Introduction to Dash and DashR

July 8, 2020

Lava Meeting

10 minutes

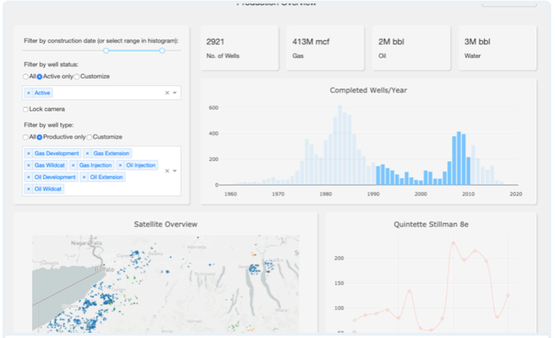
Part 1:
Introduction to Dash

Dash App Gallery

Click on a demo app's name for more info and links to Python and R source code where available. More info at https://plot.ly/dash

All Apps (63)

Search applications...

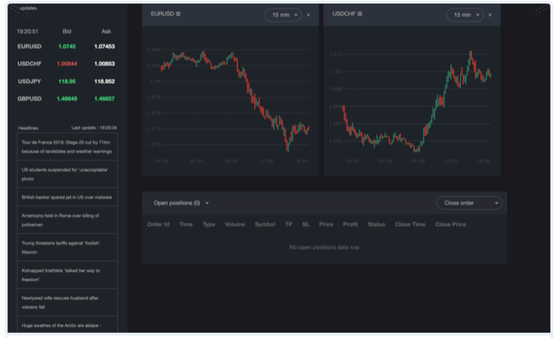


New York Oil and Gas

Energy | Cross-filtering | Geospatial

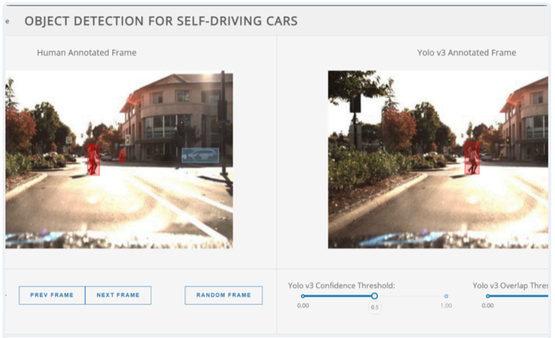
Manufacturing SPC Dashboard

Data Acquisition | Streaming



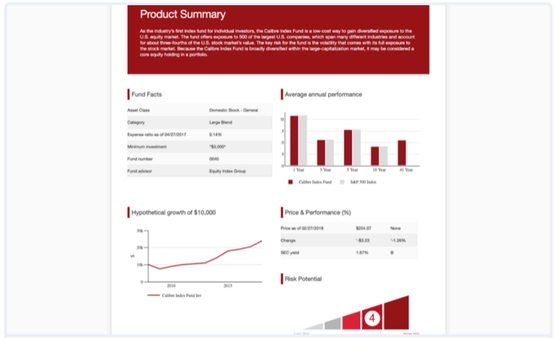
FOREX Web Trader

Streaming | Finance



Real-Time Object Detection

Artificial Intelligence | Machine Learning (+1)



Financial Report

Finance

Multipage Report

Reporting | Finance

Why Dash?

- Available in R and Python
- Web-based
- Open Source
- Easily versioned
- Fast, light-weight (built on top of Flask)
- Separate your wrangling/analysis from dashboarding
- Customizable with bootstrap components and CSS
- Continuous integration/deployment

Summary

- Dash is an awesome framework to build powerful interactive dashboards in R and Python!

- Dash will allow us to use our Altair and ggplot2 visualizations (mostly) as-is!

- Dash is open source, has a vibrant community and is a boon for reproducible dashboards

10 minutes

Part 1: Install Dash

Dash for python

```
Step 1: Open a terminal

Step 2: Run
        `pip install dash`
```

