BAPs_analysis-FS

- Core and accessory BAPs populations were examined:
    - Short individual lists of populations were manually created (refer to phylogenetic tree).
    - An R script (paths must be re-defined when choosing population and output) was then used to filter populations from ‘gene_presence_absence.Rtab' (renamed 'pangenome.csv') to create individual presence/absence matrices:

```
# Define file paths
pangenome_file <- "/Users/fjstande/Documents/Pangenome/Fastbaps/pangenome.csv"
population_file <- "/Users/fjstande/Documents/Pangenome/Fastbaps/population6.txt"
output_file <- "/Users/fjstande/Documents/Pangenome/Fastbaps/population6_filtered.csv"

# Read the pangenome matrix
pangenome_data <- read.csv(pangenome_file, check.names = FALSE)

# Read the population sample names and suppress incomplete line warnings
sample_names <- readLines(population_file, warn = FALSE)

# Filter the pangenome data to include only the columns for the specified samples
# Ensure to keep the 'Gene' column
filtered_data <- pangenome_data[, c("Gene", sample_names)]

# Write the filtered data to a new CSV file
write.csv(filtered_data, output_file, row.names = FALSE)

cat("Filtered CSV file has been created at:", output_file)
```

    - An R script was used to identify genes present in all samples (core) within individual population matrices and those not present in all but identifiable in some samples (accessory), and individual population core and accessory function files were created. Part of the script was also written to identify lists that only contained one sample (population 6):

```
# Define file paths
filtered_file <- "/Users/fjstande/Documents/Pangenome/Fastbaps/population6_filtered.csv"
output_core_file <-
"/Users/fjstande/Documents/Pangenome/Fastbaps/population6CoreFunctions.csv"
output_acc_file <- "/Users/fjstande/Documents/Pangenome/Fastbaps/population6AccFunctions.csv"

# Read the filtered pangenome matrix
filtered_data <- read.csv(filtered_file, check.names = FALSE)

# Check if there is only one sample
if (ncol(filtered_data) == 2) {
  # If there's only one sample, select rows where the sample value is 1 (present)
  core_genes <- filtered_data[filtered_data[, 2] == 1, ]
  acc_genes <- filtered_data[filtered_data[, 2] != 1, ]
} else {
  # For multiple samples, identify gene functions present (1) in all samples
  core_genes <- filtered_data[rowSums(filtered_data[,-1] == 1) == ncol(filtered_data) - 1, ]

  # Identify accessory genes (those not present in all samples)
  acc_genes <- filtered_data[rowSums(filtered_data[,-1] == 1) != ncol(filtered_data) - 1, ]
}

# Write the core gene functions to a new CSV file
write.csv(core_genes, output_core_file, row.names = FALSE)

# Write the accessory gene functions to a new CSV file
write.csv(acc_genes, output_acc_file, row.names = FALSE)
```

```
cat("Core gene functions saved to:", output_core_file)
cat("Accessory gene functions saved to:", output_acc_file)
```

- An R script was used to match functions in the individual population core and accessory files to COG sub-categories (represented by letters) in eggNOG annotations, and written to new files:

```
# Load necessary library
library(dplyr)

# Read the CSV files
population_df <- read.csv("population5AccFunctions.csv")
eggnog_df <- read.csv("eGGNOGAnnotations.csv")

# Ensure the column names match (assuming the first column in eGGNOGAnnotations is 'Gene')
colnames(eggnog_df)[1] <- "Gene"

# Merge the data frames and rename the new column
merged_df <- population_df %>%
  left_join(eggnog_df %>% select(Gene, COG_category), by = "Gene") %>%
  rename(COG_sub_category = COG_category)

# Save the updated data frame to a new CSV file
write.csv(merged_df, "population5AccFunctions_with_COG.csv", row.names = FALSE)
```

- An R script was used to count and calculate % of each COG sub-category in these files which were written to new files:

```
# Load necessary library
library(dplyr)

# Read the merged CSV file
merged_df <- read.csv("population5AccFunctions_with_COG.csv")

# Calculate the percentage of each COG category
cog_percentage <- merged_df %>%
  group_by(COG_sub_category) %>%
  summarise(Count = n()) %>%
  mutate(Percentage = (Count / sum(Count)) * 100)

# Print the COG percentage table to the console
print(cog_percentage)

# Save the COG percentage table to a text file
write.table(cog_percentage, "ACCcog_percentage5.txt", sep = "\t", row.names = FALSE, quote =
FALSE)

# Optionally, you can also save it to a CSV file if needed
write.csv(cog_percentage, "ACCcog_percentage5.csv", row.names = FALSE)
```

- These COG sub-category counts and % were copied and pasted into Excel.
- ggCaller mixed many sub-category COGs together, so these letters had to be manually distributed out to single categories, e.g. mixed category KLV would count as 1 K, 1 L, and 1 V.
- These new counts and % were manually converted into pie bar charts on Excel.