Choropleth Maps in Python

How to make choropleth maps in Python with Plotly.

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 **outline** choropleth maps, but you can also build choropleth **tile maps** (/python/tile-county-choropleth).

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

ometries
h

Base Map Configuration

Plotly figures made with Plotly Express (/python/plotly-express/) px.scatter_geo, px.line_geo or px.choropleth functions or containing go.Choropleth or go.Scattergeo graph objects (/python/graph-objects/) have a go.layout.Geo object which can be used to control the appearance of the base map (/python/map-configuration/) onto which data is plotted.

Introduction: main parameters for choropleth outline maps

Making choropleth maps requires two main types of input:

1. Geometry information:
   A. This can either be a supplied GeoJSON file where each feature has either an id field or some identifying value in properties; or
   B. one of the built-in geometries within plotly: US states and world countries (see below)
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 (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.

Choropleth Map with plotly.express

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/).

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'}
```


ometries
h

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()
```

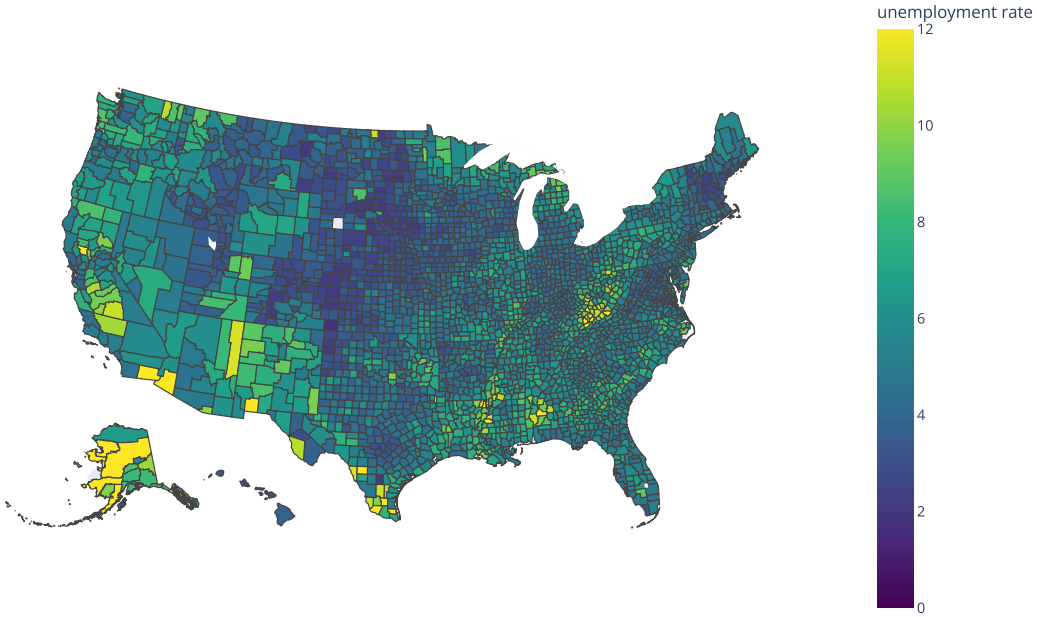
	fips	unemp
0	01001	5.3
1	01003	5.4
2	01005	8.6
3	01007	6.6
4	01009	5.5

