Faecal microbiota and metabolome analysis of dogs affected by CRGV

Whitehouse *et al*. ‘The Gut Microbiota and Metabolome of Cutaneous and Renal Glomerular Vasculopathy affected Dogs is Characteristic of Early-onset Starvation’.

This document contains all of the post-processing analysis conducted for the manuscript in R (16S rRNA amplicon sequencing and proton nuclear magnetic resonance) and MatLab (proton nuclear magnetic resonance). Note that some MatLab scripts used to call in functions are not provided due to their sensitive nature within external research groups.

NB. The scripts are named 1 – 6 and can be run separately once the phyloseq object is created. Script 7 requires information generated from the proton nuclear magnetic resonance data and 16S rRNA gene sequencing data.

All data to reproduce analysis can be found here: https://github.com/JWhitehouse-gut/CRGV_16S_1HNMR.

16S rRNA amplicon sequencing

```
################################################################

# Script 1: Data import and basic preprocessing

# Description: Import phyloseq object, clean data, and

# prepare counts and metadata for downstream analyses

################################################################

# load required libraries

library(qiime2R)

library(phyloseq)

library(phyloseqCompanion)

# load required qimee2 artifacts

physeq<-qza_to_phyloseq(features="R_data/table.qza", tree="R_data/rooted_tree.qza", "R_data/taxonomy.qza", metadata = "R_data/metadata.tsv")

# check the phyloseq object structure

physeq

# optional: check the number of bacterial families present in the physeq object

families=tax_table(physeq)[,'Family']

families=families[!is.na(families)]

length(families)

length(unique(families))
```

```
# check the consistency of the physeq object

metadata=sample_data(physeq)

metadata[1:5,]

# take a look at the raw counts

countMatrix=t(otu.matrix(otu_table(physeq)))

countMatrix[1:5,1:5]

# take a look at the taxonomy matrix

tax_matrix=tax_table(physeq)

tax_matrix[1:5,]

# aggregate the physeq object to the family level

physeq=phyloseq::tax_glom(physeq,taxrank='Family', NArm=FALSE)

# extract all the elements from the phyloseq object

countMatrix=t(otu.matrix(otu_table(physeq)))

metadata=data.frame(sample_data(physeq))

TaxaTable=tax_table(physeq)

# modify metadata table

new_metadata=read.csv('Metadata_3_V1_JW_test.csv',row.names=1)

dim(metadata)

dim(new_metadata)

setdiff(colnames(countMatrix),rownames(new_metadata))

setdiff(rownames(new_metadata),colnames(countMatrix))

new_metadata=new_metadata[colnames(countMatrix),]

identical(rownames(new_metadata),colnames(countMatrix))

metadata=new_metadata

# re-generate phyloseq object

my_otu_table=otu_table(countMatrix,taxa_are_rows=TRUE)

my_sample_data=sample_data(metadata)

my_taxonomyTable=tax_table(TaxaTable)

physeq=phyloseq(my_otu_table,my_sample_data,my_taxonomyTable)

save(physeq,file='R_objects/physeq.RData')
```

```
###########################################################
# Script 2: Normalise phyloseq object and explore count depth
###########################################################
# load libraries
library(phyloseq)
