Early prognostic biomarkers of corticosteroid response in severe alcohol-related hepatitis: insights from a retrospective study

Damien Esparteiro

2025-07-22

This script aims at producing the graphical and statistical outputs presented in the "Early prognostic biomarkers of corticosteroid response in severe alcohol-related hepatitis: insights from a retrospective study" publication by Esparteiro et al. Each output file can be found in output_PR_vs_PNR folder, and the final plots are stored in a corresponding subfolder.

Basically, this code performs the necessary transformation of the MetaPhlAn output into several phyloseq objects that are then analysed using the traditional approaches (ecology metrics, differential abundance analysis). The code also analyses the clinical and biological variables presented in the article and stored in the metadata file.

Start with loading useful packages, objects, and at-home scripts (those are loaded and used in these chunks, see the "functions" folder) . The data.csv file comes from MetaPhlAn and contain absolute read counts of micro-organisms.

```
Sys.setenv(LANGUAGE = "en")
set.seed(123)

required_packages <- c(
  "here", "phyloseq", "gridExtra", "dplyr", "openxlsx", "tidyr", "purrr", "ggpubr", "rstatix",  "RVAidel
)

bioc_packages <- c("phyloseq", "Maaslin2", "microbiome", "ComplexHeatmap")

if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

install_if_missing <- function(pkg) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    tryCatch({
      if (pkg %in% bioc_packages) {
        BiocManager::install(pkg, ask = FALSE, update = FALSE)
      } else {
        install.packages(pkg, dependencies = TRUE)
      }
    }, error = function(e) {
      stop("Script stopped due to package installation failure.")
    })
  }
}

install_packages <- FALSE # Toggle if needed
if (install_packages) {
```

```
  invisible(sapply(required_packages, install_if_missing))
}

invisible(lapply(required_packages, library, character.only = TRUE))

output_folder <- paste0(here::here("output_PR_vs_PNR", "/"))

if (dir.exists(output_folder)) {
  message(sprintf("Deleting existing folder: %s", output_folder))
  unlink(output_folder, recursive = TRUE, force = TRUE)
}
dir.create(output_folder)
message(sprintf("Created new folder: %s", output_folder))

metadata <- read.csv(here::here("data", "metadata_GM_Response.csv"), sep = ";", header = TRUE)
metadata$Row.names <- NULL
rownames(metadata) <- metadata$Sample

metadata$Response <- as.factor(metadata$Response)
metadata_sample_data <- sample_data(metadata)
comparisons <- c("Response")
order_by_Response <- c("PR", "PNR")
colors_two_groups <- c("blue", "red")

taxonomic_levels <- c("Phylum", "Genus", "Species")

my_theme <- ttheme_default(
  core = list(fg_params = list(hjust = 0.5, x = 0.5, cex = 0.8),
              bg_params = list(fill = "white", col = "black")),
  colhead = list(fg_params = list(hjust = 0.5, x = 0.5),
                 bg_params = list(fill = "white", col = "black")),
  rowhead = list(fg_params = list(hjust = 0.5, x = 0.5),
                 bg_params = list(fill = "white", col = "black"))
)

data <- read.delim(here::here("data", "data_GM_Response.txt"), header = TRUE)

data[data == "-"] <- "0" # No SGB-based taxonomy

function_files <- list.files(path = here::here("functions"), pattern = "^perform.*\\.R$", full.names = T

for (f in function_files) {
  message(sprintf("Sourcing function: %s", f))
  source(f)
}
```

Then, perform the generation of baseline clinico-biological characteristics between PR and PNR : a full_output file is generated with all the relevant data (both mean and median are stored alongside dispersion metrics and statistical tests p-values).

```
outputs <- list()
for (comparison in comparisons) {
  output_name <- paste("output_", comparison, sep = "")
```

```
  output <- perform_full_stats(
    dataset = metadata,
    grouping_variable = comparison,
    grouping_variable_levels = get(paste("order_by_", comparison, sep = "")),
    pairing = FALSE,
    rounding_number = 1,
    columns_to_remove = NULL,
    ID = "Sample",
    sample_name = "Sample"
  )
  outputs[[output_name]] <- output
}
for (output_name in names(outputs)) {
  dataset <- outputs[[output_name]]
  dataset[["Variable"]] <- row.names(dataset)
  row.names(dataset) <- NULL
  dataset <- dataset[, c("Variable", setdiff(names(dataset), "Variable"))]
  filename <- file.path(output_folder, paste0("full_stats_", comparison, ".xlsx"))
  write.xlsx(dataset, filename, quote = FALSE, row.names = FALSE)
}
```

```
output[11,] # For example, the LBP
```

```
##                                          Total : n = 63
## LBP 115.1 ± 75.6 [n = 26] / 108.4 (47.1-166.1) [n = 26]
##                                             PR : n = 36
## LBP 130.6 ± 73.6 [n = 14] / 123.4 (77.9-184.9) [n = 14]
##                                          PNR : n = 27 Shapiro_p_value_PR
## LBP 97.0 ± 76.9 [n = 12] / 73.6 (41.3-127.0) [n = 12]          0.5513322
##     Shapiro_p_value_PNR Levene_p_value unpaired_t_test_p_value
## LBP          0.09161398      0.8116019               0.2670249
##     unpaired_Wilcoxon_p_value Total Fisher_p_value Chi_square_p_value
## LBP                        NA  <NA>             NA                 NA
```

After loading the tables, we can continue and transform the MetaPhlAn abundance table into a phyloseq object for further data handling. Create multiple phyloseq object for our needs.

```
for (taxonomic_level in taxonomic_levels) {
  result <- perform_metaphlan_to_phyloseq(
    data,
    metadata = NULL,
    version = 4,
    verbose = FALSE,
    tax_lvl = taxonomic_level
  )
  sample_data(result) <- metadata_sample_data
  for (comparison in comparisons) {
    comparison_level <- get(paste("order_by_", comparison, sep = ""))
    result@sam_data[[comparison]] <- factor(result@sam_data[[comparison]], levels = comparison_level)
  }
  assign(paste("phyloseq_", taxonomic_level, sep = ""), result)
  rm(result)
```

```
}
# Duplicate objects for downstream modifications
phyloseq_Genus_2 <- phyloseq_Genus
phyloseq_Species_2 <- phyloseq_Species
phyloseq_objects <-  c("phyloseq_Species", "phyloseq_Genus", "phyloseq_Phylum"
)
```

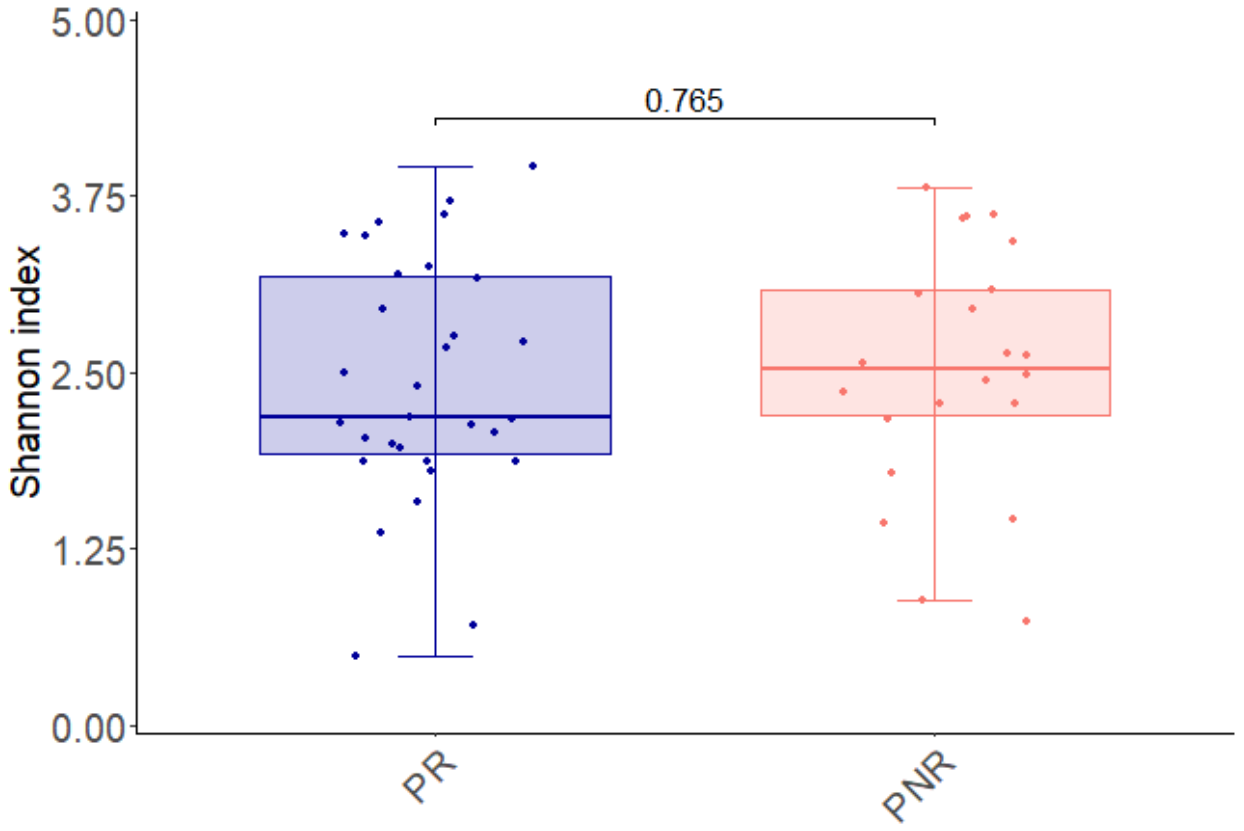
```
phyloseq_Genus # Phyloseq object at the genus level
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 347 taxa and 53 samples ]
## sample_data() Sample Data:       [ 53 samples by 101 sample variables ]
## tax_table()   Taxonomy Table:    [ 347 taxa by 6 taxonomic ranks ]
```

