IM-UH 1511 **Introduction to Digital Humanities**

HOMEWORK 8

Tracking the global coronavirus outbreak

50 points totally

Please create a folder named "html_map_output" in the same place where you're running this notebook and then submit the zipped the contents of this folder together with your homework notebook.

In [ ]:
```
# tracker
import COVID19Py # pip install COVID19Py
# https://pypi.org/project/COVID19Py/
# https://github.com/ExpDev07/coronavirus-tracker-api

# folium
import folium # pip install folium OR conda install -c conda-forge folium
from folium import plugins
import folium.plugins
from folium.plugins import MarkerCluster
MarkerCluster()
from folium.plugins import StripePattern

# essential libraries
import random
import datetime
from datetime import timedelta
import numpy as np
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# visualization
import matplotlib.pyplot as plt
import seaborn as sns
import calmap
# color pallette
cnf, dth, rec, act = '#393e46', '#ff2e63', '#21bf73', '#fe9801'

import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots

# html embedding
from IPython.display import Javascript
from IPython.core.display import display
from IPython.core.display import HTML

# warnings handling
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.simplefilter('ignore')
```

In [ ]:
```
covid19 = COVID19Py.COVID19()
locations = covid19.getLocations()
country=[]
country_code=[]
province=[]
latitude=[]
longitude=[]
confirmed=[]
deaths=[]
recovered=[]
last_updated=[]
for i in range(len(locations)):
    column=locations[i]
    country.append(column['country'])
    country_code.append(column['country_code'])
    province.append(column['province'])
    latitude.append(float(column['coordinates']['latitude']))
    longitude.append(float(column['coordinates']['longitude']))
    confirmed.append(int(column['latest']['confirmed']))
    deaths.append(int(column['latest']['deaths']))
    recovered.append(int(column['latest']['recovered']))
    last_updated.append(column['last_updated'])
df = pd.DataFrame(
    {'country':country,
     'country_code':country_code,
     'province':province,
     'latitude': latitude,
     'longitude':longitude,
     'confirmed':confirmed,
     'deaths':deaths,
     'recovered':recovered,
     'last_updated':last_updated
    })
df=df[['country_code','country','province','latitude','longitude','confirme
df['last_updated'] = pd.to_datetime(df['last_updated'], format="%Y-%m-%d %H
df
```

In [ ]:
```
dfn=df.copy()
dfc=df.copy()
ccsd=sorted(set(dfn.country.tolist()))
dfn['confirmed'] = dfn['confirmed'].astype(int)
dfn['deaths'] = dfn['deaths'].astype(int)
dfn['recovered'] = dfn['recovered'].astype(int)
tconfirmed = dfn['confirmed'].sum()
tdeaths = dfn['deaths'].sum()
trecovered = dfn['recovered'].sum()
dtnow=datetime.datetime.now()
dtupd=[str(d)[:11] for d in dfn['last_updated'].tolist()][0]
print("Time now:", dtnow)
print("Time data upadated:",dtupd)
print(len(ccsd), "Countries")
print(tconfirmed,'confirmed')
print(tdeaths,'deaths')
# print(trecovered,'recovered')
```

In [ ]:
```
# cases
cases = ['confirmed', 'deaths', 'recovered', 'active']

# Active Case = confirmed - deaths - recovered
df['active'] = df['confirmed'] - df['deaths'] - df['recovered']

# # replacing Mainland china with just China
# full_table['Country/Region'] = full_table['Country/Region'].replace('Main

# filling missing values
df[['province']] = df[['province']].fillna('')
df[cases] = df[cases].fillna(0)

# fixing datatypes
df['recovered'] = df['recovered'].astype(int)

df.sample(6)
```

In [ ]:
```
# cases in the ships
ship = df[df['province'].str.contains('Grand Princess')|df['country'].str.c

# china and the row
china = df[df['country']=='China']
row = df[df['country']!='China']

# latest
# df = df[full_table['Date'] == max(full_table['Date'])].reset_index()
china_latest = df[df['country']=='China']
row_latest = df[df['country']!='China']

# latest condensed
df_grouped = df.groupby('country')['confirmed', 'deaths', 'recovered', 'act
china_latest_grouped = china_latest.groupby('province')['confirmed', 'death
row_latest_grouped = row_latest.groupby('country')['confirmed', 'deaths', '
```

In [ ]:
```
log_confirmed=np.log(df_grouped["confirmed"])
df_grouped["log_confirmed"]=log_confirmed
```

