Week-8: Code-Along, Challenge

Elise Wong

2023-10-09

Week-8

Slides 6 to 10 - Example App

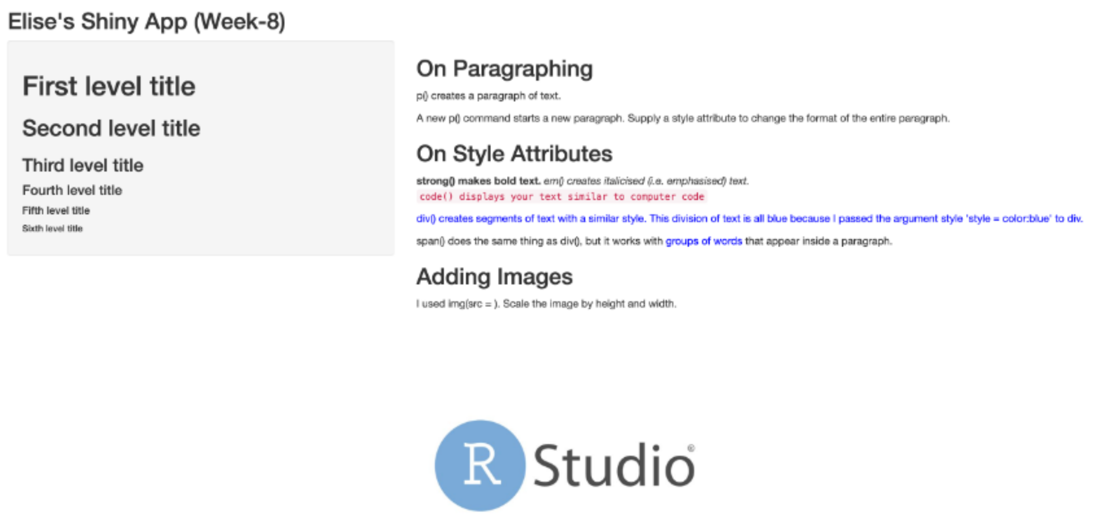
```
# Example, Slide 6
library(shiny)
# runExample("01_hello")

# User Interface, Slide 9
library(shiny)
# Define UI for app that draws a histogram
ui <- fluidPage(
  titlePanel("Hello Shiny!"),
   sidebarLayout(
   sidebarPanel(
   sliderInput(inputId = "bins",
               label = "Number of bins:",
               min = 1,
               max = 50,
                value = 30)),
  mainPanel(

plotOutput(outputId = "distPlot"))))

# Server Function, Slide 10
server <- function(input, output) {
  output$distPlot <- renderPlot({
    x <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = "#007bc2", border = "white",
        xlab = "Waiting time to next eruption (in mins)",
        main = "Histogram of waiting times")
    })
}
```

Slides 11 to End - App-1

```
# runApp("App-1", display.mode = "showcase")
```

Image of App

```r
# Initial image,
library(knitr)
knitr::include_graphics("app_page.png")
```



Elise's Shiny App (Week-8)

First level title

Second level title

Third level title

Fourth level title

Fifth level title

Sixth level title

On Paragraphing

p() creates a paragraph of text.

A new p() command starts a new paragraph. Supply a style attribute to change the format of the entire paragraph.

On Style Attributes

strong() makes bold text. *em() creates italicised (i.e. emphasised) text.*
`code() displays your text similar to computer code`

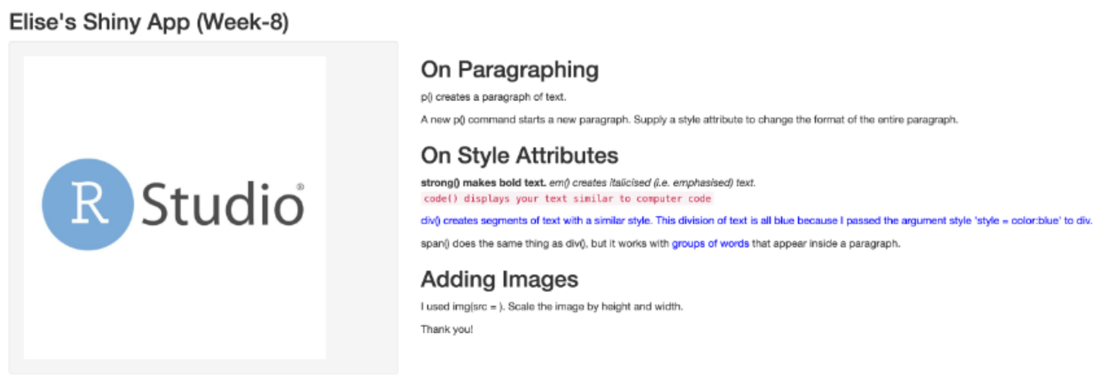
div() creates segments of text with a similar style. This division of text is all blue because I passed the argument style 'style = color:blue' to div.

span() does the same thing as div(), but it works with groups of words that appear inside a paragraph.

Adding Images

I used img(src =). Scale the image by height and width.

```r
# Image in side panel,
knitr::include_graphics("app_page2.png")
```



Elise's Shiny App (Week-8)

On Paragraphing

p() creates a paragraph of text.

A new p() command starts a new paragraph. Supply a style attribute to change the format of the entire paragraph.

On Style Attributes

strong() makes bold text. *em() creates italicised (i.e. emphasised) text.*
`code() displays your text similar to computer code`

div() creates segments of text with a similar style. This division of text is all blue because I passed the argument style 'style = color:blue' to div.

span() does the same thing as div(), but it works with groups of words that appear inside a paragraph.

Adding Images

I used img(src =). Scale the image by height and width.

