3주차-3

1. 오늘은 무엇을 배우는가?
2. Choropleth map
    2.1. 패키지 불러오기
    2.2. 데이터 불러오기
        2.2.1. 데이터 설명
    2.3. Choropleth 함수
        2.3.1. Choropleth 파라미터 설명
3. 인터랙티브 데이터 : TimeSliderChoropleth
    3.1. 함수 구성
    3.2. 미세먼지 데이터 시각화

1. 오늘은 무엇을 배우는가?

1. 패키지 불러오기
2. Choropleth map(단계 구분도)
    a. 창원 인구 시각화
3. 인터랙티브 데이터 : TimeSliderChoropleth
    a. 2022년 상반기 전국 미세먼지 현황 시각화

2. Choropleth map

- Choropleth map(단계 구분도)
    - Choropleth map은 주제도(thematic map) 중 하나로, 데이터를 색으로 표현한 지도입니다.
    - 인구 밀도, 온도 등 수치상의 비교를 한 눈에 파악하기에 용이합니다.

2.1. 패키지 불러오기

- 데이터를 다루는 데 필요한 패키지를 불러옵니다.

```
import pandas as pd
import folium
from folium.plugins import TimeSliderChoropleth
pd.options.display.float_format = '{:.2f}'.format


# geopandas는 colab에 설치되어 있지 않기 때문에 따로 설치할 필요가 있습니다.
# colab이 아니라 python에서 실습하는 경우 그냥 설치한 다음 import하면 됩니다.
try:
    import geopandas as gpd
except ModuleNotFoundError:
    !pip3 install geopandas --quiet --progress-bar off
    import geopandas as gpd
    print("geopandas 설치 완료.")
```

2.2. 데이터 불러오기

- 창원시 인구 데이터를 불러옵니다.

```
인구 = pd.read_excel("인구.xlsx")
인구["EMD_CD"] = 인구["EMD_CD"].astype("str")
인구
```

- 데이터 출처
  - 경상남도 창원시_읍면동 행정동별 인구(1세 단위)
  - https://bigdata.gyeongnam.go.kr/index.gn?menuCd=DOM_000000114002001000&publicdatapk=15004972

2.2.1. 데이터 설명

- 인구 데이터에 대한 설명입니다.

	EMD_CD	EMD_KOR_NM	한국인-남	한국인-여	외국인-남	외국인-여
0	48121101	북동	25350	24699	73	168
1	48121102	중동	25350	24699	73	168
2	48121103	서상동	25350	24699	73	168
3	48121104	소답동	25350	24699	73	168
4	48121105	도계동	20841	20135	299	332
...	...	...	...	...	...	...
176	48125310	구산면	2164	2008	176	12
177	48125320	진동면	6171	5841	82	90
178	48125330	진북면	1802	1548	274	30
179	48125340	진전면	1882	1941	60	20
180	48127250	내서읍	31522	31283	217	208

181 rows × 6 columns

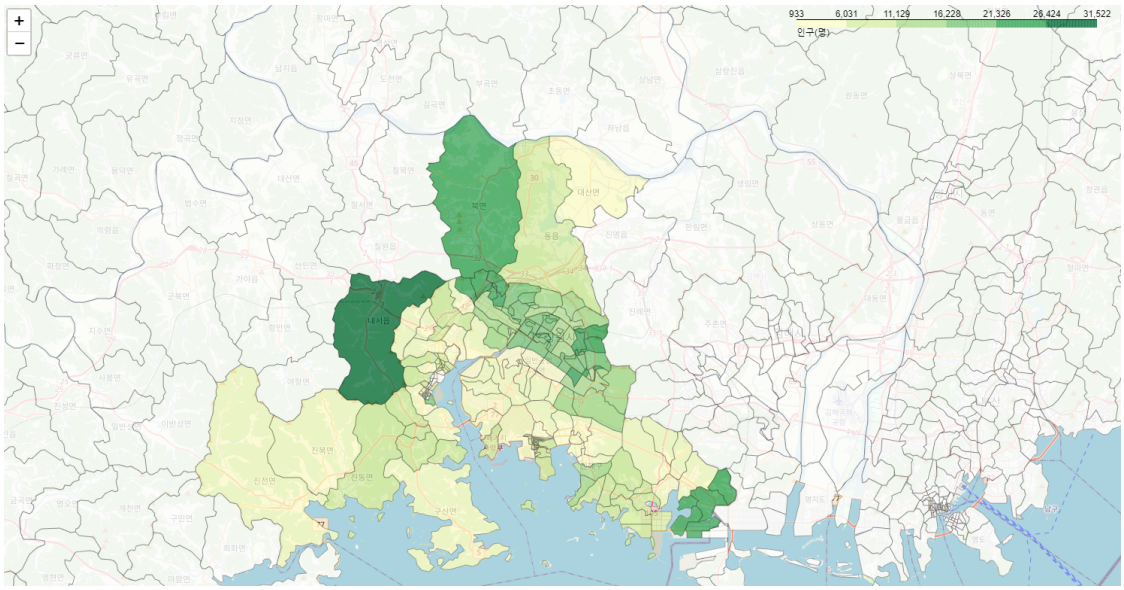
- EMD_CD : 읍면동(EMD) 코드(CD)라는 의미입니다. 부구동, 구산면, 내서읍 등의 지역에 대응하는 코드가 들어있습니다.
- EMD_KOR_NM : 읍면동(EMD) 한글(KOR) 명칭(NM, Name)이라는 의미입니다.
- 한국인-남, 한국인-여, 외국인-남, 외국인-여 : 분류별 인구 수입니다.

