STAT3622 Data Visualization (with Python)

Lecture 9

Wenbin Du
The University of Hong Kong

29 March 2021

Mapbox

Mapbox is an American provider of custom online maps for websites and applications such as Foursquare, Lonely Planet, the Financial Times, The Weather Channel, Instacart Inc. and Snapchat.

Since 2010, it has rapidly expanded the niche of custom maps, as a response to the limited choice offered by map providers such as Google Maps.

https://en.wikipedia.org/wiki/Mapbox

go.Scattermapbox

Construct a new Scattermapbox object

The data visualized as scatter point, lines or marker symbols on a Mapbox GL geographic map is provided by longitude/latitude pairs in lon and lat.

Parameters

> lon – Sets the longitude coordinates (in degrees East)
>
> lat – Sets the latitude coordinates (in degrees North)
>
> …

go.Scattermapbox

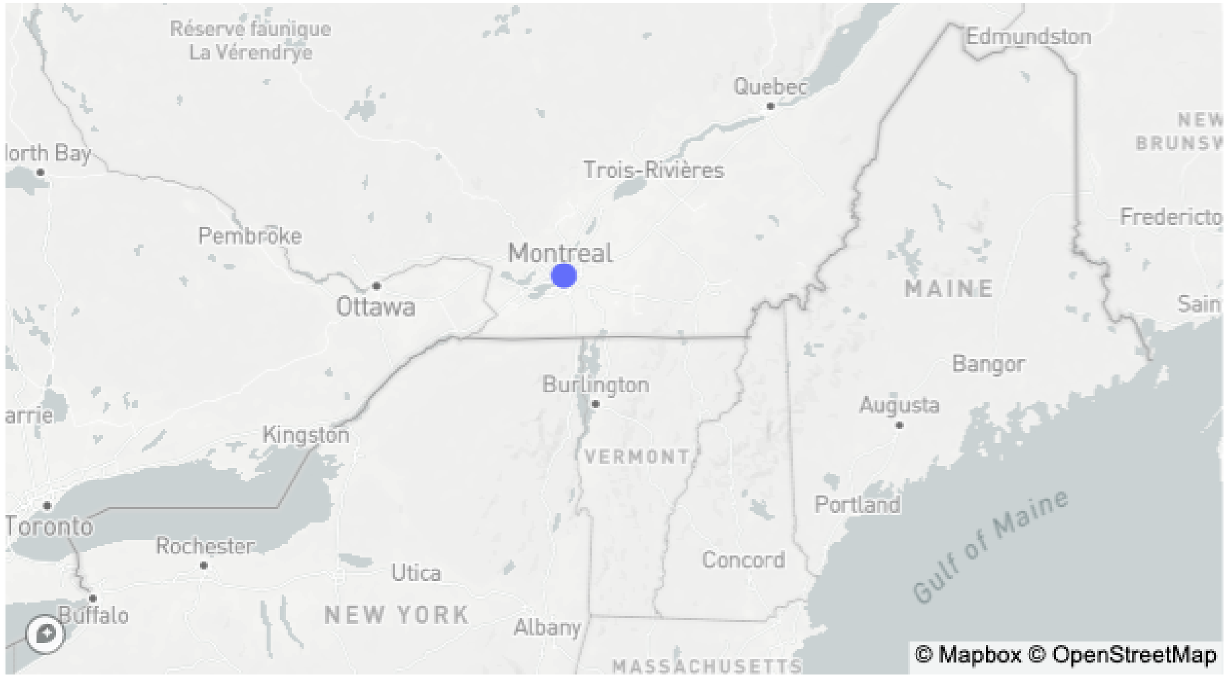
```
import plotly.graph_objects as go
token='pk.eyJ1Ijoic3l2aW5jZSIsImEiOiJjazZrNTcwY3kwMHBrM2txaGJqZWEzNWExIn0.tLQHY_OoiR2NMxnYHXUI
# To plot on Mapbox maps with Plotly you may need a Mapbox account and a public Mapbox Access
# https://www.mapbox.com/studio
fig = go.Figure(go.Scattermapbox(
        lat=['45.5017'],
        lon=['-73.5673'],
        mode='markers',
        marker=go.scattermapbox.Marker(size=14),
        text=['Montreal'],
    ))
fig.update_layout(
    hovermode='closest',
    mapbox=dict(
        accesstoken=token,
         center=go.layout.mapbox.Center(
            lat=45,lon=-73
        ),# The desired center.
         zoom=5, #The desired zoom level
        bearing=0,pitch=0,)
)
fig.show()
```

go.Scattermapbox

```python
import plotly.graph_objects as go
token='pk.eyJ1Ijoic3l2aW5jZSIsImEiOiJjazZrNTcwY3kwMHBrM2txaGJqZWEzNWExIn0.tLQHY_OoiR2NMxnYHXU
fig.update_layout(
    hovermode='closest',
    mapbox=dict(
        accesstoken=token,
        center=go.layout.mapbox.Center(
            lat=45,lon=-73
        ),# The desired center.
        zoom=5, #The desired zoom level
         bearing=0,
         pitch=0,)
)
```

> bearing(number): The desired bearing in degrees. The bearing is the compass direction that is "up". For example, bearing: 90 orients the map so that east is up.
> pitch(number): The desired pitch in degrees. The pitch is the angle towards the horizon measured in degrees with a range between 0 and 60 degrees. For example, pitch: 0 provides the appearance of looking straight down at the map, while pitch: 60

