MANGO SOLUTIONS

Introduction to Shiny

Women in Data Workshop
30th November 2017

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

Minimum requirements:

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

Workshop structure

- 1 hour 25 minutes
- 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

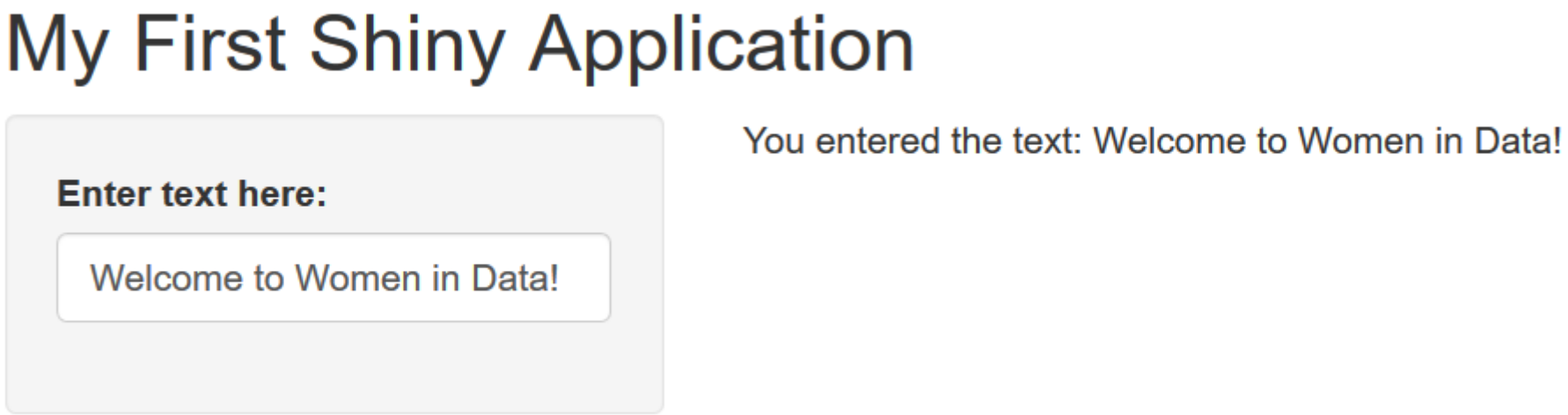
- Defines the components of the user interface
  - Page titles
  - Input options
  - Outputs

- Defines what the user will see and interact with

The Server Script

- Contains the information to build the app
- Contains a function with parameters input and output
- Defines what happens in R

Worked Example 1



Worked Example 1 - UI

```
library(shiny)

fluidPage(
  titlePanel("My First Shiny Application"),
  sidebarLayout(
    sidebarPanel(
        textInput(inputId = "TXT", label = "Enter text here:",
            value = "Welcome to Women in Data!")
        ),
    mainPanel(
            textOutput(outputId = "myTextOutput")
        )
  )
)
```

Worked Example 1 - Server

```
library(shiny)

function(input, output){

  output$myTextOutput <- renderText(
      paste("You entered the text:", input$TXT))

}
```

Layouts

- Example 1 used a sidebarLayout
- There are a number of possible layouts
- In this workshop we will only use the sidebarLayout

Sidebar Panel

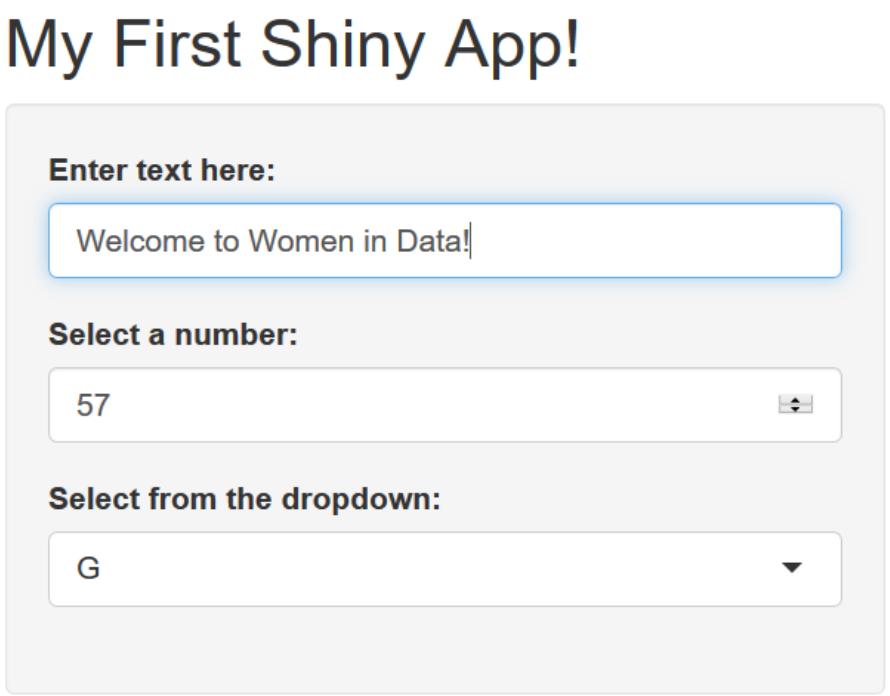
- Define the contents of the sidebar using the `sidebarPanel` function

