

빅데이터 처리

-Big Data Processing



12주차

■ 프로젝트

- 주제 변경 될 경우 새로운 ppt 첨부 후 메일 (기존 ppt 삭제 x)

■ 계획

- 9주차 : 머신 러닝 분석
- 10주차 : 데이터 시각화
- 11주차 : 데이터 시각화
- 12주차 : **지리 정보 분석**
- 13주차 : 텍스트 분석
- 14주차 : 시계열 데이터 분석
- 15주차

■ Folium

- Marker
- CircleMarker
- Choropleth



Folium

■ Geopandas

- plot

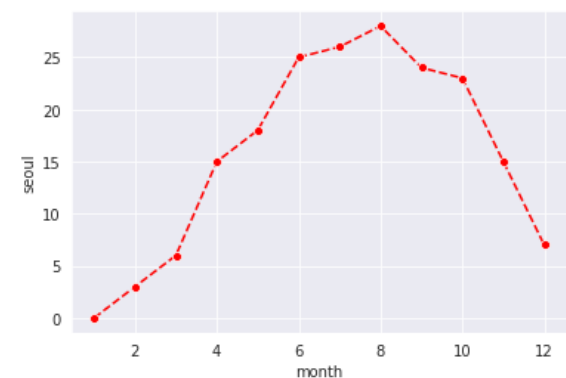


GeoPandas

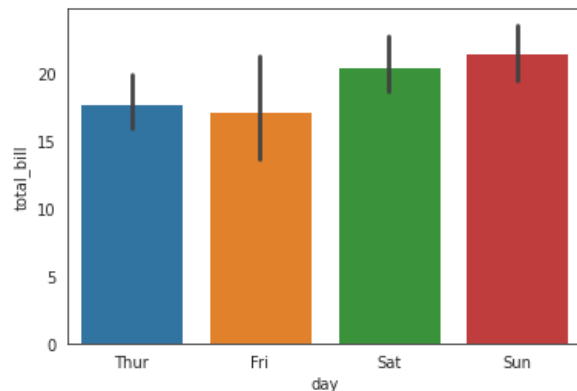
seaborn

인하공전 컴퓨터 정보 과

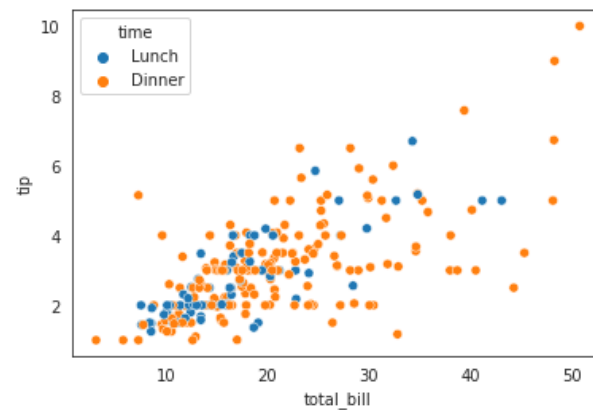
(`data=tips,x='total_bill',y='tip', hue='time'`)



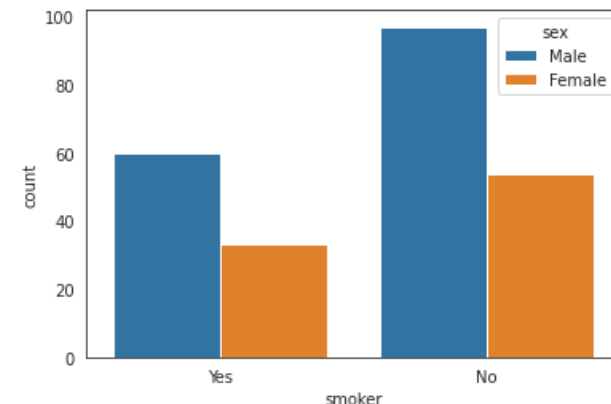
lineplot



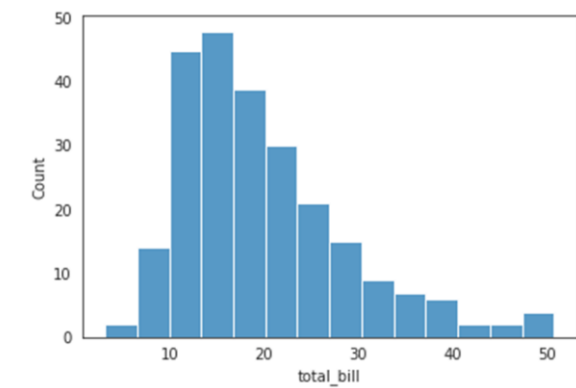
barplot



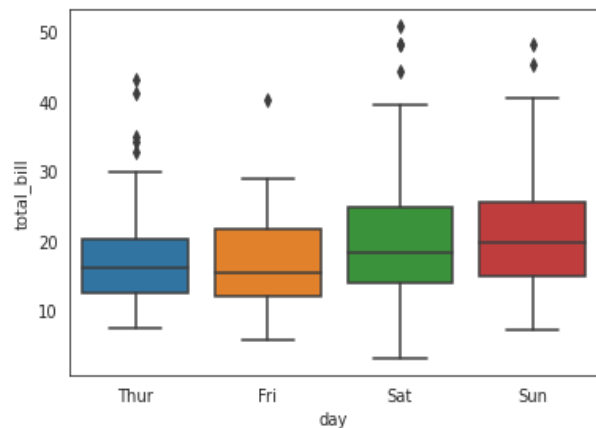
scatterplot



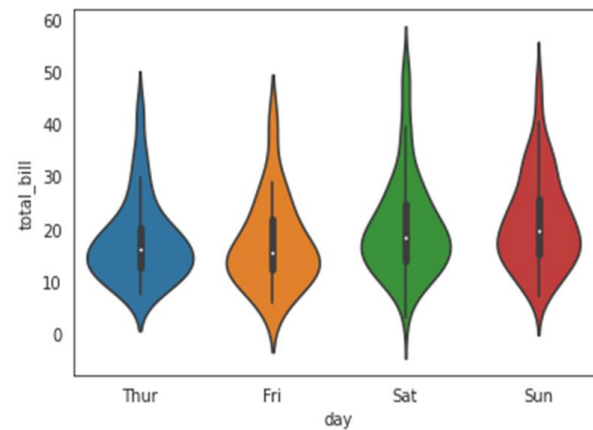
countplot



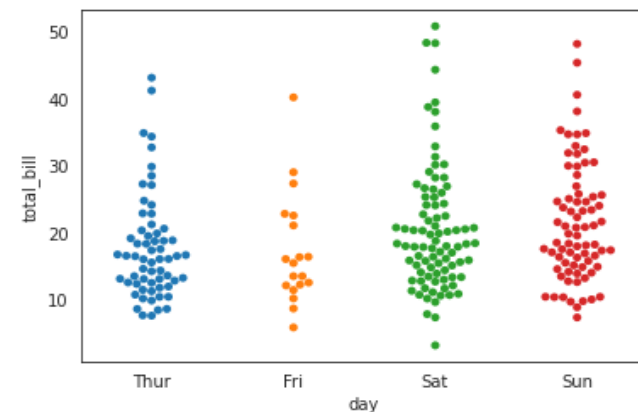
histplot



boxplot



violin plot



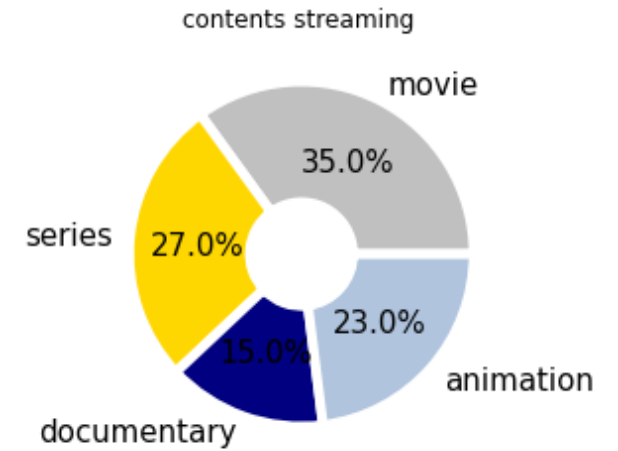
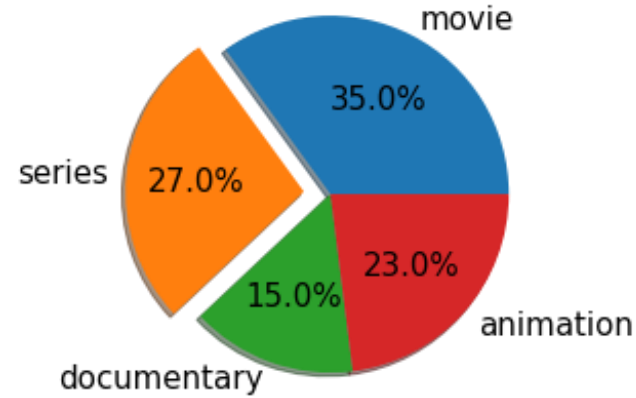
swarm plot

