Introduction to Shiny

LondonR Workshop
November 21st 2017

Nick Howlett
Data Scientist
Email: nhowlett@mango-solutions.com
WiFi

The Cloud WiFi
Workshop Aim

Be able to develop a simple Shiny App with standard inputs and outputs
Outline

• A Basic Shiny app
• Defining the User Interface
• Displaying Outputs
• Reactivity
• Beyond the Basics
Workshop resources

- R (version 3.1.2)
- RStudio
- Shiny (version 0.11)
Workshop structure

• 2 hours
• Presentation format
• Worked examples of creating apps
• Exercises during the workshop
What is Shiny?

• R Package for Interactive Web Apps developed by RStudio

• Gives the power of R in a convenient user interface

• Can be written entirely in R
A Basic Shiny App

• A basic app requires:
  – A user interface script
  – A "Server" script

• Runs using the `runApp()` function
The User Interface Script

- Defines the components of the user interface
  - Page titles
  - Input options
  - Outputs

- Defines what the user will see and interact with
The Server Script

- Contains the information to build the app

- Requires a call to the function `shinyServer()`
  - Contains a function with parameter input and output

- Defines what happens in R
Worked Example 1

My First Shiny App!

Enter text here:
Welcome to LondonR!

You entered the text: Welcome to LondonR!
Worked Example 1 - UI

```r
fluidPage(
  titlePanel("My First Shiny App!"),
  sidebarLayout(
    sidebarPanel(
      textInput("myText", "Enter text here:"),
    ),
    mainPanel(
      textOutput("niceTextOutput")
    )
  )
)
```
Worked Example 1 - Server

```r
function(input, output){
  output$niceTextOutput <- renderText(paste("You entered the text: ", input$myText))
}
```
Schematic of a Basic Shiny app

ui.R

shinyUI(..
  textInput(“TXT”),
  textOutput(“text”)
  ..)

server.R

shinyServer(..
  output$text <- renderText(“TXT”)
  ..)

App

My First Shiny App!

“You entered the text: Welcome to LondonR!”

R
Layouts

• Example 1 used `sidebarLayout()`
• There are a number of possible layouts
• In this workshop we will only use
  – `sidebarPanel()`
    • Useful for `.Input()` functions
  – `mainPanel()`
    • Useful for `.Output()` functions
Shiny Workshop UI Boiler Plate

```r
fluidPage(
  titlePanel("Title Here!"),
  sidebarLayout(
    sidebarPanel(
      #INPUTS GO HERE
    ),
    mainPanel(
      #OUTPUTS GO HERE
    )
  )
)
```
Input Controls

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>textInput()</code></td>
<td>Text string input</td>
</tr>
<tr>
<td><code>numericInput()</code></td>
<td>Numeric value input</td>
</tr>
<tr>
<td><code>selectInput()</code></td>
<td>Select single or multiple values from drop down list</td>
</tr>
<tr>
<td><code>sliderInput()</code></td>
<td>Numeric (single or range) “slider” input</td>
</tr>
<tr>
<td><code>radioButtons()</code></td>
<td>Set of radio button inputs</td>
</tr>
<tr>
<td><code>fileInput()</code></td>
<td>File upload control</td>
</tr>
</tbody>
</table>
Worked Example 2

My First Shiny App!

Enter text here:
Welcome to LondonR!

Select a number:
50

Select from the dropdown:
A
sidebarPanel(
    textInput("myTextInput", "Enter text here:"),
    numericInput("myNumberInput", "Select a number:",
                  value = 50, min = 0, max = 100, step = 1),
    selectInput("mySelectInput", "Select from the dropdown:",
               choices = LETTERS[1:10])
)
Main Panel

• Define the contents of the main panel using the function `mainPanel()` function

• Can contain outputs using the *Output functions

• Can include HTML using a series of functions that replicate the HTML tags
HTML Formatting

- We don't need to use HTML tags
- Shiny includes a series of equivalent functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p()</code></td>
<td>A paragraph of text</td>
</tr>
<tr>
<td><code>h*()</code></td>
<td>A level * (1, 2, 3,...) header</td>
</tr>
<tr>
<td><code>code()</code></td>
<td>A block of code</td>
</tr>
<tr>
<td><code>img()</code></td>
<td>An image</td>
</tr>
<tr>
<td><code>strong()</code></td>
<td>Bold text</td>
</tr>
<tr>
<td><code>em()</code></td>
<td>Italic text</td>
</tr>
</tbody>
</table>
My First Shiny App!

