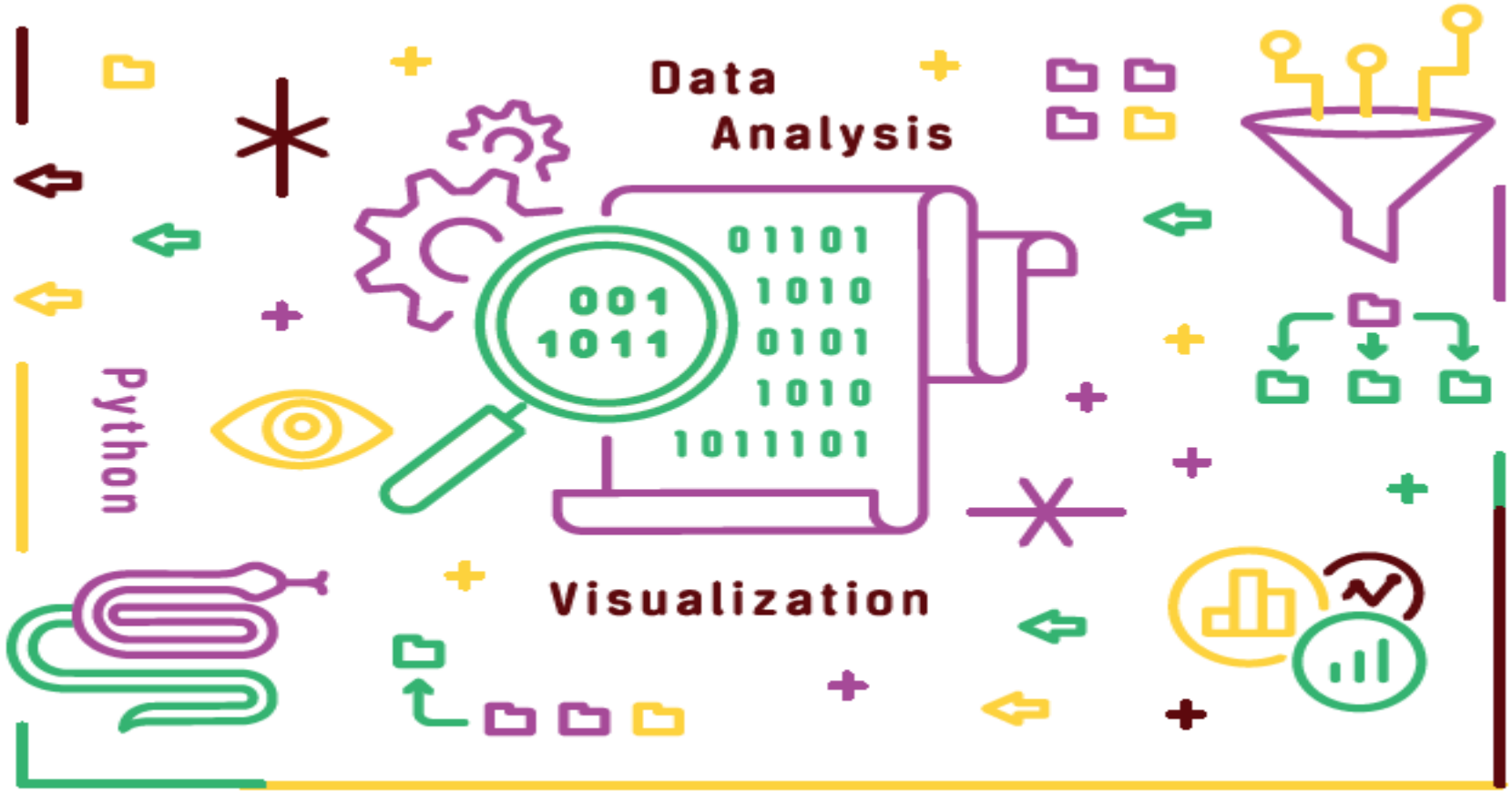


<8장> 지리정보 분석 및 시각화

학습 목표

- folium 라이브러리를 사용하여 기본 지도 생성 방법을 익힌다.
- 지도에 특정좌표에 마커를 추가하여 위치 정보를 시각화 할 수 있다
- 수집한 주소 데이터를 정리하고 분석하여 시각화할 수 있다.
- 주소를 분석하여 위도와 경도의 GPS 정보를 찾아낼 수 있다.
- 행정구역을 찾아서 지도에 나타난 정보를 분석할 수 있다.

목차

01

folium

1. folium

- 포리움(Folium)
  - 참조 사이트 : http://python-visualization.github.io/folium/
  - Python에서 Leaflet.js 기반의 인터랙티브한 지도를 쉽게 만들 수 있게 해주는 시각화 라이브러
  - Python으로 데이터를 조작한 다음, Folium을 통해 리플릿 맵에서 시각화
  - 위도(latitude) : 적도를 기준으로 남쪽으로 남극점까지 90˚, 북쪽으로 북극점까지 90˚로 나누어 표시(우리나라 적도의 북쪽인 북위 34˚ ~ 38˚ 사이에 위치)
  - 경도(longitude) : 런던 그리니치 천문대를 지나는 본초 자오선을 중심으로 동서로 나누어 동경 180˚, 서경 180˚로 분리 (서울의 경우 동경 127˚에 위치).

- **포리움 설치**

```
pip install folium # folium 설치
```

1. folium

- Folium 설치 및 라이브러리 로드

```
! pip install folium  # folium 설치
import folium
import numpy as np
import pandas as pd
```

- Folium 객체 생성

```
latitude=37.394945  # 위도
longitude=127.111104   #경도
m = folium.Map(location=[latitude, longitude])
m.save('data/map1.html') ; m  #파일이 저장하고 화면에 표시
```

- zoom_start 속성

```
#zoom_start : 지도 zoom 확대/축소 상태, 범위 :0~19, default :10
m = folium.Map([latitude, longitude],  zoom_start=11)
m=folium.Map(location=[latitude,longitude], width=800, height=800)
m.save(os.path.join('results',  ' map2.html')) ; m
```

1. folium

- 맵의 유형
  - 포리움은 기본적으로 'OpenStreetMap'을 기반으로 동작
  - '지도의 테마 스타일 ['cartodb positron','Cartodb dark_matter', 'OpenStreetMap']

```
m = folium.Map(location=[37.566345, 126.977893], zoom_start=10, tiles=OpenStreetMap')
m.save('data_al/map3.html')
m = folium.Map(location=[37.566345, 126.977893], zoom_start=17, tiles= 'cartodb positron')
m.save('data_al/map4.html')
```

1. folium

- 맵의 유형
  - ImageOverlay 사용

```
m = folium.Map([latitude, longitude], zoom_start=5)
folium.raster_layers.ImageOverlay(
    image="https://upload.wikimedia.org/wikipedia/commons/f/f4/Mercator_projection_SW.jpg",
    name="I am a jpeg",
    bounds=[[-82, -180], [82, 180]],
    opacity=1,
    interactive=False,
    cross_origin=False,
    zindex=1,
    alt="Wikipedia File:Mercator projection SW.jpg",
).add_to(m)
folium.LayerControl().add_to(m) ; m
```

1. folium

- 마커(Marker)와 팝업(Popup), 툴팁(tooltip)의 설정
  - 포리움은 다양한 형식의 마커(특정 위치를 표시하는 표식)과 마커를 클릭하였을 때 나타나는 정보(Popup)을 지정할 수 있다.
  - Marker()' 메소드를 이용하여 생성
  - 마커의 인자 값으로 위경도 값 리스트와 마커를 클릭할 시 보여줄 문자열을 전달하고, 생성한 포리움 객체에 추가(.add_to()) 하면 간단하게 마커를 생성

```
m = folium.Map(location=[37.566345, 126.977893], zoom_start=17)
folium.Marker([37.566345, 126.977893], popup='서울특별시청').add_to(m)
folium.Marker([37.5658859, 126.9754788], popup='덕수궁').add_to(m)
m.save('data_al/map5.html');m
```

1. folium

- 마커(Marker)와 팝업(Popup), 툴팁(tooltip)의 설정

```
# 마크 그리기
tooltip="클릭해주세요"
m=folium.Map(location=[35.1605598,129.0560362], zoom_start=17)
folium.Marker([35.1625598,129.0560362], popup='<i>위</i>',tooltip=tooltip).add_to(m)
folium.Marker([35.1615598,129.0560362], popup='<i>가운데</i>',tooltip=tooltip).add_to(m)
folium.Marker([35.1605598,129.0560362], popup='<i>아래</i>',tooltip=tooltip).add_to(m)
m
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - 포리움 마커는 부트스트랩(bootstrap)을 이용, 아이콘 타입을 설정할 수 있으며, 범위를 설정하기 위하여 circle 속성을 줄 수 있다.
  - 덕수궁의 위치를 좀더 크게 마커로 표시하고, 서울특별시청은 적색의 'info-sign' 마커로 표시한 예

```
m=folium.Map(location=[37.566345, 126.977893], zoom_start=17)
folium.Marker([37.566345, 126.977893], popup='서울특별시청',
icon=folium.Icon(color='red',icon='info-sign')).add_to(m)
folium.CircleMarker([37.5658859, 126.9754788], radius=100,
color='#3186cc', fill_color='#3186cc', popup='덕수궁').add_to(m)
m.save('data/map6.html');m
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - 아이콘(Icon) 설정

```
tooltip="클릭해주세요"
m=folium.Map(location=[35.1605598,129.0560362], zoom_start=17, tiles='Stamen Toner')
folium.Marker([35.1625598,129.0560362],icon=folium.Icon(icon='cloud'),
              popup='<i>위</i>',tooltip=tooltip).add_to(m)
folium.Marker([35.1615598,129.0560362], icon=folium.Icon(icon='star',color='green', icon_color='yellow'),
              popup='<i>가운데</i>',tooltip=tooltip).add_to(m)
folium.Marker([35.1605598,129.0560362], icon=folium.Icon(icon='info-sign',color='red'),
              popup='<i>아래</i>',tooltip=tooltip).add_to(m)
m
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - 아이콘(Icon) 정보보기

