Mapping Covid-19 Data

This exercise introduces some basic plotting of geospatial time series data - Global and US statistics on Covid-19 deaths and confirmed cases.

The 'Geopandas' analysis module will be used to download and visualize the data in map and chart form>

install the needed additional modules

The python environment in Colaboratory has many modules for scientific computing already installed, but it is straightforward to add additional modules. For this exercise, we will need the 'Geopandas' module to work with geospatial data.

```
!pip install geopandas
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.6/dist-packages (0.8.1
Requirement already satisfied: fiona in /usr/local/lib/python3.6/dist-packages (from geop
Requirement already satisfied: pandas>=0.23.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: shapely in /usr/local/lib/python3.6/dist-packages (from ge
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.6/dist-packages (from 
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.6/dist-packages (from f
Requirement already satisfied: munch in /usr/local/lib/python3.6/dist-packages (from fion
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (fr
```

```
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from py
```

```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import folium
```

Get the data

We will work with a dataset maintained by John Hopkins University on github. The .csv files online can be directly read in as 'pandas' datasets.

> Time Series Datasets - https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_US.csv

> Source of the data on github - https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_dataIndented

```
# needs to point to the 'raw' csv file on github, not the nicely rendered html version
url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/css
usdeathsurl = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_
worlddeathsurl = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_
# Get a nice .csv with ISO3 codes to match later...
luurl = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/U
```

```
# read in the data
usdeaths = pd.read_csv(usdeathsurl)
confirmed = pd.read_csv(url)
gdeaths = pd.read_csv(worlddeathsurl)
lutab = pd.read_csv(luurl)
```

Have a look at the data

Check out the global confirmed cases globally

```
confirmed.head(3)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	

3 rows × 298 columns

The us deaths by county, etc.

```
usdeaths.head(3)
```

	UID	iso2	iso3	code3	FIPS	Admin2	Province_State	Country_Region	Lat	
0	84001001	US	USA	840	1001.0	Autauga	Alabama	US	32.539527	-86
1	84001003	US	USA	840	1003.0	Baldwin	Alabama	US	30.727750	-87
2	84001005	US	USA	840	1005.0	Barbour	Alabama	US	31.868263	-85

3 rows × 306 columns

Global deaths by country, but there are some matching issues later

```
gdeaths.head(3)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	

3 rows × 298 columns

```
# look at first 6 fields in the data
list(usdeaths)[:6]
```

```
['UID', 'iso2', 'iso3', 'code3', 'FIPS', 'Admin2']
```

Make the data geospatial

This can be done:

- directly into points using the Lat/Long coordinates
- by joining the tables to existing polygon data

To convert directly into points, create a 'geometry' field from the Lat and Long fields

```
# make into point geopandas dataframes

usdeaths = gpd.GeoDataFrame(usdeaths, crs=('epsg:4326'), geometry=gpd.points_from_xy(usdeaths.
confirmed = gpd.GeoDataFrame(confirmed, crs=('epsg:4326'), geometry=gpd.points_from_xy(confirm
gdeaths = gpd.GeoDataFrame(gdeaths, crs=('epsg:4326'), geometry=gpd.points_from_xy(gdeaths.Lon
```

A 'geometry' column is at the end of the datasets now

```
list(gdeaths)[-3:]
```

```
['11/9/20', '11/10/20', 'geometry']
```

```
list(confirmed)[-3:]
```

```
['11/9/20', '11/10/20', 'geometry']
```

The data on the most recent number of cases or deaths in the the second to last column now

```
mostrecent = list(usdeaths)[-2] #last item in list is geometry
mostrecent
```

```
'11/10/20'
```

Plot the point data

- use the most recent deaths to create graduated sizes

A basic plot of just the points is easily done, using default colours. The size of the points (markersize) is specified in points?

https://stackoverflow.com/questions/14827650/pyplot-scatter-plot-marker-size#:~:text=The%20standard%20size%20of%20points,is%20hence%201%2F72%20inches.&text=If%20you%20are%20interested%20in,data%20units%2C%20check%20this%20answer.

https://www.delftstack.com/howto/matplotlib/how-to-set-marker-size-of-scatter-plot-in-matplotlib/