go.Scattermapbox



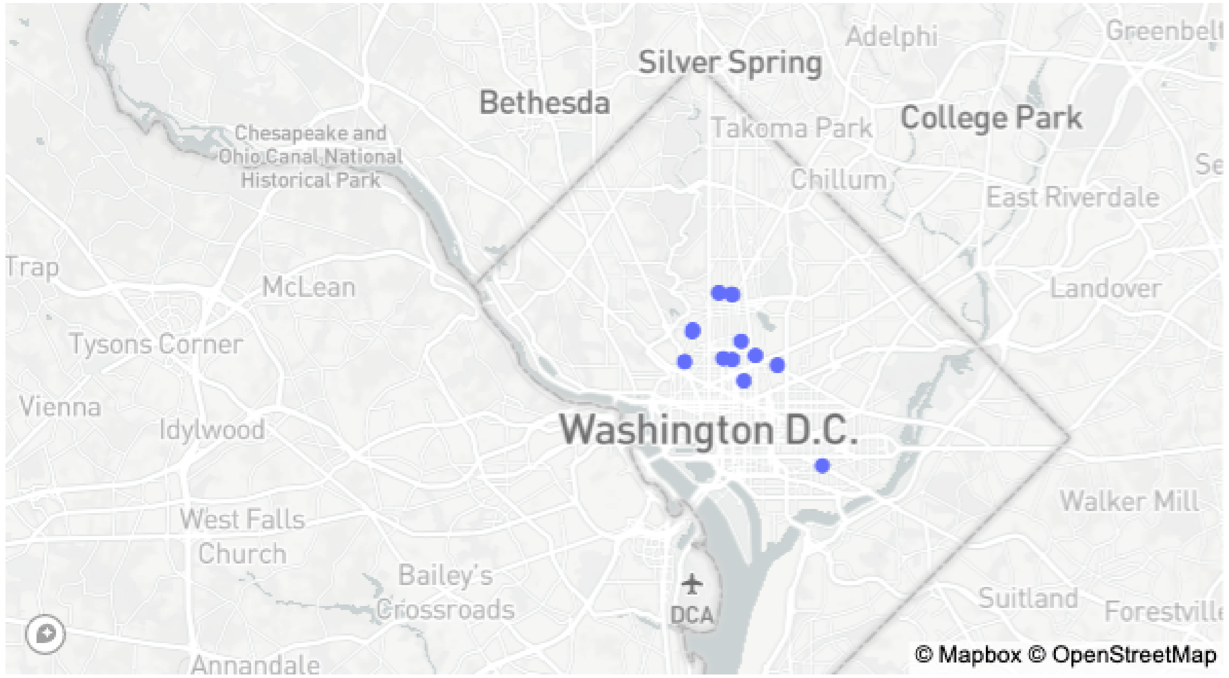
go.Scattermapbox

```python
import plotly.graph_objects as go
token='pk.eyJ1Ijoic3l2aW5jZSIsImEiOiJjazZrNTcwY3kwMHBrM2txaGJqZWEzNWExIn0.tLQHY_OoiR2NMxnYHXU
fig = go.Figure(go.Scattermapbox(
         lat=['38.91427','38.91538','38.91458', #multiple dots
             '38.92239','38.93222','38.90842',
             '38.91931','38.93260','38.91368',
             '38.88516','38.921894','38.93206',
             '38.91275'],
         lon=['-77.02827','-77.02013','-77.03155',
             '-77.04227','-77.02854','-77.02419',
             '-77.02518','-77.03304','-77.04509',
             '-76.99656','-77.042438','-77.02821',
             '-77.01239'],
        mode='markers',
        marker=go.scattermapbox.Marker(
            size=9
        ),
        text=["The coffee bar","Bistro Bohem","Black Cat",
             "Snap","Columbia Heights Coffee","Azi's Cafe",
             "Blind Dog Cafe","Le Caprice","Filter",
             "Peregrine","Tryst","The Coupe",
             "Big Bear Cafe"],
    ))
```

go.Scattermapbox

```
fig.update_layout(
    autosize=True,
    hovermode='closest',
    mapbox=dict(
        accesstoken=token,
        bearing=0,
        center=dict(
            lat=38.92,
            lon=-77.07
        ),
        pitch=0,
        zoom=10
    ),
)
fig.show()
```

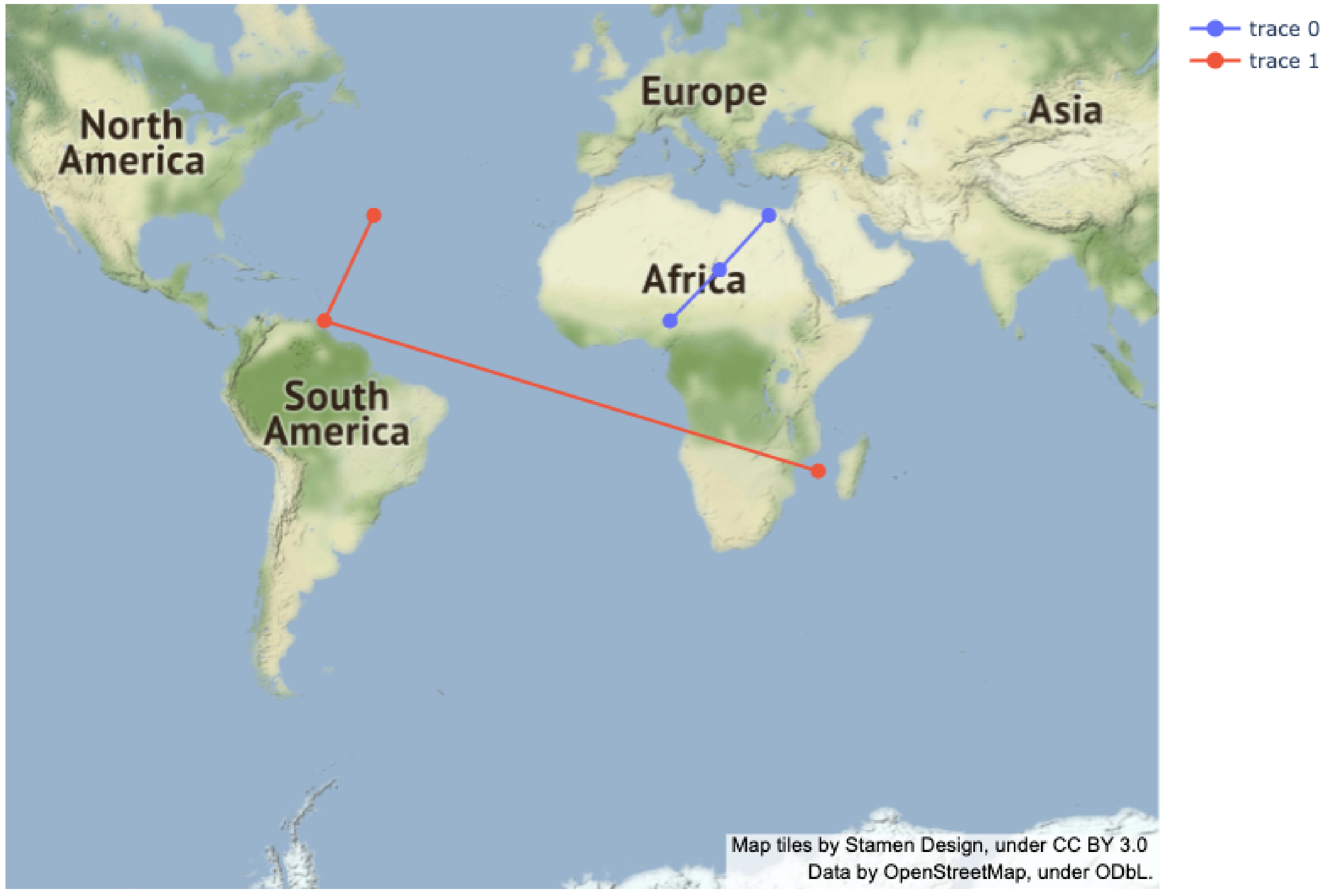
go.Scattermapbox



go.Scattermapbox

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattermapbox(
     mode = "markers+lines",
    lon = [10, 20, 30],
    lat = [10, 20,30],
    marker = {'size': 10}))

fig.add_trace(go.Scattermapbox(
     mode = "markers+lines",
    lon = [-50, -60,40],
    lat = [30, 10, -20],
    marker = {'size': 10}))

fig.update_layout(
    margin ={'l':0,'t':0,'b':0,'r':0},
    mapbox = {
        'center': {'lon': 10, 'lat': 10},
        'style': "stamen-terrain",
        'center': {'lon': -20, 'lat': -20},
        'zoom': 1})