library(phyloseqCompanion)
library(magrittr)
# load phyloseq object
physeq_file <- "R_objects/physeq.RData"
load(physeq_file)
metadata <- data.frame(sample_data(physeq))
counts_matrix <- t(otu.matrix(otu_table(physeq)))
stopifnot(identical(colnames(counts_matrix), rownames(metadata)))
# total read counts per sample
total_counts <- signif(sort(colSums(counts_matrix)), 4)
# define filtering threshold
cutoff_percentile <- 0.05
cutoff_value <- quantile(total_counts, cutoff_percentile)
samples_keep   <- names(total_counts[total_counts > cutoff_value])
samples_delete <- names(total_counts[total_counts <= cutoff_value])
plot_title <- paste(
  "Total samples =", length(total_counts),
  "| cutoff percentile =", cutoff_percentile,
  "| keep =", length(samples_keep),
  "| delete =", length(samples_delete)
)
# plot sequencing depth
pdf("Total_counts_per_sample.pdf", width = 10, height = 2)
par(mar = c(6, 4, 1, 1), cex = 0.6)
plot(
  x = seq_along(total_counts),
  y = total_counts,
  xlab = "",
  ylab = "Total read counts",
  axes = FALSE,
  main = plot_title
)
abline(h = seq(0, max(total_counts), by = 5000), col = "lightgray")
abline(h = median(total_counts), col = "red")
abline(h = quantile(total_counts, c(0.05, 0.10, 0.90, 0.95)), col = "red", lty = 2)
abline(h = cutoff_value, col = "green", lty = 2)
axis(1, at = seq_along(total_counts), labels = names(total_counts), las = 2)
axis(2)
dev.off()
# filter samples by sequencing depth
filtered_physeq <- physeq %>% subset_samples(SampleID %in% samples_keep)
# sanity check
sample_data(filtered_physeq)
# save filtered phyloseq object
save(filtered_physeq, file = "R_objects/physeq_filtered.RData")
# explore total counts vs metadata
total_counts <- signif(colSums(counts_matrix), 4)
stopifnot(identical(names(total_counts), rownames(metadata)))
test_variables <- c("Sex", "Neuter_status", "Breed_Class_1", "Age_class_1", "Cohort")
pdf("Total_counts_boxplots.pdf", width = 5, height = 6)
par(mfrow = c(3, 2), mar = c(6, 4, 1, 1), cex = 0.6)
for (var in test_variables) {
  boxplot(total_counts ~ metadata[[var]],xlab = var,ylab = "Total read counts")
dev.off()
```

```
# selected boxplots

pdf("Total_counts_breed_boxplots.pdf", width = 5, height = 6)

par(mfrow = c(3, 1), mar = c(6, 4, 2, 1), cex = 0.6)

boxplot(total_counts ~ metadata$Cohort, xlab = "", ylab = "Total read counts", las = 2)

boxplot(total_counts ~ metadata$Sex, xlab = "", ylab = "Total read counts", main = "All samples")

boxplot(total_counts ~ metadata$Breed_Class_1, xlab = "", ylab = "Total read counts", main = "All samples")

dev.off()

# rarefaction normalisation

min_sample_size <- min(sample_sums(filtered_physeq))

norm_physeq <- rarefy_even_depth(

  filtered_physeq,

  sample.size = min_sample_size,

  rngseed = 1,

  replace = FALSE,

  trimOTUs = TRUE,

  verbose = TRUE)

# save normalised phyloseq object

save(norm_physeq, file = "R_objects/norm_phyloseq.RData")

############################################################

# Script 3: Calculate alpha diversity with QC checks

############################################################

# load libraries

library(phyloseq)

library(phyloseqCompanion)

library(stringr)

