Introduction to Shiny for Python

Aaron Geller

Northwestern | INFORMATION TECHNOLOGY

What is Shiny and why should I use it with Python?

Shiny has traditionally been the go-to R package for creating interactive web applications without needing to know HTML, CSS or JavaScript. Now the folks at Posit are working on Shiny for Python.

Why use Shiny when there are other tools in Python for this?

- fairly simple to create apps with widgets (easier than plotly, bokeh)
- easy to customize the look and feel
- recognizable look (for those used to R Shiny apps)
- free hosting on shinyapps.io (though with caveats for now)

Before starting

- Think conceptually about what you want to create. Maybe make a sketch or outline.
- Consider both
    - **functionality** (e.g., how do I want to interactively manipulate the data) and
    - **form** (e.g., what layout is going to be the most straightforward to the user).
- Create a working static version of your figure (or table) first (without Shiny).

Basic Shiny App Structure

app.py

```
from shiny import App, render, ui

app_ui = ui.page_fluid()

def server(input, output, session):

app = App(app_ui, server)
```

Basic Shiny App Structure

app.py

```
from shiny import App, render, ui

app_ui = ui.page_fluid()

def server(input, output, session):

app = App(app_ui, server)
```

- Load the Shiny library so the functions we need are available.

Basic Shiny App Structure

app.py

```
from shiny import App, render, ui

app_ui = ui.page_fluid()

def server(input, output, session):

app = App(app_ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).

Basic Shiny App Structure

app.py

```
from shiny import App, render, ui

app_ui = ui.page_fluid()

def server(input, output, session):

app = App(app_ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).
- The server contains a series of functions to define what the app does (e.g., to create plots).

Basic Shiny App Structure

app.py

```
from shiny import App, render, ui

app_ui = ui.page_fluid()

def server(input, output, session):

app = App(app_ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).
- The server contains a series of R statements to define what the app does.
- Create the Shiny App.

How to run the app

from a terminal

- Within a directory that has your `app.py` file, run the following:

  ```
  shiny run --reload app.py
  ```

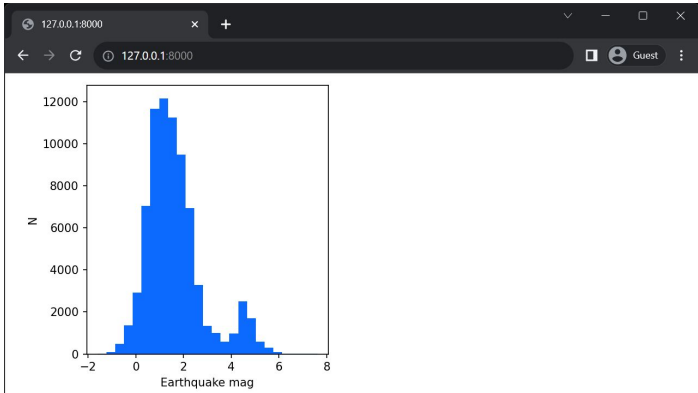
- Then open your browser to `http://127.0.0.1:8000/` to view your app. (If you change the `app.py` file and save it, the app should automatically reload in the browser.)

Examples

1. Just a plot, to show how to start the server
2. Add radio buttons that control the plot content, change layout
3. Add a scatter plot and sliders to control the symbol size
4. Add descriptive text
5. Add checkboxes and `panel_conditional` to control the ui + plots
6. Use `cartopy` to show map outlines, (try to) fix histogram heights

Example 1

Just a plot, to show how to start the server



- **Exercise 1.1:** add a second plot next to the existing one showing a histogram of `elevation` in the volcano data in red
- **Exercise 1.2:** change the earthquake histogram to show the `depth` column and the volcano histogram to show the `population_within_100_km` column

Example 2

Add radio buttons that control the plot content, change layout

