```
################################################################################
### TITLE: Integrated HYBRID Analysis Pipeline for GSE65194
### SUBTITLE: Limma DGE, Pathway Enrichment, STATIC (PNG) & INTERACTIVE (HTML)
Plots
###
### DATASET:    GSE65194 (Affymetrix Human Genome U133 Plus 2.0 Array)
### COMPARISON: HER2 Positive vs. Triple Negative Breast Cancer (TNBC)
### OUTPUT:     D:/DOWNLOADS/GSE65194_Results
################################################################################

# ==============================================================================
# SECTION 1: SYSTEM SETUP & DIRECTORY MANAGEMENT
# ==============================================================================

# 1.1 Define Output Path
base_path <- "D:/DOWNLOADS"
output_dir <- file.path(base_path, "GSE65194_Results")
gse_id <- "GSE65194"

# 1.2 Create the Directory
if (!dir.exists(output_dir)) {
  dir.create(output_dir, recursive = TRUE)
}
cat(">>> [SETUP] All files will be saved to:", output_dir, "\n")

# 1.3 Load and Install Required Libraries
cat(">>> [SETUP] Checking and loading libraries...\n")

cran_packages <- c("ggplot2", "dplyr", "plotly", "DT", "heatmaply",
"htmlwidgets", "pheatmap", "ggrepel")
bioc_packages <- c("GEOquery", "limma", "Biobase", "hgu133plus2.db",
"clusterProfiler", "org.Hs.eg.db", "ReactomePA", "enrichplot",
"EnhancedVolcano")

suppressPackageStartupMessages({
  if (!require("BiocManager", quietly = TRUE)) install.packages("BiocManager")

  # Install CRAN packages
  for (pkg in cran_packages) {
    if (!require(pkg, character.only = TRUE)) install.packages(pkg)
    library(pkg, character.only = TRUE)
  }

  # Install Bioconductor packages
  for (pkg in bioc_packages) {
    if (!require(pkg, character.only = TRUE)) BiocManager::install(pkg, update =
FALSE, ask = FALSE)
    library(pkg, character.only = TRUE)
  }
})
cat(">>> [SETUP] Libraries loaded successfully.\n")


# ==============================================================================
# SECTION 2: DATA LOADING & PRE-PROCESSING (GSE65194)
# ==============================================================================

cat(paste(">>> [DATA] Downloading", gse_id, "from GEO...\n"))

# 2.1 Load Data
# We keep the original 'gse_data' object safe so we can always reset if needed
gse <- getGEO(gse_id, GSEMatrix = TRUE)
gse_data_original <- gse[[1]]
```

```
# 2.2 SMART FILTERING: Select Only HER2 and TNBC Samples
cat(">>> [DATA] Filtering for HER2 and TNBC samples...\n")

# --- RESET DATA ---
expression_data <- exprs(gse_data_original)
meta_data <- pData(gse_data_original)

# --- VALIDATED FILTERING LOGIC ---
# Search ONLY specific columns to avoid "cell line" false positives
target_cols <- grep("title|source_name|characteristics", colnames(meta_data),
value = TRUE, ignore.case = TRUE)
sample_info <- apply(meta_data[, target_cols, drop = FALSE], 1, paste, collapse
= " ")

# Identify indices
is_cell_line <- grepl("cell line", sample_info, ignore.case = TRUE)
is_her2 <- grepl("Her2", sample_info, ignore.case = TRUE) & !is_cell_line
is_tnbc <- grepl("TNBC", sample_info, ignore.case = TRUE) & !is_cell_line

keep_indices <- which(is_her2 | is_tnbc)

if (length(keep_indices) == 0) {
  stop("CRITICAL ERROR: No samples found. Check filtering logic.")