Choropleth map using GeoJSON

Note In this example we set layout.geo.scope to usa to automatically configure the map to display USA-centric data in an appropriate projection. See the Geo map configuration documentation (/python/map-configuration/) for more information on scopes.

```
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(df, geojson=counties, locations='fips', color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                           scope="usa",
                           labels={'unemp':'unemployment rate'}
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

ometries
h



Indexing by GeoJSON Properties

If the GeoJSON you are using either does not have an id field or you wish to 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:

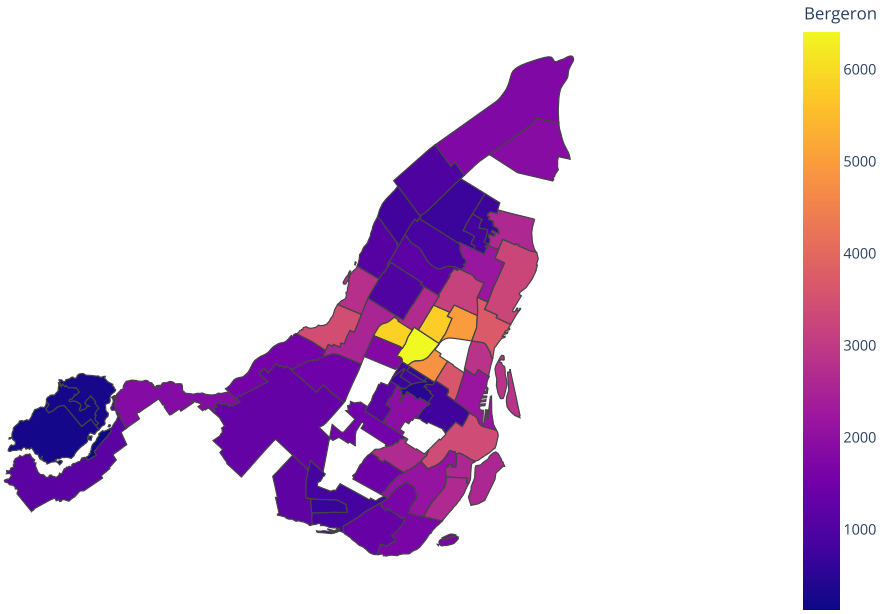
```
import plotly.express as px

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'}
```


e set locations to district and featureidkey to "properties.district". The color is set to the number of votes by the candidate named Bergeron.

set layout.geo.visible to False to hide the base map and frame, and we set layout.geo.fitbounds to 'locations' to automatically zoom the map nterest. See the Geo map configuration documentation (/python/map-configuration/) for more information on projections and bounds.

```
import plotly.express as px

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

fig = px.choropleth(df, geojson=geojson, color="Bergeron",
                    locations="district", featureidkey="properties.district",
                    projection="mercator"
                   )
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

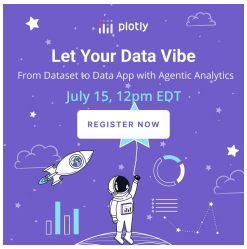


ɔmetries
:h

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.

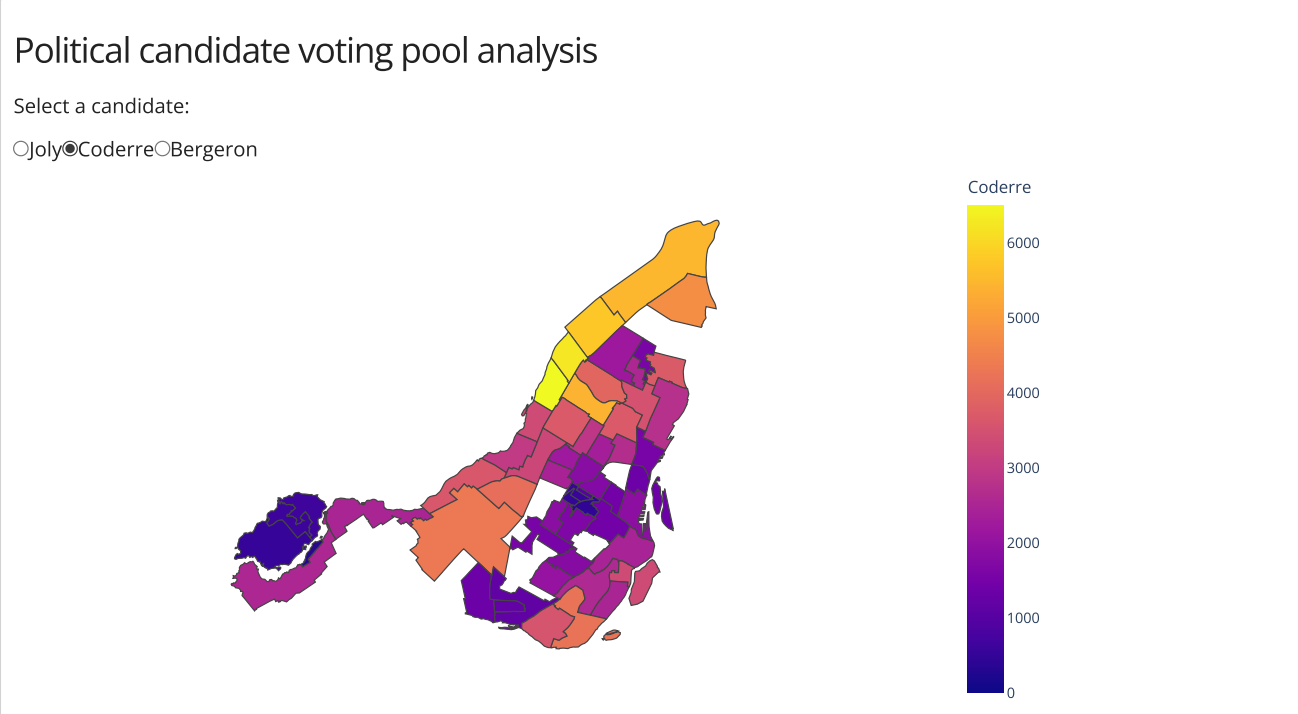
Get started with the official Dash docs (https://dash.plotly.com/installation) and **learn how to effortlessly** style (https://plotly.com/dash/design-kit/) **&** deploy (https://plotly.com/dash/app-manager/) **apps like this with** Dash Enterprise (https://plotly.com/dash/).



