#!/usr/bin/env Rscript

# plot\_go\_template.R

# Template for plotting GO-term enrichment (log2 Combined.Score)

# - Does NOT contain any real GO terms, scores, or absolute file paths.

# - Usage:

# Rscript plot\_go\_template.R [input\_csv] [output\_image]

# If no input CSV is provided, a synthetic example dataset will be used.

#

# Expected input CSV columns: Term, Combined.Score

# Example CSV header:

# Term,Combined.Score

# "Term A",1234.5

# -------------------------

# Basic setup / packages

# -------------------------

required\_pkgs <- c("ggplot2")

for (pkg in required\_pkgs) {

if (!suppressWarnings(require(pkg, character.only = TRUE))) {

message(sprintf("Package '%s' not found. Attempting to install from CRAN...", pkg))

install.packages(pkg, repos = "https://cloud.r-project.org")

library(pkg, character.only = TRUE)

}

}

# -------------------------

# Parse command line args

# -------------------------

args <- commandArgs(trailingOnly = TRUE)

input\_path <- if (length(args) >= 1 && nzchar(args[1])) args[1] else "data/go\_enrichment.csv"

output\_path <- if (length(args) >= 2 && nzchar(args[2])) args[2] else "plots/GO\_Terms\_Log2CombinedScore.jpeg"

# -------------------------

# Read data (or create a synthetic example)

# -------------------------

if (!file.exists(input\_path)) {

message(sprintf("Input file not found at '%s'. Using synthetic example data. To use real data, provide a CSV with columns 'Term' and 'Combined.Score'.", input\_path))

set.seed(42)

example\_terms <- paste("Term", LETTERS[1:8])

example\_scores <- round(runif(length(example\_terms), min = 50, max = 8000), 2)

data <- data.frame(Term = example\_terms, Combined.Score = example\_scores, stringsAsFactors = FALSE)

} else {

data <- read.csv(input\_path, stringsAsFactors = FALSE, check.names = FALSE)

required\_cols <- c("Term", "Combined.Score")

if (!all(required\_cols %in% names(data))) {

stop("Input CSV must contain columns: 'Term' and 'Combined.Score' (case-sensitive).")

}

# keep only required columns (in case CSV has extra columns)

data <- data[, required\_cols]

}

# -------------------------

# Data validation / cleaning

# -------------------------

# Ensure Combined.Score is numeric

data$Combined.Score <- as.numeric(data$Combined.Score)

# If any non-positive values exist, replace them with a small positive number to allow log2

if (any(!is.finite(data$Combined.Score) | is.na(data$Combined.Score))) {

stop("One or more Combined.Score values are NA / non-finite. Fix your input CSV.")

}

if (any(data$Combined.Score <= 0)) {

min\_pos <- min(data$Combined.Score[data$Combined.Score > 0], na.rm = TRUE)

if (!is.finite(min\_pos)) {

# fallback

min\_pos <- 1

}

replacement <- min\_pos \* 1e-3

warning(sprintf("Found non-positive Combined.Score values. Replacing them with %g to allow log2 transform.", replacement))

data$Combined.Score[data$Combined.Score <= 0] <- replacement

}

# -------------------------

# Transform and prepare for plotting

# -------------------------

data$log2CombinedScore <- log2(data$Combined.Score)

# Optionally: keep top N terms by Combined.Score (uncomment and change N if desired)

# top\_n <- 20

# if (nrow(data) > top\_n) {

# data <- data[order(-data$Combined.Score), ][1:top\_n, ]

# }

# -------------------------

# Plot

# -------------------------

p <- ggplot(data, aes(x = reorder(Term, log2CombinedScore), y = log2CombinedScore, fill = log2CombinedScore)) +

geom\_col(width = 0.7) +

scale\_fill\_gradient(low = "lightpink", high = "red") +

coord\_flip() +

theme\_minimal(base\_size = 14) +

labs(title = "GO Term Enrichment (example / sanitized)",

x = "Ontology Term",

y = "log2(Combined Score)",

fill = "log2 Score")

# -------------------------

# Ensure output dir exists and save

# -------------------------

out\_dir <- dirname(output\_path)

if (nzchar(out\_dir) && !dir.exists(out\_dir)) {

dir.create(out\_dir, recursive = TRUE, showWarnings = FALSE)

}

# Use ggsave which automatically chooses device from file extension

ggsave(filename = output\_path, plot = p, width = 12, height = 8, units = "in", dpi = 150)

message(sprintf("Plot saved to: %s", normalizePath(output\_path, mustWork = FALSE)))