We can begin with the analysis at the species level of the GM diversity, starting with the alpha-diversity Shannon index.

```
measures <- c("Shannon", "Observed")
richness <- estimate_richness(phyloseq_Species, measures = measures)
richness$Sample <- rownames(richness)
richness <- merge(richness, metadata, by = "Sample", all.x = FALSE)
for (comparison in comparisons) {
  for (index in measures) {
    breaks <- NULL
    limits <- NULL
    expand <- NULL
    ypositions <- NULL
      label_y <- "Species Richness"
    if (index == "Shannon" && comparison == "Response") {
      breaks = seq(0, 5, by = 1.25)
      limits = c(0, 5)
      expand = c(0.01, 0)
      ypositions = c(4.3)
        label_y <- "Shannon index"

    }
    perform_mean_comparison(
      richness,
      name = index,
      response_var = index,
      grouping_var = comparison,
      breaks = breaks,
      limits = limits,
      expand = expand,
      ypositions = ypositions,
      paired = FALSE,
      order = get(paste0("order_by_", comparison)),
      add_p_value = TRUE,
      output_folder = paste0(output_folder, "alpha_metrics/"),
      label_y = label_y
    )
  }
}
```

```
plot_Shannon_by_Response # No difference in alpha diversity
```



Taxonomic analysis can further be completed using differential abundance analysis : are there taxa significantly enriched in some groups ? What are these ? We can use MaAslin2 for this purpose. In addition, we can obtain microbial prevalence. This chunk of code performs Chi^2 tests to compare prevalence, and use MaAslin to test for differential abundance.

```
for (phyloseq_object in phyloseq_objects) {
  for (comparison in comparisons) {
    fixed_effects <- c("Age", "Sex", "Antibiotics", comparison)
    perform_prevalence(phyloseq_object, comparison , output_folder = output_folder)
    perform_maaslin(
      phyloseq_object = phyloseq_object,
      metadata = metadata,
      comparison = comparison,
      fixed_effects = fixed_effects,
      random_effects = NULL,
      output_folder = output_folder,
      if (comparison %in% c("Response")) {
  references = c(comparison, "PR")
} else {
  references = NULL
}
    )
  }
}
```

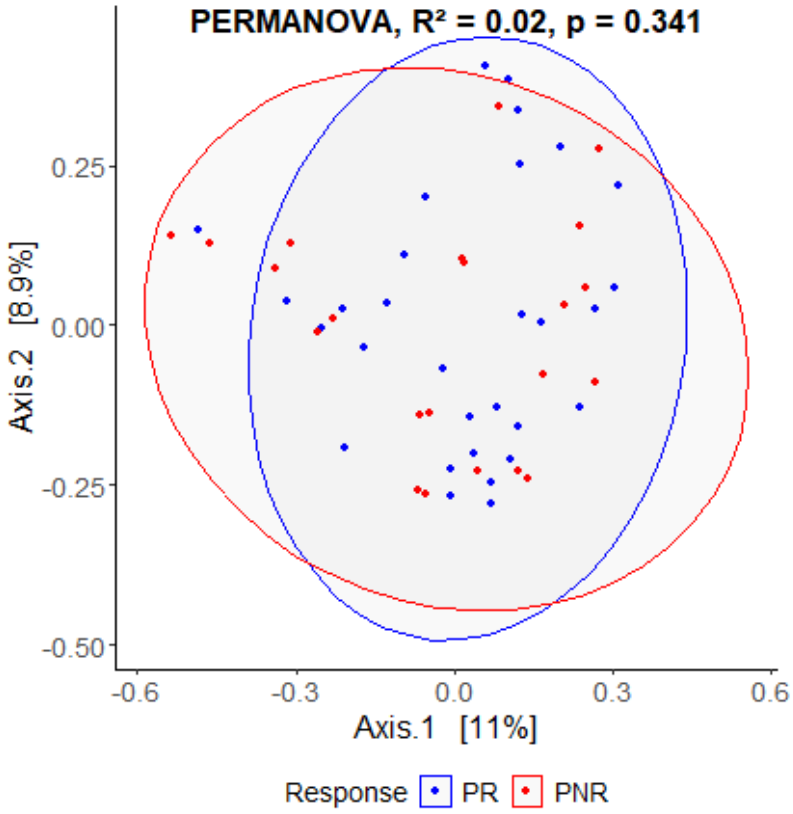
We may continue with the beta-diversity analysis : how different are the GM, at the level of the community ? We first have to merge the phyloseq object with a phylogenetic tree provided by MetaPhlAn in order to perform Weighted UniFrac analysis.

```
tree = read.tree(here::here("data","metaphlan_tree.nwk"))
bug_list = read.delim(here::here("data", "metaphlan_species.txt"), , header = F)
bug_list$V1 = gsub("SGB|_group|EUK", "", bug_list$V1)
column_names = c("V2", paste0("Other", 1:110))
bug_list = separate(bug_list, V2, into = column_names, sep = ",")
bug_list$V2 <- sub(".*s__", "s__", bug_list$V2)
bug_list = bug_list[!duplicated(bug_list[c("V2")]), ]
bug_list2 = subset(bug_list, V1 %in% tree$tip.label)
tree2 = drop.tip(tree, setdiff(tree$tip.label, bug_list2$V1))
bug_list2 = bug_list2[match(tree2$tip.label, bug_list2$V1), ]
tree2$tip.label = bug_list2$V2
phyloseq_Species <- merge_phyloseq(phyloseq_Species, tree2)
comm <- t(otu_table(phyloseq_Species))
```

Bray-Curtis plot :

```
for (comparison in comparisons) {
  for (index in c("bray", "wunifrac", "jaccard", "uunifrac")) {
    perform_beta_diversity(
      physeq_object = phyloseq_Species,
      beta_metric = index,
      grouping_var = comparison,
      output_folder = output_folder,
      title = ""
    )
  }
}
```

```
plot_bray_by_Response # No difference in beta diversity
```

