Tile Choropleth Maps in Python

How to make tile choropleth maps in Python with Plotly.

s and

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)

A Choropleth Map (https://en.wikipedia.org/wiki/Choropleth_map) is a map composed of colored polygons. It is used to represent spatial variations of a quantity. This page documents how to build **tile-map** choropleth maps, but you can also build **outline** choropleth maps (/python/choropleth-maps).

bjects

Below we show how to create Choropleth Maps using either Plotly Express' px.choropleth_map function or the lower-level go.Choroplethmap graph object.

Introduction: main parameters for choropleth tile maps

Making choropleth maps requires two main types of input:

1. GeoJSON-formatted geometry information where each feature has either an id field or some identifying value in properties.
2. A list of values indexed by feature identifier.

The GeoJSON data is passed to the geojson argument, and the data is passed into the color argument of px.choropleth_map (z if using graph_objects), in the same order as the IDs are passed into the location argument.

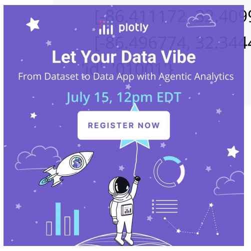
Note the geojson attribute can also be the URL to a GeoJSON file, which can speed up map rendering in certain cases.

GeoJSON with feature.id

Here we load a GeoJSON file containing the geometry information for US counties, where feature.id is a FIPS code (https://en.wikipedia.org/wiki/FIPS_county_code).

```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

counties["features"][0]
```

```
{'type': 'Feature',
 'properties': {'GEO_ID': '0500000US01001',
  'STATE': '01',
  'COUNTY': '001',
  'NAME': 'Autauga',
  'LSAD': 'County',
  'CENSUSAREA': 594.436},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[-86.496774, 32.344437],
    [-86.717897, 32.402814],
    [-86.814912, 32.340803],
    [-86.890581, 32.502974],
    [-86.917595, 32.664169],
    [-86.71339, 32.661732],
    [-86.714219, 32.705694],
    [-86.413116, 32.707386],
    [-86.411172, 32.409937],
    [-86.496774, 32.344437]]]},
 'id': '01001'}
```



Data indexed by id

Here we load unemployment data by county, also indexed by FIPS code (https://en.wikipedia.org/wiki/FIPS_county_code).

```
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16.csv",
                   dtype={"fips": str})
df.head()
```

	fips	unemp
0	01001	5.3
1	01003	5.4
2	01005	8.6
3	01007	6.6
4	01009	5.5

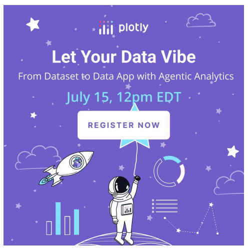
s and

ɔbjects

Choropleth map using plotly.express and carto base map

Plotly Express (/python/plotly-express/) is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data (/python/px-arguments/) and produces easy-to-style figures (/python/styling-plotly-express/).

With px.choropleth_map, each row of the DataFrame is represented as a region of the choropleth.



s and

›bjects

```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16.csv",
                   dtype={"fips": str})

import plotly.express as px

fig = px.choropleth_map(df, geojson=counties, locations='fips', color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                           map_style="carto-positron",
                           zoom=3, center = {"lat": 37.0902, "lon": -95.7129},
                           opacity=0.5,
                           labels={'unemp':'unemployment rate'}
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

unemployment rate
12
10
8
6
4
2
0
UNITED STATES
MEXICO
THE BAHAMAS
CUBA
DOMINICAN REPUBLIC
JAMAICA
BELIZE

Choropleth maps in Dash

Dash (https://plotly.com/dash/) is the best way to build analytical apps in Python using Plotly figures. To run the app below, run pip install dash, click "Download" to get the code and run python app.py.

Get started with the official Dash docs (https://dash.plotly.com/installation) and **learn how to effortlessly** style (https://plotly.com/dash/design-kit/) **&** deploy (https://plotly.com/dash/app-manager/) **apps like this with** Dash Enterprise (https://plotly.com/dash/)**.**

s and

bjects

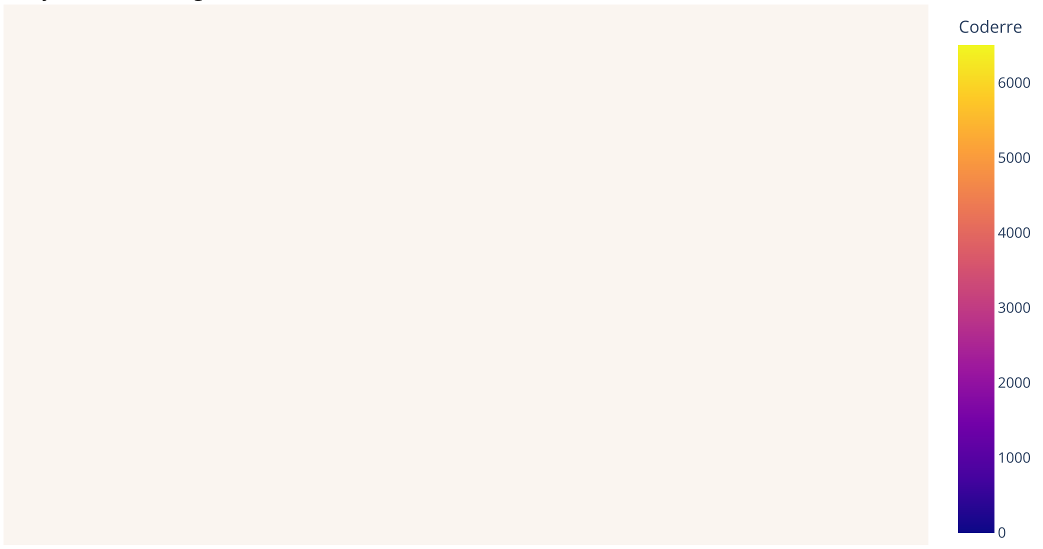
DOWNLOAD

