

Ch11 지도 시각화

지도에 지역별 통계적 특징을 색깔로 표현

11-1 시군구별 인구 단계 구분도 만들기

[실습] 시군구별 인구 단계 구분도 만들기

> 단계 구분도(choropleth map)는 지도에 지역별 통계치를 색깔 차이로 표현한 지도

In []:

1. 시군구 경계 데이터 준비하기

> SIG.geojson 파일이 대한민국 시군구별 경계 좌표를 제공함

In [24]: ## SIG.geojson 파일 데이터를 딕셔너리로 만들기

```
import json
geo = json.load(open('SIG.geojson', encoding = 'UTF-8'))
type(geo)
```

Out[24]: dict

In [25]: ## geo 딕셔너리를 리스트로 읽어내기

```
geo_list = list((k, v) for k, v in geo.items())
```

In [26]: ## geo의 리스트에서 앞 3개 item만 확인

```
geo_list[:3]
```

Out[26]: [('type', 'FeatureCollection'),

```
    ('name', 'sig'),
```

```
    ('crs',
```

```
        {'type': 'name', 'properties': {'name': 'urn:ogc:def:crs:OGC:1.3:CRS84'}})]
```

geo 딕셔너리 데이터 구성

```
{'type': 'FeatureCollection',
```

```
    'name': 'sig',
```

```
    'crs': {'type': 'name', 'properties': {'name': 'urn:ogc:def:crs:OGC:1.3:CRS84'}},
```

```
    'features': [ {'type': 'Feature', 'properties': {'SIG_CD': '42110', 'SIG_ENG_NM': 'Chuncheon-si', 'SIG_KOR_NM': '춘천시'},
```

```
        'geometry': {'type': 'MultiPolygon', 'coordinates': [[[127.58508551154958, 38.08062321552708], ... ]}}
```

In [27]: ## 제일 앞에 있는 행정 구역 정보 출력

```
geo['features'][0]['properties']
```

Out[27]: { 'SIG_CD': '42110', 'SIG_ENG_NM': 'Chuncheon-si', 'SIG_KOR_NM': '춘천시' }

```
In [28]: ## 앞에서 두번째에 있는 행정 구역 정보 출력  
geo['features'][1]['properties']
```

```
Out[28]: {'SIG_CD': '42130', 'SIG_ENG_NM': 'Wonju-si', 'SIG_KOR_NM': '원주시'}
```

```
In [30]: ## 행정구역의 위도, 경도 좌표값 확인  
geo['features'][0]['geometry']['coordinates'][0][0][0]
```

```
Out[30]: [127.58508551154958, 38.08062321552708]
```

```
In [31]: ## 위도, 경도 좌표값 출력 : 5개만 출력  
for i in range(5):  
    print(geo['features'][0]['geometry']['coordinates'][0][0][i])
```

```
[127.58508551154958, 38.08062321552708]  
[127.58565575732702, 38.0802009066172]  
[127.58777905808203, 38.080354190085544]  
[127.58890487394689, 38.080881783588694]  
[127.59031267326897, 38.080596307998306]
```

```
In [ ]:
```

2. 시군구별 인구 데이터 준비하기

```
In [32]: ## 시군구별 인구 통계 데이터 만들기  
import pandas as pd  
df_pop = pd.read_csv('Population_SIG.csv')  
df_pop
```

```
Out[32]:   code      region  pop  
  0     11    서울특별시  9509458  
  1   11110        종로구  144683  
  2   11140        중구  122499  
  3   11170        용산구  222953  
  4   11200        성동구  285990  
 ...     ...        ...    ...  
273   48880        거창군   61073  
274   48890        합천군   42935  
275     50  제주특별자치도  676759  
276   50110        제주시  493096  
277   50130        서귀포시  183663
```