```
help(folium.Icon)
```

  - 아이콘 파라미터
    - color
    - icon_color
    - icon
    - 아이콘 이미지 제공 URL https://github.com/lvoogdt/Leaflet.awesome-markers

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - 다양한 Mark 아이콘(Icon) 사용

```
tooltip="클릭해주세요"
m=folium.Map(location=[lat1, lng1], zoom_start=17)
folium.Marker([lat1, lng1], popup='서울시청', tooltip=tooltip,
              icon=folium.Icon(color='red', icon='info-sign', icon_color='orange')).add_to(m)
folium.CircleMarker([37.5658859, 126.9754788],radius=100, color="#3186cc",
                    fill_color='#cc8631',popup='덕수궁').add_to(m)
folium.Marker([37.5637859, 126.9743788], tooltip=tooltip, icon=folium.Icon(icon='star',
color='green',icon_color='red')).add_to(m)
folium.Marker([37.5667859, 126.9763788], tooltip=tooltip, icon=folium.Icon(icon='home')).add_to(m)
folium.Marker([37.5667859, 126.9743788], tooltip=tooltip, icon=folium.Icon(icon='cloud')).add_to(m)
folium.Marker([37.5637859, 126.9763788], tooltip=tooltip, icon=folium.Icon(icon='glass')).add_to(m)
m
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - MarkerCluster() 사용

```
N = 100
data = np.array(
    [
        np.random.uniform(low=35, high=38, size=N), # Random latitudes.
        np.random.uniform(low=125, high=129, size=N), # Random longitudes.
    ] ).T
popups = [str(i) for i in range(N)] # Popups texts are simple numbers.
m = folium.Map([37.566345, 126.977893], zoom_start=8)
plugins.MarkerCluster(data, popups=popups).add_to(m)
m.save(os.path.join('results', 'Plugins_1.html'))
m
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - ClickForMark() 사용

```
m.add_child(folium.ClickForMarker(popup='마커'))
m
```

  - 파업에 위도 경도 표시

```
m.add_child(folium.LatLngPopup())
```

1. folium

- 마커(Marker)와 팝업(Popup) 설정
  - MarkerCluster() 사용

```
N = 100
data = np.array(
    [
        np.random.uniform(low=35, high=38, size=N), # Random latitudes.
        np.random.uniform(low=125, high=129, size=N), # Random longitudes.
    ] ).T
popups = [str(i) for i in range(N)] # Popups texts are simple numbers.
m = folium.Map([37.566345, 126.977893], zoom_start=8)
plugins.MarkerCluster(data, popups=popups).add_to(m)
m.save(os.path.join('results', 'Plugins_1.html'))
m
```

1. folium

- 실습1
  - 부산진구 CCTV 설치 위치 마크 표시

```
df=pd.read_csv('data/CCTV_20190917.csv')
df
```

```
df1=df.loc[df['카메라대수']>=1,:]
df1
```

```
popup=list(df1['소재지도로명주소'])
#print(df1['위도'])
lat_avg=np.array(list(df1['위도'])).mean()
lng_avg=np.array(list(df1['경도'])).mean()
data=np.array([list(df1['위도']),
list(df1['경도'])]).T
print(popup)
print(data)
print(lat_avg, lng_avg)
```

```
m=folium.Map(location=[lat_avg, lng_avg], zoom_start=15)
plugins.MarkerCluster(data, popups=popup).add_to(m)
m.save(os.path.join('data','cctvmap.html'))
m
```

02

지리정보 분석 후 맵 생성

2. 지리정보 분석 후 맵 생성

- **단계구분도** (choropleth map)
  - 색상이나 패턴을 사용하여 특정 통계에 대한 데이터를 사전 정의된 영역과 관련시켜 시각화 한 지도 유형
  - 지도 시각화는 점 데이터로 표현된 정보 보다는, 특정 구역에 대한 통계 데이터를 시각화 하는 데에 적절
  - 예 : 서울시 구 별 교통사고 건수, 구 별 소득, 등등 '지역 별 통계'를 지도에 시각화

2. 지리정보 분석 후 맵 생성

- choropleth map

```
import requests
import json

# 서울 행정구역 json raw파일(githubcontent)

url='https://raw.githubusercontent.com/southkorea/seoul-maps/master/kostat/2013/json/seoul_municipalities_geo_simple.json'
r = requests.get(url)
c = r.content
seoul_geo = json.loads(c)
seoul_geo
```

```
map=folium.Map(location=[37.559819, 126.96389], zoom_start=11)
folium.GeoJson(seoul_geo, name='지역구').add_to(map)
map
```

2. 지리정보 분석 후 맵 생성

- choropleth map
  - coffeBean 매장의 수로 choropleth map 그리기

```
cb=pd.read_csv('data/CoffeeBean.csv')
cb
```

```
addrs=list(cb['address'])
addr_list=[]
for addr in addrs:
    addr_list.append(addr.split()[:2])
addr_list
df=pd.DataFrame(addr_list, columns=['sido','gugun'])
df
```

```
addr_df=pd.DataFrame(cb['address'].apply(lambda v: v.split()[:2])
                     .to_list(),columns=['시도','구군'])
addr_df
```

2. 지리정보 분석 후 맵 생성

- choropleth map
  - coffeBean 매장의 수로 choropleth map 그리기

```
addr_df['시도'].unique()
```

```
#시도명 정규화
addr_aliase={'서울':'서울특별시','서울시':'서울특별시'}
addr_df['시도']=addr_df['시도'].apply(lambda
v:addr_aliase.get(v,v))
addr_df['시도'].unique()
```

```
# 서울지역 데이터 추출
addr_df2=addr_df.loc[addr_df.시도=='서울특별시']
addr_df2
```

```
# groupby()사용하여 지역의 개수 구함
addr_group=addr_df2.groupby('구군')['시도'].count()
addr_group
```

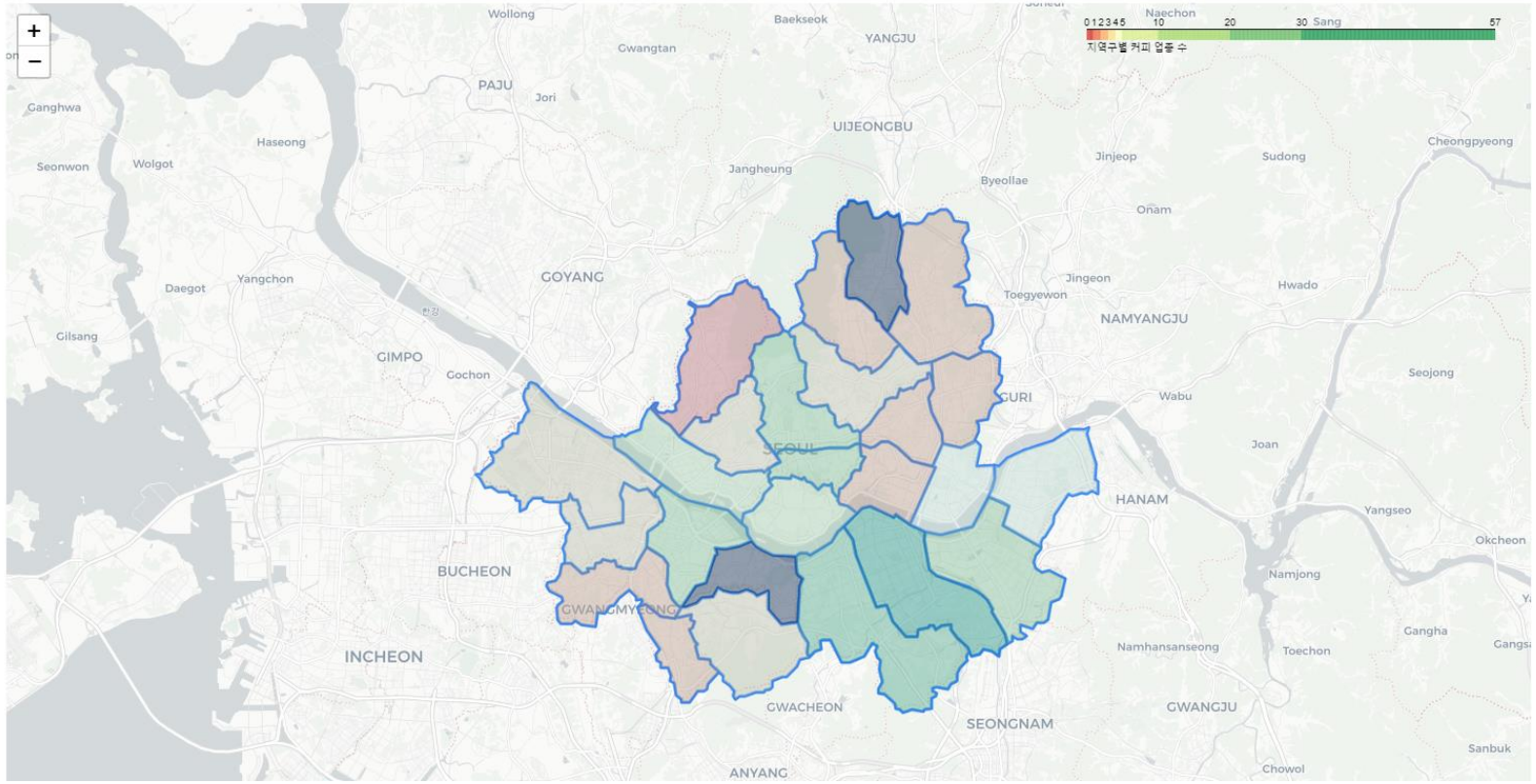
2. 지리정보 분석 후 맵 생성

- choropleth map
  - coffeBean 매장의 수로 choropleth map 그리기

