# Threshold_experiment-FS

- 6 AMR++ pipeline outputs were used (following Conor's deamination simulations):

deamOral_25ss.csv ('ancient')
deamOral_50ss.csv ('ancient')
deamOral_75ss.csv ('ancient')
modernOral_25.csv (modern)
modernOral_50.csv (modern)
modernOral_75.csv (modern)

- R script was used to filter out genes under 100 reads.

```
# Load necessary libraries
library(tidyverse)

# Read the CSV file
data <- read.csv("file_name.csv")

# Convert data to long format and filter
filtered_data <- data %>%
  pivot_longer(cols = starts_with("SRR"),
               names_to = "Sample",
               values_to = "Read_Count") %>%
  filter(grepl("^MEG", gene_accession), Read_Count >= 100)

# Save filtered data to CSV
write.csv(filtered_data, "file_nameFiltered.csv", row.names = FALSE)

# Print message
cat("Filtered data saved to filtered_data.csv\n")
```

- New filtered outputs:

deamOral_25Filtered.csv ('ancient')
deamOral_50Filtered.csv ('ancient')
deamOral_75Filtered.csv ('ancient')
modernOral_25Filtered.csv (modern)
modernOral_50Filtered.csv (modern)
modernOral_75Filtered.csv (modern)

- An R script was used to:
  - Perform ANOVA and Tukey test to show statistical significance in MEG number counts between each threshold group (**`F value = 0.917, p = 0.469`**).
  - Identify MEG numbers unique to each group (there were none).
  - Create a table showing how many times each MEG number (gene type) was counted in each threshold group.

```
# Load required libraries
library(dplyr)
library(tidyr)
library(multcomp)

# Read the CSV files
df25 <- read.csv("deamOral_25Filtered.csv")
df50 <- read.csv("deamOral_50Filtered.csv")
df75 <- read.csv("deamOral_75Filtered.csv")
df25a <- read.csv("modernOral_25Filtered.csv")
df50a <- read.csv("modernOral_50Filtered.csv")
df75a <- read.csv("modernOral_75Filtered.csv")

# Create a list of data frames and add a 'Group' column to each
df_list <- list(df25 = df25, df50 = df50, df75 = df75, df25a = df25a, df50a = df50a, df75a =
df75a)
df_list <- lapply(names(df_list), function(name) {
  df <- df_list[[name]]
  df$Group <- name
  return(df)
})

# Combine data frames into one
combined_df <- bind_rows(df_list)

# Count the frequency of each 'gene_accession' in each group
frequency_table <- combined_df %>%
  group_by(Group, gene_accession) %>%
  summarise(Frequency = n(), .groups = 'drop')

# Perform ANOVA
anova_result <- aov(Frequency ~ Group, data = frequency_table)

# Print ANOVA Summary
anova_summary <- summary(anova_result)
print("ANOVA Summary:")
print(anova_summary)

# Extract the p-value from the ANOVA result
p_value <- anova_summary[[1]]$`Pr(>F)`[1]

# Perform Tukey HSD test regardless of ANOVA result
tukey_result <- TukeyHSD(anova_result)

# Print Tukey HSD results
print("Tukey HSD Test Results:")
print(tukey_result)

# Identify unique MEG numbers in each group
unique_meg <- combined_df %>%
  group_by(gene_accession) %>%
  summarise(Unique_in = toString(unique(Group)), .groups = 'drop') %>%
  filter(nchar(gsub("[^,]", "", Unique_in)) == 0)

print("Unique MEG numbers in each group:")
print(unique_meg)

# Write unique MEG numbers to a CSV file
write.csv(unique_meg, "unique_meg_numbers.csv", row.names = FALSE)

# Identify and output all differences in MEG numbers between groups
differences <- frequency_table %>%
  pivot_wider(names_from = Group, values_from = Frequency, values_fill = list(Frequency = 0))
%>%
  rowwise() %>%
  mutate(Difference = max(c_across(starts_with("df"))) - min(c_across(starts_with("df"))))

print("All differences in MEG numbers between files:")
print(differences)

# Write all differences to a CSV file
write.csv(differences, "all_meg_number_differences.csv", row.names = FALSE)
```

- This data frame ('all_meg_number_differences.csv') was copied and pasted into an excel table 'TPFPTNFN_TableSim' and false positives and false negatives and true positives and true negatives were calculated (using excel formula) by:
    - Quantifying the difference between the threshold groups for each MEG number, e.g. **=IF(G2=B2, G2,ABS(G2-B2))**
    - Then by using the difference, false positives and negatives and true positives and negatives were determined in each group, e.g. **=IF(AND(G2=0,B2=0),"TN", IF(G2=B2, "TP", IF(G2<B2, "FP",IF(G2>B2, "FN"))))**

- False positives, false negatives, true positives, and true negatives were counted using the **Cmd + F** function. These values were manually entered into an Excel table, and bar charts were manually created using the data in these tables (Figure 1). True positives and true negatives were represented as percentages, while false positives and false negatives were displayed as counts.