Matplotlib, pandas

인하공전 컴퓨터 정보 과

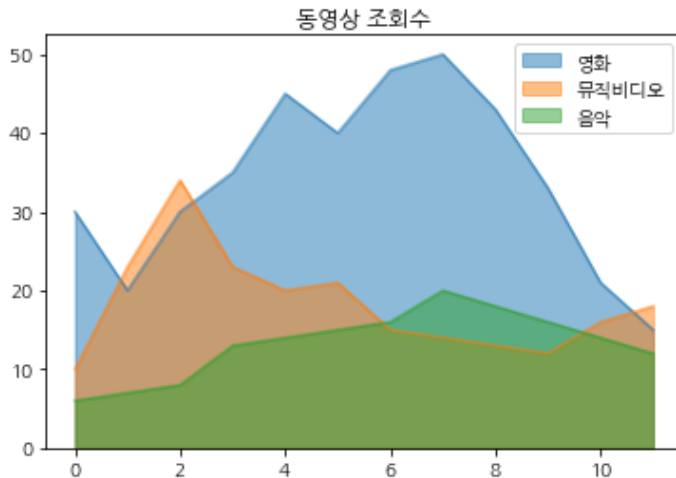
Matplotlib

- pie
(explode, color, wedgeprops)

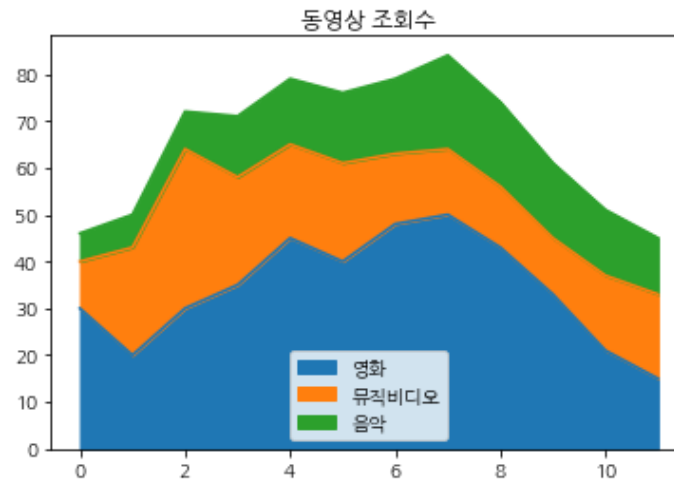


Pandas

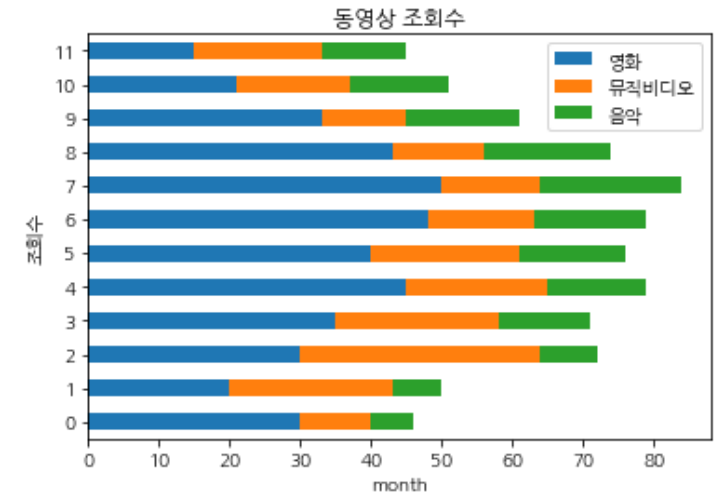
- plot(kind='area',stacked=False)



- plot(kind='area',stacked=True)

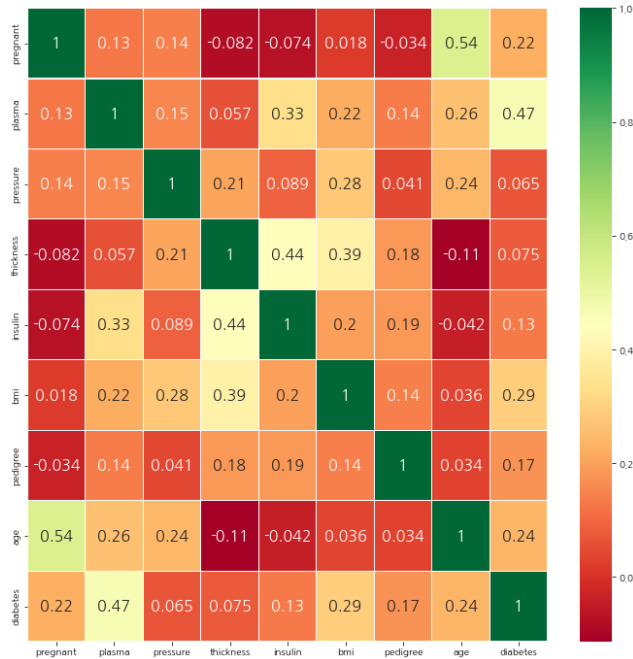


- plot(kind='barh',stacked=True)

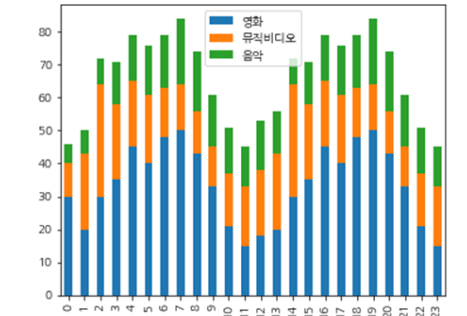
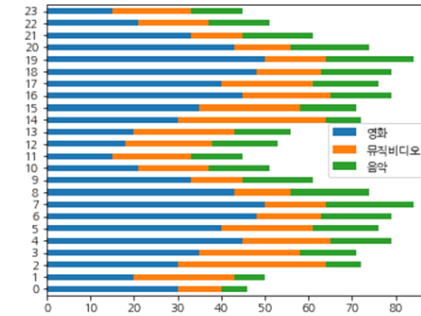
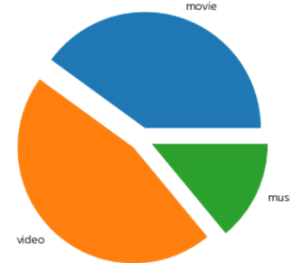
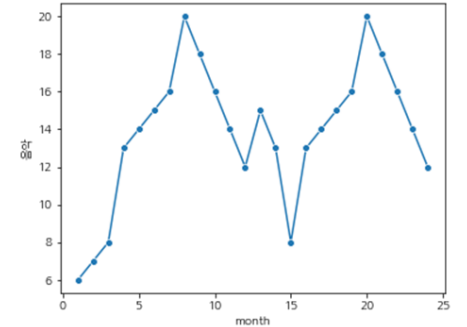
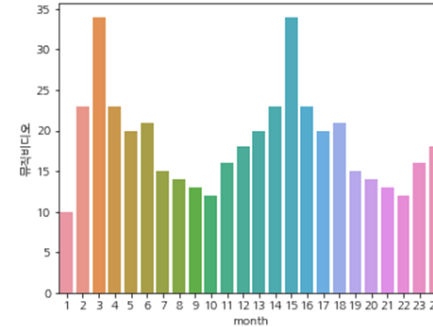
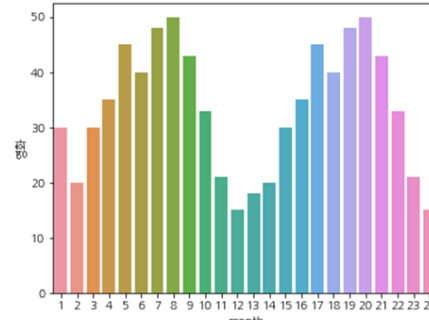


