Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor: Fonseca Lizarraga Cinthia Yaneth

Trabajo Encargado - N° 11

App Shiny – PSM Simplificado

Introducción

Este trabajo presenta una aplicación interactiva desarrollada en **R Shiny** que permite realizar un análisis de **Propensity Score Matching (PSM)** de forma simplificada, sin utilizar librerías especializadas como `MatchIt`. El objetivo es estimar el efecto causal de la **vacunación** sobre la **hospitalización** en una población simulada, controlando por covariables de confusión mediante emparejamiento manual por puntaje de propensión.

Métodos

- **Generación de datos sintéticos** con covariables clínicas y sociodemográficas.
- **Cálculo del propensity score** mediante regresión logística.
- **Emparejamiento 1:1** sin reemplazo, con caliper ajustable.
- **Evaluación del balance** mediante el **Standardized Mean Difference (SMD)**.
- **Estimación del efecto causal** como diferencia de tasas de hospitalización.

Variables

- **Tratamiento**: Vacunado (1) vs No vacunado (0)
- **Resultado**: Hospitalización (1) vs No hospitalización (0)
- **Covariables**: Edad, sexo, diabetes, hipertensión, inmunocompromiso, escolaridad, seguro médico

```

#  ================================================================================
# APP SHINY     PSM SIMPLIFICADO (Sin MatchIt)
#  ================================================================================
```

```r

library(shiny)
library(shinythemes)
library(ggplot2)
library(dplyr)
library(plotly)
library(DT)

# 1. Generar datos
generar_datos_vacuna <- function(n = 1500) {
  set.seed(123)

  edad <- pmax(18, pmin(90, rnorm(n, 45, 15)))
  sexo <- rbinom(n, 1, 0.52)
  diabetes <- rbinom(n, 1, 0.15)
  hipertension <- rbinom(n, 1, 0.25)
  inmunocompromiso <- rbinom(n, 1, 0.05)
  escolaridad <- pmax(6, pmin(20, rnorm(n, 12, 3)))
  seguro <- rbinom(n, 1, 0.7)

  ps_logit <- -2 + 0.02*edad + 0.3*sexo + 0.4*diabetes +
    0.3*hipertension - 0.1*inmunocompromiso +
    0.05*escolaridad + 0.5*seguro

  ps <- plogis(ps_logit)
  vacunado <- rbinom(n, 1, prob = ps)

  resultado_logit <- -3 + 0.04*edad + 0.3*diabetes +
    0.25*hipertension + 0.8*inmunocompromiso -
    0.6*vacunado

  hospitalizacion <- rbinom(n, 1, prob = plogis(resultado_logit))

  data.frame(
    id = 1:n,
    edad = edad,
    sexo = sexo,
    diabetes = diabetes,
    hipertension = hipertension,
    inmunocompromiso = inmunocompromiso,
    escolaridad = escolaridad,
    seguro = seguro,
    ps = ps,
    vacunado = vacunado,
    hospitalizacion = hospitalizacion
  )
}

# 2. Matching manual por PS
realizar_psm_manual <- function(datos, caliper = 0.25) {
  vacunados <- datos[datos$vacunado == 1, ]
  controles <- datos[datos$vacunado == 0, ]

  matched_data <- data.frame()
```

```r

  for (i in seq_len(nrow(vacunados))) {
    v_ps <- vacunados$ps[i]
    diffs <- abs(controles$ps - v_ps)

    if (min(diffs) <= caliper) {
      match_idx <- which.min(diffs)
      matched_data <- rbind(
        matched_data,
        vacunados[i, ],
        controles[match_idx, ]
      )
      controles <- controles[-match_idx, ]
    }
  }

  matched_data
}

# 3. Balance
calcular_balance <- function(antes, despues) {
  vars <- c("edad", "sexo", "diabetes", "hipertension", "inmunocompromiso")
  res <- data.frame()

  for(v in vars) {
    mv_a <- mean(antes[[v]][antes$vacunado == 1], na.rm = TRUE)
    mc_a <- mean(antes[[v]][antes$vacunado == 0], na.rm = TRUE)
    mv_d <- mean(despues[[v]][despues$vacunado == 1], na.rm = TRUE)
