

Advanced Topic Presentation

Shiny

An Application Investigation

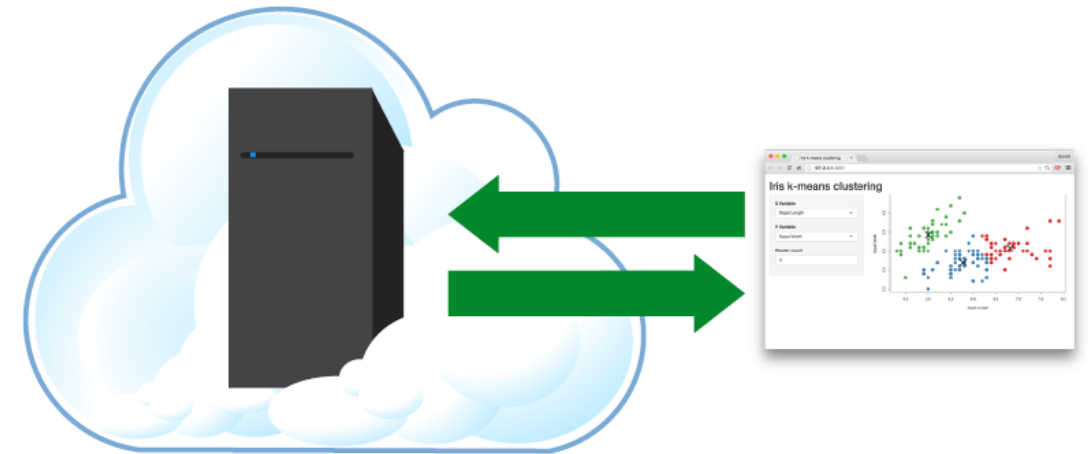
By: Thomas Bahng

ABOUT SHINY

Rapid Prototyping Web Applications

- Shiny is an open-source R package for building interactive web applications developed by RStudio.
- Shiny apps are created using the R programming language, library functions, and extensible using JS / CSS / HTML.
- The RStudio IDE provides everything needed to build and deploy the app.
- Web app development is hard; Shiny apps are easy to write.

Every Shiny app is maintained by a computer running R



HISTORY

2012 - Present

- First announced to beta testers during the JSM (Joint Statistical Meetings) 2012 conference by Joe Cheng @ RStudio.
- Since then there have been 5,045 commits, 72 branches, 58 releases, and 45 contributors to the Shiny package on GitHub.



Functionality

- Every app consists of two components, a user-interface and a server function.
- These two components are built using functions from the library and the app is run with a function call.
- The UI defines the “look-and-feel” of the app through page layout, display panels, and input controls.
- The server contains instructions needed to build the app. It renders the output with inputs from the UI.
- Most, if not all, of the R code that generates the content is written in the server function.

WHAT'S IN AN APP?

```
library(shiny)
ui <- fluidPage()
```

User interface
controls the layout and
appearance of app

```
server <- function(input, output) {}
```

Server function
contains instructions
needed to build app

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

INPUTS

User Interaction with the App

- Shiny provides users the ability to input choices that trigger interactivity within the app.
- These include buttons, text boxes, drop-downs, and slider bars, all of which are called as functions within the UI component.
- User inputs determine the actions of the server.

The image shows a Shiny application window titled "INPUTS". The interface is divided into two main columns. The left column displays various input types with their corresponding R functions. The right column shows the visual representation of these inputs in a web browser.

