

Interactive Data Analysis with FigureWidget ipywidgets in Python

Interactive Data Analysis with Plotly

Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. [Sign up for early access now.](https://plotly.com/studio/?utm_medium=graphing-libraries&utm_campaign=studio_early_access&utm_content=sidebar) (https://plotly.com/studio/?utm_medium=graphing-libraries&utm_campaign=studio_early_access&utm_content=sidebar)

NYC Flights Database

```
import datetime
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from ipywidgets import widgets
```

We'll be making an application to take a look at delays from all flights out of NYC in the year 2013.

```
df = pd.read_csv(
    'https://raw.githubusercontent.com/yankev/testing/master/datasets/nycflights.csv')
df = df.drop(df.columns[[0]], axis=1)
```

```
df.sample(3)
```

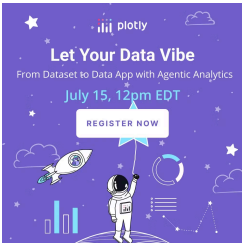
	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
62447	2013	11	8	639.0	2.0	918.0	-11.0	UA	N429UA	240	EWB	MCO	135.0	937	6.0	39.0
170353	2013	4	6	1318.0	19.0	1452.0	21.0	UA	N493UA	308	EWB	ORD	125.0	719	13.0	18.0
56303	2013	11	1	1245.0	0.0	1555.0	-5.0	UA	N37462	1289	EWB	SFO	349.0	2565	12.0	45.0

Let's get the set of all the airlines, so that we can type the right things into the search box later.

```
df['carrier'].unique()

array(['UA', 'AA', 'B6', 'DL', 'EV', 'MQ', 'US', 'WN', 'VX', 'FL', 'AS',
      '9E', 'F9', 'HA', 'YV', 'OO'], dtype=object)
```

Let's assign the widgets that we're going to be using in our app. In general all these widgets will be used to filter the data set, and thus what we visualize.



```

month = widgets.IntSlider(
    value=1.0,
    min=1.0,
    max=12.0,
    step=1.0,
    description='Month:',
    continuous_update=False
)

use_date = widgets.Checkbox(
    description='Date: ',
    value=True,
)

container = widgets.HBox(children=[use_date, month])

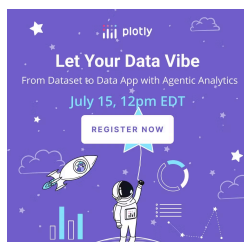
textbox = widgets.Dropdown(
    description='Airline: ',
    value='DL',
    options=df['carrier'].unique().tolist()
)

origin = widgets.Dropdown(
    options=list(df['origin'].unique()),
    value='LGA',
    description='Origin Airport:',
)

# Assign an empty figure widget with two traces
trace1 = go.Histogram(x=df['arr_delay'], opacity=0.75, name='Arrival Delays')
trace2 = go.Histogram(x=df['dep_delay'], opacity=0.75, name='Departure Delays')
g = go.FigureWidget(data=[trace1, trace2],
                    layout=go.Layout(
                        title=dict(
                            text='NYC FlightDatabase'
                        ),
                        barmode='overlay'
                    ))