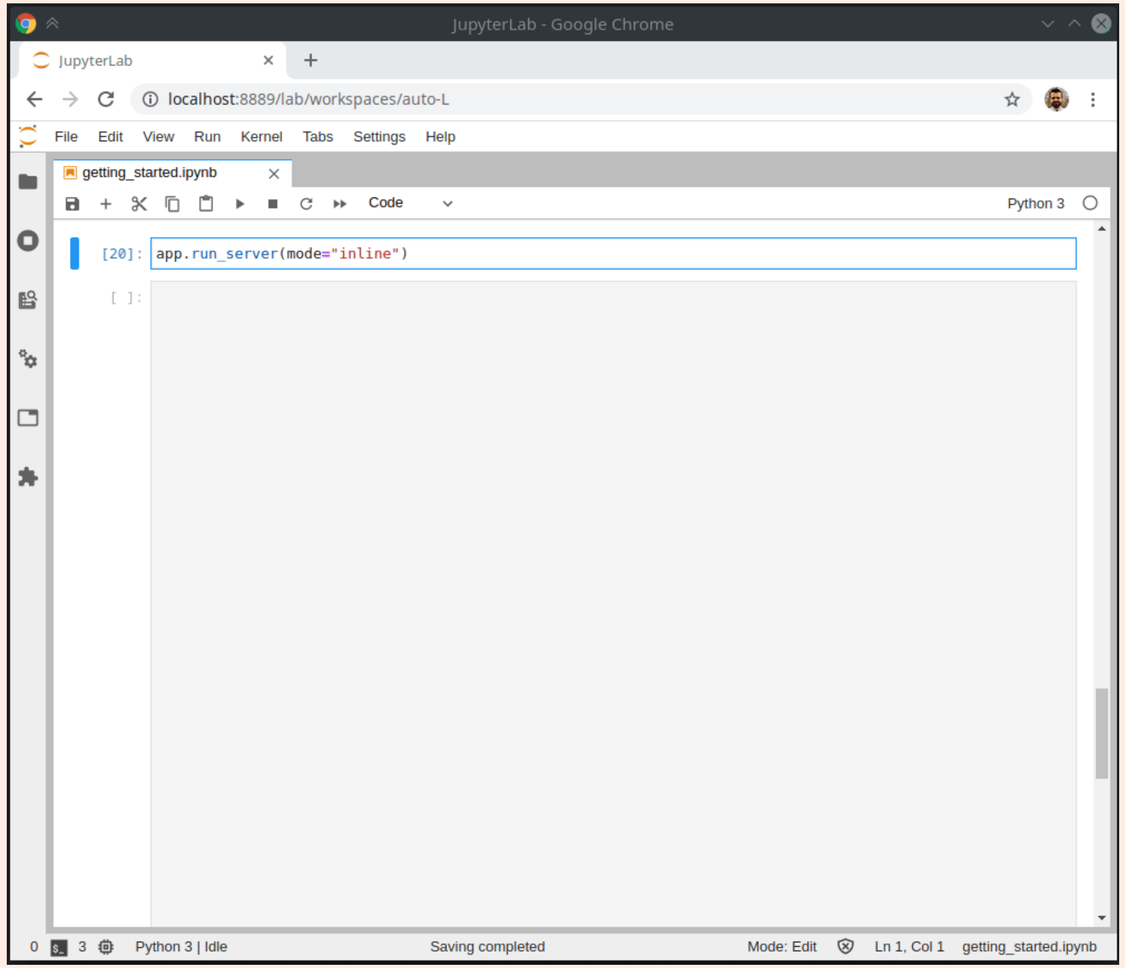
https://dash.plotly.com/installation

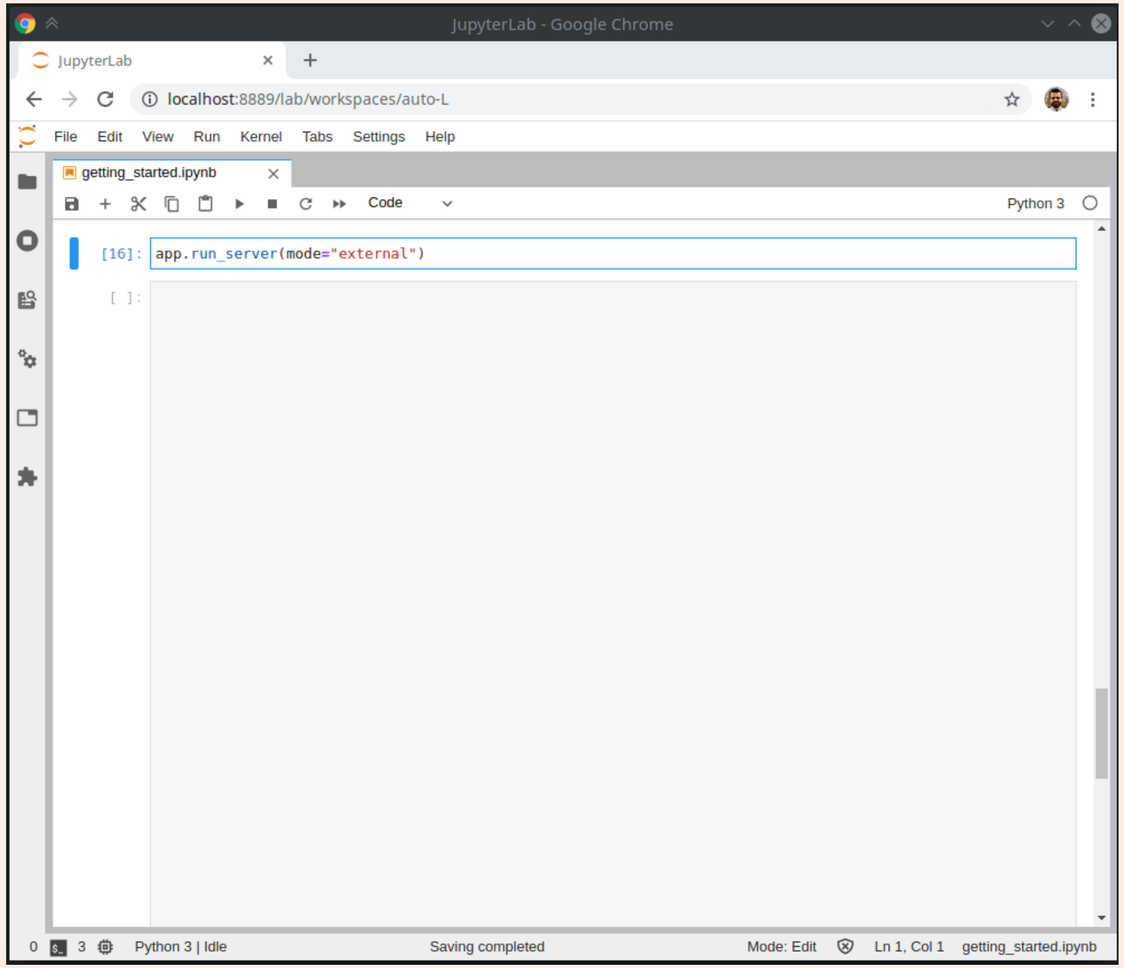
Dash for R

```
Step 1:Launch R (or RStudio)

Step 2: Run
        `install.packages('dash')`
```