fig.show()
```

go.Scattermapbox

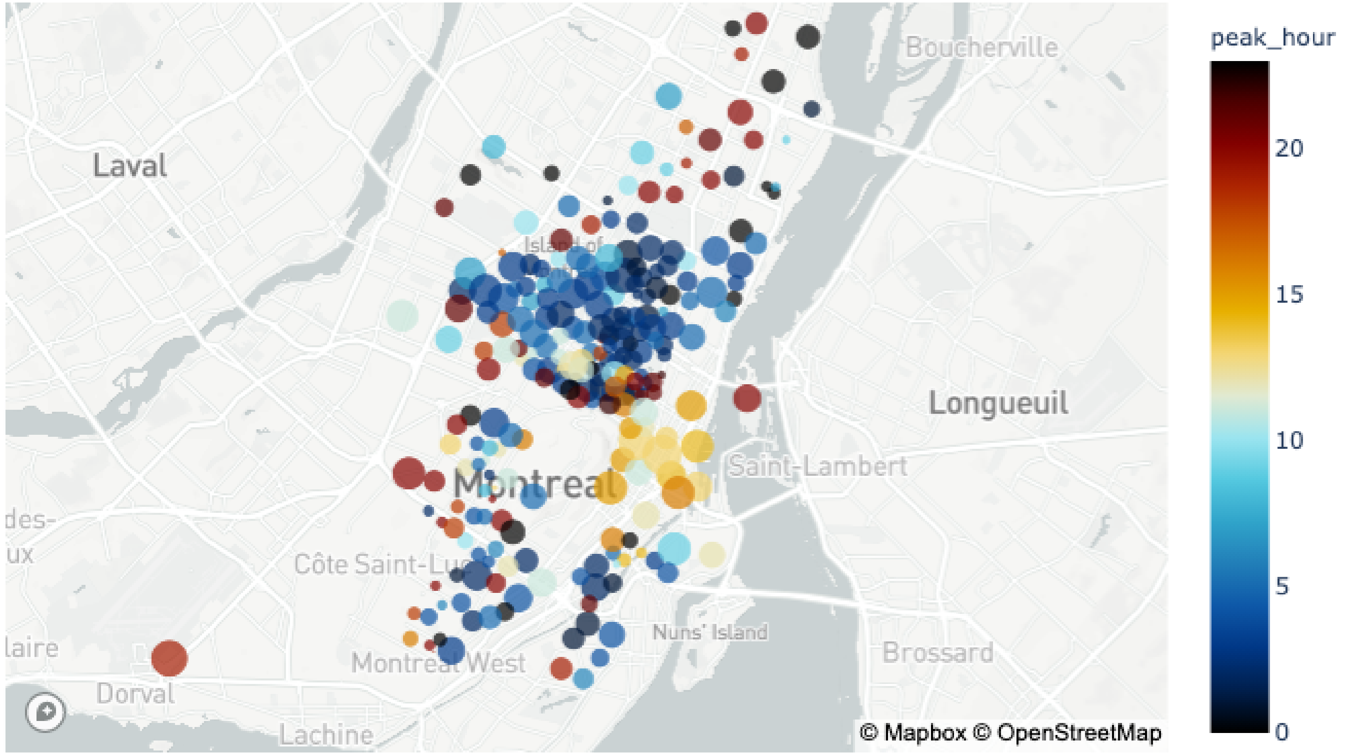

trace 0
trace 1
North America
Europe
Asia
Africa
South America
Map tiles by Stamen Design, under CC BY 3.0
Data by OpenStreetMap, under ODbL.

px.scatter_mapbox

Use scatter_mapbox from plotly.express for fast configuration

```python
import plotly.express as px
token='pk.eyJ1Ijoic3l2aW5jZSIsImEiOiJjazZrNTcwY3kwMHBrM2txaGJqZWEzNWExIn0.tLQHY_OoiR2NMxnYHXU
px.set_mapbox_access_token(token)
df = px.data.carshare()
fig = px.scatter_mapbox(df, lat="centroid_lat", lon="centroid_lon",
                         color="peak_hour",
                         size="car_hours",
                        color_continuous_scale=px.colors.cyclical.IceFire,
                        size_max=15, zoom=10)
fig.show()
```

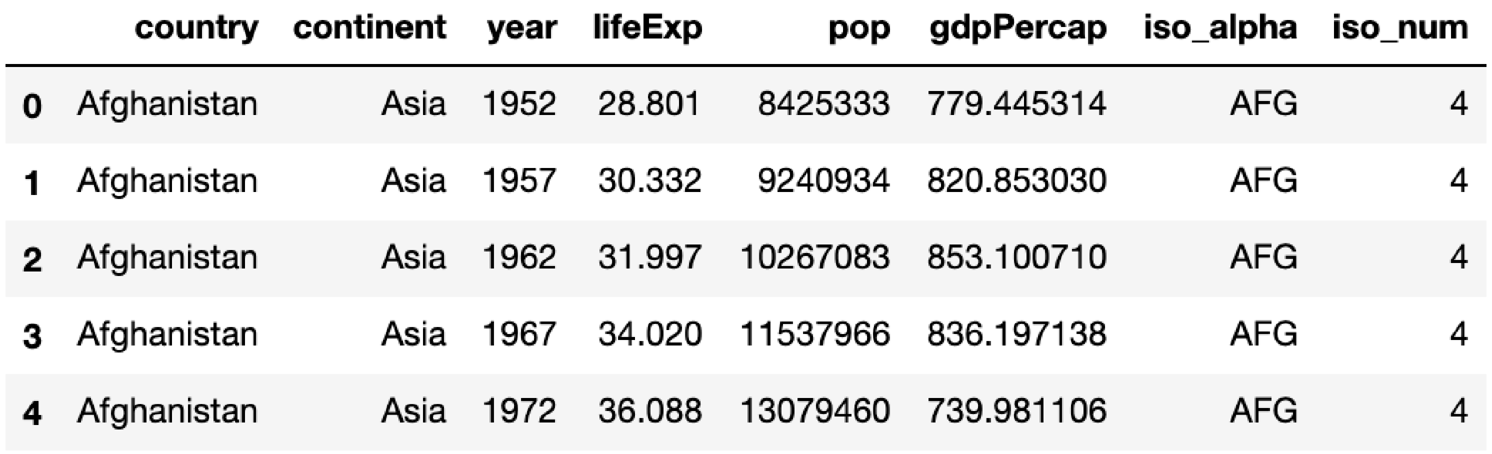
px.scatter_mapbox



px.scatter_geo

In a geographic scatter plot, each row of data_frame is represented by a symbol mark on a map.