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

app = Dash(__name__)


app.layout = html.Div([
    html.H4('Political 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()
```

DOWNLOAD

ometries
h

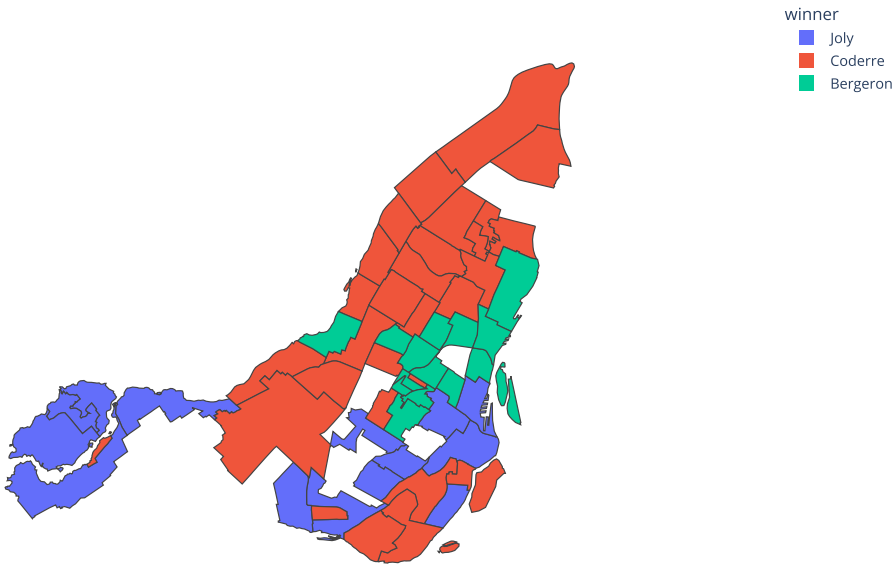


Discrete Colors

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.

Note In this example we set layout.geo.visible to False to hide the base map and frame, and we set layout.geo.fitbounds to 'locations' to automatically zoom the map to show just the area of interest. See the Geo map configuration documentation (/python/map-configuration/) for more information on projections and bounds.