Action	Function
Link	<code>actionLink(inputId, label, icon, ...)</code>
Choice 1 Choice 2 Choice 3	<code>checkboxGroupInput(inputId, label, choices, selected, inline)</code>
Check me	<code>checkboxInput(inputId, label, value)</code>
Calendar	<code>dateInput(inputId, label, value, min, max, format, startview, weekstart, language)</code>
Range Calendar	<code>dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)</code>
Choose File	<code>fileInput(inputId, label, multiple, accept)</code>

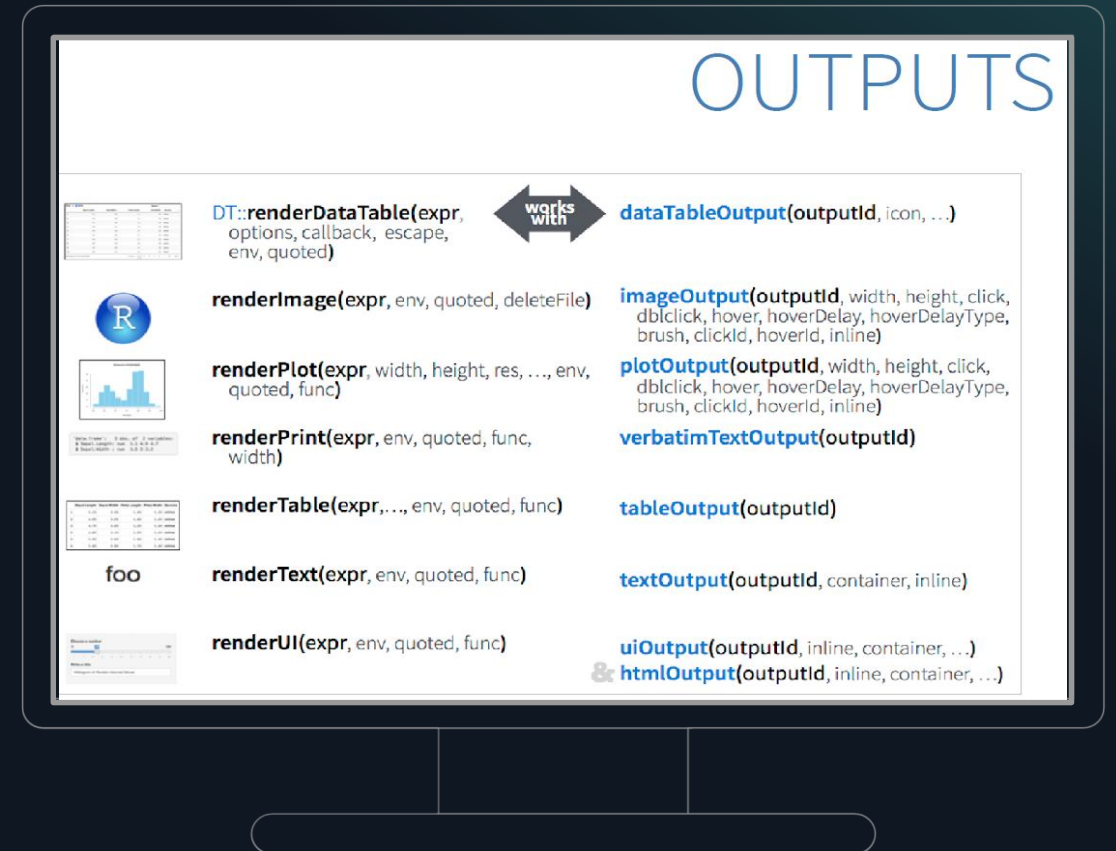
The right column shows the visual representation of these inputs:

- `numericInput(inputId, label, value, min, max, step)`: A numeric input field with the value 1.
- `passwordInput(inputId, label, value)`: A password input field with masked characters (dots).
- `radioButtons(inputId, label, choices, selected, inline)`: Three radio buttons labeled Choice A, Choice B, and Choice C. Choice A is selected.
- `selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also selectizeInput())`: A select dropdown menu with choices Choice 1 and Choice 2. Choice 1 is selected.
- `sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)`: A slider bar with a range from 0 to 10. The value is set to 5.
- `submitButton(text, icon)` (Prevents reactions across entire app): A blue button labeled "Apply Changes".
- `textInput(inputId, label, value)`: A text input field with the placeholder text "Enter text".