```

Let now write a function that will handle the input from the widgets, and alter the state of the graph.



```
def validate():
    if origin.value in df['origin'].unique() and textbox.value in df['carrier'].unique():
        return True
    else:
        return False

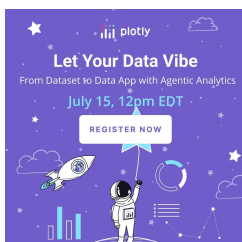
def response(change):
    if validate():
        if use_date.value:
            filter_list = [i and j and k for i, j, k in
                           zip(df['month'] == month.value, df['carrier'] == textbox.value,
                               df['origin'] == origin.value)]
            temp_df = df[filter_list]

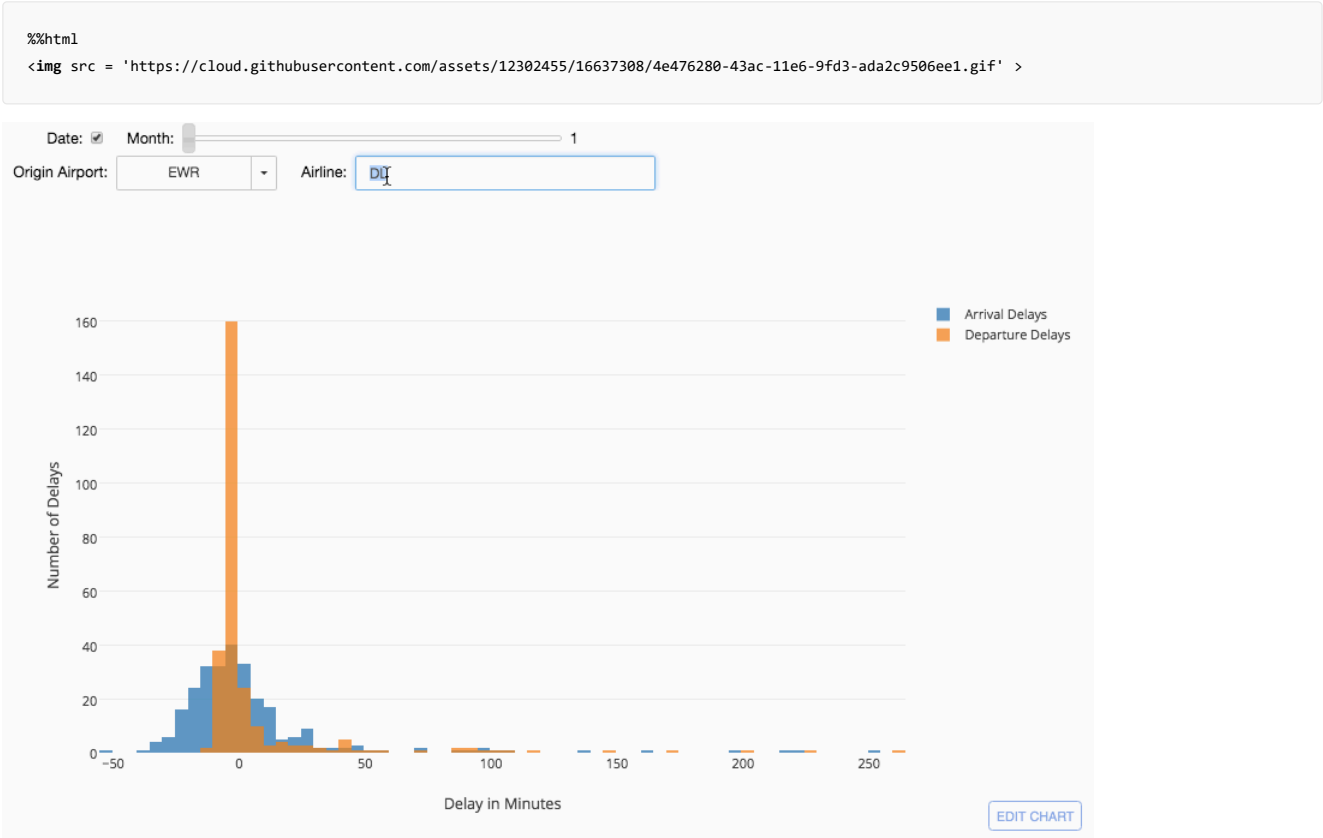
        else:
            filter_list = [i and j for i, j in
                           zip(df['carrier'] == 'DL', df['origin'] == origin.value)]
            temp_df = df[filter_list]
        x1 = temp_df['arr_delay']
        x2 = temp_df['dep_delay']
        with g.batch_update():
            g.data[0].x = x1
            g.data[1].x = x2
            g.layout.barmode = 'overlay'
            g.layout.xaxis.title = 'Delay in Minutes'
            g.layout.yaxis.title = 'Number of Delays'

    origin.observe(response, names="value")
    textbox.observe(response, names="value")
    month.observe(response, names="value")
    use_date.observe(response, names="value")
```

Time to try the app out!!

```
container2 = widgets.HBox([origin, textbox])
widgets.VBox([container,
              container2,
              g])
```





What About Dash?

Dash (<https://dash.plot.ly/>) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at <https://dash.plot.ly/installation> (<https://dash.plot.ly/installation>).


Everywhere in this page that you see `fig.show()`, you can display the same figure in a Dash application by passing it to the `figure` argument of the `Graph` component (<https://dash.plot.ly/dash-core-components/graph>) from the built-in `dash_core_components` package like this:

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False) # Turn off reloader if inside Jupyter
```



Dash your way to interactive web apps.

No JavaScript required!

GET STARTED NOW


My First App with Data, Graph, and Controls

pop

lifeExp

gdpPerCap

country	pop	continent	lifeExp	gdpPerCap
Afghanistan	31889923	Asia	43.828	974.5883384
Albania	3600523	Europe	76.423	5937.829525999999
Algeria	33333216	Africa	72.381	6223.367465
Angola	12420476	Africa	42.731	4707.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	706573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1761.253792
Belgium	10391226	Europe	79.441	33962.04968
Benin	8878314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3821.137884



(https://dash.plotly.com/tutorial?utm_medium=graphing_libraries&utm_content=python_footer)

JOIN OUR MAILING LIST

Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!

SUBSCRIBE
(<https://go.plot.ly/subscription>)

About Us

Careers (<https://plotly.com/careers>)
Resources (<https://plotly.com/resources/>)
Blog (<https://medium.com/@plotlygraphs>)

Products

Dash (<https://plotly.com/dash/>)
Consulting and Training
(<https://plotly.com/consulting-and-oem/>)

Support

Community Support (<https://community.plot.ly/>)
Documentation (<https://plotly.com/graphing-libraries>)

Pricing

Enterprise Pricing (<https://plotly.com/get-pricing/>)