```
import plotly.express as px

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

fig = px.choropleth(df, geojson=geojson, color="winner",
                    locations="district", featureidkey="properties.district",
                    projection="mercator", hover_data=["Bergeron", "Coderre", "Joly"]
                   )
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

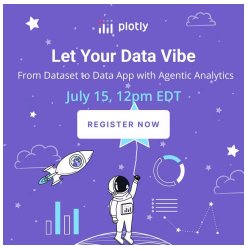


ometries

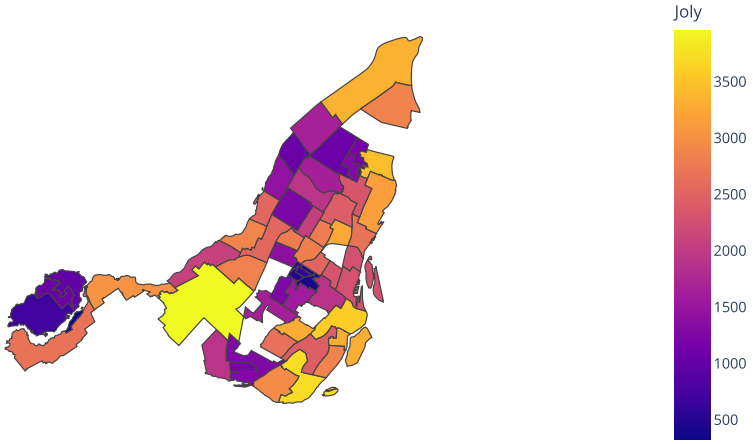
h

Using GeoPandas Data Frames

px.choropleth 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(geo_df,
                    geojson=geo_df.geometry,
                    locations=geo_df.index,
                    color="Joly",
                    projection="mercator")
fig.update_geos(fitbounds="locations", visible=False)
fig.show()
```



ometries
h

Using Built-in Country and State Geometries

Plotly comes with two built-in geometries which do not require an external GeoJSON file:

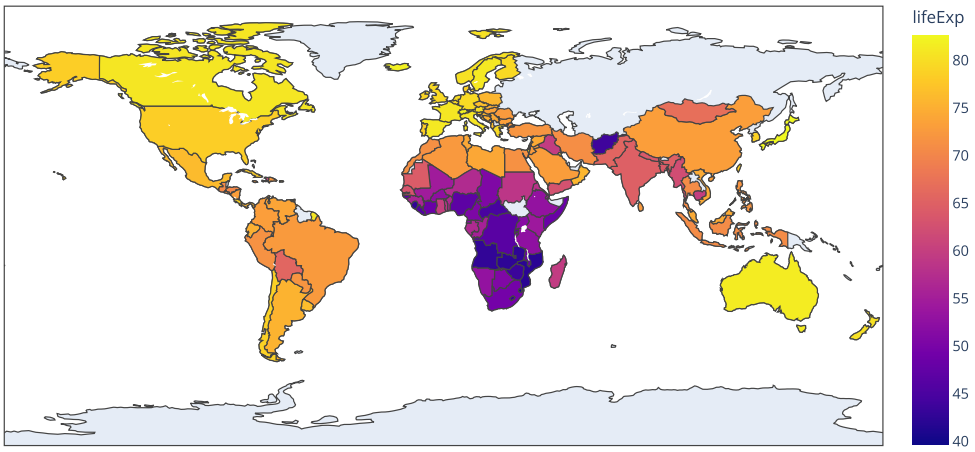
1. USA States
2. Countries as defined in the Natural Earth dataset.

Note and disclaimer: cultural (as opposed to physical) features are by definition subject to change, debate and dispute. Plotly includes data from Natural Earth "as-is" and defers to the Natural Earth policy regarding disputed borders (https://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/) which read:

> Natural Earth Vector draws boundaries of countries according to defacto status. We show who actually controls the situation on the ground.

To use the built-in countries geometry, provide locations as three-letter ISO country codes (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3).

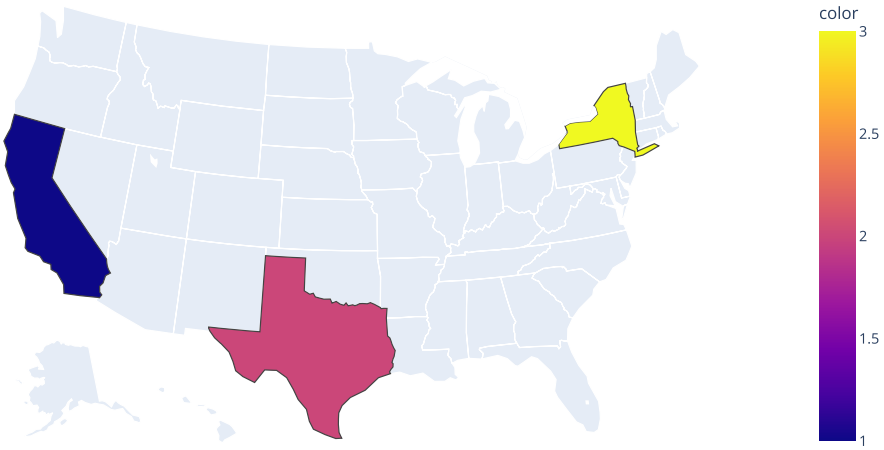
```
import plotly.express as px

