USING SHINY R TO CREATE APPS FOR STUDENTS TO USE ONLINE

Deborah Shepherd
Louisiana State University Shreveport
One University Place, Shreveport, LA 71115
Debbie.Shepherd@lsus.edu

Shiny R apps can be highly effective in helping students understand complex statistical concepts by providing an interactive and visual interface. Shiny apps allow students to explore and experiment with different statistical concepts in real-time and online free of charge. This can help them develop a deeper understanding of the material and retain information better compared to traditional methods such as lectures or textbooks. Incorporating Shiny R apps into the educational experience can greatly benefit students in their quest to gain a comprehensive understanding of statistical concepts and techniques. Some knowledge or the R language and RStudio is needed to build a Shiny app.

What is the R language, RStudio, and Shiny R?

R is a free and open-source programming language and software environment for statistical computing and graphics. To make R easy for users, there is an open-source IDE (integrated development environment) for R called RStudio. RStudio's interface is organized in panels so that the user can clearly view graphs, R code, and output all at the same time. It also offers an easy install feature that allows users to import a rich set of R packages and libraries for various statistical methods, machine learning algorithms, and data analysis tasks. R is being used by a variety of industries, including finance, healthcare, marketing, and scientific research, to name a few. R can be downloaded from the official CRAN (Comprehensive R Archive Network) website at the following URL: https://cran.r-project.org/. R Studio can be downloaded at the following URL: https://rstudio.com/products/rstudio/download/.

One of R's packages is the Shiny package that allows users to create interactive web applications using R code. With Shiny, users can build dynamic, responsive user interfaces that allow users to interact with the data and the models behind it. There are many diverse apps that can be built using Shiny (see https://Shiny.rstudio.com/gallery/) , but the apps referred to in this paper are apps built for students to visualize statistical concepts or practice problems typical in an elementary statistics course. The link to a Shiny app may be placed on learning platforms such as Moodle, Blackboard, Canvas, etc.

In this paper, I present several Shiny apps I post on Moodle (the LSUS learning platform) for my students. The first app presented below demonstrates how Shiny apps may be used to help students practice problems in elementary statistics courses while the second app showcases the use of Shiny apps as a tool for visualizing complex ideas.

One of the concepts the elementary statistics students struggle with is hypothesis testing. The following is an illustration of what a student will encounter when utilizing the Shiny app designed for practicing lower-tail hypothesis tests.

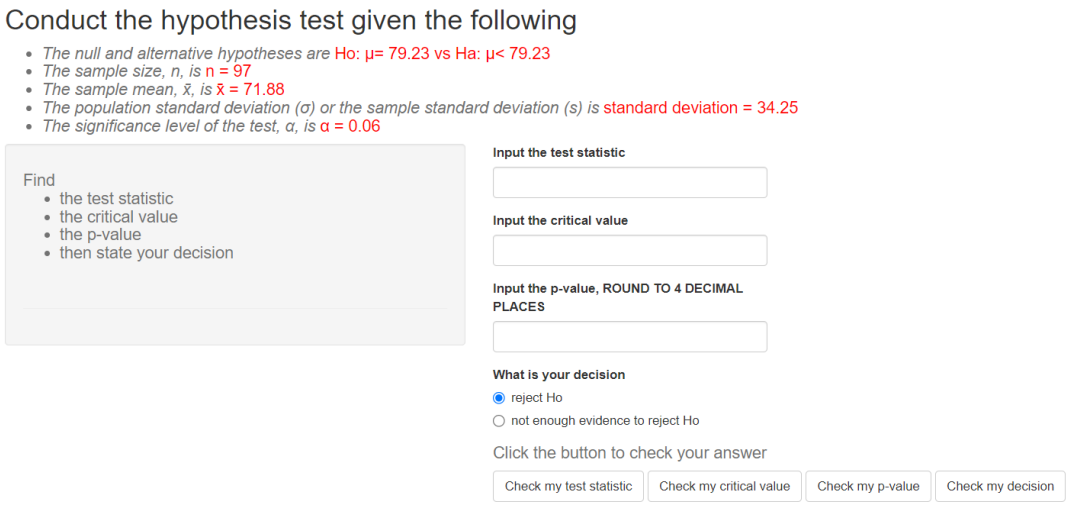

Figure 1. A Shiny app created for students to practice calculating various components of a hypothesis test.

Figure 1 illustrates a Shiny app created for students to practice calculating various components of a hypothesis test. The student is given enough information to calculate the test statistic, the critical value, the p-value, and conclusion of the test. The app is interactive in that a student may check their answer up to 3 times before the correct answer is given. If the student gets the answers correct, the "check" button at the bottom will "create" text that tells the student their answer is correct, otherwise, the text will tell the student their answer is incorrect and give the student the number of tries that are left. For example, Figure 2 below shows a screenshot of the app when a student has given the correct answer for the test statistic, an incorrect answer for the critical value, and a correct answer for the p-value and decision.