```
# 차트 작성
map=folium.Map(location=[37.559819, 126.96389], zoom_start=11)
folium.GeoJson(seoul_geo, name='지여구').add_to(map)
folium.Choropleth(geo_data=seoul_geo,
                  data=addr_group,
                  fill_color='YlOrRd',
                  bins=[0,2,5,10,15,20,25,57],
                  fill_opacity=0.6,
                  line_opacity=0.2,
                  key_on='properties.name',
                  legend_name='서울특별시 구별 커피빈 매장 수').add_to(map)
```

2. 지리정보 분석 후 맵 생성

- choropleth map
  - coffeBean 매장의 수로 choropleth map 그리기



2. 지리정보 분석 후 맵 생성

- choropleth map
  - 전국 인구수로 choropleth 작성

```
# map 생성
map=folium.Map(location=[37.559819, 126.96389], zoom_start=7)
```

```
# 전국 시도 행정구역 geojson 파일로드(github에서 다운로두)
sido=open('data/ctprvn.json','r', encoding='utf-8')
sido_geo=json.load(sido)
sido_geo
```

```
# 전국 시도 경계선 map 작성
folium.GeoJson(sido_geo, name='시도').add_to(map)
map
```

2. 지리정보 분석 후 맵 생성

- choropleth map
  - 전국 인구수로 choropleth 작성

```
#국가통계 포털에서 인구수 정보 다운로드
df10=pd.read_excel('data/행정구역_시군구_별__성별_인구수.xlsx')
df10['gugun']=df10['gugun'].apply(lambda v:v.strip())
#df10.to_csv('data/df10.csv', index=False)
df11=df10.loc[df10['total']>0,:]
df11=df11.loc[df11['gugun']=='소계',:]
df11
```

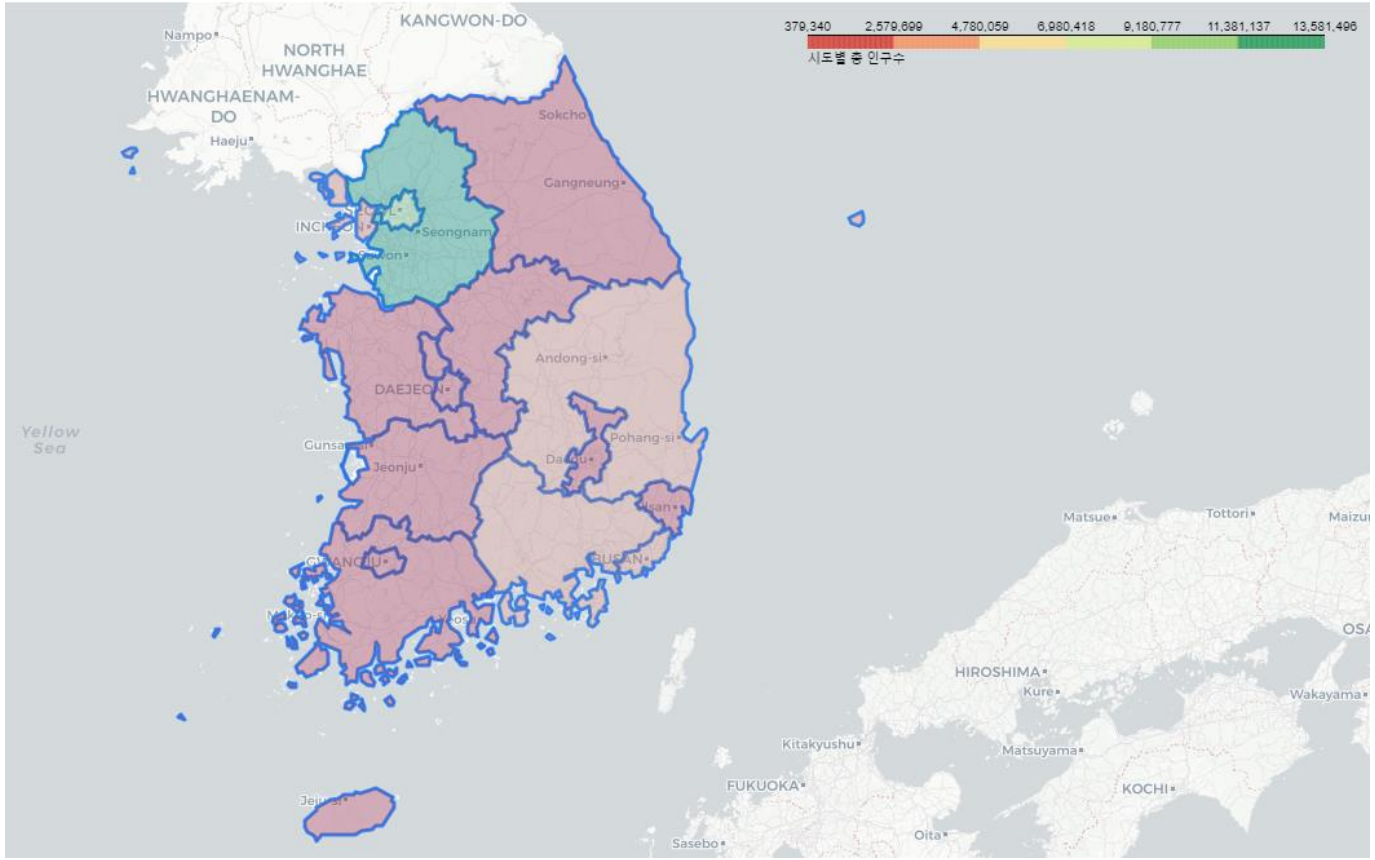
2. 지리정보 분석 후 맵 생성

- choropleth map
  - 전국 인구수로 choropleth 작성

```
# 시도별 인구수 choropleth 맵 생성
folium.Choropleth(geo_data=sido_geo,
                  data=df11,
                  columns=['sido','total'],
                  fill_color='RdYlGn',
                  fill_opacity=0.3,
                  line_opacity=0.1,
                  key_on='properties.CTP_KOR_NM',
                  legend_name='시도별 총 인구수'
                  ).add_to(map)
```

2. 지리정보 분석 후 맵 생성

- choropleth map
  - 전국 시구 geoJson 데이터로 choropleth 작성



2. 지리정보 분석 후 맵 생성

- choropleth map
  - 전국 시구 geoJson 데이터로 choropleth 작성

```
sig=open('data/sig.json','r',encoding='utf-8')
sig_geo=json.load(sig)
sig_geo
```

```
map=folium.Map(location=[37.559819, 126.96389], zoom_start=7)
folium.GeoJson(sig_geo,name='시군구').add_to(map)
map
```

03

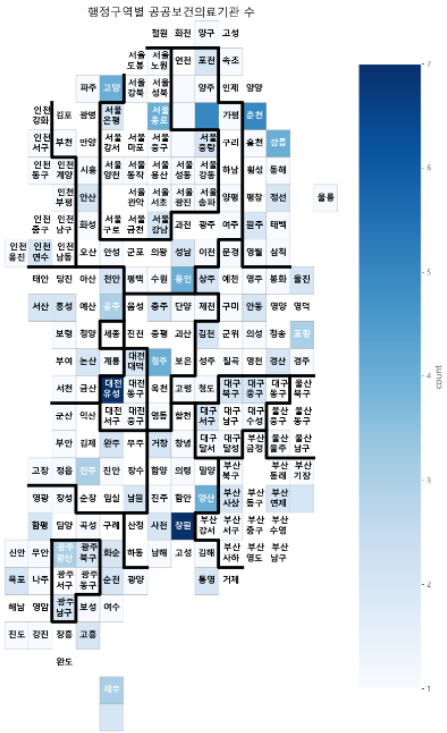
행정구역별 의료기간 현황 분석하기

3. 행정구역별 의료기간 현황분석하기

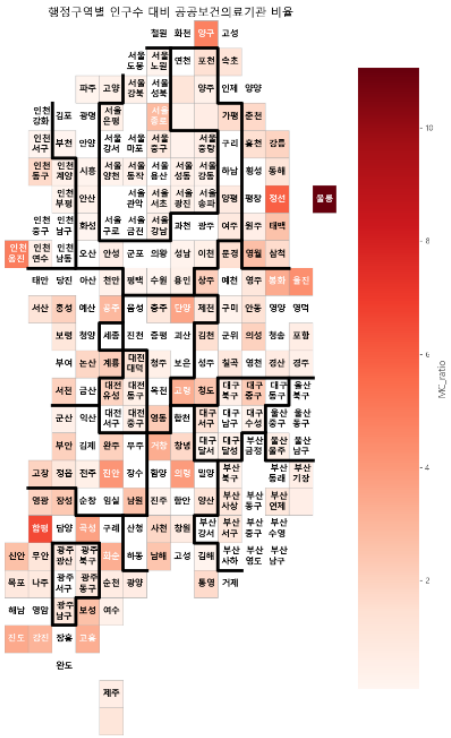
미리보기

행정구역별 의료기관 현황 분석하기	
목표	행정구역별로 공공보건의료기관 수를 파악하고 인구수 대비 공공보건의료기관 비율을 비교 분석한다. 분석 결과는 블록맵으로 시각화한다.
핵심 개념	블록맵
데이터 수집	▪ 공공보건의료기관현황.csv: 공공데이터포털에서 다운로드 ▪ 행정구역_시군구_별__성별_인구수_2.xlsx: 9장 01절의 프로젝트에서 정리한 파일
데이터 준비 및 탐색	▪ 행정구역 이름으로 주소 수정 ▪ 행정구역별 공공보건의료기관 수 집계 ▪ 행정구역별 인구수 데이터 정리 ▪ 테이블에 필요한 컬럼 추출 후 테이블 병합
분석 모델 구축 및 결과 시각화	

▪ 행정구역별 공공보건의료기관 수



▪ 행정구역별 인구수 대비 공공보건의료기관 비율



3. 행정구역별 의료기간 현황분석하기

목표설정

1. 행정구역별로 공공보건의료기관 수를 파악
2. 행정구역 별로 인구수 대비 공공보건의료기관 비율을 비교 분석

핵심 개념 이해

- 블록맵
  - 구역의 경계선을 단순화한 뒤 블록 형태로 그려서 지도를 나타내는 시각화 기법
  - 행정구역별 데이터 크기를 시각화할 때 많이 사용

데이터 수집

- 전국 공공보건의료기관 현황 데이터
  - 행정구역별 공공보건의료기관 수를 파악하고 인구수 대비 공공보건의료기관 비율을 비교 분석할 때 사용
  - 공공데이터포털 사이트에서 다운로드
- 행정구역별 인구수 데이터
  - 행정구역별 인구수 대비 공공보건의료기관 비율을 비교 분석할 때 사용하는 데이터

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

❸ 데이터 수집

- 전국 공공보건의료기관 현황 데이터 수집하기

1. 공공데이터포털 사이트(www.data.go.kr)에서 '**공공보건 의료기관 현황**'으로 검색

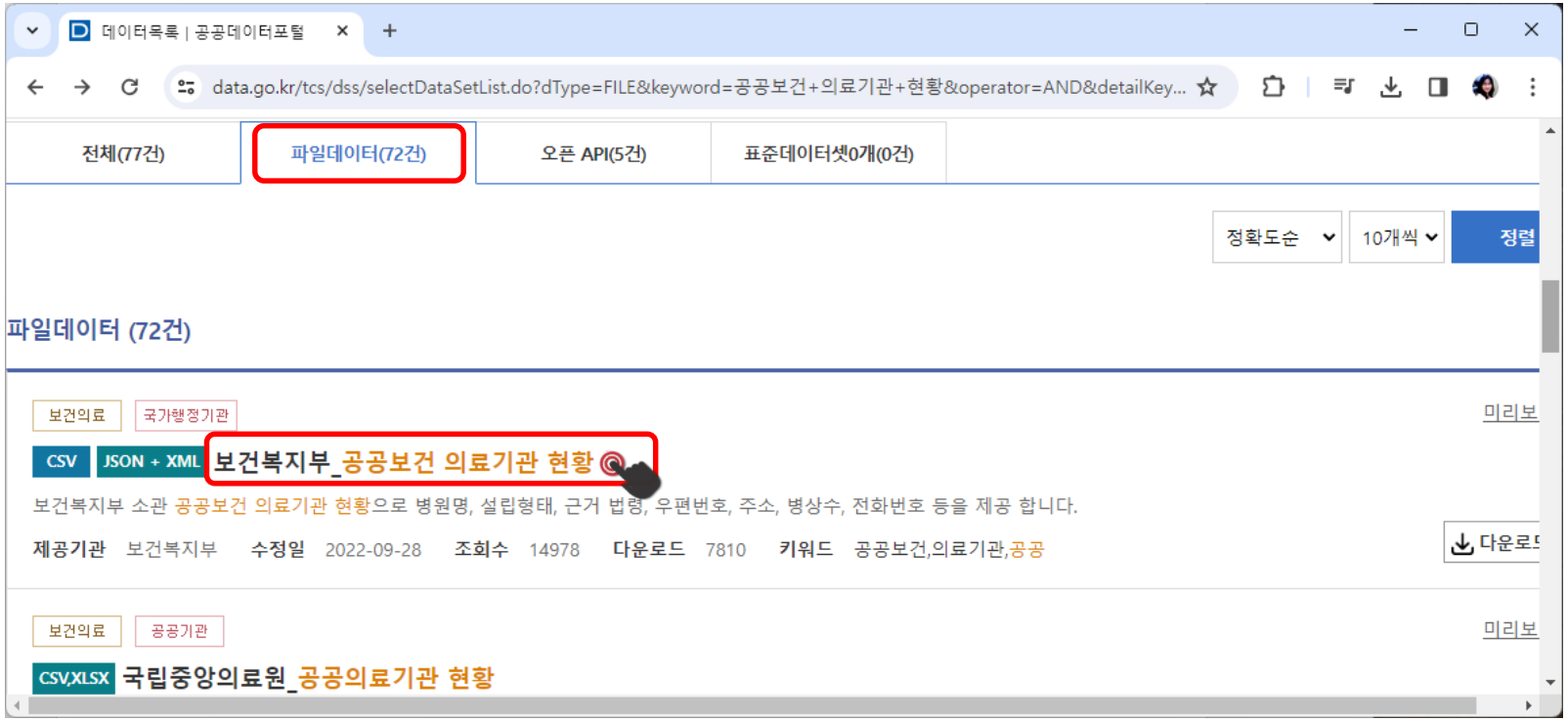


그림 9-27 '공공보건 의료기관 현황'으로 검색한 데이터 목록

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

❸ 데이터 수집

- 전국 공공보건의료기관 현황 데이터 수집하기

2. 파일데이터 상세 페이지가 나타나면 [다운로드] 버튼 클릭
3. 다운로드한 파일의 파일명을 '**공공보건의료기관현황.csv**'로 수정 한 후에, [9장_data] 폴더에 저장

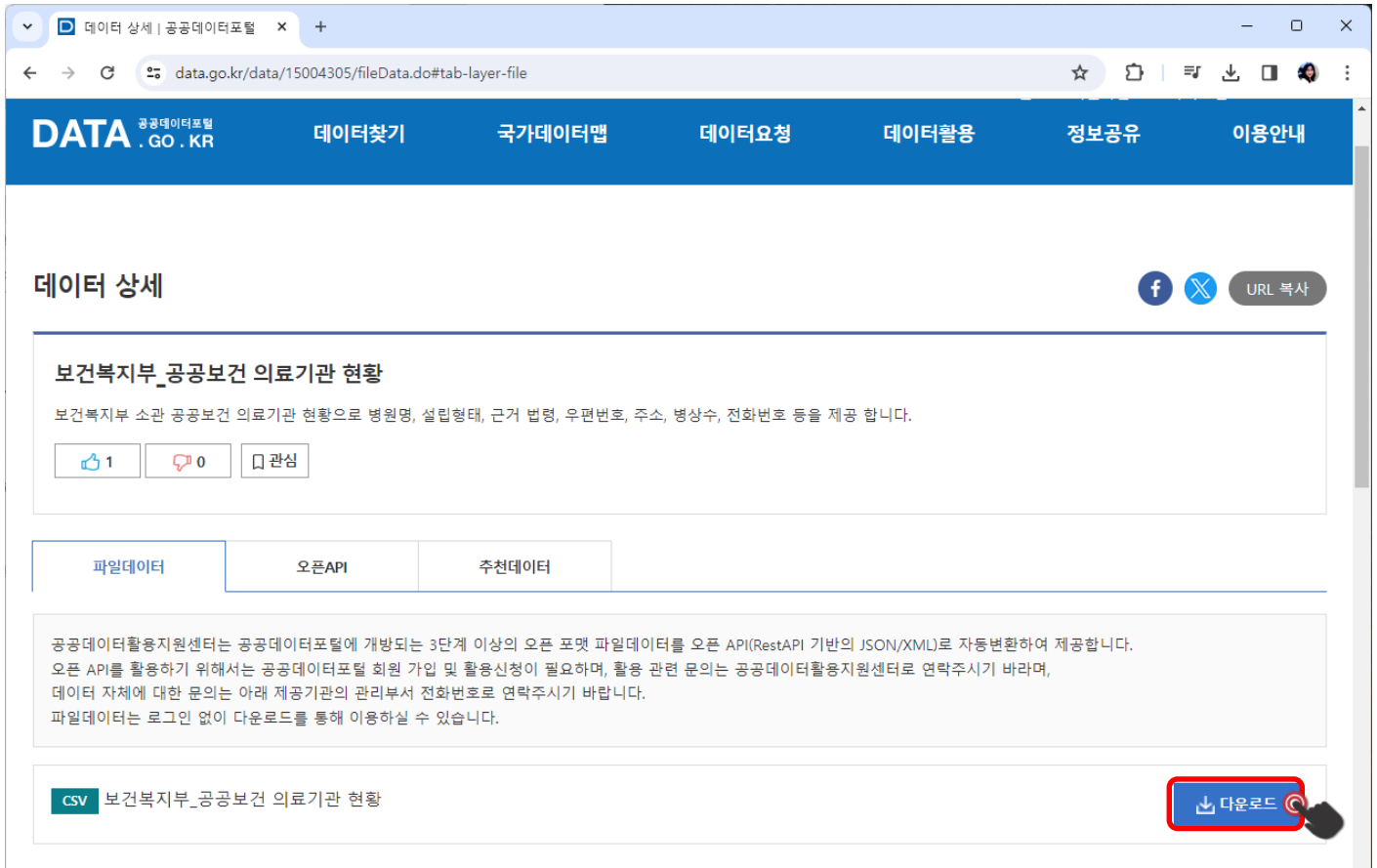


그림 9-28 공공보건 의료기관 현황 데이터 다운로드

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

▪ 공공보건의료기관 현황 데이터 준비하기

- Jupyter Notebook 실행. [My_Python] 폴더 안에 노트북 페이지 추가. '**9장_행정구역데이터분석**'으로 파일 이름 변경

1. 데이터 파일 확인하기

In [1]: '공공보건의료기관현황.csv' 파일을 data 객체로 로드하고, 상위 다섯 개 행의 데이터를 출력하여data.head( ) 확인

In [1]:

```
import pandas as pd
pd.set_option('mode.chained_assignment', None)
import numpy as np

data = pd.read_csv('9장_data/공공보건의료기관현황.csv', index_col = 0,
                   encoding = 'CP949', engine = 'python')
data.head()  # 작업 확인용 출력
```

Out[1]:

연번	병원 명	설립형태	근거 법령	관계 행정기관	관계 공공단체	심평원 요양기관번호	종별 구분	병상수	소재지 우편번호	주소	홈페이지	대표전화	FAX	비고
1	강원도 재활병원	시도립	강원도재활병원설치 및운영에관한조례	강원도	해당없음	32200641	병원	165	24227	강원도 춘천시 충열로 142번길 24-16	www.grh.or.kr	033-248-7700	033-248-7723	NaN
2	강원도 삼척의료원	특수법인	지방의료원의 설립 및 운영에 관한 법률	보건복지부(강원도)	지방의료원	32100060	종합병원	152	25920	강원도 삼척시 오십천로 418	http://ksmc.or.kr	033-572-1141	033-573-8424	NaN
3	강원도 영월의료원	특수법인	지방의료원의 설립 및 운영에 관한 법률	보건복지부(강원도)	지방의료원	32100078	종합병원	214	26234	강원도 영월군 영월읍 중앙1로 59	http://www.youngwol.org	033-370-9117	033-370-9137	NaN
4	강원도 원주의료원	특수법인	지방의료원의 설립 및 운영에 관한 법률	보건복지부(강원도)	지방의료원	32100086	종합병원	237	26448	강원도 원주시 서원대로 387(개운동)	www.kwmc.or.kr	033-760-4500	033-761-5121	NaN
5	강원도 강릉의료원	특수법인	지방의료원의 설립 및 운영에 관한 법률	보건복지부(강원도)	지방의료원	32100159	종합병원	137	25535	강원도 강릉시 경강로 2007(남문동 164-1)	http://www.gnmc.or.kr	033-646-6910	033-610-1415	NaN

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

2. 주소 정리하기

In [2]: data 객체에서 ['주소'] 컬럼의 값을 띄어쓰기를 기준으로 분리하여split( ), 시군과 군구 정보에 해당하는 0~1번 컬럼[:2]을 추출하여 컬럼 이름을 '시도', '군구'로 나타내고columns=(' 시도', '군구'), 데이터프레임 객체인 addr을 생성. 생성된 addr 객체의 내용을 출력하여addr.head( ) 확인

In [2]:
```
# 주소에서 시도, 군구 정보 분리
addr = pd.DataFrame(data['주소'].apply(lambda v: v.split()[:2]).
                    tolist(), columns = ('시도', '군구'))
addr.head()  # 작업 확인용 출력
```

Out[2]:

	시도	군구
0	강원도	춘천시
1	강원도	삼척시
2	강원도	영월군
3	강원도	원주시
4	강원도	강릉시

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

3. 시도 이름에서 잘못된 내용이 있는지 확인

In [3]: addr 객체의 ['시도'] 컬럼 값에서 고유값 확인addr['시도'].unique( ).

In [3]:	addr['시도'].unique()
Out[3]:	array(['강원도', '경기도', '경기', '경남', '창원시', '경상남도', '경상북도', '경산시', '경북', '인천광역시', '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도', '부산광역시', '전라북도', '충북', '서울특별시', '서울시', '부산특별시', '대전시', '충남', '전남', '충청남도', '울산광역시', '전북', '천안시', '충청북도'], dtype = object)

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

4. 잘못된 위치를 찾아서 값 수정(창원시)

In [4]: ['시도'] 컬럼 값이 '창원시'로 되어 있는 행 번호 찾기에서 27번과 31번을 확인함
In [5]: 27번과 31번의 값을 ['경상남도', '창원시']로 수정
In [6],[7]: 수정한 내용 확인

In [4]:
```
addr[addr['시도'] == '창원시']
```

Out[4]:

	시도	군구
27	창원시	의창구
31	창원시	마산합포구3.15대로

In [5]:
```
addr.iloc[27] = ['경상남도', '창원시']
addr.iloc[31] = ['경상남도', '창원시']
```

In [6]:
```
addr.iloc[27]
```

Out[6]:
```
시도    경상남도
군구     창원시
Name: 27, dtype: object
```

In [7]:
```
addr.iloc[31]
```

Out[7]:
```
시도    경상남도
군구     창원시
Name: 31, dtype: object
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

5. 잘못된 위치를 찾아서 값 수정(경산시, 천안시)

In [8]:
```
addr[addr['시도'] == '경산시']
```

Out[8]:

	시도	군구
47	경산시	경안로

In [9]:
```
addr.iloc[47] = ['경상북도', '경산시']
```

In [10]:
```
addr[addr['시도'] == '천안시']
```

Out[10]:

	시도	군구
209	천안시	동남구
210	천안시	동남구

In [11]:
```
addr.iloc[209] = ['충청남도', '천안시']
addr.iloc[210] = ['충청남도', '천안시']
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

6. 다시 addr 객체의 ['시도'] 컬럼 값에서 수정할 내용이 있는지 확인

In [12]:	addr['시도'].unique()
Out[12]:	array(['강원도', '경기도', '경기', '경남', '경상남도', '경상북도', '경북', '인천광역시', '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도', '부산광역시', '전라북도', '충북', '서울특별시', '서울시', '부산특별시', '대전시', '충남', '전남', '충청남도', '울산광역시', '전북', '충청북도'], dtype = object)

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

7. '경기', '경남'과 같이 축약된 이름을 정확한 표준 이름으로 수정

In [13]: 변경할 이름에 대한 '축약이름:표준이름' 형식으로 addr_aliases 딕셔너리 정의
In [14]: addr_aliases 딕셔너리를 적용하여 ['시도'] 컬럼의 값 변경
In [15]: addr 객체의 ['시도'] 컬럼 고유값을 출력하여 빠짐없이 변경되었는지 확인

In [13]:	addr_aliases = {'경기':'경기도', '경남':'경상남도', '경북':'경상북도', '충북':'충청북도', '서울시':'서울특별시', '부산특별시': '부산광역시', '대전시':'대전광역시', '충남':'충청남도', '전남':'전라남도', '전북':'전북특별자치도', '전라북도':'전 북특별자치도','강원도':'강원특별자치도'}
In [14]:	addr['시도'] = addr['시도'].apply(lambda v: addr_aliases.get(v, v))
In [15]:	addr['시도'].unique()
Out[15]:	array(['강원특별자치도', '경기도', '경상남도', '경상북도', '인천광역시', '대구광역시', '전라남도', '대전광역시', '광주광역시', '제주특별자치도', '부산광역시', '전북특별자치도', '충청북도', '서울특별시', '충청남도', '울산광역시'], dtype=object)

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

8. ['군구'] 컬럼에서 정리할 사항이 있는지 탐색

In [16]: addr 객체의 ['군구'] 컬럼의 고유값 확인

- '아란13길'을 인터넷에서 검색해보면 제주시에 있는 도로명이므로 '제주시'로 수정

In [16]:
```
addr['군구'].unique()
```

Out[16]:
```
array(['춘천시', '삼척시', '영월군', '원주시', '강릉시', '속초시', '정선군',
       '수원시', '이천시', '안성시', '의정부시', '포천시', '파주시',
       '용인시', '평택시', '시흥시', '여주시', '남양주시', '동두천시',
       '안산시', '부천시', '통영시', '사천시', '창원시', '김해시', '양산시',
       '거창군', '남해군', '의령군', '포항시', '김천시', '안동시', '울진군',
       '경주시', '구미시', '영주시', '상주시', '문경시', '경산시', '의성군',
       '청도군', '고령군', '칠곡군', '봉화군', '울릉군', '부평구', '북구',
       '순천시', '대덕구', '태백시', '동해시', '화성시', '광산구', '남구',
       '중구', '아란13길', '서구', '전주시', '진주시', '청주시', '종로구',
       '성남시', '동구', '화순군', '강동구', '사상구', '달서구', '해운대구',
       '유성구', '가평군', '양주시', '고양시', '홍천군', '양구군', '청원군',
       '계룡시', '논산시', '함평군', '양평군', '수성구', '달성군', '연수구',
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

8. ['군구'] 컬럼에서 정리할 사항이 있는지 탐색

In [17]: ['군구'] 컬럼 값이 '아란13길'로 되어 있는 행 번호를 찾으면 75번.

