MANGO SOLUTIONS

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

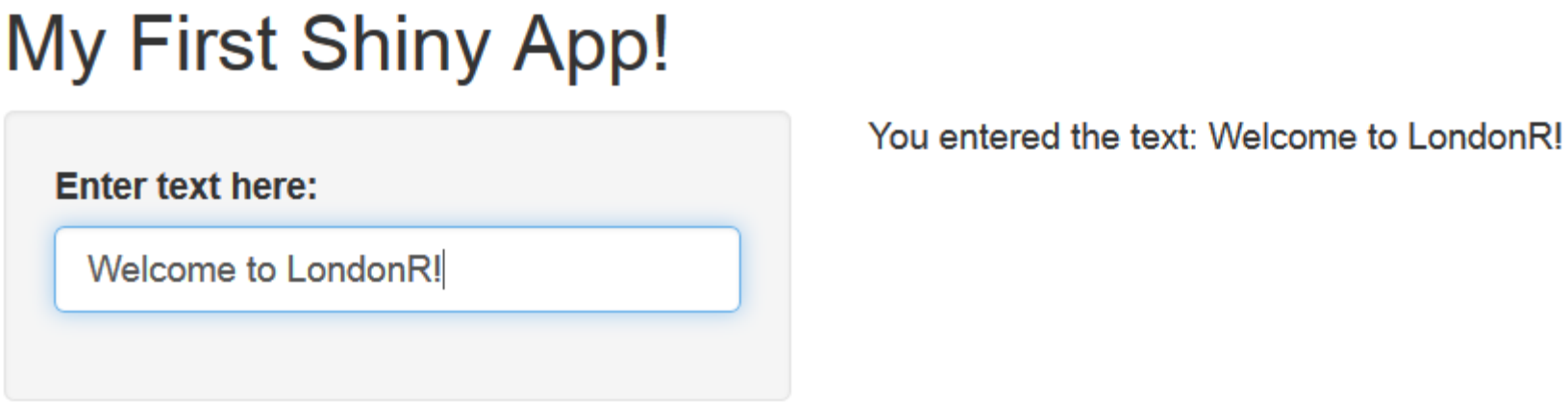
The User Interface Script

- 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



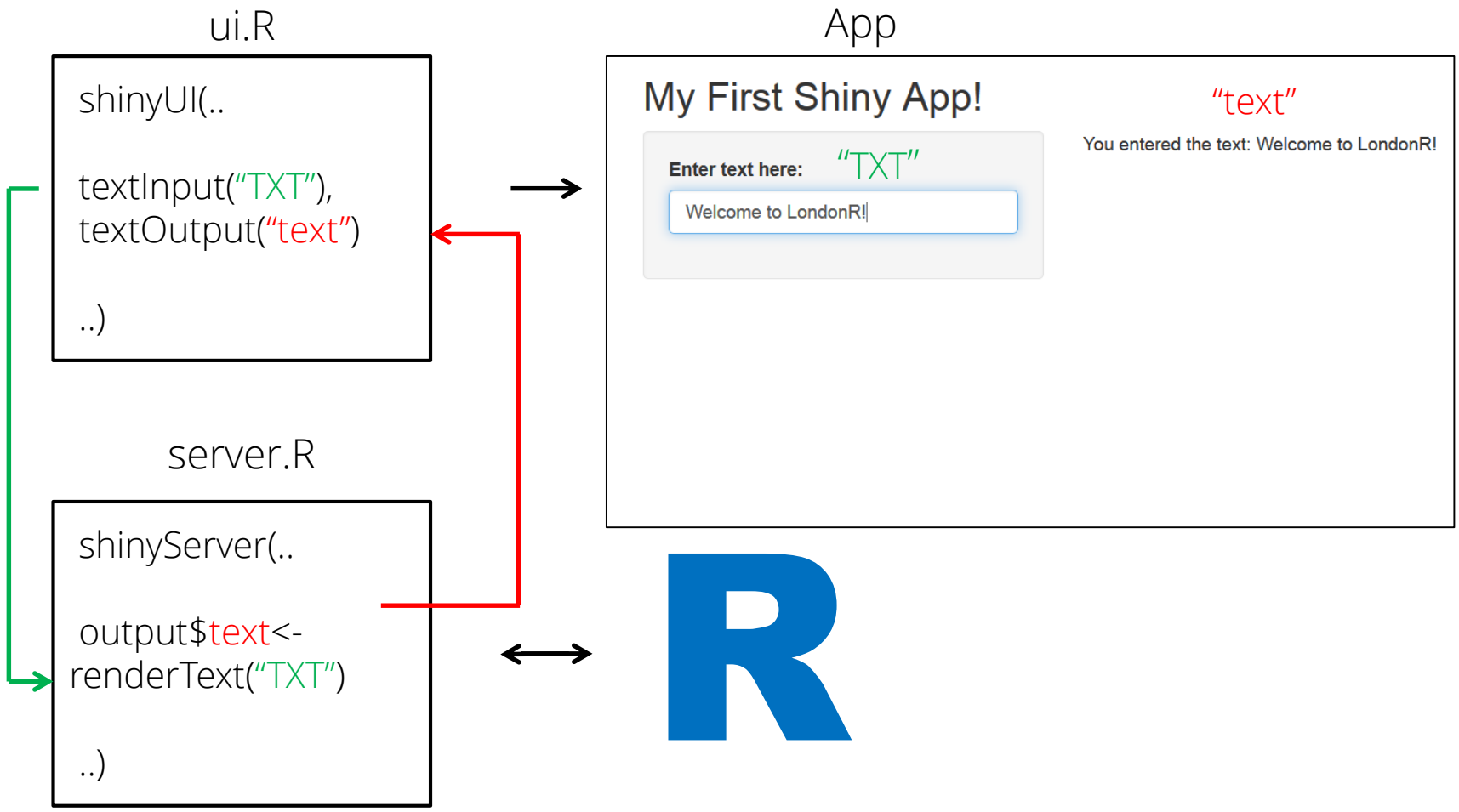
Worked Example 1 - UI

```
fluidPage(
  titlePanel("My First Shiny App!"),
  sidebarLayout(
    sidebarPanel(
      textInput("myText", "Enter text here:")
    ),
    mainPanel(
      textOutput("niceTextOutput")
    )
  )
)
```