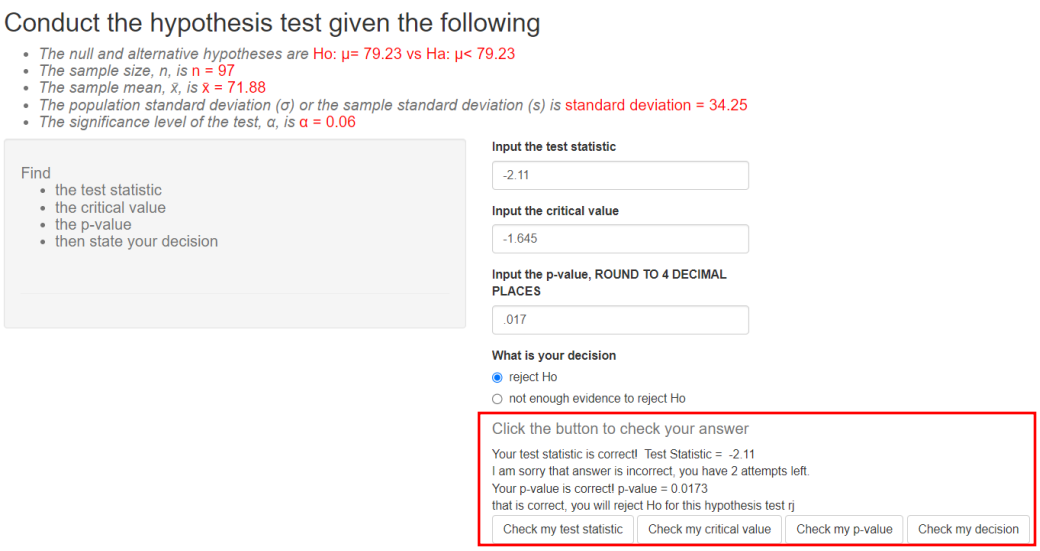

Figure 2. Shiny app showing the student they have given the correct answer for the test statistic, an incorrect answer for the critical value, and a correct answer for the p-value and decision.

Each time the student refreshes the app, the numbers in the problem will change, allowing the student to practice the problems as many times as desired. The interactive nature of the app provides an engaging way for students to enhance their understanding and proficiency in performing these calculations.

Aside from practicing problems in an elementary statistics course, a Shiny app can be utilized to enhance the student's comprehension of the relationship between components of a hypothesis test through visual representation. The following app is an example of a Shiny R app used to visualize the relationship between the p-value, test statistic, critical value, and significance level (alpha) of a hypothesis test. The app allows students to interact with a plot of the standard normal distribution, revealing the connection between the significance level (alpha) and the critical value(s), and the link between the test statistic and the p-value. The app allows users to change the significance level and the test statistic in order to observe the effect on the critical value and the p-value, respectively. Additionally, the app enables the user to choose between upper, lower, or two-tailed tests and demonstrates the impact of their selection on the components of the test. A screenshot of the app is shown below in Figure 3.

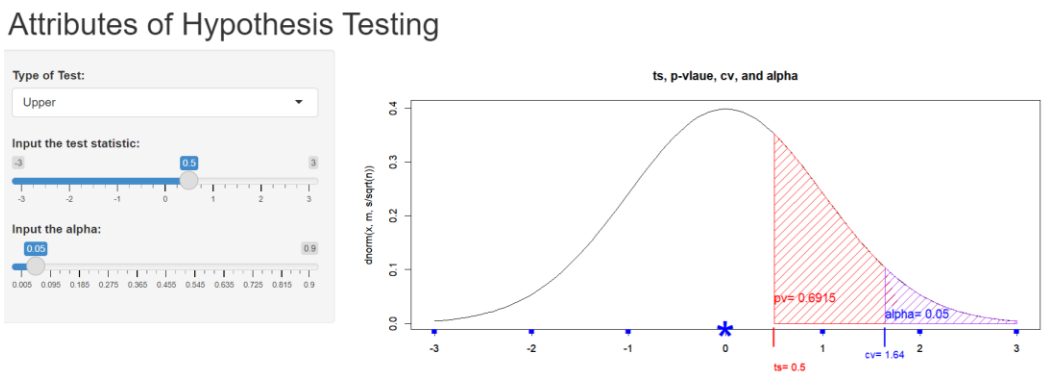

Figure 3. A Shiny app used to visualize the relationship between the p-value, test statistic, critical value, and significance level (alpha) of a hypothesis test.