Check out the highest value in the most recent data

[ ] ↳ *5 cells hidden*

Add a background map

[ ] ↳ *8 cells hidden*

Adjust the map area

We can filter the points by latitude and longitude to focus more on North America.

https://kanoki.org/2020/01/21/pandas-dataframe-filter-with-multiple-conditions/

[ ] ↳ *3 cells hidden*

A little fancier styling

A US Map

https://www.earthdatascience.org/courses/scientists-guide-to-plotting-data-in-python/plot-spatial-data/customize-vector-plots/python-change-spatial-extent-of-map-matplotlib-geopandas/

Although since the data is lat/long it would be straightforward to set the extent directly with xmin, etc.

[ ] ↳ *1 cell hidden*

Styling a global map

Now, lets return to the global data, and put some text in the figure.

https://matplotlib.org/3.2.2/gallery/pyplots/text_commands.html#sphx-glr-gallery-pyplots-text-commands-py

[ ] ↳ *8 cells hidden*

Make a function

That looks okay - now to adapt the code into a function so that any date could be plotted automatically.

[ ] ↳ *4 cells hidden*

Plot a time series from the data

Since the data is a time series, we might as well have a look at this aspect as well.

The table is convenient, but not formatted for easy plotting over time, so will have to do some manipulations to easily plot a time series for a country.

[ ] ↳ *17 cells hidden*

Look at it with Folium

Another way to look at the data is with an online map. We an use the 'folium' module to do this. The code may look a bit intimidating, but check it out.

You can also look at this article, or search online for more references:

- Data Visualization with Python Folium Maps
- https://towardsdatascience.com/data-visualization-with-python-folium-maps-a74231de9ef7

```
# Make an empty map
m = folium.Map(width=1000,height=500,location=[50,-100], zoom_start=3)

date = '10/29/20'
loc = 'Covid-19 US Deaths'
title_html = '''
             <h3 align="center" style="font-size:16px"><b>{}</b> - {}</h3>
             '''.format(loc,date)
```

```
m.get_root().html.add_child(folium.Element(title_html))

usdeaths.describe()

# get the data
data = usdeaths
# data.head()
# I can add marker one by one on the map
for i in range(0,len(data)):
   folium.Circle(
      location=[data.iloc[i]['Lat'], data.iloc[i]['Long_']],
      popup=data.iloc[i]['Province_State'],
      radius=data.iloc[i][date]/10,
      color='crimson',
      fill=True,
      fill_color='crimson'
   ).add_to(m)

# Save it as html
m.save(f'USCovid.html')

# view the map
m
```

Make this Notebook Trusted to load map: File -> Trust Notebook

Covid-19 US Deaths - 10/29/20

The End

The last command should have also created an html file in your workspace called "GlobalCovid.html" - download this and have a look at it.

This part of the exercise is over. Your task now is to modify or extend some of the code above to plot the data for the usdeaths instead of for global deaths or global confirmed cases.

The folium map could be adapted by adding 'pop-ups' or changing the display.

Figure 1

```
def date_to_us_deaths(date):

    # Get spatial extent  - to zoom in on the map rather than clipping
    aoi_bounds = nw_world_us.geometry.total_bounds

    # Create x and y min and max objects to use in the plot boundaries
    xmin, ymin, xmax, ymax = aoi_bounds

    # Plot the data with a modified spatial extent
    fig, ax = plt.subplots(figsize = (10,6))
```

```
    xlim = ([xmin,  xmax])
    ylim = ([ymin,  ymax])

    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

    nw_world_us.plot(color='red', alpha = .5, ax = ax, markersize=nw_world_us[date]/10)
    world.plot(color='grey', ax=ax, alpha=.5)

    ax.set(title=f'US Deaths from Covid \n {date}')
    plt.show()

    # pull useful columns out of us deaths dataframe, sort in descending order (by deaths for
    # note: would not print to screen unless I returned it
    return nw_world_us[['Combined_Key', mostrecent, 'Population', 'Lat', 'Long_']].sort_values

date_to_us_deaths(mostrecent)
```

