Aplicación Shiny en R para Prueba de Normalidad

Waldir Y. Velásquez Quispe

1. Introducción

En este trabajo se presenta una aplicación en R utilizando `Shiny` para cargar archivos y realizar pruebas de normalidad. El nivel de significancia se representará como α.

2. Código en R

A continuación se muestra un fragmento del código en R utilizado en la aplicación:

```
datos_cargados <- reactiveVal(NULL)

library(shiny)
library(ggplot2)
library(nortest)

ui <- fluidPage(
  titlePanel("Prueba de Normalidad"),

  sidebarLayout(
    sidebarPanel(
      h4("Fuente de Datos"),
      radioButtons("tipo_datos", "Seleccione el tipo de
         entrada:",
                   choices = c("Ingresar datos
                      manualmente" = "manual",
```

```
                        "Cargar archivo" = "
                           archivo")),

conditionalPanel(
  condition = "input.tipo_datos == 'manual'",
  textAreaInput("datos_manual",
                "Ingrese los datos (separados por
                   comas o espacios):",
                value = "23, 25, 28, 29, 30, 32,
                   35, 36, 38, 40",
                rows = 5)
),

conditionalPanel(
  condition = "input.tipo_datos == 'archivo'",
  fileInput("archivo", "Cargar archivo (CSV o Excel
     )",
            accept = c(".csv", ".xlsx", ".xls")),
  uiOutput("select_columna")
),

hr(),
h4("Pruebas de Normalidad"),
checkboxGroupInput("pruebas", "Seleccione las
   pruebas:",
                   choices = c("Shapiro-Wilk" = "
                      shapiro",
                               "Kolmogorov-Smirnov"
                                  = "ks",
                               "Anderson-Darling" =
                                  "ad",
                               "Jarque-Bera" = "jb"
                                  ),
                   selected = c("shapiro", "ks")),

sliderInput("alpha", "Nivel de significancia (  ):"
   ,
            min = 0.01, max = 0.10, value = 0.05,
               step = 0.01),
```

```
      actionButton("ejecutar", "Ejecutar Pruebas", class
         = "btn-primary")
    ),

    mainPanel(
      tabsetPanel(
        tabPanel("Resultados",
                 h4("Resumen de Datos"),
                 verbatimTextOutput("resumen"),
                 hr(),
                 h4("Resultados de las Pruebas"),
                 tableOutput("tabla_resultados"),
                 hr(),
                 h4("Interpretaci n"),
                 verbatimTextOutput("interpretacion")
        ),

        tabPanel("Gr ficos",
                 h4("Histograma y Curva Normal"),
                 plotOutput("histograma", height = "300px
                    "),
                 hr(),
                 h4("Q-Q Plot"),
                 plotOutput("qqplot", height = "300px"),
                 hr(),
                 h4("Boxplot"),
                 plotOutput("boxplot", height = "300px")
        ),

        tabPanel("Datos",
                 h4("Vista de los datos"),
                 verbatimTextOutput("vista_datos")
        )
      )
    )
  )
)

server <- function(input, output, session) {
```

```
# Datos reactivos
datos_cargados <- reactiveVal(NULL)

# Cargar archivo
observeEvent(input$archivo, {
  req(input$archivo)

  ext <- tools::file_ext(input$archivo$name)

  tryCatch({
    if (ext == "csv") {
      df <- read.csv(input$archivo$datapath,
          stringsAsFactors = FALSE)
    } else if (ext %in% c("xlsx", "xls")) {
      df <- readxl::read_excel(input$archivo$datapath)
    }
    datos_cargados(df)
  }, error = function(e) {
    showNotification(paste("Error al cargar el archivo:
        ", e$message), type = "error")
  })
})

# Selector de columna para archivo
output$select_columna <- renderUI({
  req(datos_cargados())
  df <- datos_cargados()
  columnas_numericas <- names(df)[sapply(df, is.numeric
     )]

  if (length(columnas_numericas) == 0) {
    return(helpText("No se encontraron columnas
        num ricas"))
  }

  selectInput("columna", "Seleccione la columna a
     analizar:",
              choices = columnas_numericas)
})
```

```
# Obtener datos para an lisis
obtener_datos <- reactive({
  if (input$tipo_datos == "manual") {
    texto <- gsub(",", " ", input$datos_manual)
    datos <- as.numeric(unlist(strsplit(texto, "\\s+"))
       )
    datos <- datos[!is.na(datos)]
    return(datos)
  } else {
    req(datos_cargados(), input$columna)
    df <- datos_cargados()
    datos <- df[[input$columna]]
    datos <- datos[!is.na(datos)]
    return(datos)
  }
})

# Resumen de datos
output$resumen <- renderPrint({
  input$ejecutar
  isolate({
    datos <- obtener_datos()
    if (length(datos) < 3) {
      cat("Error: Se necesitan al menos 3 observaciones
         \n")
      return()
    }
    cat("N mero de observaciones:", length(datos), "\n
       ")
    cat("Media:", round(mean(datos), 4), "\n")
    cat("Desviaci n est ndar:", round(sd(datos), 4),
       "\n")
    cat("Mediana:", round(median(datos), 4), "\n")
    cat("M nimo:", round(min(datos), 4), "\n")
    cat("M ximo:", round(max(datos), 4), "\n")
  })
})

# Realizar pruebas
resultados_pruebas <- reactive({