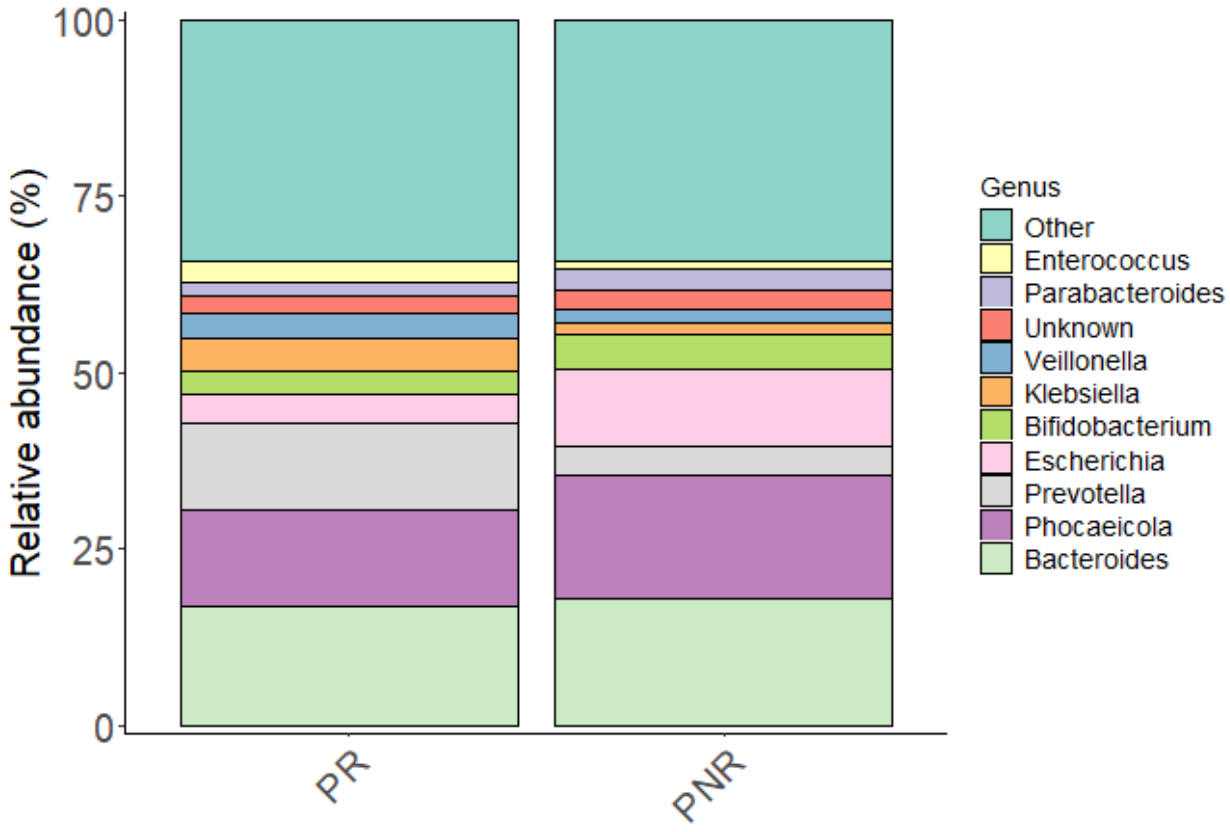


Now that the diversities have been evaluated, we can proceed to the taxonomy analysis. Start with generating barplots using the function perform_stack_barplots. Phyloseq objects need some handling before being passed to this function. We also generate boxplots for the main taxa at each level.

```
for (object_name in phyloseq_objects)
  {
  for (comparison in comparisons) {
    current_object <- get(object_name)
    grouping_variable <- comparison
    non_na_samples <- !is.na(sample_data(current_object)[[grouping_variable]])
    current_object <- prune_samples(non_na_samples, current_object)
    taxa_level <- str_match(object_name, "phyloseq_(\\w+)")[, 2]
    rel_current_object <- microbiome::transform(current_object, "compositional")
    NOTU <- ifelse(grepl("Phylum", taxa_level), 7, 10)
    taxa_sums_result <- taxa_sums(rel_current_object)
    ordered_taxa <- names(sort(taxa_sums_result, decreasing = TRUE))
    Less_abundant_taxa <- ordered_taxa[(NOTU + 1):length(ordered_taxa)]
    new_current_object <- merge_taxa(rel_current_object, Less_abundant_taxa, 1)
    tax_order <- c(ordered_taxa[1:NOTU], Less_abundant_taxa[1])
    otu_table <- t(as.data.frame(otu_table(new_current_object)))
    sam_data <- data.frame(sample_data(new_current_object))
    dataframe <- cbind(otu_table, sam_data)
    colnames(dataframe) <- gsub(colnames(dataframe),
                                pattern = ".*__",
                                replacement = "")
    colnames(dataframe)[1:11] <- gsub("_", " ", colnames(dataframe)[1:11])
    tax_order <- gsub(tax_order, pattern = ".*__", replacement = "")
```

```
    tax_order <- gsub(tax_order, pattern = "_", replacement = " ")
    taxa_to_remove <- tax_order[NOTU + 1]
    tax_order[NOTU + 1] <- "Other"
    colnames(dataframe) <- sapply(colnames(dataframe), function(x) {
      if (x == taxa_to_remove)
        return("Other")
      else
        return(x)
    })
    dataframe <- dataframe %>%
      mutate_at(vars(tax_order), ~ . * 100)
    assign(paste0(gsub("phyloseq", "dataframe", object_name)), dataframe, envir = .GlobalEnv)
    perform_stacked_barplots(
      dataframe = dataframe,
      abundance_type = "relative",
      title = taxa_level,
      variables = rev(tax_order),
      grouping_var = grouping_variable,
      output_folder = output_folder,
      individual = "Sample",
      hide_x_text = TRUE,
      y_axis_breaks = c(0, 25, 50, 75, 100),
      y_axis_limits = c(0, 100),
      color_palette = NULL,
      label_y = "Relative abundance (%)"
    )
    for (taxa in rev(tax_order)) {
      perform_mean_comparison(
        dataframe,
        response_var = taxa,
        grouping_var = grouping_variable,
        output_folder = output_folder,
        name = taxa_level,
        order = get(paste0("order_by_", grouping_variable)),
        breaks = c(0, 25, 50, 75, 100),
        limits = c(0, 100),
        expand = c(0.01, 0.01),
        ypositions = NULL,
        paired = FALSE,
        add_p_value = TRUE
      )
    }
  }
}
```

```
plot_average_stacked_Genus_by_Response # Taxonomic profiles (genus)
```



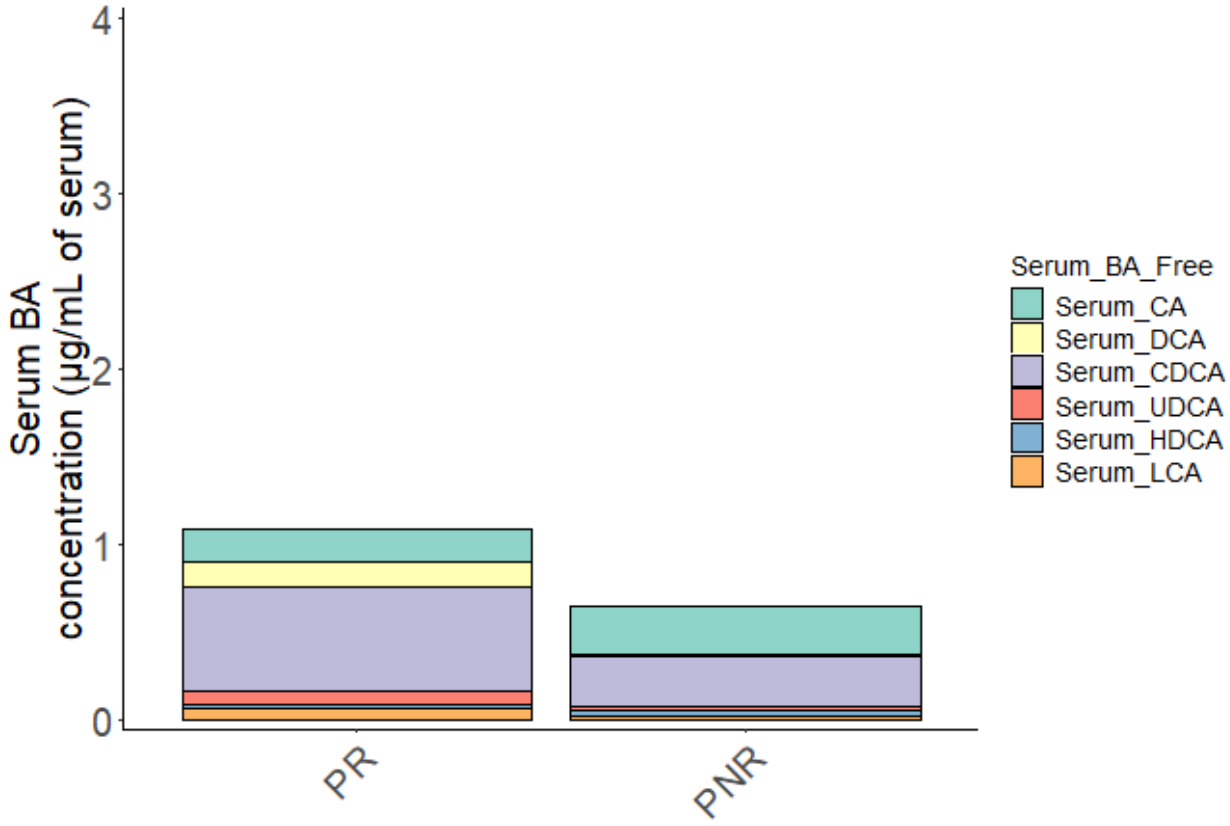
Bile Acids plots :

```
variables_Bile_Acids <- grep("Stool|Serum", colnames(metadata), value = TRUE)
metadata_Bile_Acids <- subset(metadata, Stool_Total > 0)
for (comparison in comparisons) {
  for (variable in variables_Bile_Acids) {
    perform_mean_comparison(
      metadata,
      name = "Bile_Acids",
      response_var = variable,
      grouping_var = comparison,
      output_folder = output_folder,
      breaks = NULL,
      limits = NULL,
      expand = NULL,
      ypositions = NULL,
      paired = FALSE,
      add_p_value = TRUE,
      order = get(paste("order_by_", comparison, sep = ""))
    )
  }
}
variables_Stool_BA_Free <- c("Stool_CA",
                             "Stool_DCA",
                             "Stool_CDCA",
                             "Stool_UDCA",
                             "Stool_HDCA",
```

```
                              "Stool_LCA")

variables_Stool_BA_Glyco <- c(
  "Stool_Glyco_CA",
  "Stool_Glyco_DCA",
  "Stool_Glyco_CDCA",
  "Stool_Glyco_UDCA",
  "Stool_Glyco_HDCA",
  "Stool_Glyco_LCA"
)

variables_Serum_BA_Free <- c("Serum_CA",
                             "Serum_DCA",
                             "Serum_CDCA",
                             "Serum_UDCA",
                             "Serum_HDCA",
                             "Serum_LCA")

variables_Serum_BA_Glyco <- c(
  "Serum_Glyco_CA",
  "Serum_Glyco_DCA",
  "Serum_Glyco_CDCA",
  "Serum_Glyco_UDCA",
  "Serum_Glyco_HDCA",
  "Serum_Glyco_LCA"
)

variables_Serum_BA_Tauro <- c(
  "Serum_Tauro_CA",
  "Serum_Tauro_DCA",
  "Serum_Tauro_CDCA",
  "Serum_Tauro_UDCA",
  "Serum_Tauro_HDCA",
  "Serum_Tauro_LCA"
)

variable_groups <- list(
  Stool_BA_Free = variables_Stool_BA_Free,
  Stool_BA_Glyco = variables_Stool_BA_Glyco,
  Serum_BA_Free = variables_Serum_BA_Free,
  Serum_BA_Glyco = variables_Serum_BA_Glyco,
  Serum_BA_Tauro = variables_Serum_BA_Tauro
)

for (comparison in comparisons) {
  for (group_name in names(variable_groups)) {
    if (any(grepl("Stool", group_name))) {
      label <- "Fecal BA\nconcentration (µg/g of stool)"
    }
    if (any(grepl("Serum", group_name))) {
      label <- "Serum BA\nconcentration (µg/mL of serum)"
    }
```

```
    if (group_name == "Stool_BA_Free") {
      ylimits <- c(0, 400)
      ybreaks <- c(0, 100, 200, 300, 400)
    }
    if (group_name == "Stool_BA_Glyco") {
      ylimits <- c(0, 20)
      ybreaks <- c(0, 5, 10, 15, 20)
    }

    if (group_name == "Serum_BA_Free") {
      ylimits <- c(0, 4)
      ybreaks <- c(0, 1, 2, 3, 4)
    }

    variable_group <- variable_groups[[group_name]]
    perform_stacked_barplots(
      dataframe = metadata,
      title = group_name,
      output_folder = output_folder,
      variables = variable_group,
      grouping_var = comparison,
      individual = "Sample",
      hide_x_text = TRUE,
      color_palette = NULL,
      label_y = label,
      abundance_type = "absolute",
      y_axis_limits = ylimits,
      y_axis_breaks = ybreaks
    )
  }
}
```

```
plot_average_stacked_Serum_BA_Free_by_Response # Example of free serum BA
```

