

Constructing a medium complexity shiny app for power analysis

Ronald (Ryy) Glenn Thomas

2024-02-23

Table of contents

1	Introduction	2
2	Methods	2
2.1	User interface	2
2.2	Parameters	2
2.3	Visualization	2
2.4	Reporting formats	2
2.5	Specifying effect size	2
2.6	Itt and completers	2
3	Code	2
4	setting up modules	10
5	setting up with golem. step by step	10
6	References	10



1 Introduction

One of the most common tasks for the practicing 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 needed to be set by the team of statisticians and the study investigators.

In this post we'll describe the process we followed to create a shiny app of medium complexity for this task.

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

2 Methods

2.1 User interface

2.2 Parameters

2.3 Visualization

2.4 Reporting formats

2.5 Specifying effect size

2.6 Itt and completers

3 Code

```
library(pacman)
p_load(DT, ggplot2, pwr, shiny, bsicons, bslib, ggplot2, plotly)
qc <- bs_icon("info-circle")
N_tooltip <- tooltip(qc, "Total Sample Size. Both groups combined.")
dropout_tooltip <- tooltip(
  qc,
  "Percent of participants expected to withdraw before the final visit."
```

```

)

ui <- page_sidebar(
  includeCSS("power1_style.css"),
  title = "Power Calculator for Two Group Parallel Designs",
  sidebar = sidebar(
    sliderInput("N", list("Sample Size", N_tooltip), 0, 100, 50, 1),
    sliderInput("dropout", list("Dropout Rate:", dropout_tooltip), 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 = 0, max = 10, value = c(0, 8)
      )
    ),
    conditionalPanel(
      condition = "input.dmeth == 'diff' |
        input.dmeth == 'pct' | input.dmeth == 'active' ",
      numericInput("sd0", "Placebo SD:", 10),
      numericInput("d0", "Placebo Change:", 10)
    )
  ),

```

```

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("onesided", "One sided testing")
)
),
layout_column_wrap(
  width = 1 / 2,
  card(full_screen = TRUE, card_header("Power"), plotOutput("plot")),
  card(full_screen = TRUE, card_header("Data"), DT::dataTableOutput("df")),
  card(card_header("Summary"), verbatimTextOutput("eff1")),
  card(
    card_header("Pdf report"),
    radioButtons("format", "Document format", c("PDF", "HTML", "Word"), inline = TRUE),
    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$d0)
  actv <- reactive(-(delv() * input$sd0) + input$d0)
  n1comp <- reactive(input$ratio * input$N / (input$ratio + 1) *
    ((1 - (input$dropin + input$dropout))))
  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$ratio + 1))
  sided <- reactive(input$onesided + 1)
  pow <- reactive(sapply(
    delv(),
    function(x) {

```

```

    pwr.t2n.test(n1comp(), n2comp(),
      sig.level = input$type1 * sided(), d = x
    )$power
  }
})
out <- reactive(data.frame(cbind(
  std = delv(), pct = pctv(),
  diff = dffv(), active = actv(), power = pow()
)))

output$df <- DT::renderDataTable({
  power_data()
})

power_data <- function() {
  datatable(out(),
    filter = "top", extensions = "Buttons",
    options = list(
      paging = FALSE, scrollCollapse = TRUE,
      buttons = c("copy", "csv", "pdf"),
      dom = "Bt", scrollX = 300, scrollY = 200
    )
  ) |> formatRound(1:ncol(out()), digits = 2)
}

output$plot <- renderPlot({
  power_plot()
})

power_plot <- function() {
  ggplot(out(), aes(x = .data[[input$dmeth]], y = power)) +
    geom_line() +
    geom_hline(yintercept = 0.8, color = "red") +
    geom_vline(xintercept = 11()$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(l1()$xaxis2_text),
      name = "Std. Effect Units"
    )
  ) +
  theme_bw()
}
l1 <- reactive(
  if (input$dmeth == "std") {
    list(
      xaxis2_text = "~ . ",
      effect_units = "std deviations",
      xintercept_value = pwr.t2n.test(n1comp(), n2comp(),
        sig.level = input$type1 * sided(), power = .8
      )$d
    )
  } else if (input$dmeth == "diff") {
    list(
      xaxis2_text = "~ . / input$sd0",
      effect_units = "measurement units",
      xintercept_value = input$sd0 * pwr.t2n.test(n1comp(), n2comp(),
        sig.level = input$type1 * sided(), power = .8
      )$d
    )
  } else if (input$dmeth == "pct") {
    list(
      xaxis2_text = "~ . * input$d0 / input$sd0",
      effect_units = "fractional reduction from placebo",
      xintercept_value = pwr.t2n.test(n1comp(), n2comp(),
        sig.level = input$type1 * sided(), power = .8
      )$d * (input$sd0 / input$d0)
    )
  } else if (input$dmeth == "active") {
    list(
      xaxis2_text = "~ (input$d0 - .) / input$sd0",
      effect_units = "measurement units",
      xintercept_value = -pwr.t2n.test(n1comp(), n2comp(),
        sig.level = input$type1 * sided(), power = .8
      )$d * input$sd0 + input$d0
    )
  }
}