```python
import plotly.express as px
df = px.data.gapminder().query("year==2007")
fig = px.scatter_geo(df,
                     locations="iso_alpha",
                     color="continent",
                     hover_name="country", size="pop",
                     projection="natural earth")
fig.show()
```



	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

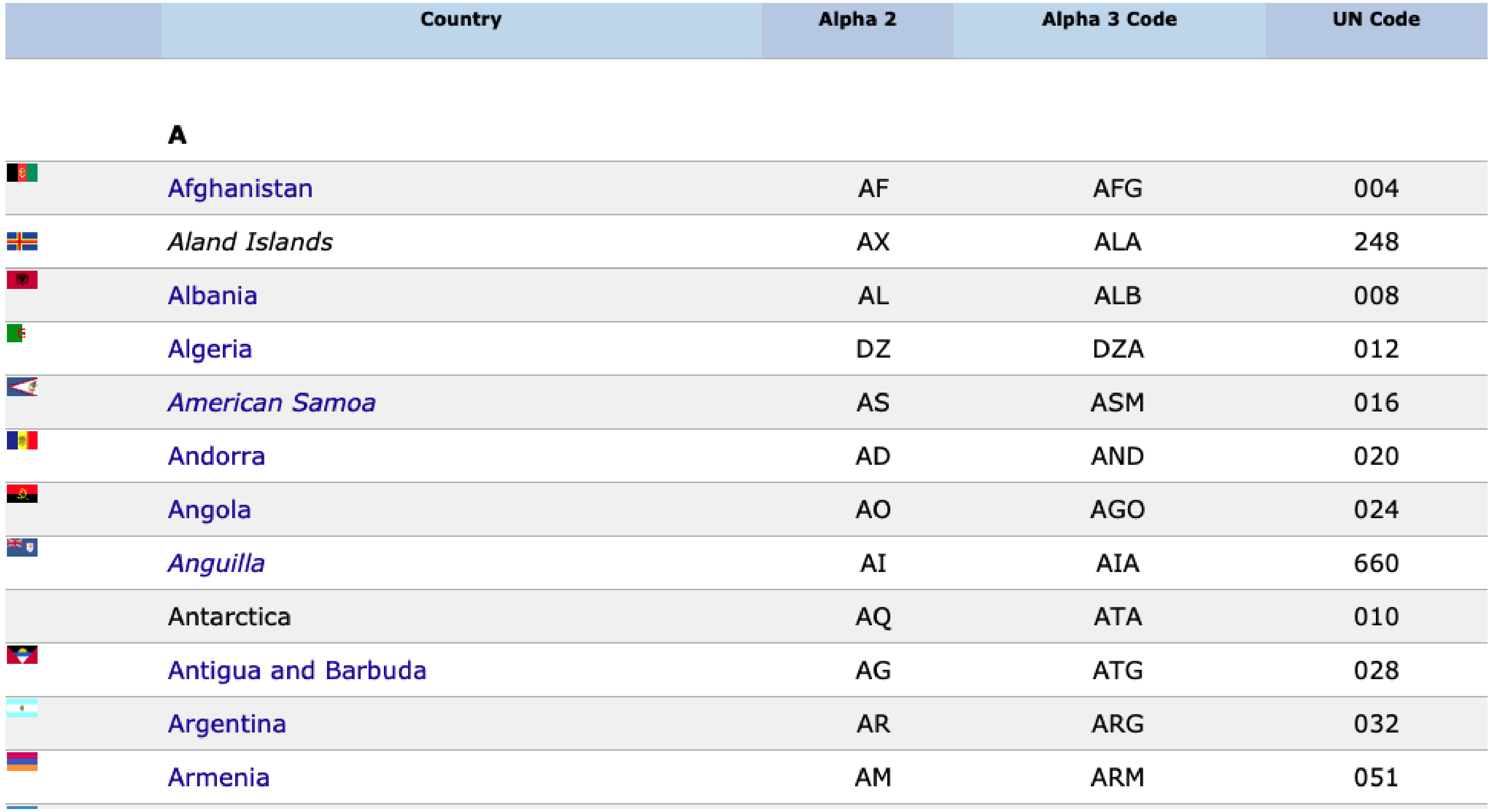
px.scatter_geo

In a geographic scatter plot, each row of data_frame is represented by a symbol mark on a map.

> locations (str or int or Series or array-like) – Either a name of a column in data_frame, or a pandas Series or array_like object. Values from this column or array_like are to be interpreted according to locationmode and mapped to longitude/latitude.
>
> locationmode (str) – One of ‘ISO-3’, ‘USA-states’, or ‘country names’ Determines the set of locations used to match entries in locations to regions on the map.

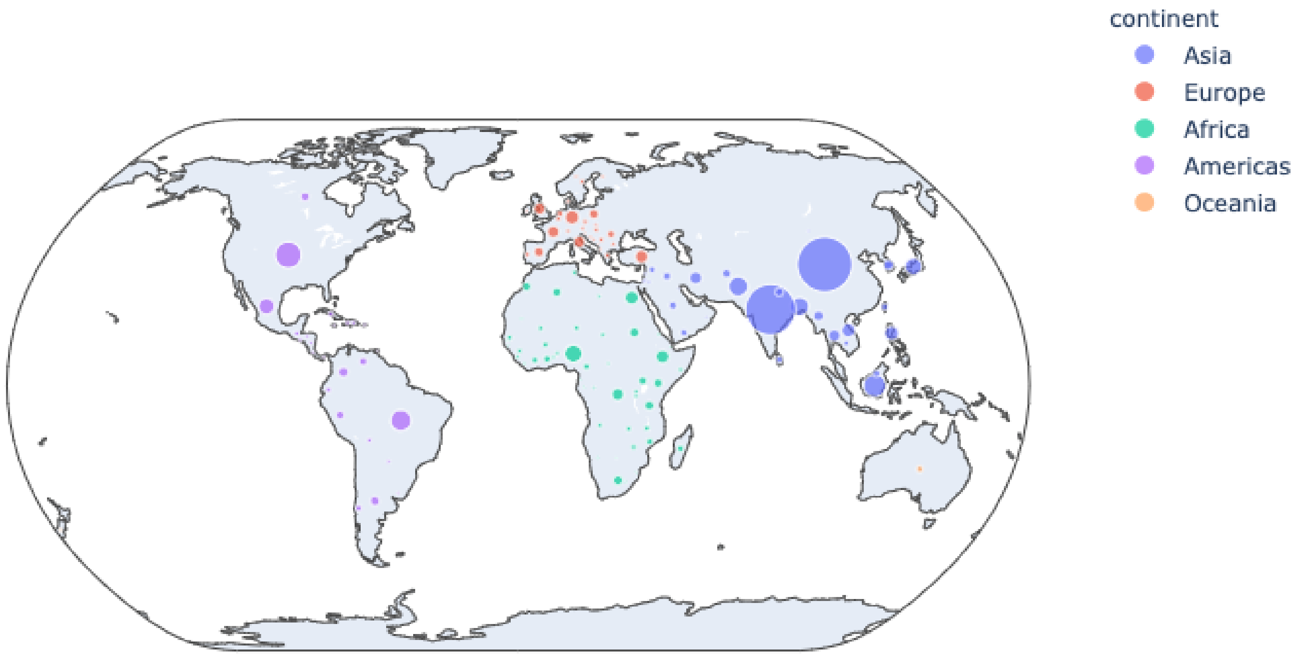
iso_alpha

The ISO country codes are internationally recognized codes that designate every country and most of the dependent areas a two-letter combination or a three-letter combination; it is like an acronym, that stands for a country or a state.



	Country	Alpha 2	Alpha 3 Code	UN Code
	A			
	Afghanistan	AF	AFG	004
	Aland Islands	AX	ALA	248
	Albania	AL	ALB	008
	Algeria	DZ	DZA	012
	American Samoa	AS	ASM	016
	Andorra	AD	AND	020
	Angola	AO	AGO	024
	Anguilla	AI	AIA	660
	Antarctica	AQ	ATA	010
	Antigua and Barbuda	AG	ATG	028
	Argentina	AR	ARG	032
	Armenia	AM	ARM	051

px.scatter_geo



px.scatter_geo

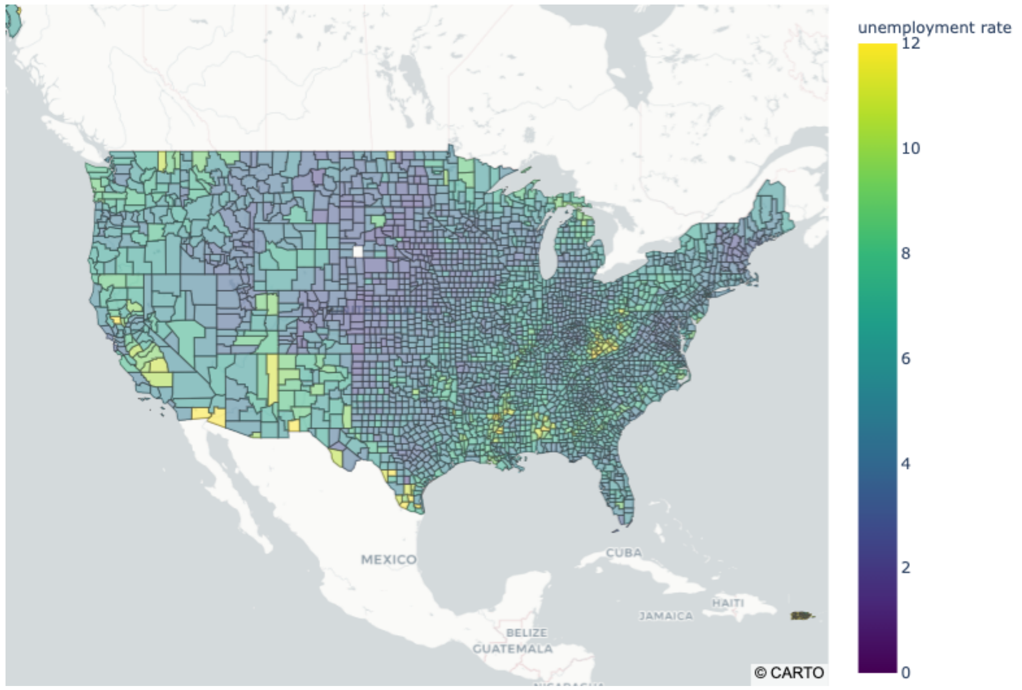
Using different locationmodes