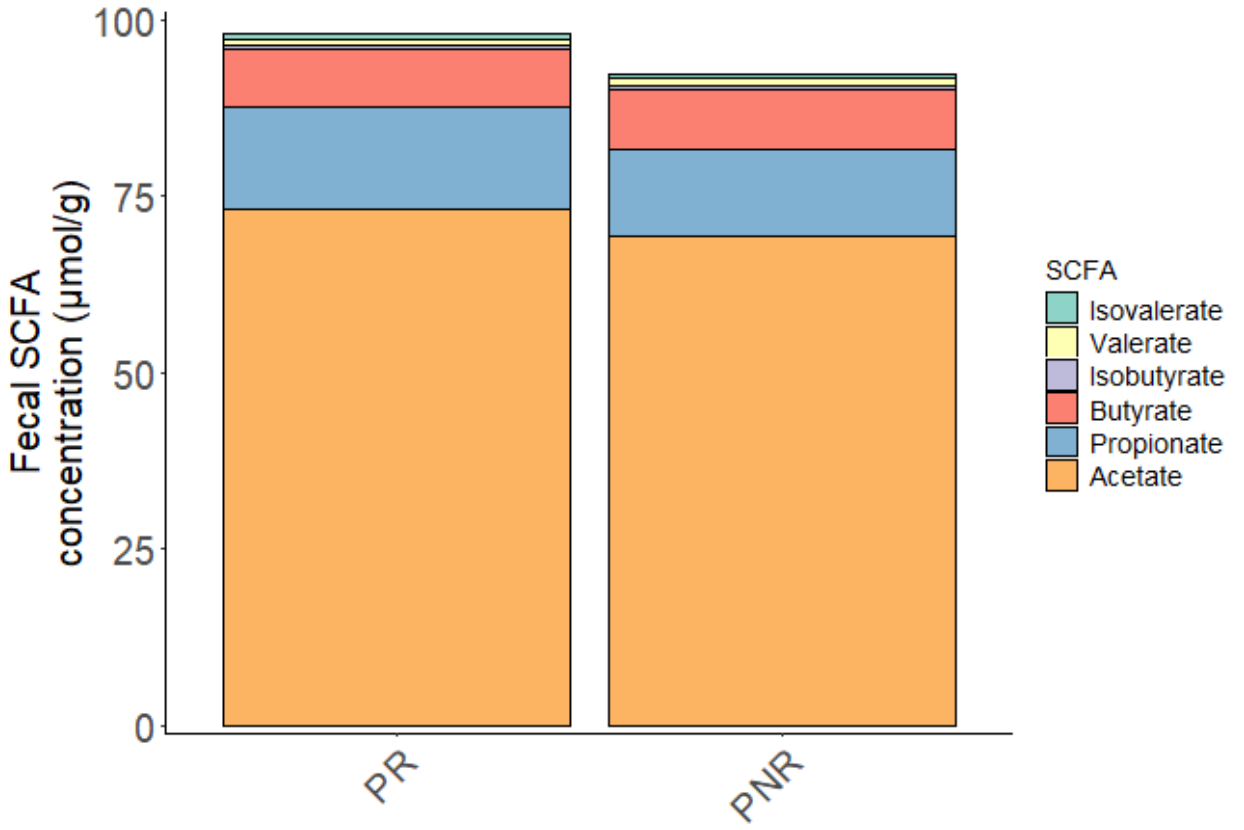


Now, the Short Chain Fatty Acids analysis using similar functions

```
variables_Stool_SCFA <- c(
  "Acetate",
  "Propionate",
  "Isobutyrate",
  "Butyrate",
  "Isovalerate",
  "Valerate",
  "Total_SCFA"
)
metadata_Stool_SCFA <- subset(metadata, Total_SCFA > 0)
for (comparison in comparisons) {
  for (variable in variables_Stool_SCFA) {

    if (variable == "Total_SCFA") {
      ypositions = 255
    } else {
      ypositions = NULL
    }
    perform_mean_comparison(
      metadata_Stool_SCFA,
      response_var = variable,
      grouping_var = comparison,
      output_folder = output_folder,
      breaks = NULL,
      limits = NULL,
```

```
      expand = NULL,
      ypositions = ypositions,
      name = "SCFA",
      add_p_value = TRUE,
      order = get(paste0("order_by_", comparison)),
      paired = FALSE
    )
  }
}

variables_Stool_SCFA <- c("Isovalerate",
                          "Valerate",
                          "Isobutyrate",
                          "Butyrate",
                          "Propionate",
                          "Acetate")

for (comparison in comparisons) {
  perform_stacked_barplots(
    dataframe = metadata_Stool_SCFA,
    title = "SCFA",
    variables = variables_Stool_SCFA,
    output_folder = output_folder,
    grouping_var = comparison,
    individual = "Sample",
    hide_x_text = TRUE,
    color_palette = NULL,
    abundance_type = "absolute",
    label_y = "Fecal SCFA\nconcentration (µmol/g)",
    y_axis_breaks = c(0, 25, 50, 75, 100),
    y_axis_limits = c(0, 100)

  )
}
```

```
plot_average_stacked_SCFA_by_Response # SCFA barplots
```



For the first three main figures, we need to save the panels and to export them. Some already-existing plot modifications are due beforehand, as well as the creation of some missing plots.

```
plot_Prevotella_by_Response$labels$y <- "Prevotella spp.\nrelative abundance (%)"
plot_Prevotella_by_Response <-  plot_Prevotella_by_Response + scale_y_continuous(limits = c(0, 100), bre

s__Prevotella_copri_clade_A <- perform_target_comparison(
  phyloseq_Species_2,
  target = "s__Prevotella_copri_clade_A",
  formatted_target = "s__Prevotella_copri_clade_A",
  grouping_var = "Response",
  output_folder = paste0(output_folder, "/Other/")
)

s__Prevotella_copri_clade_A$labels$y <- "Prevotella copri clade A\nrelative abundance (%)"
s__Prevotella_copri_clade_A <-  s__Prevotella_copri_clade_A + scale_y_continuous(limits = c(0, 100), bre
s__Prevotella_copri_clade_A

plot_Total_SCFA_by_Response$labels$y <- "Fecal SCFA\nconcentration (µmol/g)"
plot_Total_SCFA_by_Response <- plot_Total_SCFA_by_Response + scale_y_continuous(limits = c(0, 260),
                                                                                breaks = c(0, 65, 130, 1



plot_Stool_Total_by_Response$labels$y <- "Fecal BA\nconcentration (µg/g)"
plot_Stool_Total_by_Response <- plot_Stool_Total_by_Response + scale_y_continuous(limits = c(0, 1000),
                                                                                  breaks = c(0, 250, 500
plot_Stool_Total_by_Response$layers[[4]]$data$y.position <- c(950)
```

```
plot_Stool_Primary_by_Response$labels$y <- "Fecal primary\nBA concentration (µg/g)"
plot_Stool_Primary_by_Response <- plot_Stool_Primary_by_Response + scale_y_continuous(limits = c(0, 1000
                                                                                      breaks = c(0, 250
plot_Stool_Primary_by_Response$layers[[4]]$data$y.position <- c(950)

variables <- c(
  "Serum_DCA",
  "Serum_Glyco_LCA",
  "Serum_Tauro_LCA",
  "Serum_UDCA",
  "Prevotella_spp_qPCR_copies",
  "Prevotella_copri_qPCR_copies"
)
for (comparison in comparisons) {
  for (variable in variables) {
    title <- paste0(variable, "_by_", comparison)
    y_title <- title
    breaks = NULL
    limits = NULL

    if (variable == "Serum_DCA") {
  breaks = c(0, 0.4, 0.8,1.2,1.6)
  limits = c(0, 1.6)
  ypositions = 1.3
  y_title <- "Serum DCA (ng/µL) "
} else if (variable == "Serum_Glyco_LCA") {
  breaks = c(0,0.2,0.4,0.6,0.8)
  limits = c(0, 0.8)
  y_title <- "Serum glyco-LCA (ng/µL)"
  ypositions = 0.7
} else if (variable == "Serum_Tauro_LCA") {
  breaks = c(0, 0.1, 0.2, 0.3, 0.4)
  limits = c(0, 0.4)
  y_title <- "Serum tauro-LCA (ng/µL)"
  ypositions = 0.37
} else if (variable == "Serum_UDCA") {
  breaks = c(0,0.1,0.2,0.3,0.4)
  limits = c(0, 0.4)
  y_title <- "Serum UDCA (ng/µL) "
  ypositions = 0.3
} else if (variable == "Prevotella_spp_qPCR_copies") {
  breaks = c(0, 2.5, 5,7.5, 10)
  limits = c(0, 10)
y_title <- "Prevotella spp.\nLog10 copies/ng DNA"

  ypositions = 7.5
} else if (variable == "Prevotella_copri_qPCR_copies") {
  breaks = c(0, 2.5,5,7.5,10)
  limits = c(0, 10)
y_title <- "Prevotella copri\nLog10 copies/ng DNA"
  ypositions = 7.5
} else {
  ypositions = NULL
```

```
}


    plot <- perform_mean_comparison(
      dataframe = metadata,
      name = title,
      response_var = variable,
      grouping_var = comparison,
      order = get(paste0("order_by_", comparison)),
      colors = NULL,
      expand = NULL,
      breaks = breaks,
      limits = limits,
      ypositions = ypositions,
      paired = FALSE,
      label_y = y_title,
      add_p_value = TRUE,
      output_folder = paste0(output_folder, "/Other/")
    )
  }
}
```

Now that the changes are made, we can save the final plots.

