width = 12, height = 8, dpi = 300)
```

```
## Warning: Unknown or uninitialised column: `ggraph_index`.
```