Seaborn, matplotlib

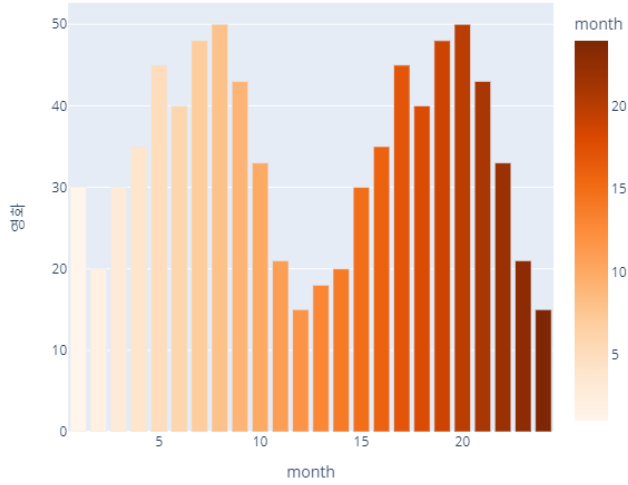
인하공전 컴퓨터 정보 과



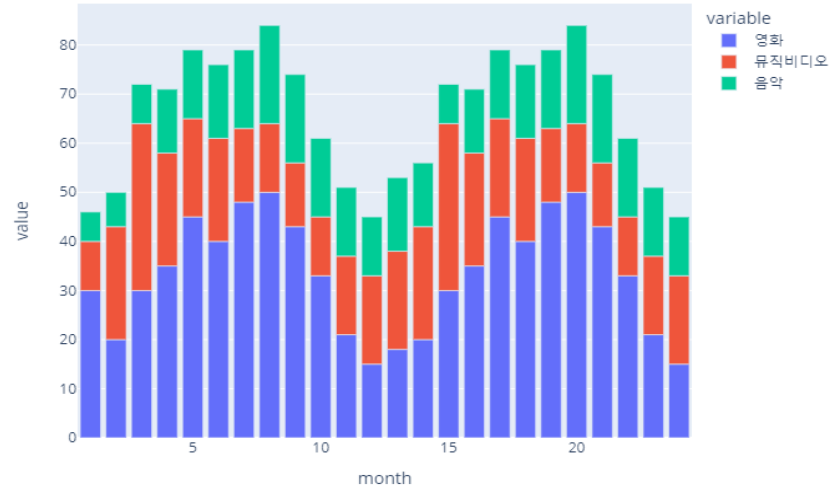
sns.heatmap



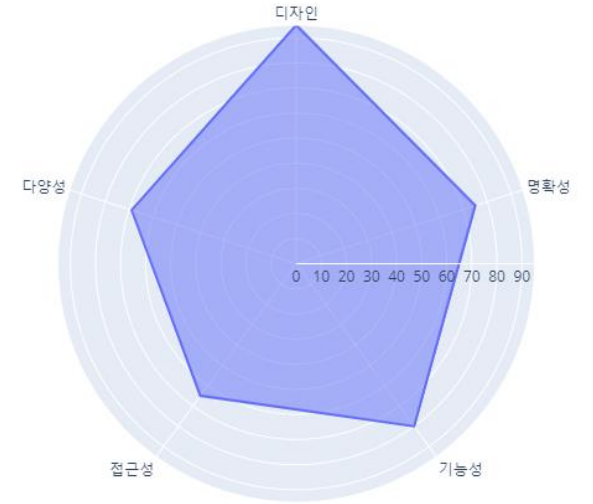
Subplot(row,column, index)



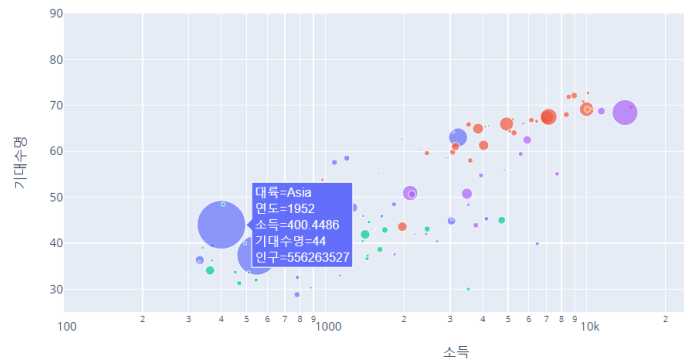
bar



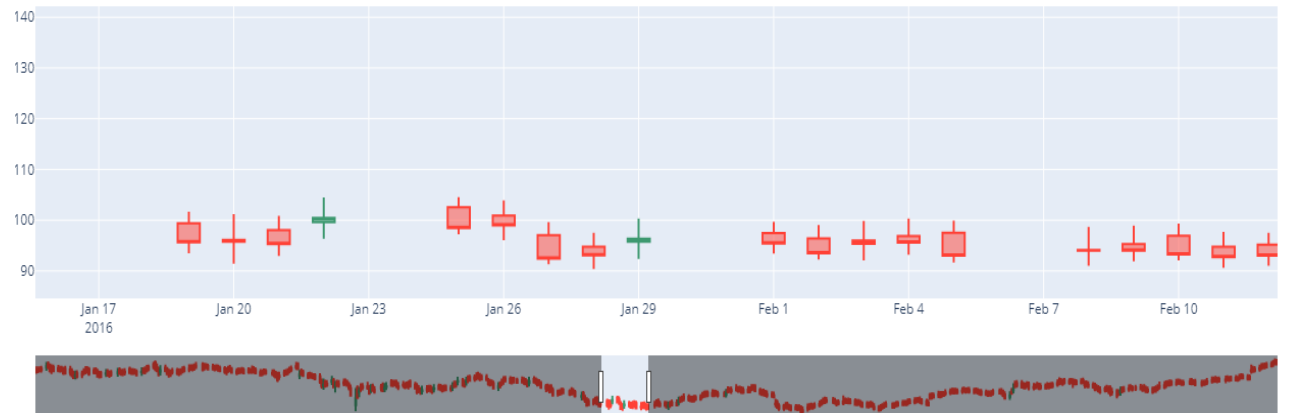
bar



line_polar



scatter



candlestick

PLOTLY- animation

인하공전 컴퓨터 정보 과

```
df = px.data.gapminder()
df.rename(columns={'gdpPercap':'소득','lifeExp':'기대수명','year':'연도','continent':'대륙','pop':'인구'},inplace=True)
print(df.head())
```

```
fig = px.scatter(df, x="소득", y="기대수명", animation_frame="연도", animation_group="country",
                size="인구", color="대륙",
                log_x=True, size_max=55, range_x=[100,100000], range_y=[25,90])
```

```
fig["layout"].pop("updatemenus") # optional, drop animation buttons
fig.update_layout(width=900,height=650)
fig.show()
```