```

```

    }
  )

  state <- reactiveValues(sdel = c(NULL, NULL))
  observeEvent(input$sd0 | input$dff, {
    state$sdel <- input$dff / input$sd0
  })
  observeEvent(input$del, {
    state$sdel <- input$del
  })
  observeEvent(input$pct | input$sd0 | input$d0, {
    state$sdel <- input$pct * input$d0 / input$sd0
  })
  observeEvent(input$active | input$sd0 | input$d0, {
    state$sdel[1] <- (input$d0 - input$active[2]) / input$sd0
    state$sdel[2] <- (input$d0 - input$active[1]) / input$sd0
  })

  observeEvent(state$sdel, {
    if (!identical(input$del, state$sdel)) {
      # browser()
      updateSliderInput(session, "del", value = state$sdel)
    }

    if (!identical(input$dff / input$sd0, state$sdel)) {
      updateSliderInput(session, "dff", value = state$sdel * input$sd0)
    }

    if (!identical(input$pct * input$d0 / input$sd0, state$sdel)) {
      updateSliderInput(session, "pct",
        value = state$sdel * input$sd0 / input$d0
      )
    }
  })
  output$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)
  ))

  output$eff1 <- renderText({
    power_eff()
  })

parameter_text = c(
  "Sample size ",
  "type 1 error",
  "dropout rate",
  "dropin rate",
  "active to placebo ratio",
  "effect size method",
  "placebo standard deviation",
  "placebo change ",
  "A sample size of ",
)

inputs = c(
  "input$N",
  "input$type1",
  "input$dropout",
  "input$dropin",
  "input$ratio",
  "input$dmeth",
  "input$sd0",
  "input$d0",
  "input$N",
)

eff_text = paste(parameter_text, inputs)
power_eff <- function() {
  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$dmeth,
"\nplacebo standard deviation = ", input$sd0,
"\nplacebo change ", input$d0,
"\nA sample size of ", input$N, " has power of .80 to detect:\n",
"   ", round(l1()$xintercept_value, 3), " ", l1()$effect_units
)
}
observeEvent(input$N, {
  # browser()
})

output$report <- downloadHandler(
  # filename = "report.pdf",
  filename = function() {
    paste("report", sep = ".", switch(input$format,
      PDF = "pdf",
      HTML = "html",
      Word = "docx"
    ))
  },
  content = function(file) {
    # report_path = tempfile(fileext = ".Rmd")
    # file.copy("report.Rmd", report_path, overwrite = TRUE)

    aa <- paste("power_plot_", Sys.Date(), ".pdf", sep = "")
    ggsave(aa, power_plot())
    params <- list(
      dt_data = out(),
      plot1 = aa,
      text1 = power_eff(),
      plot2 = power_plot()
    )

    rmarkdown::render("report.Rmd",
      switch(input$format,
        PDF = pdf_document(),
        HTML = html_document(),
        Word = word_document()
      ),

```

```
        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](#)