df = px.data.gapminder().query("year==2007")
fig = px.choropleth(df, locations="iso_alpha",
                    color="lifeExp", # lifeExp is a column of gapminder
                    hover_name="country", # column to add to hover information
                    color_continuous_scale=px.colors.sequential.Plasma)
fig.show()
```



ometries
h

To use the USA States geometry, set locationmode='USA-states' and provide locations as two-letter state abbreviations:

```
import plotly.express as px

fig = px.choropleth(locations=["CA", "TX", "NY"], locationmode="USA-states", color=[1,2,3], scope="usa")
fig.show()
```



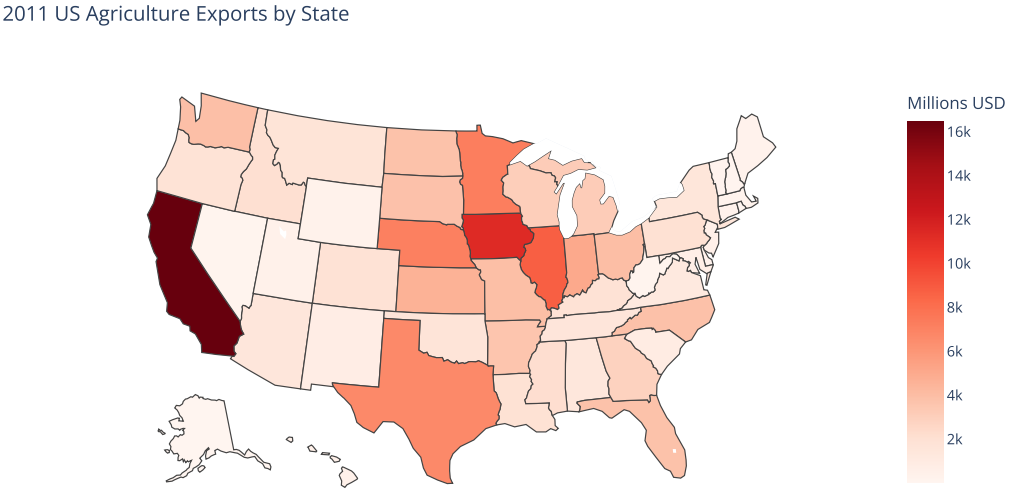
Maps with go.Choropleth

horopleth Map

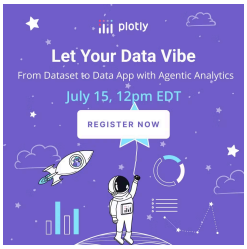
```
import plotly.graph_objects as go

import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2011_us_ag_exports.csv')

fig = go.Figure(data=go.Choropleth(
    locations=df['code'], # Spatial coordinates
    z = df['total exports'].astype(float), # Data to be color-coded
    locationmode = 'USA-states', # set of locations match entries in `locations`
    colorscale = 'Reds',
    colorbar_title = "Millions USD",
))

fig.update_layout(
    title_text = '2011 US Agriculture Exports by State',
    geo_scope='usa', # limite map scope to USA
)

