utils.R

abdulbasir

2025-12-09

```r
# utility functions for fda oncology approval analysis
#
# contains helper functions used throughout the analysis pipeline

# loading required libraries
suppressPackageStartupMessages({
  library(dplyr)
  library(readr)
  library(ggplot2)
})

# loading csv file
load_csv = function(file_path) {
  data = read_csv(file_path, show_col_types = FALSE)
  return(data)
}

# saving csv file
save_csv = function(data, file_path) {
  write_csv(data, file_path)
  cat(paste("saving results to:", file_path, "\n"))
}


# saving figure with high resolution
save_figure = function(plot, file_path, width = 12, height = 8, dpi = 300) {
  ggsave(
    filename = file_path,
    plot = plot,
    width = width,
    height = height,
    dpi = dpi,
    bg = "white"
  )
  cat(paste("saving figure to:", file_path, "\n"))
}

# printing section header
print_section_header = function(text) {
  cat(paste("\n", paste(rep("=", 70), collapse = ""), "\n", text, "\n",
            paste(rep("=", 70), collapse = ""), "\n\n", sep = ""))
}
```

```r
# assigning regulatory era (unified function for all phases)
# standardized boundaries:  1992 | 1993-2002 | 2003-2012 | 2013+
# based on regulatory milestones: PDUFA I (1992), PDUFA III (2002), FDASIA (2012)
assign_regulatory_era = function(year) {
  era = case_when(
    year <= 1992 ~ "Pre-PDUFA",
    year >= 1993 & year <= 2002 ~ "Early-PDUFA",
    year >= 2003 & year <= 2012 ~ "Mid-PDUFA",
    year >= 2013 ~ "Post-FDASIA",
    TRUE ~ "Unknown"
  )
  return(era)
}

# formatting p-value for display
format_p_value = function(p) {
  if (is.na(p)) {
    return("NA")
  } else if (p < 0.001) {
    return("p<0.001")
  } else {
    return(sprintf("p=%.3f", p))
  }
}

# calculating eta-squared effect size
calculate_eta_squared = function(ss_effect, ss_total) {
  eta_sq = ss_effect / ss_total
  return(eta_sq)
}

# interpreting eta-squared magnitude
interpret_eta_squared = function(eta_sq) {
  interpretation = case_when(
    eta_sq < 0.01 ~ "negligible",
    eta_sq < 0.06 ~ "small",
    eta_sq < 0.14 ~ "medium",
    TRUE ~ "large"
  )
  return(interpretation)
}

# performing keyword search in text field
contains_oncology_keyword = function(text, keywords) {
  if (is.na(text)) {
    return(FALSE)
  }

  text_lower = tolower(text)
  for (keyword in keywords) {
    if (grepl(keyword, text_lower, fixed = TRUE)) {
      return(TRUE)
    }
  }
  return(FALSE)
}

# calculating cell means for 2x2 design
calculate_cell_means = function(data, response_var, factor1, factor2) {
  cell_means = data %>%
    group_by(across(all_of(c(factor1, factor2)))) %>%
    summarise(
      mean = mean(.data[[response_var]], na.rm = TRUE),
      n = n(),
      sd = sd(.data[[response_var]], na.rm = TRUE),
      .groups = "drop"
    )
  return(cell_means)
}

# calculating marginal means
calculate_marginal_means = function(data, response_var, factor) {
  marginal_means = data %>%
    group_by(across(all_of(factor))) %>%
    summarise(
      mean = mean(.data[[response_var]], na.rm = TRUE),
      n = n(),
      sd = sd(.data[[response_var]], na.rm = TRUE),
      .groups = "drop"
    )
  return(marginal_means)
}

# calculating interaction sum of squares
calculate_interaction_ss = function(data, response_var, factor1, factor2) {
  # calculating grand mean
  grand_mean = mean(data[[response_var]], na.rm = TRUE)

  # calculating cell means
  cell_means = data %>%
    group_by(across(all_of(c(factor1, factor2)))) %>%
    summarise(
      cell_mean = mean(.data[[response_var]], na.rm = TRUE),
      n = n(),
      .groups = "drop"
    )

  # calculating main effect means
  factor1_means = data %>%
    group_by(across(all_of(factor1))) %>%
    summarise(
      factor1_mean = mean(.data[[response_var]], na.rm = TRUE),
      .groups = "drop"
    )

  factor2_means = data %>%
    group_by(across(all_of(factor2))) %>%
    summarise(
      factor2_mean = mean(.data[[response_var]], na.rm = TRUE),
      .groups = "drop"
    )

  # merging means back to cell data
  cell_data = cell_means %>%
    left_join(factor1_means, by = factor1) %>%
    left_join(factor2_means, by = factor2)

  # calculating interaction ss
  ss_interaction = sum(cell_data$n *
    (cell_data$cell_mean - cell_data$factor1_mean -
     cell_data$factor2_mean + grand_mean)^2)

  return(ss_interaction)
}

# setting random seed for reproducibility
set_seed = function(seed = 123) {
  set.seed(seed)
  cat(paste("random seed set to:", seed, "\n"))
}
```