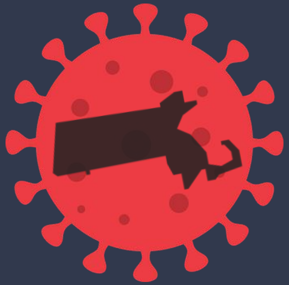
# Visualizing synthetic COVID-19 data in Massachusetts

Technology Review Presentation

Aja Sutton
Andrew Teng
Jason Thomas
Nanhsun Yuan



## Background

COVID-19 data, whether real or synthetic, can be overwhelming and the need to visualize the data is an important task for researchers and the public alike.

We propose to create a geovisualization tool to show how the COVID-19 outbreak has progressed over time in the state of Massachusetts.

We will visualize aerial data (density like case counts etc. grouped by geographic level) and point data such as hospitals in the area, etc.

Our data are synthetic, yet formatted in a common data model.

## Use Case

Health Analytics Personnel who:

- Have limited-to-no experience with Geographic Information Science (GIS)
- Are interested in geo-visualizing disease distribution data based on raw input (e.g. counts)
- Have time-series data in discrete bins they want to interrogate
- Want to be able to convey numerical data visually to an audience with variable expertise

# Potential Python Libraries

Folium

+ Purpose-built for geovisualization: it makes choropleth maps quickly.
+ Basemap is built in, and easily customizable.
+ Can integrate with Dash using iFrames to create an interactive dashboard.
+ Intuitive for users familiar with geovisualization techniques (e.g. can use raster or vector layers).

- Software external to Dash; extra step passing it through iFrame to prepare output.
- Only plots choropleth maps (pretty basic, but impactful depending on use case).
- Poor flexibility for interactivity (e.g. fancy tooltips)

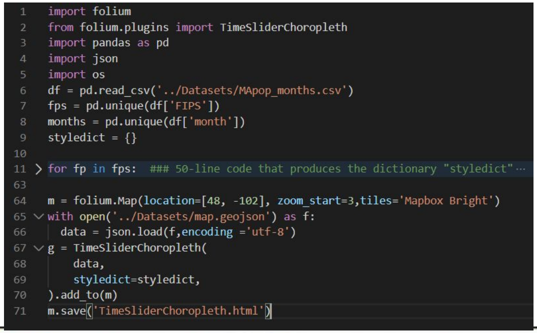
plotly

+ Smoother integration to Dash (which is built directly on Plotly, among others)
+ More powerful -- not just mapping
+ Easy to integrate multiple types of data visualizations into final deliverable (e.g. maps, charts, graphs, etc.)

- More complex. Steeper learning curve (multiple libraries, e.g. Mapbox, Express, etc.). Cool, but overwhelming!
- Multiple ways to do same thing, and sometimes unclear which is simplest solution (e.g. animation vs time slider do the same thing, but animation is one line of code, and time slider is ~50).