    mc_d <- mean(despues[[v]][despues$vacunado == 0], na.rm = TRUE)
    sd_a <- sd(antes[[v]])

    res <- rbind(res, data.frame(
      Variable = v,
      SMD_Antes = abs(mv_a - mc_a) / sd_a,
      SMD_Despues = abs(mv_d - mc_d) / sd_a
    ))
  }
  res
}

# 4. Resultados
analizar_resultados <- function(dm) {
  tv <- mean(dm$hospitalizacion[dm$vacunado == 1]) * 100
  tc <- mean(dm$hospitalizacion[dm$vacunado == 0]) * 100
  list(tasa_vacunado = tv, tasa_control = tc, efectividad = (tc - tv) / tc * 100)
}

# 5. UI
ui <- fluidPage(
  theme = shinytheme("darkly"),
  tags$head(tags$style(HTML("
    body { font-family: 'Arial', sans-serif; }
```

```
    .header-box {
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      color: white;
      padding: 30px;
      border-radius: 10px;
      margin-bottom: 20px;
    }
    .metric-box {
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      color: white;
      padding: 20px;
      border-radius: 8px;
      text-align: center;
    }
    .metric-value { font-size: 28px; font-weight: bold; margin-top: 10px; }
    .metric-label { font-size: 11px; text-transform: uppercase; opacity: 0.9; }
    .section-title {
      font-size: 18px;
      font-weight: bold;
      margin: 25px 0 15px 0;
      color: #667eea;
      border-bottom: 2px solid #667eea;
      padding-bottom: 10px;
    }
  "))),

  div(class = "header-box",
      h1("PSM: Analisis Causal"),
      p("Vacunacion vs Hospitalizacion")),

  fluidRow(
    column(4,
           h4("Parametros"),
           sliderInput("n", "Tama o muestral:", 500, 3000, 1500, 100),
           sliderInput("caliper", "Caliper:", 0.05, 0.5, 0.25, 0.05)),
    column(4,
           br(),
           actionButton("run", "Ejecutar", class = "btn btn-primary btn-lg", width = "100%"),
           br(), br(),
           actionButton("clr", "Limpiar", class = "btn btn-secondary", width = "100%")),
    column(4,
           h4("Estado"),
           textOutput("status"),
           textOutput("details"))
  ),

  conditionalPanel(
    condition = "output.ready",
    h2(class = "section-title", "Indicadores"),
    fluidRow(
```

```
      column(3, div(class = "metric-box",
                    div(class = "metric-label", "Efectividad"),
                    div(class = "metric-value", textOutput("ef")))),
      column(3, div(class = "metric-box",
                    div(class = "metric-label", "Parejas"),
                    div(class = "metric-value", textOutput("pairs")))),
      column(3, div(class = "metric-box",
                    div(class = "metric-label", "Exclusion"),
                    div(class = "metric-value", textOutput("exc")))),
      column(3, div(class = "metric-box",
                    div(class = "metric-label", "Reduccion"),
                    div(class = "metric-value", textOutput("red"))))
    ),

    h2(class = "section-title", "Graficos"),
    fluidRow(column(6, plotlyOutput("ps", height = "350px")),
             column(6, plotlyOutput("bal", height = "350px"))),
    fluidRow(column(6, plotlyOutput("hosp", height = "350px")),
             column(6, plotlyOutput("bimp", height = "350px"))),

    h2(class = "section-title", "Tablas"),
    fluidRow(column(6, h4("Antes"), DTOutput("t1")),
             column(6, h4("Despues"), DTOutput("t2"))),

    h2(class = "section-title", "Balance"),
    DTOutput("tbal"),

    div(style = "background: #2b2b2b; padding: 20px; border-radius: 8px;
   margin-top: 20px;",
        h3("Conclusion", style = "color: #667eea;"),
