GapSeq_metabolic_pathway_analysis-FS

- Core and accessory metabolic pathways were separated across all samples:
    - An R script was used to convert the individual table files (.tbl) produced by gapSeq into one data frame where all the samples and pathways were combined into a true/false matrix:

```
# Load necessary library
library(dplyr)
library(readr)
library(tibble)
# Define the path to your files
path <- "~/Dropbox/microArc/Francesca/fileSwap/gapseq"
# Get a list of TSV files in the directory
files <- list.files(path = path, pattern = "-all-Pathways.tbl$", full.names = TRUE)#Create an
empty dataframe
allSamples <- data.frame(ID = character())# Loop through each file
for (file in files) {
  #read in the file, chop the first 3 lines off (as they are comments) and then separate into
tabs
  currentFile <- readLines(file)
  currentFile <- currentFile[-(1:3)]
  currentDF <- read.table(text = currentFile, sep = "\t")  #Place the 1st row as the column
names
  colnames(currentDF) <-currentDF[1,]
  currentDF <- currentDF[-1,]  # Extract the sample name from the file name
  sample_name <- gsub("-all-Pathways.tbl$", "", basename(file))  #Get just the first 3 columns
and rename the prediction column to the sample name
  currentDF_cols <- currentDF %>%
    select(ID, Prediction) %>%
    rename(!!sample_name := Prediction)  #join into the main datafram
  allSamples <- allSamples %>%
    merge(currentDF_cols, by = "ID", all = TRUE)
}
```

    - The true/false matrix was renamed 'pathways_matrix_output.tsv'.
    - Using R scripts, pathways that were true in all samples (core pathways) and pathways that were false in all samples (irrelevant) were made into their own files:

```
# Load necessary library
library(dplyr)

# Read the pathways matrix from the TSV file
pathways_matrix <- read.table("pathways_matrix_output.tsv", header = TRUE, sep = "\t")

# Check the structure of the data to confirm TRUE/FALSE values
str(pathways_matrix)

# Ensure that logical comparison works by converting to logical type
pathways_matrix[-1] <- lapply(pathways_matrix[-1], as.logical)

# Select rows where all sample columns are TRUE
true_only_rows <- pathways_matrix %>%
  filter(rowSums(select(., -ID)) == ncol(select(., -ID)))

# Extract the pathways (IDs) that are TRUE in all samples
pathways_true_only <- true_only_rows$ID

# Create a data frame to include the title
output_df <- data.frame(ID = pathways_true_only)

# Print the resulting pathways
print(output_df)

# Save the result to a TSV file with the title "ID"
```

```
write.table(output_df, "true_only_pathways.tsv", row.names = FALSE, col.names = TRUE, sep =
"\t", quote = FALSE)
```

And

```
# Load necessary library
library(dplyr)

# Read the pathways matrix from the TSV file
pathways_matrix <- read.table("pathways_matrix_output.tsv", header = TRUE, sep = "\t")

# Check the structure of the data to confirm TRUE/FALSE values
str(pathways_matrix)

# Ensure that logical comparison works by converting to logical type
pathways_matrix[-1] <- lapply(pathways_matrix[-1], as.logical)

# Select rows where all sample columns are FALSE
false_only_rows <- pathways_matrix %>%
  filter(rowSums(select(., -ID) == FALSE) == ncol(select(., -ID)))

# Extract the pathways (IDs) that are FALSE in all samples
pathways_false_only <- false_only_rows$ID

# Create a data frame to include the title
output_false_df <- data.frame(ID = pathways_false_only)

# Print the resulting pathways
print(output_false_df)

# Save the result to a TSV file with the title "ID"
write.table(output_false_df, "false_only_pathways.tsv", row.names = FALSE, col.names = TRUE,
sep = "\t", quote = FALSE)
```

- An R script was used to filter out these data from the main true/false matrix 'pathways_matrix_output.tsv', leaving a mix of true and false pathways which represented the accessory pathways:

```
# Load necessary library
library(dplyr)

# Read the original pathways matrix
pathways_matrix <- read.table("pathways_matrix_output.tsv", header = TRUE, sep = "\t")

# Read the true and false pathways files
true_pathways <- read.table("true_only_pathways.tsv", header = TRUE, sep = "\t")
false_pathways <- read.table("false_only_pathways.tsv", header = TRUE, sep = "\t")

# Extract IDs to remove
ids_to_remove <- unique(c(true_pathways$ID, false_pathways$ID))

# Create a new matrix excluding the rows with IDs in true and false pathways
filtered_matrix <- pathways_matrix %>%
  filter(!ID %in% ids_to_remove)

# Print the resulting filtered matrix
print(filtered_matrix)

# Optionally, save the filtered matrix to a new TSV file
write.table(filtered_matrix, "filtered-accessory_pathways_matrix.tsv", row.names = FALSE,
col.names = TRUE, sep = "\t", quote = FALSE)
```