OUTPUTS

What you see on screen

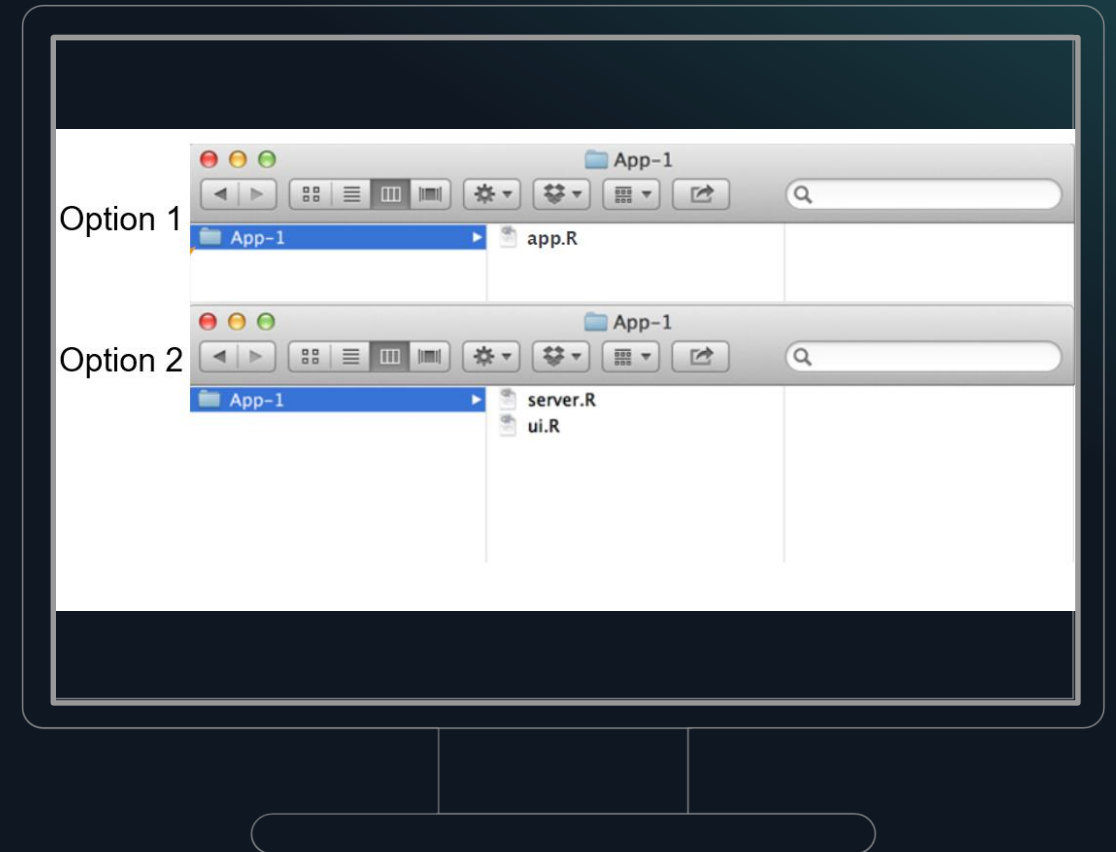
- The library contains “render” and “Output” functions respective to different types of output.
- These functions tag-team off each other where what is rendered in the server component is returned as output in the UI component.
- Output can be a plot, image, text, table, HTML, etc.



FILE STRUCTURE

Saving your Shiny app

- One directory with every file the app needs
- The app can be in a single file, "app.R" (your script which ends with a call to `shinyApp()`); *(recommended for deploying)*
- Or the app can be in multiple files, "ui.R" and "server.R"
- Other files in directory can include datasets, images, CSS, helper scripts, etc.





