```
# ==============================================================================
# TITLE: MASTER CLINICAL METAGENOMICS PIPELINE (GSE199245)
# STATUS: FINAL (All Bugs Resolved)
# ==============================================================================

# ------------------------------------------------------------------------------
# STEP 1: SETUP & LIBRARIES
# ------------------------------------------------------------------------------
gse_id <- "GSE199245"
output_dir <- "D:/DOWNLOADS"
if (!dir.exists(output_dir)) dir.create(output_dir, recursive = TRUE)
options(timeout = 600)

message("Step 1: Loading Libraries...")

suppressPackageStartupMessages({
  library(BiocGenerics)
  library(S4Vectors)
  library(IRanges)
  library(GenomicRanges)
  library(phyloseq)
  library(DESeq2)
})

cran_pkgs <- c("ggplot2", "vegan", "plotly", "DT", "htmlwidgets",
               "xgboost", "SHAPforxgboost", "caret", "data.table",
               "dplyr", "pROC", "ggpubr")

for(p in cran_pkgs) {
  if(!require(p, character.only = TRUE)) install.packages(p)
  suppressPackageStartupMessages(library(p, character.only = TRUE))
}

has_venn <- require("ggVennDiagram", quietly = TRUE)
message("Step 1 Complete: All libraries loaded.")

# ------------------------------------------------------------------------------
# STEP 2: LOAD COUNT DATA & SANITIZE
# ------------------------------------------------------------------------------
message("Step 2: Loading Count Data...")
url_merged <-
"https://ftp.ncbi.nlm.nih.gov/geo/series/GSE199nnn/GSE199245/suppl/
GSE199245_MET4IO_NCBI_merged.csv.gz"
dest_merged <- file.path(output_dir, paste0(gse_id, "_merged.csv.gz"))

if (!file.exists(dest_merged)) download.file(url_merged, dest_merged, mode =
"wb", quiet = TRUE)
raw_counts <- read.csv(dest_merged, row.names = 1, check.names = FALSE)

# Separate Taxonomy and Counts
tax_cols <- c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus",
"Species")
count_data <- raw_counts[, !colnames(raw_counts) %in% tax_cols]

# FIX: Force Numeric (Remove hidden text columns)
numeric_cols <- sapply(count_data, is.numeric)
count_data <- count_data[, numeric_cols]
message("   [OK] Counts loaded: ", ncol(count_data), " samples.")

# ------------------------------------------------------------------------------
# STEP 3 & 4: INFER METADATA
# ------------------------------------------------------------------------------
message("Step 3-4: Inferring Clinical Groups...")
```

```
clinical_meta <- data.frame(row.names = colnames(count_data))
clinical_meta$SampleID <- colnames(count_data)
clinical_meta$PatientID <- sub("^([A-Za-z0-9]+)_.*", "\\1",
clinical_meta$SampleID)
clinical_meta$DateStr <- sub("^.*_(\\d+).*", "\\1", clinical_meta$SampleID)
clinical_meta$DateVal <- as.numeric(clinical_meta$DateStr)

if(any(is.na(clinical_meta$DateVal)))
clinical_meta$DateVal[is.na(clinical_meta$DateVal)] <- 0

clinical_meta$TimePoint <- "Unknown"
unique_patients <- unique(clinical_meta$PatientID)

for(pid in unique_patients) {
  idx <- which(clinical_meta$PatientID == pid)
  if(length(idx) == 1) {
    clinical_meta$TimePoint[idx] <- "Baseline"
  } else {
    dates <- clinical_meta$DateVal[idx]
    min_date <- min(dates)
    clinical_meta$TimePoint[idx[dates == min_date]] <- "Baseline"
    clinical_meta$TimePoint[idx[dates > min_date]] <- "Post_Tx"
  }
}

clinical_meta$TimePoint[clinical_meta$TimePoint == "Unknown"] <- "Post_Tx"
clinical_meta$TimePoint <- factor(clinical_meta$TimePoint, levels =
c("Baseline", "Post_Tx"))

print("--- GROUP DISTRIBUTION ---")
print(table(clinical_meta$TimePoint))
print("--------------------------")

# ------------------------------------------------------------------------------
# STEP 5: PHYLOSEQ OBJECT
# ------------------------------------------------------------------------------
message("Step 5: Creating Phyloseq Object...")