Figure 1 :

```
dir.create(paste0(output_folder, "final_plots/"))

if ("Response" %in% comparisons) {
  grid_6_panels <- plot_grid(
    plot_Shannon_by_Response,
    plot_bray_by_Response,
    plot_average_stacked_Phylum_by_Response,
    plot_average_stacked_Genus_by_Response,
    plot_Prevotella_spp_qPCR_copies_by_Response,
    plot_Prevotella_copri_qPCR_copies_by_Response,
    ncol = 2,
    labels = "AUTO",
    label_size = 20
  )

  Cairo::Cairo(
    26,
    40,
    file = paste0(output_folder, "final_plots/Figure 1", ".tif"),
    type = "tif",
    bg = "white",
    dpi = 600,
    units = "cm"
  )
  plot(grid_6_panels)
  dev.off()
}
```

```
plot(grid_6_panels) # Figure 1
```

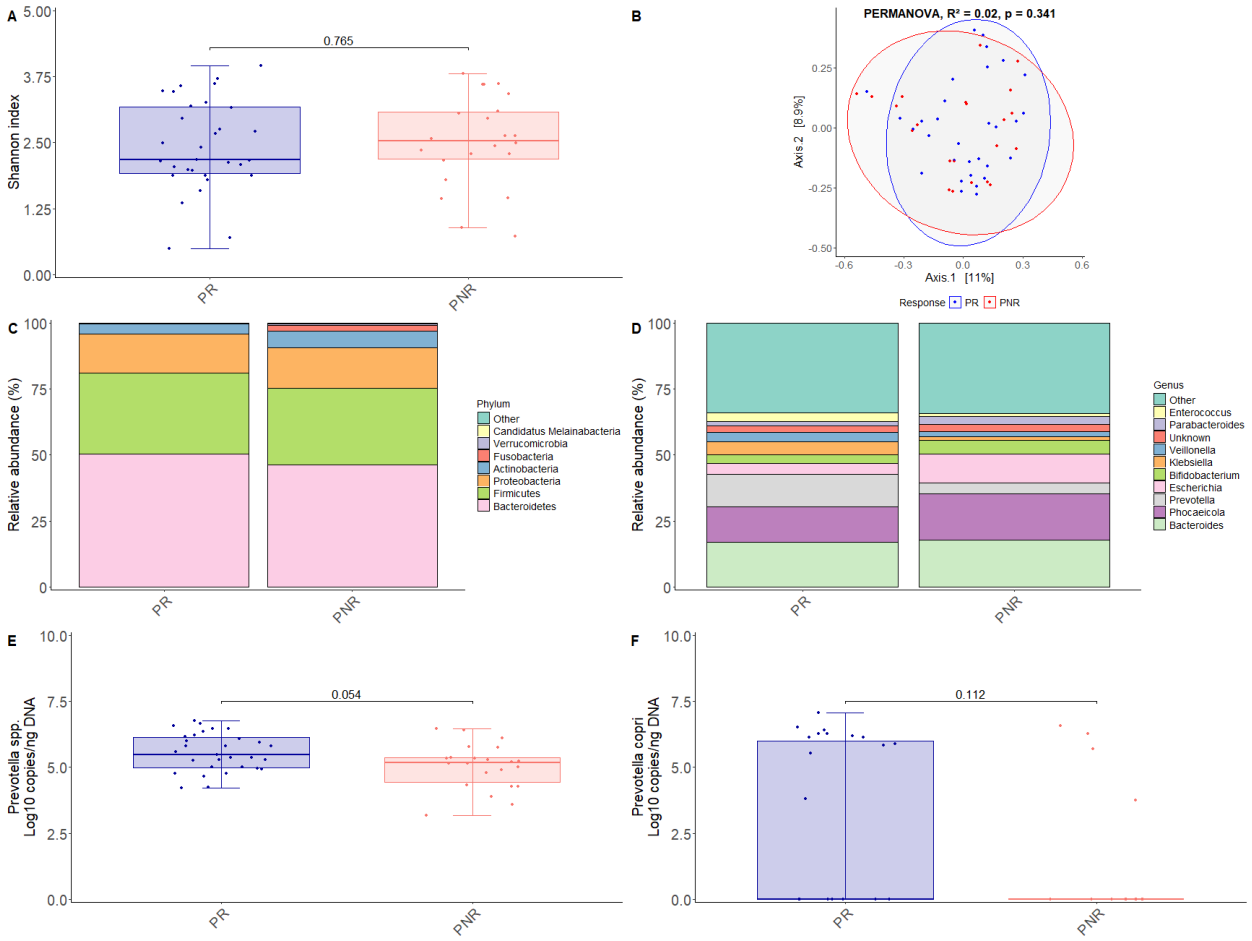


Figure 2 :

```
GM_Fecal_Metabolites <- plot_grid(
  plot_Total_SCFA_by_Response,
  plot_average_stacked_SCFA_by_Response,
  plot_Stool_Total_by_Response,
  plot_Stool_Primary_by_Response,
  ncol = 2,
  labels = "AUTO",
  label_size = 20
)

Cairo::Cairo(
  26,
  30,
  file = paste0(output_folder, "final_plots/Figure 2", ".tif"),
  type = "tif",
  bg = "transparent",
  dpi = 600,
  units = "cm"
)
```

```
plot(GM_Fecal_Metabolites)
dev.off()
```

```
plot(GM_Fecal_Metabolites) # Figure 2
```

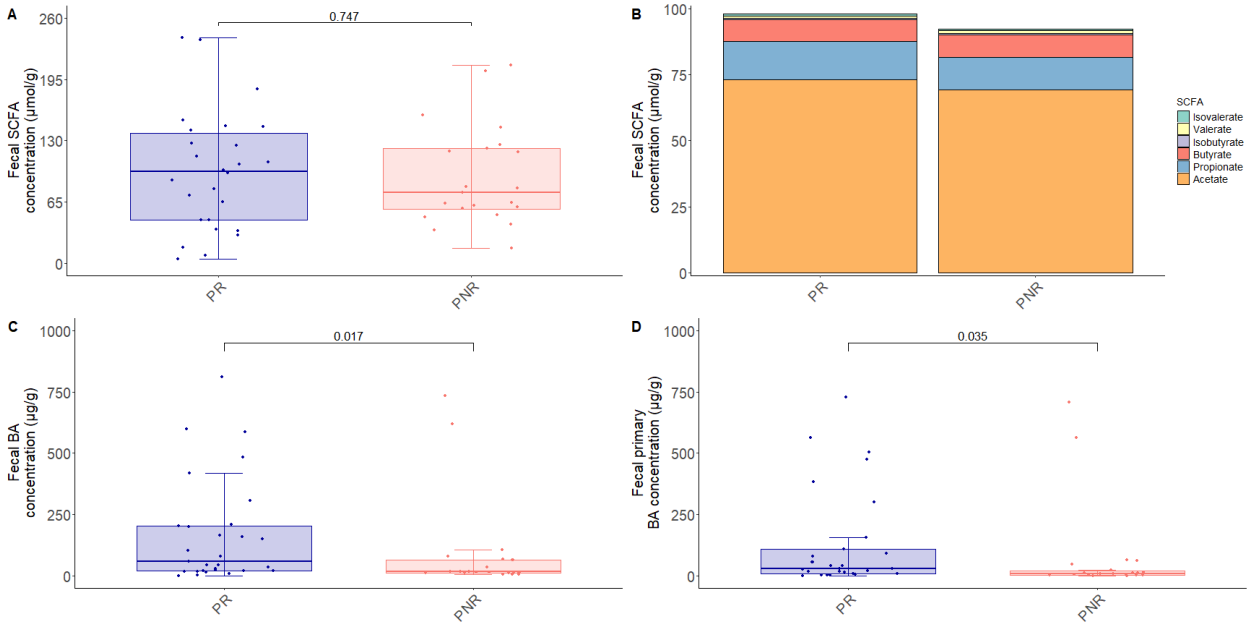
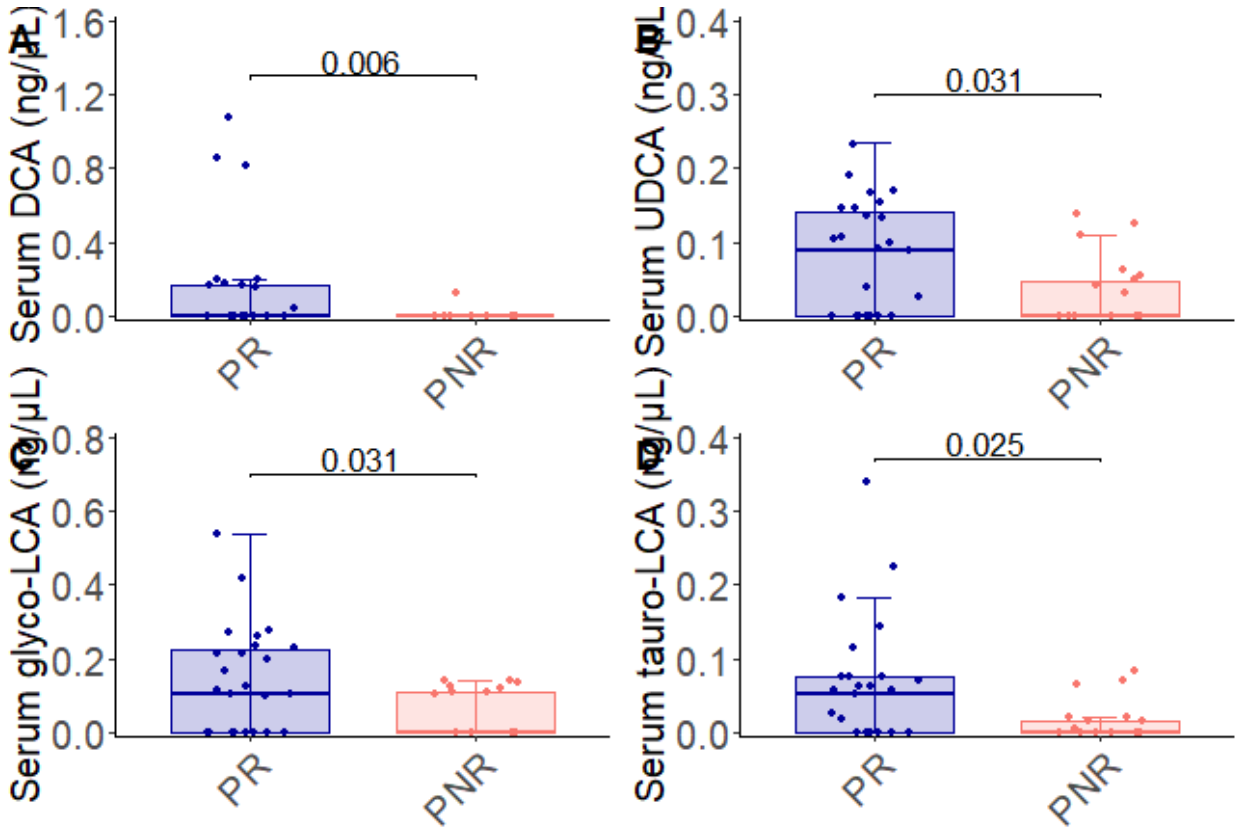