- Accepts `*Input` functions that specify the app inputs

Input Controls

Input	Description
textInput	Text string input
numericInput	Numeric value input
selectInput	Select single or multiple values from drop down list
sliderInput	Numeric range “slider” input
radioButtons	Set of radio button inputs
fileInput	File upload control

Worked Example 2



Worked Example 2 - UI

```
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

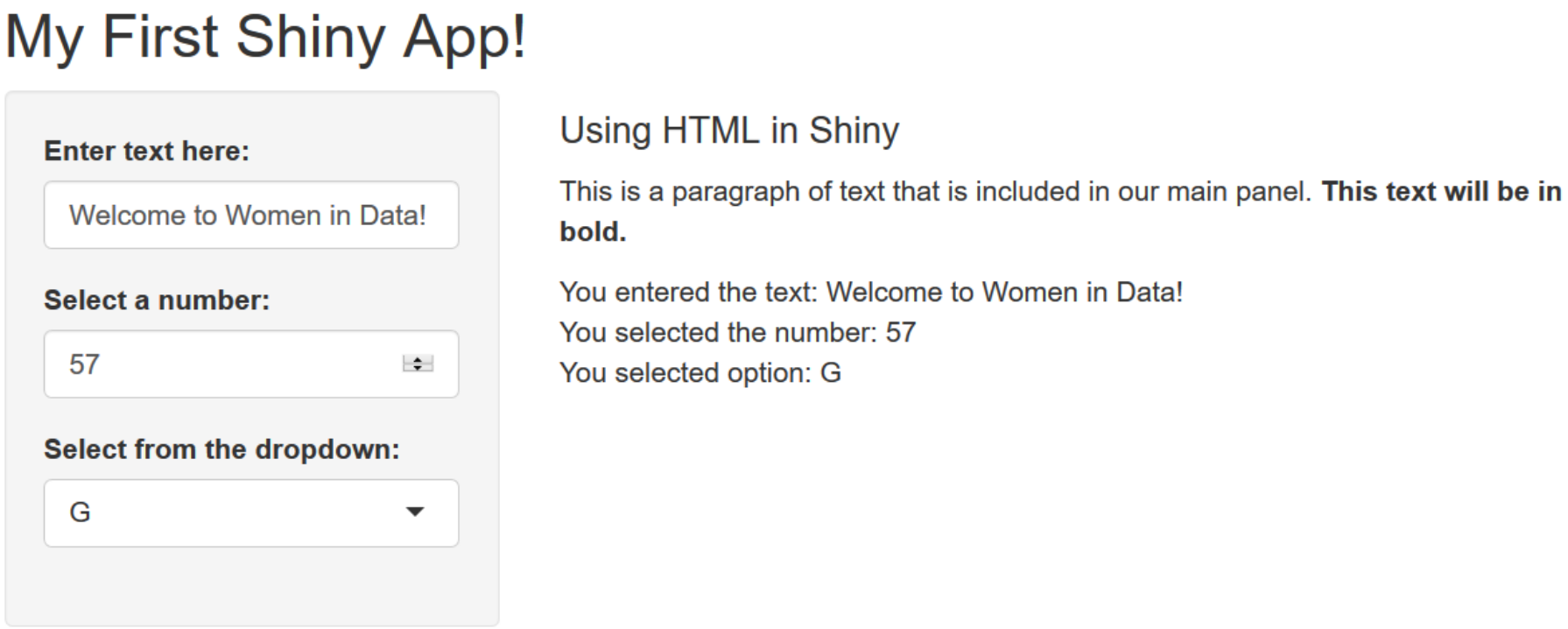
- Define the contents of the main panel using the function `mainPanel` function
- Can contain outputs using the `*Output` functions
- Can include HTML using a series of functions that replicate the HTML tags

HTML Formatting

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

Function	Usage
p	A paragraph of text
h*	A level * (1, 2, 3,...) header
code	A block of code
img	An image
strong	Bold text
em	Italic text

Worked Example 2



Worked Example 2 - UI

```
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("textOutput"),
        textOutput("numberOutput"),
        textOutput("selectOutput")
    )
```

Exercise 1

Build a simple Shiny application that takes a date string input (e.g. “30-11-2017”) 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

Hint: try using the dateInput and format functions

Exercise 1 - UI

```
library(shiny)

fluidPage(

  # Define the header for the page
  titlePanel("Exercise 1"),

  # Set up the page to have a sidebar
  sidebarLayout(
    # Define the contents of the sidebar
    sidebarPanel(
      dateInput("dateInput", "Select date")
    ),

    # Define the contents of the main panel
    mainPanel(
      textOutput("dateOutput")
    )
  )
)
```

Exercise 1 - Sever

```
library(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

Output Type	server.R Function	ui.R Function
Text	renderPrint	textOutput
Data	renderDataTable	dataTableOutput
Plot	renderPlot	plotOutput
Image	renderImage	imageOutput

Worked Example 3 - Render Data

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

- Display the data in a dataTable

Worked Example 3 - Render Data

Render Data in a Shiny App

Select from the dropdown:

airquality

Show 25 entries

Search:

Ozone	Solar.R	Wind	Temp	Month	Day
41	190	7.4	67	5	1
36	118	8	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
		14.3	56	5	5
28		14.9	66	5	6
23	299	8.6	65	5	7
19	99	13.8	59	5	8
8	19	20.1	61	5	9
	194	8.6	69	5	10
7		6.9	74	5	11
16	256	9.7	69	5	12
11	290	9.2	66	5	13

Worked Example 3 - UI

```
sidebarLayout(
  sidebarPanel(
      selectInput(inputId = "selectInput",
                  label = "Select from the dropdown:",
          choices = c("airquality", "iris",   "mtcars"))
    ),
    mainPanel(
      dataTableOutput(outputId = "dataOutput")
    )
  )
```

Worked Example 3 - Server

```
output$dataOutput <-
  renderDataTable(switch(input$selectInput,
   "airquality" = airquality,
   "iris" = iris,
   "mtcars" = mtcars)
  )
```

Worked Example 4 - Render Plots

- Select a column of the data from a drop down menu

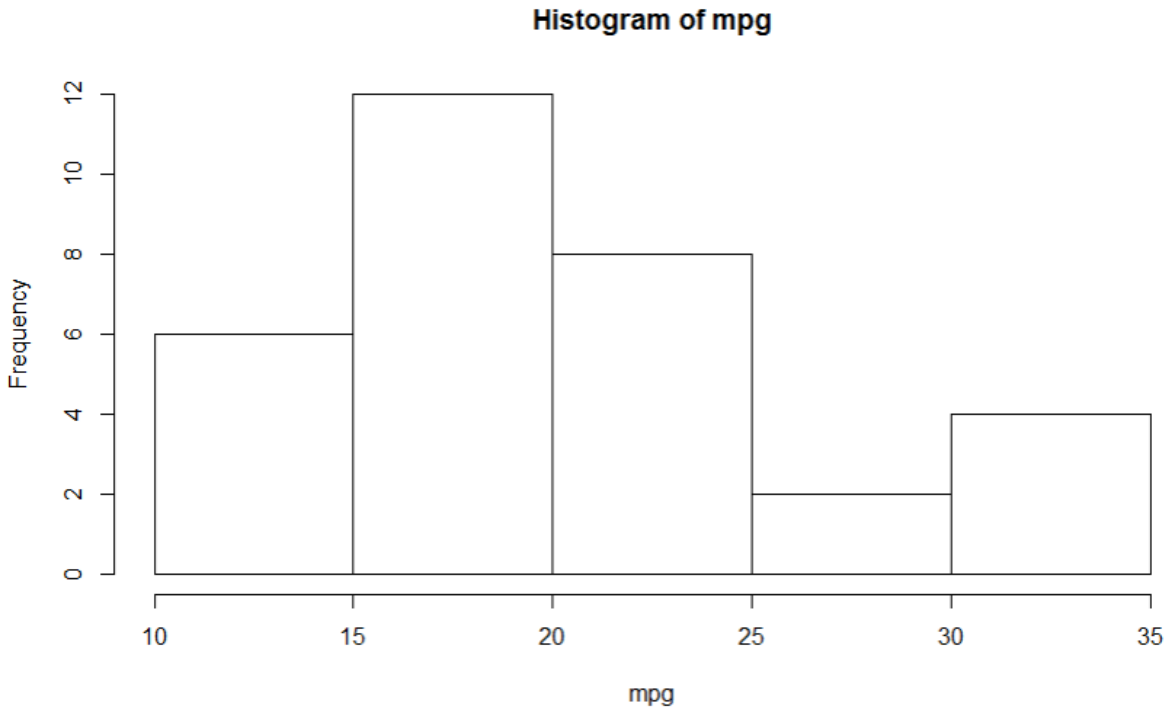
- Plot a histogram of the data

Worked Example 4 - Render Plots

Render Plot in a Shiny App

Select column:

mpg



Worked Example 4 - UI

```
sidebarLayout(
  sidebarPanel(
      selectInput("selectInput", "Select
column:", choices = colnames(mtcars))
    ),
  mainPanel(
      plotOutput("plotOutput")
    )
  )
```

Worked Example 4 - Server

```
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

```
library(shiny)

fluidPage(

  # Define the header for the page
  titlePanel("Render Plot in a Shiny App"),

  # Set up the page to have a sidebar
  sidebarLayout(
    # Define the contents of the sidebar
    sidebarPanel(
      numericInput("numberInput", "Select size of data:", min = 0, max = 500, value = 100),
      selectInput("colInput", "Select a colour", choices = c("red", "yellow", "blue",
"green"))
    ),

    # Define the contents of the main panel
    mainPanel(
      plotOutput("plotOutput")
    )
  )

)
```

Exercise 2 - Server

```
library(shiny)

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

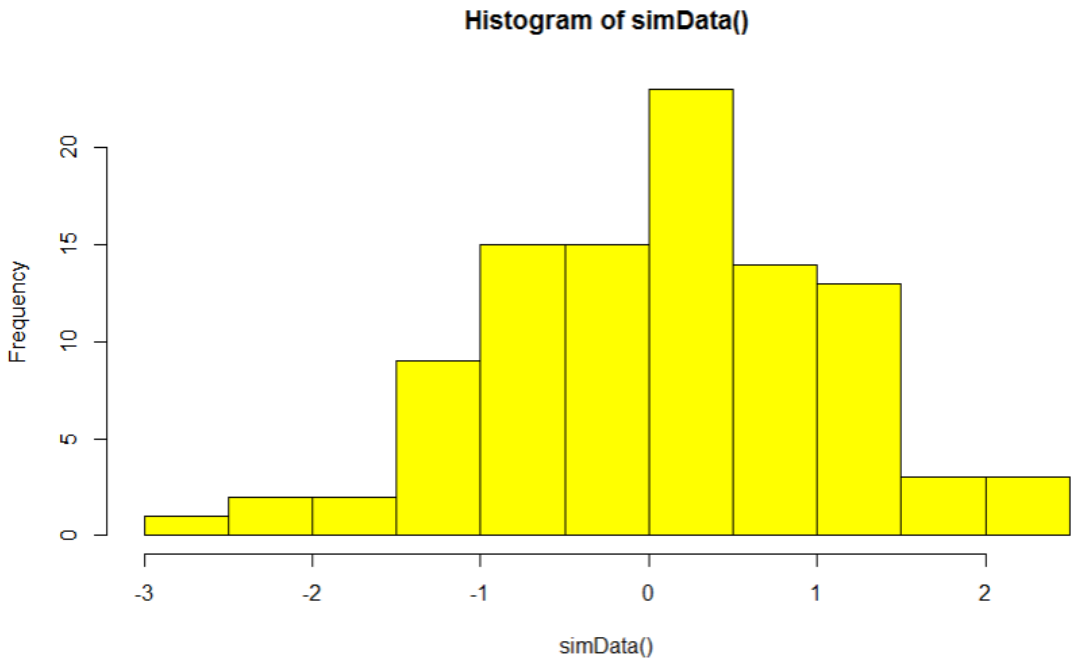
Select size of data:

100

Select a colour

yellow

- red
- yellow
- blue
- green

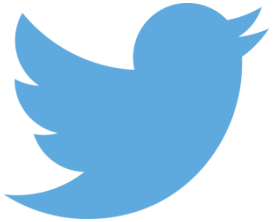


Worked Example 5 - Server

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


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

Follow Mango!



@mangothecat
@earlconf

MangoTheCat