Creating Dashboards with Python

More specifically – plotly & Dash

Summer 2023

Housekeeping

- In case of technical problems:
  - Something wrong on my end (e.g. power outage), I will send you an email.
  - Something wrong on your end, please send me a text message. 508-769-6446
  - jcodygroup@gmail.com
- We have 4 hours for each session
  - I will try to give you an opportunity to stand and stretch every hour.
  - We will take at least one 15-minute break near the halfway point.

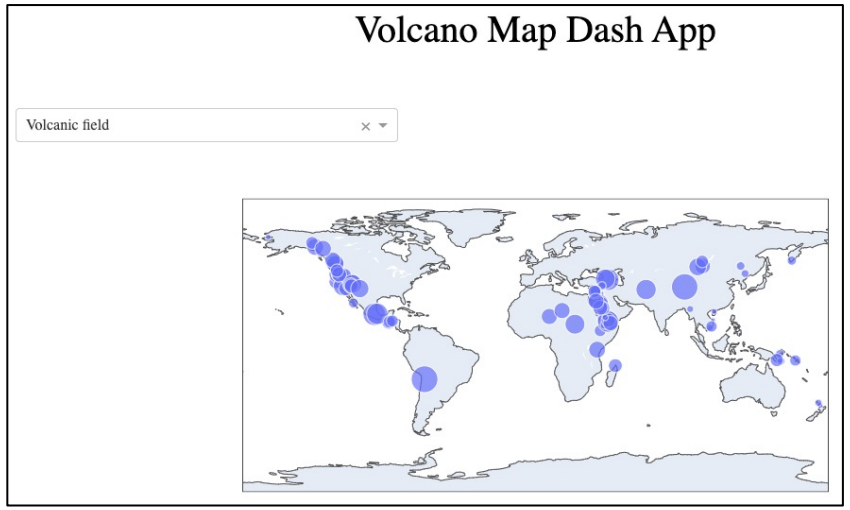
About me

- **Experience:**
  - 25+ years consulting and training experience
  - Extensive work with “big data” and analytics
  - 15 years working with various data visualization tools
- **Education**
  - Ed. M., Technology, Innovation & Education, Harvard University
  - PhD Candidate, Education Policy, University of Massachusetts, Amherst

A dashboard created with Python and Dash

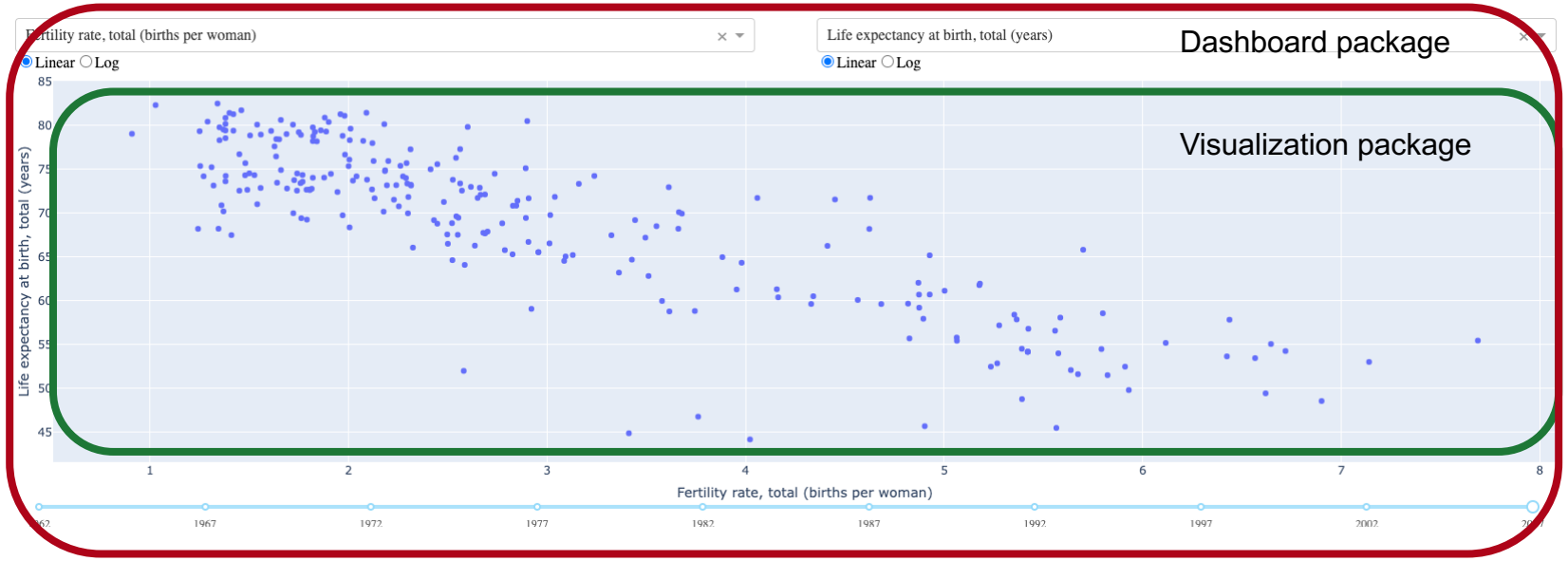
https://live-volcano-map-dash-app.onrender.com/