# Folium vs. Plotly

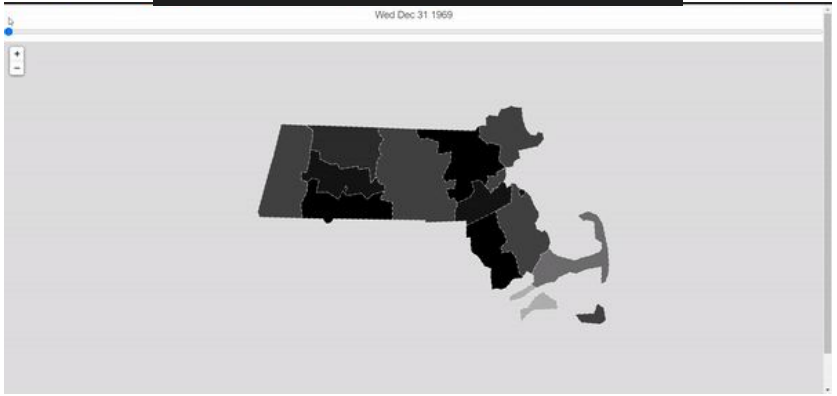
## Folium



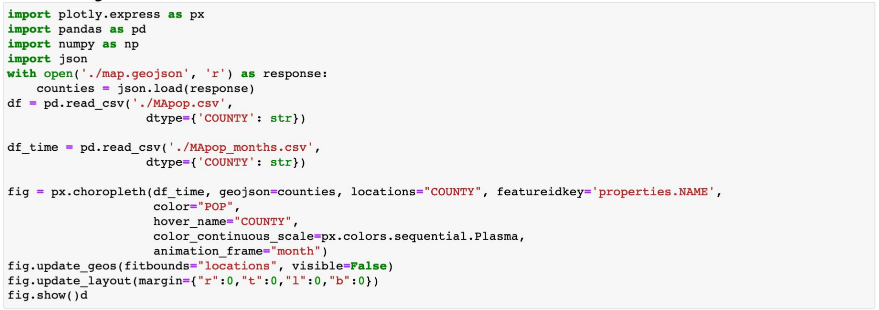
```python
import folium
from folium.plugins import TimeSliderChoropleth
import pandas as pd
import json
import os
df = pd.read_csv('../Datasets/MApop_months.csv')
fps = pd.unique(df['FIPS'])
months = pd.unique(df['month'])
styledict = {}

for fp in fps:  ### 50-line code that produces the dictionary "styledict"···

m = folium.Map(location=[48, -102], zoom_start=3,tiles='Mapbox Bright')
with open('../Datasets/map.geojson') as f:
    data = json.load(f,encoding ='utf-8')
g = TimeSliderChoropleth(
    data,
    styledict=styledict,
).add_to(m)
m.save('TimeSliderChoropleth.html')
```



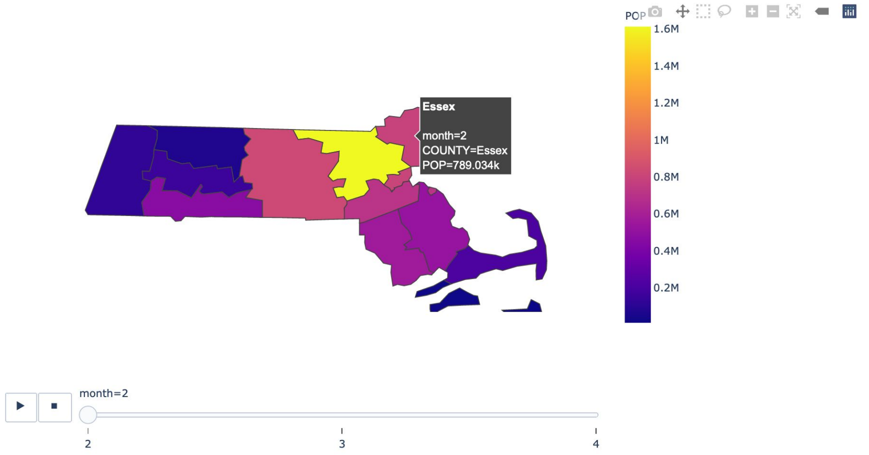
## Plotly



```python
import plotly.express as px
import pandas as pd
import numpy as np
import json
with open('./map.geojson', 'r') as response:
    counties = json.load(response)
df = pd.read_csv('./MApop.csv',
                   dtype={'COUNTY': str})

df_time = pd.read_csv('./MApop_months.csv',
                   dtype={'COUNTY': str})

fig = px.choropleth(df_time, geojson=counties, locations="COUNTY", featureidkey='properties.NAME',
                    color="POP",
                    hover_name="COUNTY",
                    color_continuous_scale=px.colors.sequential.Plasma,
                    animation_frame="month")
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()d
```


POP
1.6M
1.4M
1.2M
1M
0.8M
0.6M
0.4M
0.2M
Essex
month=2
COUNTY=Essex
POP=789.034k
month=2
2
3
4