A basic guide to making an R Shiny app and publishing it online

Creating a Shiny app

STEP 1. Install R package 'shiny'.

Other potentially useful packages:
- 'DT' (more options for formatting tables)
- 'shinydashboard' (more options for UI layout)
- 'shinyWidgets' (more options for buttons and sliders
- 'rsconnect' (for publishing the app online in www.ShinyApps.io)

STEP 2. Figure out what elements are needed, the general layout of the page, what inputs to take, and what outputs to generate.

STEP 3. Create an empty R script called **app.R** in a folder where you want to store the app's files, e.g., datasets. It's best if each app lives in a unique directory. Some people write separate ui.R and server.R files instead of a single app.R file containing everything; either way is fine.

Each app.R script should contain 3 things (example code on next page):

1. a "**ui**", where details about the layout and appearance are specified; everything from where elements are located on the page to the actual text content, font size and spacing

```
ui <- fluidPage( … )
```

2. a "**server**", which contains instructions on how to build the displayed content; this can include taking user input that's entered into the UI, to change what's displayed

```
server <- function(input, output) { … }
```

3. calling '**shinyApp**' at the end of the script, which runs the app by combining 'ui' and 'server' instructions

```
shinyApp(ui, server)
```

The different elements on a page are called "panels". Panels can be listed using the '**fluidPage**' function, where you lay out elements in a specified number of rows and columns, The heights and widths of the elements self-adjust as the size of the UI window changes (hence 'fluid'). For example, there could be 1 column of 3 panels: one panel containing the page title, a second panel with a numeric input, and a third panel displaying a graph. Other functions can be used within 'fluidPage', like 'fluidRow' or 'mainPanel'.

In the '**ui**', different types of input (e.g. numeric, checkbox, slider, drop-down menu) can be specified in different panels. These inputs can be called later by the '**server**' and used to generate some output, like a calculated variable or a plot. Inside 'server', variables can be specified as being "**reactive**", meaning that the user can keep entering different information into the UI (e.g. adjusting a slider on a scale), and the output will change in response.

<u>Basic structure of app.R file</u>

```
library(shiny)

ui <- fluidPage(

      sliderInput() #user inputs go here

      sliderInput()

      mainPanel( plotOutput( #someID ) )  # a panel will be generated

      )

server <- function(input, output) {

      # “input” covers any variable that had a label above: VarA, VarB

      # specify how to use inputs, enclosing expressions with curly brackets
      plotInput <- reactive({
            # call user inputs using input$VarA, input$VarB
            ggplot()
            })

      # generate the plot into panel specified earlier
      # curly { } may not be necessary here, but best practice to include
      output$someID <- renderPlot( {plotInput} )

shinyApp(ui, server)  # put the ‘ui’ and ‘server’ together into an app
```

<u>To run the app</u>

Method A. Click ‘Run’ button in RStudio

Method B. In console: `setwd(‘directory’); runApp(‘app.R’)`

While the app is running, the console window will be busy. To stop, close the App window or click on Stop sign.

If you make changes to the code in the app.R file, save those changes, then click ‘Reload App’ to incorporate the changes in the running app.

<u>**Publishing a Shiny app online**</u>

Install package ‘rsconnect’ in R.

Create an account in www.shinyapps.io

There is a one-time process to authorize a connection between the ShinyApps website and Rstudio on your local computer. Go to Account > Tokens to generate a token. Then type some commands in the RStudio console (check the instructions in www.shinyapps.io):

```
rsconnect::setAccountInfo(name="…", token="…", secret="…")
```

There should be at least an app.R file and the associated dataset stored together in some folder on the local computer (or separate ui.R and server.R files instead of a single app.R file).

In RStudio, set the working directory to where the app.R is stored, and type `runApp("filename")` in the console to check that the app works properly.

If the app runs properly, click the Publish button in the top right corner of RStudio.
Check the project name, and project files. ShinyApps seems to like having a standard filename of “app.R”, so humor it if you like. There should be some progress messages displayed in the RStudio console as the app is being published.

When the app is published online, a browser window should pop up with the new web address of the Shiny app.