}

# Subset the data
expression_data <- expression_data[, keep_indices]
meta_data_subset <- meta_data[keep_indices, ]

cat(paste("    -> Kept", length(keep_indices), "samples (HER2 + TNBC).\n"))

# 2.3 Create Group Factor
subset_info <- apply(meta_data_subset[, target_cols, drop=FALSE], 1, paste,
collapse = " ")
group_labels <- ifelse(grepl("Her2", subset_info, ignore.case = TRUE), "HER2",
"TNBC")

subtype_factor <- factor(group_labels, levels = c("TNBC", "HER2")) # TNBC is
Reference
cat("    -> Sample Distribution:\n")
print(table(subtype_factor))

# 2.4 Normalization & Filtering
cat(">>> [DATA] Normalizing and filtering...\n")

# Quantile Normalization
exprs_norm <- normalizeBetweenArrays(expression_data, method = "quantile")

# Filter low expression probes (Remove bottom 25% of intensity)
means <- rowMeans(exprs_norm)
cutoff <- quantile(means, 0.25)
exprs_filt <- exprs_norm[means > cutoff, ]

cat("    -> Final Dimensions:", dim(exprs_filt)[1], "Genes x", dim(exprs_filt)
[2], "Samples\n")


# ==============================================================================
# SECTION 3: DIFFERENTIAL EXPRESSION ANALYSIS (LIMMA)
# ==============================================================================

cat(">>> [ANALYSIS] Running Limma Differential Expression...\n")

# 3.1 Linear Modeling
```

```
design <- model.matrix(~0 + subtype_factor)
colnames(design) <- levels(subtype_factor)

# Contrast: HER2 minus TNBC
cont.matrix <- makeContrasts(Contrast = HER2 - TNBC, levels = design)

fit <- lmFit(exprs_filt, design)
fit2 <- contrasts.fit(fit, cont.matrix)
fit2 <- eBayes(fit2)

# 3.2 Extract Results
results <- topTable(fit2, adjust = "fdr", number = Inf)

# 3.3 Annotate Probes (ProbeID -> Gene Symbol)
cat(">>> [ANNOTATION] Mapping Probes to Gene Symbols...\n")
probes <- rownames(results)
results$Symbol <- mapIds(hgu133plus2.db, keys = probes, column = "SYMBOL",
keytype = "PROBEID", multiVals = "first")
results$Entrez <- mapIds(hgu133plus2.db, keys = probes, column = "ENTREZID",
keytype = "PROBEID", multiVals = "first")

# Remove probes that did not map to a gene
results_clean <- results[!is.na(results$Symbol), ]

# 3.4 Save Full Results CSV
write.csv(results_clean, file.path(output_dir, paste0(gse_id,
"_Full_DGE_Results.csv")))
cat("    -> Saved CSV: Full DGE Results\n")

# 3.5 Save Significant Genes (Adjusted P < 0.05 and |LogFC| > 1)
sig_genes <- results_clean %>% filter(adj.P.Val < 0.05 & abs(logFC) > 1)
write.csv(sig_genes, file.path(output_dir, paste0(gse_id,
"_Significant_DEGs.csv")))
cat(paste("    -> Found", nrow(sig_genes), "significant DEGs.\n"))


# ================================================================================
# SECTION 4: VISUALIZATION (BOTH PNG & HTML)
# ================================================================================

cat(">>> [VISUALIZATION] Generating Plots (Static PNG & Interactive HTML)...\n")

# --- 4.1 DATA TABLE (Interactive Only) ---
cat("    -> 1. Data Table...\n")
dt_obj <- datatable(results_clean, options = list(pageLength = 15, scrollX =
TRUE), filter = 'top',
                    caption = paste(gse_id, "Results: HER2 vs TNBC"))
saveWidget(dt_obj, file.path(output_dir, paste0(gse_id,
"_1_Interactive_Table.html")))

# --- 4.2 PCA PLOT (Static & 3D Interactive) ---
cat("    -> 2. PCA Plots...\n")
pca <- prcomp(t(exprs_filt), scale. = TRUE)
pca_df <- data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], PC3 = pca$x[,3],
                     Group = subtype_factor, Sample = rownames(pca$x))

# A. STATIC PNG
p_pca_static <- ggplot(pca_df, aes(x=PC1, y=PC2, color=Group)) +
  geom_point(size=3, alpha=0.8) + theme_minimal() +
  scale_color_manual(values = c("TNBC" = "#E74C3C", "HER2" = "#3498DB")) +
  ggtitle(paste(gse_id, "PCA Plot (Static)"))
ggsave(file.path(output_dir, paste0(gse_id, "_2_PCA_Plot.png")), plot =
p_pca_static, width = 8, height = 6)
```

```
# B. INTERACTIVE 3D HTML
p_pca_3d <- plot_ly(pca_df, x = ~PC1, y = ~PC2, z = ~PC3, color = ~Group,
                    colors = c("#E74C3C", "#3498DB"),
                    text = ~paste("Sample:", Sample), type = "scatter3d", mode =
"markers") %>%
  layout(title = paste(gse_id, "3D PCA"))
saveWidget(p_pca_3d, file.path(output_dir, paste0(gse_id,
"_2_Interactive_3D_PCA.html")))

# --- 4.3 HEATMAP (Static & Interactive) ---
cat("    -> 3. Heatmaps...\n")
# Top 50 variable genes
top_var_indices <- order(apply(exprs_filt, 1, var), decreasing = TRUE)[1:50]
mat_heatmap <- exprs_filt[top_var_indices, ]
rownames(mat_heatmap) <- results$Symbol[match(rownames(mat_heatmap),
rownames(results))]
if(anyDuplicated(rownames(mat_heatmap))) rownames(mat_heatmap) <-
make.unique(rownames(mat_heatmap))