library(data.table)

library(car)

# create output directory

out_dir <- "Diversity"

if (!dir.exists(out_dir)) dir.create(out_dir)
```

```
# load normalised phyloseq object

load("R_objects/norm_phyloseq.RData")

# sanity check: sequencing depth

counts_matrix <- t(otu.matrix(otu_table(norm_physeq)))

stopifnot(length(unique(colSums(counts_matrix))) == 1)

# extract metadata

metadata <- data.frame(sample_data(norm_physeq))

metadata$cohort <- factor(metadata$Cohort)

# calculate alpha diversity

diversity_matrix <- estimate_richness(norm_physeq)

# clean row names (remove leading 'X')

rownames(diversity_matrix) <- str_replace(rownames(diversity_matrix), "^X", "")

# save diversity matrix

write.csv(diversity_matrix, file = file.path(out_dir, "diversity_matrix.csv"))

# plot diversity vs metadata

diversity_metrics <- c("Observed", "Chao1", "ACE", "Shannon", "Simpson", "InvSimpson", "Fisher")

selected_variables <- c("Cohort", "Sex", "Age_class_1", "Breed_Class_1", "Neuter_status")

selected_variable_names <- c(Cohort = "Cohort", Sex = "Sex", Age_class_1 = "Age", Breed_Class_1 =
"Breed class", Neuter_status = "Neuter status")

pdf("Diversity_and_Metadata.pdf", width = 7, height = 4)

par(mfrow = c(1, 3), cex = 0.6, mar = c(15, 5, 3, 3))

for (diversity in diversity_metrics) {Y <- diversity_matrix[, diversity] names(Y) <-
rownames(diversity_matrix)

for (var in selected_variables) { data <- data.frame(sample_data(norm_physeq))

# linear model + ANOVA

formula <- as.formula(paste("Y ~", var))

model <- lm(formula, data = data)

aov_res <- car::Anova(model)

# group sizes

totals <- summary(factor(data[[var]]))
levels_var <- levels(factor(data[[var]]))

x_labels <- sapply(levels_var, function(level) {bquote(.(level) ~ italic(n) == .(totals[level]))})

p_label <- bquote(italic(p) == .(signif(aov_res$`Pr(>F)`[1], 2)))

# boxplot

boxplot(Y ~ data[[var]], ylab = diversity, las = 2, main = selected_variable_names[[var]], xlab = "", names
= do.call(expression, x_labels))

mtext(side = 3, text = p_label, cex = 0.6)

# jittered points

stripchart(Y ~ data[[var]], method = "jitter", pch = 19, cex = 0.7, col = "darkorange", vertical = TRUE, add
= TRUE)

 }

}

dev.off()

# QC plot: check factor level ordering

pdf("Diversity_and_Metadata_forQC.pdf", width = 7, height = 4)

par(mfrow = c(1, 3), cex = 0.6, mar = c(15, 3, 3, 3))

diversity <- "Observed"

Y <- diversity_matrix[, diversity]

names(Y) <- rownames(diversity_matrix)

qc_variable <- "Breed_Class_1"

data <- data.frame(sample_data(norm_physeq))

model <- lm(Y ~ data[[qc_variable]])

aov_res <- car::Anova(model)

p_label <- paste("p =", signif(aov_res$`Pr(>F)`[1], 2))

totals <- summary(factor(data[[qc_variable]]))

levels_var <- levels(factor(data[[qc_variable]]))

x_labels <- paste(levels_var, "n =", totals[levels_var])

print(x_labels)

boxplot( Y ~ data[[qc_variable]], ylab = diversity, las = 2, main = paste(qc_variable, p_label), xlab = "")

dev.off()
```

```r
##############################################################
# Script 4: Calculate beta diversity (NMDS)
##############################################################