As you can see on the left-hand side, there is a panel where the type of test (lower-tail, upper-tail, or two-tailed) can be selected along with sliders to input the test statistic and the value of alpha.

The p-value is the area shaded in the red and corresponds to the area under the curve either to the right, left, or both sides of the test statistic depending on whether the test is an upper, lower, or two-tailed test, respectively. Alpha is the area shaded in blue and corresponds with the critical value(s) of the test. Again, alpha will be the area under the curve to the right, left, or both sides of the critical value(s) depending on the direction of the hypothesis test.

How Do Shiny Apps Work?

A Shiny app consists of two main parts: the user interface (UI) and the server function.

User Interface (UI): The UI defines the layout of the app, including the inputs, outputs, and design elements. The UI is created using R code and is written using functions from the Shiny library.

Server Function: The server function is responsible for processing the inputs from the user interface and generating the outputs. The server function is also written using R code and is where the data analysis, modeling, and computation takes place. The server function takes the inputs from the UI, performs the calculations, and returns the results to be displayed in the UI.

Together, the UI and the server function define the functionality and behavior of the Shiny app. When the app is run, Shiny connects the UI and server function and allows the user to interact with the app. The output generated by the server function is displayed in the UI, and changes made by the user in the UI are communicated to the server function in real-time.

Figure 4 gives an example of a simple Shiny app that allows users to input a number and an exponent and outputs the result of the number raised to the exponent. The UI and server function are in the same file. The call to shinyApp(ui,server) will deploy the app and connect the UI and the server function. The file needs to be named app.R.

```
# Define UI for inputting a number and an exponent  and outputting that number raised to the exponent.

ui <- fluidPage(

# App title ----
titlePanel("EXAMPLE -- SHINY"),

# Sidebar layout with input and output definitions ----
sidebarLayout(

# Sidebar panel for inputs ----
sidebarPanel(

# Input: Text for providing a caption ----
# Note: Changes made to the caption in the textInput control
# are updated in the output area immediately as you type
textInput(inputId = "caption",
label = "You can change the title in the output here:",
value = "Print Your Own Title Here"),

# Input: Provide a blank for the user to input a value, let the default value be 0
numericInput(inputId = "number",
label = "Type in a Number",
value=0),

# Input: Provide a blank for the user to input an exponent, let the default value be 10
numericInput(inputId = "exp",
label = "Raise your number to this power",
value = 10)
),

# Main panel for displaying outputs ----
mainPanel(

# Output: Formatted text for caption ----
h3(textOutput("caption", container = span)),

# Output: Provide a space for R to return the number raised to the exponent
verbatimTextOutput("Pownumber"),
)
)
)

# Define server logic to read the number given by the user and raise it to the given exponent
server <- function(input, output) {

# Create caption ----
# The output$caption is computed based on a reactive expression that returns input$caption. When the user
# changes the "caption" field:
#
# 1. This function is automatically called to recompute the output
# 2. New caption is pushed back to the browser for re-display
#

output$caption <- renderText({
input$caption
})

# Generate the output
# The output$Pownumber depends on the numericInput reactive expressions input$number and input$exp, so
# will be re-executed whenever input$number or input$exp changes.

output$Pownumber <- renderPrint({
paste("the number ",input$number," rasied to the power ",input$exp," is ",input$number^input$exp)
})
}

#this makes a call to Shiny package that will deploy the app connecting the ui and the server function.
shinyApp(ui,server)
```

Figure 4. A simple Shiny app that allows users to input a number and an exponent and outputs the result of the number raised to the exponent.

The UI in the app in Figure 4 is defined using the fluidPage function from the Shiny library and consists of two panels: a sidebarPanel for input fields and a mainPanel for displaying outputs. The server logic is defined using a server function that calculates the output based on the inputs and returns the results to the UI for display. The final line of code, shinyApp(ui, server), connects the UI and server functions and launches the Shiny app. Figure 5 below shows a screenshot of the app after it is deployed.

Figure 5. Simple Shiny app that takes a number and raises it to an exponent.

Once the Shiny app is written, you may publish it for free at shinyapps.io, a server that is maintained by RStudio and provides hosting services for Shiny apps. To publish a Shiny app on shinyapps.io, follow these steps:

- Set shinyapps.io as your publishing default.
- Run your Shiny app from the RStudio console and click the "Publish" button on the top right.
- Choose the files you want to publish and select your shinyapps.io account.
- Provide a title for your Shiny app and click "Publish."
- To find the URL of your Shiny app, go to your shinyapps.io account where the URL will be displayed.

References:

Shiny User Guide: https://Shiny.rstudio.com/tutorial/

Shiny Gallery: https://Shiny.rstudio.com/gallery/

R Development Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/

Note: The Shiny package is developed and maintained by RStudio, Inc.