```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px


app = Dash(__name__)

app.layout = html.Div([
    html.H4('Polotical candidate voting pool analysis'),
    html.P("Select a candidate:"),
    dcc.RadioItems(
        id='candidate',
        options=["Joly", "Coderre", "Bergeron"],
        value="Coderre",
        inline=True
    ),
    dcc.Graph(id="graph"),
])


@app.callback(
    Output("graph", "figure"),
    Input("candidate", "value"))
def display_choropleth(candidate):
    df = px.data.election() # replace with your own data source
    geojson = px.data.election_geojson()
```

Polotical candidate voting pool analysis

Select a candidate:

○Joly [✓ ●Coderre] ○Bergeron



▼

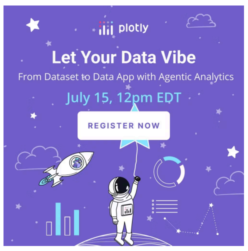
© CARTO, © OpenStreetMap contributors

Sign up for Dash Club → Free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. Join now (https://go.plotly.com/dash-club?utm_source=Dash+Club+2022&utm_medium=graphing_libraries&utm_content=inline).

Indexing by GeoJSON Properties

If the GeoJSON you are using either does not have an id field or you wish you use one of the keys in the properties field, you may use the featureidkey parameter to specify where to match the values of locations.

In the following GeoJSON object/data-file pairing, the values of properties.district match the values of the district column:



```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

print(df["district"][2])
print(geojson["features"][0]["properties"])
```

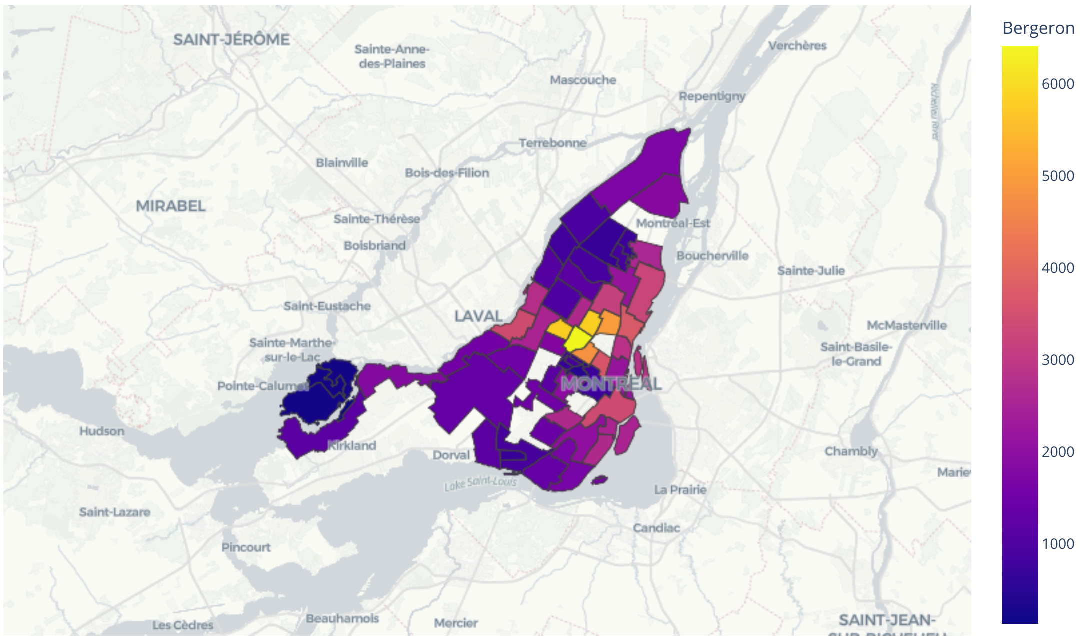
```
11-Sault-au-Récollet
{'district': '11-Sault-au-Récollet'}
```

s and

To use them together, we set locations to district and featureidkey to "properties.district". The color is set to the number of votes by the candidate named Bergeron.