https://dashR.plotly.com/installation

Really cool way of developing Dash apps





Source: JupyterDash

Let’s try it now!

Click this link (then wait a couple of minutes):

https://mybinder.org/v2/gh/plotly/jupyter-dash/master?urlpath=tree/notebooks/getting_started.ipynb

10 minutes

Part 3: Dash Overview

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

```python
## 1. Import packages

import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
```

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

```
 8  ## 2. Setup app
 9  app = dash.Dash(__name__)
10  server = app.server
11  app.title = 'Dash app with pure Altair HTML'
12
```

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

```
13  ## 3. Data Analysis
14
15  ...
16
```

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

```
17  ## 4. Layout (dash components)
18
19  app.layout = html.Div([
20
21      html.H1('This is a level 1 heading'),
22      html.H2('This is a level 2 subheading'),
23      dcc.Markdown('''
24
25          Inside this, I can *write* normally!
26
27          ## Headings are also respected
28
29          '''),
```

Skeleton Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

You can even add iFrames to bring in ANY html content!

```
31      html.H3('Here is an image'),
32      html.Img(src='https://upload.wikimedia.org/wi
33              width='10%'),
34      html.H3('Here is our first plot:'),
35      html.Iframe(
36          sandbox='allow-scripts',
37          id='plot',
38          height='450',
39          width='625',
40          style={'border-width': '0'},
41
42          ################ The magic happens here
43          srcDoc=open('chart.html').read()
44          ################
45          ),
46  ])
```

Skeleton of a Dash app

app.py

1.import packages

2.Setup app

3.Data Analysis

4.Layout (dash components)

5. Run app

```
48  if __name__ == '__main__':
49      app.run_server(debug=True)
50
```

Launching a Dash app

Step 1: Open a terminal

Step 2: Navigate to the directory that has app.py

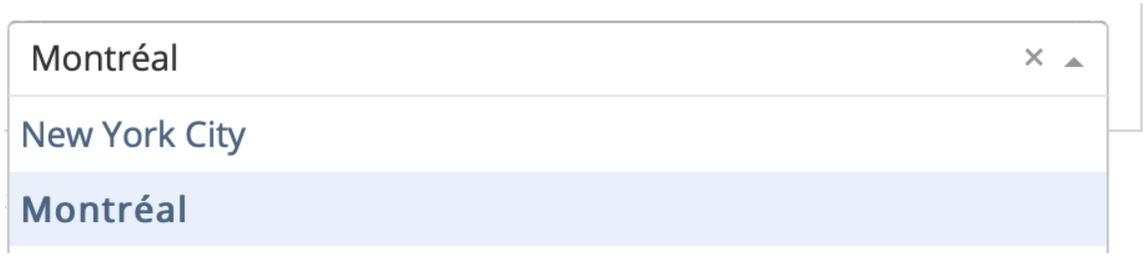
Step 3: Run `python3 app.py`

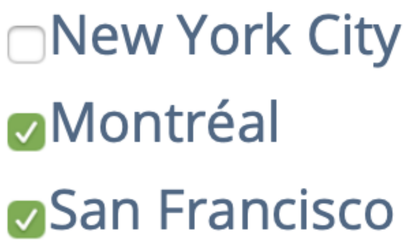
Step 4: Open http://127.0.0.1:8050 in a browser!

