CREATING INTERACTIVE SHINY APS IN R

R Shiny Apps allow authors to write small self contained applications to better display results/data in an interactive manner or to allow skeptical readers to perform live robustness tests.

Here is an example of what a finished App could look like. Click on the *March-June Pillar 1 Synthetic Controls* tab to perform your own live robustness analysis for this recent paper looking at the impact of *Track and Trace* on Covid-19 spread on the Isle of Wight.

Creating your own App

I will use the default Shiny App example to illustrate its features. You can follow along by simply selecting File - New File - Shiny Web App in R-studio. First, we will load the library.

```
library(shiny)
```

Every Shiny App has three components- the UI (user interface), the server, and the one that calls the application. The UI determines the *front end* of your app i.e. how it looks and feels to the user. In this case, it'll contain two tabs- one that displays a histogram, and the other that will just contain a piece of text.

```
# Define UI for application that draws a histogram and another which
# has some simple text. In the first tab, I include a sidebar with a slider
# allowing the user to change the number of bins in a histogram and then
# plot the histogram next to it.

ui <- fluidPage(
   tabsetPanel(

   tabPanel("Old Faithful Geyser Data",
   # Application title

   # Sidebar with a slider input for number of bins
   sidebarLayout(
      sidebarPanel(
        sliderInput("bins",
                    "Number of bins:",
                    min = 1,
                    max = 50,
                    value = 30)
      ),

      # Show a plot of the generated distribution
      mainPanel(
```

```
          plotOutput("distPlot")
        )
      )
    ),
  tabPanel("Another Tab",
         textOutput("text1")
    ) # end "About" tab
    ) # end tabsetPanel
)
```

The second component (i.e. the *server*) draws the dynamic items from user inputs. This section is a little more heavy on the R code. We take *bins* from the user input and define some plot object *distPlot* to pass back to the UI.

```
# Define server logic required to draw a histogram
server <- function(input, output) {

  output$distPlot <- renderPlot({
    # generate bins based on input$bins from ui.R
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })

   # now make the text output
   output$text1 <- renderText({
      "This is another tab, SO exciting!"
   })
}
```

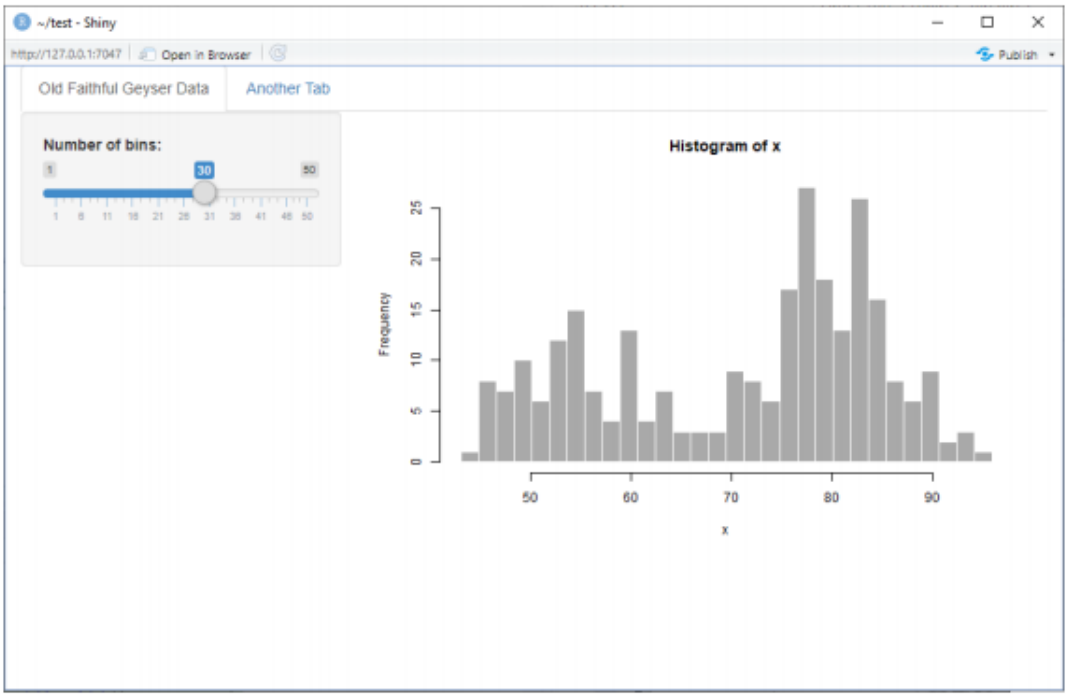
Finally, we can run the application by combining the front end and back end together.

```
# Run the application
shinyApp(ui = ui, server = server)
```

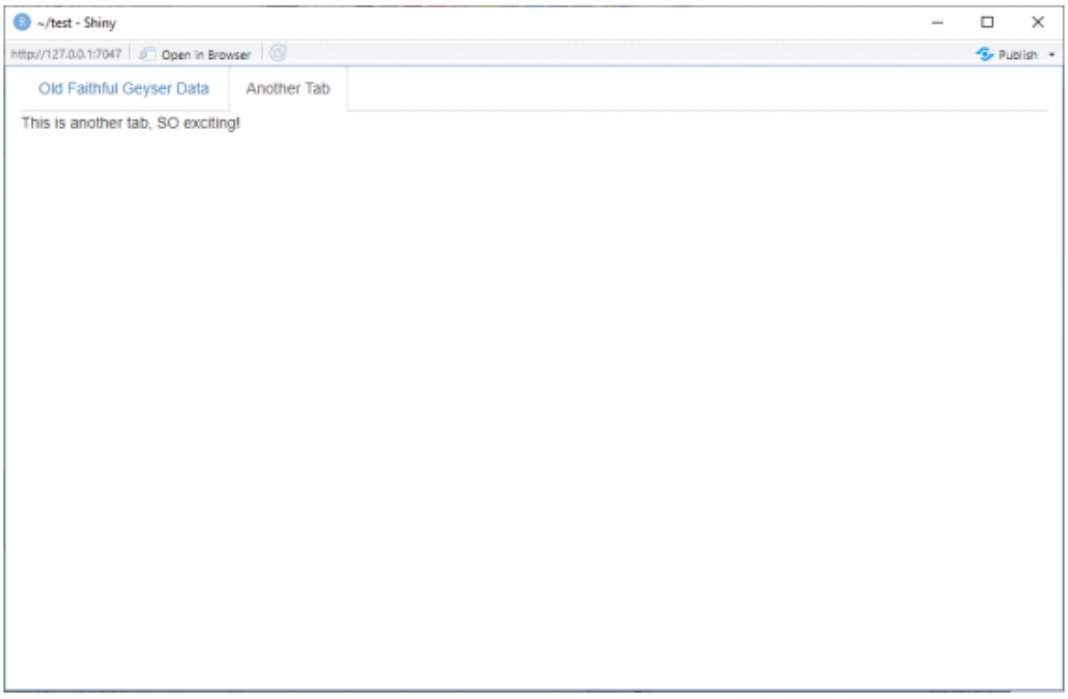
The finished app looks like this:

Figure 1: Screen shots of the Shiny App

(a) Main page



(b) Another tab



Once you've created your own Shiny App, you can publish it using shineyapps.io or another method described here.

Luke Milsom, DPhil Candidate in Economics, Oxford
19 November 2020