```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_map(df, geojson=geojson, color="Bergeron",
                           locations="district", featureidkey="properties.district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           map_style="carto-positron", zoom=9)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

bjects



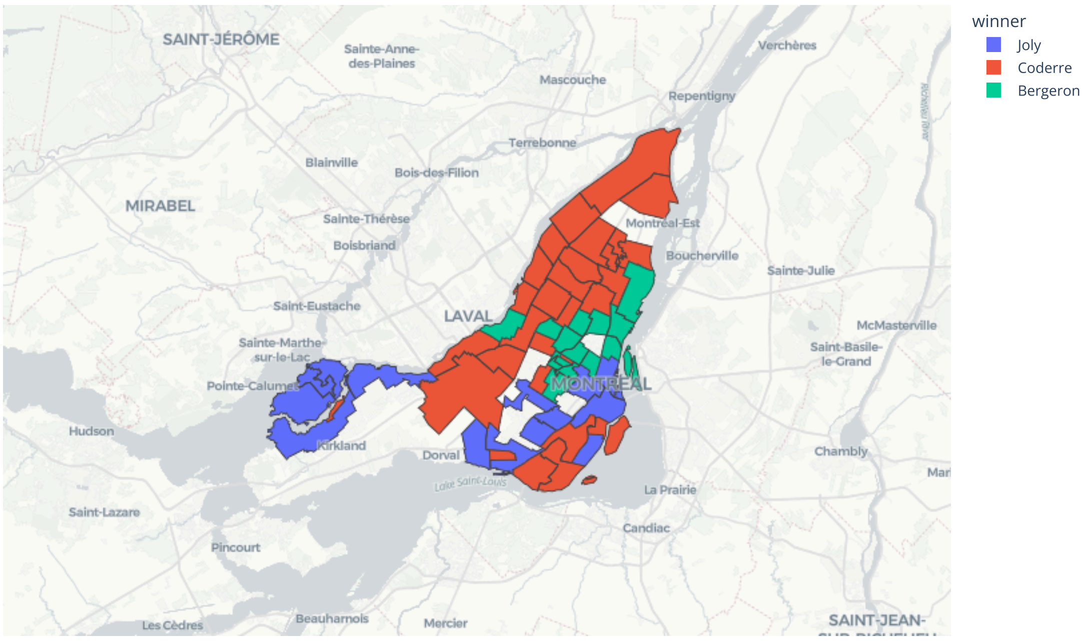
Discrete Colors

In addition to continuous colors (/python/colorscales/), we can discretely-color (/python/discrete-color/) our choropleth maps by setting color to a non-numerical column, like the name of the winner of an election.

```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_map(df, geojson=geojson, color="winner",
                           locations="district", featureidkey="properties.district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           map_style="carto-positron", zoom=9)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

s and

›bjects



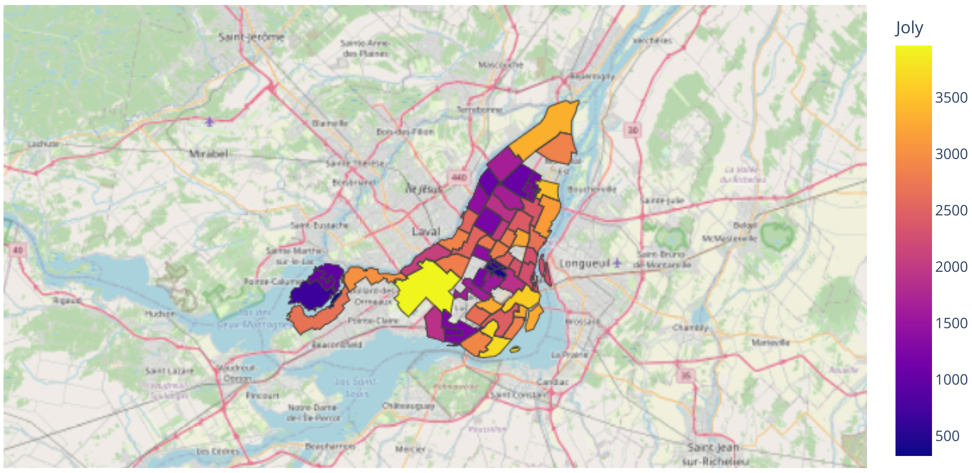
Using GeoPandas Data Frames

px.choropleth_map accepts the geometry of a GeoPandas (https://geopandas.org/) data frame as the input to geojson if the geometry contains polygons.

