Constructing a medium complexity shiny app for power analysis

Ronald (Ryy) Glenn Thomas

10/23/23

Table of contents

Figure 1: under construction

1 Introduction

One of the most common tasks for biostatistician is the calculation of a required sample size for a two group comparison based on a two sample students t-test. While common this exercise is not trivial. There are numerous parameters to be set by the investigator. and differing approaches to address several of the parameters.

2 Methods

Syncing

The range of effect size input across multiple widgets.

Aesethetics

Colors Size Font

Graph

plot(), ggplot(), plotly(), D3

3 - Code

```
library(pacman)

vline <- function(x = 0, color = "#94c0ff") {
  list(
    type = "line",
    y0 = 0,
    y1 = 1,
    yref = "paper",
    x0 = x,
    x1 = x,
    line = list(color = color)
  )
}

hline <- function(y = 0, color = "#caff94") {
  list(
    type = "line",
    x0 = 0,
    x1 = 1,
    xref = "paper",
    y0 = y,
    y1 = y,
    line = list(color = color)
  )
}
brewer.pal(n = 8, name = "YlGn")
p_load(ggplot2, pwr, shiny, bslib, ggplot2, plotly, "RColorBrewer")
```

```
ui <- page_sidebar(
  tags$style(HTML(".js-irs-0 .irs-single, .js-irs-0 .irs-bar-edge, .js-irs-0 .irs-bar {backg
  # tags$style(HTML(".js-irs-1 .irs-single, .js-irs-1 .irs-bar-edge, .js-irs-1 .irs-bar {bac
  # tags$style(HTML(".js-irs-2 .irs-single, .js-irs-2 .irs-bar-edge, .js-irs-2 .irs-bar {bac
  title = "Power Calculator for Two Group Parallel Designs",
sidebar = sidebar(
sliderInput("N", "Total Sample Size:",
        min = 0, max = 100, value = 50,
        step = 1
      ),
      sliderInput("dropout", "Dropout Rate:", min = 0, max = .40, value = .10),
      selectInput(
        "dmeth", "Method for Delta (Effect Size) Specification",
        c(
          "Standard Deviation Units (Cohen)" = "std",
          "Difference in change scores between groups" = "diff",
          "Percent reduction/increase in active
             change score from placebo" = "pct",
          "Change in active group" = "active"
        )
      ),
      conditionalPanel(
        condition = "input.dmeth == 'std'",
        sliderInput("del", " delta",
          min = 0.0, max = 1.5, value = c(.50, 1.5),
          step = .1
        )
      ),
      conditionalPanel(
        condition = "input.dmeth == 'diff' |
  input.dmeth == 'pct' | input.dmeth == 'active' ",
        sliderInput("sd0", "Placebo Standard Deviation:", 10.0,
          min = 1, max = 20,
          step = .1
        ),
        sliderInput("d0", "Placebo Change Score:", 10.0,
          min = 0, max = 20,
          step = .1
        )
```

```
      ),
      conditionalPanel(
        condition = "input.dmeth == 'diff'",
        sliderInput("dff", "diff in change scores",
          min = 0, max = 30,
          value = c(5, 10), step = .1
        )
      ),
      conditionalPanel(
        condition = "input.dmeth == 'pct'",
        sliderInput("pct", "Fraction Reduction",
          min = 0, max = 1,
          value = c(.1, .9), step = .1
        )
      ),
      conditionalPanel(
        condition = "input.dmeth == 'active'",
        sliderInput("active", "Treatment group change",
          min = 1, max = 50, value = c(1, 50)
        )
      ),
      checkboxInput("choice", "Additional parameter settings"),
      conditionalPanel(
        condition = "input.choice == 1",
        numericInput("type1", "Type one error:",
          min = 0.001, max = .2,
          value = .05, step = .005
        ),
        numericInput("ratio", "Ratio of active to control subjects:",
          min = .5,
          max = 5.0, value = 1, step = .5
        ),
        htmlOutput("sizes"),
        sliderInput("dropin", "Drop-in rate:", min = 0, max = .4, value = 0),
        htmlOutput("dropoutsizes"),
      ),
  ),
layout_column_wrap(
  width = 1/2,
```

```
  height = 300,
card(
    full_screen = TRUE,
    card_header("Power"),
    plotOutput("plot")
  ),
card(
    full_screen = TRUE,
    card_header("Data"),
    DT::dataTableOutput("df") ),
),
verbatimTextOutput("eff0"),
      verbatimTextOutput("eff"),
      verbatimTextOutput("eff2"),
    tags$head(tags$style("#eff{color:gray; font-size:12px;
font-style:italic; text-align:left;
max-height: 130px; background: ghostwhite;}")),
    )

server <- function(input, output, session) {
  N <- reactive(input$N)
  del <- reactive(input$del)
  dff <- reactive(input$dff)
  pct <- reactive(input$pct)
  delv <- reactive(seq(del()[1], del()[2], (del()[2] - del()[1]) / 15))
  dffv <- reactive(seq(dff()[1], dff()[2], (dff()[2] - dff()[1]) / 15))
  pctv <- reactive(seq(pct()[1], pct()[2], (pct()[2] - pct()[1]) / 15))
  activev <- reactive(seq(
    active()[1], active()[2],
    (active()[2] - active()[1]) / 15
  ))
  d0 <- reactive(input$d0)
  sd0 <- reactive(input$sd0)

  pow <- reactive(sapply(
    delv(),
    function(x) pwr.t2n.test(n1(), n2(), sig.level = type1(), d = x)$power
  ))
  powdff <- reactive(sapply(
    dffv(),
```

```
  function(x) {
    pwr.t2n.test(n1(), n2(),
      sig.level = type1(),
      d = x / sd0()
    )$power
  }
))
powpct <- reactive(sapply(
  pctv(),
  function(x) {
    pwr.t2n.test(n1(), n2(),
      sig.level = type1(),
      d = (x * d0()) / sd0()
    )$power
  }
))

out1 <- reactive(data.frame(cbind(
  delta = delv() |> round(3),
  power = pow() |> round(3)
)))
out2 <- reactive(data.frame(cbind(
  delta = dffv() |> round(3),
  power = powdff() |> round(3)
)))
out3 <- reactive(data.frame(cbind(
  delta = pctv() |> round(3),
  power = powpct() |> round(3)
)))

out <- reactive(if (input$dmeth == "std") {
  out1()
} else if (input$dmeth == "diff") {
  out2()
} else if (input$dmeth == "pct") out3())

xaxis2_text <- reactive(if (input$dmeth == "std") {
  "~ . "
} else if (input$dmeth == "diff") {
  "~ . / sd0()"
```

```
} else if (input$dmeth == "pct") "~ . * d0()/ sd0()")

xintercept_value <- reactive(if (input$dmeth == "std") {
  pwr.t2n.test(n1(), n2(), sig.level = type1(), power = .8)$d
} else if (input$dmeth == "diff") {
  sd0() * pwr.t2n.test(n1(), n2(), sig.level = type1(), power = .8)$d
} else if (input$dmeth == "pct") {
  pwr.t2n.test(n1(), n2(), sig.level = type1(), power = .8)$d * (sd0() / d0())
})

output$df <- renderDataTable(out())
R <- reactive(input$ratio)
type1 <- reactive(input$type1)
dropin <- reactive(input$dropin)
dropout <- reactive(input$dropout)
active <- reactive(input$active)
dmeth <- reactive(input$dmeth)
n1 <- reactive(R() * N() / (R() + 1) * ((1 - (dropin() + dropout()))))
n2 <- reactive(N() / (R() + 1) * ((1 - (dropin() + dropout()))))



observeEvent(input$N, {
  # browser()
  print(paste0("N: ", input$N))
})
observeEvent(input$dmeth, {
  print(paste0("dmeth: ", input$dmeth))
})




output$df <- DT::renderDataTable(out(),
  server = FALSE,
  filter = "top", extensions = "Buttons",
  options = list(
    paging = FALSE, scrollCollapse = TRUE,
```

```
    buttons = c("copy", "csv", "pdf"),
    dom = "Bt", scrollX = 300, scrollY = 200
  )
)

observeEvent(input$N, {
  # browser()
})
output$plot <- renderPlot(ggplot(out(), aes(x = delta, y = power)) +
  geom_line() +
  geom_hline(yintercept = 0.8, color = "red") +
  geom_vline(xintercept = xintercept_value(), color = "blue") +
  scale_y_continuous(name = "Power", limits = c(0, 1.0), breaks = seq(0, 1, .1)) +
  scale_x_continuous(
    name = dmeth(),
    sec.axis = sec_axis(
      trans = as.formula(xaxis2_text()),
      name = "Std. Effect Units"
    )
  ) +
  theme_bw())


effsize <- renderText(round(pwr.t2n.test(n1(), n2(),
  sig.level = type1(), power = .8
)$d, 3))
output$sizes <- renderText(paste0(
  "ITT analysis per group sample size: N<sub>active</sub> = ",
  round(N() * (R() / (1 + R())), 0), ", N<sub>control</sub>= ",
  round(N() / (1 + R()), 0)
))
output$dropoutsizes <- renderText(paste0(
  "Expected number of completers: N<sub>active</sub> = ",
  round(n1(), 0), ", N<sub>control</sub>= ", round(n2(), 0)
))
output$eff <- renderText(paste0(
  "\ntype 1 error = ", type1(),
  "\ndropout rate = ", dropout(),
  "\ndropin rate = ", dropin(),
```

```
    "\nactive to placebo ratio = ", R()
  ))
  output$eff0 <- renderText(
    paste0("In summary, given the parameters: ")
  )
  output$eff2 <- renderText(
    paste0(
      "A sample size of ", N(), " has 80% power to detect an effect of ",
      effsize()
    )
  )

  state <- reactiveValues(sdel = c(NULL, NULL))

  observeEvent(pct() | sd0() | d0(), {
    print("change in pct begets change in sdel1")
    state$sdel <- pct() * d0() / sd0()
  })
  observeEvent(sd0() | dff(), {
    print("change in pct begets change in sdel1")
    state$sdel <- dff() / sd0()
  })
  observeEvent(sd0() | del(), {
    state$sdel <- del()
  })
  observeEvent(state$sdel, {
    if (!identical(dff() / sd0(), state$sdel)) {
      updateSliderInput(session, "dff", value = state$sdel * sd0())
    }
    if (!identical(del(), state$sdel)) {
      updateSliderInput(session, "del", value = state$sdel)
    }
    if (!identical(pct() * d0() / sd0(), state$sdel)) {
      updateSliderInput(session, "pct", value = state$sdel * sd0() / d0())
    }
  })
  output$variableprint <- renderText(state$sdel)
}
```

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