Shiny II - Excercises

Exercise1

Draw the reactive graph for the following server functions:

```
server1 <- function(input, output, session) {
  c <- reactive(input$a + input$b)
  e <- reactive(c() + input$d)
  output$f <- renderText(e())
}

server2 <- function(input, output, session) {
  x <- reactive(input$x1 + input$x2 + input$x3)
  y <- reactive(input$y1 + input$y2)
  output$z <- renderText(x() / y())
}

server3 <- function(input, output, session) {
  d <- reactive(c() ^ input$d)
  a <- reactive(input$a * 10)
  c <- reactive(b() / input$c)
  b <- reactive(a() + input$b)
}
```

Exercise 2

Why will this code fail?

```
var <- reactive(df[input$var])
range <- reactive(range(var(), na.rm = TRUE))
```

Why is `var()` a bad name for a reactive?

Exercise 3

Update the options for renderDataTable() below so that the table is displayed, but nothing else (i.e. remove the search, ordering, and filtering commands). You'll need to read ?renderDataTable and review the options at https://datatables.net/reference/option/.

```
ui <- fluidPage(
  dataTableOutput("table")
)

server <- function(input, output, session) {
  output$table <- renderDataTable(mtcars, options = list(pageLength = 5))
}
```

Exercise 4

Modify the Central Limit Theorem app below so that the sidebar is on the right instead of the left.

```
ui <- fluidPage(
  titlePanel("Central limit theorem"),
  sidebarLayout(
    sidebarPanel(
      numericInput("m", "Number of samples:", 2, min = 1, max = 100)
    ),
    mainPanel(
      plotOutput("hist")
    )
  )
)
server <- function(input, output, session) {
  output$hist <- renderPlot({
    means <- replicate(1e4, mean(runif(input$m)))
    hist(means, breaks = 20)
  }, res = 96)
}
```

Exercise 5

Browse the themes available in the shinythemes package, pick an attractive theme, and apply it to the Central Limit Theorem app.

Exercise 6

Create an app that contains two plots, each of which takes up half the app (regardless of what size the whole app is)

Exercise 7

Make a plot with click handle that shows all the data returned in the input.

Exercise 8

Make a plot with click, dblclick, hover, and brush output handlers and nicely display the current selection in the sidebar. Plot the plot in the main panel.

Exercise 9

Compute the limits of the distance scale using the size of the plot.

```
output_size <- function(id) {
  reactive(c(
    session$clientData[[paste0("output_", id, "_width")]],
    session$clientData[[paste0("output_", id, "_height")]]
  ))
}
```

Exercise 10

Use the ambient package by Thomas Lin Pedersen to generate worley noise and download a PNG of it.

Exercise 11

Create an app that lets you upload a csv file, select a variable, and then perform a `t.test()` on that variable. After the user has uploaded the csv file, you'll need to use `updateSelectInput()` to fill in the available variables. See Section 10.1 for details.

Exercise 12

Create an app that lets the user upload a csv file, select one variable, draw a histogram, and then download the histogram. For an additional challenge, allow the user to select from .png, .pdf, and .svg output formats.

Exercise 13

Write an app that allows the user to create a Lego mosaic from any .png file using Ryan Timpe's brickr package. Once you've completed the basics, add controls to allow the user to select the size of the mosaic (in bricks), and choose whether to use "universal" or "generic" colour palettes.