- Drug-resistance groups were counted across threshold groups:
    - On excel 'ResGroup', an empty table was manually created listing drug-resistance groups down one column and the threshold groups across the top.
    - An R script was used to count the number of times a drug-resistant group was identified in each threshold group using a key word (e.g. group name 'Spiropyrimidinetriones'):

```
# Load necessary package
library(dplyr)

# Read the CSV file
data <- read.csv("all_meg_number_differences.csv", stringsAsFactors = FALSE)

# Specify the keyword
keyword <- "Spiropyrimidinetriones"

# Columns to be processed
columns_to_sum <- c("df25", "df25a", "df50", "df50a", "df75", "df75a")

# Filter rows that contain the keyword
filtered_data <- data %>%
  filter(apply(data, 1, function(row) any(grepl(keyword, row, ignore.case = TRUE))))

# Initialize a list to store the sums
total_sums <- list()

# Loop through each column and calculate the sum
for (column in columns_to_sum) {
  # Convert the column to numeric and sum the values
  total_sum <- sum(as.numeric(filtered_data[[column]]), na.rm = TRUE)
  # Store the result in the list
  total_sums[[column]] <- total_sum
}

# Print the total sums for each column
for (column in columns_to_sum) {
  cat("The total sum of values in", column, "associated with 'drug in question' is:",
total_sums[[column]], "\n")
}
```

- The output would look something like this:

```
The total sum of values in df25 associated with 'drug in question' is: 154
The total sum of values in df25a associated with 'drug in question' is: 154
The total sum of values in df50 associated with 'drug in question' is: 154
The total sum of values in df50a associated with 'drug in question' is: 154
The total sum of values in df75 associated with 'drug in question' is: 152
The total sum of values in df75a associated with 'drug in question' is: 152
```

And these numbers were manually entered (copied and pasted) in the empty table on excel.

- The drug-resistant group type table was converted into a csv file and an ANOVA test was performed with an R script (**`F value = 0.337, p = 0.89`**):

```
# Load necessary libraries
library(tidyr)
library(dplyr)

# Step 1: Read the CSV file
data <- read.csv("/path/to/ResGroup.csv")

# Print column names to verify
print(names(data))

# Step 2: Transform data into a long format
data_long <- data %>%
  pivot_longer(
    cols = c("Ancient_.25..", "Modern_.25..", "Ancient_.50..", "Modern_.50..",
"Ancient_.75..", "Modern_.75.."),
    names_to = "Threshold_Group",
    values_to = "Resistance_Count"
  )

# Step 3: Perform ANOVA
anova_result <- aov(Resistance_Count ~ Threshold_Group, data = data_long)

# Step 4: Save the ANOVA results to a text file
anova_summary <- summary(anova_result)
capture.output(anova_summary, file = "/path/to/anova_result.txt")

# Optional: print the results to console as well
print(anova_summary)
```

- An R script was used to create a heatmap to visualise the drug-resistant group type table (Figure 2).

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

# Read the CSV file into a data frame
file_path <- "ResGroup.csv"
data <- read.csv(file_path)

# 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 = "Resistance_group", variable.name = "Threshold_Group",
value.name = "Value")

# Define plot size based on the number of categories
```

```
num_x_categories <- length(unique(data_long$Threshold_Group))  # Number of threshold groups
num_y_categories <- length(unique(data_long$Resistance_group))  # Number of resistance groups

# Keep the tile size the same
tile_size <- 1  # Tile size remains the same

# Calculate plot dimensions
width <- min(num_x_categories * tile_size * 2, 30)  # Adjust width based on number of
categories
height <- min(num_y_categories * tile_size * 2, 50)  # Adjust height based on number of
categories

# Create a heatmap using ggplot2
heatmap_plot <- ggplot(data_long, aes(x = Threshold_Group, y = Resistance_group, fill =
Value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(x = "Threshold Group",
       y = "Resistance Group",
       fill = "Value") +
  scale_x_discrete(labels = c("Ancient (25%)", "Modern (25%)", "Ancient (50%)",
                              "Modern (50%)", "Ancient (75%)", "Modern (75%)")) +  # Custom x-
axis labels
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 28),  # Increased x-axis text
size
        axis.text.y = element_text(size = 28),  # Increased y-axis text size
        axis.title.x = element_text(size = 32, margin = margin(t = 20)),  # Increased x-axis
title size
        axis.title.y = element_text(size = 32),  # Increased y-axis title size
        plot.title = element_blank(),  # Remove the title
        legend.text = element_text(size = 16),  # Keep legend text size
        legend.title = element_text(size = 18),  # Keep 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 = ~Threshold_Group, y = ~Resistance_group, z =
~Value, type = "heatmap", colors = c("white", "blue"))

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