

Power Analysis App

Astra S. Bryant

5/3/2020

Introduction/Abstract

The goal of this project is to generate a Shiny web app for conducting simple power analyses on user-provided data. Most online resources currently require users to provide pre-processed data in the form of individual and group averages. Excel spreadsheets with embedded lookup tables for calculating power analyses require users to be comfortable with several advanced excel features, and assumes familiarity with lookup tables such as those provided by Bausell and Li (2002).

These requirements may act as a barrier for some researchers. The overall goal of this project is to encourage researchers to conduct power analyses on pilot data by providing a user-friendly web-based application that takes a csv file with raw data as input, and calculates sample sizes for specified power/alpha levels over a set of statistical tests. As much as possible, the results provided by the script have been cross-validating using G*Power.

Dependencies

The shiny script is located in the file `Power_Analysis_App.R`, which in turn calls `ComputeSampleSize.R` for the actual calculations. Taken together, the app uses several libraries to calculate effect sizes and power analyses for the separate statistical tests.

Key amongst these is the WebPower library, which handles the power analyses. For background/additional references relating to the WebPower library, the manual is available [here](#).

```
library(shiny)
library(WebPower)
library(reshape2)
library(effsize)
library(tidyverse)
library(DescTools)
library(powerAnalysis)
library(shinythemes)
source("ComputeSampleSize.R")
```

Define UI

For the version 1.0, the user interface was designed to take a .csv file input containing raw data that will act as pilot data for calculating an effect size. The effect size will then be used to calculate the required sample size, given selected alpha and power levels. Users can select the alpha/power levels they desire; defaults are set to an alpha level of 0.01 and a power level of 0.9.

```
ui <- fluidPage(
  theme = shinytheme("darkly"),

  # App title
```

```

titlePanel("Sample Size Calculator"),

# Sidebar layout with input and output definitions
fluidRow(

  # Sidebar panel for inputs
  #sidebarPanel(
    column(4,
      wellPanel(
        # Input: Select a file
        fileInput("loadfile",
                  "Load a file",
                  multiple = FALSE)
      ),
      wellPanel(
        # Input: Select a statistical test
        selectInput("test_type",
                    "Type of Statistical Test:",
                    choices = c("Unpaired T-test",
                                "Paired T-test",
                                "Chi-squared",
                                "One-way ANOVA",
                                "Two-way ANOVA"),
                    selected = "Unpaired T-test"),

        # Input: Select an alpha level
        radioButtons("alpha",
                     "Alpha Level",
                     choices = c(0.05, 0.01),
                     selected = 0.01),

        # Input: Select a power level
        radioButtons("power",
                     "Power Level",
                     choices = c(0.80, 0.90, 0.95),
                     selected = 0.90)
      )),

  # Main panel for displaying outputs
  column(8,
    conditionalPanel(condition = "output.sample_size",
                     wellPanel(
                       # Output: Test type
                       tags$h3(textOutput("test_type"),
                                class = "text-warning"),

                       # Outout: Results table with
                       # sample size calculation
                       tableOutput("sample_size"),

                       # Output: Notes re: power analysis
                       h4(textOutput("test_notes"),

```

```

class = "text-primary"))

),
conditionalPanel(condition = "output.error",
  wellPanel(
    # Output: display error codes
    htmlOutput("error")))

)),

## Useful input for the user
fluidRow(
  column(12,
    wellPanel(
      h3("Application Instructions",
        class = "text-primary"),
      p("To begin, load a .csv file containing pilot data.  

        Then select the desired statistical  

        test using the pull down menu.  

        You may also select the desired  

        alpha and power levels.  

        The suggested default is an alpha  

        of 0.01 and a power of 0.9."),
      h4("Formatting your .CSV input files",
        class = "text-primary"),
      h5("T-tests",
        class = "text-warning"),
      p("Data should be separated into two  

        adjacent columns (e.g. control vs experimental)."),
      tags$ul(
        tags$li("For unpaired tests,  

        the n per condition does  

        not have to be equal."),
        tags$li("For paired tests, the number of rows  

        must either match;  

        uneven rows will be ignored.")
      ),
      h5("Chi-squared tests",
        class="text-warning"),
      p("Data (proportions) should be  

        separated into two adjacent columns."),
      h5("One-way ANOVA",
        class = "text-warning"),
      p("Data should be separated into at least 3  

        adjacent columns. The number of  

        rows does not have to be even  

        across all conditions."),
      h5("Two-way ANOVA",
        class = "text-warning"),
      p("The power analysis for both main  

        effects and the interaction  

        effect will be calculated.  

        Data should be arranged in columns as follows:"),