In [ ]:
```
temp = df.groupby(['country', 'province'])['confirmed', 'deaths', 'recovere
temp['global_mortality'] = temp['deaths']/temp['confirmed']
temp['deaths_per_100_confirmed_cases'] = temp['global_mortality']*100
temp.style.background_gradient(cmap='Pastel1')
temp.sample(10)
```

In [ ]:
```
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)

temp_f = df_grouped.sort_values(by='confirmed', ascending=False)
temp_f = temp_f[['country', 'confirmed', 'active', 'deaths']] #, 'recovered
temp_f = temp_f.reset_index(drop=True)

temp_f.style.background_gradient(cmap="Blues", subset=['confirmed', 'active
            .background_gradient(cmap="Greens", subset=[])\
            .background_gradient(cmap="Reds", subset=['deaths'])
```

In [ ]:
```
temp_flg = temp_f[temp_f['deaths']>0][['country', 'deaths']]
temp_flg['Deaths / 100 Cases'] = round((temp_f['deaths']/temp_f['confirmed'
temp_flg.sort_values('deaths', ascending=False).reset_index(drop=True).styl
```

In [ ]:
```
# World wide

m = folium.Map(location=[0, 0], tiles='cartodbpositron',
               zoom_start=1) #min_zoom=1, max_zoom=4,

full_latest=df

for i in range(0, len(full_latest)):
    folium.Circle(
        location=[full_latest.iloc[i]['latitude'], full_latest.iloc[i]['lon
        color='crimson',
        tooltip =   '<li><bold>Country : '+str(full_latest.iloc[i]['country
                    '<li><bold>Province : '+str(full_latest.iloc[i]['provin
                    '<li><bold>Confirmed : '+str(full_latest.iloc[i]['confi
                    '<li><bold>Deaths : '+str(full_latest.iloc[i]['deaths']
        radius=int(full_latest.iloc[i]['confirmed'])**1.1).add_to(m)
m
```

In [ ]:
```
def sdf(df):
    locations = covid19.getLocations()
    country=[]
    country_code=[]
    province=[]
    latitude=[]
    longitude=[]
    confirmed=[]
    deaths=[]
    recovered=[]
    last_updated=[]
    for i in range(len(locations)):
        column=locations[i]
        country.append(column['country'])
        country_code.append(column['country_code'])
        province.append(column['province'])
        latitude.append(float(column['coordinates']['latitude']))
        longitude.append(float(column['coordinates']['longitude']))
        confirmed.append(str(column['latest']['confirmed']))
        deaths.append(str(column['latest']['deaths']))
        recovered.append(str(column['latest']['recovered']))
        last_updated.append(column['last_updated'])
    df = pd.DataFrame(
        {'country':country,
         'country_code':country_code,
         'province':province,
         'latitude': latitude,
         'longitude':longitude,
         'confirmed':confirmed,
         'deaths':deaths,
         'recovered':recovered,
         'last_updated':last_updated
        })
    df=df[['country_code','country','province','latitude','longitude','conf
    df['last_updated'] = pd.to_datetime(df['last_updated'], format="%Y-%m-%
    return df
```

MANUAL INSERTION OF DATE BELOW!

In [ ]:
```
dfc=sdf(dfc)
map_center = [dfc["latitude"].mean(), dfc["longitude"].mean()]
map_3 = folium.Map(location=[dfc['latitude'].mean(),
                                dfc['longitude'].mean()], #tiles='CartoDB p
                                zoom_start=1)

locations = dfc[['latitude', 'longitude']]
locationlist = locations.values.tolist()

mc = MarkerCluster().add_to(map_3)

for point in range(0, len(locationlist)):
    folium.Marker(locationlist[point], popup=dfc['country'][point]+' '+dfc[

title_html = '''
             <h3 align="center" style="font-size:20px"><b>Clustered Countri
             '''
map_3.get_root().html.add_child(folium.Element(title_html))

map_3.save('html_map_output/'+dtupd+'_covid19locations_clusters.html')
map_3
```

In [ ]:
```
# Confirmed
sst='Countries with Confirmed Cases on %s' %dtupd
fig = px.choropleth(df_grouped, locations="country",
                    locationmode='country names', color="log_confirmed", #n
                    hover_data=['country','confirmed','deaths'], #hover_nam
                    color_continuous_scale="Sunsetdark",
                    title=sst, #Countries with Confirmed Cases'+'_dtupd', #
#                     projection='natural earth',
                    width=800 #,height=800
                   )
fig.update(layout_coloraxis_showscale=False)
fig.show()
```

In [ ]:
```
import plotly

plotly.offline.plot(fig, filename='html_map_output/'+dtupd+'_covid19locatio
```

In [ ]:
```

```