Web Pages: HTML tags, CSS & javascript




```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Dash</title>
        <link rel="icon" type="image/x-icon" href="/_favicon.ico?v=2.6.0">

    </head>
    <body>

<div id="react-entry-point">
    <div class="_dash-loading">
        Loading...
    </div>
</div>

        <footer>
            <script id="_dash-config" type="application/json">{"url_base_pathname":null,"requests_pathname_prefix":"/"
            <script src="/_dash-component-suites/dash/deps/polyfill@7.v2_6_0m1665699684.12.1.min.js"></script>
<script src="/_dash-component-suites/dash/deps/react@16.v2_6_0m1665699684.14.0.min.js"></script>
<script src="/_dash-component-suites/dash/deps/react-dom@16.v2_6_0m1665699684.14.0.min.js"></script>
<script src="/_dash-component-suites/dash/deps/prop-types@15.v2_6_0m1665699684.8.1.min.js"></script>
<script src="/_dash-component-suites/dash/dash-renderer/build/dash_renderer.v2_6_0m1665699684.min.js"></script>
<script src="/_dash-component-suites/dash/dcc/dash_core_components.v2_6_0m1665699684.js"></script>
<script src="/_dash-component-suites/dash/dcc/dash_core_components-shared.v2_6_0m1665699684.js"></script>
<script src="/_dash-component-suites/dash/html/dash_html_components.v2_0_4m1665699684.min.js"></script>
<script src="/_dash-component-suites/dash/dash_table/bundle.v5_1_4m1665699684.js"></script>
            <script id="_dash-renderer" type="application/javascript">var renderer = new DashRenderer();</script>
        </footer>
    </body>
</html>
```

Creating a web-based application



Create javascript

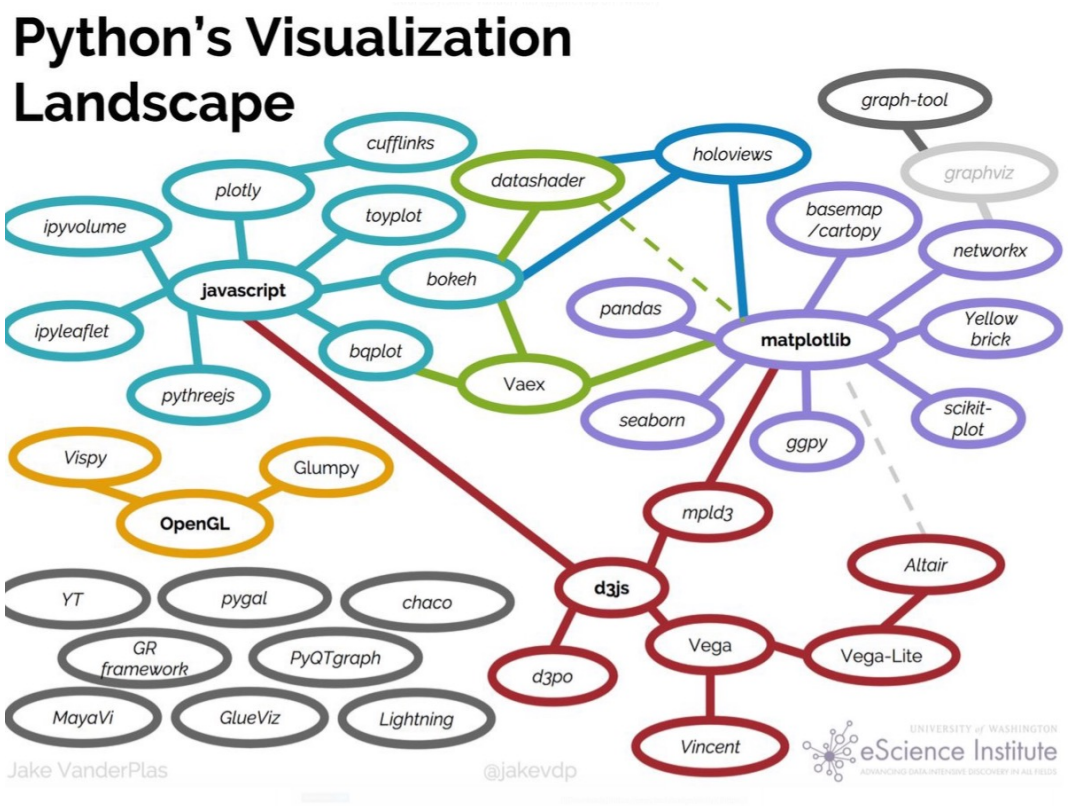
Dashboard Options

Dash
Voila
Streamlit
Bokeh
Panel

Dash App Structure

```python
# The general structure of a dashboard application:

imports .....

app = JupyterDash(__name__)     # This is the start of the application

get the data....

create a figure(plot)...

app.layout =                    # Describe what the page will look like

    layout code

    dcc.Graph()                 # What plot will be included

@app.callback(
    what are the inputs?
    what are the outputs?

    resusable component )       # This processes the input and creates the output

app.run_server(mode='inline')   # .run_server() is the method to run the code


```

```python
# An example of a callback from documentation
# Just changes the text that appears - no plotting

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import dcc
from dash import html

app = JupyterDash(__name__)

app.layout = html.Div([
    html.H6("Change the value in the text box to see callbacks in action!"),
    html.Div([
        "Input: ",
        dcc.Input(id='my-input', value='initial value', type='text')
    ]),
    html.Br(),
    html.Div(id='my-output'),

])


@app.callback(
    Output(component_id='my-output', component_property='children'),
    Input(component_id='my-input', component_property='value')
)
def update_output_div(input_value):
    return 'Output: {}'.format(input_value)

app.run_server(mode='inline')
#app.run_server(mode='external', port = 8071)
```