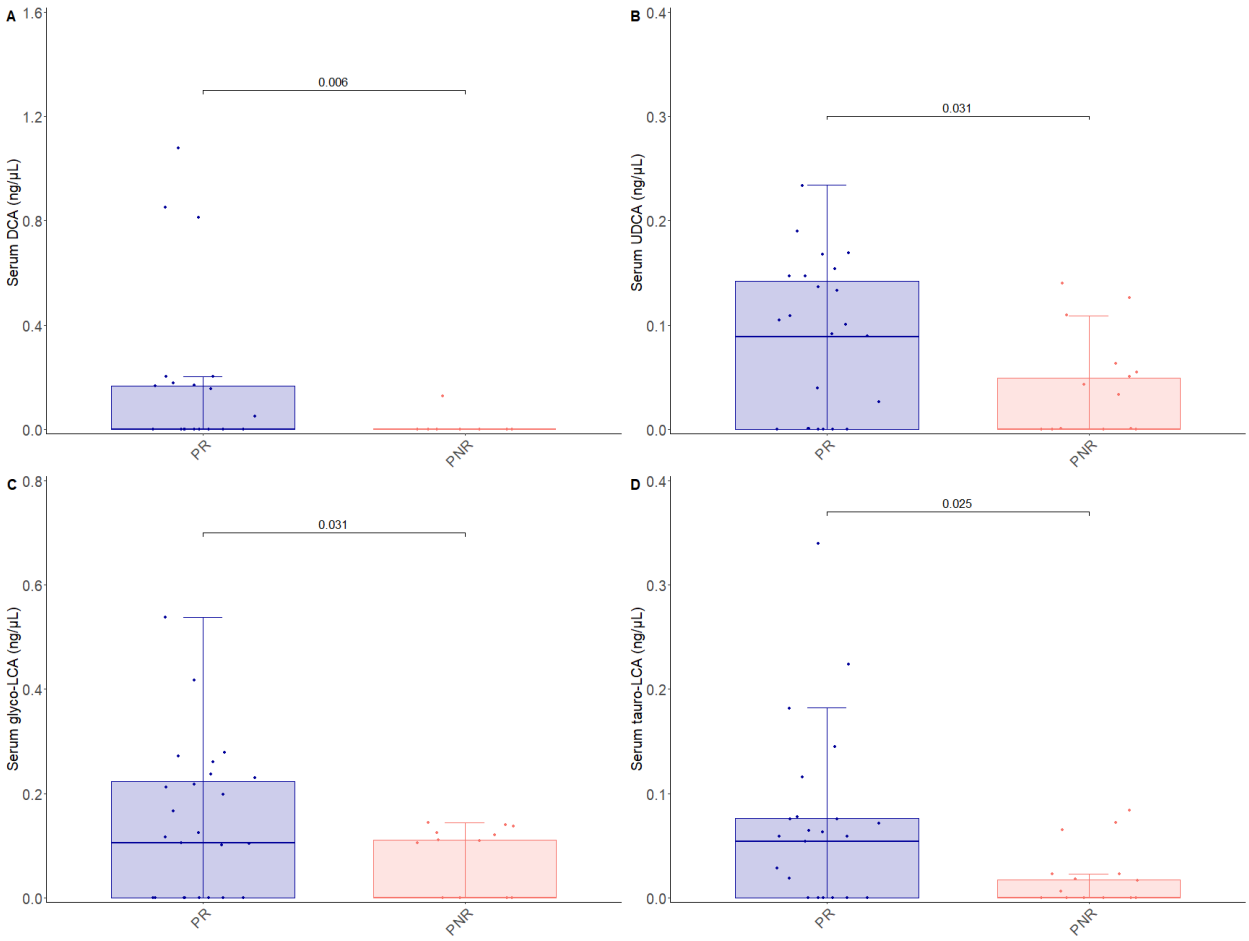
Figure 3 :

```
GM_Serum_BA <- plot_grid(
  plot_Serum_DCA_by_Response,
  plot_Serum_UDCA_by_Response,
  plot_Serum_Glyco_LCA_by_Response,
  plot_Serum_Tauro_LCA_by_Response,
  ncol = 2,
  labels = "AUTO",
  label_size = 20
)
GM_Serum_BA
```




```
Cairo::Cairo(
  26,
  30,
  file = paste0(output_folder, "final_plots/Figure 3", ".tif"),
  type = "tif",
  bg = "transparent",
  dpi = 600,
  units = "cm"
)
plot(GM_Serum_BA)
dev.off()
```

```
plot(GM_Serum_BA) # Figure 3
```



We also want to check that the qPCR and the metagenomics sequencing bring close results : a correlation between Prevotella and Prevotella copri abundance using the two methods (Figure Supplementary 2).

First extract the relative abundance of those two taxa from the phyloseq object.

```
metadata2 <- metadata
rownames(metadata2) <- metadata2$Sample

phyloseq_relab <- transform_sample_counts(phyloseq_Genus_2, function(x)
  x / sum(x))
otu_table_relab <- as.data.frame(otu_table(phyloseq_relab))

taxonomy_table <- as.data.frame(phyloseq_Genus_2@tax_table)

prevotella_row <- which(taxonomy_table$Genus == "g__Prevotella")
prevotella_abundance <- otu_table_relab[prevotella_row, ]
prevotella_abundance <- as.data.frame(t(prevotella_abundance))
colnames(prevotella_abundance) <- "g__Prevotella"
prevotella_abundance$Sample <- rownames(prevotella_abundance)

metadata_with_prevotella <- metadata2 %>%
  left_join(prevotella_abundance, by = "Sample")
```

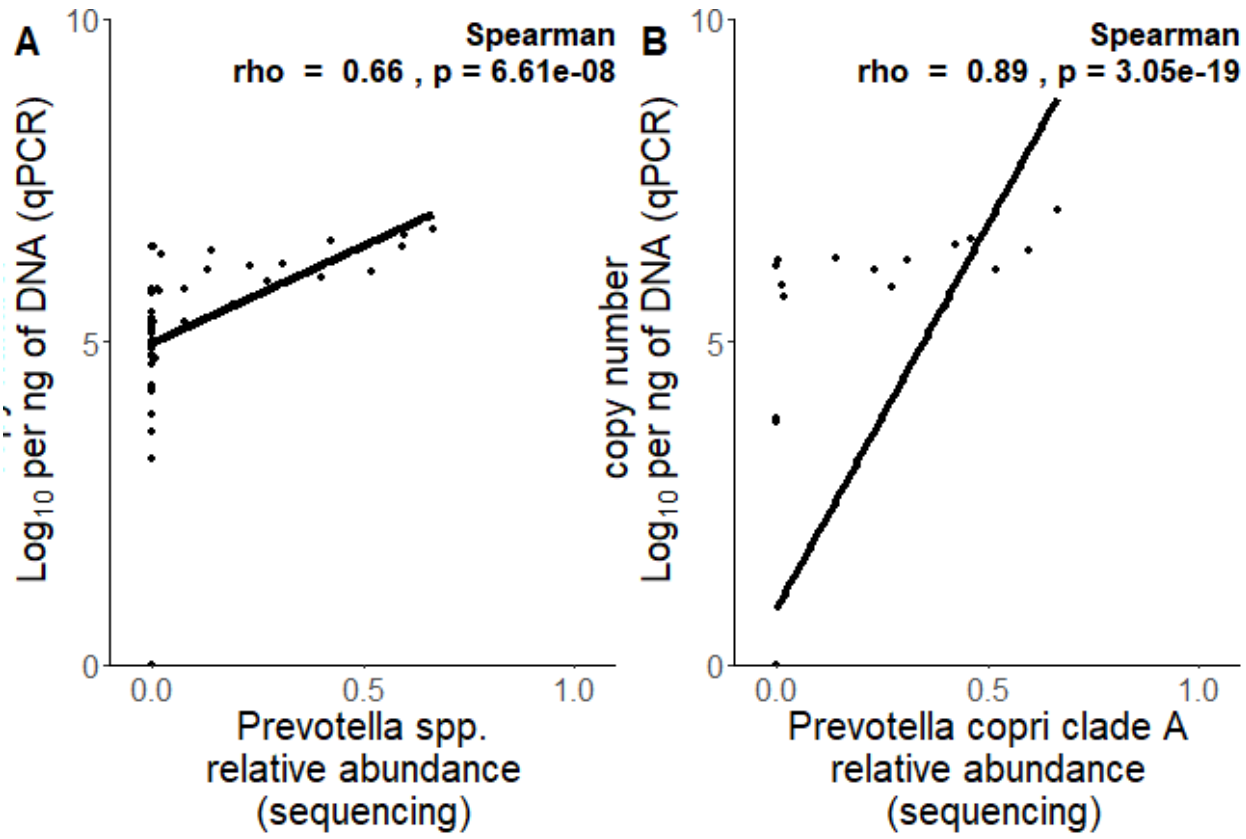
```
phyloseq_relab <- transform_sample_counts(phyloseq_Species_2, function(x)
  x / sum(x))
otu_table_relab <- as.data.frame(otu_table(phyloseq_relab))

taxonomy_table <- as.data.frame(phyloseq_Species_2@tax_table)

prevotella_row <- which(taxonomy_table$Species == "s__Prevotella_copri_clade_A")
prevotella_abundance <- otu_table_relab[prevotella_row, ]
prevotella_abundance <- as.data.frame(t(prevotella_abundance))
colnames(prevotella_abundance) <- "s__Prevotella_copri_clade_A"
prevotella_abundance$Sample <- rownames(prevotella_abundance)

metadata_with_prevotella <- metadata_with_prevotella %>%
  left_join(prevotella_abundance, by = "Sample")
```