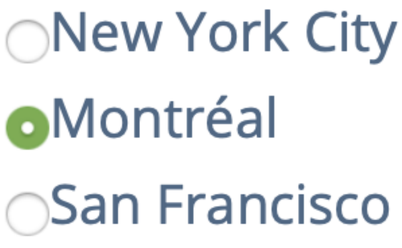
```
~/dash_demo  master ✗
▶ python3 app.py
Dash is running on http://127.0.0.1:8050/

 Warning: This is a development server. Do not use app.run_server
 in production, use a production WSGI server like gunicorn instead.

 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

5 minutes

Part 4: Dash Components

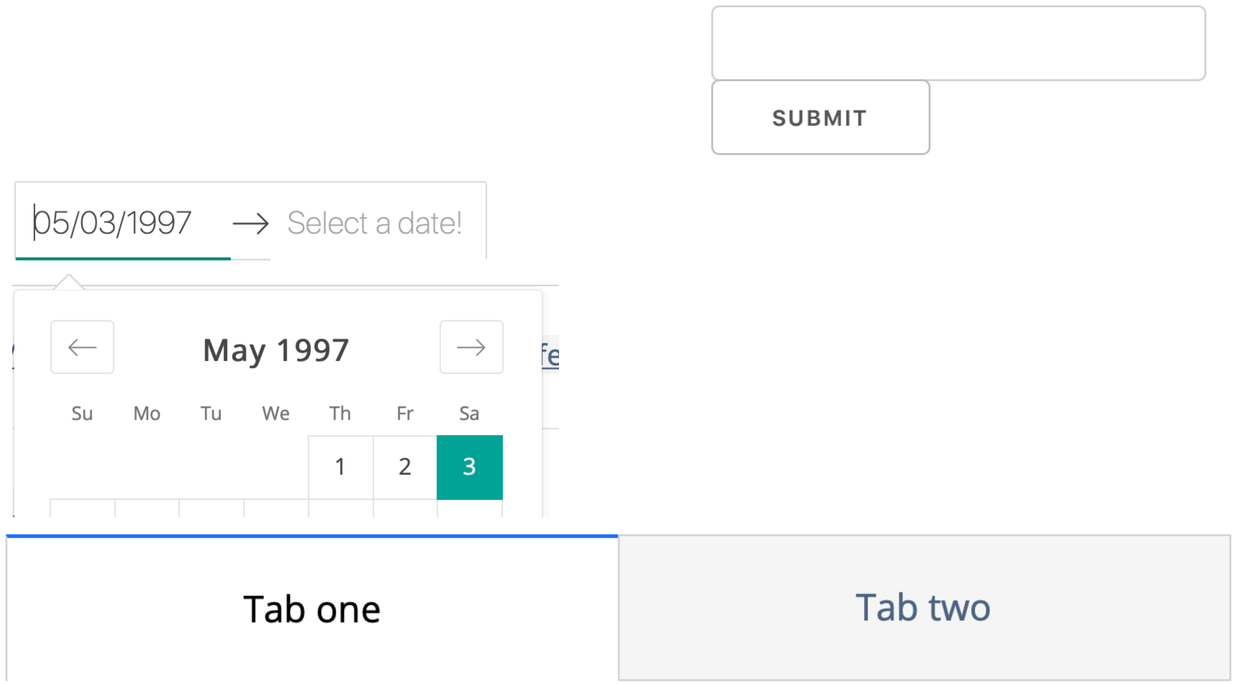






Source: Dash Core Components

Component Name	Description
Dropdown	List of items in a dropdown menu
Slider	Single element that selects a **value**
RangeSlider	Dual elements that selects a **range**
Checklist	Check all that apply
RadioItems	Select an item from list



Component Name	Description
Button	Button-click to produce an action
DatePickerRange	Date-picker
Tabs	Separate areas of your dashboard with tabs
DataTable	View, sort, manipulate a table
Upload	Upload your own data
Markdown	Use Markdown syntax for text formatting

Source: Dash Core Components

DataTable

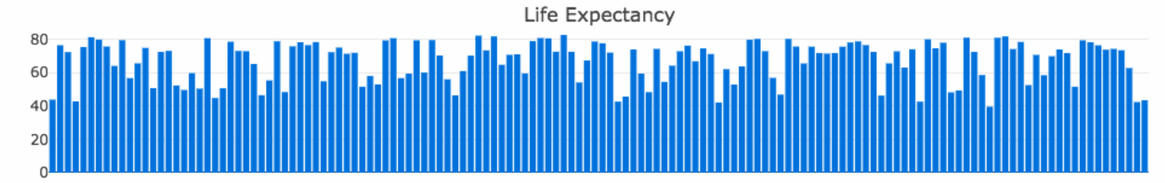
- Dash DataTable is the newest, and coolest component in the Dash library
- It's designed to work best with plot.ly (another visualization library) graphs

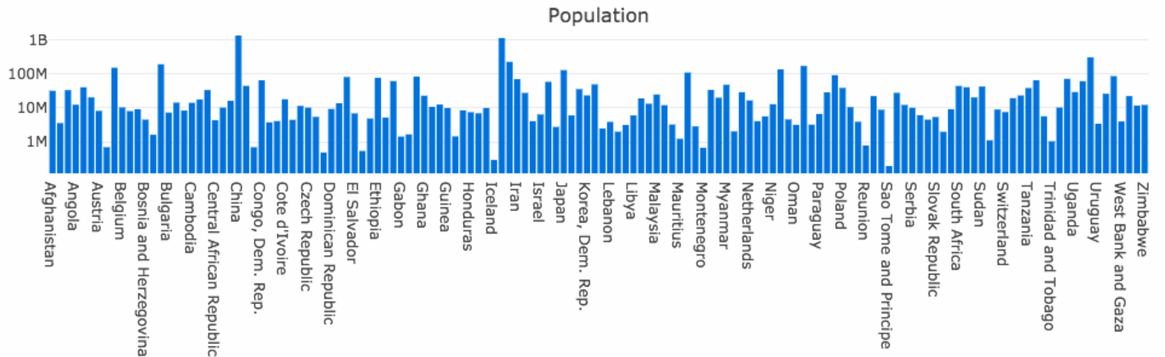
Code: Dash Core Components

Dash DataTable

FILTER ROWS

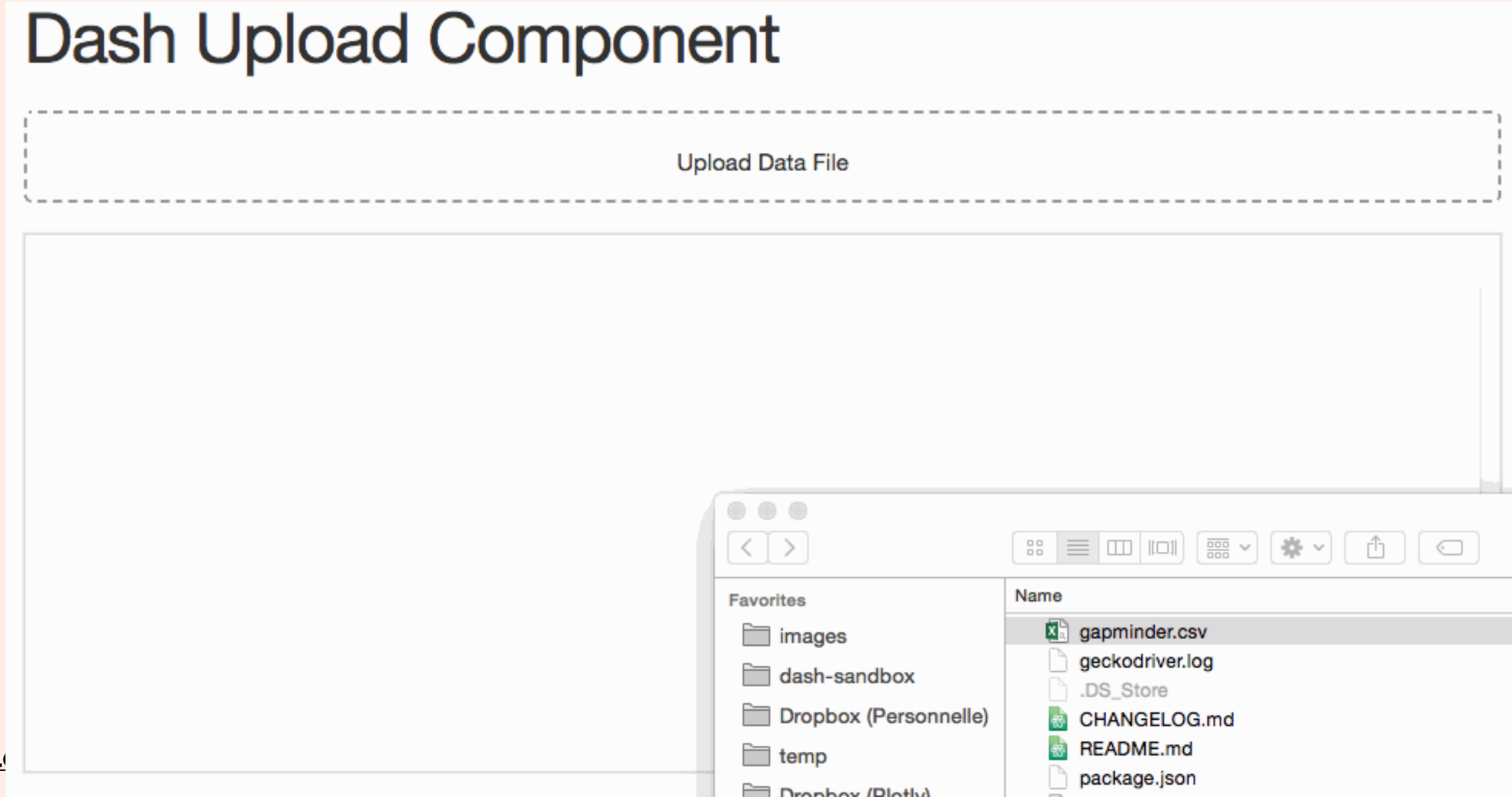
continent	country	gdpPercap	lifeExp	pop	year
Asia	Afghanistan	974.5803384	43.828	31889923	2007
Europe	Albania	5937.02952599999	76.423	3600523	2007
Africa	Algeria	6223.367465	72.301	33333216	2007
Africa	Angola	4797.231267	42.731	12420476	2007
Americas	Argentina	12779.3796400000	75.32	40301927	2007
Oceania	Australia	34435.3674399999	81.235	20434176	2007
Europe	Austria	36126.4927	79.829	8199783	2007
Asia	Bahrain	29796.0483399999	75.635	708573	2007
Asia	Bangladesh	1391.253792	64.062	150448339	2007





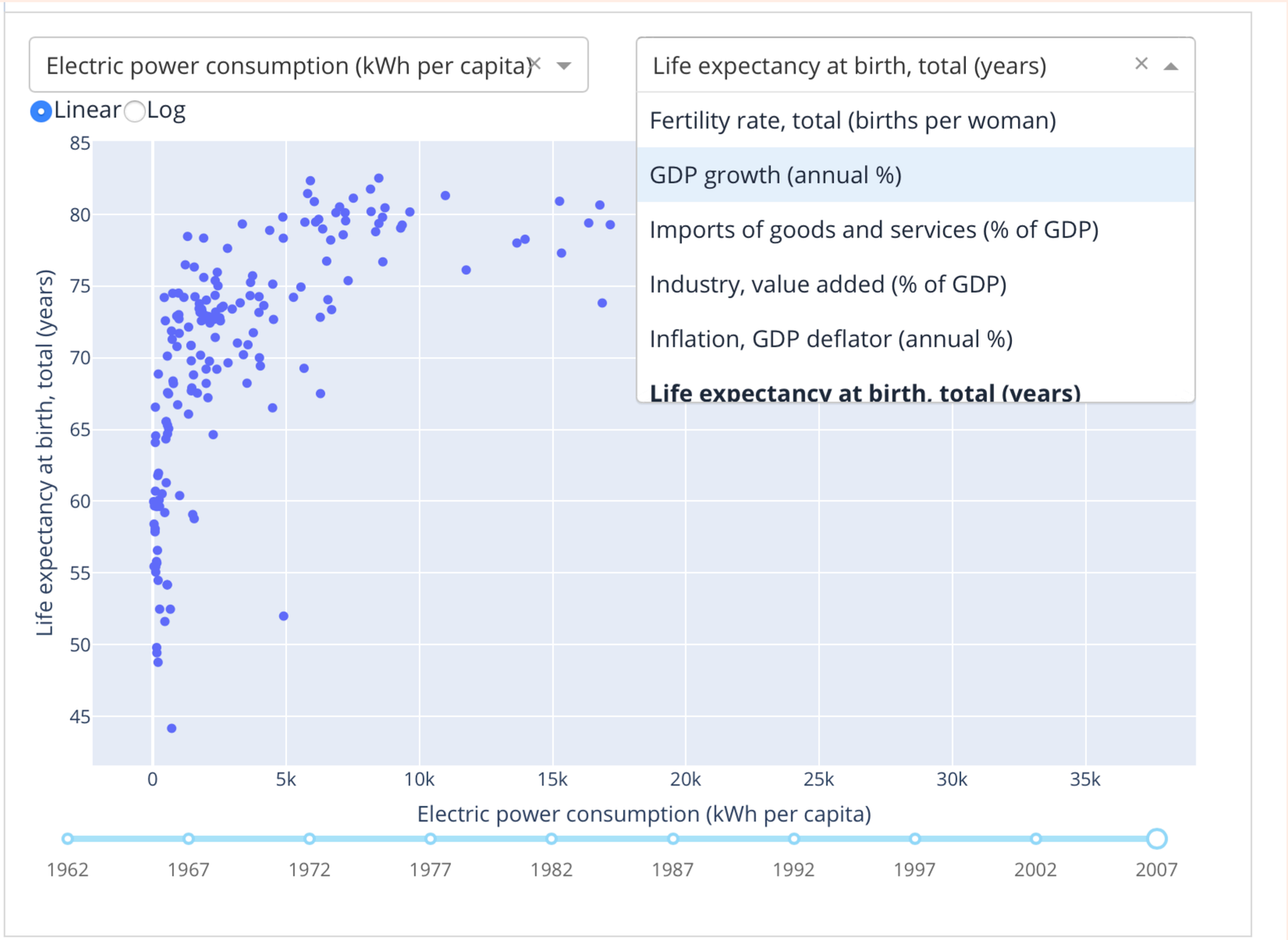
Upload

- dcc.Upload is another interesting component that allows users to upload custom files into your dashboard and then visualize/analyze it



Code: Dash Core Compon

Dash Callbacks



Source: Dash Callbacks Tutorial

Summary

- Dash Components are awesome, use them in your dashboard apps!

- App Callbacks will allow you to use the dash components above to interactively change your plots

- Interactivity within plots (zooming, panning, lines, annotations, highlights, linked plots can all be done with Altair, plotly, bokeh, ggplotly etc…)

Part 5: Customizing Dash apps

Positioning your components in Dash

`4.Layout (dash components)`

Two ways:

1) Use CSS
2) Use `dash-bootstrap-components (dbc)`

Positioning your components in Dash

Layout

Components for laying out your Dash app, including wrapping containers, and a powerful, responsive grid system.

