

SHINY

Jeff Goldsmith, PhD
Department of Biostatistics

What is Shiny?

- Framework for building interactive plots and web applications in R
- Shiny allows you to create a graphical user interface (GUI)
 - Users can interact with your code without knowing R!
 - Communicate visualizations, models, algorithms to collaborators
- Uses HTML, CSS, and JavaScript framework
 - You don't need to know these to use Shiny
 - The syntax can be tricky at first, though
 - Knowing more can help you get fancy
- Brought to you by R Studio in 2012



What is Shiny?

- Package for creating web-apps
- Don't need to learn how to code apps directly; you write R code and shiny creates then app
 - Analogous to creating HTML files by writing R Markdown and knitting
- Adds interactivity – your app can take user input and update outputs accordingly
- For a quick example, run `shiny::runExample("01_hello")` in your R console

What is Shiny?

- Pack
- Don't creat
- A
- Add
- acco
- For a

The screenshot shows a Shiny web application interface. At the top, the URL is `http://127.0.0.1:4343` and there are buttons for "Open in Browser" and "Publish". The main heading is "Hello Shiny!". Below the heading is a slider control labeled "Number of bins:" with a range from 1 to 50 and a current value of 30. To the right of the slider is a histogram titled "Histogram of waiting times". The x-axis is labeled "Waiting time to next eruption (in mins)" and ranges from approximately 45 to 95. The y-axis is labeled "Frequency" and ranges from 0 to 25. The histogram shows a distribution of waiting times with a peak frequency of about 26.

Waiting time (mins)	Frequency
45	1
47	8
49	7
51	10
53	6
55	12
57	15
59	7
61	4
63	13
65	4
67	7
69	3
71	3
73	9
75	8
77	6
79	17
81	26
83	18
85	13
87	26
89	16
91	8
93	6
95	9
97	2
99	3
101	1

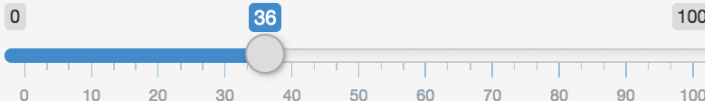
How does Shiny work?

- Shiny applications have two components:
 - A user interface to obtain inputs
 - Code that reacts to inputs and produces outputs
- R code executes in the background
- Because you need R to use Shiny, sharing Shiny-based products requires some thought
 - Not as “easy” as sending / hosting HTML files produced only by R Markdown

Getting inputs

- Widgets are text elements that users can interact with
 - Examples include scroll bars, buttons, text, ect
 - Take in user input

Slider



0 36 100

0 10 20 30 40 50 60 70 80 90 100

Current Value:

```
[1] 36
```

[See Code](#)

Radio buttons

Choice 1
 Choice 2
 Choice 3

Current Values:

```
[1] "1"
```

[See Code](#)

Text input

Current Value:

```
[1] "Hey there DSI.."
```

[See Code](#)

Producing outputs

- These are functions that react to user input from widgets
 - `renderPrint()` -- prints output of a function
 - `renderText()` -- outputs text
 - `renderTable()` -- for making tables
 - `renderPlot()` -- outputs plot made using ggplot2 (and base R, ...)
 - `renderPlotly()` -- outputs plot made with plotly library

Flexdashboard + Shiny

- R-Markdown-based **Shiny document**
- Relatively easy to use (given an understanding of dashboards / markdown)
- Adds dynamic elements to a flexdashboard
 - Input / output elements are added directly to the R Markdown file

Shiny applications

- Standalone web-app framework
- Not built within an R Markdown document
 - Separate .R files control UI and “server” computations for input / output
 - Alternatively, UI and server objects included in a single app file
- Potentially more flexible than piggybacking on R Markdown / flexdashboard

Shiny applications

- ui
 - Controls layout and appearance
 - Where you add widgets

 - ui.R
- server
 - Instructions your computer needs to build the app
 - R code for plots, etc

 - server.R

Sharing shiny products

- Not always easy – Shiny requires R to run in the background
- Providing files
 - Send “raw” files (.rmd, .R, data, etc), maybe as an R project
 - Recipient knits the file / runs the app through Rstudio
- Hosting online
 - Needs a server that runs R in the background, and github doesn't
 - shinyapps.io is pretty common way to permanently host document / app