matplotlib

Version 3.4.3

seaborn: statistical data visualization

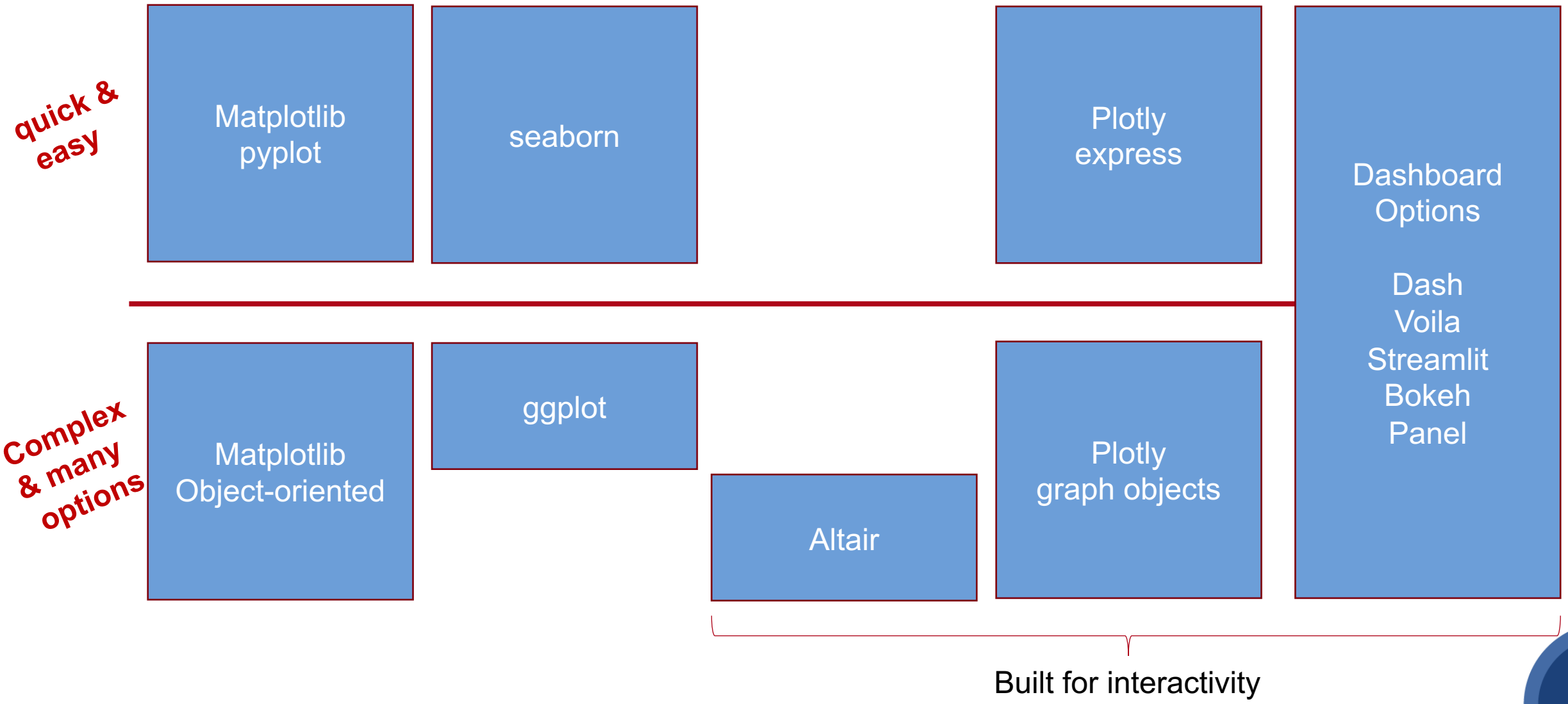
The Bokeh Visualization Library

plotnine 0.8.0 API Gallery Tutorials Site Page

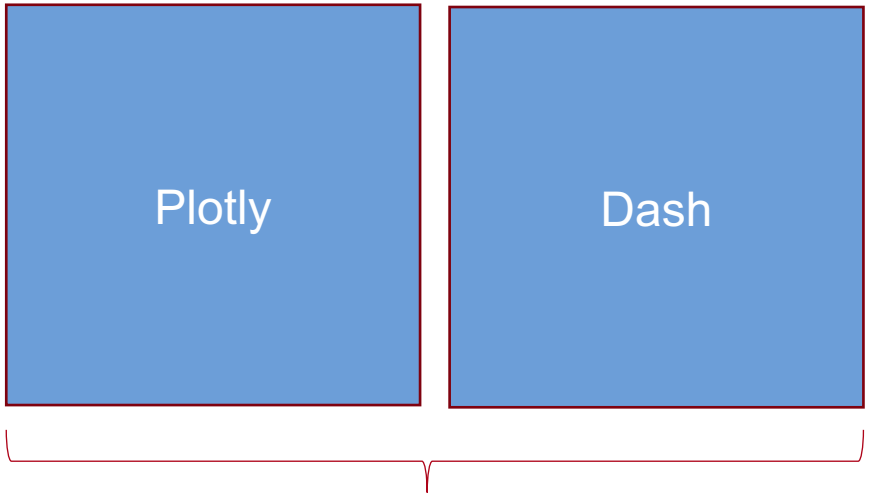
A Grammar of Graphics for Python

Altair: Declarative Visualization in Python

Visualization packages



Our tasks


Plotly
Dash
Python
(notebooks
ipywidgets)