Enter text here:

Welcome to LondonR!

Select a number:

50

Select from the dropdown:

A

Using HTML in Shiny

This is a paragraph of text that is included in our main panel. **This text will be in bold.**

You entered the text: Welcome to LondonR!
You selected the number: 50
You selected option: A
Worked Example 2 - UI

```r
mainPanel(
  h4("Using HTML in Shiny"),

  p("This is a paragraph of text that is included in our main panel.", strong("This text will be in bold.")),

  textOutput("niceTextOutput"),

  textOutput("niceNumberOutput"),

  textOutput("niceSelectOutput")
)
```
Worked Example 2 - Server

```r
shinyServer(function(input, output){
  output$niceTextOutput <- renderText(paste("You entered text: ", input$myTextInput))
  output$niceNumberOutput <- renderText(paste("You selected the number: ", input$myNumberInput))
  output$niceSelectOutput <- renderText(paste("You selected option: ", input$mySelectInput))
})
```
Exercise 1

Build a simple Shiny application that takes a date input and returns the following text:

– What day of the week is it (e.g. “Wednesday”)
– What month it is (e.g. “December”)
– What year it is

> format(Sys.Date(), "Day: %A Month: %B Year: %Y")
[1] "Day: Thursday Month: November Year: 2017"
>
Exercise 1 - UI

```r
require(shiny)
fluidPage(
  titlePanel("Exercise 1"), # Define the header for the page
  sidebarLayout( # Set up the page to have a sidebar
    sidebarPanel( # Define the contents of the sidebar
      dateInput("dateInput", label = "Select a date:"
    ),
    mainPanel( # Define the contents of the main panel
      textOutput("dateOutput")
    )
  )
)
```
Exercise 1 - Sever

```r
require(shiny)

function(input, output) {
  output$dateOutput <- renderText(
    format(input$dateInput,
           format = "A %A in %B. The year is %Y")
  )
}
```
Defining Outputs

• So far we have just output text

• Shiny also allows us to output graphics, data and images

• We have to define the output in the UI and the Server scripts using different functions
## Rendering Outputs

<table>
<thead>
<tr>
<th>Output Type</th>
<th>server.R Function</th>
<th>ui.R Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td><code>renderPrint()</code></td>
<td><code>textOutput()</code></td>
</tr>
<tr>
<td>Data</td>
<td><code>renderDataTable()</code></td>
<td><code>dataTableOutput()</code></td>
</tr>
<tr>
<td>Plot</td>
<td><code>renderPlot()</code></td>
<td><code>plotOutput()</code></td>
</tr>
<tr>
<td>Image</td>
<td><code>renderImage()</code></td>
<td><code>imageOutput()</code></td>
</tr>
</tbody>
</table>
Worked Example 3 - Render Data

• From the user interface select a dataset from a dropdown menu

• Display the data in a dataTable
## Render Data in a Shiny App

### Select from the dropdown:
- `airquality`

### Table
<table>
<thead>
<tr>
<th>Ozone</th>
<th>Solar.R</th>
<th>Wind</th>
<th>Temp</th>
<th>Month</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>190</td>
<td>7.4</td>
<td>67</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>118</td>
<td>8</td>
<td>72</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>149</td>
<td>12.6</td>
<td>74</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>313</td>
<td>11.5</td>
<td>62</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.3</td>
<td>56</td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>14.9</td>
<td>66</td>
<td></td>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>299</td>
<td>8.6</td>
<td>65</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>19</td>
<td>99</td>
<td>13.8</td>
<td>59</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>20.1</td>
<td>61</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>8.6</td>
<td>69</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>6.9</td>
<td>74</td>
<td>5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>9.7</td>
<td>69</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>290</td>
<td>9.2</td>
<td>66</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>274</td>
<td>10.9</td>
<td>68</td>
<td>5</td>
<td>14</td>
</tr>
</tbody>
</table>
sidebarLayout(
    sidebarPanel(
        selectInput("selectInput", "Select from the dropdown:",
            choices = c("airquality", "iris", "mtcars"))
    ),
    mainPanel(
        dataTableOutput("dataOutput")
    )
)
output$dataOutput <- renderDataTable(switch(input$selectInput, 
  "airquality" = airquality, 
  "iris" = iris, 
  "mtcars" = mtcars) 
)
Worked Example 4 - Render Plots