Layout in Bootstrap is controlled using the grid system. The Bootstrap grid has twelve columns, and five responsive tiers (allowing you to specify different behaviours on different screen sizes, see below). The width of your columns can be specified in terms of how many of the twelve grid columns it should span, or you can allow the columns to expand or shrink to fit either their content or the available space in the row.

There are three main layout components in *dash-bootstrap-components*: `Container`, `Row`, and `Col`.

The `Container` component can be used to center and horizontally pad your app's content. The docs you are currently reading are themselves a Dash app built with *dash-bootstrap-components*. The content on this page has been centered by wrapping it in a `Container` component. By default the container has a responsive pixel width. Use the keyword argument `fluid=True` if you want your `Container` to fill available horizontal space and resize fluidly.

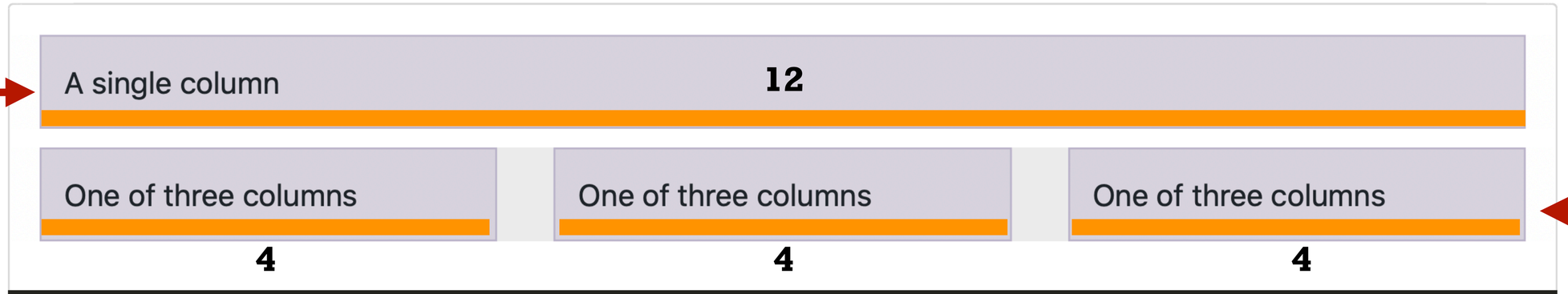
The `Row` component is a wrapper for columns. The layout of your app should be built as a series of rows of columns.

When using the grid layout, content should be placed in columns, and only `Col` components should be immediate children of `Row`.

For much more detail on the Bootstrap grid system, see the Bootstrap documentation.

Source: Bootstrap Docs

Positioning your components in Dash