```
import plotly.express as px
df = px.data.gapminder().query("year==2007")
fig = px.scatter_geo(df,
                      locations="iso_alpha",
                      color="continent",
                      locationmode='ISO-3',
                     hover_name="country", size="pop",
                     projection="natural earth")
fig.show()
```

```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter_geo(df,
                     locations="country",
                     color="continent",
                     locationmode='country names',
                     hover_name="country", size="pop",
                     projection="natural earth")
fig.show()
```

Choropleth map

A Choropleth Map is a map composed of colored polygons. It is used to represent spatial variations of a quantity.



jason

> JSON (JavaScript Object Notation) is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications.

```
{ "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {"streetAddress": "21 2nd Street",
              "city": "New York",},
  "phoneNumbers": [
            {"type": "home",
              "number": "212 555-1234"},
            {"type": "office",
              "number": "646 555-4567"}
            ],
  "children": [],
  "spouse": null
}
```

geojson

GeoJSON is a format for encoding a variety of geographic data structures.

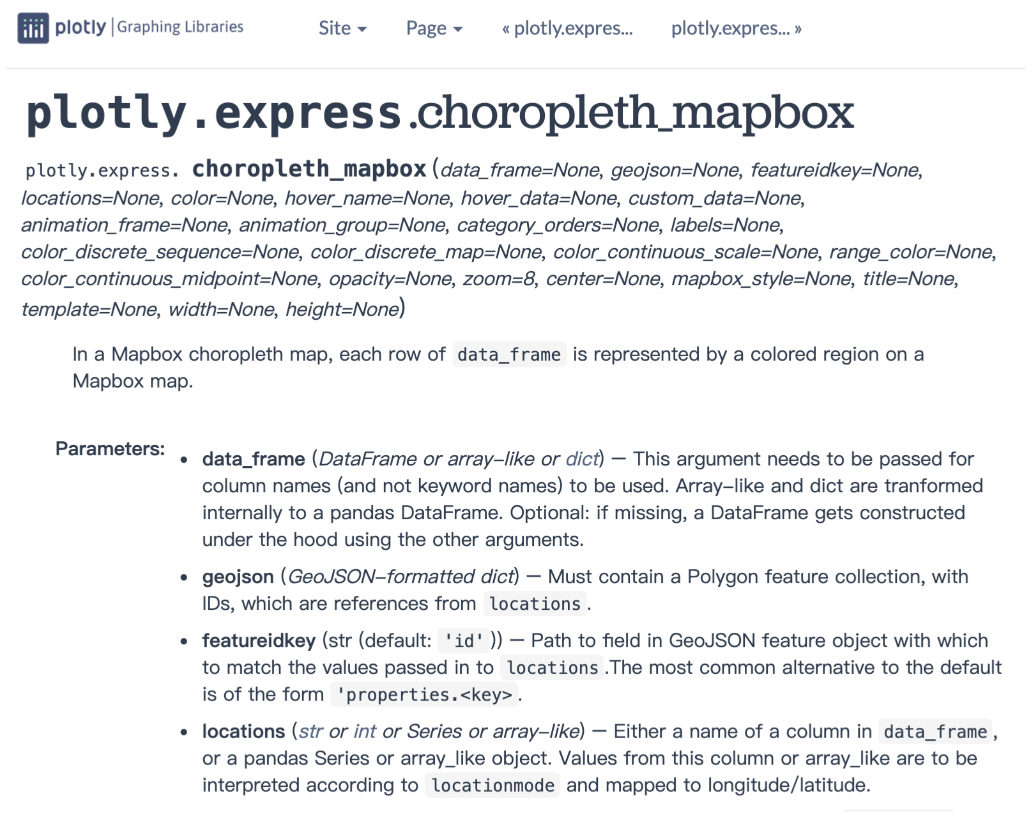
GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. Geometric objects with additional properties are Feature objects. Sets of features are contained by FeatureCollection objects.

```json
{"type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Learn more about GeoJSON https://www.youtube.com/watch?v=8RPfrhzRw2s

px.choropleth_mapbox

In a Mapbox choropleth map, each row of data_frame is represented by a colored region on a Mapbox map. Search for `px.choropleth_mapbox documentation`



plotly.express.choropleth_mapbox

`plotly.express.` **`choropleth_mapbox`**(*data_frame=None, geojson=None, featureidkey=None, locations=None, color=None, hover_name=None, hover_data=None, custom_data=None, animation_frame=None, animation_group=None, category_orders=None, labels=None, color_discrete_sequence=None, color_discrete_map=None, color_continuous_scale=None, range_color=None, color_continuous_midpoint=None, opacity=None, zoom=8, center=None, mapbox_style=None, title=None, template=None, width=None, height=None*)

In a Mapbox choropleth map, each row of `data_frame` is represented by a colored region on a Mapbox map.