Then perform the correlation plots.

```
Prevotella_spp_correlation <- perform_correlation(
  DATAFRAME = metadata_with_prevotella,
  VARIABLE_X = "g__Prevotella",
  VARIABLE_Y = "Prevotella_spp_qPCR_copies",
  LABEL_X = "Prevotella spp.\nrelative abundance\n(sequencing)",
  output_folder = output_folder,
LABEL_Y = expression(Log[10]~"copy number\nper ng of DNA (qPCR)"),
  BREAKS_X = c(0, 0.5, 1),
  BREAKS_Y = c(0, 5, 10),
  LIMITS_X = c(0, 1),
  LIMITS_Y = c(0, 10),
  EXPAND_X = c(0, 0.1),
  EXPAND_Y = c(0, 0)
)

Prevotella_copri_correlation <- perform_correlation(
  DATAFRAME = metadata_with_prevotella,
  VARIABLE_X = "s__Prevotella_copri_clade_A",
  VARIABLE_Y = "Prevotella_copri_qPCR_copies",
  LABEL_X = "Prevotella copri clade A\nrelative abundance\n (sequencing)",
  output_folder = output_folder,
LABEL_Y = expression(Log[10]~"copy number\nper ng of DNA (qPCR)"),
  BREAKS_X = c(0, 0.5, 1),
  BREAKS_Y = c(0, 5, 10),
  LIMITS_X = c(0, 1),
  LIMITS_Y = c(0, 10),
  EXPAND_X = c(0, 0.1),
  EXPAND_Y = c(0, 0)
)

if ("Response" %in% comparisons) {
  panel_Prevotella_correlation_qPCR <- plot_grid(
    Prevotella_spp_correlation,
    Prevotella_copri_correlation,
    ncol = 2,
        labels = "AUTO",
```

```
    label_size = 20
  )

  Cairo::Cairo(
    26,
    15,
    file = paste0(output_folder, "final_plots/Figure Sup. 2", ".tif"),
    type = "tif",
    bg = "transparent",
    dpi = 600,
    units = "cm"
  )

  plot(panel_Prevotella_correlation_qPCR)
  dev.off()
}
```

```
  plot(panel_Prevotella_correlation_qPCR) # Sequencing and qPCR are correlated
```



We also generate the correlation matrix (Figure 4)

```
biomarkers <- c(
  "Prevotella_spp_qPCR_copies",
  "PNE",
  "PNB",
  "Serum_DCA",
```

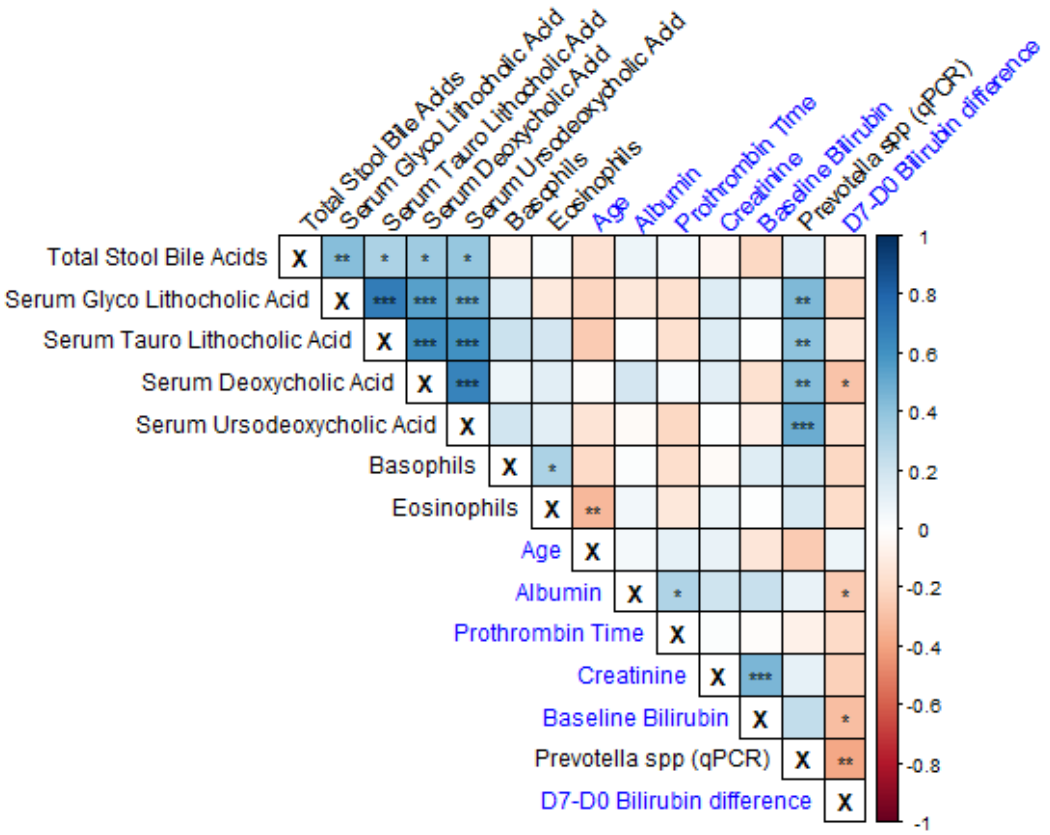
```
  "Serum_UDCA",
  "Serum_Glyco_LCA",
  "Serum_Tauro_LCA",
  "Stool_Total"
)
lille_components <- c("Age",
                      "Albumin",
                      "Creatinin",
                      "Prothrombin",
                      "Bilirubin",
                      "Delta_bilirubin")

variable_labels <- c(
  "Prevotella_spp_qPCR_copies" = "Prevotella spp (qPCR)",
  "PNE" = "Eosinophils",
  "PNB" = "Basophils",
  "Serum_DCA" = "Serum Deoxycholic Acid",
  "Serum_UDCA" = "Serum Ursodeoxycholic Acid",
  "Serum_Glyco_LCA" = "Serum Glyco Lithocholic Acid",
  "Serum_Tauro_LCA" = "Serum Tauro Lithocholic Acid",
  "Stool_Total" = "Total Stool Bile Acids",
  "Age" = "Age",
  "Albumin" = "Albumin",
  "Creatinin" = "Creatinine",
  "Prothrombin" = "Prothrombin Time",
  "Bilirubin" = "Baseline Bilirubin",
  "Delta_bilirubin" = "D7-D0 Bilirubin difference"
)
correlation_data <- metadata[, c(biomarkers, lille_components)]

test_normality <- function(variable) {
  if (length(na.omit(variable)) > 3) {
    return(shapiro.test(variable)$p.value)
  } else {
    return(NA)
  }
}

normality_results <- sapply(correlation_data, test_normality)

cor_method <- ifelse(normality_results < 0.05, "spearman", "pearson")

cor_matrix <- matrix(
  NA,
  ncol = ncol(correlation_data),
  nrow = ncol(correlation_data),
  dimnames = list(colnames(correlation_data), colnames(correlation_data))
)

p_matrix <- cor_matrix

for (i in 1:(ncol(correlation_data) - 1)) {
  for (j in (i + 1):ncol(correlation_data)) {
```

```
    method <- ifelse(cor_method[i] == "spearman" |
                       cor_method[j] == "spearman",
                     "spearman",
                     "pearson")

    test <- cor.test(correlation_data[, i],
                     correlation_data[, j],
                     method = method,
                     use = "pairwise.complete.obs")

    cor_matrix[i, j] <- test$estimate
    cor_matrix[j, i] <- test$estimate

    p_matrix[i, j] <- test$p.value
    p_matrix[j, i] <- test$p.value
  }
}

stars <- ifelse(p_matrix < 0.001, "***", ifelse(p_matrix < 0.01, "**", ifelse(p_matrix < 0.05, "*", ""))

cor_results <- data.frame(
  Variable_1 = character(),
  Variable_2 = character(),
  Method = character(),
  Correlation_Coefficient = numeric(),
  P_Value = numeric(),
  stringsAsFactors = FALSE
)

for (i in 1:(ncol(correlation_data) - 1)) {
  for (j in (i + 1):ncol(correlation_data)) {
    method <- ifelse(cor_method[i] == "spearman" |
                       cor_method[j] == "spearman",
                     "Spearman",
                     "Pearson")

    cor_results <- rbind(
      cor_results,
      data.frame(
        Variable_1 = colnames(correlation_data)[i],
        Variable_2 = colnames(correlation_data)[j],
        Method = method,
        Correlation_Coefficient = cor_matrix[i, j],
        P_Value = p_matrix[i, j]
      )
    )
  }
}

write.csv(
  cor_results,
  paste0(output_folder, "correlation_summary.csv"),
  row.names = FALSE,
```

```
  sep = "\t"
)




if ("Response" %in% comparisons) {
  rownames(cor_matrix) <- variable_labels[rownames(cor_matrix)]
  colnames(cor_matrix) <- variable_labels[colnames(cor_matrix)]
  rownames(p_matrix) <- variable_labels[rownames(p_matrix)]
  colnames(p_matrix) <- variable_labels[colnames(p_matrix)]

  dist_matrix <- as.dist(1 - abs(cor_matrix))
  clustering <- hclust(dist_matrix, method = "ward.D2")
  ordered_vars <- clustering$labels[clustering$order]

  # Reorder matrices
  cor_matrix <- cor_matrix[ordered_vars, ordered_vars]
  p_matrix <- p_matrix[ordered_vars, ordered_vars]

  # Color vector for labels
  tl.col.vector <- sapply(ordered_vars, function(var) {
    original_var <- names(variable_labels)[which(variable_labels == var)]
    if (original_var %in% biomarkers) {
      return("black")
    } else if (original_var %in% lille_components) {
      return("blue")
    } else {
      return("black")
    }
  })

  Cairo::Cairo(
    26,
    15,
    file = paste0(output_folder, "final_plots/Figure 4", ".tif"),
    type = "tif",
    bg = "transparent",
    dpi = 600,
    units = "cm"
  )

  corrplot(
    cor_matrix,
    p.mat = p_matrix,
    method = 'color',
    type = "upper",
    sig.level = c(0.001, 0.01, 0.05),
    insig = "label_sig",
    order = "original",
    rect.col = "black",
    rect.lwd = 1.5,
    pch.cex = 0.9,
```

```
    tl.col = tl.col.vector,  # pass your color vector here
    na.label = "X",
    addgrid.col = "black",
    tl.srt = 45,
    diag = TRUE
  )
  dev.off()
}
```

```
corrplot(
    cor_matrix,
    p.mat = p_matrix,
    method = 'color',
    type = "upper",
    sig.level = c(0.001, 0.01, 0.05),
    insig = "label_sig",
    order = "original",
    rect.col = "black",
    rect.lwd = 1.5,
    pch.cex = 0.9,
    tl.col = tl.col.vector,
    na.label = "X",
    addgrid.col = "black",
    tl.srt = 45,
    diag = TRUE)
```