```
import dash_bootstrap_components as dbc
import dash_html_components as html

row = html.Div(
    [
        dbc.Row(dbc.Col(html.Div("A single column"))),
        dbc.Row(
            [
                dbc.Col(html.Div("One of three columns")),
                dbc.Col(html.Div("One of three columns")),
                dbc.Col(html.Div("One of three columns")),
            ]
        ),
    ]
)
```

Source: Bootstrap Docs

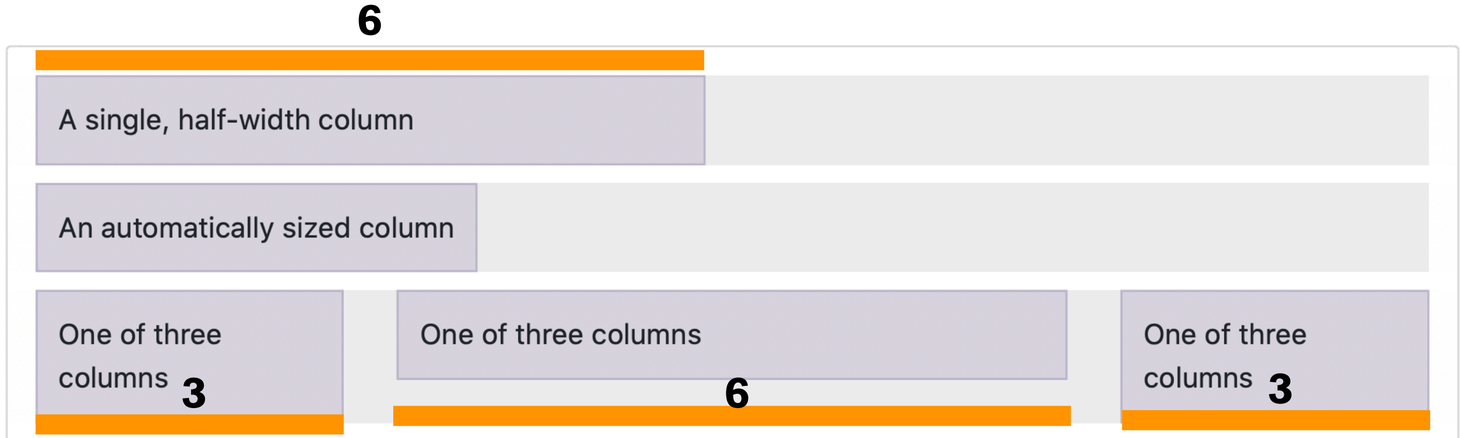
Positioning your components in Dash

Source: Bootstrap Docs

Specify width

Specify the desired width of each column using the `width` keyword argument. The basic options are:

- `True`, the default, column will expand to fill the available space.
- `"auto"`, column will be sized according to the natural width of its content.
- An integer `1`,...,`12`. Column will span this many of the 12 grid columns. For a half width column, set `width=6`, for a third width column, set `width=4`, and so on.