Deploy to Web
git
heroku
and through
kaggle

Anatomy of a figure

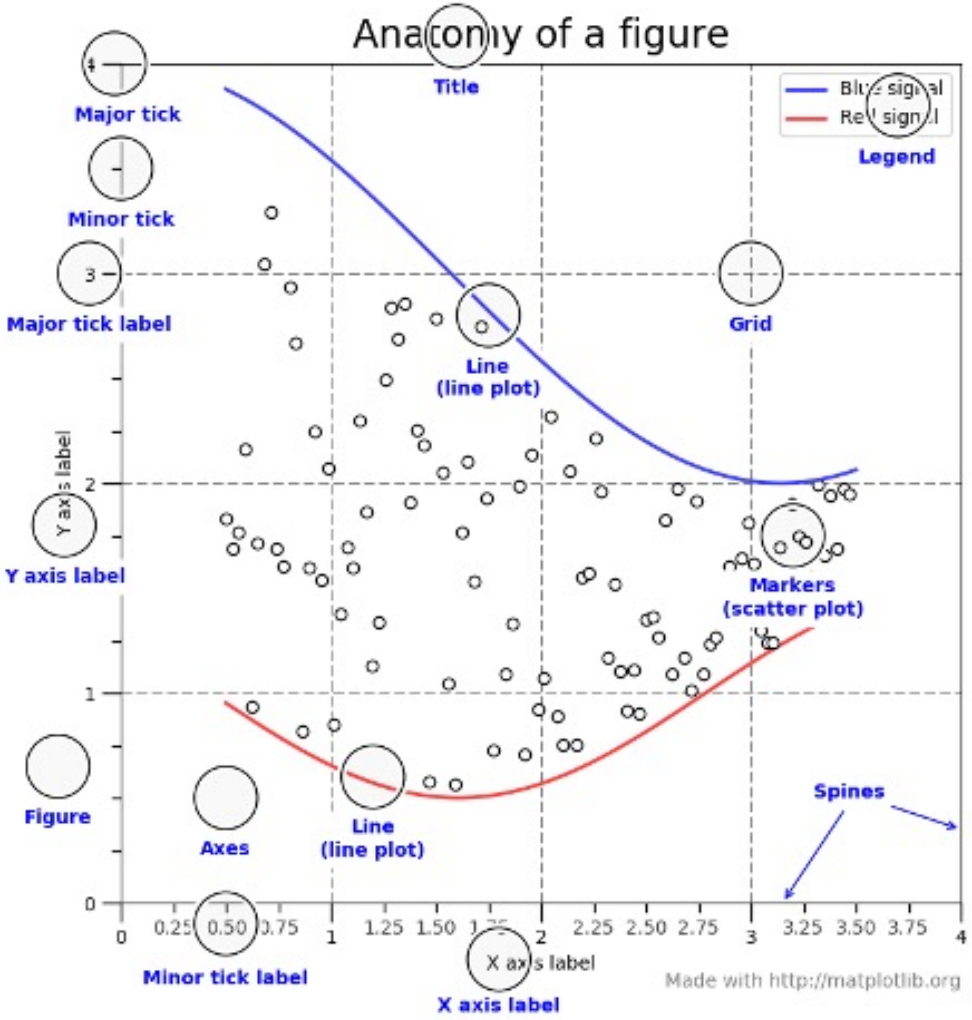
quick & easy

Matplotlib pyplot

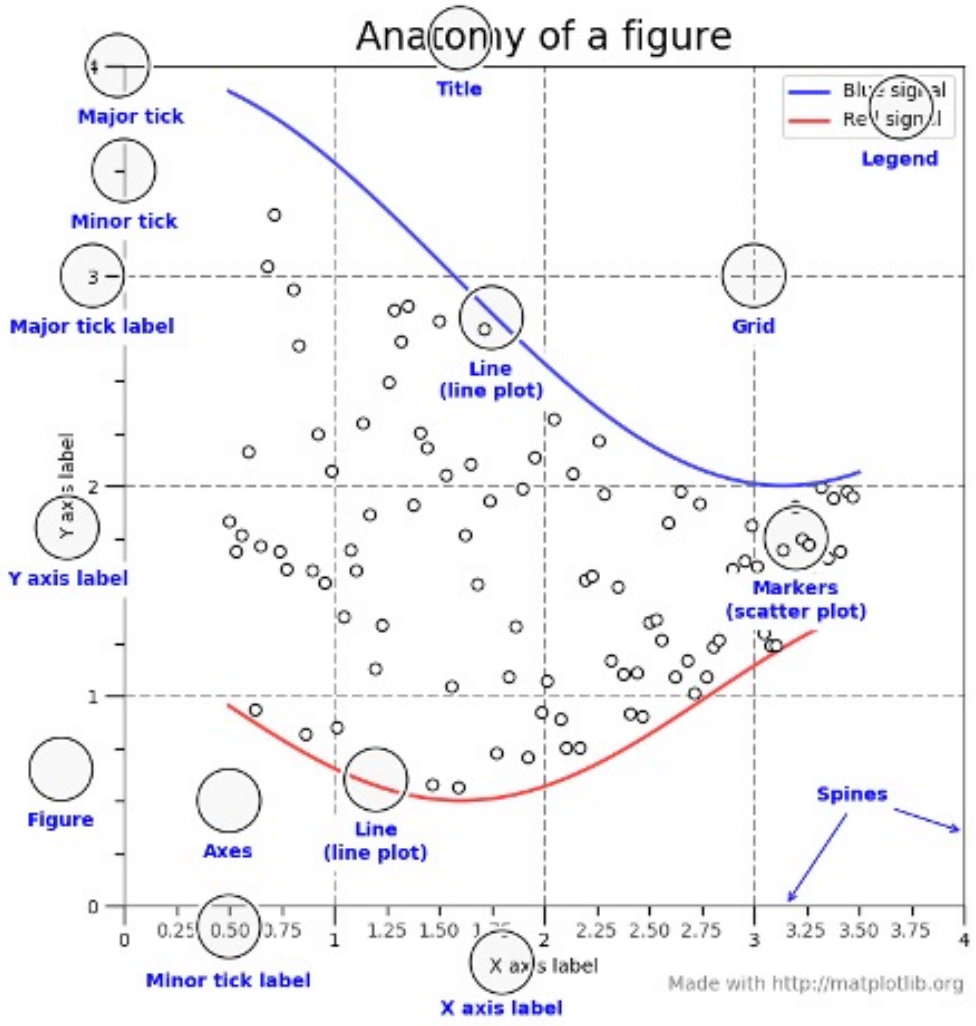
seaborn

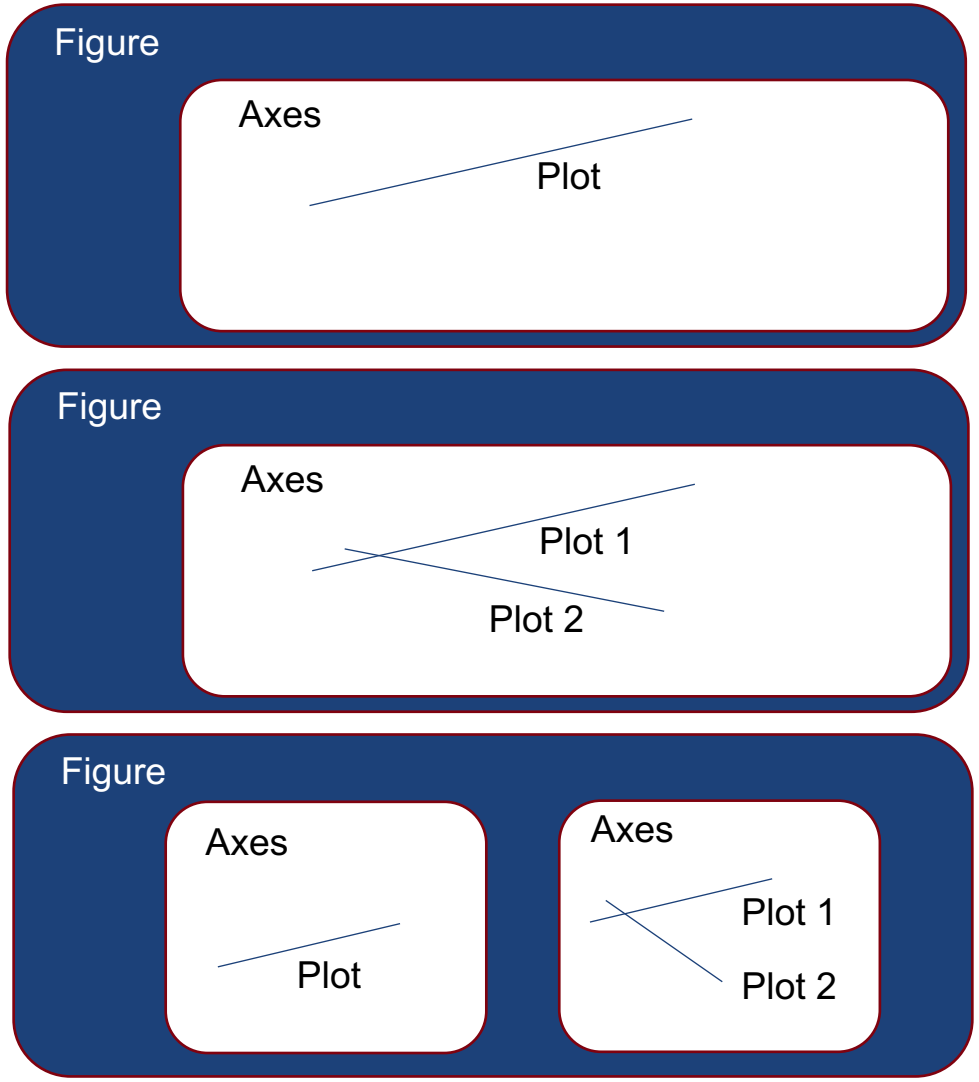
Complex & many options

Matplotlib Object-oriented

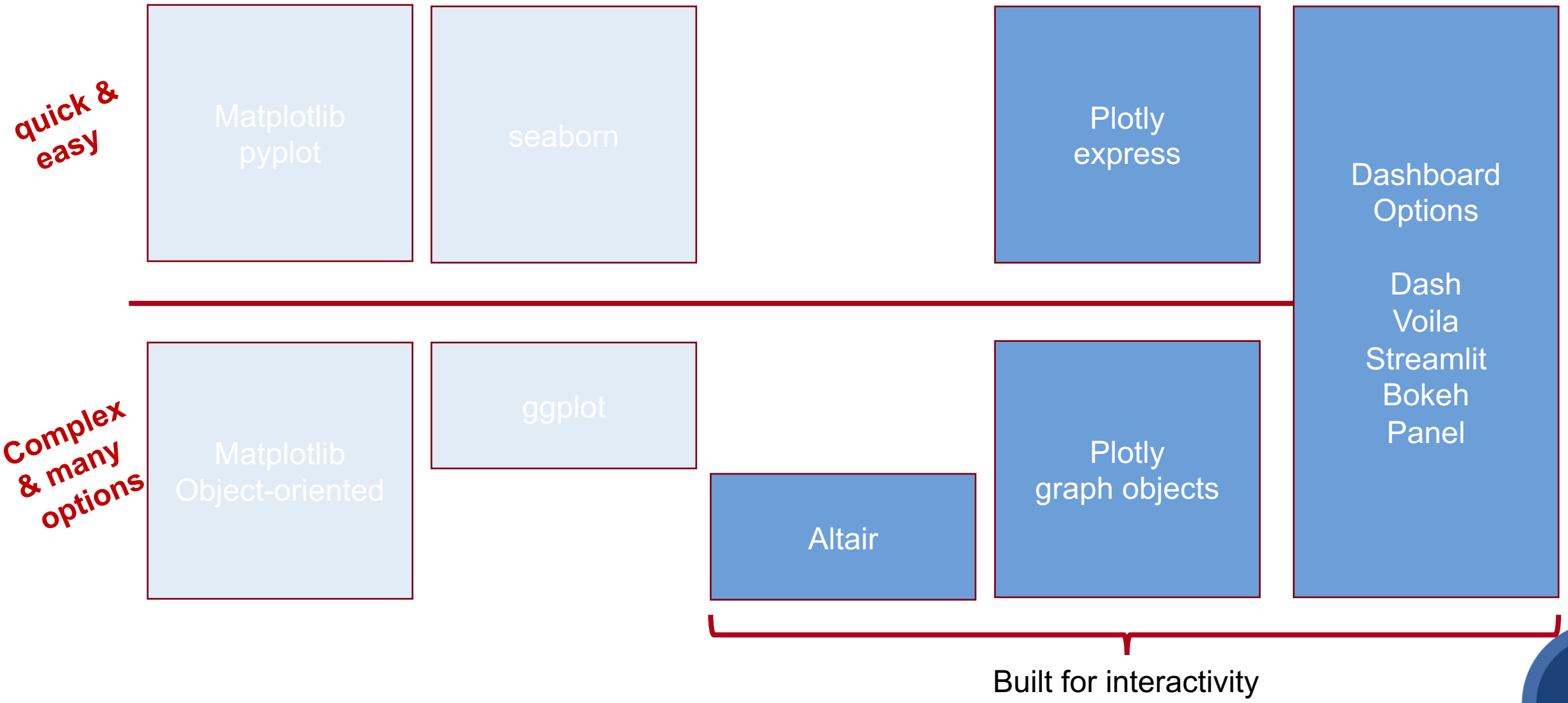


Matplotlib/Seaborn: Two big concepts

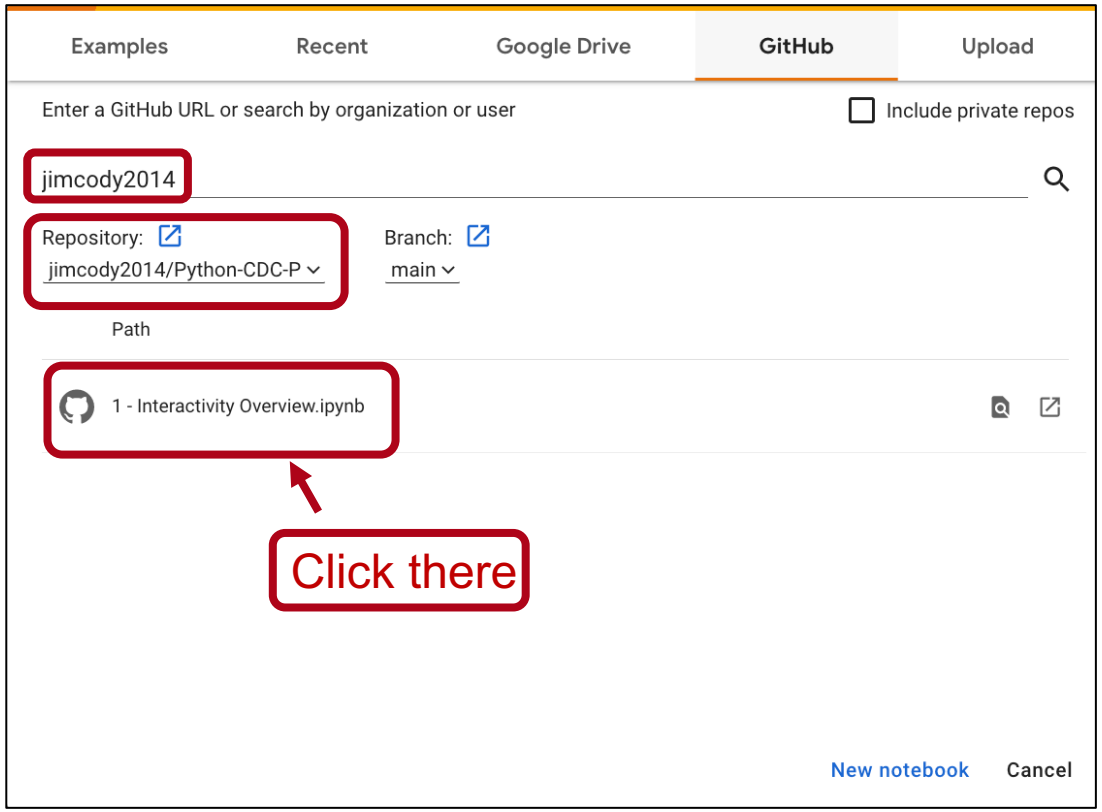


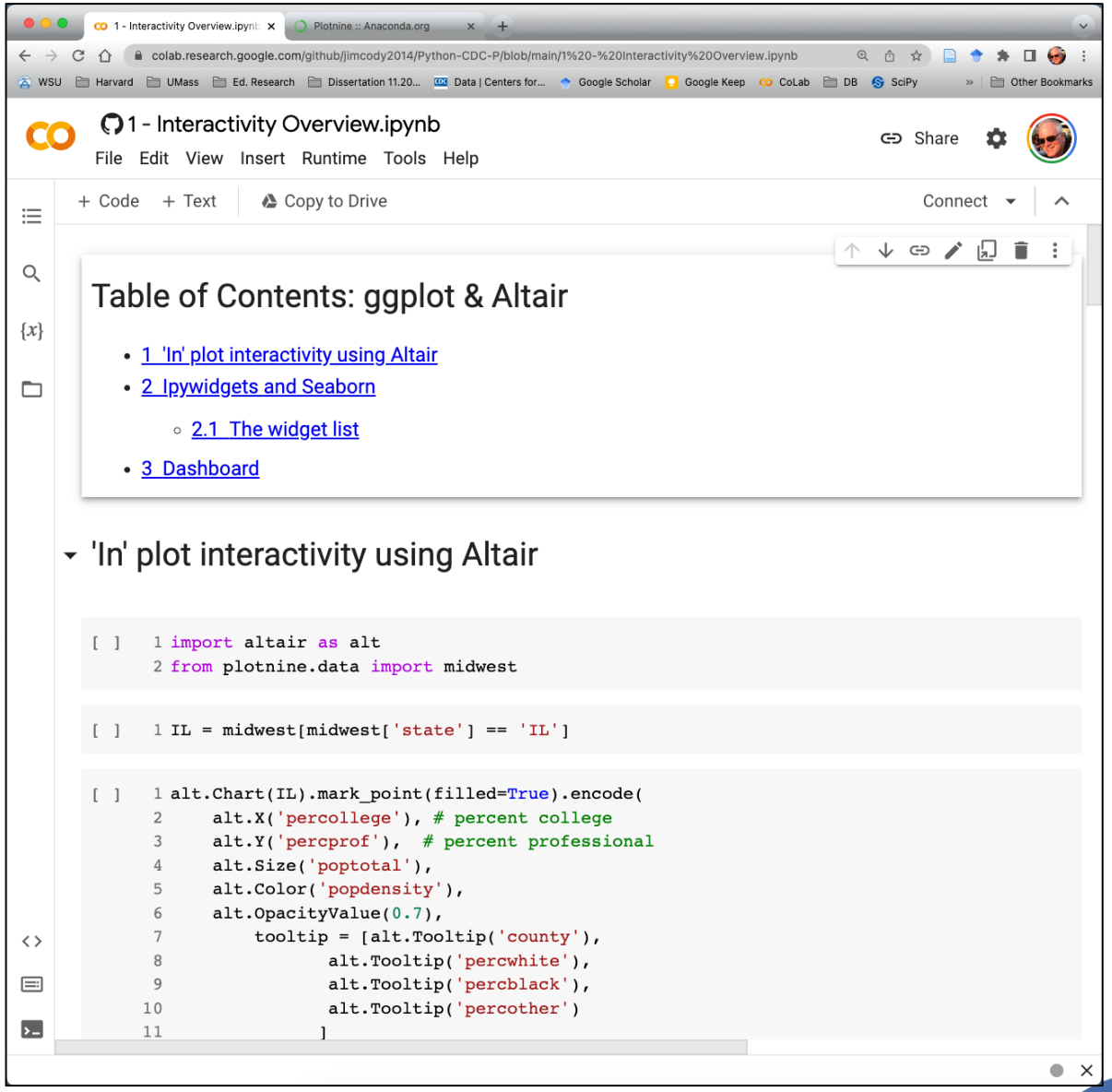


Built for interactivity

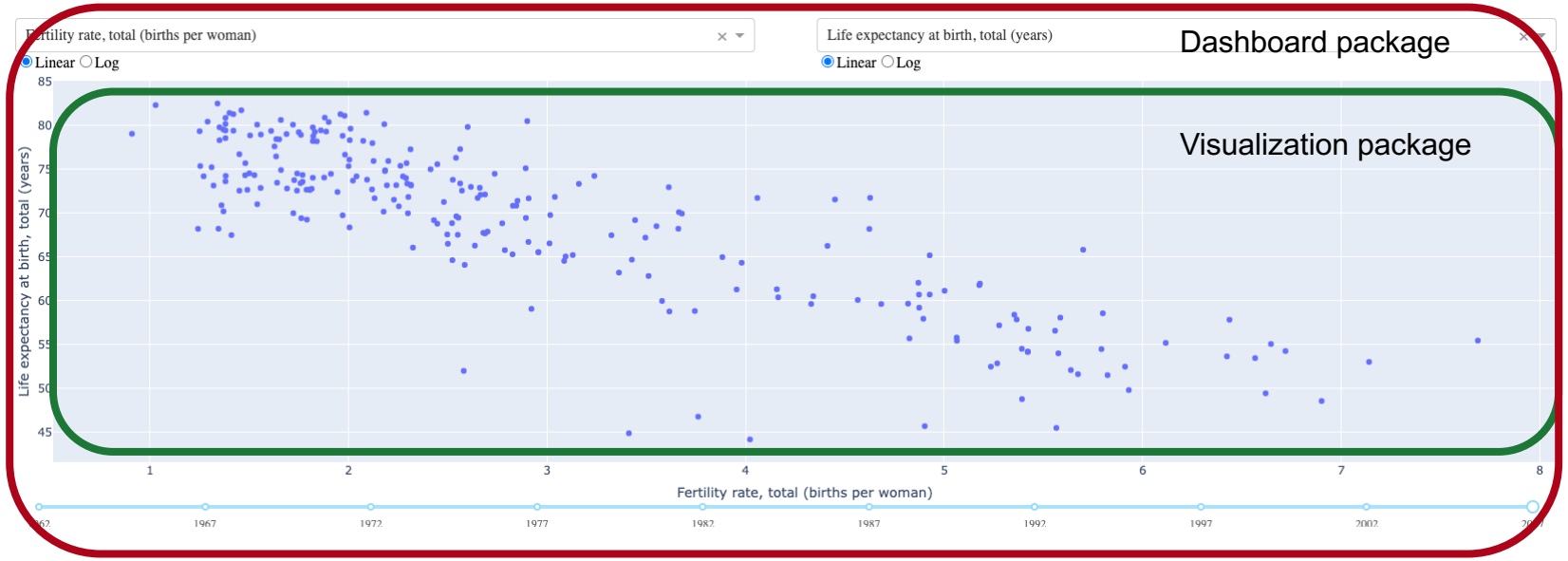


Interactivity notebook





Creating a web-based application