```
25  # UI
26  ui.panel_well(
27      ui.input_radio_buttons(
28          "eCol", "Attribute for earthquake data", {"mag": "Magnitude", "depth": "Depth (km)"}
29      ),
30  ),
```

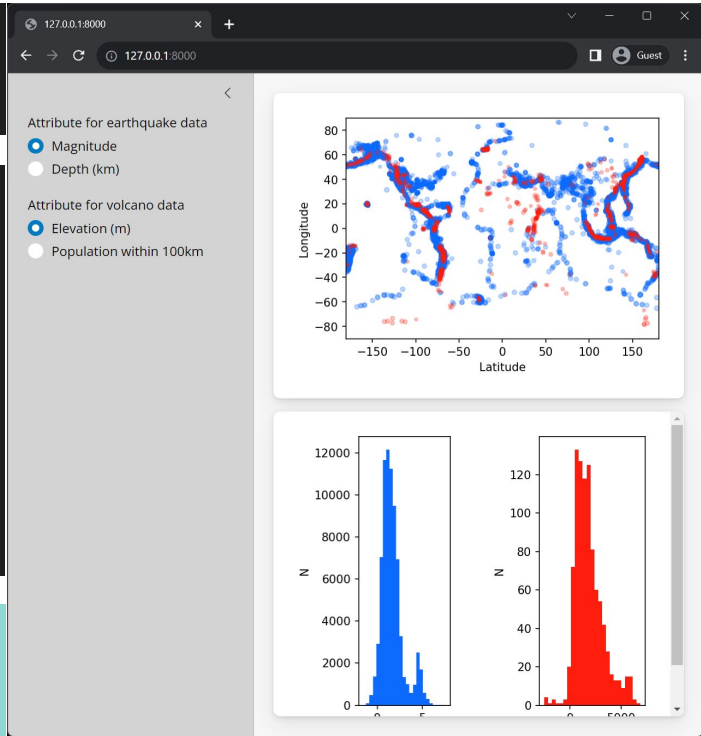
- **Exercise 2.1:** add radio buttons to control the volcano column
- **Exercise 2.2:** change `ui.page_fluid` to `ui.page_sidebar` and `ui.panel_well` to `ui.sidebar` (bonus: wrap the plots in a `ui.card`)

Example 3

Add a scatter plot and sliders to control the symbol size

```
37    # scatter plot
38    ui.card(
39        ui.output_plot("evscatter")
40    ),
41
```

```
79    # create a scatter plot of longitude vs. latitute
80    @output
81    @render.plot()
82    def evscatter():
83        esz = 20.*(edf[input.eCol()] - min(edf[input.eCol()]))/(max(edf[input.eCol()]) - min(edf[input.
          eCol()]))
84        vsz = 20.*(vdf[input.vCol()] - min(vdf[input.vCol()]))/(max(vdf[input.vCol()]) - min(vdf[input.
          vCol()]))
85        fig, ax = plt.subplots()
86        ax.scatter(edf['longitude'], edf['latitude'], color = ecolor, s = esz, alpha = 0.25)
87        ax.scatter(vdf['longitude'], vdf['latitude'], color = vcolor, s = vsz, alpha = 0.25)
88        ax.set_ylabel('Longitude')
89        ax.set_xlabel('Latitude')
90        ax.set_xlim(-180, 180)      You, 6 days ago • finished examples
91        ax.set_ylim(-90, 90)
92        return fig
```



- **Exercise 3.1:** add two `ui.input_slider` to scale the size of the points in the scatter plot

Example 4

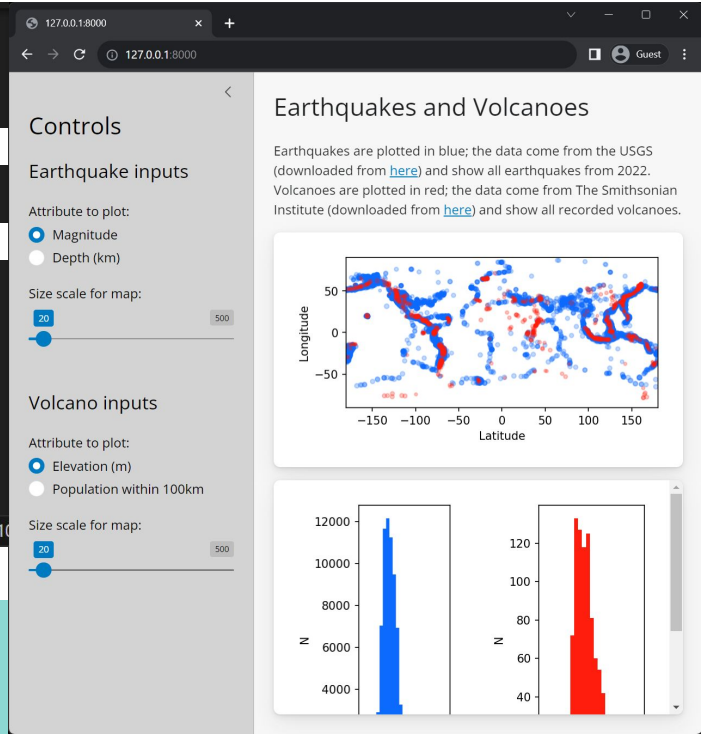
Add descriptive text