Worked Example 1 - Server

```
function(input, output){

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

}
```

Schematic of a Basic Shiny app



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

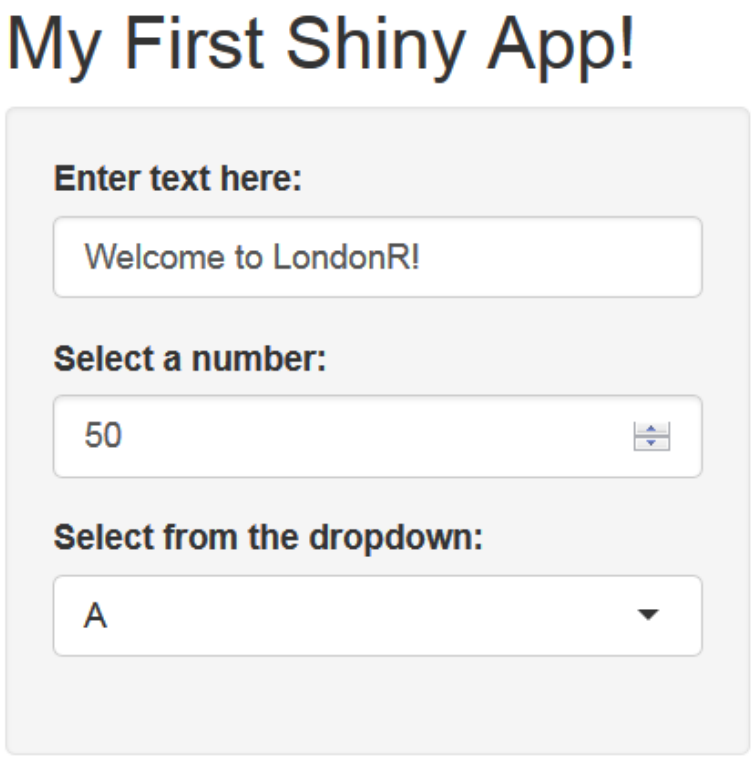
Shiny Workshop UI Boiler Plate

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

Input Controls

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

Worked Example 2



Worked Example 2 - UI

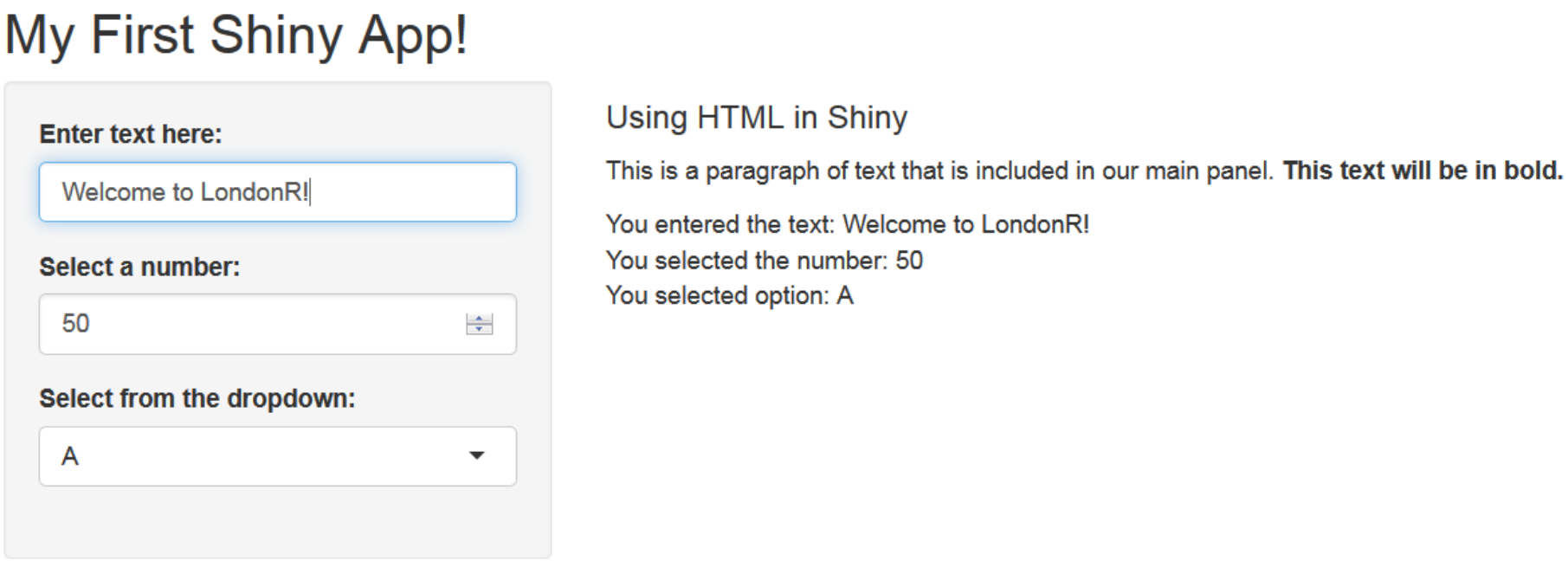
```
sidebarPanel(
  textInput("myTextInput", "Enter text here:"),

  numericInput("myNumberInput", "Select a number:",
               value = 50, min = 0, max = 100, step = 1),

  selectInput("mySelectInput", "Select from the dropdown:",
              choices = LETTERS[1:10])
)
```

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("niceTextOutput"),

  textOutput("niceNumberOutput"),

  textOutput("niceSelectOutput")
)
```

Worked Example 2 - Server

```
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

```
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 - Server

```
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

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
14	274	10.9	68	5	14

Worked Example 3 - UI

```
sidebarLayout(
  sidebarPanel(
    selectInput("selectInput", "Select from the dropdown:",
                choices = c("airquality", "iris", "mtcars"))
  ),
  mainPanel(
     dataTableOutput("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 mtcars data from a drop down menu

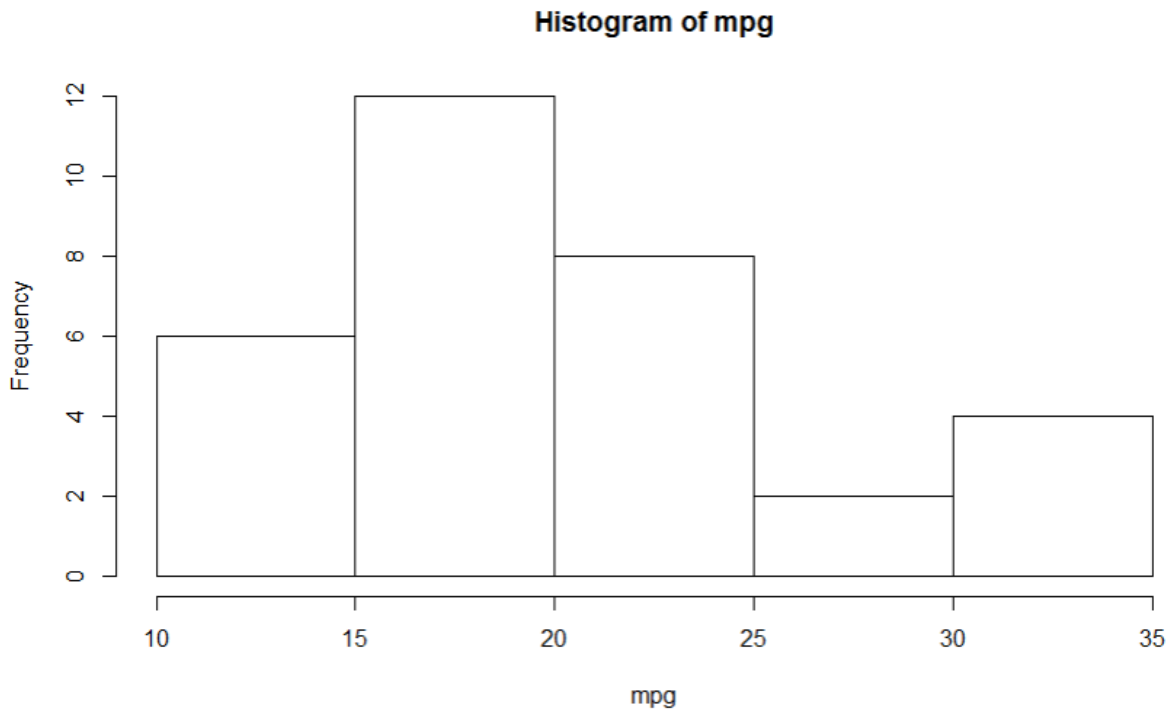
- Plot a histogram of the data

Worked Example 4 - Render Plots

Render Plot in a Shiny App

Select column:

mpg



Worked Example 4 - UI

```
fluidPage(
  titlePanel("My First Shiny App!"),
  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

```
fluidPage(
  titlePanel("Exercise 2 - Render Plot in a Shiny App"),
  sidebarLayout(
    sidebarPanel(
      numericInput("numberInput", "Select size of data:",
                   min = 1, max = 500, value = 100),
      selectInput("colInput", "Select a colour:",
                  choices = c("red", "yellow", "blue“, "green"))
    ),
      mainPanel(
        plotOutput("plotOutput")
      )
   )
)
```

Exercise 2 - Server

```
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

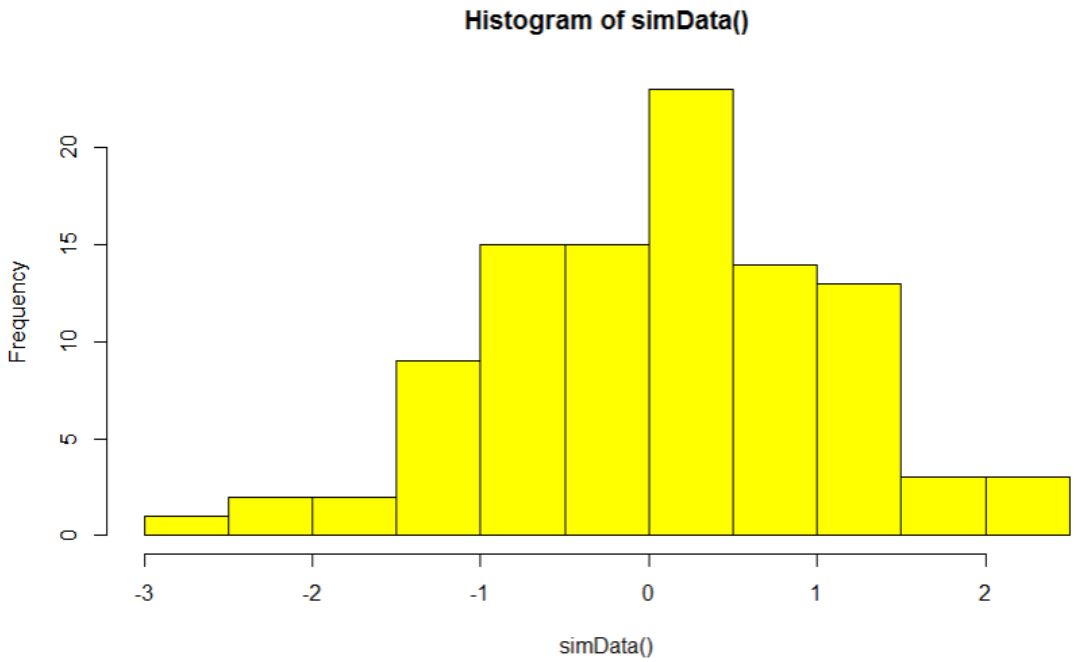
Select size of data:

100

Select a colour

yellow

- red
- yellow
- blue
- green



Worked Example 5 - Server