Parameters:

- **data_frame** (*DataFrame or array–like or dict*) – This argument needs to be passed for column names (and not keyword names) to be used. Array–like and dict are tranformed internally to a pandas DataFrame. Optional: if missing, a DataFrame gets constructed under the hood using the other arguments.
- **geojson** (*GeoJSON–formatted dict*) – Must contain a Polygon feature collection, with IDs, which are references from `locations`.
- **featureidkey** (str (default: `'id'`)) – Path to field in GeoJSON feature object with which to match the values passed in to `locations`.The most common alternative to the default is of the form `'properties.<key>'`.
- **locations** (*str or int or Series or array–like*) – Either a name of a column in `data_frame`, or a pandas Series or array_like object. Values from this column or array_like are to be interpreted according to `locationmode` and mapped to longitude/latitude.

https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth_mapbox.html

px.choropleth_mapbox

```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.j
    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_mapbox(df, geojson=counties, locations='fips', color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                            zoom=3, #Between 0 and 20. Sets map zoom level.
                            center = {"lat": 37.0902, "lon": -95.7129},
                           #(dict) – Dict keys are 'lat' and 'lon' Sets the center point of th
                            opacity=0.5, #Value between 0 and 1. Sets the opacity for markers.
                           mapbox_style="carto-positron",
                           labels={'unemp':'unemployment rate'}
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

px.choropleth_mapbox

```
fig = px.choropleth_mapbox(df, geojson=counties, locations='fips', color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                           zoom=3, #Between 0 and 20. Sets map zoom level.
                           center = {"lat": 37.0902, "lon": -95.7129},
                           #(dict) – Dict keys are 'lat' and 'lon' Sets the center point of tl
                           opacity=0.5, #Value between 0 and 1. Sets the opacity for markers.
                            mapbox_style="carto-positron",
                           labels={'unemp':'unemployment rate'}
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```

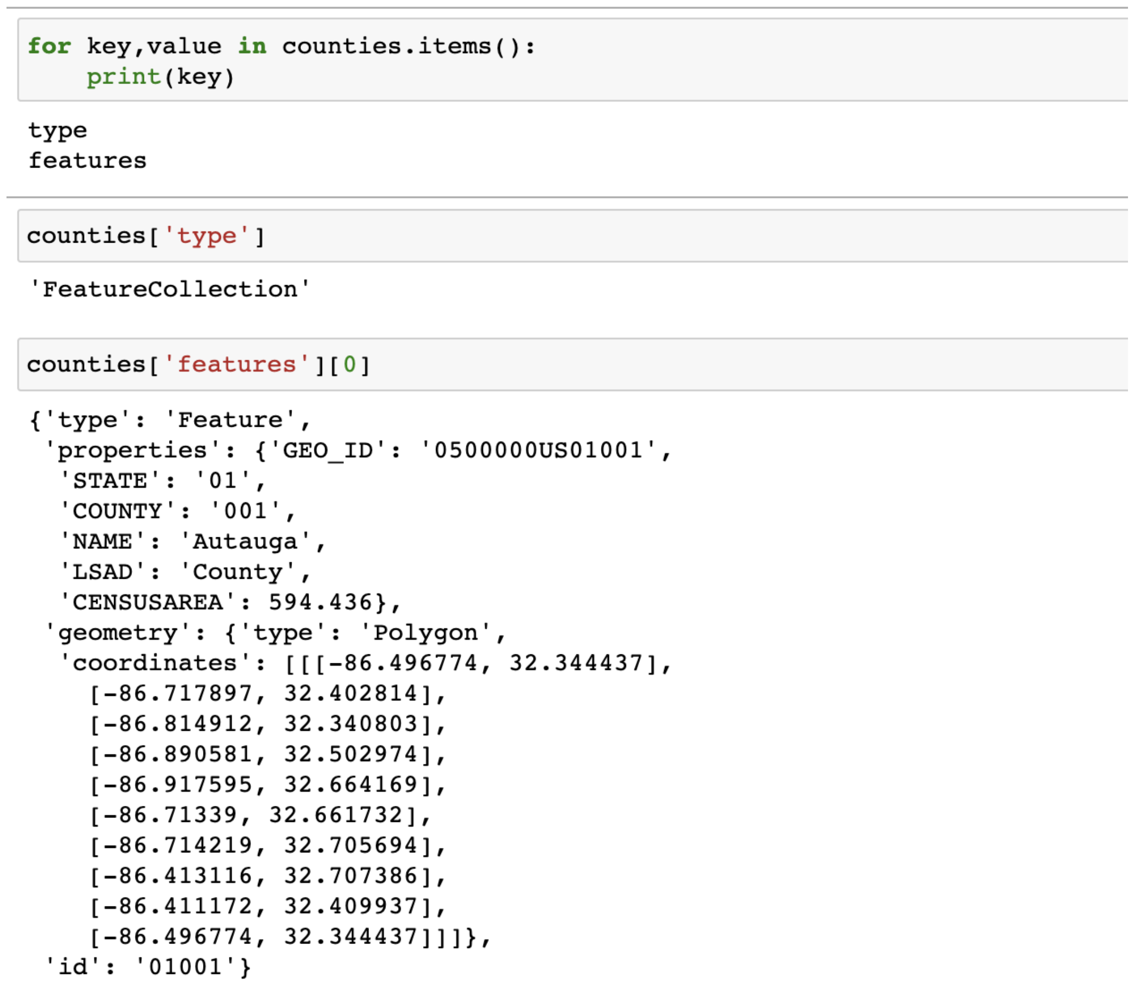
> mapbox_style (str (default 'basic', needs Mapbox API token)) – Identifier of base map style, some of which require a Mapbox API token to be set using plotly.express.set_mapbox_access_token(). Allowed values which do not require a Mapbox API token are 'open-street-map', 'white-bg', 'carto-positron', 'carto-darkmatter', 'stamen- terrain', 'stamen-toner', 'stamen-watercolor'. Allowed values which do require a Mapbox API token are 'basic', 'streets', 'outdoors', 'light', 'dark', 'satellite', 'satellite- streets'.

px.choropleth_mapbox

The Federal Information Processing Standard Publication 6-4 (FIPS 6-4) was a five-digit Federal Information Processing Standards code which uniquely identified counties and county equivalents in the United States, certain U.S. possessions, and certain freely associated states.

```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.
    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})
df.head()
```

px.choropleth_mapbox



```python
for key,value in counties.items():
    print(key)
```

```
type
features
```

```python
counties['type']
```

```
'FeatureCollection'
```

```python
counties['features'][0]
```