```

```
input$ejecutar
isolate({
  datos <- obtener_datos()

  if (length(datos) < 3) {
    return(NULL)
  }

  resultados <- list()

  if ("shapiro" %in% input$pruebas && length(datos)
     >= 3 && length(datos) <= 5000) {
    test <- shapiro.test(datos)
    resultados$Shapiro <- list(
      Prueba = "Shapiro-Wilk",
      Estad stico = test$statistic,
      P_valor = test$p.value
    )
  }

  if ("ks" %in% input$pruebas && length(datos) >= 3)
     {
    test <- suppressWarnings(ks.test(datos, "pnorm",
       mean(datos), sd(datos)))
    resultados$KS <- list(
      Prueba = "Kolmogorov-Smirnov",
      Estad stico = test$statistic,
      P_valor = test$p.value
    )
  }

  if ("ad" %in% input$pruebas && length(datos) >= 7)
     {
    test <- ad.test(datos)
    resultados$AD <- list(
      Prueba = "Anderson-Darling",
      Estad stico = test$statistic,
      P_valor = test$p.value
    )
  }
```

```
    if ("jb" %in% input$pruebas && length(datos) >= 3)
       {
      n <- length(datos)
      m3 <- mean((datos - mean(datos))^3)
      m4 <- mean((datos - mean(datos))^4)
      s <- sqrt(mean((datos - mean(datos))^2))
      skew <- m3 / s^3
      kurt <- m4 / s^4
      jb_stat <- n * (skew^2 / 6 + (kurt - 3)^2 / 24)
      p_val <- 1 - pchisq(jb_stat, 2)

      resultados$JB <- list(
        Prueba = "Jarque-Bera",
        Estad stico = jb_stat,
        P_valor = p_val
      )
    }

    return(resultados)
  })
})

# Tabla de resultados
output$tabla_resultados <- renderTable({
  res <- resultados_pruebas()
  if (is.null(res) || length(res) == 0) {
    return(data.frame(Mensaje = "No hay resultados
       disponibles"))
  }

  df <- data.frame(
    Prueba = sapply(res, function(x) x$Prueba),
    Estad stico = sapply(res, function(x) round(x$
       Estad stico, 4)),
    P_valor = sapply(res, function(x) round(x$P_valor,
       4)),
    Conclusi n = sapply(res, function(x) {
      ifelse(x$P_valor < input$alpha,
             "Rechazar H0 (No normal)",
```

```
            "No rechazar H0 (Normal)")
    })
  )
  rownames(df) <- NULL
  df
}, striped = TRUE, hover = TRUE, bordered = TRUE)

# Interpretaci n
output$interpretacion <- renderPrint({
  res <- resultados_pruebas()
  if (is.null(res) || length(res) == 0) {
    cat("Ejecute las pruebas para ver la
       interpretaci n\n")
    return()
  }

  cat("INTERPRETACI N:\n")
  cat("================\n\n")
  cat("Hip tesis:\n")
  cat("H0: Los datos provienen de una distribuci n
     normal\n")
  cat("H1: Los datos NO provienen de una distribuci n
     normal\n\n")
  cat("Nivel de significancia:    =", input$alpha, "\n\
     n")

  rechazos <- sum(sapply(res, function(x) x$P_valor <
     input$alpha))
  total <- length(res)

  if (rechazos == 0) {
    cat("    CONCLUSI N: Los datos parecen seguir una
       distribuci n normal\n")
    cat("  (Todas las pruebas no rechazan H0)\n")
  } else if (rechazos == total) {
    cat("    CONCLUSI N: Los datos NO siguen una
       distribuci n normal\n")
    cat("  (Todas las pruebas rechazan H0)\n")
  } else {
    cat("    CONCLUSI N: Resultados mixtos\n")
```

```
    cat(" ", rechazos, "de", total, "pruebas rechazan
       la normalidad\n")
    cat("  Se recomienda analizar los gr ficos y el
       contexto de los datos\n")
  }
})

# Histograma
output$histograma <- renderPlot({
  input$ejecutar
  isolate({
    datos <- obtener_datos()
    if (length(datos) < 3) return(NULL)

    df <- data.frame(x = datos)
    ggplot(df, aes(x = x)) +
      geom_histogram(aes(y = after_stat(density)), bins
          = 30,
                     fill = "steelblue", color = "black
                        ", alpha = 0.7) +
      stat_function(fun = dnorm,
                    args = list(mean = mean(datos), sd
                       = sd(datos)),
                    color = "red", linewidth = 1) +
      labs(title = "Histograma con Curva Normal",
           x = "Valores", y = "Densidad") +
      theme_minimal()
  })
})

# Q-Q Plot
output$qqplot <- renderPlot({
  input$ejecutar
  isolate({
    datos <- obtener_datos()
    if (length(datos) < 3) return(NULL)

    qqnorm(datos, main = "Q-Q Plot", col = "steelblue",
        pch = 19)
    qqline(datos, col = "red", lwd = 2)
```

```
    })
  })

  # Boxplot
  output$boxplot <- renderPlot({
    input$ejecutar
    isolate({
      datos <- obtener_datos()
      if (length(datos) < 3) return(NULL)

      df <- data.frame(x = "Datos", y = datos)
      ggplot(df, aes(x = x, y = y)) +
        geom_boxplot(fill = "steelblue", alpha = 0.7) +
        labs(title = "Boxplot", x = "", y = "Valores") +
        theme_minimal()
    })
  })

  # Vista de datos
  output$vista_datos <- renderPrint({
    datos <- obtener_datos()
    if (length(datos) == 0) {
      cat("No hay datos disponibles\n")
      return()
    }
    cat("Datos cargados (", length(datos), "observaciones
       ):\n\n")
    print(datos)
  })
}

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

3. Conclusión

La implementación de esta aplicación facilita la carga de datos y la verificación de supuestos estadísticos como la normalidad.