Creating web apps for interactive data visualization with R Shiny

A Tufts TIDAL workshop from the Data Intensive Studies Center (DISC)

Karin Knudson
karin.knudson@tufts.edu

https://karink520.github.io/intro-r-shiny-workshop/

Creating web apps for interactive data visualization with R Shiny

A Tufts TIDAL workshop from the Data Intensive Studies Center (DISC)

Karin Knudson
karin.knudson@tufts.edu

https://karink520.github.io/intro-r-shiny-workshop/

If you're here early, check out examples of Shiny apps at: https://shiny.rstudio.com/gallery/

“Shiny makes it **incredibly easy** to build **interactive web applications** with R. Automatic **"reactive" binding** between inputs and outputs and extensive **prebuilt widgets** make it possible to build **beautiful**, **responsive**, and **powerful applications with minimal effort**.”

— the Shiny documentation

TO DO:

1. Open the workshop page and keep the materials handy throughout the session:

   https://karink520.github.io/intro-r-shiny-workshop/

2. Open RStudio and install Shiny using the command:

```
install.packages("shiny")
```

STRUCTURE OF A SHINY APP

```
library(shiny)

ui <- ... # Appearance and layout of app

server <- ... # Code the server needs to run,
          e.g. draw plots and process data

shinyApp(ui = ui, server = server)
```

A BASIC TEMPLATE

```
ui <- fluidPage(

)

server <- function(input, output) {

}

shinyApp(ui = ui, server = server)
```

LAYOUT AND PANEL FUNCTIONS TO STRUCTURE

- Content as arguments, separated by commas
- Start with `fluidPage()`, `navbarPage()`, or `fixedPage()`
- See `fluidRow()/fluidPage()` and `column()` docs for more flexibility laying out page on a 12-column grid
- Bootstrap 3 framework

CREATE HTML WITH SHINY

Many html tags have their own Shiny helper function, including

`a, div, h1…h6, img, p, span`

HTML tags can also be created with `tags`, e.g.:

`tags$a, tags$code`

Some HTML tags have names that would conflict with native R functions so must be accessed within `tags`, rather than with their name alone

BASIC REACTIVITY WITH *INPUT, *OUTPUT, RENDER*

Add inputs (e.g. premade widgets) and outputs to `ui`, each with an id

Inputs and outputs are passed to the server function in a list, accessed by their ids

Connect inputs and outputs with `render*` in the server function

When an input used within `render*` changes, code block re-runs.

WIDGET EXAMPLES

Buttons

Action

Submit

Single checkbox

Choice A

Checkbox group

Choice 1

Choice 2

Choice 3

Date input

2014-01-01

Date range

2017-06-21 to 2017-06-21

File input

Browse... No file selected

Help text

Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Numeric input

1

Radio buttons

Choice 1

Choice 2

Choice 3

Select box

Choice 1

Sliders

0 50 100

0 10 20 30 40 50 60 70 80 90 100

0 25 75 100

0 10 20 30 40 50 60 70 80 90 100

Text input

Enter text...

image from: https://shiny.rstudio.com/tutorial/written-tutorial/lesson3/

YOU TRY (EXERCISES)

QUESTIONS, RECAP, PAUSE

NEXT: A DEEPER DIVE INTO REACTIVITY

Including additional functions for more control over reactive behavior

FOR FUN, LET'S ADD A THEME

```
install.packages("shinythemes")
```

Add a theme argument to fluidPage, e.g:

```
ui = fluidPage(theme = shinytheme("darkly"),
               …
```

More about themes: https://rstudio.github.io/shinythemes/