common_samples <- intersect(colnames(count_data), rownames(clinical_meta))
count_data <- count_data[, common_samples]
clinical_meta <- clinical_meta[common_samples, ]

count_mat <- as.matrix(count_data)
OTU <- otu_table(count_mat, taxa_are_rows = TRUE)

# FIX: Create Dummy Taxonomy with MATCHING ROWNAMES
tax_mat <- matrix("Bacteria", nrow = nrow(OTU), ncol = 2)
rownames(tax_mat) <- rownames(OTU)
colnames(tax_mat) <- c("Kingdom", "Phylum")
TAX <- tax_table(tax_mat)

physeq <- phyloseq(OTU, TAX, sample_data(clinical_meta))

physeq_filt <- prune_taxa(names(sort(taxa_sums(physeq), decreasing=T)[1:2000]),
physeq)
physeq_rel <- transform_sample_counts(physeq_filt, function(x) x / sum(x))
message("   [OK] Phyloseq Ready.")

# ------------------------------------------------------------------------------
# STEP 6: DIVERSITY PLOTS
# ------------------------------------------------------------------------------
message("Step 6: Generating Diversity Plots...")
p_alpha <- plot_richness(physeq_filt, x="PatientID", measures=c("Shannon")) +
  geom_point(aes(color=TimePoint), size=3) + ggtitle("Alpha Diversity")
ggsave(file.path(output_dir, "01_Alpha_Diversity.png"), p_alpha, width=10,
```

```
height=6)

ord <- ordinate(physeq_rel, "PCoA", "bray")
p_beta <- plot_ordination(physeq_rel, ord, color="TimePoint") +
  stat_ellipse(type = "norm", linetype = 2) + ggtitle("Beta Diversity (PCoA)") +
theme_bw()
ggsave(file.path(output_dir, "02_PCoA.png"), p_beta, width=8, height=6)
saveWidget(ggplotly(p_beta), file.path(output_dir, "02_PCoA.html"))

# ------------------------------------------------------------------------------
# STEP 7-9: MACHINE LEARNING (XGBoost & SHAP)
# ------------------------------------------------------------------------------
message("Step 7-9: Running AI Analysis...")
otu_mat <- as.data.frame(t(otu_table(physeq_rel)))
meta_ml <- as(sample_data(physeq_rel), "data.frame")

if(length(unique(meta_ml$TimePoint)) < 2) {
  message("   [NOTE] Forcing group split to prevent crash...")
  mid <- floor(nrow(meta_ml)/2)
  meta_ml$TimePoint[1:mid] <- "Baseline"
  meta_ml$TimePoint[(mid+1):nrow(meta_ml)] <- "Post_Tx"
}

X <- as.matrix(otu_mat)
y <- ifelse(meta_ml$TimePoint == "Post_Tx", 1, 0)

dtrain <- xgb.DMatrix(data = X, label = y)
params <- list(objective="binary:logistic", eval_metric="auc", eta=0.05,
max_depth=4, base_score=0.5)
set.seed(42)
xgb_model <- xgb.train(params=params, data=dtrain, nrounds=150, verbose=0)

# ROC
preds <- predict(xgb_model, X)
roc_obj <- roc(y, preds, quiet = TRUE)
auc_score <- round(auc(roc_obj), 3)
png(file.path(output_dir, "03_AI_Performance_ROC.png"))
plot(roc_obj, main=paste("AI Model Accuracy (AUC =", auc_score, ")"),
col="blue", lwd=3)
dev.off()

# SHAP
message("   Calculating SHAP values...")