Gapminder 데이터 셋

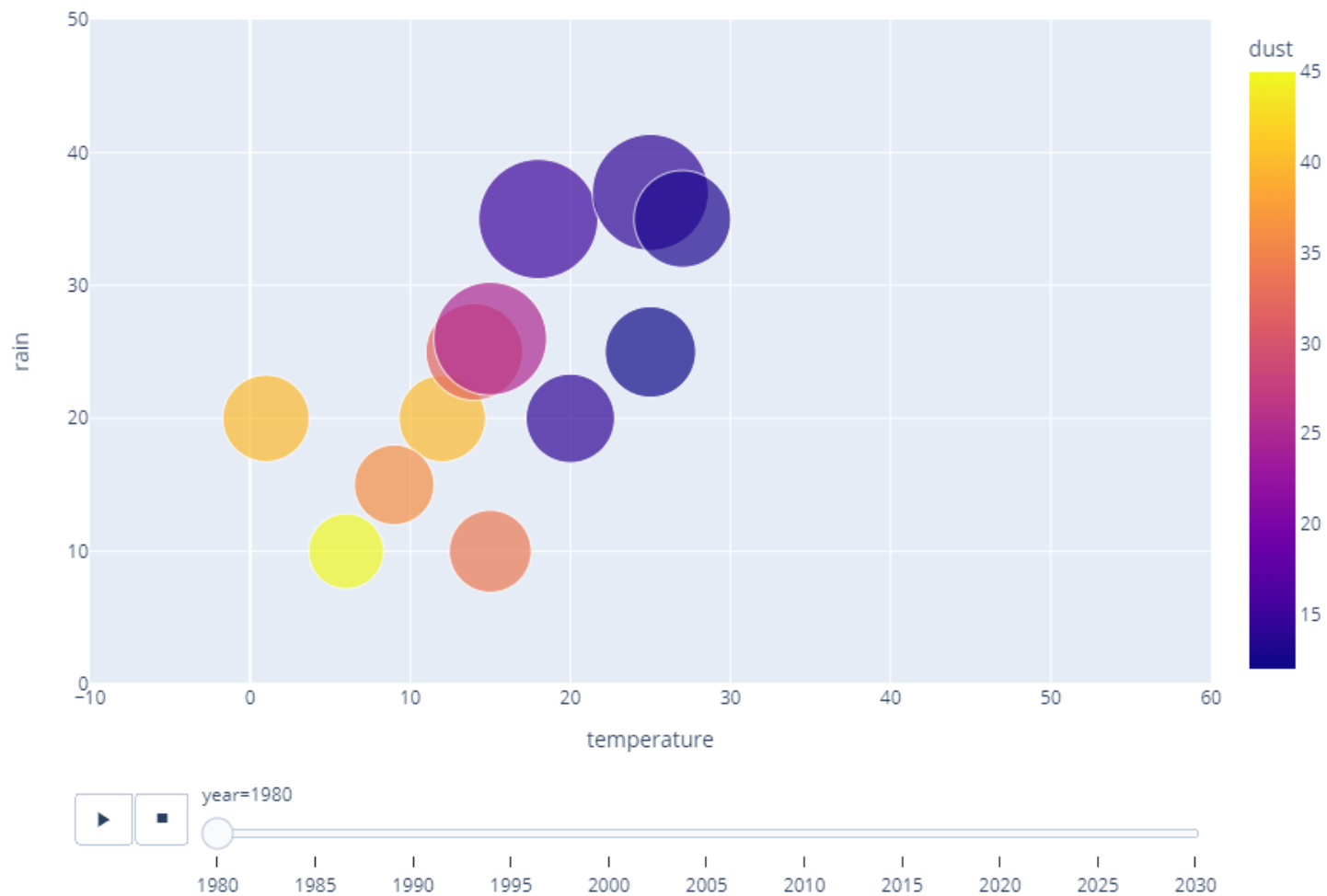
인하공전 컴퓨터 정보 과

- Plotly에 포함된 데이터 셋 (`df = px.data.gapminder()`)
- 연도별, 국가 별로 기대수명, 인구, 소득 데이터를 포함

	country	대륙	연도	기대수명	인구	소득	iso_alpha	₩
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	
5	Afghanistan	Asia	1977	38.438	14880372	786.113360	AFG	
6	Afghanistan	Asia	1982	39.854	12881816	978.011439	AFG	
7	Afghanistan	Asia	1987	40.822	13867957	852.395945	AFG	
8	Afghanistan	Asia	1992	41.674	16317921	649.341395	AFG	
9	Afghanistan	Asia	1997	41.763	22227415	635.341351	AFG	
10	Afghanistan	Asia	2002	42.129	25268405	726.734055	AFG	
11	Afghanistan	Asia	2007	43.828	31889923	974.580338	AFG	
12	Albania	Europe	1952	55.230	1282697	1601.056136	ALB	
13	Albania	Europe	1957	59.280	1476505	1942.284244	ALB	

PLOTLY- animation

인하공전 컴퓨터 정보 과





지도 위의 시각화를 지원 하는 라이브러리

Import folium

Map() 함수 지도 객체 생성

```
import folium
```

```
seoul_map=folium.Map(location=[37.5658,126.975146], zoom_start=16)
```

덕수궁 경도, 위도

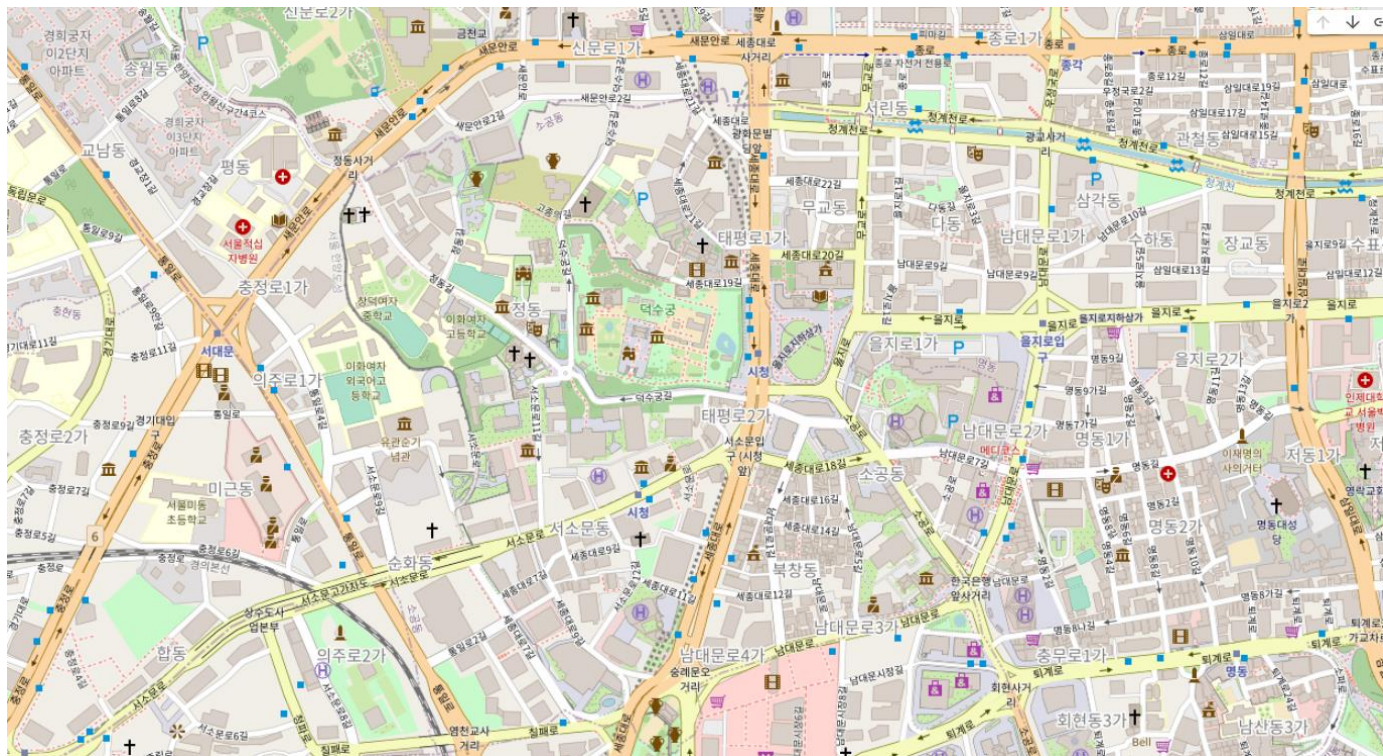
```
seoul_map
```

```
#seoul_map.save('./seoul.html')
```