```

```

        ul(
          li("Column 1: Factor A Condition 1"),
          li("Column 2: Factor A Condition 2"),
          li("Column 3: Factor B Condition 1"),
          li("Column 4: Factor B Condition 2")
        )
      ),
    wellPanel(
      h4("About this application",
        class = "text-primary"),
      p("This Shiny app calculates statistical power analyses on user-provided data pilot data. The app takes a .csv file with raw data as input, and calculates sample sizes for specified power and alpha levels for several common statistical tests.", class = "text-muted"),
      p("These calculations are primarily powered by the WebPower Library. For background and/or additional references relating to the WebPower library, ",
        a(href = "https://webpower.psychstat.org/wiki/",
          "a manual and a wiki site are available online",
          .noWS = "outside"), '. ',
        .noWS = c("after-begin", "before-end"),
        class = "text-muted"
      )
    )
  )
)

```

Define Server Logic / Run the App

The server logic section calls a separate R script containing the sample size calculator (see below), and provides text/table output to the UI. The final line runs the app.

```

server <- function (input, output){

  dataOutput <- reactive({
    req(input$loadfile) ## Don't run the code unless a file has been selected
    filename <- (input$loadfile$datapath)
    dat <- read.csv(filename)
    result<-ComputeSampleSize(dat, input)
  })

  output$sample_size<- renderTable({
    result<-dataOutput()
    if (!is.null(result)){
      dplyr::select(result,-c(note,method,url))}
  })

  output$test_type <- renderText({
    result<-dataOutput()
    if (!is.null(result)){
      dplyr::pull(result,method)%>%
        dplyr::first()
    }
  })

  output$test_notes <- renderText({

```

```

        result<-dataOutput()
        if (!is.null(result)){
            dplyr::pull(result,note) %>%
            dplyr::first()
        }
    })

    output$error <- renderUI({
        result<-dataOutput()
        if (is.null(result)){
            str0 <-c('<h3 class = "text-danger">Warning</h3>')
            str1 <-c('<p>Number of data columns or rows does
                    not match the selected statistical test. </p>
                    <p>Please pick another file.</p>')
            str2 <-c('<p class = "text-muted">
                    Instructions for correct formatting of .csv files
                    can be found under the Application Instructions
                    section below. </p>')

            HTML(paste(str0,str1, str2,sep = '<br/>'))
        }

        outputOptions(output, 'sample_size', suspendWhenHidden = FALSE)
        outputOptions(output, 'error', suspendWhenHidden = FALSE)
    })
}
shinyApp(ui = ui, server = server)

```

Sample Size Calculator

The sample size is calculated in a separate script `ComputeSampleSize.R`.

Inputs/Outputs

Input from the UI includes the raw data, and the user-selected statistical test type, alpha level, and power level.

Output to the UI includes the sample size for the desired statistical test, as well as the effect size, alpha level, power level, and notes about whether the given sample sizes refer to the number within a group. In some cases, the output of the WebPower script sample size calculation

Types of Power Calculations

Version 1.0 will calculate the required sample sizes for 5 different types of statistical tests:

- Unpaired T-test
- Paired T-test
- Chi-squared Test
- One-way ANOVA
- Two-way ANOVA

For all these tests, the assumption is made that the data are pulled from a normal distribution, i.e. that the statistical test used will be parametric. User's should keep in mind that sample sizes provided may be an underestimation in the case where the intention is to use non-parametric statistical tests. The Prism User Guide suggests that in the absence of easy-to-apply mathematical tools for conducting power analyses of

non-parametric data, values can be estimated calculating the sample size for a parametric test and adding 15%.

Unpaired T-test

```
if (input$test_type == "Unpaired T-test"){
  if (ncol(dat) > 2){return(NULL)}
  dat_melt <- melt(dat,
    variable.name = 'condition',
    value.name = 'result',
    na.rm = TRUE)

  ES <- cohen.d(d = dat_melt$result, f = dat_melt$condition)

  Sample_size <- wp.t(d = ES$estimate,
    alpha = as.numeric(input$alpha),
    power = as.numeric(input$power),
    type = "two.sample") %>%
    unclass() %>%
    as_tibble
  Sample_size$note <- paste("Note:", Sample_size$note)
  print(Sample_size)
```

Paired T-test

```
} else if (input$test_type == "Paired T-test"){
  if (ncol(dat) > 2){return(NULL)}
  dat_complete <- dat[complete.cases(dat),]
  n <- nrow(dat_complete)
  dat_complete$id <- (1:n)

  dat_melt <- reshape2::melt(dat_complete,
    measure.vars = c("Control", "Experimental"),
    variable.name = 'condition',
    value.name = 'result',
    na.rm = TRUE)

  ES <- cohen.d(d = dat_melt$result,
    f = dat_melt$condition,
    subject = dat_melt$id,
    paired = TRUE)

  Sample_size <- wp.t(d = ES$estimate,
    alpha = as.numeric(input$alpha),
    power = as.numeric(input$power),
    type = "paired") %>%
    unclass() %>%
    as_tibble
  Sample_size$note <- paste("Note:", Sample_size$note)
  print(Sample_size)
```