```
24    # UI
25    ui.sidebar(
26        ui.h2("Controls"),
27        ui.h4("Earthquake inputs"),

34        ui.br(),
35        ui.h4("Volcano inputs"),

45    # title
46    ui.h1("Earthquakes and Volcanoes"),
47    ui.p("Earthquakes are plotted in blue; the data come from the USGS (downloaded from ", ui.a("here",
      href = "https://www.kaggle.com/datasets/thedevastator/
      uncovering-geophysical-insights-analyzing-usgs-e"), ") and show all earthquakes from 2022.
      Volcanoes are plotted in red; the data come from The Smithsonian Institute (downloaded from ", ui.a
      ("here", href = "https://www.kaggle.com/datasets/jessemostipak/volcano-eruptions"), ") and show all
      recorded volcanoes."),
48
```

You, 6 days ago Ln 32, Col 37 Spaces: 4 UTF-8 LF Python 3.8.1



- **Exercise 4.1:** experiment with editing this text and adding your own

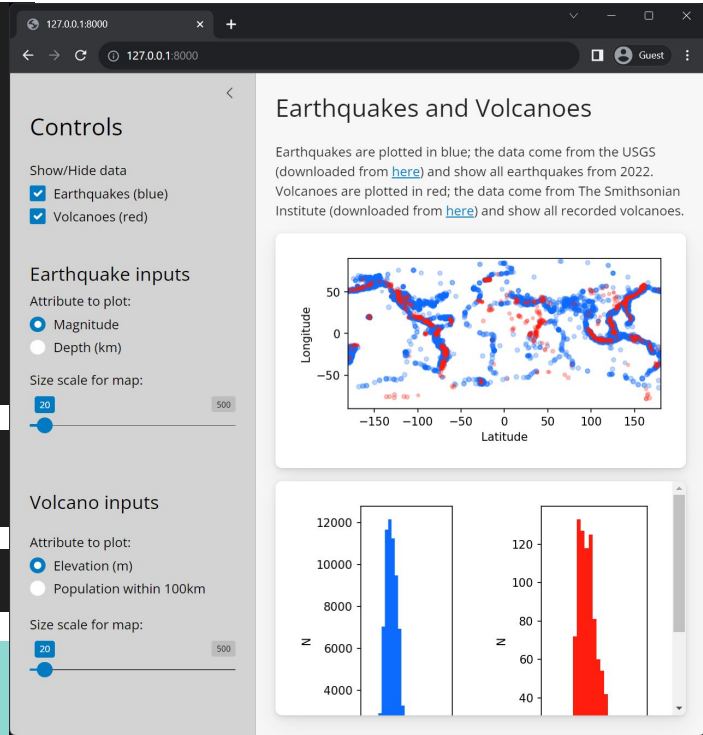
Example 5

Add checkboxes and `panel_conditional` to control the ui + plots