- A heatmap was made to visualise the core and accessory pathways:
  - To create a matrix that included both core and accessory pathways, pathways that were false in all samples which was written to a new matrix file 'filtered_matrix_core_and_acc.tsv' using an R script:

```
# Load necessary library
library(dplyr)

# Read the original pathways matrix
pathways_matrix <- read.table("pathways_matrix_output.tsv", header = TRUE, sep = "\t")

# Read the false pathways file
false_pathways <- read.table("false_only_pathways.tsv", header = TRUE, sep = "\t")

# Extract IDs to remove
ids_to_remove <- false_pathways$ID

# Create a new matrix excluding the rows with IDs in false pathways
filtered_matrix_true_only <- pathways_matrix %>%
  filter(!ID %in% ids_to_remove)

# Print the resulting filtered matrix
print(filtered_matrix_true_only)

# Optionally, save the filtered matrix to a new TSV file
write.table(filtered_matrix_true_only, "filtered_matrix_core_and_acc.tsv", row.names = FALSE,
col.names = TRUE, sep = "\t", quote = FALSE)
```

- In order for the data to become readable in R, the new matrix 'filtered_matrix_core_and_acc.tsv' was converted into numeric format 'pathways_numeric_matrix.tsv' using an R script:

```
# Load necessary libraries
library(data.table)

# Read the pathways matrix (input file)
input_file <- "filtered_matrix_core_and_acc.tsv"  # Replace with your input file name
output_file <- "pathways_numeric_matrix.tsv"  # Output file name

# Load the data
pathways_matrix <- fread(input_file)

# Convert TRUE/FALSE to 1/0
numeric_matrix <- pathways_matrix
numeric_matrix[, (2:ncol(numeric_matrix)) := lapply(.SD, function(x) as.integer(x == TRUE)),
.SDcols = 2:ncol(numeric_matrix)]

# Rename the first column to Metabolic_pathway_ID if needed
setnames(numeric_matrix, "ID", "Metabolic_pathway_ID") # Replace 'pathway_name' with your
actual column name if different

# Save the numeric matrix to a file
fwrite(numeric_matrix, output_file, sep = "\t", quote = FALSE)

cat("Conversion complete. File saved as", output_file, "\n")
```

- An R script was used to visualise the new numeric matrix as a heatmap:

```
# Load necessary libraries
library(ggplot2)
library(reshape2)
library(plotly)
library(htmlwidgets)

# Read the TSV file into a data frame
file_path <- "pathways_numeric_matrix.tsv"
data <- read.csv(file_path, sep = "\t", header = TRUE)  # Ensure to use header = TRUE

# Assign a title to the first column for pathways
colnames(data)[1] <- "Metabolic_pathway_ID"

# Clean column names
colnames(data) <- gsub("\\(", "_", colnames(data)) # Replace '(' with '_'
colnames(data) <- gsub("\\)", "", colnames(data)) # Remove ')'
colnames(data) <- gsub(" ", "_", colnames(data)) # Replace spaces with '_'
```

```
# Reshape the data from wide to long format
data_long <- melt(data, id.vars = "Metabolic_pathway_ID", variable.name = "Sample", value.name
= "Presence_Absence")

# Convert Presence_Absence to numeric (1 and 0)
data_long$Presence_Absence <- as.numeric(as.character(data_long$Presence_Absence))

# Define plot size based on the number of categories
num_samples <- length(unique(data_long$Sample))  # Number of samples
num_pathways <- length(unique(data_long$Metabolic_pathway_ID))  # Number of metabolic pathways

# Adjust tile size to make squares smaller
tile_size <- 0.3  # Smaller tile size for even smaller squares

# Calculate plot dimensions
width <- min(num_samples * tile_size * 2, 30)  # Adjust width based on number of samples
height <- min(num_pathways * tile_size * 2, 50)  # Adjust height based on number of pathways

# Create a heatmap using ggplot2
heatmap_plot <- ggplot(data_long, aes(x = Sample, y = Metabolic_pathway_ID, fill =
Presence_Absence)) +
  geom_tile() +
  scale_fill_gradient(low = "yellow", high = "seagreen") +
  labs(x = "Sample",
       y = "Metabolic Pathway",
       fill = "Presence/Absence") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 20),  # Adjust x-axis text
size
        axis.text.y = element_text(size = 16),  # Adjust y-axis text size
        axis.title.x = element_text(size = 22),  # Adjust x-axis title size
        axis.title.y = element_text(size = 22),  # Adjust y-axis title size
        plot.title = element_blank(),  # Remove the title
        legend.text = element_text(size = 16),  # Adjust legend text size
        legend.title = element_text(size = 20),  # Adjust legend title size
        legend.key.size = unit(1.5, "cm"),  # Increase size of legend color key
        plot.margin = margin(20, 25, 20, 25)) +  # Adjust margins for readability
  coord_fixed(ratio = 1)  # Keep aspect ratio as 1 for uniform tiles

# Save the plot as a PDF with adjusted dimensions
ggsave("heatmap_plot_adjusted.pdf", plot = heatmap_plot, width = width, height = height, dpi =
300, limitsize = FALSE)

# Create interactive heatmap with plotly
interactive_heatmap <- plot_ly(data_long, x = ~Sample, y = ~Metabolic_pathway_ID, z =
~Presence_Absence, type = "heatmap", colors = c("white", "blue"))

# Save interactive heatmap to HTML
saveWidget(interactive_heatmap, "interactive_heatmap.html")
```