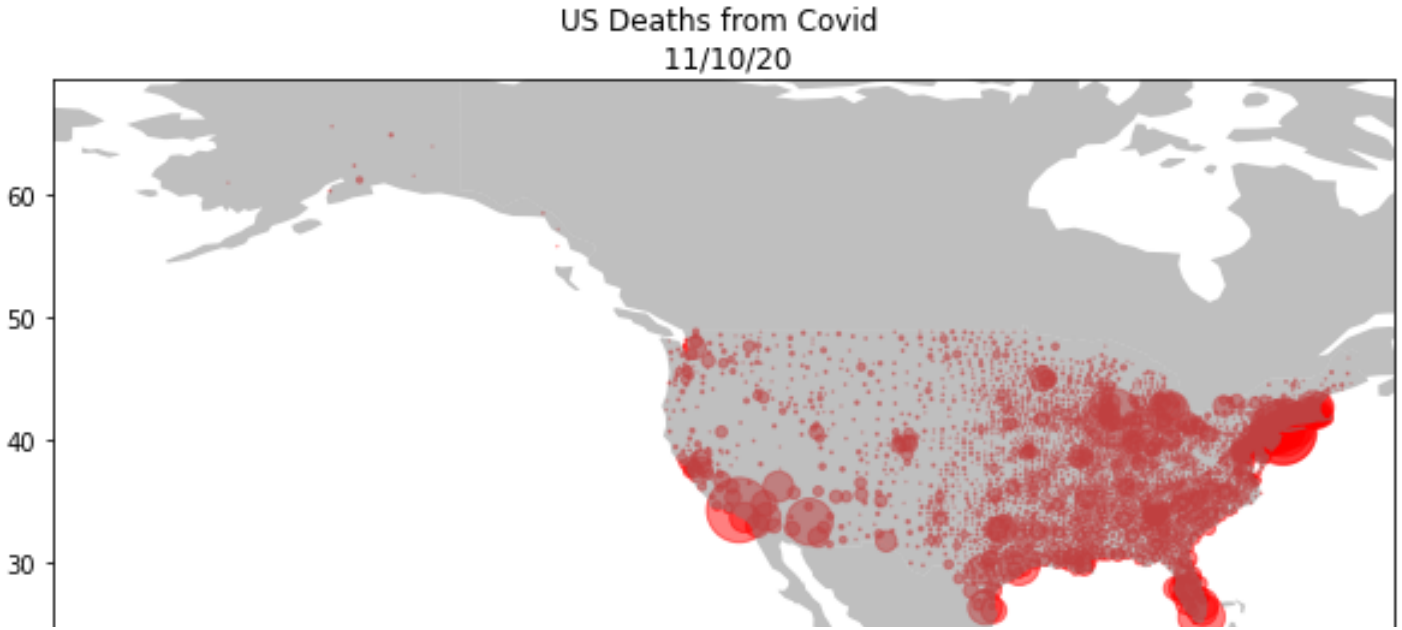

Figure 2

```
                −160          −140          −120          −100          −80
```

```
def date_to_us_deaths_folium(date):
    # Make an empty map
    m = folium.Map(width=1000,height=500,location=[50,-100], zoom_start=3)

    title_html = f'''
      <h3 align="center" style="font-size:16px"><b>Covid-19 US Deaths</b> - {date}</h3>
      '''
    m.get_root().html.add_child(folium.Element(title_html))

    usdeaths.describe()

    # get the data
    data = usdeaths
    # add markers one by one on the map
    for i in range(0,len(data)):
```

```
        folium.Circle(
            location=[data.iloc[i]['Lat'], data.iloc[i]['Long_']],
            popup=data.iloc[i]['Province_State'],
            radius=data.iloc[i][date]/10,
            color='crimson',
            fill=True,
            fill_color='crimson'
        ).add_to(m)

    # save it as html. file systems might not like '/' in path so replace it
    m.save(f'USCovid_{mostrecent.replace("/", "_")}.html')

    # view the map. note: did not print to screen unless returned
    return m

date_to_us_deaths_folium(mostrecent)
```

Make this Notebook Trusted to load map: File -> Trust Notebook

Covid-19 US Deaths - 11/10/20