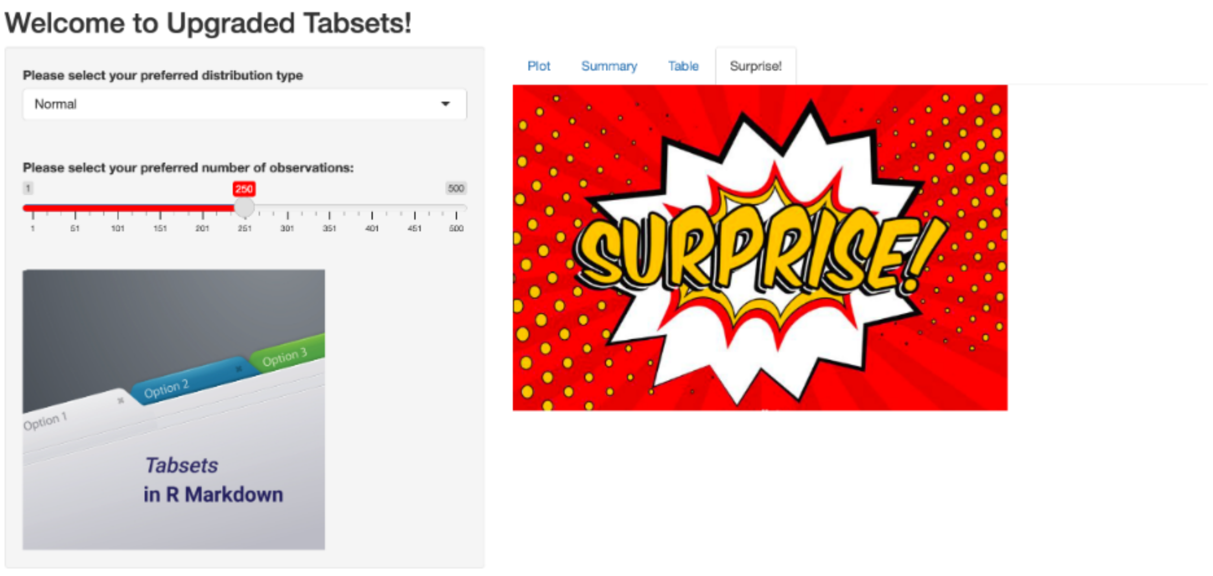
Thank you!

Challenge-8

```r
library(shiny)
# runExample("06_tabsets")
```

```r
# Run App-2,
library(shiny)
# runApp("App-2", display.mode = "showcase")
# Explaining changes from Tabsets Example 6 into App-2,

library(shiny)
library(dplyr)

ui <- fluidPage(

# I first updated the title.
  titlePanel("Welcome to Upgraded Tabsets!"),

  sidebarLayout(
    sidebarPanel(

# Here, I changed the input option from Radio Buttons to a Drop-down format.
# I also changed the prompt text.
      selectInput("dist", "Please select your preferred distribution type",
                  c("Normal" = "norm",
                    "Uniform" = "unif",
                    "Log-Normal" = "lnorm",
                    "Exponential" = "exp")),

      br(),

# I changed the color of the slider here, as well as the
# prompt text and numbers (halved).
      tags$style(HTML(".js-irs-0 .irs-single, .js-irs-0 .irs-bar-edge, .js-irs-0
                       .irs-bar {background: red}")),
      sliderInput("n",
                  "Please select your preferred number of observations:",
                  value = 250,
                  min = 1,
                  max = 500),

# Here, I added an additional photo for 'Tabsets' in the side bar,
# and scaled it as appropriate.
      br(),
      img(src = "tabsetsphoto.jpeg", height = 300)
    ),


    mainPanel(

# I added an additional tab panel here named 'Surprise!', and included a photo
# under the panel.
      tabsetPanel(type = "tabs",
                  tabPanel("Plot", plotOutput("plot")),
                  tabPanel("Summary", verbatimTextOutput("summary")),
                  tabPanel("Table", tableOutput("table")),
                  tabPanel("Surprise!", img(src = "surprise1.png", width = "50%")
                  )
      )
    )
  )
)

  server <- function(input, output) {

    d <- reactive({
      dist <- switch(input$dist,
                     norm = rnorm,
                     unif = runif,
                     lnorm = rlnorm,
                     exp = rexp,
                     rnorm)

# To obtain an output later for the table section in descending order,
# I had to ensure the output of the values were in a data frame.
      data.frame(Value = dist(input$n))
    })

# I had issues with ensuring an output for the plot due to the above data.frame
# code, so I had to employ coercion to ensure that 'x' was a numeric value. It
# includes a conditional check to determine whether the data is numeric before
# generating the histogram plot, as error messages I had received indicated that
# 'x' was not, in fact, a numeric value.
    output$plot <- renderPlot({
      data_df <- d()
      if(is.numeric(data_df$Value)) {
        dist <- input$dist
        n <- input$n

        hist(data_df$Value,
             main = paste("r", dist, "(", n, ")", sep = ""),
             col = "darkgrey", border = "white")
      }
    })

    output$summary <- renderPrint({
      summary(d())
    })

# Lastly, I changed the table to output the values in descending order.
    output$table <- renderTable({
      sorted_data <- arrange(d(), desc(Value))
      sorted_data
    })

  }

  shinyApp(ui, server)
```

```r
knitr::include_graphics("tabsetpage_1.png")
```



Welcome to Upgraded Tabsets!

Please select your preferred distribution type

Normal

Please select your preferred number of observations:

1 250 500

1 51 101 151 201 251 301 351 401 451 500

Plot Summary Table Surprise!