278 rows × 3 columns

```
In [33]: ## 통계 데이터 정보 확인  
# 구성: 행정구역코드, 지역, 인구  
df_pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 3 columns):
 # ... Column Non-Null Count Dtype 
 --- ... -- -- -- -- -- -- -- -- 
 0 ... code    278 non-null   int64 
 1 ... region  278 non-null   object 
 2 ... pop     278 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 6.6+ KB
```

```
In [34]: ## 행정구역코드(code 열) 데이터 타입 변경
# int > str
df_pop['code'] = df_pop['code'].astype(str)
df_pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 3 columns):
 # ... Column Non-Null Count Dtype 
 --- ... -- -- -- -- -- -- -- -- 
 0 ... code    278 non-null   object 
 1 ... region  278 non-null   object 
 2 ... pop     278 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 6.6+ KB
```

```
In [35]: ## 행정구역코드 Key로 행정구역 확인
# 지도 데이터의 SIG_CD : '42110'로 인구 데이터에서 확인
df_pop[df_pop['code'] == '42110']
```

```
Out[35]:   code  region  pop
133  42110  춘천시  284594
```

```
In [ ]:
```

3. 단계 구분도 만들기

folium 라이브러리 사용

> pip install folium 필요

> folium는 인터랙티브한 지도를 생성하고 사용할 수 있게 지원

> folium.Choropleth()

>> Choropleth()는 지도 데이터와 통계 데이터를 연계하여 시각화하는 메서드

>> geo_data: GeoJSON 또는 TopoJSON 형식의 지도 데이터

>> data: 시각화하려는 실제 데이터

>> columns: 첫 번째 열은 데이터를 시각화할 때 사용할 값 열, 두 번째 열은 색상 매핑을 위한 열

>> key_on: geo_data 데이터와 지리적 데이터를 매핑하는 키

```
pip install folium
```

(1) 배경 지도 만들기

```
In [39]: ## folium으로 지도 표현하기
import folium
folium.Map(location = [35.95, 127.7], # 지도 중심 좌표
           zoom_start = 8) # 확대 단계
```

Out[39]: Make this Notebook Trusted to load map: File -> Trust Notebook

```
In [49]: ## folium으로 지도 객체 표현하기 : 밝은 지도
map_sig = folium.Map(location = [35.95, 127.7], # 지도 중심 좌표
                      zoom_start = 8, # 확대 단계
                      tiles = 'CartoDB Positron') # 지도 종류
map_sig
```

Out[49]: Make this Notebook Trusted to load map: File -> Trust Notebook

folium.Map()에서 tiles의 종류

- > "OpenStreetMap" (기본값): 기본 배경 지도
- > "Mapbox Control Room": Mapbox의 스타일링 및 맞춤형 지도
- > "Stamen Terrain": Stamen Design에서 만든 지형 및 지형 데이터를 기반으로 한 스타일

- > "Stamen Toner": Stamen Design에서 만든 흑백 스타일
- > "Stamen Watercolor": 수채화 스타일로 묘사된 스타일
- > "CartoDB Positron": CartoDB에서 제공하는 간결한 스타일
- > "CartoDB Dark Matter": CartoDB에서 제공하는 어두운 스타일
- > "Esri": Esri ArcGIS Online 지도 서비스를 기반으로 하는 스타일

(2) 단계 구분도 만들기

```
In [50]: ## 단계 구분도 만들기
# df_pop 행정 구역 코드, 인구
# 행정구역코드(SIG_CD) 기준
folium.Choropleth(geo_data = geo,
                    data = df_pop,
                    columns = ('code', 'pop'),
                    #지도 데이터에서 행정구역코드(SIG_CD) Key 지정
                    key_on = 'feature.properties.SIG_CD') \
                    .add_to(map_sig)

map_sig
```

Out[50]: Make this Notebook Trusted to load map: File -> Trust Notebook

(3) 계급 구간 정하기

```
In [51]: ## 인구 수를 계급 구간으로 등분하는 기준 리스트(bins용) 만들기
# quantile() 활용 0~1.0까지 비율 경계 값으로 분할
bins = list(df_pop['pop'].quantile([0, 0.25, 0.5, 0.75, 1]))
```

