July 12, 2021

Randall Reese, PhD
INL Data Science

Rshiny and (Py)Dash

Creating interactive data platforms for the web

Randall Reese

- INL Data Scientist
- BS, MS, Mathematics. *Brigham Young University*
- PhD Statistics. *Utah State University*

Preliminaries and Assumptions

- Basic experience in R and Python.
- An understanding of basic web design terms.

Caveats

- The focus will be on functionality over beauty.
- I do not know every corner case of Shiny or Dash.
- I am on a Windows machine, but will try to stay as OS agnostic as possible.
- Everything here is free.

GitHub

- www.github.com/rr1964/Bootcamp
- All slides and code found there.
- It's sparse and as is.
- MIT License.

Installations (For R)

- R (Rshiny requires ≥ R v3.0.2)
  - https://cran.r-project.org/mirrors.html
  - Download for your OS
  - Base installation (Newest is 4.1.0 “Camp Pontanezen”)
  - My R version 4.0.4 (2021-02-15) -- "Lost Library Book"

- RStudio
  - https://www.rstudio.com/products/rstudio/download/#download

Installing RShiny

- In RStudio:
  - Tools -> Install Packages -> shiny (Repository CRAN)

- In RGui:

```
install.packages("shiny", lib="</data/Rpackages/>")
library(shiny, lib.loc="</data/Rpackages/>")

OR

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

Getting Started in RShiny

Everything we write will be “single file”

```
library(shiny)
ui <- ...
server <- ...
shinyApp(ui = ui, server = server)
```

All code must be in a file called app.R.
Keep different Shiny apps separate by using separate directories.

In RStudio, create a new project ->
New Directory ->
Shiny app to get a demo app.

Also can create by hand.

Create input components

```
sliderInput("probSlider", h5("Probability of heads"), min = 0, max = 1, value = 0.5)
```

- Input component type
- Internal name
- Display text
- Component properties

Define output in UI

Output function	Creates
dataTableOutput	DataTable
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	raw HTML
verbatimTextOutput	text

Rendering in Shiny

Define in the server function.

render function	creates
renderDataTable	DataTable
renderImage	images (saved as a link to a source file)
renderPlot	plots
renderPrint	any printed output
renderTable	data frame, matrix, other table like structures
renderText	character strings
renderUI	a Shiny tag object or HTML

Combining UI output and Server rendering

- Use an *Output function in the ui to place reactive objects in your Shiny app.
- Use a render* function in the server to tell Shiny how to build your objects.
- Surround R expressions by curly braces, {}, in each render* function.
- Save your render* expressions in the output list, with one entry for each reactive object in your app.
- Create reactivity by including an input value in a render* expression.

Hosting Shiny apps

- Apps can be hosted on shinyapps.io
- A variety of levels of accounts exist.
- The free tier is limited, but works for small apps.

Installation (Dash)

- I will work out of PyCharm, but you are free to use whatever environment you feel comfortable with.
- Install a newish version of Python 3 (Not 2). (I'm in v3.7.8).
- Install Dash (version 1.1.13 or newer should be fine):
  - `$ pip install dash` (or use conda). This will take a few minutes.
  - This should install several libraries:
    - dash, dash-core-components, dash-html-components
- Install jupyter-dash if you want to use Jupyter notebooks. (Not covered here).
- Ensure you have Pandas, plotly, and numpy installed.

Flask in Python

- Flask is a web framework written in Python
- For our purposes today, we will only use Flask tacitly (Dash).

Dash app basic features

- app = dash.Dash()
- app.layout = ….
- if __name__ == '__main__’:
  app.run_server(debug=True)

Plotly

- Intrinsic to Dash is Plotly
- https://plotly.com/python/
- One of the strongest benefits of Dash is the easy integration with the features of Plotly.

User input and output

- Use the function decorator

  `@app.callback([Output(), …, Output()], [Input()…, Input()])`
- This tells Dash what function to look to for input/output operations.
- In practice, the Output is usually a single element. Lists are not always absolutely necessary. But that syntax is clearer.

```
@app.callback(
    Output("output", "children"),
    Input("input1", "value"),
    Input("input2", "value"),
)
def update_output(input1, input2):
    ...
    return output
```

Idaho National Laboratory