        textOutput("conc"))
  )
)

# 6. Server
server <- function(input, output, session) {

  val <- reactiveValues(d = NULL, m = NULL, bal = NULL, res = NULL, ready
   = FALSE)

  observeEvent(input$run, {
    output$status <- renderText("Generando datos...")
    tryCatch({
      d0 <- generar_datos_vacuna(input$n)
      dm <- realizar_psm_manual(d0, caliper = input$caliper)

      if (nrow(dm) == 0) {
        output$status <- renderText("Sin matches. Aumenta caliper.")
        return()
      }

      val$d <- d0
      val$m <- dm
      val$bal <- calcular_balance(d0, val$m)
```

```
      val$res <- analizar_resultados(val$m)
      val$ready <- TRUE
      output$status <- renderText("Listo")
      output$details <- renderText(paste("Muestra:", nrow(val$d), "| Parejas:", nrow(val$m) / 2))
    }, error = function(e) {
      output$status <- renderText(paste("Error:", e$message))
      val$ready <- FALSE
    })
  })

  observeEvent(input$clr, {
    val$ready <- FALSE
    output$status <- renderText("Listo para nuevo analisis")
    output$details <- renderText("")
  })

  output$ready <- reactive(val$ready)
  outputOptions(output, "ready", suspendWhenHidden = FALSE)

  output$ef <- renderText(if(val$ready) paste0(round(val$res$efectividad, 1), "%"))
  output$pairs <- renderText(if(val$ready) as.integer(nrow(val$m) / 2))
  output$exc <- renderText(if(val$ready) paste0(round((1 - nrow(val$m) / nrow(val$d)) * 100, 1), "%"))
  output$red <- renderText(if(val$ready) {
    paste0(round(abs(mean(val$d$hospitalizacion[val$d$vacunado == 0]) -
                     mean(val$m$hospitalizacion[val$m$vacunado == 0])) * 100, 1), "%")
  })

  output$ps <- renderPlotly({
    if (!val$ready) return(NULL)
    g <- ggplot(val$m, aes(x = ps, fill = factor(vacunado, labels = c("Control", "Vacunado")))) +
      geom_histogram(bins = 40, alpha = 0.7, position = "identity") +
      scale_fill_manual(values = c("#e74c3c", "#27ae60")) +
      labs(title = "PS despues matching", x = "Propensity Score", y = "Frecuencia", fill = "") +
      theme_minimal() + theme(legend.position = "top")
    ggplotly(g, tooltip = "none")
  })

  output$bal <- renderPlotly({
    if (!val$ready) return(NULL)
    b <- data.frame(
      Variable = rep(val$bal$Variable, 2),
      SMD = c(val$bal$SMD_Antes, val$bal$SMD_Despues),
      Per = rep(c("Antes", "Despues"), each = nrow(val$bal))
    )
    g <- ggplot(b, aes(x = reorder(Variable, -SMD), y = SMD, fill = Per)) +
      geom_col(position = "dodge", alpha = 0.8) +
      geom_hline(yintercept = 0.1, linetype = "dashed", color = "red") +
```

```
260       scale_fill_manual(values = c("#e74c3c", "#27ae60")) +
261       labs(title = "SMD", y = "SMD", x = "", fill = "") +
262       theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust
      = 1))
263     ggplotly(g, tooltip = "none")
264   })
265
266   output$hosp <- renderPlotly({
267     if (!val$ready) return(NULL)
268     s <- data.frame(
269       Grupo = c("Control", "Vacunado"),
270       Tasa = c(mean(val$m$hospitalizacion[val$m$vacunado == 0]) * 100,
271                mean(val$m$hospitalizacion[val$m$vacunado == 1]) * 100)
272     )
273     g <- ggplot(s, aes(x = Grupo, y = Tasa, fill = Grupo)) +
274       geom_col(alpha = 0.8) +
275       geom_text(aes(label = sprintf("%.1f%%", Tasa)), vjust = -0.3,
     fontface = "bold") +
276       scale_fill_manual(values = c("#e74c3c", "#27ae60")) +
277       labs(title = "Tasa hospitalizacion", y = "Tasa (%)", x = "") +