Out[51]: [8867.0, 62658.5, 210067.5, 374649.75, 13565450.0]

(4) 디자인 수정하기

```
In [53]: ## 배경 지도 만들기
import folium
map_sig = folium.Map(location = [35.95, 127.7],
                      zoom_start = 7,
```

```

    tiles = 'cartodbpositron')

## 단계 구분도 만들어 배경 지도에 더하기
# 지도 데이터에 통계치 데이터 연결, 디자인 특성 반영
folium.Choropleth(geo_data = geo,
                    data = df_pop,
                    columns = ('code', 'pop'),
                    key_on = 'feature.properties.SIG_CD',
                    fill_color = 'YIGnBu', # 컬러맵
                    fill_opacity = 1, # 투명도
                    line_opacity = 0.5, # 경계선 투명도
                    bins = bins) # 계급 구간 기준값
                    .add_to(map_sig)

map_sig

```

Out[53]: Make this Notebook Trusted to load map: File -> Trust Notebook

folium.Choropleth() fill_color 매개변수로 사용 가능한 colormap

"BuGn": 파란색에서 녹색으로 변화하는 colormap.

"YlOrRd": 노란색에서 주황색에서 빨간색으로 변화하는 colormap.

"GnBu": 녹색에서 파란색으로 변화하는 colormap.

"YIGnBu": 노란색에서 녹색에서 파란색으로 변화하는 colormap.

"YlOrBr": 노란색에서 주황색에서 갈색으로 변화하는 colormap.

"Reds": 빨간색 계열의 colormap.

"Blues": 파란색 계열의 colormap.

"Greens": 녹색 계열의 colormap.

"Purples": 보라색 계열의 colormap.

"Oranges": 주황색 계열의 colormap.

In []:

11-2 서울시 동별 외국인 인구 단계 구분도 만들기

[실습] 서울시 동별 외국인 인구 단계 구분도 만들기

1. 서울시 동 경계 지도 데이터 준비하기

> EMD_Seoul.geojson 파일은 서울시의 동별 행정구역 코드, 동 이름, 동 경계 위도, 경도 데이터를 제공

```
In [54]: ## EMD_Seoul.geojson 파일로부터 서울시 동별 지도 데이터 만들기
import json
geo_seoul = json.load(open('EMD_Seoul.geojson', encoding = 'UTF-8'))
type(geo_seoul)
```

```
Out[54]: dict
```

```
In [55]: ## 딕셔너리를 리스트로 읽어내기
geo_seoul_list = list((k, v) for k, v in geo_seoul.items())
```

```
In [56]: geo_seoul_list[:3]
```

```
Out[56]: [('type', 'FeatureCollection'),
('name', 'seoul5'),
('crs',
... {'type': 'name', 'properties': {'name': 'urn:ogc:def:crs:OGC:1.3:CRS84'}})]
```

geo_seoul 딕셔너리 데이터 구성

```
{'type' : 'FeatureCollection',
'name' : 'sig',
'crs' : {'type': 'name', 'properties': {'name': 'urn:ogc:def:crs:OGC:1.3:CRS84'}},
'features' : [{"type': 'Feature', 'properties': {'BASE_DATE': '20200630', 'ADM_DR_CD': '1101053', 'ADM_DR_NM': '사직동', 'OBJECTID': '1'},
'geometry': {'type': 'MultiPolygon', 'coordinates': [[[126.97398562200112, 37.578232670691676], ... ]}}
```

```
In [57]: ## 행정 구역 코드 출력
geo_seoul['features'][0]['properties']
```

```
Out[57]: {'BASE_DATE': '20200630',
'ADM_DR_CD': '1101053',
'ADM_DR_NM': '사직동',
'OBJECTID': '1'}
```

```
In [58]: ## 위도, 경도 좌표 출력 : 일부 출력
geo_seoul['features'][0]['geometry']['coordinates'][0][0][0]
```

```
Out[58]: [126.97398562200112, 37.578232670691676]
```

```
In [ ]:
```