# helper functions
get_difference_angle <- function(angle1, angle2) {
  n <- length(angle1)
  angles <- numeric(n)
  for (i in seq_len(n)) {
    x1 <- cos(angle1[i] * pi / 180)
    y1 <- sin(angle1[i] * pi / 180)
    x2 <- cos(angle2[i] * pi / 180)
    y2 <- sin(angle2[i] * pi / 180)
    d12 <- sqrt((x1 - x2)^2 + (y1 - y2)^2)
    angles[i] <- acos((-(d12^2) + 2) / 2) * 180 / pi
  }
  angles
}
get_angle_from_xy <- function(x, y) {
  angle <- atan(y / x) * 360 / (2 * pi)
  out <- numeric(length(x))
  out[x >  0 & y >  0] <- angle[x >  0 & y >  0]
  out[x <= 0 & y >  0] <- 180 + angle[x <= 0 & y >  0]
  out[x <= 0 & y <= 0] <- 180 + angle[x <= 0 & y <= 0]
  out[x >  0 & y <= 0] <- 360 + angle[x >  0 & y <= 0]
  out
}
# load libraries
library(vegan)
library(phyloseq)
library(phyloseqCompanion)
library(shape)

library(stringr)

source("R_script/functions/piecewise_kn_V1.R")

# load normalised phyloseq object

load("R_objects/norm_phyloseq.RData")

# identify outliers (optional / slow)

dist_mat <- distance(norm_physeq, method = "bray", type = "samples")

k <- 0.05 * nsamples(norm_physeq)

outlier_test <- piecewise_kn_V1(

  dist_mat,

  test.ix = rownames(sample_data(norm_physeq)),

  k = k)

outliers <- rownames(sample_data(norm_physeq))[outlier_test$pvals < 0.05]

outliers = "n/a"

# extract data from phyloseq

count_matrix <- otu.matrix(otu_table(norm_physeq))

tax_table_df <- data.frame(tax_table(norm_physeq))

stopifnot(identical(rownames(tax_table_df), colnames(count_matrix)))

taxa_names <- str_replace(tax_table_df$Family, "^f_", "")

names(taxa_names) <- rownames(tax_table_df)

metadata <- data.frame(sample_data(norm_physeq))

# define colour schemes

breed_class_colors <- c(Crossbreed = "#8B0000", Gundog    = "#00008B", Hound     = "#8B8000", Pastoral   = "#ff8c00", Terrier    = "#301934", Utility    = "#014D4E", Working    = "#AA336A", Unknown    = "#B5B5B5")

age_class_colors <- c("Early Senior"  = "#8B0000", Geriatric     = "#00008B", Juvenile      = "#8B8000", "Late Senior"   = "#006400", "Mature Adult" = "#ff8c00", "Young Adult"   = "#301934", Unknown       = "#014D4E")

cohort_colors <- c(CRGV = "darkorange", Healthy = "darkblue")

sex_colors <- c(Female = "#AA336A", Male = "#00008B", Unknown = "darkorange")
```

```
# filter taxa (present in ≥10 samples)

presence <- count_matrix

presence[presence > 0] <- 1

taxa_keep <- colSums(presence) > 9

count_matrix <- count_matrix[, taxa_keep]

# NMDS dimensionality check

stress <- numeric(4)

for (k_dim in 1:4) {

  set.seed(465)

  stress[k_dim] <- metaMDS(count_matrix, k = k_dim, distance = "bray")$stress}

plot(stress, xlab = "k (dimensions)", ylab = "Stress")

abline(h = 0.2)

# run NMDS

set.seed(654)

nmds <- metaMDS(count_matrix, distance = "bray")

plot(nmds)

stressplot(nmds)

plot(goodness(nmds))

data_scores <- scores(nmds)$sites

# fit taxa vectors

envfit_res <- envfit(nmds, count_matrix, permutations = 999)

ax <- 1.2

vector_scores <- scores(envfit_res, "vectors") * ax

r_sorted <- sort(envfit_res$vectors$r, decreasing = TRUE)

pvals <- envfit_res$vectors$pvals

# select taxa to plot (angle-based)

candidates <- names(pvals[pvals < 0.05])