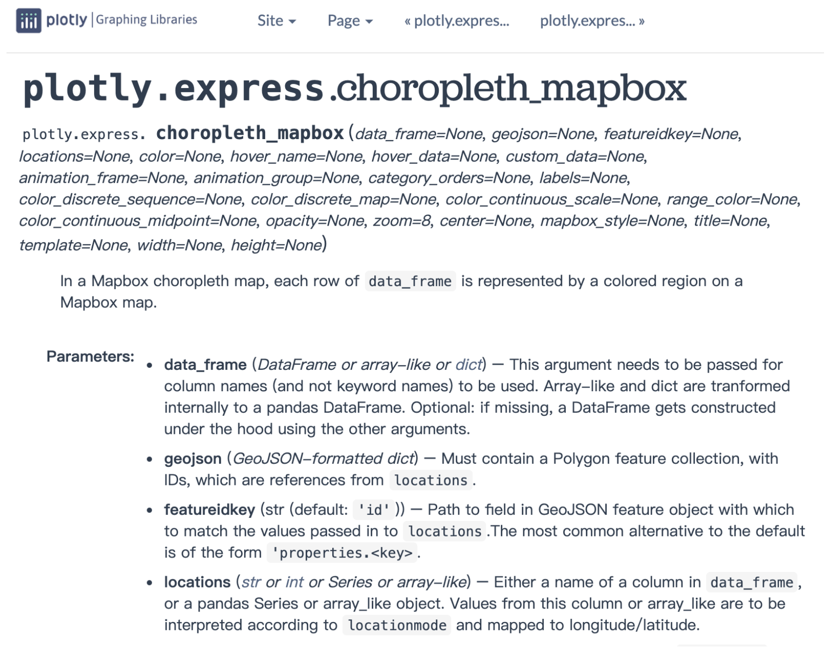
```
{'type': 'Feature',
 'properties': {'GEO_ID': '0500000US01001',
  'STATE': '01',
  'COUNTY': '001',
  'NAME': 'Autauga',
  'LSAD': 'County',
  'CENSUSAREA': 594.436},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[-86.496774, 32.344437],
    [-86.717897, 32.402814],
    [-86.814912, 32.340803],
    [-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'}
```

px.choropleth_mapbox

```
fig = px.choropleth_mapbox(df,
                           geojson=counties,
                           locations='fips',
                           color='unemp',
                           color_continuous_scale="Viridis",
                           range_color=(0, 12),
                           mapbox_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()
```

px.choropleth_mapbox

Search for `px.choropleth_mapbox documentation`



plotly.express.choropleth_mapbox

`plotly.express.` **`choropleth_mapbox`**(*data_frame=None, geojson=None, featureidkey=None, locations=None, color=None, hover_name=None, hover_data=None, custom_data=None, animation_frame=None, animation_group=None, category_orders=None, labels=None, color_discrete_sequence=None, color_discrete_map=None, color_continuous_scale=None, range_color=None, color_continuous_midpoint=None, opacity=None, zoom=8, center=None, mapbox_style=None, title=None, template=None, width=None, height=None*)

In a Mapbox choropleth map, each row of `data_frame` is represented by a colored region on a Mapbox map.

Parameters:

- **data_frame** (*DataFrame or array–like or dict*) — This argument needs to be passed for column names (and not keyword names) to be used. Array–like and dict are tranformed internally to a pandas DataFrame. Optional: if missing, a DataFrame gets constructed under the hood using the other arguments.
- **geojson** (*GeoJSON–formatted dict*) — Must contain a Polygon feature collection, with IDs, which are references from `locations`.
- **featureidkey** (str (default: `'id'`)) — Path to field in GeoJSON feature object with which to match the values passed in to `locations`.The most common alternative to the default is of the form `'properties.<key>`.
- **locations** (*str or int or Series or array–like*) — Either a name of a column in `data_frame`, or a pandas Series or array_like object. Values from this column or array_like are to be interpreted according to `locationmode` and mapped to longitude/latitude.

https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth_mapbox.html

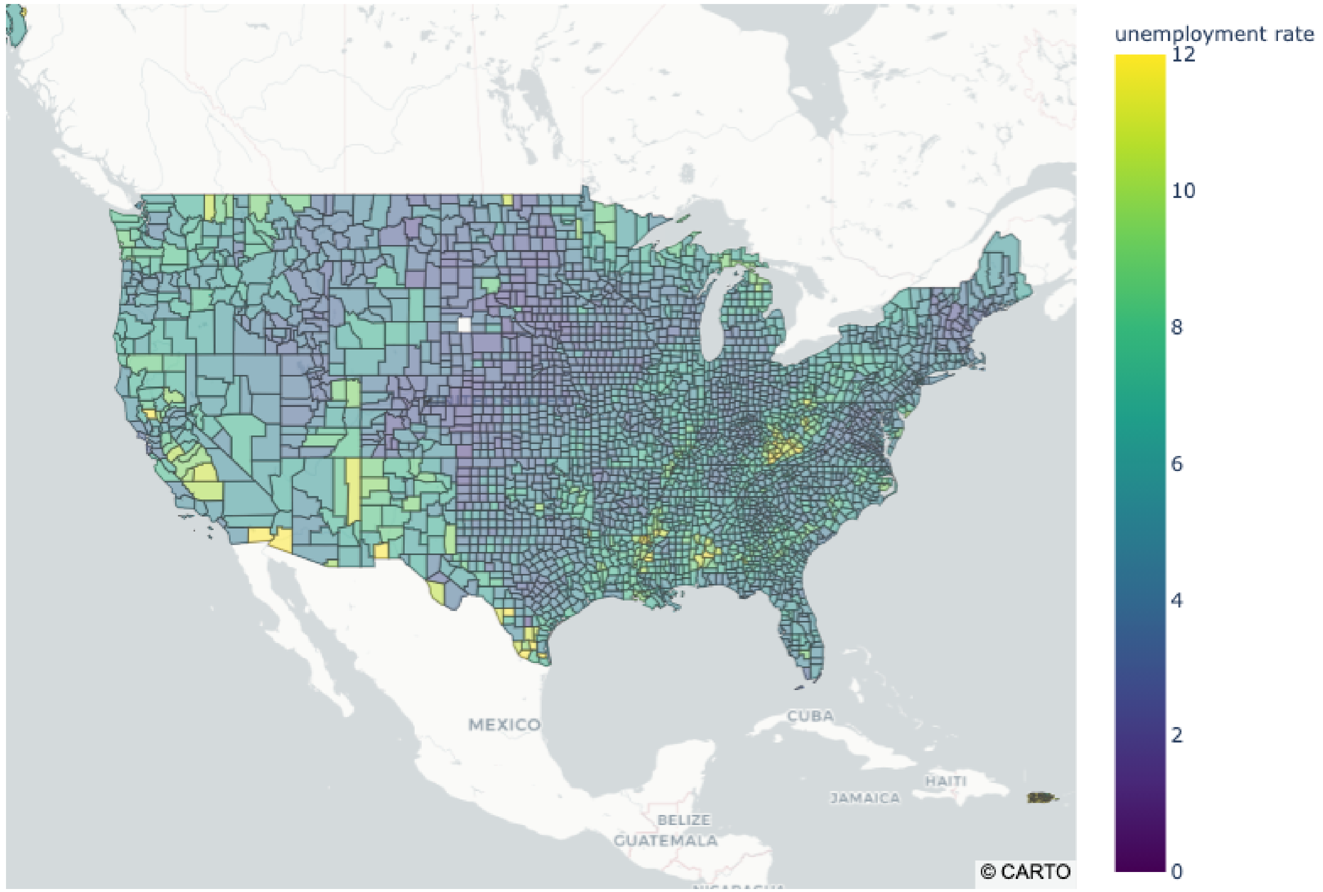
px.choropleth_mapbox

Search for `px.choropleth_mapbox documentation`

> geojson (GeoJSON-formatted dict) – Must contain a Polygon feature collection, with IDs, which are references from locations.
>
> locations (str or int or Series or array-like) – Either a name of a column in data_frame, or a pandas Series or array_like object. Values from this column or array_like are to be interpreted according to locationmode and mapped to longitude/latitude.

https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth_mapbox.html

px.choropleth_mapbox



px.choropleth_mapbox

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.