location: 중심 위치의 경도, 위도

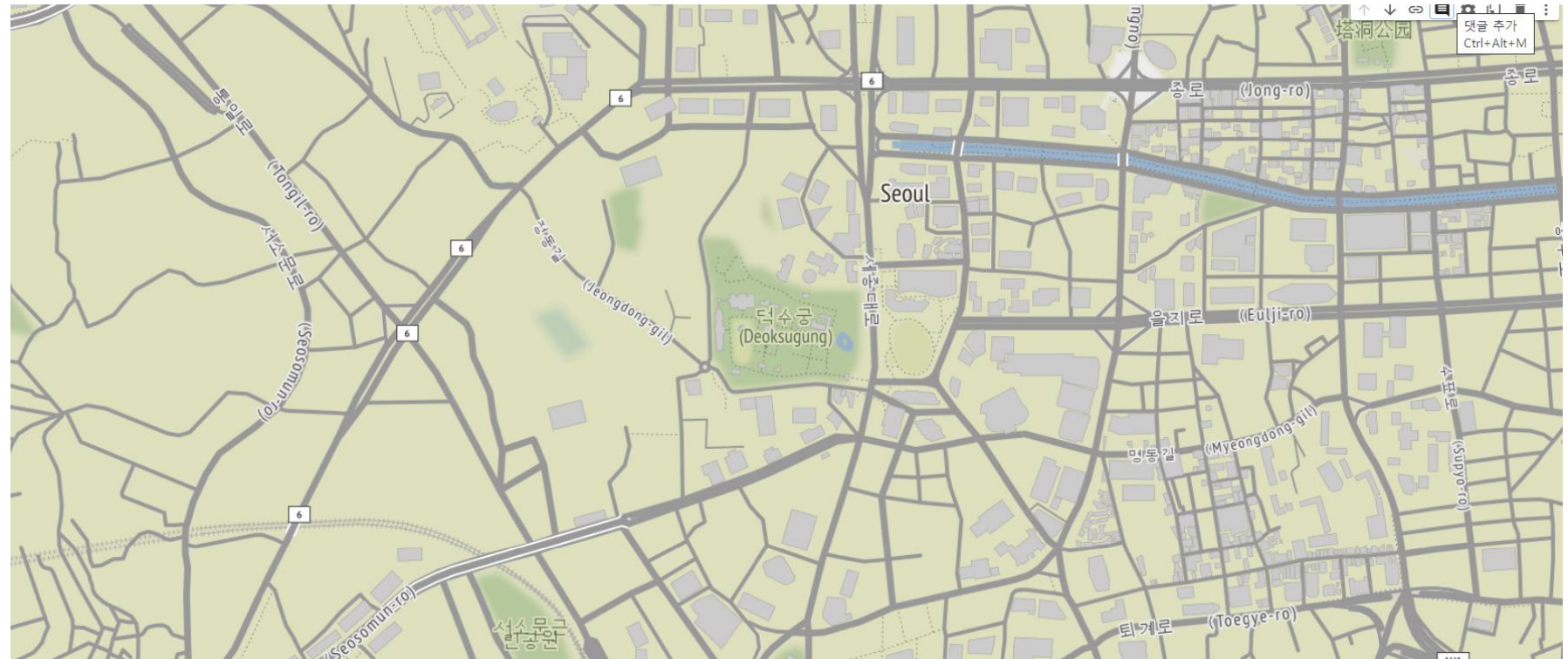
zoom_start: 확대 비율

참고 : 파이썬 머신러닝 판다스 데이터 분석





```
seoul_map=folium.Map(location=[37.5658,126.975146],tiles='Stamen Terrain',zoom_start=16)
```



참고 : 파이썬 머신러닝 판다스 데이터 분석





지도위에 마커 표시 : Marker() 함수에 위도, 경도 정보 전달

```
df = pd.read_csv('/content/인천광역시 연수구_주유소.csv', encoding = 'CP949')
```

```
print(df)
```

```
# 서울 지도 만들기
```

```
smap = folium.Map(location=[37.4101597 , 126.6783087 ], tiles='Stamen Terrain', zoom_start=14)
# 연수구청 중심
```

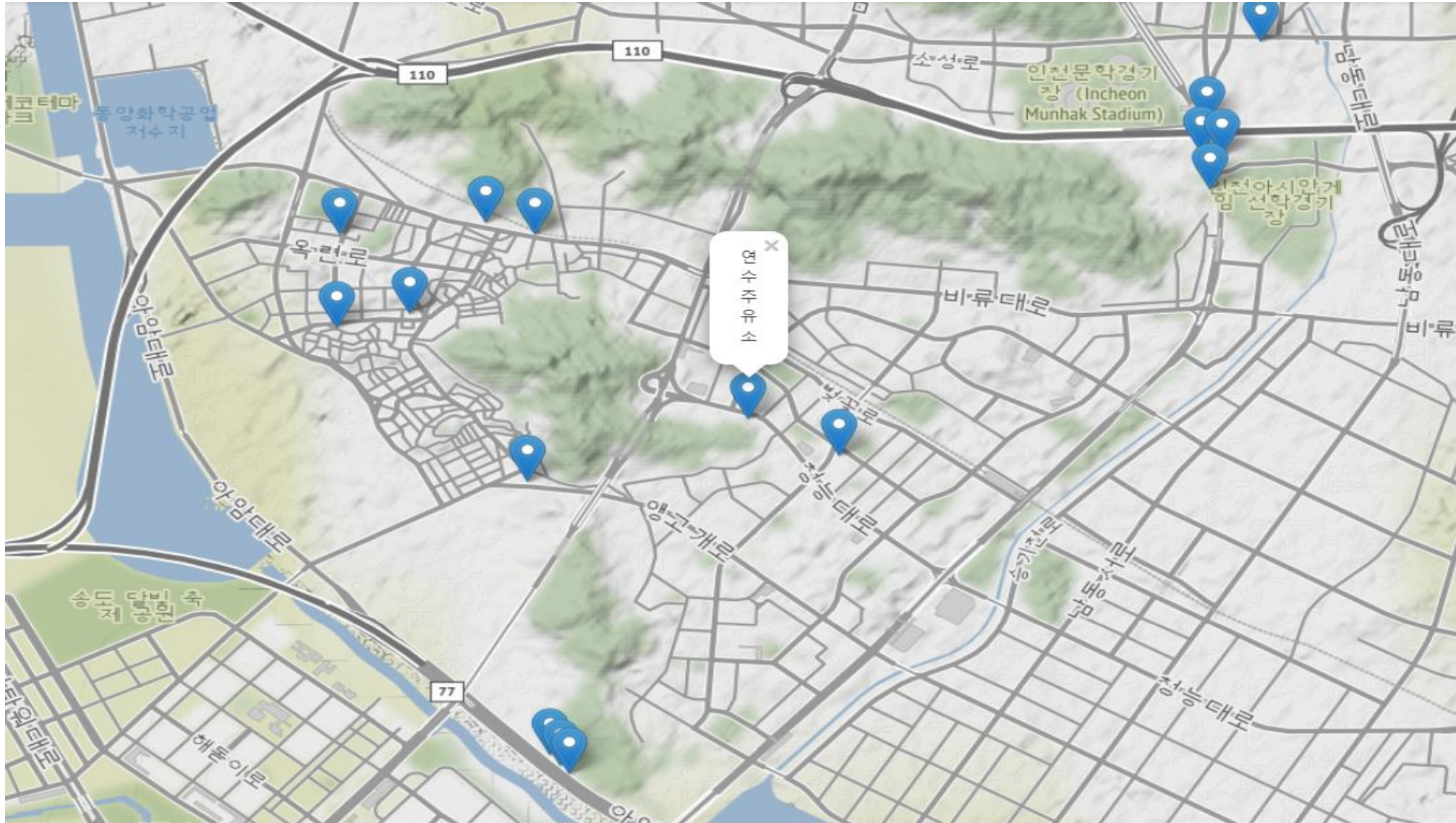
```
# 주유소 위치정보를 Marker로 표시
```

```
for name, lat, lng in zip(df.업소명, df.위도, df.경도):
```

```
    folium.Marker([lat, lng], popup=name).add_to(smap)
```

```
smap
```