r_sorted <- sort(envfit_res$vectors$r[candidates], decreasing = TRUE)

candidates <- names(r_sorted)
```

```
angles <- get_angle_from_xy(
  vector_scores[candidates, 1],
  vector_scores[candidates, 2])
names(angles) <- candidates
toplot <- candidates[1]
for (i in 2:length(angles)) {
  diffs <- sapply(toplot, function(x)
    get_difference_angle(angles[i], angles[x]))
  if (min(diffs) > 30) {
    toplot <- c(toplot, names(angles[i]))
  }
}
# NMDS plots by metadata
color_maps <- list(Breed_Class = breed_class_colors, Age_class = age_class_colors, Cohort = cohort_colors, Sex = sex_colors)
for (var in names(color_maps)) {
  mycolors <- color_maps[[var]]
  values <- as.character(metadata[[var]])
  values[is.na(values)] <- "Unknown"
  point_colors <- mycolors[values]
  pdf(paste0("NMDS_coloured_by_", var, ".pdf"), width = 3, height = 3)
  par(mar = c(2, 2, 0, 0), cex = 0.6)
  lim <- range(data_scores) + c(-0.3, 0.3)
  plot(lim, lim, type = "n", axes = FALSE, xlab = "", ylab = "")
  axis(1); axis(2)
  abline(h = 0, v = 0, col = "lightgray", lty = 3)
  set.seed(1)
  ord <- sample(seq_len(nrow(data_scores)))
  points(data_scores[ord, 1], data_scores[ord, 2], col = point_colors[ord], pch = 19, cex = 0.55)
  # add taxa arrows
  arrows(0, 0, vector_scores[toplot, 1], vector_scores[toplot, 2], length = 0.08)
```

```
 text(vector_scores[toplot, 1] * 1.1, vector_scores[toplot, 2] * 1.1, taxa_names[toplot], cex = 0.5)
 dev.off()
 # legend
 pdf(paste0("NMDS_", var, "_key.pdf"), width = 0.8, height = 3)
 par(mar = c(0, 0, 0, 0), cex = 0.4)
 plot.new()
 legend("topleft", names(mycolors), col = mycolors, pch = 19, bty = "n")
 dev.off()}
############################################################
# Script 5: Relative abundance plots
############################################################
# load libraries
library(phyloseq)
library(dplyr)
library(tidyr)
library(ggplot2)
# load phyloseq object
load("R_objects/physeq.RData")
# parameters
families_to_plot <- c("Enterobacteriaceae", "Enterococcaceae", "Prevotellaceae", "Veillonellaceae")
cohort_col <- "Cohort"
cohort_colors <- c("CRGV-affected" = "#F68D1F", "MH" = "#272D7D")
# extract metadata
metadata <- as.data.frame(sample_data(physeq))
metadata$SampleID <- rownames(metadata)
# rename cohort levels
metadata[[cohort_col]] <- recode(metadata[[cohort_col]], "CRGV"     = "CRGV-affected", "Healthy" = 
"MH")
# extract taxonomy and count data
taxonomy <- as.data.frame(tax_table(physeq))
```

```
count_matrix <- t(otu.matrix(otu_table(physeq)))

total_counts <- colSums(count_matrix, na.rm = TRUE)

# calculate relative abundance

rel_abund <- sweep(count_matrix, 2, total_counts, "/")

# collect data for selected families

plot_data <- lapply(families_to_plot, function(family) {

 taxa_ids <- rownames(taxonomy)[taxonomy$Family == family]

 taxa_ids <- taxa_ids[!is.na(taxa_ids)]

 if (length(taxa_ids) == 0) {

  message("Family not found: ", family)

  return(NULL)}

 abundance <- colSums(rel_abund[taxa_ids, , drop = FALSE], na.rm = TRUE)

 data.frame(SampleID = names(abundance), abundance = abundance, Family = family)

}) |>

 bind_rows() |>

 left_join(metadata, by = "SampleID")