# A. STATIC PNG (pheatmap)
annotation_df <- data.frame(Subtype = subtype_factor)
rownames(annotation_df) <- colnames(mat_heatmap)
pheatmap(mat_heatmap, annotation_col = annotation_df, scale = "row",
show_rownames = TRUE,
         main = paste(gse_id, "Top 50 Variable Genes"),
         filename = file.path(output_dir, paste0(gse_id, "_3_Heatmap.png")))

# B. INTERACTIVE HTML (heatmaply)
heatmaply(mat_heatmap, file = file.path(output_dir, paste0(gse_id,
"_3_Interactive_Heatmap.html")),
          main = paste(gse_id, "Top 50 Variable Genes"), scale = "row", colors =
RdBu(256))

# --- 4.4 VOLCANO PLOT (Static & Interactive) ---
cat("    -> 4. Volcano Plots...\n")
results_clean$Significance <- "NS"
results_clean$Significance[results_clean$adj.P.Val < 0.05 &
abs(results_clean$logFC) > 1] <- "Significant"

# A. STATIC PNG (EnhancedVolcano)
p_vol_static <- EnhancedVolcano(results_clean, lab = results_clean$Symbol, x =
'logFC', y = 'adj.P.Val',
                                title = paste(gse_id, 'HER2 vs TNBC'), pCutoff =
0.05, FCcutoff = 1.0)
ggsave(file.path(output_dir, paste0(gse_id, "_4_Volcano_Plot.png")), plot =
p_vol_static, width = 10, height = 8)

# B. INTERACTIVE HTML (Plotly)
p_int_vol <- ggplot(results_clean, aes(x = logFC, y = -log10(adj.P.Val),
                                       text = paste("Gene:", Symbol, "<br>FC:",
round(logFC,2), "<br>FDR:", format(adj.P.Val, digits=3)),
                                       color = Significance)) +
  geom_point(alpha = 0.6) + scale_color_manual(values = c("gray", "red")) +
theme_minimal() +
  labs(title = paste(gse_id, "Interactive Volcano"))
saveWidget(ggplotly(p_int_vol, tooltip="text"), file.path(output_dir,
paste0(gse_id, "_4_Interactive_Volcano.html")))


# ================================================================================
# SECTION 5: FUNCTIONAL ENRICHMENT (BOTH PNG & HTML)
# ================================================================================

cat(">>> [ENRICHMENT] Running Pathway Analysis...\n")
```

```r
gene_list <- unique(sig_genes$Entrez)
gene_list <- gene_list[!is.na(gene_list)]

if (length(gene_list) > 10) {

  # --- 5.1 GO (Biological Process) ---
  cat("    -> GO Analysis...\n")
  ego <- enrichGO(gene = gene_list, OrgDb = org.Hs.eg.db, ont = "BP",
pAdjustMethod = "BH", readable = TRUE)
  if (!is.null(ego) && nrow(ego) > 0) {
    write.csv(as.data.frame(ego), file.path(output_dir, paste0(gse_id,
"_6_GO_Results.csv")))

    # Save PNG
    p_go <- dotplot(ego, showCategory = 15) + ggtitle(paste(gse_id, "GO:
Biological Process"))
    ggsave(file.path(output_dir, paste0(gse_id, "_6_GO_Dotplot.png")), plot =
p_go, width = 10, height = 8)

    # Save HTML
    saveWidget(ggplotly(p_go), file.path(output_dir, paste0(gse_id,
"_6_Interactive_GO.html")))
  }

  # --- 5.2 KEGG Pathways ---
  cat("    -> KEGG Analysis...\n")
  kk <- enrichKEGG(gene = gene_list, organism = 'hsa', pvalueCutoff = 0.05)
  if (!is.null(kk) && nrow(kk) > 0) {
    write.csv(as.data.frame(kk), file.path(output_dir, paste0(gse_id,
"_7_KEGG_Results.csv")))

    # Save PNG
    p_kegg <- dotplot(kk, showCategory = 15) + ggtitle(paste(gse_id, "KEGG
Pathways"))
    ggsave(file.path(output_dir, paste0(gse_id, "_7_KEGG_Dotplot.png")), plot =
p_kegg, width = 10, height = 8)

    # Save HTML
    saveWidget(ggplotly(p_kegg), file.path(output_dir, paste0(gse_id,
"_7_Interactive_KEGG.html")))
  }

  # --- 5.3 Reactome Pathways ---
  cat("    -> Reactome Analysis...\n")
  reac <- enrichPathway(gene = gene_list, pvalueCutoff = 0.05, readable = TRUE)
  if (!is.null(reac) && nrow(reac) > 0) {
    write.csv(as.data.frame(reac), file.path(output_dir, paste0(gse_id,
"_8_Reactome_Results.csv")))

    # Save PNG
    p_reac <- dotplot(reac, showCategory = 15) + ggtitle(paste(gse_id, "Reactome
Pathways"))
    ggsave(file.path(output_dir, paste0(gse_id, "_8_Reactome_Dotplot.png")),
plot = p_reac, width = 10, height = 8)

    # Save HTML
    saveWidget(ggplotly(p_reac), file.path(output_dir, paste0(gse_id,
"_8_Interactive_Reactome.html")))
  }

} else {
  cat(">>> [WARNING] Not enough significant genes for pathway analysis.\n")
}
```

```
# ==============================================================================
# SECTION 6: COMPLETION
# ==============================================================================

cat("\n#########################################################################
\n")
cat(" ANALYSIS COMPLETE FOR", gse_id, "\n")
cat(" All files (PNG Images, HTML Interactive, CSV Tables) are saved in:\n")
cat(" ", output_dir, "\n")
cat("#########################################################################\n
")
```