```
import plotly.express as px
import geopandas as gpd

df = px.data.election()
geo_df = gpd.GeoDataFrame.from_features(
    px.data.election_geojson()["features"]
).merge(df, on="district").set_index("district")

fig = px.choropleth_map(geo_df,
                           geojson=geo_df.geometry,
                           locations=geo_df.index,
                           color="Joly",
                           center={"lat": 45.5517, "lon": -73.7073},
                           map_style="open-street-map",
                           zoom=8.5)
fig.show()
```

s and

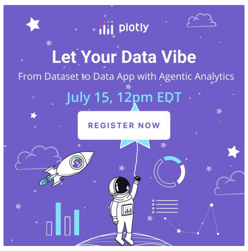
›bjects



© OpenStreetMap (https://www.openstreetmap.org/copyright) contributors

Choropleth map using plotly.graph_objects and carto base map

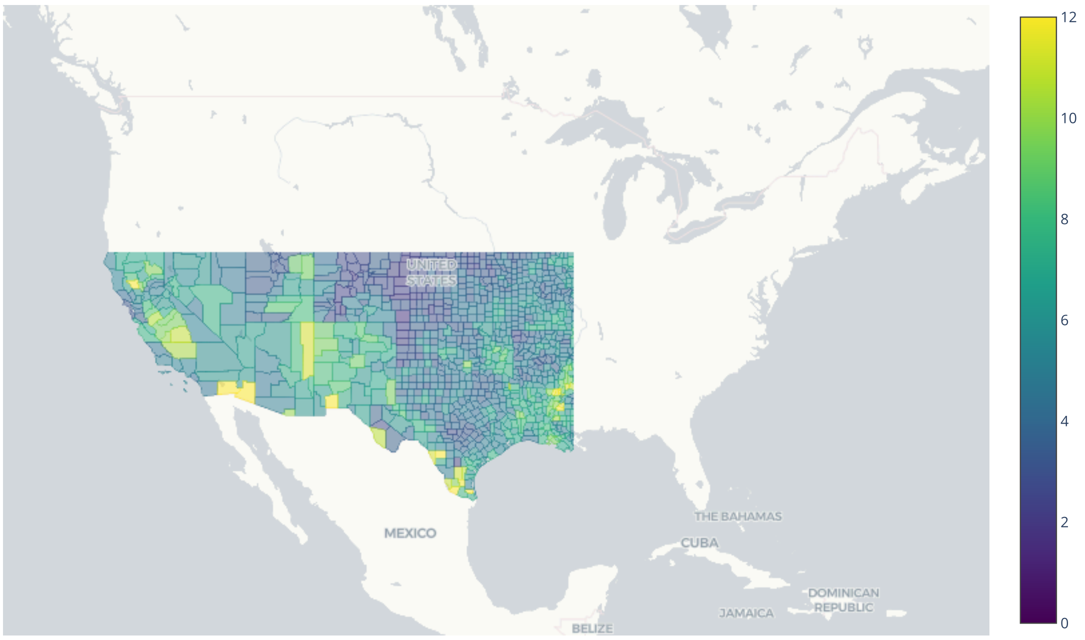
If Plotly Express does not provide a good starting point, it is also possible to use the more generic go.Choroplethmap class from plotly.graph_objects (/python/graph-objects/).



s and

ɔbjects

```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16.csv",
                   dtype={"fips": str})

import plotly.graph_objects as go

fig = go.Figure(go.Choroplethmap(geojson=counties, locations=df.fips, z=df.unemp,
                                    colorscale="Viridis", zmin=0, zmax=12,
                                    marker_opacity=0.5, marker_line_width=0))
fig.update_layout(map_style="carto-positron",
                  map_zoom=3, map_center = {"lat": 37.0902, "lon": -95.7129})
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



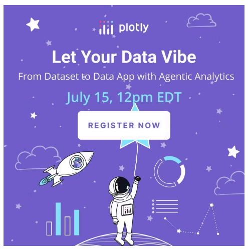
Mapbox Maps

Mapbox traces are deprecated and may be removed in a future version of Plotly.py.