- Select a column of the mtcars data from a drop down menu
- Plot a histogram of the data
Render Plot in a Shiny App

Histogram of mpg

Select column:

mpg
Worked Example 4 - UI

```r
sidebarLayout(
    sidebarPanel(
        selectInput("selectInput", "Select column:",
                     choices = colnames(mtcars))
    ),
    mainPanel(
        plotOutput("plotOutput")
    )
)
```
Worked Example 4 - Server

```r
output$plotOutput <- renderPlot(hist(mtcars[,input$selectInput],
   main = paste("Histogram of", input$selectInput),
   xlab = input$selectInput))
```
Exercise 2

Create a Shiny application that takes:

– A numeric value between 1 and 500
– A colour
– A main title

Use these inputs to create an output histogram of random data from any distribution where n is the numeric input
Exercise 2 - UI

```r
fluidPage(
  titlePanel("Exercise 2 - Render Plot in a Shiny App"),
  sidebarLayout(
    sidebarPanel(
      numericInput("numberInput", "Select size of data:",
        min = 0, max = 500, value = 100),
      selectInput("colInput", "Select a colour:",
        choices = c("red", "yellow", "blue", "green"))
    ),
    mainPanel(
      mainPanel(
        plotOutput("plotOutput")
      )
    )
  )
)
shinyServer(function(input, output){
  output$plotOutput <- renderPlot(
    hist(rnorm(input$numberInput), col = input$colInput)
  )
})
Reactivity

• Consider the last exercise...
  – Suppose we want to change the colour of the plot, what happens to the data?
Reactivity

• Each time we change an option the data is simulated again

• Suppose this was reading in a large dataset, connecting to a database etc.
The `reactive()` Function

- This lets us create a reactive function

- The function is only called when the relevant inputs are updated

- Our data is only updated when the number of simulations is changed
Worked Example 5

Render Plot in a Shiny App

Select size of data:
100

Select a colour
yellow
red
yellow
blue
green

Histogram of simData()
Worked Example 5 - Server

```r
simData <- reactive({
  rnorm(input$numberInput)
})

output$plotOutput <-
  renderPlot(hist(simData(), col = input$colInput))
```
Beyond the basics

• Changes to layouts
• Including tabbed pages
• Include CSS to style the page
• Incorporate Shiny and Markdown
• Share your app
• ...

All covered on our 1 day Getting Started with Shiny course
Shiny Themes

• A new package that allows us to change the bootstrap theme

• Requires Shiny v0.11

• Available on CRAN

http://rstudio.github.io/shinythemes/
Worked Example 6

Render Plot in a Shiny App

Select size of data:

100

Select a colour

red

Histogram of simData()
require(shinythemes)

fluidPage(
  theme = shinytheme("cerulean"),
  ...
)

Worked Example 6 - UI
Shiny Dashboard

• Package developed by RStudio for producing Dashboards with Shiny

• Available on github
What Next?

- This evening's LondonR meeting!
- EARL 2018 – keep an eye out for abstracts opening at the end of the month
LondonR tonight

• Dependency Elicitation using Expert Judgement Victory Idowu, London School of Economics and Political Science

• Development of Shiny app tools to simplify and standardize the analysis of hemostasis assay data Colin Longstaff, NIBSC

• Generalised linear models in R Markus Gesmann, Vario Partners
Follow Mango!

@mangothecat
@earlconf