In [18]: 75번 행의 값을 ['제주특별자치도', '제주시']로 수정

In [17]:
```
addr[addr['군구'] == '아란13길']
```

Out[17]:

	시도	군구
75	제주특별자치도	아란13길

In [18]:
```
addr.iloc[75] = ['제주특별자치도', '제주시']
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

9. 행정구역별 공공보건의료기관의 수 구하기

In [19]: ['시도']와 ['군구'] 컬럼 값을 연결하여 만든 값을, addr 객체에 새로운 ['시도군구'] 컬럼으로 추가

In [19]:
```
addr['시도군구'] = addr.apply(lambda r: r['시도'] + ' ' + r['군구'], axis = 1)
addr.head()   # 작업 확인용 출력
```

Out[19]:

	시도	군구	시도군구
0	강원특별자치도	춘천시	강원특별자치도 춘천시
1	강원특별자치도	삼척시	강원특별자치도 삼척시
2	강원특별자치도	영월군	강원특별자치도 영월군
3	강원특별자치도	원주시	강원특별자치도 원주시
4	강원특별자치도	강릉시	강원특별자치도 강릉시

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

9. 행정구역별 공공보건의료기관의 수 구하기

In [20]: addr 객체에 ['count'] 컬럼을 추가

In [20]:
```
addr['count'] = 0
addr.head()   # 작업 확인용 출력
```

Out[20]:

	시도	군구	시도군구	count
0	강원특별자치도	춘천시	강원특별자치도 춘천시	0
1	강원특별자치도	삼척시	강원특별자치도 삼척시	0
2	강원특별자치도	영월군	강원특별자치도 영월군	0
3	강원특별자치도	원주시	강원특별자치도 원주시	0
4	강원특별자치도	강릉시	강원특별자치도 강릉시	0

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

9. 행정구역별 공공보건의료기관의 수 구하기

In [21]: ['시도'], ['군구'], ['시도군구'] 컬럼을 기준으로 그룹을 만듦addr.groupby(['시도', '군구', '시도군구'], as_index = False).
그룹별 원소의 개수를 구하여count( ) ['count'] 컬럼에 저장

In [21]:
```
addr_group = pd.DataFrame(addr.groupby(['시도', '군구', '시도군구'],
                                       as_index = False).count())
addr_group.head()   # 작업 확인용 출력
```

Out[21]:

	시도	군구	시도군구	count
0	강원특별자치도	강릉시	강원특별자치도 강릉시	4
1	강원특별자치도	동해시	강원특별자치도 동해시	1
2	강원특별자치도	삼척시	강원특별자치도 삼척시	1
3	강원특별자치도	속초시	강원특별자치도 속초시	1
4	강원특별자치도	양구군	강원특별자치도 양구군	1

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 공공보건의료기관 현황 데이터 준비하기

10. 데이터 병합에 사용할 인덱스를 설정

In [22]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

In [22]:
```
addr_group = addr_group.set_index("시도군구")
addr_group.head()    # 작업 확인용 출력
```

Out[22]:

시도군구	시도	군구	count
강원특별자치도 강릉시	강원특별자치도	강릉시	4
강원특별자치도 동해시	강원특별자치도	동해시	1
강원특별자치도 삼척시	강원특별자치도	삼척시	1
강원특별자치도 속초시	강원특별자치도	속초시	1
강원특별자치도 양구군	강원특별자치도	양구군	1

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 행정구역별 인구수 데이터 준비하기

1. 데이터 정리하기

In [23]: '행정구역_시군구_별__성별_인구수_2.xlsx' 파일을 population 객체로 로드하고, 출력하여 확인

In [24]: rename() 함수를 사용하여 컬럼 이름을 변경

In [23]:
```
population = pd.read_excel('9장_data/행정구역_시군구_별__성별_인구수_2.xlsx')
population.head()    # 작업 확인용 출력
```

Out[23]:

	행정구역(시군구)별(1)	행정구역(시군구)별(2)	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
0	전국	합계	51313912	25558944	25754968
1	서울특별시	소계	9384325	4538354	4845971
2	서울특별시	종로구	139378	67240	72138
3	서울특별시	중구	121322	58651	62671
4	서울특별시	용산구	212175	101793	110382

In [24]:
```
population = population.rename(columns = {'행정구역(시군구)별(1)': '시
                                          도', '행정구역(시군구)별(2)': '군구'})


population.head()  # 작업 확인용 출력
```

Out[24]:

	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
0	전국	합계	51313912	25558944	25754968
1	서울특별시	소계	9384325	4538354	4845971
2	서울특별시	종로구	139378	67240	72138
3	서울특별시	중구	121322	58651	62671
4	서울특별시	용산구	212175	101793	110382

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

▪ 행정구역별 인구수 데이터 준비하기

2. ['군구'] 컬럼에 포함되어 있는 왼쪽 띄어쓰기 공백 제거

In [25]: ['군구'] 컬럼의 문자열 앞뒤에 포함된 띄어쓰기 공백을 모두 제거strip( )

In [26]: ['시도']와 ['군구'] 컬럼 값을 연결하여 새로운 ['시도군구'] 컬럼에 추가

In [25]:
```
for element in range(0,len(population)):
    population['군구'][element] = population['군구'][element].strip()
```

In [26]:
```
population['시도군구'] = population.apply(lambda r: r['시도'] + ' ' +
                                          r['군구'], axis = 1)

population.head()  # 작업 확인용 출력
```

Out[26]:

	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	시도군구
0	전국	합계	51313912	25558944	25754968	전국 합계
1	서울특별시	소계	9384325	4538354	4845971	서울특별시 소계
2	서울특별시	종로구	139378	67240	72138	서울특별시 종로구
3	서울특별시	중구	121322	58651	62671	서울특별시 중구
4	서울특별시	용산구	212175	101793	110382	서울특별시 용산구

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 행정구역별 인구수 데이터 준비하기

2. ['시도군구'] 컬럼을 만들고 addr_group과 병합하기 위해 인덱스로 설정

In [27]: ['군구'] 컬럼 값이 '합계', '소계'인 행은 필요 없으므로 제외

In [28]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

In [27]:
```
population = population[population.군구 != '합계']
population = population[population.군구 != '소계']
```

In [28]:
```
population = population.set_index("시도군구")

population.head()  # 작업 확인용 출력
```

Out[28]:

시도군구	시도	군구	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
서울특별시 종로구	서울특별시	종로구	139378	67240	72138
서울특별시 중구	서울특별시	중구	121322	58651	62671
서울특별시 용산구	서울특별시	용산구	212175	101793	110382
서울특별시 성동구	서울특별시	성동구	277090	134390	142700
서울특별시 광진구	서울특별시	광진구	335335	161154	174181

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 행정구역별 인구수 데이터 준비하기

3. addr_group과 population을 인덱스 기준으로 병합: 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [29]: addr_group과 population을 내부병합how = 'inner' 으로 병합

In [29]:
```
addr_population_merge = pd.merge(addr_group,population,  how = 'inner',
                                 left_index = True, right_index = True)

addr_population_merge.head()   # 작업 확인용 출력
```

Out[29]:

시도군구	시도_x	군구_x	count	시도_y	군구_y	총인구수 (명)	남자인구수 (명)	여자인구수 (명)
강원특별자치도 강릉시	강원특별자치도	강릉시	4	강원특별자치도	강릉시	209174	103616	105558
강원특별자치도 동해시	강원특별자치도	동해시	1	강원특별자치도	동해시	88591	45007	43584
강원특별자치도 삼척시	강원특별자치도	삼척시	1	강원특별자치도	삼척시	62309	31645	30664
강원특별자치도 속초시	강원특별자치도	속초시	1	강원특별자치도	속초시	82021	40606	41415
강원특별자치도 양구군	강원특별자치도	양구군	1	강원특별자치도	양구군	20975	10994	9981

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 행정구역별 인구수 데이터 준비하기

3. addr_group과 population을 인덱스 기준으로 병합: 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [30]: 필요한 컬럼 4개만 추출하여 local_MC_Population 객체를 생성

In [30]:
```
local_MC_Population = addr_population_merge[['시도_x', '군구_x',
                                              'count', '총인구수 (명)']]


local_MC_Population.head()   # 작업 확인용 출력
```

Out[30]:

시도군구	시도_x	군구_x	count	총인구수 (명)
강원특별자치도 강릉시	강원특별자치도	강릉시	4	209174
강원특별자치도 동해시	강원특별자치도	동해시	1	88591
강원특별자치도 삼척시	강원특별자치도	삼척시	1	62309
강원특별자치도 속초시	강원특별자치도	속초시	1	82021
강원특별자치도 양구군	강원특별자치도	양구군	1	20975

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

4 데이터 준비 및 탐색

- 행정구역별 인구수 데이터 준비하기

3. addr_group과 population을 인덱스 기준으로 병합: 필요한 데이터를 하나의 데이터프레임으로 정리할 수 있음

In [31]: 인구수 대비 공공보건의료기관 비율을 구하여 local_MC_Population의 ['MC_ ratio'] 컬럼에 추가

In [31]:

```
# 컬럼 이름 변경
local_MC_Population = local_MC_Population.rename(columns = {'시도_x':
                      '시도', '군구_x': '군구','총인구수 (명)': '인구수'})

MC_count = local_MC_Population['count']
local_MC_Population['MC_ratio'] = MC_count.div(local_MC_Population['
                                  인구수'], axis = 0)*100000

local_MC_Population.head()   # 작업 확인용 출력
```

Out[31]:

시도군구	시도	군구	count	인구수	MC_ratio
강원특별자치도 강릉시	강원특별자치도	강릉시	4	209174	1.912284
강원특별자치도 동해시	강원특별자치도	동해시	1	88591	1.128783
강원특별자치도 삼척시	강원특별자치도	삼척시	1	62309	1.604905
강원특별자치도 속초시	강원특별자치도	속초시	1	82021	1.219200
강원특별자치도 양구군	강원특별자치도	양구군	1	20975	4.767580

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 바 차트 그리기

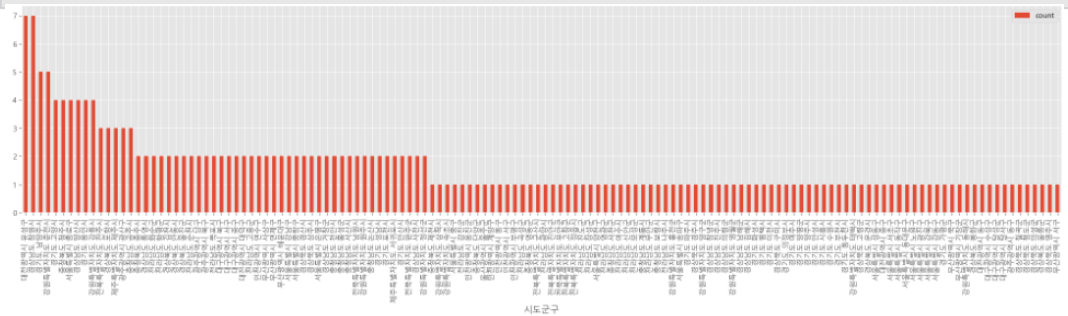
1. 행정구역별 공공보건의료기관 수에 대한 바 차트 그리기

In [33]: local_MC_Population 객체의 ['count'] 컬럼 값을 오름차순으로 정렬하여, 행정 구역별 공공보건의료기관 수에 대한 바 차트 그리기

In [32]:
```
from matplotlib import pyplot as plt
from matplotlib import rcParams, style
style.use('ggplot')


plt.rc('font', family = 'Malgun Gothic')
```

In [33]:
```
MC_ratio = local_MC_Population[['count']]
MC_ratio = MC_ratio.sort_values('count', ascending = False)
plt.rcParams["figure.figsize"] = (25, 5)
MC_ratio.plot(kind = 'bar', rot = 90)
plt.show()
```



02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

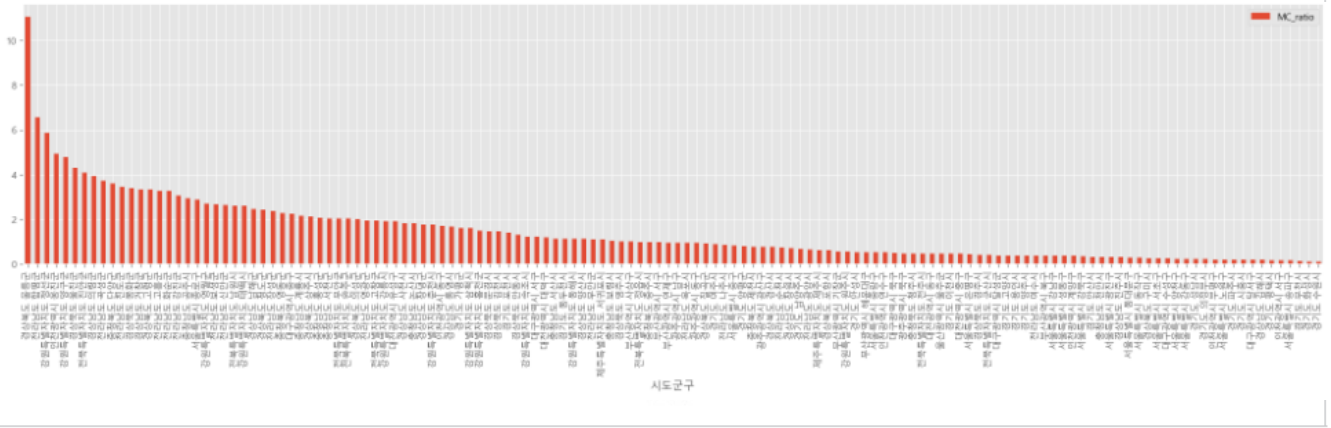
- 바 차트 그리기

2. 행정구역별로 인구수 대비 공공보건의료기관 비율에 대한 바 차트 그리기

In [34]: local_MC_Population 객체의 ['MC_ratio'] 컬럼 값을 오름차순으로 정렬하여, 행정구역별 인구수 대비 공공보건의료 기관 비율 바 차트 그리기

In [34]:
```
MC_ratio = local_MC_Population[['MC_ratio']]
MC_ratio = MC_ratio.sort_values('MC_ratio', ascending = False)
plt.rcParams["figure.figsize"] = (25, 5)
MC_ratio.plot(kind = 'bar', rot = 90)
plt.show()
```

Out[34]:



02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

1. 데이터 준비하기

In [35]: 현재 사용 중인 폴더(디렉토리)의 경로를 구하여 path에 저장

In [36]: **data_draw_korea.csv** 파일을 로드하여 data_draw_korea 객체에 저장하고, 출력하여 내용을 확인 (data_draw_korea.csv 파일은 제공하는 파일 사용. 또는 인터넷에서 검색하여 다운로드하여 사용)

In [35]:
```
import os
path = os.getcwd()
```

In [36]:
```
data_draw_korea = pd.read_csv(path+'\\9장_data\\data_draw_korea.csv',
                              index_col = 0, encoding = 'UTF-8', engine = 'python')

data_draw_korea.head()   # 작업 확인용 출력
```

Out[36]:

	인구수	shortName	x	y	면적	광역시도	행정구역
0	202520	강릉	11	4	1040.07	강원도	강릉시
1	25589	고성(강원)	9	0	664.19	강원도	고성군
2	86747	동해	11	5	180.01	강원도	동해시
3	63986	삼척	11	8	1185.80	강원도	삼척시
4	76733	속초	9	1	105.25	강원도	속초시

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

2. 행정구역 이름 매핑하기

In [37]: 2023년 6월에 변경된 '강원특별자치도'와 '전북특별자치도' 행정구역명 수정하기

In [38]: ['광역시도']와 ['행정구역'] 컬럼 값을 연결하여 새로운 ['시도군구'] 컬럼으로 추가

In [37]:

```
#2023년 6월 이후에 변경된 행정구역명으로 수정하기
addr_aliases = { '강원도':'강원특별자치도', '전라북도':'전북특별자치도'}
data_draw_korea['광역시도'] = data_draw_korea['광역시도'].
apply(lambda v: addr_aliases.get(v, v))
data_draw_korea.head() # 작업 확인용 출력
```

Out[37]:

	인구수	shortName	x	y	면적	광역시도	행정구역
0	202520	강릉	11	4	1040.07	강원특별자치도	강릉시
1	25589	고성(강원)	9	0	664.19	강원특별자치도	고성군
2	86747	동해	11	5	180.01	강원특별자치도	동해시
3	63986	삼척	11	8	1185.80	강원특별자치도	삼척시
4	76733	속초	9	1	105.25	강원특별자치도	속초시

In [38]:

```
data_draw_korea['시도군구'] = data_draw_korea.apply(lambda r: r['광역
                                시도'] + ' ' + r['행정구역'], axis = 1)
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

2. 행정구역 이름 매핑하기

In [39]: ['시도군구'] 컬럼을 데이터프레임 병합에 사용할 인덱스로 설정

In [39]:
```
data_draw_korea = data_draw_korea.set_index("시도군구")

data_draw_korea.head()  # 작업 확인용 출력
```

Out[39]:

시도군구	인구수	shortName	x	y	면적	광역시도	행정구역
강원특별자치도 강릉시	202520	강릉	11	4	1040.07	강원특별자치도	강릉시
강원특별자치도 고성군	25589	고성(강원)	9	0	664.19	강원특별자치도	고성군
강원특별자치도 동해시	86747	동해	11	5	180.01	강원특별자치도	동해시
강원특별자치도 삼척시	63986	삼척	11	8	1185.80	강원특별자치도	삼척시
강원특별자치도 속초시	76733	속초	9	1	105.25	강원특별자치도	속초시

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

2. 행정구역 이름 매핑하기

In [40]: data_draw_korea와 local_MC_Population을 외부병합how = 'outer'으로 병합한다

In [40]:
```
data_draw_korea_MC_Population_all = pd.merge(data_draw_korea,local_MC_
                                    Population, how = 'outer', left_
                                    index = True, right_index = True)

data_draw_korea_MC_Population_all.head()  # 작업 확인용 출력
```

Out[40]:

시도군구	인구수_x	shortName	x	y	면적	광역시도	행정구역	시도	군구	count	인구수_y	MC_ratio
강원특별자치도 강릉시	202520	강릉	11	4	1040.07	강원특별자치도	강릉시	강원특별자치도	강릉시	4.0	209174.0	1.912284
강원특별자치도 고성군	25589	고성(강원)	9	0	664.19	강원특별자치도	고성군	NaN	NaN	NaN	NaN	NaN
강원특별자치도 동해시	86747	동해	11	5	180.01	강원특별자치도	동해시	강원특별자치도	동해시	1.0	88591.0	1.128783
강원특별자치도 삼척시	63985	삼척	11	8	1185.80	강원특별자치도	삼척시	강원특별자치도	삼척시	1.0	62309.0	1.604905
강원특별자치도 속초시	76733	속초	9	1	105.25	강원특별자치도	속초시	강원특별자치도	속초시	1.0	82021.0	1.219200

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

3. 블록맵으로 시각화하기

In [41]: 블록맵의 행정구역 경계선을 그리기 위해 행정구역의 블록 위치 x, y 데이터 정의

In [41]:
```
BORDER_LINES = [
    [(3, 2), (5, 2), (5, 3), (9, 3), (9, 1)], # 인천
    [(2, 5), (3, 5), (3, 4), (8, 4), (8, 7), (7, 7), (7, 9), (4, 9),
     (4, 7), (1, 7)], # 서울
    [(1, 6), (1, 9), (3, 9), (3, 10), (8, 10), (8, 9),
     (9, 9), (9, 8), (10, 8), (10, 5), (9, 5), (9, 3)], # 경기도
    [(9, 12), (9, 10), (8, 10)], # 강원도
    [(10, 5), (11, 5), (11, 4), (12, 4), (12, 5), (13, 5),
    (13, 4), (14, 4), (14, 2)], # 충청남도
    [(11, 5), (12, 5), (12, 6), (15, 6), (15, 7), (13, 7),
     (13, 8), (11, 8), (11, 9), (10, 9), (10, 8)], # 충청북도
    [(14, 4), (15, 4), (15, 6)], # 대전시
    [(14, 7), (14, 9), (13, 9), (13, 11), (13, 13)], # 경상북도
    [(14, 8), (16, 8), (16, 10), (15, 10),
     (15, 11), (14, 11), (14, 12), (13, 12)], # 대구시
    [(15, 11), (16, 11), (16, 13)], # 울산시
    [(17, 1), (17, 3), (18, 3), (18, 6), (15, 6)], # 전라북도
    [(19, 2), (19, 4), (21, 4), (21, 3), (22, 3), (22, 2), (19, 2)], # 광주시
    [(18, 5), (20, 5), (20, 6)], # 전라남도
    [(16, 9), (18, 9), (18, 8), (19, 8), (19, 9), (20, 9), (20, 10)], # 부산시
]
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

3. 블록맵으로 시각화하기

In [42]: 블록맵의 블록에 데이터를 매핑하고 색을 표시하여 블록맵을 그린 뒤 저장하는 함수 정의

In [42]:

```
def draw_blockMap(blockedMap, targetData, title, color ):
    whitelabelmin = (max(blockedMap[targetData]) -
                     min(blockedMap[targetData])) * 0.25 +
                     min(blockedMap[targetData])

    datalabel = targetData

    vmin = min(blockedMap[targetData])
    vmax = max(blockedMap[targetData])

    mapdata = blockedMap.pivot(index = 'y', columns = 'x', values = targetData)
    masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

    plt.figure(figsize = (8, 13))
    plt.title(title)
    plt.pcolor(masked_mapdata, vmin = vmin, vmax = vmax, cmap =
               color, edgecolor = '#aaaaaa', linewidth = 0.5)

    # 지역 이름 표시
    for idx, row in blockedMap.iterrows():
      annocolor = 'white' if row[targetData] > whitelabelmin else
                  'black'
      # 광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시
      if row['광역시도'].endswith('시') and not row['광역시도'].
      startswith('세종'):
          dispname = '{}\n{}'.format(row['광역시도'][:2], row['행정구역'][:-1])
          if len(row['행정구역']) <= 2:
              dispname += row['행정구역'][-1]
      else:
          dispname = row['행정구역'][:-1]
      # 서대문구, 서귀포시같이 이름이 3자 이상이면 작은 글자로 표시
      if len(dispname.splitlines()[-1]) >= 3:
          fontsize, linespacing = 9.5, 1.5
      else:
          fontsize, linespacing = 11, 1.2

      plt.annotate(dispname, (row['x']+0.5, row['y']+0.5), weight = 'bold',
          fontsize = fontsize, ha = 'center', va = 'center', color = annocolor,
          linespacing = linespacing)

# 시도 경계 그리기
for path in BORDER_LINES:
    ys, xs = zip(*path)
    plt.plot(xs, ys, c = 'black', lw = 4)

plt.gca().invert_yaxis()
#plt.gca().set_aspect(1)
plt.axis('off')

cb = plt.colorbar(shrink = 1, aspect = 10)
cb.set_label(datalabel)

plt.tight_layout()
plt.savefig('.\\9장_data\\' + 'blockMap_' + targetData + '.png')
plt.show()
```

02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

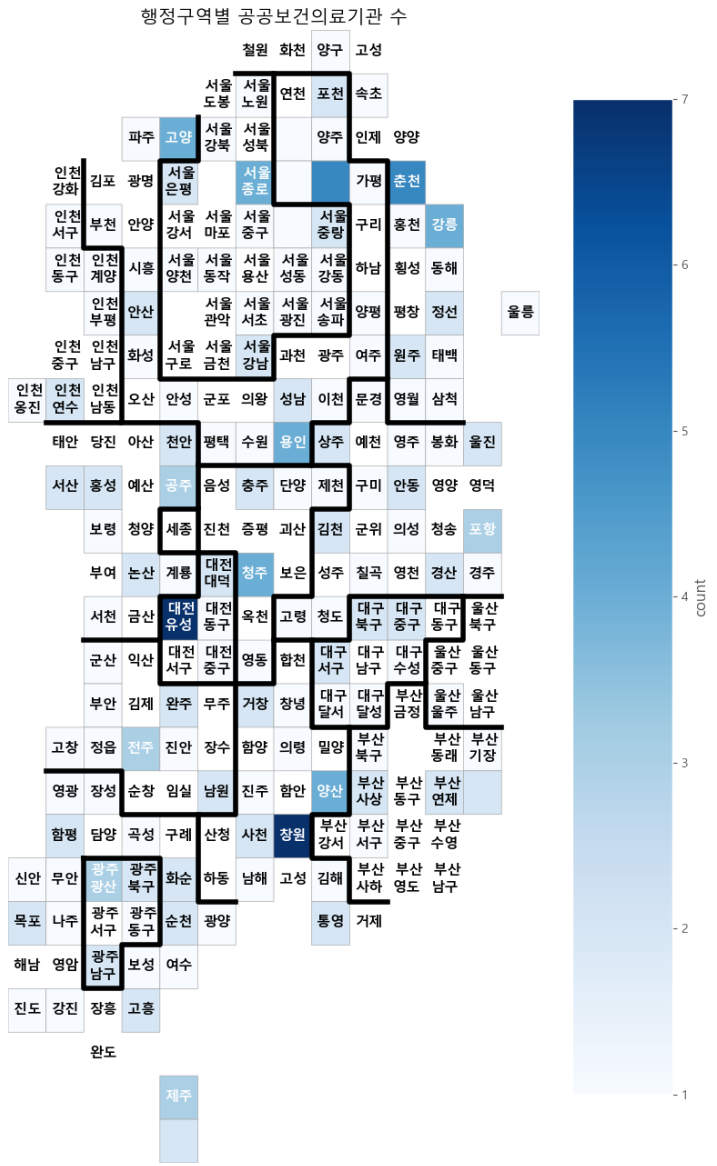
5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

4. 행정구역별 공공보건의료기관 수를 블록맵으로 시각화

In [43]: data_draw_korea_MC_Population_all 객체의 ['count'] 컬럼 값에 대해 Blues 색상 스펙트럼을 사용하여 블록맵 작성

```
In [43]: draw_blockMap(data_draw_korea_MC_Population_all, 'count',
                       '행정구역별 공공보건의료기관 수', 'Blues')
```



02. [행정구역별 데이터 분석+블록맵] 행정구역별 의료기관 현황 분석하기

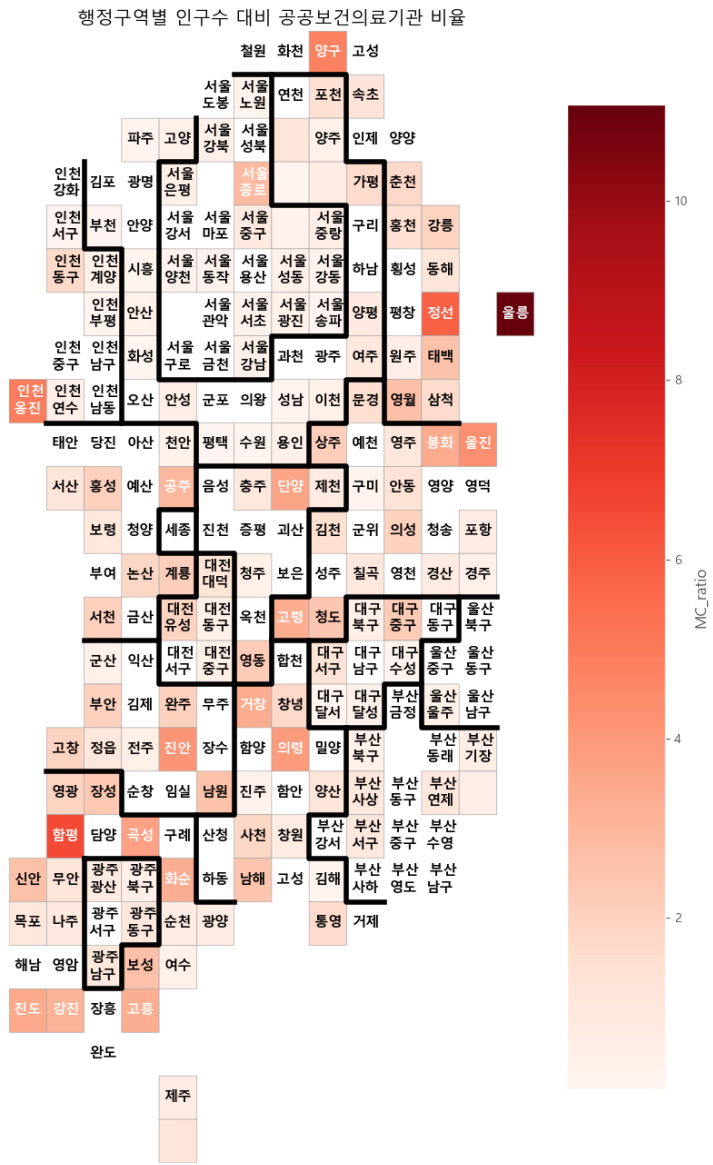
5 분석 모델 구축 및 시각화

- 블록맵으로 시각화하기

5. 인구수 대비 공공보건의료기관 비율을 블록맵으로 시각화

In [44]: data_draw_korea_MC_Population_all 객체의 ['MC_ratio'] 컬럼 값에 대해 Reds 색상 스펙트럼을 사용하여 블록맵 작성

```
In [44]: draw_blockMap(data_draw_korea_MC_Population_all, 'MC_ratio',
                       '행정구역별 인구수 대비 공공보건의료기관 비율', 'Reds')
```



04

실습

happy_mege.csv 사용 choropleth map

- 전국 시도별 행복지수를 사용하여 choropleth map을 작성하여 제출

Q&A