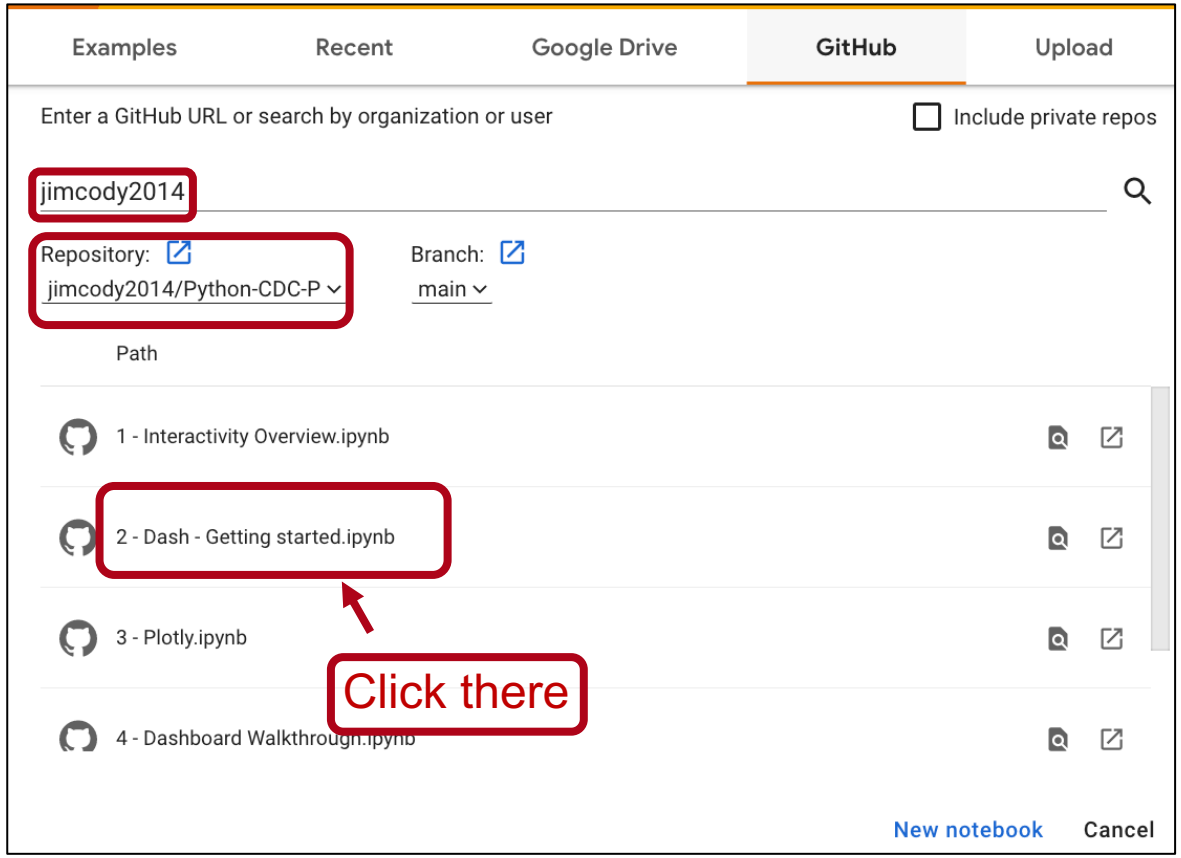


Create javascript

Dashboard Options

Dash
Voila
Streamlit
Bokeh
Panel

Notebook: 2 – Dash – Getting started



What is Dash?

- Dash is a 'low-code' framework (set of tools and procedures) for building dashboard applications using Python, Julia or R.
- It is tightly integrated with plotly, the data visualization library.
- The dashboard can be rendered as part of the Jupyter notebook, in an HTML file on a local machine or as an HTML file on a web server.

Building blocks

Layouts and Callbacks