```
import dash_bootstrap_components as dbc
import dash_html_components as html

row = html.Div(
    [
        dbc.Row(dbc.Col(html.Div("A single, half-width column"), width=6)),
        dbc.Row(
            dbc.Col(html.Div("An automatically sized column"), width="auto")
        ),
        dbc.Row(
            [
                dbc.Col(html.Div("One of three columns"), width=3),
                dbc.Col(html.Div("One of three columns")),
                dbc.Col(html.Div("One of three columns"), width=3),
            ]
        ),
    ]
```

Positioning your components in Dash

Row without 'gutters'

By default, horizontal spacing is added between the columns. Use `no_gutters=True` to disable this.

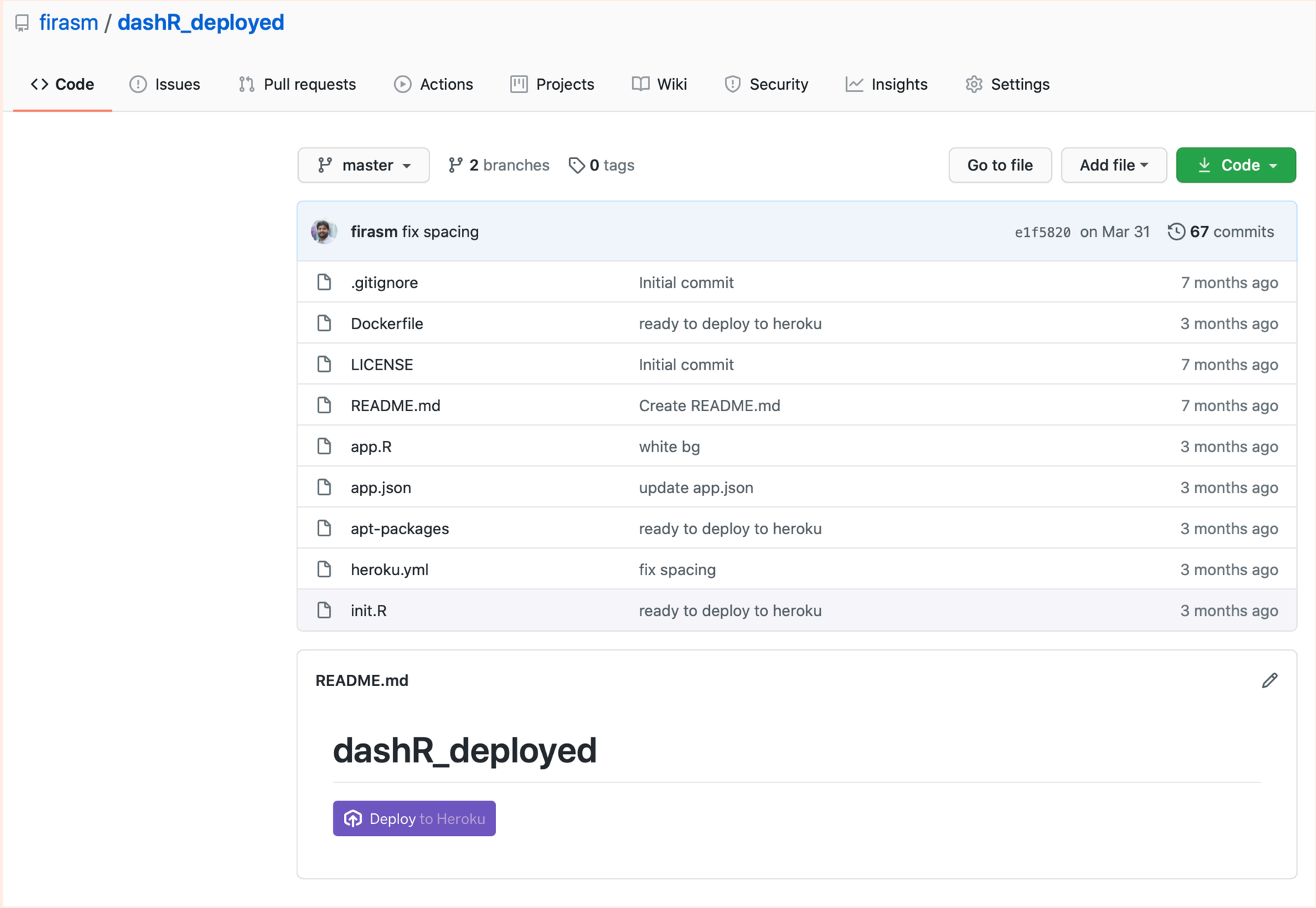

```
import dash_bootstrap_components as dbc
import dash_html_components as html

row = dbc.Row(
    [
        dbc.Col(html.Div("One of three columns")),
        dbc.Col(html.Div("One of three columns")),
        dbc.Col(html.Div("One of three columns")),
    ],
    no_gutters=True,
)
```

Source: Bootstrap Docs

Part 6: Deploying Dash apps

Deploy

One-click deploy to Heroku!



https://dashboard.heroku.com/apps/firas-12112