2.3. Choropleth 함수

- Choropleth 함수를 통해 Choropleth map을 그립니다.

```
창원광장 = [35.226367, 128.682232]
m = folium.Map(location=창원광장, zoom_start=11)

읍면동_데이터 = "읍면동.json"
지도에_표현할_열 = ["EMD_CD", "한국인-남"]
데이터와_지도_매칭_기준 = "properties.EMD_CD"

folium.Choropleth(
    geo_data=읍면동_데이터, # geo 데이터, .json

    data=인구, # 데이터입니다. pandas로 불러온 데이터를 입력합니다.
    columns=지도에_표현할_열, # 데이터에는 여러 열이 있는데, 그 중에서 사용할 열만 지정합니다.
    key_on=데이터와_지도_매칭_기준, # 지리 데이터와, 인구수 데이터를 매칭할 때 쓸 key를 정합니다.
```

```
    legend_name="인구(명)", # 우상단 막대에 표시할 범례 이름

    fill_color="YlGn", # 색깔 테마입니다. BrBG, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral 등이 있습니다.
    fill_opacity=0.7, # 칠하는 색 투명도입니다. 0이면 색이 안 보입니다.
    line_opacity=0.3, # 지역 간 구분선 투명도입니다.. 0이면 외곽선이 없습니다.

    nan_fill_color = "white", # 빈 공간 색깔입니다.
    nan_fill_opacity = 0.8 # 빈 공간 투명도입니다.
).add_to(m)

m
```



2.3.1. Choropleth 파라미터 설명

- **geo_data**
  - 지리 정보 데이터를 받습니다.
  - folium으로 불러온 지도는 경계를 구분하는 기능이 없습니다.
  - 그래서 특정 경계를 구분해 둔 데이터를 따로 불러와야 하는데, 이를 지리 정보 데이터라고 합니다.
  - 예를 들어 읍면동.json 파일은 읍, 면, 동 단위로 경계를 구분한 데이터입니다.
- **data**
  - 데이터가 들어 있는 변수를 받습니다.
  - 지도에 표현할 데이터가 필요하며, 이에 따라 데이터가 들어 있는 변수를 함수에 제공합니다.
- **columns**
  - key와 value로 사용될 두 데이터 열 이름을 받습니다.
  - value는 인구 밀도, 온도와 같이 지도에 색으로 표현될 데이터 열입니다.
  - key는 데이터와 지리 정보 데이터를 연결해 줄 데이터 열입니다.
    - key에 '창원시'라는 글자가 있고, geo_data에도 '창원시'라는 데이터가 있으면 두 데이터가 매핑되는 방식입니다.
    - 지금 사용할 데이터는 EMD_CD를 기준으로 매핑하게 됩니다.
- **key_on**

- **지리 정보**와 **데이터**를 서로 연결할 데이터를 받습니다.
  읍면동_데이터.json는 다음과 같은 데이터로 구성됩니다.


```
{"type":"Feature","geometry":{"type":"Polygon","coordinates":[[[126.97555981186262,37.58968065072607],[126.97358712298828,37.59327062428288],[126.97189481294204,37.593875276725],[126.96658843417845,37.592971994440549],[126.96419540977288,37.589057330358386],[126.96143497169824,37.586582373988556],[126.96504416132 12,37.58558542366072],[126.96914890505388,37.585844248418084],[126.97031416603146,37.58417632420739],[126.9739984733833,37.58654064299155],[126.97555981186262,37.58968065072607]]]},"properties":{"EMD_CD":"11110101","EMD_ENG_NM":"Cheongun-dong","EMD_KOR_NM":"청운동"}},
```


  - **columns**에서 매핑할 데이터 열로 EMD_CD를 설정했기 때문에, json 파일에서도 EMD_CD가 있는 부분을 명시해주어야 합니다.
  - "properties":{"EMD_CD"… 의 형태로 EMD_CD를 발견할 수 있으므로, key_on에는 "properties.EMD_CD"의 값을 넘겨주면 됩니다.
- legend_name="인구(명)"

  - 우상단에 나타나는 범례의 이름입니다.
- fill_color="YlGn",
  fill_opacity=0.7,
  line_opacity=0.3,
  - 색 설정입니다.
- nan_fill_color = "white",
  nan_fill_opacity = 0.8
  - 데이터가 없는 부분의 색 설정입니다.