fig.show()
```

ometries

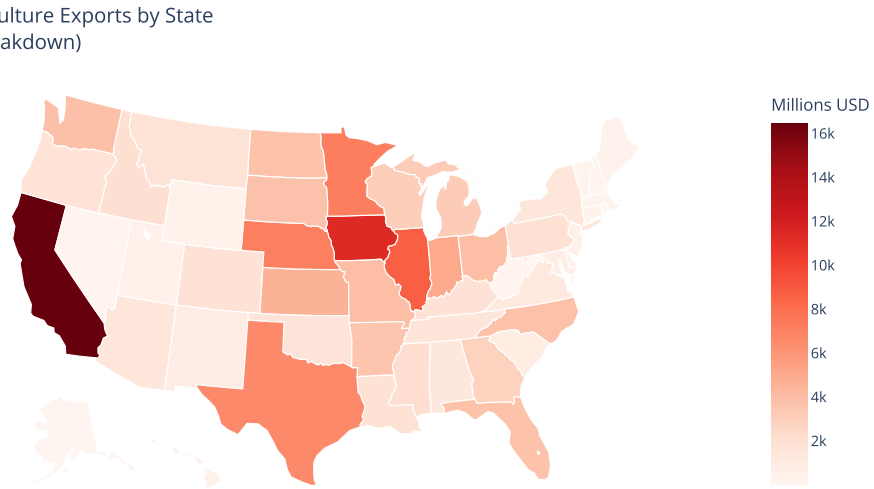
h



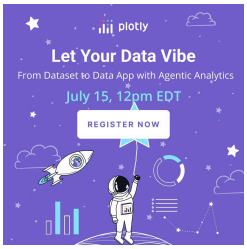
Customize choropleth chart



```
import plotly.graph_objects as go

import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2011_us_ag_exports.csv')

for col in df.columns:
    df[col] = df[col].astype(str)

df['text'] = df['state'] + '<br>' + \
    'Beef ' + df['beef'] + ' Dairy ' + df['dairy'] + '<br>' + \
    'Fruits ' + df['total fruits'] + ' Veggies ' + df['total veggies'] + '<br>' + \
    'Wheat ' + df['wheat'] + ' Corn ' + df['corn']

fig = go.Figure(data=go.Choropleth(
    locations=df['code'],
    z=df['total exports'].astype(float),
    locationmode='USA-states',
    colorscale='Reds',
    autocolorscale=False,
    text=df['text'], # hover text
    marker_line_color='white', # line markers between states
    colorbar=dict(
        title=dict(
            text="Millions USD"
            )
    )
))

fig.update_layout(
    title_text='2011 US Agriculture Exports by State<br>(Hover for breakdown)',
    geo = dict(
        scope='usa',
        projection=go.layout.geo.Projection(type = 'albers usa'),
        showlakes=True, # lakes
        lakecolor='rgb(255, 255, 255)'),
)

fig.show()
```

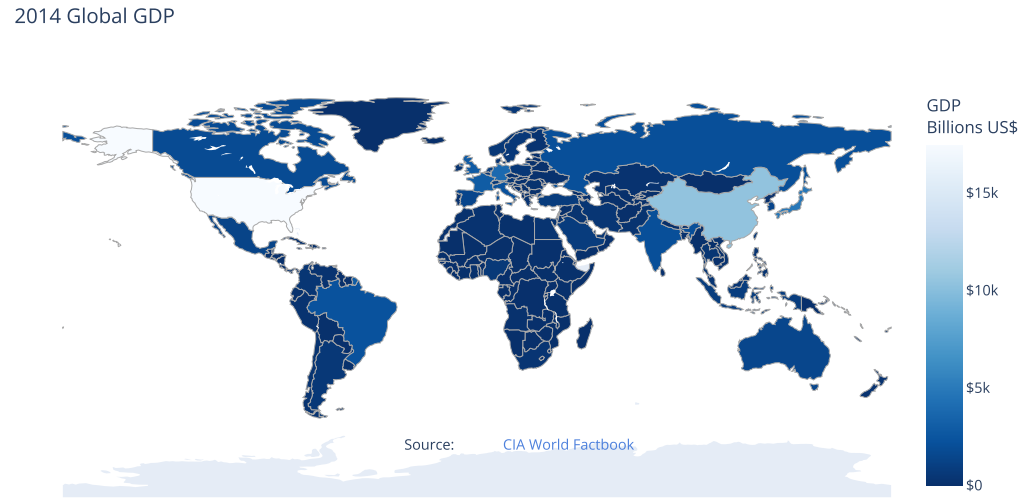


World Choropleth Map



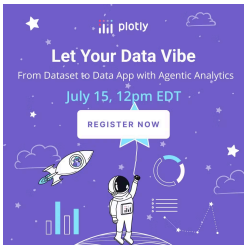
```
import plotly.graph_objects as go
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_world_gdp_with_codes.csv')