```
27      ui.input_checkbox_group(
28          "toggle", "Show/Hide data", {"Earthquakes": "Earthquakes (blue)", "Volcanoes": "Volcanoes
            (red)"},
29          selected = ["Earthquakes", "Volcanoes"]
30      ),
31      ui.panel_conditional(
32          "input.toggle.includes('Earthquakes')",
33          ui.br(),
34          ui.h4("Earthquake inputs"),
35          ui.input_radio_buttons(
36              "eCol", "Attribute to plot:", {"mag": "Magnitude", "depth": "Depth (km)"}
37          ),
38          ui.input_slider(
39              "eSize", "Size scale for map:", min=1, max=500, value=20, step=1
40          ),
41      ),
```

```
69              ui.panel_conditional(
70                  "input.toggle.includes('Earthquakes')",
71                  ui.output_plot("ehist")
72              ),
```

```
113     if ('Earthquakes' in input.toggle()):
114         ax.scatter(edf['longitude'], edf['latitude'], color = ecolor, s = esz, alpha = 0.25)
```



- **Exercise 5.1:** repeat this process for the volcano data

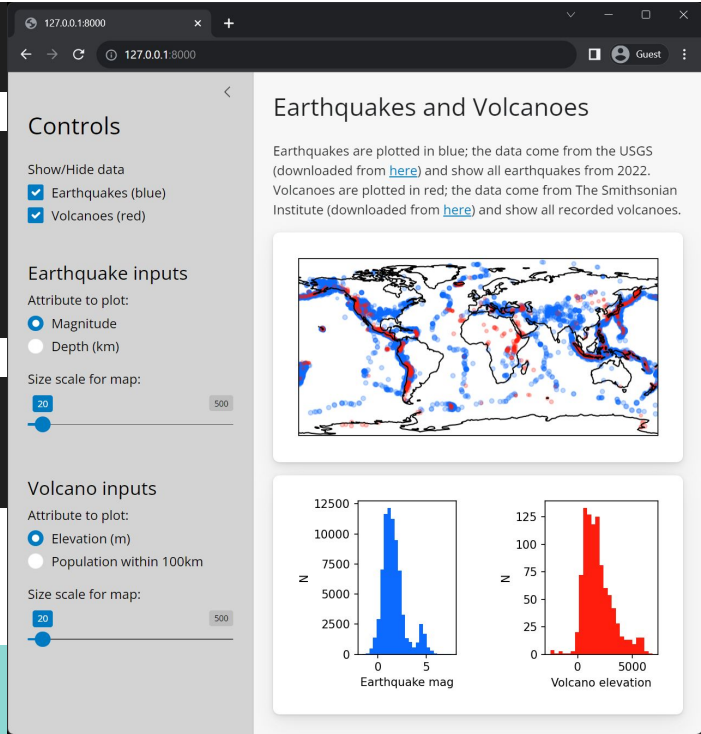
Example 6

Use `cartopy` to show map outlines, (try to) fix histogram heights

```
4   import cartopy.crs as ccrs
5   import cartopy.feature as cfeature

85                      ui.output_plot("vhist", height = 260)
86                  ),
87              ),
88          ),
89          height = 450,

126         fig = plt.figure()
127         ax = fig.add_subplot(1,1,1, projection = ccrs.PlateCarree())
128         ax.add_feature(cfeature.COASTLINE)
```



- **Exercise 6.1:** experiment!

Hosting on shinyapps.io

First, sign up for an account on shinyapps.io .

Then see documentation here.

- The similar app to this example is hosted on shinyapps.io here .
- Full disclosure, getting this app running on shinyapps.io was *much* more difficult than doing so in R.
  - Most/all of this difficulty was from the map (`cartopy` didn't work, and `plotly` introduced other challenges).
  - Also shinyapps.io only supports certain Python versions.
  - You can read about this in my blog post here.