Constructing a medium complexity shiny app for power analysis

Ronald (Ryy) Glenn Thomas

2024-01-23

Table of contents

Figure 1: under construction

1 Introduction

One of the most common tasks for a biostatistician is the calculation of a required sample size for a two group comparison based on a two sample Student's t-test. While common, this exercise is not trivial. There are numerous parameters to be set by the study investigators.

Want to be able to add CSS to shiny app: read:

2 Methods

3 Code

```
library(pacman)
p_load(DT, ggplot2, pwr, shiny,bsicons, bslib, ggplot2, plotly)
qc = "question-circle"
t1 = "Sample Size:"
t2 = "Total Sample Size. Both groups combined."
ui <- page_sidebar(
  includeCSS("power1_style.css"),
  title = "Power Calculator for Two Group Parallel Designs",
  sidebar = sidebar(
    sliderInput("N", tooltip(list( t1, bs_icon(qc) ), t2 ), 0, 100, 50, 1),
    sliderInput("dropout", "Dropout Rate:", 0, .40, .10),
    htmlOutput("ittsizes"),
    htmlOutput("dropoutsizes"),
    radioButtons("dmeth", "Method for Effect Size", c(
      "SD Units (Cohen)" = "std",
      "Pct reduction" = "pct",
      "Diff in change scores" = "diff",
      "Change in active group" = "active"
    )),
    conditionalPanel(
      condition = "input.dmeth == 'std'",
      sliderInput("del", " delta", 0.0, 1.8, c(.20, 1.8), .1),
    ),
    conditionalPanel(
      condition = "input.dmeth == 'diff'",
      sliderInput("dff", " diff", 0, 10.0, c(1, 9), .1),
    ),
    conditionalPanel(
      condition = "input.dmeth == 'pct'",
      sliderInput("pct", "Pct. Reduction", 0, 1, c(.1, .9), .1),
    ),
    conditionalPanel(
      condition = "input.dmeth == 'active'",
      sliderInput("active", "Treatment group change",
```

```
      min = 1, max = 50, value = c(1, 50)
    )
  ),
  conditionalPanel(
    condition = "input.dmeth == 'diff' |
    input.dmeth == 'pct' | input.dmeth == 'active' ",
    sliderInput("sd0", "Placebo SD:", 1, 20, 5.0, .1),
    sliderInput("d0", "Placebo Change:", 0, 20, 10.0, .1),
  ),
  # sliderInput("del", " delta", 0.0, 1.8, c(.20, 1.8), .1),
  # sliderInput("dff", " diff", 0, 10.0, c(1, 9), .1),
  # sliderInput("pct", "Pct. Reduction", 0, 1, c(.1, .9), .1),
  # sliderInput("sd0", "Placebo SD:", 1, 20, 5.0, .1),
  # sliderInput("d0", "Placebo Change:", 0, 20, 10.0, .1),
  # sliderInput("d1", "Active Change:", 0, 20, 7.0, .1),

  checkboxInput("choice", "Additional parameter settings"),
  conditionalPanel(
    condition = "input.choice == 1",
    numericInput("ratio", "Ratio of active to ctrl:", 1, .5, 5.0, .5),
    sliderInput("dropin", "Drop-in rate:", 0, .4, 0),
    numericInput("type1", "Type one error:", .05, .01, .2, .005),
  checkboxInput("sided", "One sided testing"),
  ),
),
layout_column_wrap(
  width = 1 / 2,
  card(
    height = 700,
    full_screen = TRUE, card_header("Power"), plotOutput("plot")
  ),
  card(
    height = 700,
    full_screen = TRUE, card_header("Data"), DT::dataTableOutput("df")
  ),
  card(
    max_height = 250,
    card_header("Summary"), verbatimTextOutput("eff1")
  ),
  card(
```

```
      max_height = 250,
      card_header("Pdf report"), downloadButton("report", "Download report")
    )
  ),
  # verbatimTextOutput("eff0"),
  # verbatimTextOutput("eff1"),
  # verbatimTextOutput("eff2"),
  # downloadButton("report", "Download report")
)

server <- function(input, output, session) {
  delv <- reactive(seq(input$del[1], input$del[2], (input$del[2] - input$del[1]) / 15))
  dffv <- reactive(delv() * input$sd0)
  pctv <- reactive(delv() * input$sd0 / input$d0)
  n1comp <- reactive(input$ratio * input$N / (input$ratio + 1) * ((1 - (input$dropin + input$d
  n2comp <- reactive(input$N / (input$ratio + 1) * ((1 - (input$dropin + input$dropout))))
  n1itt <- reactive(input$ratio * input$N / (input$ratio + 1))
  n2itt <- reactive(input$N / (input$ratio + 1))

  pow <- reactive(sapply(
    delv(),
    function(x) pwr.t2n.test(n1comp(), n2comp(), d = x)$power
  ))

  powpct <- reactive(sapply(
    pctv(),
    function(x) pwr.t2n.test(n1comp(), n2comp(), d = x * input$d0 / input$sd0)$power
  ))

  out <- reactive(data.frame(cbind(
    std = delv() |> round(3),
    pct = pctv() |> round(3),
    diff = dffv() |> round(3),
    power = pow() |> round(3)
  )))
  out1 <- reactive(
    out()[, c(input$dmeth, "power")] |> setNames(c("delta", "power"))
  )

  xaxis2_text <- reactive(if (input$dmeth == "std") {
```