	A	B	C	D	E	F
1	업소명	소재지	위도	경도	전화번호	
2	송도주유소	인천광역시	37.42751	126.6484	032-833-5189	
3	송화주유소	인천광역시	37.41323	126.6591	032-832-5156	
4	연수주유소	인천광역시	37.41684	126.6718	032-819-1151	
5	송도신도시주유소	인천광역시	37.39692	126.661	032-833-0405	
6	명품셀프주유소	인천광역시	37.43859	126.7014	032-434-5189	
7	선학제일주유소	인천광역시	37.43392	126.6983	032-433-5188	
8	지에스칼텍스(주)선학로	인천광역시	37.43012	126.6985	032-815-5189	
9	현대오일뱅크(주)직영	인천광역시	37.4282	126.6567	032-832-6111	
10	원천제3주유소	인천광역시	37.42753	126.6596	032-832-5181	
11	지에스칼텍스(주)연수	인천광역시	37.41478	126.6771	032-813-5184	
12	크로바주유소	인천광역시	37.42213	126.6482	032-834-8500	
13	현대오일뱅크(주)직영	인천광역시	37.39638	126.6616	032-833-7020	





```

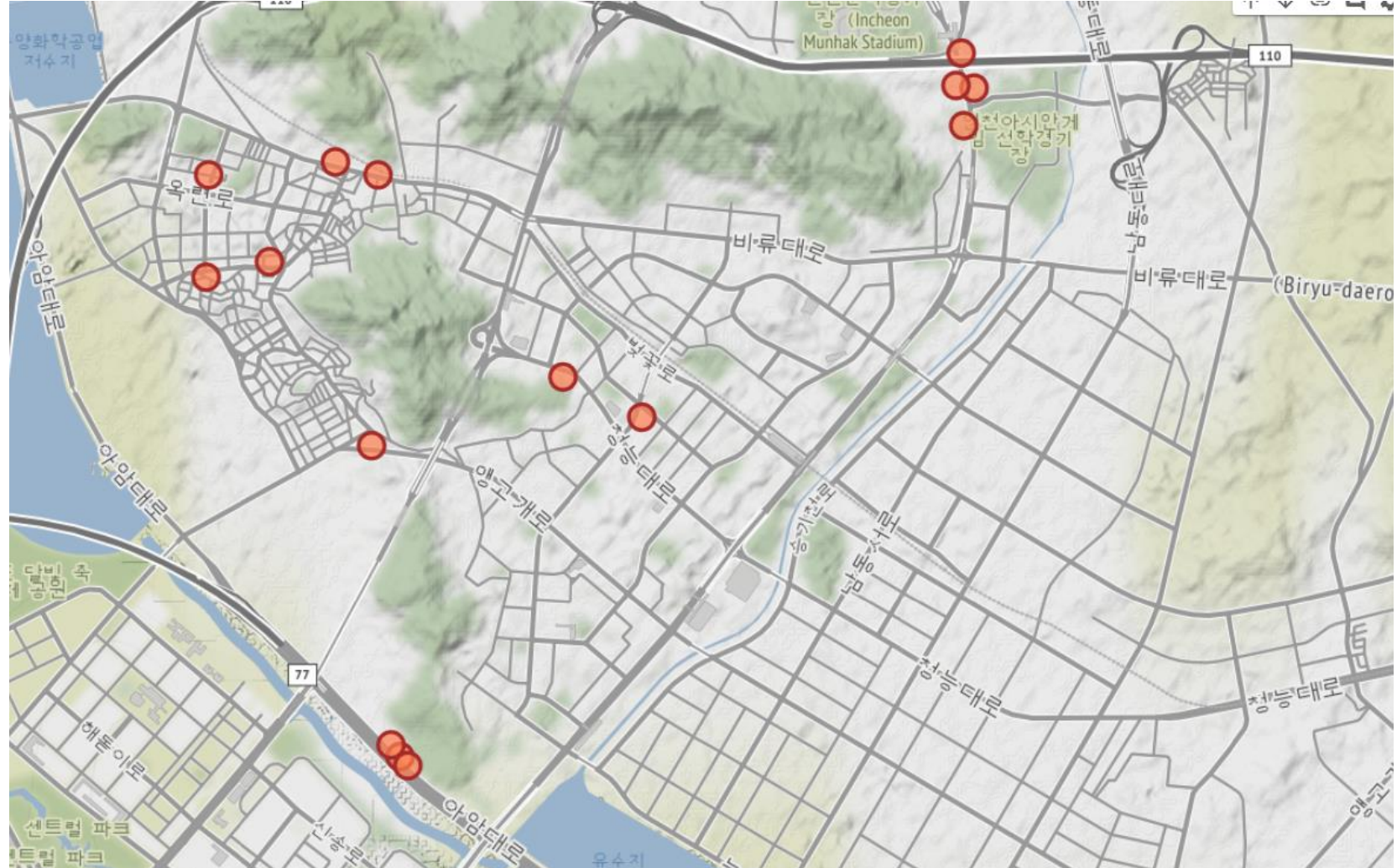
df = pd.read_csv('/content/인천광역시 연수구_주유소.csv', encoding = 'CP949')

print(df)
# 지도 만들기
smap = folium.Map(location=[37.4101597 , 126.6783087 ], tiles='Stamen Terrain', zoom_start=14)
# 연수구청 37.4101597 126.6783087

for name, lat, lng in zip(df.업소명, df.위도, df.경도):
    folium.CircleMarker([lat, lng],
                        radius=10, # 원의 반지름
                        color='brown', # 원의 둘레 색상
                        fill=True,
                        fill_color='coral', # 원을 채우는 색
                        fill_opacity=0.7, # 투명도
                        popup=name
    ).add_to(smap)

```

	A	B	C	D	E	F
1	업소명	소재지	위도	경도	전화번호	
2	송도주유소	인천광역시	37.42751	126.6484	032-833-5189	
3	송화주유소	인천광역시	37.41323	126.6591	032-832-5156	
4	연수주유소	인천광역시	37.41684	126.6718	032-819-1151	
5	송도신도시주유소	인천광역시	37.39692	126.661	032-833-0405	
6	명품셀프주유소	인천광역시	37.43859	126.7014	032-434-5189	
7	선학제일주유소	인천광역시	37.43392	126.6983	032-433-5188	
8	지에스칼텍스(주)선학동	인천광역시	37.43012	126.6985	032-815-5189	
9	현대오일뱅크(주)직영	인천광역시	37.4282	126.6567	032-832-6111	
10	원천제3주유소	인천광역시	37.42753	126.6596	032-832-5181	
11	지에스칼텍스(주)연수	인천광역시	37.41478	126.6771	032-813-5184	
12	크로바주유소	인천광역시	37.42213	126.6482	032-834-8500	
13	현대오일뱅크(주)직영	인천광역시	37.39638	126.6616	032-833-7020	