```
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

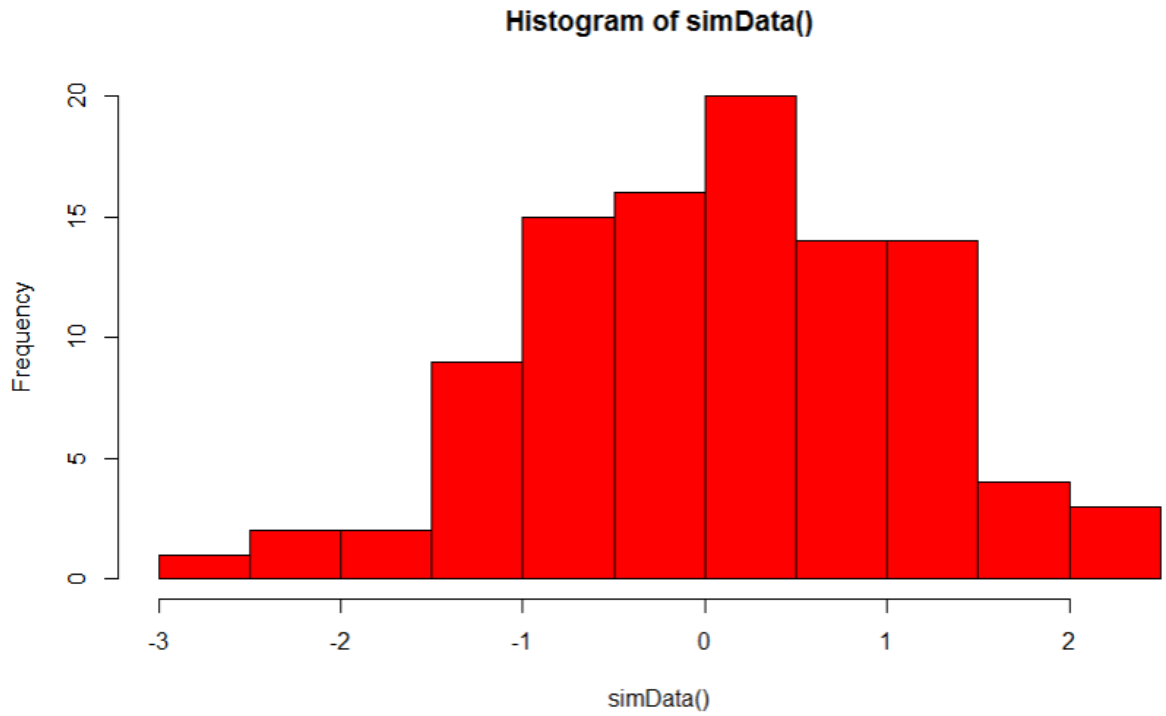
Render Plot in a Shiny App

Select size of data:

100

Select a colour

red



Worked Example 6 - UI

```
require(shinythemes)

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

Shiny Dashboard

- Package developed by RStudio for producing Dashboards with Shiny

- Available on github + CRAN

What Next?

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

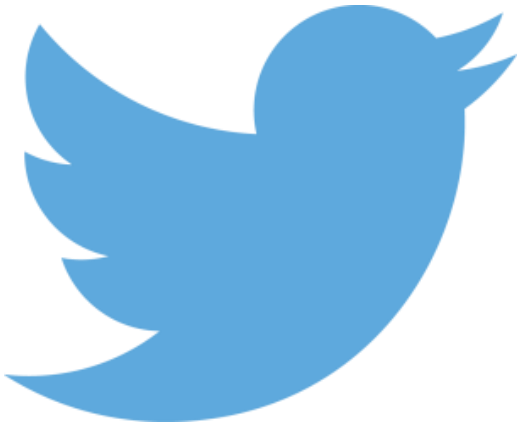
LondonR tonight

- **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