Chi-squared Test

```

} else if (input$test_type == "Chi-squared"){
  if (ncol(dat) >2){return(NULL)}
  if (nrow(dat) >2){return(NULL)}
  dat_melt <- melt(dat,
    variable.name = 'condition',
    value.name = 'proportion')
  ES <- (2 * asin(sqrt(dat_melt$proportion[[1]]))) -
    (2 * asin(sqrt(dat_melt$proportion[[2]])))
  Sample_size <- wp.prop(h = ES,
    alpha = 0.01, ##as.numeric(input$alpha),
    power = 0.9, ##as.numeric(input$power),
    type = "2p") %>%
    unclass() %>%
    as_tibble
  Sample_size$note <- paste("Note:", Sample_size$note)
  print(Sample_size)

```

One-way ANOVA

```

} else if (input$test_type == "One-way ANOVA"){
  if (ncol(dat) < 3){return(NULL)}
  k = ncol(dat) # Number of groups

  dat_melt <- melt(dat,
    variable.name = 'condition',
    value.name = 'result',
    na.rm = TRUE
  )

  anova1 <- aov(formula = dat_melt$result ~ dat_melt$condition)
  summary(anova1)
  etasquared <- EtaSq(anova1)
  CohensF <- sqrt(etasquared[2]/(1-etasquared[2]))
  Sample_size <- wp.anova(k = k,
    f = CohensF,
    alpha = as.numeric(input$alpha),
    power = as.numeric(input$power)) %>%
    unclass() %>%
    as_tibble

  Sample_size$n <- Sample_size$n/Sample_size$k
  Sample_size$note <- c("Note: n is the sample size *in each group*")
  Sample_size$method <- c("One-way ANOVA")
  print(Sample_size)

```

Two-way ANOVA

```

} else if (input$test_type == "Two-way ANOVA"){
  if (ncol(dat) <4){return(NULL)}
  k <- ncol(dat) # Total number of groups
  r_num <- 2 ## Number of factors located in rows,
  ## for a 2 way anova this is hard-wired
  c_num <- k/r_num ## Number of factors located in columns

```

```

## Next couple of lines thanks of K. Zalocusky, PhD
id_list <- c(1:r_num)
id_list2 <- c(1:c_num)

ids <- expand.grid(id_list, id_list2)
ids <- ids[rep(seq_len(nrow(ids)), each = nrow(dat)), ]

dat_melt <- melt(dat)
dat_melt <- cbind(dat_melt, ids)
dat_melt <- dat_melt[!is.na(dat_melt$value),] ## Get rid of NA values

dat_melt$Var1 <- as_factor(dat_melt$Var1)
dat_melt$Var2 <- as_factor(dat_melt$Var2)

anova2 <- aov(formula = dat_melt$value ~ dat_melt$Var1 * dat_melt$Var2)
etasquared <- EtaSq(anova2)
CohensF <- sqrt(etasquared[,2]/(1-etasquared[,2]))
ndf <- summary.aov(anova2) %>%
  flatten_df() %>%
  dplyr::select(Df) %>%
  pull()

Sample_size_Interaction <- wp.kanova(f = CohensF[
  "dat_melt$Var1:dat_melt$Var2"],
                                   ndf = ndf[3],
                                   alpha = as.numeric(input$alpha),
                                   power = as.numeric(input$power),
                                   ng = 4) %>%
  unclass() %>%
  as_tibble %>%
  add_column(name = 'Interaction Effect', .before = "n")

Sample_size_Var1 <- wp.kanova(f = CohensF["dat_melt$Var1"],
                             ndf = ndf[1],
                             alpha = as.numeric(input$alpha),
                             power = as.numeric(input$power),
                             ng = 4) %>%
  unclass() %>%
  as_tibble %>%
  add_column(name = 'Main Effect of Rows', .before = "n")

Sample_size_Var2 <- wp.kanova(f = CohensF["dat_melt$Var2"],
                             ndf = ndf[2],
                             alpha = as.numeric(input$alpha),
                             power = as.numeric(input$power),
                             ng = 4) %>%
  unclass() %>%
  as_tibble() %>%
  add_column(name = 'Main Effect of Cols', .before = "n")

Sample_size <- full_join(x = Sample_size_Var1, y = Sample_size_Var2) %>%
  full_join(y = Sample_size_Interaction)

```



```
Sample_size$n <- Sample_size$n/Sample_size$ng  
Sample_size$note <- c("Note: n is the sample size *in each group*")  
Sample_size$method <- c("Two-way ANOVA")  
print(Sample_size)  
}
```