fig = go.Figure(data=go.Choropleth(
    locations = df['CODE'],
    z = df['GDP (BILLIONS)'],
    text = df['COUNTRY'],
    colorscale = 'Blues',
    autocolorscale=False,
    reversescale=True,
    marker_line_color='darkgray',
    marker_line_width=0.5,
    colorbar_tickprefix = '$',
    colorbar_title = 'GDP<br>Billions US$',
))

fig.update_layout(
    title_text='2014 Global GDP',
    geo=dict(
        showframe=False,
        showcoastlines=False,
        projection_type='equirectangular'
    ),
    annotations = [dict(
        x=0.55,
        y=0.1,
        xref='paper',
        yref='paper',
        text='Source: <a href="https://www.cia.gov/library/publications/the-world-factbook/fields/2195.html">\
            CIA World Factbook</a>',
        showarrow = False
    )]
)

fig.show()
```

ometries
h



County Choropleth Figure Factory

Plotly also includes a legacy "figure factory" for creating US county-level choropleth maps (/python/county-choropleth/).



```
import plotly.figure_factory as ff

import numpy as np
import pandas as pd

df_sample = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/laucnty16.csv')
df_sample['State FIPS Code'] = df_sample['State FIPS Code'].apply(lambda x: str(x).zfill(2))
df_sample['County FIPS Code'] = df_sample['County FIPS Code'].apply(lambda x: str(x).zfill(3))
df_sample['FIPS'] = df_sample['State FIPS Code'] + df_sample['County FIPS Code']

colorscale = ["#f7fbff", "#ebf3fb", "#deebf7", "#d2e3f3", "#c6dbef", "#b3d2e9", "#9ecae1",
    "#85bcdb", "#6baed6", "#57a0ce", "#4292c6", "#3082be", "#2171b5", "#1361a9",
    "#08519c", "#0b4083", "#08306b"
]
endpts = list(np.linspace(1, 12, len(colorscale) - 1))
fips = df_sample['FIPS'].tolist()
values = df_sample['Unemployment Rate (%)'].tolist()