Also perform the bootstrap analysis

```r
boot_spearman <- function(var1, var2, data, R = 1000) {
  fun <- function(d, i) {
    d2 <- d[i, ]
    return(cor(d2[[var1]], d2[[var2]], method = "spearman", use = "pairwise.complete.obs"))
  }
  set.seed(626)
  boot_obj <- boot(data, statistic = fun, R = R)
  ci <- boot.ci(boot_obj, type = "norm")
  ci_low <- if (!is.null(ci$normal)) ci$normal[2] else NA
  ci_high <- if (!is.null(ci$normal)) ci$normal[3] else NA
  list(correlation = boot_obj$t0, ci_lower = ci_low, ci_upper = ci_high)
}

# Define variable pairs
cor_pairs <- list(
  list(var1 = "Age", var2 = "PNE"),
  list(var1 = "Prevotella_spp_qPCR_copies", var2 = "Delta_bilirubin"),
  list(var1 = "Serum_DCA", var2 = "Delta_bilirubin")
)

# Ensure variables are numeric
correlation_data$Prevotella_spp_qPCR_copies <- as.numeric(correlation_data$Prevotella_spp_qPCR_copies)
correlation_data$Serum_DCA <- as.numeric(correlation_data$Serum_DCA)
correlation_data$Delta_bilirubin <- as.numeric(correlation_data$Delta_bilirubin)

# Prepare results table
cor_results <- data.frame()

for (pair in cor_pairs) {
  var1 <- pair$var1
  var2 <- pair$var2

  boot_result <- boot_spearman(var1, var2, correlation_data)

  test <- cor.test(correlation_data[[var1]], correlation_data[[var2]], method = "spearman", use = "pair

  # Append to table
  cor_results <- rbind(
    cor_results,
    data.frame(
      Variable1 = var1,
      Variable2 = var2,
      Spearman_rho = round(boot_result$correlation, 3),
      CI_lower_Norm = round(boot_result$ci_lower, 3),
      CI_upper_Norm = round(boot_result$ci_upper, 3),
      p_value = signif(test$p.value, 3)
    )
  )
}

# Export to CSV using correct arguments
write.csv(
  cor_results,
```

```
  file = paste0(output_folder, "bootstrap_results.csv"),
  row.names = FALSE
)
```

```
print(cor_results)
```

```
##                    Variable1       Variable2 Spearman_rho CI_lower_Norm
## 1                        Age             PNE       -0.328        -0.577
## 2 Prevotella_spp_qPCR_copies Delta_bilirubin       -0.390        -0.673
## 3                  Serum_DCA Delta_bilirubin       -0.284        -0.526
##   CI_upper_Norm p_value
## 1        -0.082 0.00864
## 2        -0.118 0.00393
## 3        -0.046 0.04800
```

```
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=French_France.utf8  LC_CTYPE=French_France.utf8
## [3] LC_MONETARY=French_France.utf8 LC_NUMERIC=C
## [5] LC_TIME=French_France.utf8
##
## time zone: Europe/Paris
## tzcode source: internal
##
## attached base packages:
## [1] tools     grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] DescTools_0.99.59      boot_1.3-31            microbiome_1.28.0
##  [4] yarrr_0.1.5            circlize_0.4.16        BayesFactor_0.9.12-4.7
##  [7] Matrix_1.7-1           coda_0.19-4.1          jpeg_0.1-10
## [10] corrplot_0.95          cowplot_1.1.3          vegan_2.6-10
## [13] lattice_0.22-6         permute_0.9-7          ape_5.8-1
## [16] stringr_1.5.1          Maaslin2_1.20.0        RVAideMemoire_0.9-83-7
## [19] rstatix_0.7.2          ggpubr_0.6.0           ggplot2_3.5.1
## [22] purrr_1.0.4            tidyr_1.3.1            openxlsx_4.2.8
## [25] dplyr_1.1.4            gridExtra_2.3          phyloseq_1.50.0
## [28] here_1.0.1
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3      rstudioapi_0.17.1       jsonlite_1.8.9
##   [4] shape_1.4.6.1           magrittr_2.0.3          farver_2.1.2
##   [7] rmarkdown_2.29          GlobalOptions_0.1.2     zlibbioc_1.52.0
```

```
##  [10] vctrs_0.6.5            multtest_2.62.0         Cairo_1.6-2
##  [13] forcats_1.0.0          htmltools_0.5.8.1       haven_2.5.4
##  [16] broom_1.0.8            cellranger_1.1.0        Rhdf5lib_1.28.0
##  [19] Formula_1.2-5          rhdf5_2.50.2            plyr_1.8.9
##  [22] rootSolve_1.8.2.4      igraph_2.1.4            lifecycle_1.0.4
##  [25] iterators_1.0.14       pkgconfig_2.0.3         R6_2.6.1
##  [28] fastmap_1.2.0          GenomeInfoDbData_1.2.13 digest_0.6.37
##  [31] Exact_3.3              colorspace_2.1-1        S4Vectors_0.44.0
##  [34] rprojroot_2.0.4        labeling_0.4.3          httr_1.4.7
##  [37] abind_1.4-8            mgcv_1.9-1              compiler_4.4.1
##  [40] proxy_0.4-27           withr_3.0.2             backports_1.5.0
##  [43] carData_3.0-5          DBI_1.2.3               biglm_0.9-3
##  [46] ggsignif_0.6.4         MASS_7.3-64             biomformat_1.34.0
##  [49] gld_2.6.7              optparse_1.7.5          zip_2.3.2
##  [52] glue_1.8.0             nlme_3.1-167            rhdf5filters_1.18.0
##  [55] Rtsne_0.17             cluster_2.1.8           reshape2_1.4.4
##  [58] ade4_1.7-22            generics_0.1.3          gtable_0.3.6
##  [61] class_7.3-23           hms_1.1.3               data.table_1.16.4
##  [64] lmom_3.2               car_3.1-3               XVector_0.46.0
##  [67] BiocGenerics_0.52.0    foreach_1.5.2           pillar_1.10.2
##  [70] logging_0.10-108       robustbase_0.99-4-1     splines_4.4.1
##  [73] getopt_1.20.4          survival_3.8-3          tidyselect_1.2.1
##  [76] Biostrings_2.74.1      pbapply_1.7-2           knitr_1.50
##  [79] IRanges_2.40.1         stats4_4.4.1            xfun_0.52
##  [82] expm_1.0-0             Biobase_2.66.0          pheatmap_1.0.12
##  [85] DEoptimR_1.1-3-1       stringi_1.8.4           UCSC.utils_1.2.0
##  [88] yaml_2.3.10            evaluate_1.0.3          codetools_0.2-20
##  [91] tibble_3.2.1           hash_2.2.6.3            BiocManager_1.30.25
##  [94] cli_3.6.3              Rcpp_1.0.14             GenomeInfoDb_1.42.3
##  [97] readxl_1.4.3           parallel_4.4.1          MatrixModels_0.5-4
## [100] mvtnorm_1.3-3          scales_1.4.0            e1071_1.7-16
## [103] pcaPP_2.0-5            crayon_1.5.3            rlang_1.1.5
```