278       theme_minimal() + theme(legend.position = "none")
279     ggplotly(g, tooltip = "none")
280   })
281
282   output$bimp <- renderPlotly({
283     if (!val$ready) return(NULL)
284     b <- val$bal
285     b$Mejora <- ((b$SMD_Antes - b$SMD_Despues) / b$SMD_Antes) * 100
286     g <- ggplot(b, aes(x = reorder(Variable, -Mejora), y = Mejora, fill =
     Mejora)) +
287       geom_col(alpha = 0.8) +
288       scale_fill_gradient(low = "#e74c3c", high = "#27ae60") +
289       labs(title = "Mejora en balance (%)", y = "% mejora", x = "") +
290       theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust
      = 1))
291     ggplotly(g, tooltip = "none")
292   })
293
294   makeTable <- function(dat) {
295     data.frame(
296       Grupo = c("Control", "Vacunado"),
297       N = c(sum(dat$vacunado == 0), sum(dat$vacunado == 1)),
298       EdadMedia = c(round(mean(dat$edad[dat$vacunado == 0], na.rm = TRUE),
      1),
299                     round(mean(dat$edad[dat$vacunado == 1], na.rm = TRUE),
      1)),
300       Diabetes = c(round(mean(dat$diabetes[dat$vacunado == 0], na.rm =
     TRUE) * 100, 1),
301                    round(mean(dat$diabetes[dat$vacunado == 1], na.rm =
     TRUE) * 100, 1)),
302       Hipertension = c(round(mean(dat$hipertension[dat$vacunado == 0], na.
     rm = TRUE) * 100, 1),
303                        round(mean(dat$hipertension[dat$vacunado == 1], na.
     rm = TRUE) * 100, 1)),
```

```r
      Hospitalizacion = c(round(mean(dat$hospitalizacion[dat$vacunado == 0], na.rm = TRUE) * 100, 1),
                          round(mean(dat$hospitalizacion[dat$vacunado == 1], na.rm = TRUE) * 100, 1))
    )
  }

  output$t1 <- renderDT(if(val$ready) makeTable(val$d), options = list(dom = 't'))
  output$t2 <- renderDT(if(val$ready) makeTable(val$m), options = list(dom = 't'))

  output$tbal <- renderDT({
    if (!val$ready) return(NULL)
    val$bal$SMD_Antes <- round(val$bal$SMD_Antes, 3)
    val$bal$SMD_Despues <- round(val$bal$SMD_Despues, 3)
    val$bal$Mejora <- round(((val$bal$SMD_Antes - val$bal$SMD_Despues) / val$bal$SMD_Antes) * 100, 1)
    val$bal
  }, options = list(dom = 't'))

  output$conc <- renderText({
    if (!val$ready) return("")
    e <- round(val$res$efectividad, 1)
    p <- nrow(val$m) / 2
    paste("Vacunacion reduce hospitalizacion", e, "%.",
          "Se formaron", p, "parejas balanceadas.",
          "Balance mejorado en todas covariables.")
  })
}

shinyApp(ui, server)
```

Listing 1: Código R completo del App Shiny

Resultados

Cuadro 1: Ejemplo de salida: Balance antes y después del emparejamiento

Variable	SMD Antes	SMD Después	Mejora (%)
Edad	0.45	0.02	95.6
Sexo	0.30	0.01	96.7
Diabetes	0.35	0.03	91.4
Hipertensión	0.28	0.02	92.9
Inmunocompromiso	0.20	0.01	95.0

- **Efectividad estimada**: 58.3 %
- **Parejas formadas**: 342

- **Exclusión por caliper**: 54.4 %
- **Reducción en sesgo**: 94.2 %

Conclusión

El App Shiny desarrollado permite:

- Realizar PSM sin dependencia de librerías externas complejas
- Visualizar interactivamente el impacto del caliper sobre el matching
- Evaluar el balance de covariables antes y después del emparejamiento
- Estimar efectos causales en contextos observacionales
- Facilitar la interpretación de resultados mediante tablas y gráficos interactivos

Este enfoque es útil para docencia y análisis rápidos en estudios epidemiológicos o económicos donde se requiere controlar por sesgo de selección.

Repositorio

Código completo disponible en:
`https://github.com/Yaneth15/mi-primer-repositorio.git`