fig = ff.create_choropleth(
    fips=fips, values=values, scope=['usa'],
    binning_endpoints=endpts, colorscale=colorscale,
    show_state_data=False,
    show_hover=True,
    asp = 2.9,
    title_text = 'USA by Unemployment %',
    legend_title = '% unemployed'
)
fig.layout.template = None
fig.show()
```

ometries

h

/home/circleci/project/doc/.venv/lib/python3.9/site-packages/plotly/figure_factory/_county_choropleth.py:808: ShapelyDeprecationWarning:

The 'type' attribute is deprecated, and will be removed in the future. You can use the 'geom_type' attribute instead.

/home/circleci/project/doc/.venv/lib/python3.9/site-packages/plotly/figure_factory/_county_choropleth.py:330: ShapelyDeprecationWarning:

The 'type' attribute is deprecated, and will be removed in the future. You can use the 'geom_type' attribute instead.

/home/circleci/project/doc/.venv/lib/python3.9/site-packages/plotly/figure_factory/_county_choropleth.py:357: ShapelyDeprecationWarning:

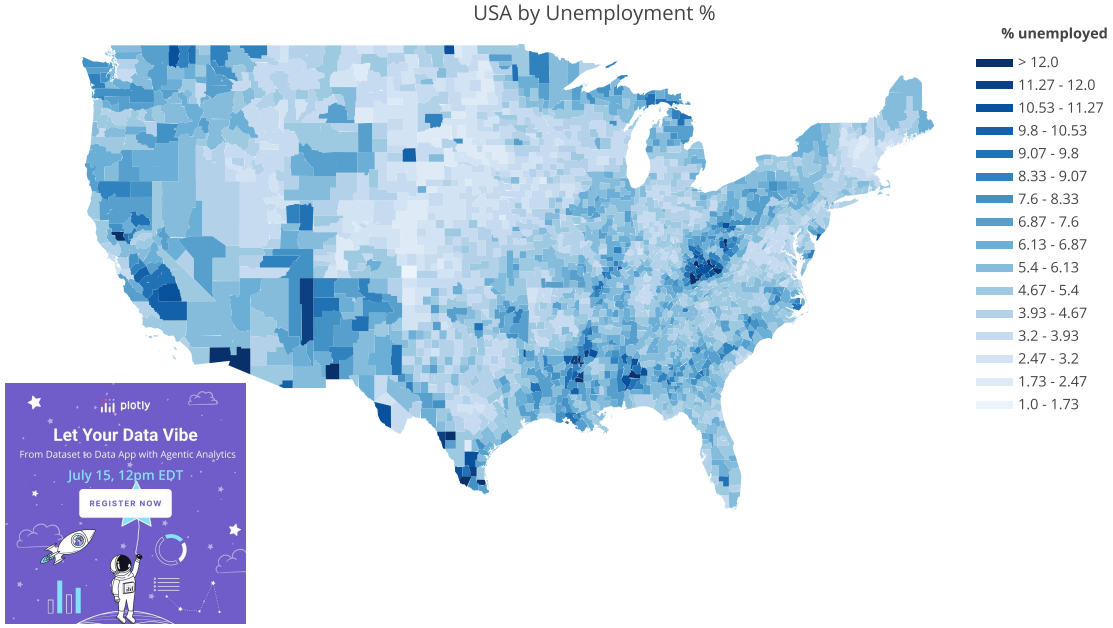
The 'type' attribute is deprecated, and will be removed in the future. You can use the 'geom_type' attribute instead.

/home/circleci/project/doc/.venv/lib/python3.9/site-packages/plotly/figure_factory/_county_choropleth.py:847: ShapelyDeprecationWarning:

The 'type' attribute is deprecated, and will be removed in the future. You can use the 'geom_type' attribute instead.

/home/circleci/project/doc/.venv/lib/python3.9/site-packages/plotly/figure_factory/_county_choropleth.py:852: ShapelyDeprecationWarning:

The 'type' attribute is deprecated, and will be removed in the future. You can use the 'geom_type' attribute instead.



Reference

See function reference for px.(choropleth) (https://plotly.com/python-api-reference/generated/plotly.express.choropleth) or https://plotly.com/python/reference/choropleth/ (https://plotly.com/python/reference/choropleth/) for more information and chart attribute options!

What About Dash?

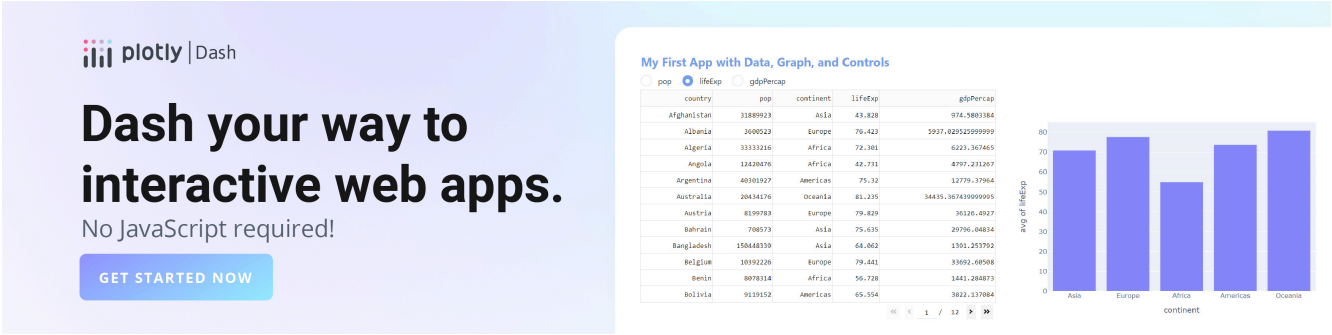
Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

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:

ometries
h

```
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
```



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