10. RShiny - An Introduction

CT5102 - J. Duggan

RShiny Overview

- Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R.
- Shiny apps are contained in a single script called app.R.
- The script app.R lives in a directory, app.R has three components:
  - a user interface object, which controls the layout and appearance of your app
  - a server function contains the instructions that your computer needs to build your app
  - a call to the shinyApp function, creates Shiny app objects from an explicit UI/server pair

User Interface Object (1/2)

```
ui <- fluidPage(
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(
    # Sidebar panel for inputs ----
    sidebarPanel(

      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),
```

User Interface Object (2/2)

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

      # Output: Histogram ----
      plotOutput(outputId = "distPlot")

    )
  )
)
```

Server Function (1/2)

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

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram
  # is wrapped in a call to renderPlot to indicate
  # that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
```

Server Function (2/2)

```
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x),
                length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB",
         border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
shinyApp(ui = ui, server = server)
```

Running the App

Observations

- At one level, the Hello Shiny server function is very simple.
- The script does some calculations and then plots a histogram with the requested number of bins.
- However, most of the script is wrapped in a call to renderPlot. This gets called every time the slider button changes value

Run the app

shinyApp(ui = ui, server = server)

Basic Template

```
library(shiny)

# Define UI ----
ui <- fluidPage(

)

# Define server logic ----
server <- function(input, output) {

}
```

Layout

- Shiny uses the function fluidPage to create a display that automatically adjusts to the dimensions of your user's browser window. Y
- You lay out the user interface of your app by placing elements in the fluidPage function.
- For example, the ui function below creates a user interface that has a title panel and a sidebar layout, which includes a sidebar panel and a main panel. Note that these elements are placed within the fluidPage function.

```
ui <- fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel("sidebar panel"),
    mainPanel("main panel")
  )
)
```

RShiny Summary

- Implements generic-function OO
- A special type of function called a generic function decides which method to call (i.e. method dispatch)
- S3 is a very casual system, it has no formal definition of classes
- Inheritance can be used to leverage existing R S3 classes