2. 서울시 동별 외국인 인구 데이터 준비하기

```
In [59]: ## 서울시 동별 외국인 인구 통계 데이터 만들기
foreigner = pd.read_csv('Foreigner_EMD_Seoul.csv')
```

```
foreigner.head()
```

```
Out[59]:
```

	code	region	pop
0	1101053	사직동	418.0
1	1101054	삼청동	112.0
2	1101055	부암동	458.0
3	1101056	평창동	429.0
4	1101057	무악동	102.0

```
In [60]: ## 통계 데이터 정보 확인  
foreigner.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3490 entries, 0 to 3489  
Data columns (total 3 columns):  
 # ... Column Non-Null Count Dtype ...  
 --- ...  
 0 code    3490 non-null int64  
 1 region  3490 non-null object  
 2 pop     3486 non-null float64  
dtypes: float64(1), int64(1), object(1)  
memory usage: 81.9+ KB
```

```
In [61]: ## 행정구역코드(code 열) 데이터 타입 변경  
# int > str  
foreigner['code'] = foreigner['code'].astype(str)  
foreigner.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3490 entries, 0 to 3489  
Data columns (total 3 columns):  
 # ... Column Non-Null Count Dtype ...  
 --- ...  
 0 code    3490 non-null object  
 1 region  3490 non-null object  
 2 pop     3486 non-null float64  
dtypes: float64(1), object(2)  
memory usage: 81.9+ KB
```

```
In [62]: ## 행정구역코드 Key로 행정구역 확인  
# 지도 데이터의 'ADM_DR_CD' : '1101053'로 인구 데이터에서 확인  
foreigner[foreigner['code'] == '1101053']
```

```
Out[62]:
```

	code	region	pop
0	1101053	사직동	418.0

```
In [ ]:
```

3. 단계 구분도 만들기

```
In [63]: ## 인구 수를 계급 구간으로 등분하는 기준 리스트(bins용) 만들기  
# quantile() 활용 0~1.0까지 비율 경계 값으로 분할  
bins = list(foreigner['pop'].quantile([0, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]))  
bins
```

```
Out[63]: [7.0, 98.0, 200.0, 280.0, 386.0, 529.5, 766.0, 1355.5, 26896.0]
```

```
In [74]: ## 배경 지도 만들기
map_seoul = folium.Map(location = [37.56, 127],
                      zoom_start = 12,
                      tiles = 'cartodbpositron')

## 단계구분도 만들기
folium.Choropleth(geo_data = geo_seoul,
                   data = foreigner,
                   columns = ('code', 'pop'),
                   key_on = 'feature.properties.ADM_DR_CD',
                   fill_color = 'Blues',
                   nan_fill_color = 'White',
                   fill_opacity = 1,
                   line_opacity = 0.5,
                   bins = bins) \
.add_to(map_seoul)

map_seoul
```

Out[74]: Make this Notebook Trusted to load map: File -> Trust Notebook

In []:

4. 구 경계선 추가하기

```
In [72]: ## 서울시 구 경계 좌표 데이터 만들기
geo_seoul_sig = json.load(open('SIG_Seoul.geojson', encoding = 'UTF-8'))
```

In [75]: ## 서울 구 라인 추가 : 이전의 map_seoul 지도에 추가

```
folium.Choropleth(geo_data = geo_seoul_sig,
                   fill_opacity = 0, #투명도, 선 두께
                   line_weight = 2) \
.add_to(map_seoul) #이전의 map_seoul 지도에 추가
map_seoul
```

Out[75]: Make this Notebook Trusted to load map: File -> Trust Notebook

In []:

(알아 두면 좋아요) folium 활용하기

> folium 라이브러리의 save() 메서드를 활용

HTML 파일로 저장하기

In [120...]

```
map_seoul.save('map_seoul.html')
```

웹 브라우저에서 html 파일 열기

In [122...]

```
import webbrowser as wb
wb.open_new('map_seoul.html')
```

Out[122]:

```
True
```

In []: