Constructing a medium complexity shiny app for power analysis

Ronald (Ryy) Glenn Thomas

10/12/23

Table of contents

1 Introduction

2 - Code

```
library(pacman)
p_load(ggplot2, pwr)

ui <- fluidPage(
  fluidRow(
    column(
      2,
      sliderInput("N", "Total Sample Size:", min = 0, max = 500, value = 100),
      # verbatimTextOutput("sizes"),
      sliderInput("dropout", "Dropout Rate:", min = 0, max = .40, value = .10),
      hr(),
      selectInput(
        "deltamethod", "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.deltamethod == 'std'",
      sliderInput("del", "Range for delta",
        min = 0.0,
        max = 1.5, value = c(0, 1)
      ),
    ),
    conditionalPanel(
      condition = "input.deltamethod == 'diff' |
input.deltamethod == 'pct' |input.deltamethod == 'active' ",
      numericInput("placebosd", "Placebo Change Score Standard Deviation:",
        10.0,
        min = 1, max = 100, value = 1
      ),
      numericInput("placebo", "Placebo Change Score:", 10,
        min = -100,
        max = 100
      ),
    ),
    conditionalPanel(
      condition = "input.deltamethod == 'diff'",
      sliderInput("diff", "Diff in change scores",
        min = 0,
        max = 30, value = c(0, 25)
      ),
    ),
    conditionalPanel(
      condition = "input.deltamethod == 'pct'",
      sliderInput("pct", "Percent Reduction",
        min = 1,
        max = 100, value = c(10, 90)
      ),
```

```
      ),
      conditionalPanel(
        condition = "input.deltamethod == '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"),
      ),
    ),
    column(2,
      align = "center", plotOutput("plot"),
      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;}")),
    column(8, DT::dataTableOutput("df"), ),
  )
)
server <- function(input, output, session) {
  N <- reactive(input$N)
  R <- reactive(input$ratio)
```

```
type1 <- reactive(input$type1)
dropin <- reactive(input$dropin)
dropout <- reactive(input$dropout)
del <- reactive(input$del)
diff <- reactive(input$diff)
pct <- reactive(input$pct)
placebo <- reactive(input$placebo)
active <- reactive(input$active)
placebosd <- reactive(input$placebosd)
deltamethod <- reactive(input$deltamethod)
delv <- reactive(seq(del()[1], del()[2], (del()[2] - del()[1]) / 30))
diffv <- reactive(seq(diff()[1], diff()[2], (diff()[2] - diff()[1]) / 30))
pctv <- reactive(seq(pct()[1], pct()[2], (pct()[2] - pct()[1]) / 30))
pctv2 <- reactive(placebo() * seq(
  pct()[1] / 100, pct()[2] / 100,
  (pct()[2] - pct()[1]) / 2000
))
activev <- reactive(seq( active()[1], active()[2],
                         (active()[2] - active()[1]) / 30 ))
activev2 <- reactive(placebo() - activev())
# placebosdrange <- reactive(input$placebosdrange)
n1 <- reactive(R() * N() / (R() + 1) * ((1 - (dropin() + dropout()))))
n2 <- reactive(N() / (R() + 1) * ((1 - (dropin() + dropout()))))

deltav <- reactive({
  if (input$deltamethod == "diff") {
    deltav <- diffv() / placebosd()
  }
  else if (input$deltamethod == "std")  {
    deltav <- delv()
    diffv <- delv()*placebosd()
    pctv = NA
  }
  else if (input$deltamethod == "pct") {
    deltav <- ((pct() / 100) * placebo()) / placebosd()
  }
  return(deltav)
})
```

```
  pow <- reactive(sapply(
    deltav(),
    function(x) pwr.t2n.test(n1(), n2(), sig.level = type1(), d = x)$power
  ))
  powdiff <- reactive(sapply(
    diffv(),
    function(x) {
      pwr.t2n.test(n1(), n2(),
        sig.level = type1(),
        d = x / placebosd()
      )$power
    }
  ))
  powpct <- reactive(sapply(
    pctv(),
    function(x) {
      pwr.t2n.test(n1(), n2(),
        sig.level = type1(),
        d = ((x / 100) * placebo()) / placebosd()
      )$power
    }
  ))

observeEvent(input$N, {
  print(paste0("N: ", input$N))
})
observeEvent(input$deltamethod, {
  print(paste0("deltamethod: ", input$deltamethod))
})
  out <- reactive(data.frame(cbind(
    N = input$N,
    SD = input$placebosd,
    Pl = input$placebo,
    deltav = round(deltav(), 3),
    diffv = round(diffv(), 3),
    pctv = round(pctv(), 3),
    powdiff = round(powdiff(), 3),
    powpct = round(powpct(), 3),
    power = round(pow(), 3)
  )))
```

```
output$df <- DT::renderDataTable(out(),
  server = FALSE,
  filter = "top", extensions = "Buttons",
  options = list(
    paging = FALSE, scrollCollapse = TRUE,
    buttons = c("copy", "csv", "pdf"),
    dom = "Bt", scrollX = 300, scrollY = 200
  )
)
xaxistext <- "test"
output$plot <- renderPlot(ggplot(out(), aes(x = diffv, y = power)) +
  geom_line() +
  geom_hline(yintercept = 0.8, color = "red") +
  geom_vline(xintercept = pwr.t2n.test(n1(), n2(),
    sig.level = type1(), power = .8
  )$d, color = "blue") +
  # scale_x_continuous(name=xaxistext, sec.axis = dup_axis(),
  scale_x_continuous(
    name = xaxistext, sec.axis = sec_axis(~ . / placebosd(),
      name = "standard deviation units"
    ),
    limits = c(input$del[[1]], input$del[[2]])
  ) +
  scale_y_continuous(
    name = "Power", limits = c(0, 1.0),
    breaks = seq(0, 1, .1)
  ) +
  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()
    )
  )
}


shinyApp(ui, server)
```