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

xintercept_value <- reactive(if (input$dmeth == "std") {
  pwr.t2n.test(n1comp(), n2comp(), sig.level = input$type1, power = .8)$d
} else if (input$dmeth == "diff") {
  input$sd0 * pwr.t2n.test(n1comp(), n2comp(), sig.level = input$type1, power = .8)$d
} else if (input$dmeth == "pct") {
  pwr.t2n.test(n1comp(), n2comp(), sig.level = input$type1, power = .8)$d * (input$sd0 / inpu
})

output$df <- DT::renderDataTable(out(),
  server = FALSE,
  filter = "top", extensions = "Buttons",
  options = list(
    paging = FALSE, scrollCollapse = TRUE,
    buttons = c("copy", "csv", "pdf"),
    dom = "Bt", scrollX = 300, scrollY = 200
  )
)
plot_rmd <- reactive({
  chart <- ggplot(out1(), aes(x = delta, y = power)) +
    geom_line() +
    geom_hline(yintercept = 0.8, color = "red") +
    geom_vline(xintercept = xintercept_value(), color = "blue") +
    scale_y_continuous(
      name = "Power",
      limits = c(0, 1.0), breaks = seq(0, 1, .1)
    ) +
    scale_x_continuous(
      name = input$dmeth,
      sec.axis = sec_axis(
        trans = as.formula(xaxis2_text()),
        name = "Std. Effect Units"
      )
    ) +
    theme_bw()
  chart
```

```
})
output$plot <- renderPlot(ggplot(out1(), aes(x = delta, y = power)) +
  geom_line() +
  geom_hline(yintercept = 0.8, color = "red") +
  geom_vline(xintercept = xintercept_value(), color = "blue") +
  scale_y_continuous(
    name = "Power",
    limits = c(0, 1.0), breaks = seq(0, 1, .1)
  ) +
  scale_x_continuous(
    name = input$dmeth,
    sec.axis = sec_axis(
      trans = as.formula(xaxis2_text()),
      name = "Std. Effect Units"
    )
  ) +
  theme_bw())


state <- reactiveValues(sdel = c(NULL, NULL))
sdelv <- reactive(seq(state$sdel[1], state$sdel[2], (state$sdel[2] - state$sdel[1]) / 15))

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

```
  if (!identical(input$pct * input$d0 / input$sd0, state$sdel)) {
    updateSliderInput(session, "pct", value = state$sdel * input$sd0 / input$d0)
  }
})
# effsize <- renderText(round(pwr.t2n.test(n1comp(), n2comp(), sig.level = type1(), power =
output$ittsizes <- renderText(paste0(
  "ITT: N<sub>active</sub> = ",
  round(n1itt(), 0), ", N<sub>control</sub>= ",
  round(n2itt(), 0)
))
output$dropoutsizes <- renderText(paste0(
  "Completers: N<sub>active</sub> = ",
  round(n1comp(), 0), ", N<sub>control</sub>= ", round(n2comp(), 0)
))
eff_rmd <- reactive(
  paste0(
    "In summary, given the parameters:\\",
    "\nSample size  = ", input$N, "\\",
    "\ntype 1 error = ", input$type1, "\\",
    "\ndropout rate = ", input$dropout, "\\",
    "\ndropin rate = ", input$dropin, "\\",
    "\nactive to placebo ratio = ", input$ratio, "\\",
    "\neffect size method = ", input$dmeth, "\\",
    "\nA sample size of ", input$N, " has  power of .80 to detect an effect of ",
    xintercept_value()
  )
)
output$eff1 <- renderText(paste0(
  "In summary, given the parameters:",
  "\nSample size  = ", input$N,
  "\ntype 1 error = ", input$type1,
  "\ndropout rate = ", input$dropout,
  "\ndropin rate = ", input$dropin,
  "\nactive to placebo ratio = ", input$ratio,
  "\neffect size method = ", input$dmeth,
  "\nA sample size of ", input$N, " has  power of .80 to detect an effect of ",
  xintercept_value()
))
# output$eff2 <- renderText(
#   paste0(
```

```
  #     "A sample size of ", input$N, " has 80% power to detect an effect of ",
  #     xintercept_value()
  #   )
  # )
  # observeEvent(input$N, {
  #   browser()
  # })


  output$report <- downloadHandler(
    filename = "report.pdf",
    content = function(file) {
      tempReport <- file.path(tempdir(), "report.Rmd")
      file.copy("report.Rmd", tempReport, overwrite = TRUE)


      params <- list(
        table1 = out(),
        plot1 = plot_rmd(),
        text1 = eff_rmd()
      )

      rmarkdown::render(tempReport,
        output_file = file,
        params = params,
        envir = new.env(parent = globalenv())
      )
    }
  )
}

shinyApp(ui, server)
```

4 setting up modules

5 setting up with golem. step by step

Start with the video

Building a basic Shiny app with Golem - Part I - YouTube

Notes: rstudio provides a template under new file

6 References

How to build a professional R Shiny app — part 1 | by Adrian Joseph, PhD | Towards Dev

How to build a professional R Shiny app — part 2 | by Adrian Joseph, PhD | Towards Dev

How to build a professional R Shiny app — part 3 | by Adrian Joseph, PhD | Towards Dev

Welcome | Outstanding User Interfaces with Shiny

Want to make it extensible: try Golem.

Introduction | Engineering Production-Grade Shiny Apps