AMRFinder Gene-Level Interpretation Documentation

This document includes the full script and expected output description.

Script:

```
# ============================
# AMR Gene-Level Frequency Script
# ============================

# Load libraries
library(dplyr)
library(tidyr)
library(readr)

# ---- INPUT FILE ----
input_file <- "/data/internship_data/nidhi/aba/output/amrfinder_output/merged/amr_merged_cleaned.tsv"

# ---- READ AMRFinder MERGED TSV ----
amr <- read_tsv(input_file, show_col_types = FALSE)

print(colnames(amr))

# ---- Identify gene column ----
gene_col <- NULL

if ("Element symbol" %in% names(amr)) {
gene_col <- "Element symbol"
} else if ("Element.symbol" %in% names(amr)) {
gene_col <- "Element.symbol"
} else if ("Gene" %in% names(amr)) {
gene_col <- "Gene"
} else {
stop("No gene column found. Check column names.")
}

# ---- Identify isolate ID column ----
id_col <- NULL

if ("species_id" %in% names(amr)) {
id_col <- "species_id"
} else if ("Genome.ID" %in% names(amr)) {
id_col <- "Genome.ID"
} else {
stop("No isolate ID column found.")
}

# ---- Filter only AMR genes ----
# You have a "Class" column, so filter out non-AMR categories

amr_genes <- amr %>%
filter(Class != "efflux", Class != "unknown", !is.na(Class))

# ---- Count gene frequencies ----
gene_freq <- amr_genes %>%
group_by(.data[[gene_col]]) %>%
summarise(
Isolate_Count = n_distinct(.data[[id_col]]),
.groups = "drop"
)

total_isolates <- amr %>% pull(.data[[id_col]]) %>% n_distinct()

gene_freq <- gene_freq %>%
mutate(
Total_Isolates = total_isolates,
Percent = round((Isolate_Count / total_isolates) * 100, 2)
) %>%
arrange(desc(Percent))

# ---- OUTPUT ----
```

```
output_file <- "/data/internship_data/nidhi/aba/output/gene_level_AMR_frequency.csv"
write.csv(gene_freq, output_file, row.names = FALSE)

print(head(gene_freq, 20))

cat("\n\nSaved gene-level AMR frequency table to:\n", output_file, "\n")
```

Expected Output File:

```
gene_level_AMR_frequency.csv
```