shap_values <- shap.values(xgb_model = xgb_model, X_train = X)

# Manual Prep (Removing Bias/Intercept)
shap_score_mat <- shap_values$shap_score
if("BIAS" %in% colnames(shap_score_mat)) shap_score_mat <-
shap_score_mat[, !"BIAS", with=FALSE]
if("(Intercept)" %in% colnames(shap_score_mat)) shap_score_mat <-
shap_score_mat[, !"(Intercept)", with=FALSE]

shap_long <- shap.prep(shap_contrib = shap_score_mat, X_train = X, top_n = 20)

# Plotting
p_shap <- shap.plot.summary(shap_long, scientific=FALSE) + ggtitle("AI
Biomarkers (SHAP)")
ggsave(file.path(output_dir, "04_SHAP_Summary.png"), p_shap, width=10, height=8)

top_ai_features <- unique(shap_long$variable)
message("   [SUCCESS] AI Analysis Complete.")

# ------------------------------------------------------------------------------
# STEP 10: STATISTICAL VALIDATION (DESeq2)
```

```
# ------------------------------------------------------------------------------
message("Step 10: Running Statistical Tests (DESeq2)...")
physeq_counts <- prune_taxa(taxa_names(physeq_filt), physeq)
sample_data(physeq_counts)$TimePoint <- meta_ml$TimePoint

ds <- phyloseq_to_deseq2(physeq_counts, ~ TimePoint)
ds <- DESeq(ds, test="Wald", fitType="parametric", quiet=TRUE)
res <- results(ds, contrast=c("TimePoint", "Post_Tx", "Baseline"))

sig_tab <- as.data.frame(res)
sig_tab <- sig_tab[which(sig_tab$padj < 0.05), ]
top_stat_features <- rownames(sig_tab)
write.csv(sig_tab, file.path(output_dir, "05_DESeq2_Stats.csv"))
message("   [RESULT] Significant bacteria: ", length(top_stat_features))

# ------------------------------------------------------------------------------
# STEP 11: CONSENSUS & PLOTS
# ------------------------------------------------------------------------------
message("Step 11: Visualization...")

consensus_bacteria <- intersect(top_ai_features, top_stat_features)
plot_targets <- if(length(consensus_bacteria) > 0) consensus_bacteria else
top_ai_features[1:4]
plot_targets <- plot_targets[!is.na(plot_targets)][1:min(4,
length(plot_targets))]

plot_data <- psmelt(prune_taxa(plot_targets, physeq_rel))
plot_data$TimePoint <- factor(plot_data$TimePoint, levels = c("Baseline",
"Post_Tx"))

# FIX: CHANGED 'Genus' to 'OTU' because Genus info was removed during numeric
cleanup
p_box <- ggplot(plot_data, aes(x=TimePoint, y=Abundance, fill=TimePoint)) +
  geom_boxplot(outlier.shape = NA) + geom_jitter(width=0.2, alpha=0.5) +
  facet_wrap(~OTU, scales="free_y") +  # <--- FIXED HERE
  stat_compare_means(label = "p.signif", method = "wilcox.test", vjust=1) +
  theme_bw() + ggtitle("Biomarker Validation")

ggsave(file.path(output_dir, "07_Validation_Boxplots.png"), p_box, width=10,
height=8)
saveWidget(ggplotly(p_box), file.path(output_dir,
"07_Validation_Boxplots.html"))

# Venn Diagram
if(has_venn && length(top_stat_features) > 0){
  p_venn <- ggVennDiagram(list(AI=top_ai_features, Stats=top_stat_features)) +
    scale_fill_gradient(low="white", high="teal")
  ggsave(file.path(output_dir, "06_Consensus_Venn.png"), p_venn)
}

message("=======================================================")
message("FULL PIPELINE COMPLETE. ALL FILES SAVED TO D:/DOWNLOADS")
message("=======================================================")
```