# plot relative abundance

ggplot(plot_data, aes(x = .data[[cohort_col]], y = abundance)) +

 geom_boxplot(

  aes(fill = .data[[cohort_col]]),

  outlier.shape = NA,

  alpha = 0.3,

  colour = "grey40"

 ) +

 geom_jitter(

  aes(colour = .data[[cohort_col]]),

  width = 0.2,

  size = 2.5,

  alpha = 0.9

 ) +
 facet_wrap(~ Family, scales = "free_y", ncol = 2) +
 scale_fill_manual(values = cohort_colors) +
 scale_color_manual(values = cohort_colors) +
 labs(
  x = "Cohort",
  y = "Relative abundance",
  title = "Family-level relative abundance across cohorts"
 ) +
 theme_bw(base_size = 14) +
 theme(
  strip.text = element_text(size = 13, face = "italic"),
  legend.position = "none",
  axis.text.x = element_text(angle = 30, hjust = 1)
 )
############################################################
# Script 6: ANCOM-BC2 global tests and significant taxa plots
############################################################
# load libraries
library(ANCOMBC)
library(microbiome)
library(mia)
library(phyloseqCompanion)
library(tidyverse)
library(DT)
# load filtered phyloseq object
load("R_objects/physeq_filtered.RData")
filtered_physeq
# inspect data
metadata <- data.frame(sample_data(filtered_physeq))
head(metadata)
```

```r
count_matrix <- t(otu.matrix(otu_table(filtered_physeq)))

count_matrix[1:5, 1:5]

tax_table_df <- data.frame(tax_table(filtered_physeq))

tax_table_df[1:5, ]

# extract taxonomy labels

taxa_labels <- paste(tax_table_df$Class, tax_table_df$Order, tax_table_df$Family, seq_len(nrow(tax_table_df)), sep = "_")

names(taxa_labels) <- rownames(tax_table_df)

# convert phyloseq → TreeSummarizedExperiment

tse <- mia::convertFromPhyloseq(filtered_physeq)

print(tse)

# factor re-levelling

tse$Breed_Class_1 <- factor(tse$Breed_Class_1, levels = c("Crossbreed", "Gundog", "Hound", "Pastoral", "Terrier", "Utility"))

tse$Age_class_1 <- factor(tse$Age_class_1, levels = c("Early Senior", "Geriatric", "Late Senior", "Mature Adult", "Puppy", "Young Adult", "Other"))

tse$Neuter_status <- factor(tse$Neuter_status, levels = c("Entire", "Neutered"))

tse$Sex <- factor(tse$Sex, levels = c("Female", "Male"))

tse$Cohort <- factor(tse$Cohort, levels = c("Healthy", "CRGV"))

# ANCOM-BC2 parameters

set.seed(658)

prv_cut <- 0.5

lib_cut <- 0

# Model 1:

output_1 <- ancombc2(

 data = tse,

 fix_formula = "Sex + Cohort + Breed_Class_1 + Age_class_1 + Neuter_status",

 group = "Cohort",

 p_adj_method = "BH",

 pseudo_sens = TRUE,

 prv_cut = prv_cut,
  lib_cut = lib_cut,
  n_cl = 2,
  struc_zero = TRUE,
  global = TRUE,
  pairwise = TRUE,
  dunnet = TRUE,
  trend = FALSE,
  verbose = TRUE
)
write.csv(cbind(output_1$res_global, output_1$res_pair), file = "output_1_ANCOMBC2.csv")
closeAllConnections()
# structural zero inspection
datatable(output_1$zero_ind, caption = "Structural zeros – model 1")
# extract results for plotting (model 1)
results <- output_1$res
rownames(results) <- results$taxon
comparison <- "CohortCRGV"
lfc <- results[[paste0("lfc_", comparison)]]
pvals <- results[[paste0("q_", comparison)]]
passed_ss <- results[[paste0("passed_ss_", comparison)]]
names(lfc) <- names(pvals) <- names(passed_ss) <- rownames(results)