3. 인터랙티브 데이터 : TimeSliderChoropleth

- TimeSliderChoropleth
  - TimeSliderChoropleth 함수는 시계열 데이터의 값 변화를 확인할 수 있는 folium 플러그인 입니다

3.1. 함수 구성

- TimeSliderChoropleth(data, styledict)
  - **data** : 지리 정보 데이터입니다. 2. Choropleth map 함수에서 썼던 그 지리 정보와 같은 파일을 사용하면 되나, to_json() 함수를 통해 변환하는 작업이 필요합니다.
  - **styledict** : 색 정보가 들어있는 데이터입니다. 날짜의 변화에 따른 색 변화를 해당 데이터에서 읽어 출력하게 됩니다.

3.2. 미세먼지 데이터 시각화

- 데이터를 읽어옵니다.

```
지리_데이터 = gpd.read_file('시도.json', encoding="euc_kr") # 시도, 시군구, 읍면동
```

```
미세먼지 = pd.read_excel("대기오염_PM10_통계 현황_2022.xlsx")
미세먼지.head()
```

- 데이터를 함수 형식에 맞게 전처리합니다.

```
미세먼지["날짜"] = 미세먼지["날짜"].astype("datetime64[ns]")
미세먼지["날짜"] = 미세먼지["날짜"].astype(int) // 10 ** 9
미세먼지["날짜"] = 미세먼지["날짜"].astype("str")
미세먼지 = 미세먼지.set_index("날짜")

색_데이터 = {}
빨간색 = "#fc2803"

시도별_최댓값 = 미세먼지.max()
전국_최댓값 = max(시도별_최댓값)

for i in range(len(지리_데이터)):
    색_데이터[str(i)] = {}
    for day in 미세먼지.index:
        해당_날짜_미세먼지 = 미세먼지.loc[day][i]
        색_데이터[str(i)][str(day)]= {'color': 빨간색, 'opacity': 해당_날짜_미세먼지 / 전국_최댓값}
```

  - 미세먼지["날짜"].astype("datetime64[ns]")
    미세먼지["날짜"] = 미세먼지["날짜"].astype(int) // 10 ** 9
    미세먼지["날짜"] = 미세먼지["날짜"].astype("str")
    미세먼지 = 미세먼지.set_index("날짜")
    - TimeSliderChoropleth에서 날짜는 Unix Time을 받습니다.
    - 날짜 데이터는 **특정 기준으로 몇 초가 지났는지**를 숫자 데이터로 표현할 수 있습니다.
    - **1970년 1월 1일 기준**으로 몇 초가 지났는지를 표시하는 방식을 Unix Time이라고 부릅니다.
    - Unix Time을 계산하려면 날짜를 정수로 변환한 뒤 10의 9승(10 ** 9)으로 나누면 됩니다.
      - https://stackoverflow.com/questions/54312802/pandas-convert-from-datetime-to-integer-timestamp
    - 이렇게 변환된 시간을 인덱스로 설정합니다.
  - 시도별_최댓값 = 미세먼지.max()
    전국_최댓값 = max(시도별_최댓값)
    - 최댓값을 구하는 이유는, 이후 불투명도opacity를 계산할 때 사용하기 때문입니다.
  - 색 데이터 준비하기
    색 데이터는 아래와 같은 형태로 준비됩니다.

```
{'0': {'1642982400': {'color': 빨간색, 'opacity': 0.1510791366906475},
       '1643068800': {'color': 빨간색, 'opacity': 0.2589928057553957},
       '1643155200': {'color': 빨간색, 'opacity': 0.4316546762589928},
       '1643241600': {'color': 빨간색, 'opacity': 0.45323741007194246},
       '1643328000': {'color': 빨간색, 'opacity': 0.2733812949640288},
        ....
      },
 '1': {'1642982400': {'color': 빨간색, 'opacity': 0.1510791366906475},
       '1643068800': {'color': 빨간색, 'opacity': 0.20863309352517986},
       '1643155200': {'color': 빨간색, 'opacity': 0.4028776978417266},
       '1643241600': {'color': 빨간색, 'opacity': 0.3597122302158273},
       '1643328000': {'color': 빨간색, 'opacity': 0.12949640287769784},
        ....
      }, ... 계속됨
```

- 중첩된 딕셔너리 자료형 형태이며, 총 0~16까지의 값을 가집니다.
  - 대한민국의 시, 도가 총 17개이기 때문에 0~16까지입니다.
- 0~16 각각에는 { 날짜 : {색깔, 불투명도}} 형태의 딕셔너리가 존재합니다.

```
'1642982400': {'color': 빨간색, 'opacity': 0.1510791366906475}
```

  - 1642982400 : 날짜입니다. 2022년 1월 24일을 정수로 바꾸고 10의 9승으로 나눈 값입니다.
  - 'opacity': 0.1510791366906475 : 해당_날짜_미세먼지 / 전국_최댓값의 값입니다. 0에 가까울수록 연한 색이 출력됩니다.

- 함수에 준비한 값들을 넣어줍니다.

```
대한민국 = [37, 127]
m = folium.Map(location=대한민국, width=1500, height=600)
TimeSliderChoropleth(data=지리_데이터.to_json(), styledict=색_데이터).add_to(m)

m
```