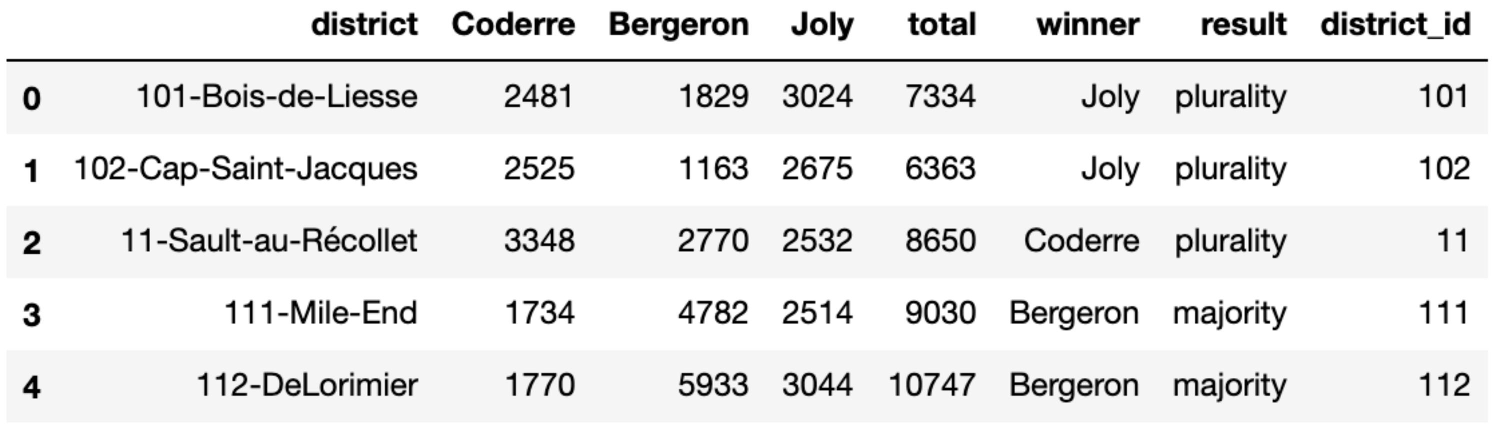
In the following GeoJSON object/data-file pairing, the values of properties.district match the values of the district column.

> featureidkey (str (default: 'id')) – Path to field in GeoJSON feature object with which to match the values passed in to locations.The most common alternative to the default is of the form 'properties.`<key>`'.

px.choropleth_mapbox

```python
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'}
```

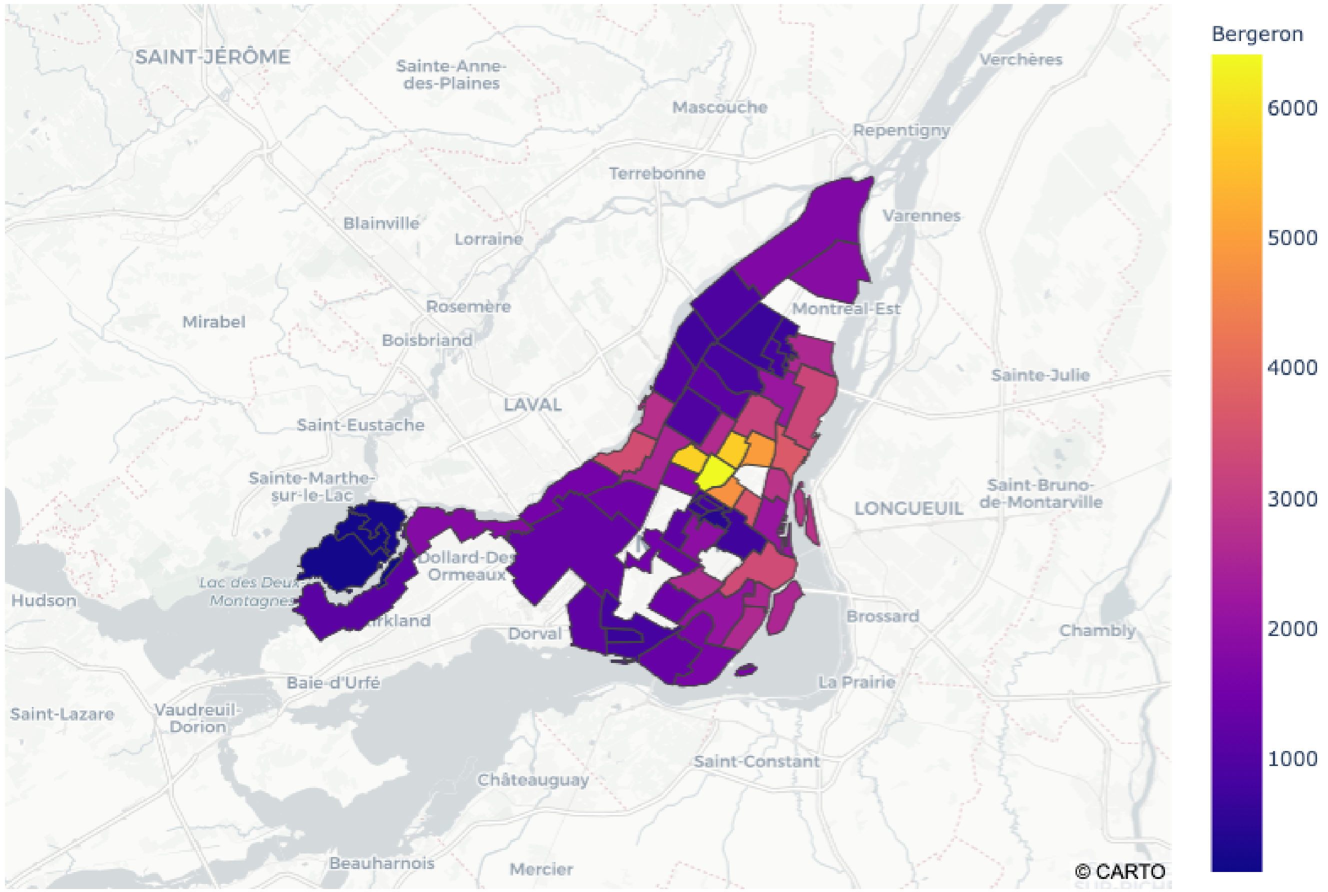


	district	Coderre	Bergeron	Joly	total	winner	result	district_id
0	101-Bois-de-Liesse	2481	1829	3024	7334	Joly	plurality	101
1	102-Cap-Saint-Jacques	2525	1163	2675	6363	Joly	plurality	102
2	11-Sault-au-Récollet	3348	2770	2532	8650	Coderre	plurality	11
3	111-Mile-End	1734	4782	2514	9030	Bergeron	majority	111
4	112-DeLorimier	1770	5933	3044	10747	Bergeron	majority	112

px.choropleth_mapbox

```python
import plotly.express as px
df = px.data.election()
geojson = px.data.election_geojson()

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

px.choropleth_mapbox



Thank you!

Q&A or Email wbdu@hku.hk