The earlier examples using px.choropleth_map and go.Choroplethmap use Maplibre (https://maplibre.org/maplibre-gl-js/docs/) for rendering. These traces were introduced in Plotly.py 5.24 and are now the recommended way to create tile-based choropleth maps. There are also choropleth traces that use Mapbox (https://docs.mapbox.com): px.choropleth_mapbox and go.Choroplethmapbox

To plot on Mapbox maps with Plotly you *may* need a Mapbox account and a public Mapbox Access Token (https://www.mapbox.com/studio). See our Mapbox Map Layers (/python/mapbox-layers/) documentation for more information.

Here's an exmaple of using the Mapbox Light base map, which requires a free token.



```
token = open(".mapbox_token").read() # you will need your own token

from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as response:
    counties = json.load(response)

import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/fips-unemp-16.csv",
                   dtype={"fips": str})

import plotly.graph_objects as go

fig = go.Figure(go.Choroplethmapbox(geojson=counties, locations=df.fips, z=df.unemp,
                                    colorscale="Viridis", zmin=0, zmax=12, marker_line_width=0))
fig.update_layout(mapbox_style="light", mapbox_accesstoken=token,
                  mapbox_zoom=3, mapbox_center = {"lat": 37.0902, "lon": -95.7129})
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

s and

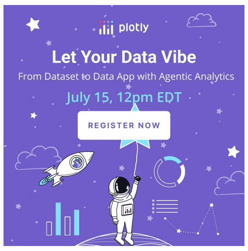
ɔbjects

```
/tmp/ipykernel_17706/3573937872.py:14: DeprecationWarning:

*choroplethmapbox* is deprecated! Use *choroplethmap* instead. Learn more at: https://plotly.com/python/mapbox-to-maplibre/
```

Reference

See function reference for px.choropleth_map (https://plotly.com/python-api-reference/generated/plotly.express.choropleth_map) or https://plotly.com/python/reference/choroplethmap/ (https://plotly.com/python/reference/choroplethmap/) for more information about the attributes available.

For (deprecated) Mapbox-based tile maps, see function reference for px.choropleth_mapbox (https://plotly.com/python-api-reference/generated/plotly.express.choropleth_mapbox) or https://plotly.com/python/reference/choroplethmapbox/ (https://plotly.com/python/reference/choroplethmapbox/).



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.

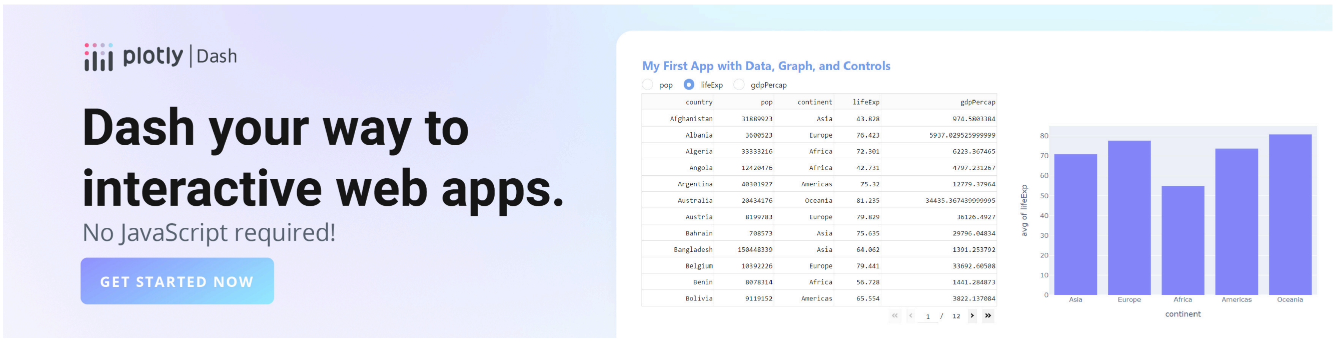
Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

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:

s and

```
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
```

objects



My First App with Data, Graph, and Controls

pop [✓ lifeExp] gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	708573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1391.253792
Belgium	10392226	Europe	79.441	33692.60508
Benin	8078314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3822.137084

1 / 12

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