- Dash apps are composed of two parts.
- The first part is the "layout" of the app and it describes what the application looks like.
- Callbacks, the second part, describes the interactivity of the application

HTML components and dash core components

- HTML components are the dash equivalant of html tags
- Core components (dcc) are graphs, markdown blocks and interactivity components like sliders, dropdowns, etc.

Our tasks

Dash Introduction

- Terminology
- Structure
- CoLab

Plotly express

- px.line
- px.scatter
- px.bar
- facets

Plotly graph objects

- Figures
- Layout
- Traces
- Data
- Update layout
- Hover text

Dash exploration

- Layout
- Add a plot
- Change plot
- Html components
- Component args.
- Div()
- Positioning
- Repositioning
- Markdown text
- Interactivity comps
- Callback
- Reusable comp
- Simple callback
- Changing variables

Dash again

- Terminology
- Structure

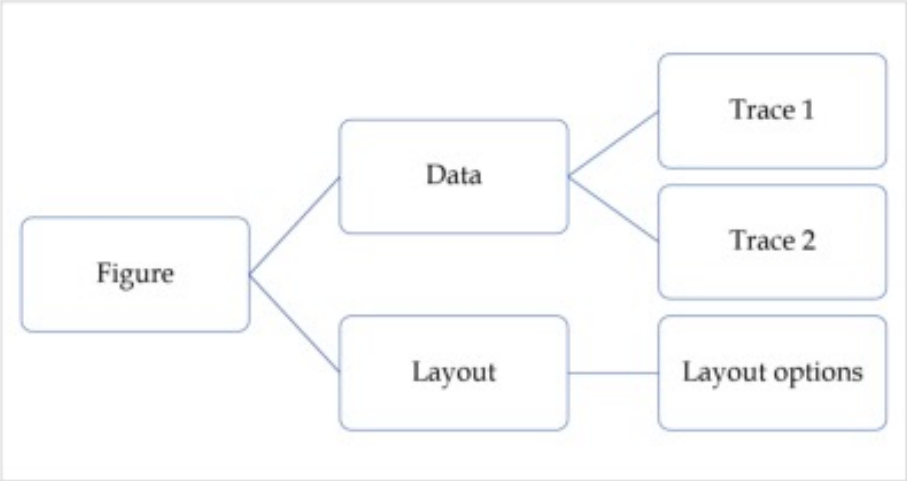
Deploy to Web

git & heroku

plotly graph objects

The **plotly.graph_objs module** is the most important module that contains all of the class definitions for the objects that make up the plots you see. Following graph objects are defined:

- Figure,
- Data,
- Layout,
- Different graph traces like **Scatter**, **Box**, **Histogram** etc.



All graph objects are dictionary- and list-like objects used to generate and/or modify every feature of a Plotly plot.

Dash Basics

- Layout & Callback
  - Layout: how the page will look
  - Callbacks: How to make the page interactive
- Html components
  - There is a ‘Dash’ component for every html tag
- Core components
  - Configurable options to ‘do things’ (e.g., make a dropdown list, place a chart, add an input text box, etc.)