Choropleth 단계 구분도

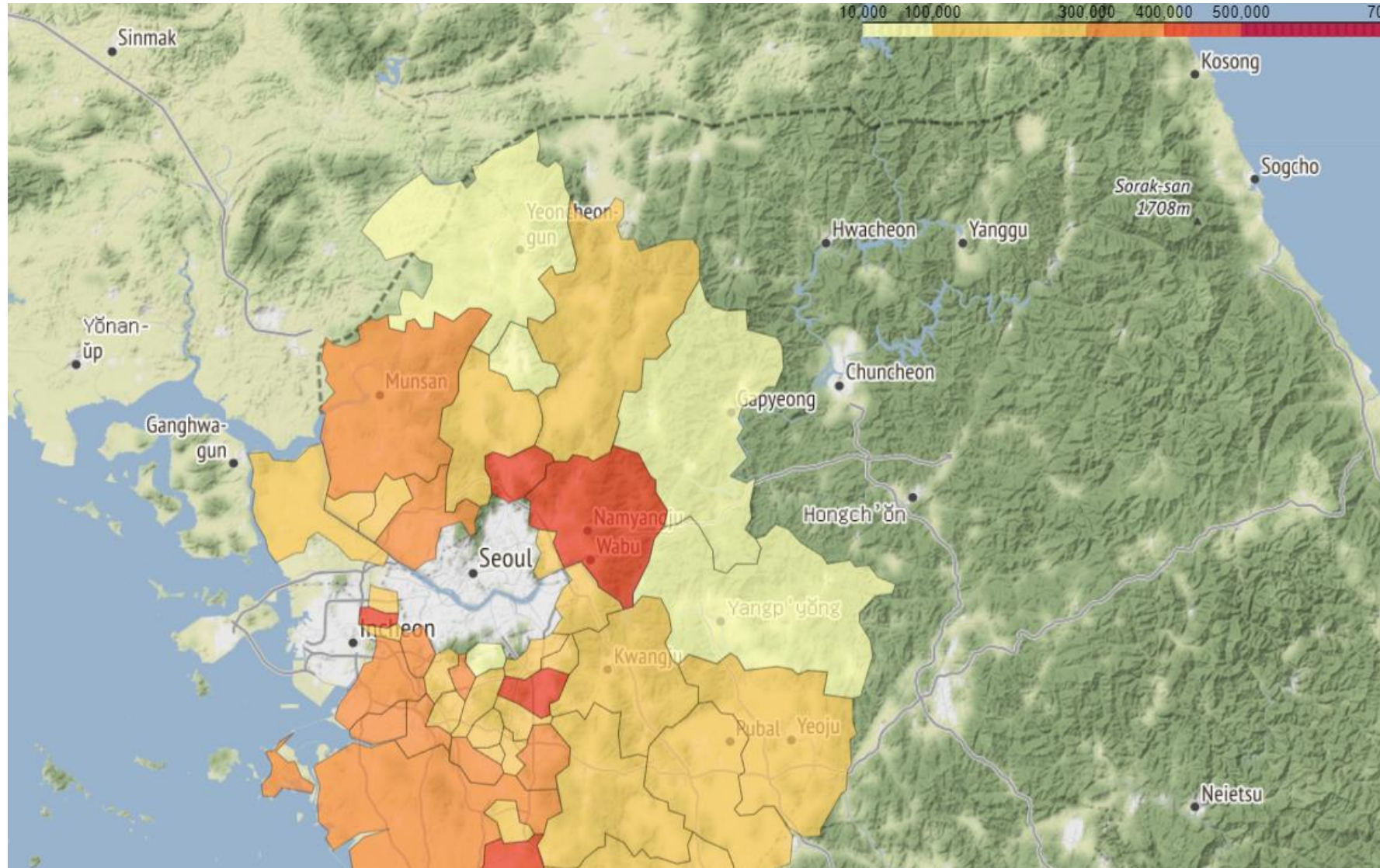
: 정보의 값에 따라 영역의
Color가 변경됨.

```
# 경기도 인구변화 데이터를 불러와서 데이터프레임으로 변환
df = pd.read_csv('./경기도인구데이터.csv')
#df.columns = df.columns.map(str)
print(df.index)
```

```
# 경기도 시군구 경계 정보를 가진 geo-json 파일 불러오기
g_geo = './경기도행정구역경계.json'
```

```
# 경기도 지도 만들기
g_map = folium.Map(location=[37.5502,126.982],
                    tiles='Stamen Terrain', zoom_start=9)
```

```
# Choropleth 클래스로 단계구분도 표시하기
folium.Choropleth(geo_data=g_geo, # 지도 경계
                  data = df,      # 표시하려는 데이터
                  columns = ['구분','2007'], # 열 지정
                  fill_color='YlOrRd', fill_opacity=0.7, line_opacity=0.3,
                  threshold_scale=[10000, 100000, 300000, 400000,500000, 700000],
                  key_on='feature.properties.name',
                  ).add_to(g_map)
```

```

},
{
  "type": "Feature",
  "properties": {
    "code": 31250,
    "name": "광주시",
    "name_eng": "Gwangju",
    "base_year": 2013
  },
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        127.30923294884336,
        37.5135706079458
      ],
      [
        127.28971082577908,

```

	A	B	C	D	E	F	G	H	I
1	구분	2007	2008	2009	2010	2011	2012	2013	2014
2	수원시장안구	287474	285803	290798	293692	290999	291757	300908	301196
3	수원시권선구	310553	308177	304097	306783	321176	332633	331773	339835
4	수원시팔달구	216945	213235	219833	216503	209772	206794	204805	203479
5	수원시영통구	252730	260210	258421	260557	266542	289074	310671	329718
6	성남시수정구	256744	248452	242161	241070	236123	233997	228167	223539
7	성남시중원구	263101	265137	259877	258093	254872	253883	256349	251982
8	성남시분당구	434115	428858	460688	481027	488328	490735	495018	499087
9	의정부시	421853	430849	431008	431801	430400	429147	430976	431112
10	안양시만안구	265881	262820	262258	266261	263077	253492	250246	247315
11	안양시동안구	358316	357459	354289	355453	352565	357920	357631	353494
12	부천시원미구	443290	441795	447136	448602	446247	446604	444207	442638
13	부천시소사구	227484	230155	229335	230557	232190	231363	229959	226400
14	부천시오정구	193263	195728	193473	196045	194015	191977	189554	186548
15	광명시	313019	310501	314257	343982	355226	355560	353100	348560
16	평택시	402458	406721	410042	419457	426886	434305	442034	449555
17	동두천시	88780	90835	93211	95653	96253	97175	97557	97595
18	안산시상록구	373969	377005	374055	379136	380880	381556	382571	381508
19	안산시단원구	331071	331252	331291	335755	334706	333552	331095	326368
20	고양시덕양구	378260	386817	388777	394375	391832	396559	410491	424423

경기도행정구역경계.json

경기도인구데이터.csv

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

CoffeeBean.csv

store	address	phone
0 학동역 DT점	서울시 강남구 학동로 211 1층	02-3444-9973
1 수서점	서울시 강남구 광평로 280 수서동 724호	02-3412-2326
2 차병원점	서울시 강남구 논현로 566 강남차병원1층	02-538-7615
3 강남대로점	서울시 서초구 강남대로 369 1층	02-588-5778
4 메가박스점	서울 강남구 삼성동 159 코엑스몰 지하2층	02-6002-3320
5 압구정에스점	서울시 강남구 압구정로 46길 3	02-541-5832
6 강남에스점	서울시 서초구 서초 1306-3호	02-593-5095
7 청담에스점	서울시 강남구 압구정로 461 네이처포엠빌딩 B108,109호	02-548-6052
8 신사점	서울시 강남구 도산대로 126	02-548-2741
9 압구정역점	서울시 강남구 논현로 842 압구정빌딩1층	02-544-6823
10 역삼점	서울시 강남구 논현로 512 지상1,2층	02-569-8051
11 양재스포타임점	서울시 서초구 강남대로 213 24호 지하1층	02-578-6833
12 청담성당점	서울시 강남구 삼성로 716 LEE76빌딩2층	02-542-2053
13 영동점	서울 서초구 반포동 736-17 P빌딩 2층	02-3443-2096
14 도곡점	서울시 강남구 언주로 30길 10,112 현대비전21 112호	02-572-2781
15 영동고앞점	서울시 강남구 선릉로 749 1,2층	02-544-3794
16 공항터미널앞점	서울시 강남구 테헤란로 87길 17 1층	02-539-5174
17 교대점	서울시 서초구 서초중앙로 118 1층	02-585-3981
18 대치한티점	서울시 강남구 선릉로 64길 23 1층	02-557-0510