# identify significant taxa
sig_taxa <- names(pvals[pvals < 0.05])
sig_pos <- intersect(sig_taxa, names(lfc[lfc > 0]))
sig_neg <- intersect(sig_taxa, names(lfc[lfc < 0]))
sig_pos <- sig_pos[order(lfc[sig_pos])]
sig_neg <- sig_neg[order(lfc[sig_neg])]
toplot <- c(sig_pos, sig_neg)
# colours
bar_colors <- setNames(
  c(rep("darkorange", length(sig_pos)),
    rep("darkblue", length(sig_neg))),
  toplot)
# taxon labels
tax_labels <- tax_table(filtered_physeq)[toplot, "Family"]
tax_labels[is.na(tax_labels)] <- paste(
  tax_table(filtered_physeq)[toplot[is.na(tax_labels)], "Class"],
  "NA",
  sep = "_")
# sensitivity analysis status
passed <- toplot[passed_ss[toplot]]
not_passed <- setdiff(toplot, passed)
# bar plot
pdf("CRGV_logFC_barplot_output_3.pdf", width = 9, height = 10)
par(mar = c(3, 5, 0, 0), oma = c(10, 0, 0, 0), cex = 1)
ylim <- range(lfc[toplot]) * 1.1
bars <- barplot(
  lfc[toplot],
  col = bar_colors,
  axes = FALSE,
  names.arg = FALSE,
  ylim = ylim)
axis(2)
mtext("Log2 fold change", side = 2, line = 2)
axis(1, at = bars[passed],
  labels = as.expression(lapply(tax_labels[passed], function(x) bquote(italic(.(x))))),
  col.axis = "green",
  tick = FALSE,
  las = 2)
```

```
axis(1, at = bars[not_passed],
 labels = as.expression(lapply(tax_labels[not_passed], function(x) bquote(italic(.(x))))),
 col.axis = "black",
 tick = FALSE,
 las = 2)
dev.off()
# save workspace
save.image("ANCOM_BC2_model_1.RData")...
############################################################
# Script 7: Generation of correlograms
############################################################
# load libraries
library(corrplot)
library(RColorBrewer)
# load data
data_file <- "R_data/integratee_B2_V1_JW.csv"
X <- read.csv(data_file, header = TRUE, row.names = 1)
# function: correlation p-values
cor_mtest <- function(mat, ...) {mat <- as.matrix(mat)
n <- ncol(mat)
p_mat <- matrix(NA, n, n)
diag(p_mat) <- 0
 for (i in 1:(n - 1)) {
  for (j in (i + 1):n) {
   test <- cor.test(mat[, i], mat[, j], ...)
   p_mat[i, j] <- p_mat[j, i] <- test$p.value}}
 colnames(p_mat) <- rownames(p_mat) <- colnames(mat)
 p_mat}
# compute correlations
cor_mat <- cor(X, method = "spearman")
```

```
p_mat <- cor_mtest(X, method = "spearman")

# colour palette

col_palette <- brewer.pal(n = 11, name = "RdYlBu")

col_palette <- rev(col_palette)

# hierarchical clustering

hc <- hclust(as.dist(1 - cor_mat), method = "complete")

plot(hc, main = "Hierarchical clustering of Spearman correlations")

plot(hc, hang = -1, main = "Hierarchical clustering (hang = -1)")

# correlogram: with labels

corrplot(cor_mat,

type = "upper",

 p.mat = p_mat,

 order = "hclust",

 hclust.method = "complete",

 col = col_palette,

 sig.level = 0.05,

 insig = "blank")

# correlogram: no text labels

corrplot(cor_mat, type = "upper", p.mat = p_mat, order = "hclust", hclust.method = "complete", tl.pos =
"n", col = col_palette, sig.level = 0.05, insig = "blank")

## save correlogram to PDF

pdf("corrplot_spearman_CRGV.pdf", width = 10, height = 10)

corrplot(cor_mat, type = "upper", p.mat = p_mat, order = "hclust", hclust.method = "complete", col =
col_palette, sig.level = 0.05, insig = "blank")

dev.off()
```

--**End of 16S rRNA gene sequencing script**---------------------------------

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Proton (1H) NMR data processing and multivariate analysis pipeline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% import and preprocess raw data

% ensure options.txt is present in the main experiment directory
```

```
spec_preproc_v5;

%% load spectral data

load('rawdata.mat')

Sp  = data.Spectra.Sp;

ppm = data.Spectra.ppm;

% reverse spectral direction

Sp  = fliplr(Sp);

ppm = fliplr(ppm);

plotSpectraCS(ppm, Sp);

%% remove poor-quality spectra

% samples with contaminant peaks

bad_samples = [27, 54, 59, 70];

Sp(bad_samples, :) = [];

plotSpectraCS(ppm, Sp);

%% remove unwanted spectral regions

% NOTE: ppm limits may vary between datasets — always verify visually

% Remove water region

water_idx = find(ppm >= 4.674, 1, 'first') : find(ppm >= 5.03, 1, 'first');

Sp(:, water_idx)  = [];

ppm(:, water_idx) = [];

plotSpectraCS(ppm, Sp);

%% remove TSP region

tsp_idx = find(ppm >= -0.5, 1, 'first') : find(ppm >= 0.5, 1, 'first');

Sp(:, tsp_idx)  = [];

ppm(:, tsp_idx) = [];

plotSpectraCS(ppm, Sp);

%% manual spectral alignment (first pass)

uiAlignment(ppm, Sp);

Xal = uiAlignmentData.SpAL;

%% manual spectral alignment (second pass)
```

```
uiAlignment(ppm, Xal);

Xal = uiAlignmentData.SpAL;

%% PQN normalisation

X = JTPnormalise(Xal, 'medianFold');

plotSpectraCS(ppm, X);

%% save pre-processed data

% ppm : chemical shift scale

% Sp  : raw, cut spectra

% Xal : aligned (unnormalised) spectra

% X   : aligned + PQN normalised spectra

save('Preprocessed')

%% PCA analysis

% create sample ID labels

n_samples = size(X, 1);

ID = (1:n_samples)';

% build PCA model (Pareto scaling, 4 components)

PCA = JTPcrossValidatedPCA(X, 'pa', 4);

% PCA score plots

pca_plotCS(PCA, ppm, X, 'plottype', 'scores', 'sample_labels', ID);

pca_plotCS(PCA, ppm, X, 'plottype', 'scores', 'Y', ID);

% moderate outliers observed: samples 39 and 52

%% PCA loadings

% PC1 loadings

colorplot(ppm, PCA.P(1,:) .* std(X), PCA.P(1,:).^2);

% PC2 loadings

colorplot(ppm, PCA.P(2,:) .* std(X), PCA.P(2,:).^2);

%% remove PCA outliers and rebuild PCA

X1 = X;

X1([39, 52], :) = [];

PCA = JTPcrossValidatedPCA(X1, 'pa', 2);
```

```
pca_plotCS(PCA, ppm, X1, 'plottype', 'scores');

save('workset')

%% OPLS-DA model

% Y must be defined prior to this step (class labels)

mjro2pls = mjrMainO2pls( ...

  X, Y, ...

  1, 1, 0, 7, ...

  'nfold', 'mc', 'no', [], ...

  'da', 'y', 'standard');

mjro2pls.cv.Q2Yhat

%% set colour map for plots

set(0, 'DefaultFigureColormap', feval('jet'))

%% OPLS-DA summary plot

mjrO2plsSummaryPlotJMP2(mjro2pls, X, Y, ppm);

%% permutation testing

% recommended permutations: ≥1000

[OPLSDA_pval, ~, ~, OPLSDA] = ...

  JTPpermutate(X1, Y, 1000, 1, 1, 7, 'mc');

%% correlation & covariance spectra

[corrVect, covVect] = ...

  mjrO2plsSummaryPlotJMP2(mjro2pls, X1, Y, ppm);

% with p-values

mjrO2plsSummaryPlotJMP2( ...

  mjro2pls, X1, Y, ppm, ...

  'pvalues', 'yes');

% significant peaks (no FDR)

mjrO2plsSummaryPlotJMP2( ...

  mjro2pls, X1, Y, ppm, ...

  'pvalues', 'yes', ...

  'mccplot', 'no_correlation');
```

```
% significant peaks (FDR corrected)
mjrO2plsSummaryPlotJMP2( ...
   mjro2pls, X1, Y, ppm, ...
   'pvalues', 'yes', ...
   'mccplot', 'S');
%% peak integration
% define integration regions and metabolite names
integration.region = [];      % [low_ppm high_ppm; ...]
integration.metabolites = {};   % {'Metabolite1','Metabolite2',...}
% integrate peaks
METAB = integrate_JMP(ppm, X1, integration.region);
% extract correlation and covariance vectors
[corrvect, covvect] = mjrO2plsSummaryPlotJMP(mjro2pls, X1, Y, ppm);
% integrate with correlation/covariance info
[METAB, summary] = integrate_JMP( ...
   ppm, X1, integration.region, corrvect, covvect);
```

-----------------------------------**End of proton nuclear magnetic resoance script**------------------------------