```
 (function() { //
http://stackoverflow.com/questions/10964966/detect-ie-version-in-javascript

function isIE () { var myNav = navigator.userAgent.toLowerCase(); return
(myNav.indexOf('msie') != -1) ? parseInt(myNav.split('msie')[1]) : false; }
if
(isIE() === 8 || isIE() === 9){ // Set an absolute base href. baseURL =
document.URL; baseURL = baseURL.substring(0, baseURL.lastIndexOf('/'));
document.getElementsByTagName('base')[0].setAttribute('href', baseURL +
'/_w_b4c8a2cd/'); } })();
json2[2014.02.04];jquery[1.11.0];shiny[0.11.1];jqueryui[1.10.4];showdown[0.
3.1];highlight.js[6.2];bootstrap[3.3.1];font-awesome[4.2.0]
                  This example demonstrates the following concepts: *
**Global
variables**: The ****is a reactive expression. Note how it re-evaluates
when the
Variable field is changed, but not when the Show Outliers box is ticked.
Sample
size for a normal population


      Sample size for a normal population




      This page calculates sample size to detect mean of a normal
population

      Difference to detect (as % of mean)
      CV% (coefficient of variation in %)
      Power (Probability of detection as %)
      Results are in the two tabs: 1. Power curve and 2. Sample size.

      Update View


        Sample size:


        Power curve:







Testing of Sample size calculation from a single normal dist



 show with app
  server.R


  ui.R



library(shiny)
dir<- getwd()
setwd(dir)
library(stats)

# Components for a onetime use
```

```
# these area as probability, not %
alfa<- 0.05
pwrLow<- 0.7
pwrUpp<- 0.99
Length<- 200
incr<- (pwrUpp-pwrLow)/Length # change here
 cbind(alfa, pwrLow, pwrUpp, Length, incr)
powr<- array(0,dim=Length)
SS<- array(0, dim=Length)

# power.t.test(delta=.1, sd=.4, sig.level=.05, power=.8)

# Define server logic required to draw a histogram

shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

 #-- request dataset

  output$SSIZE<- renderPrint({ junk<- power.t.test(delta=input$DIFF/100,
sd=input$CV/100, sig.level=alfa,type = c("one.sample"),
power=input$POWER/100)
                               junk$n
     })

  output$powerPlot <- renderPlot({
    for(i in 1:Length){
      Xpwr<- pwrLow+(i-1)*incr
      powr[i]<- Xpwr
      junk<- power.t.test(delta=(input$DIFF)/100, sd=(input$CV)/100,
sig.level=alfa,type = c("one.sample"), power=Xpwr)
      SS[i]<- junk$n
                        }
       plot(SS, powr, main= " Power curve", xlab="Sample size",
ylab="Power")

    # draw the histogram with the specified number of bins
  })

})
#getwd()
#setwd(dir)
# wdir<- getwd()
# wdir
# setwd(wdir)
#> getwd()
# "C:/Users/MSingh/Desktop/ICARDA/MSinghTHDeskTop/M Singh PC
Files/BIOMTRCS/Computer and Biometrics/SpecializedSoftware/RPackagesZipped"
#--------------------


library(shiny)


# how to read excel..


# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Sample size for a normal population"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
```

```
                    Sample size for a normal population
    sidebarPanel(
           # input boxes
      helpText("This page calculates sample size to detect mean of a normal
population"),
           numericInput("DIFF", "Difference to detect (as % of mean) ", 5),
           numericInput("CV", "CV% (coefficient of variation in %)", 15),
           numericInput("POWER", "Power (Probability of detection as %)",
80),
           helpText("Results are in the two tabs: ","1. Power curve and ", "
2. Sample size."),

           submitButton("Update View")

    ),

        # Show a plot of the generated distribution
    mainPanel(
      tabsetPanel(
        tabPanel("Sample size:", verbatimTextOutput("SSIZE")),
tabPanel("Power curve:",  plotOutput("powerPlot"))
        )
      )
    )
))
Code license: MIT
```