- The above R script was then adapted to display hierarchical clustering of samples based on the presence and absence of metabolic pathways.

```
# Load required libraries
library(readr)
library(dplyr)
library(pheatmap)

# Step 1: Read data and clean column names
original_file <- "pathways_numeric_matrix.tsv"
data <- read.delim(original_file, header = TRUE, sep = "\t")

# Rename first column
colnames(data)[1] <- "Metabolic_pathway_ID"

# Clean column names
colnames(data) <- gsub("\\(", "_", colnames(data))
colnames(data) <- gsub("\\)", "", colnames(data))
colnames(data) <- gsub(" ", "_", colnames(data))

# Step 2: Rename specific sample columns
colnames(data) <- case_when(
  colnames(data) == "industrialMoralis" ~ "SDA_bin1",
```

```
  colnames(data) == "post.IndustrialMoralis" ~ "SDA_bin2",
  colnames(data) == "modernMoralis" ~ "SDA_bin3",
  TRUE ~ colnames(data)
)

# Step 3: Convert to matrix
rownames(data) <- data$Metabolic_pathway_ID
data_matrix <- data[, -1]
data_matrix[] <- lapply(data_matrix, function(x) as.numeric(as.character(x)))
data_matrix <- as.matrix(data_matrix)

# Step 4: Create column annotation based on metadata
sample_names <- colnames(data_matrix)

# Define period classification
period_annotation <- data.frame(
  Period = case_when(
    grepl("^ERR", sample_names) ~ "Industrial/Post-industrial",
    sample_names == "ESA007_002" ~ "Industrial/Post-industrial",
    sample_names == "Calc_2102_bin3" ~ "Pre-industrial",
    grepl("^BP", sample_names) ~ "Modern",
    sample_names == "SDA_bin1" ~ "Industrial",
    sample_names == "SDA_bin2" ~ "Post-industrial",
    sample_names == "SDA_bin3" ~ "Modern",
    grepl("^GCF|^GCA", sample_names) ~ "Modern (Reference)",
    TRUE ~ "Unclassified"
  )
)
rownames(period_annotation) <- sample_names

# Step 5: Assign rainbow colors to each period group
unique_periods <- unique(period_annotation$Period)
rainbow_colors_vector <- rainbow(length(unique_periods))
names(rainbow_colors_vector) <- unique_periods
annotation_colors <- list(Period = rainbow_colors_vector)

# Step 6: Plot annotated heatmap with rainbow period colors
pheatmap(
  mat = data_matrix,
  color = colorRampPalette(c("yellow", "seagreen"))(100),
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  fontsize_row = 6,
  fontsize_col = 10,
  show_rownames = FALSE,
  border_color = NA,
  annotation_col = period_annotation,
  annotation_colors = annotation_colors,
  filename = "clustered_heatmap_with_dendrogram_rainbow_periods.pdf",  # <- NEW filename
  width = 12, height = 10
)
```