Learning Curve

From R Programmer to Shiny Developer

- Shiny is easy to pick up, provided a solid R programming background (i.e. functions, visualizations, etc.)
- Simple apps take minimal time to create and deploy.
- Shiny can get very deep into customizing reactions and appearance because it is a web application. (CSS, JS)
- A 2-hour tutorial from RStudio is available here for beginners: <https://shiny.rstudio.com/tutorial/>

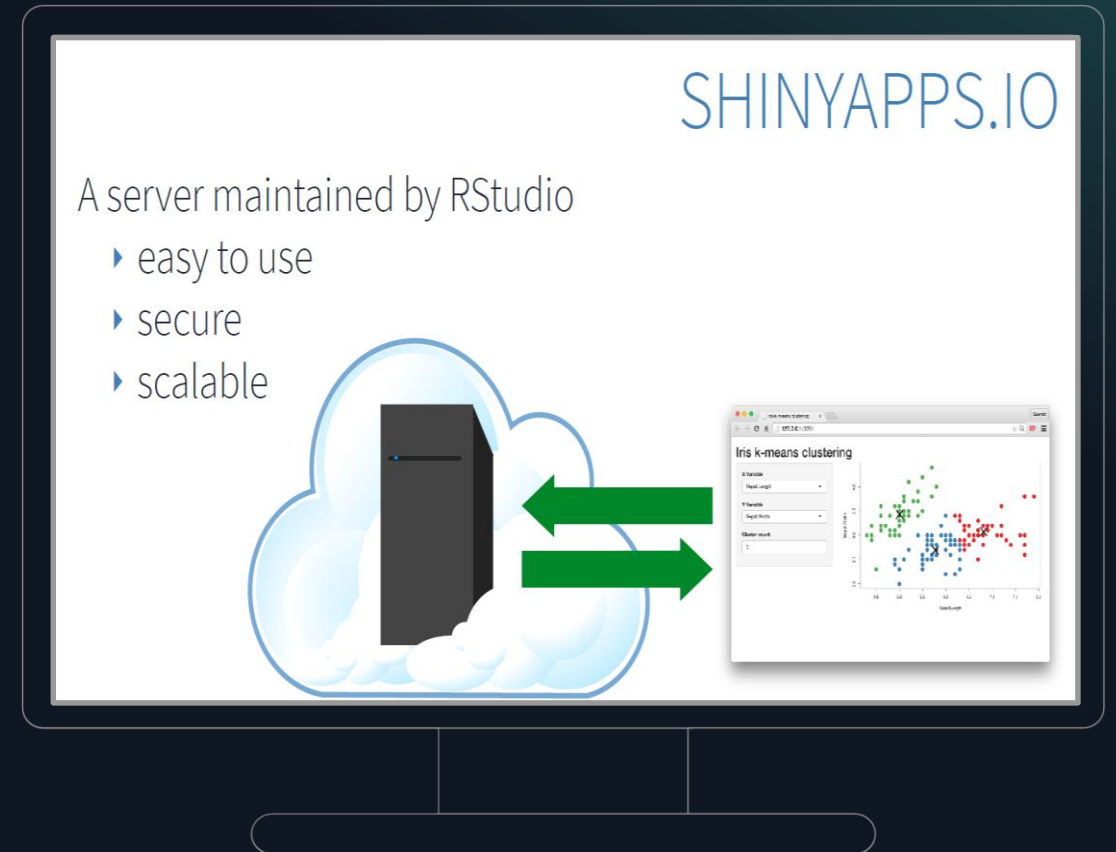
HOSTING APPS

Shinyapps.io

- <https://www.shinyapps.io>
- Requires the 'rsconnect' package

Demos

- <https://tbahng.shinyapps.io/salesreps/>
- <https://tbahng.shinyapps.io/gapminder/>
- <https://tbahng.shinyapps.io/cranlogs>



References

Shiny RStudio Homepage <https://shiny.rstudio.com/>

GitHub Repository of Example Shiny Apps <https://github.com/tbahng/IST719-AdvancedTopicShiny>

Cheat Sheets from RStudio <https://www.rstudio.com/resources/cheatsheets/>

GitHub Repository for Shiny <https://github.com/rstudio/shiny>

RStudio releases Shiny <https://www.r-bloggers.com/rstudio-releases-shiny/>

Shiny Tutorial <https://shiny.rstudio.com/tutorial/>

Shinyapps.io <https://www.shinyapps.io>

Thank You

By: Thomas Bahng