```
# Install required packages
# install.packages("shiny")
# install.packages("coin")
# install.packages("dplyr")
# install.packages("pastecs")
# install.packages("summarytools")

# Load libraries
library(summarytools)
library(shiny)
library(coin)
library(dplyr)
library(broom)
library(shinyjs)
library(DT)

# Define UI
ui <- fluidPage(
  useShinyjs(),
  titlePanel("MCTOT (Multi-Functional Cycle-To-Threshold Statistical Analysis
Tool)"),
  br(),

  sidebarLayout(
    sidebarPanel(
      radioButtons(
        "analysis",
        "Analysis:",
        c(
          "Flexible Semiparametric Regression" = 0,
          "Robust Nonparametric Two-Group Comparison (Two-Sided Test)" = 1
        ),
        inline = FALSE
      ),
      fileInput("file", "Upload a CSV (comma-separated values) file", buttonLabel =
"Upload", accept = ".csv"),
      selectInput("variable", "Select a Cq or Ct variable", choices = ""),
      selectInput("normalizer", "Select a normalizer", choices = ""),
      selectInput("group", "Select a group variable", choices = ""),
      selectInput("explanatory", "Select an explanatory variable", choices = ""),
      numericInput("Cq_cutoff", "Enter a Cq or Ct cutoff", value = 40, min = 0, max =
40),
      actionButton("go", "Calculate")
    ),
```

```
    mainPanel(
      tabsetPanel(type = "tabs",
        # ... [Other UI elements and descriptions] ...
      )
    )
  )
)

# Define server logic
server <- function(input, output, session) {
  observe({
    shinyjs::show("group")
    shinyjs::hide("explanatory")
  })

  observe({
    if (input$analysis == "1") {
      shinyjs::show("group")
      shinyjs::hide("explanatory")

      isolate(
        updateSelectInput(
          session,
          "group",
          label = "Select a group variable",
          choices = ""
        )
      )
      isolate(
        updateSelectInput(
          session,
          "explanatory",
          label = "Select an explanatory variable",
          choices = ""
        )
      )
    } else if (input$analysis == "0") {
      shinyjs::show("explanatory")
      shinyjs::hide("group")

      isolate(
        updateSelectInput(
          session,
          "group",
```

```
        label = "Select a group variable",
        choices = ""
      )
    )
    isolate(
      updateSelectInput(
        session,
        "explanatory",
        label = "Select an explanatory variable",
        choices = ""
      )
    )
  }
 })

 # Load the data
 values <- reactiveValues(df_data = NULL)

 data <- reactive({
  req(input$file)
  ext <- tools::file_ext(input$file$name)
  if (ext != "csv") {
   validate("Invalid file. Please upload a .csv file.")
  }
  read.csv(input$file$datapath, header = TRUE)
 })
}

# Update UI
observeEvent(input$file, {
 values$df_data <- data()
 updateSelectInput(
  session,
  inputId = "variable",
  choices = colnames(values$df_data)
 )
})

observeEvent(input$file, {
 updateSelectInput(
  session,
  inputId = "normalizer",
  choices = colnames(values$df_data)
 )
```

```
})

observeEvent(input$file, {
  updateSelectInput(
    session,
    inputId = "group",
    choices = colnames(values$df_data)
  )
})

observeEvent(input$file, {
  updateSelectInput(
    session,
    inputId = "explanatory",
    choices = colnames(values$df_data)
  )
})

# Create descriptive statistics of the input file
re <- eventReactive(input$go, {
  req(values$df_data)  # Ensure a file is uploaded

  if (input$analysis == "1") {
    req(input$variable, input$normalizer, input$group)
  } else if (input$analysis == "0") {
    req(input$variable, input$normalizer, input$explanatory)
  }

  summary_table <- data.frame(descr(values$df_data))

  temp <- values$df_data
  cctCol <- if_else(temp[[input$variable]] < input$Cq_cutoff,
temp[[input$variable]], input$Cq_cutoff)
  observedCol <- if_else(temp[[input$variable]] < input$Cq_cutoff, 1, 0)
  expdcctCol <- if_else(temp[[input$variable]] < input$Cq_cutoff,
exp(temp[[input$variable]] - temp[[input$normalizer]]), exp(input$Cq_cutoff -
temp[[input$normalizer]]))
  dcctCol <- if_else(temp[[input$variable]] < input$Cq_cutoff,
temp[[input$variable]] - temp[[input$normalizer]], input$Cq_cutoff -
temp[[input$normalizer]])

  temp$cct <- cctCol
  temp$observed <- observedCol
  temp$expdcct <- expdcctCol
```

```
  temp$dcct <- dcctCol

  if (input$analysis == "1") {
    fit <- logrank_test(
      as.formula(paste("Surv(expdcct, observed) ~ as.factor(", input$group, ")")),
      data = temp,
      distribution = "exact",
      type = c("Fleming-Harrington")
    )
    model1Results <- pvalue(fit)
    list(descriptive = summary_table, model1 = model1Results)
  } else if (input$analysis == "0") {
    fit <- coxph(as.formula(paste("Surv(dcct, observed) ~ ", input$explanatory)),
data = temp)
    model2Results <- data.frame(summary(fit)$coefficients)
    colnames(model2Results) <- c("Coefficient", "Exponentiated Coefficient",
"Standard Error of the Estimated Coefficient", "Z Value", "Pr(>|Z|):p-value")

    list(descriptive = summary_table, model2 = model2Results)
  }

  output$result <- renderText({
    req(re())
    results <- re()
    if (input$analysis == "1") {
      c("\n\n\n", paste("CTOT Nonparametric Two-Sided Test p-value:",
formatC(results$model1, format = "e", digits = 3)), "\n\n\n")
    }
  })

  output$result1 <- renderTable({
    req(re())
    results <- re()
    if (input$analysis == "0") {
      results$model2
    }
  }, rownames = TRUE)

  output$result2 <- renderTable({
    req(re())
    results <- re()
    results$descriptive
  }, rownames = TRUE)
})
```

```
shinyApp(ui, server)
```