참고 : 데이터 과학 기반의 파이썬 빅데이터 분석

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

■ 데이터 수집

- 행정구역 주소 체계 데이터 수집하기

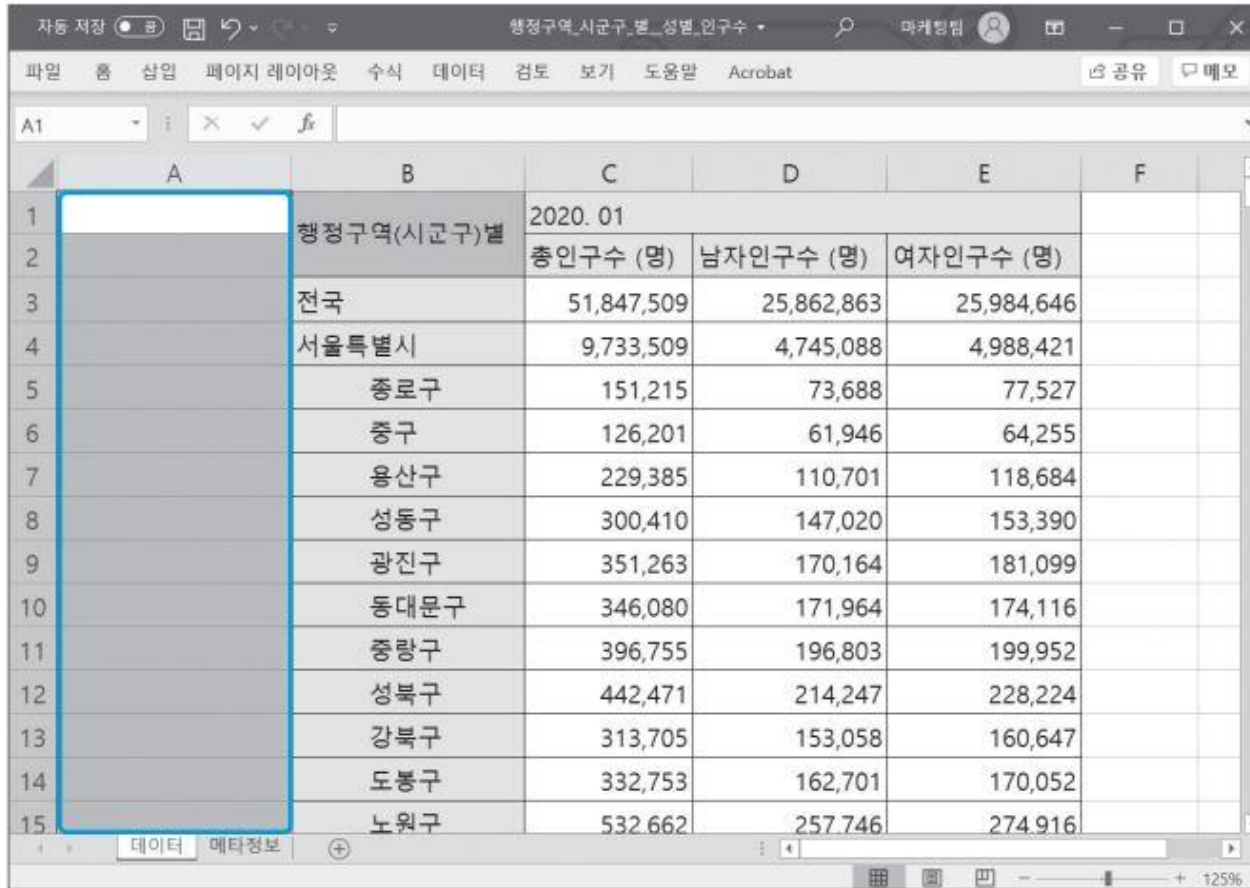
1. 국가통계포털 사이트(<http://kosis.kr>)에서 '행정구역'으로 데이터를 검색



그림 9-1 국가통계포털 사이트에서 데이터 검색

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과



행정구역, 시군구, 별, 성별, 인구수					
A	B	C	D	E	F
1		2020. 01			
2	행정구역(시군구)별	총인구수 (명)	남자인구수 (명)	여자인구수 (명)	
3	전국	51,847,509	25,862,863	25,984,646	
4	서울특별시	9,733,509	4,745,088	4,988,421	
5	종로구	151,215	73,688	77,527	
6	중구	126,201	61,946	64,255	
7	용산구	229,385	110,701	118,684	
8	성동구	300,410	147,020	153,390	
9	광진구	351,263	170,164	181,099	
10	동대문구	346,080	171,964	174,116	
11	중랑구	396,755	196,803	199,952	
12	성북구	442,471	214,247	228,224	
13	강북구	313,705	153,058	160,647	
14	도봉구	332,753	162,701	170,052	
15	노원구	532,662	257,746	274,916	

행정 구역 주소 체계 확인

그림 9-8 빈 열 삽입하기

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

■ 데이터 준비 및 탐색

- 분석할 커피 매장의 주소 데이터 준비하기
2. 주소 데이터를 행정구역 주소 체계에 맞게 정리하기

In [2]:	<pre>addr = [] for address in CB.address: addr.append(str(address).split()) addr #작업 내용 확인용 출력</pre>
Out[2]:	<pre>[['서울시', '강남구', '학동로', '211', '1층'], ['서울시', '강남구', '광평로', '280', '수서동', '724호'], ..., ['경기도', '안양시', '동안구', '시민대로', '260,', '1층', '104,105호'], ['경기도', '하남시', '미사대로', '750,', '신세계백화점', '지하1층', '식물관']]</pre>

In [2]: for 반복문을 이용하여 각 address 컬럼의 값을 분리하고 `split()` addr 리스트로 만듦.

■ 데이터 준비 및 탐색

- 분석할 커피 매장의 주소 데이터 준비하기

2. 주소 데이터를 행정구역 주소 체계에 맞게 정리하기

In [3]:	<pre>addr2 = [] for i in range(len(addr)): if addr[i][0] == "서울": addr[i][0] = "서울특별시" elif addr[i][0] == "서울시": addr[i][0] = "서울특별시" elif addr[i][0] == "부산시": addr[i][0] = "부산광역시" elif addr[i][0] == "인천": addr[i][0] = "인천광역시" elif addr[i][0] == "광주": addr[i][0] = "광주광역시" elif addr[i][0] == "대전시": addr[i][0] = "대전광역시" elif addr[i][0] == "울산시": addr[i][0] = "울산광역시" elif addr[i][0] == "세종시": addr[i][0] = "세종특별자치시" elif addr[i][0] == "경기": addr[i][0] = "경기도" elif addr[i][0] == "충북": addr[i][0] = "충청북도" elif addr[i][0] == "충남": addr[i][0] = "충청남도" elif addr[i][0] == "전북": addr[i][0] = "전라북도" elif addr[i][0] == "전남": addr[i][0] = "전라남도" elif addr[i][0] == "경북": addr[i][0] = "경상북도" elif addr[i][0] == "경남": addr[i][0] = "경상남도" elif addr[i][0] == "제주": addr[i][0] = "제주특별자치도" elif addr[i][0] == "제주도": addr[i][0] = "제주특별자치도" elif addr[i][0] == "제주시": addr[i][0] = "제주특별자치도" addr2.append(' '.join(addr[i])) addr2 #작업 내용 확인용 출력</pre>
Out[3]:	<pre>['서울특별시 강남구 학동로 211 1층', '서울특별시 강남구 광평로 280 수서동 724호', ..., '경기도 안양시 동안구 시민대로 260, 1층 104,105호', '경기도 하남시 미사대로 750, 신세계백화점 지하1층 식품관']</pre>

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

1. Geocoder-Xr를 사용하기 위해 설치하기

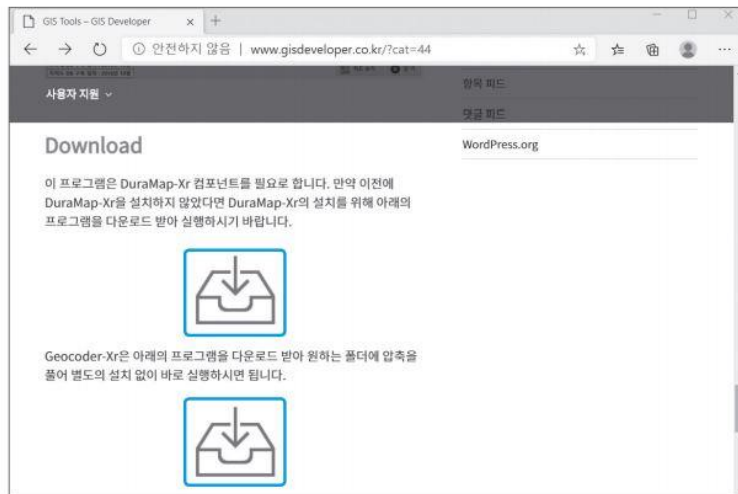


그림 9-16 DuraMap-Xr 컴포넌트와 Geocoder-Xr 다운로드 버튼

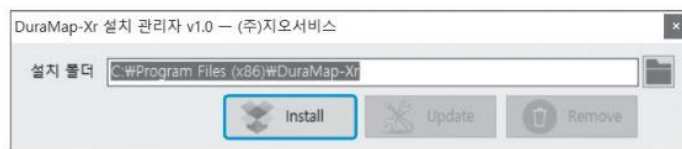


그림 9-17 DuraMap-Xr 컴포넌트 설치

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

■ 분석 모델 구축 및 시각화

- 지도 객체에 커피 매장 위치 표시하기

3. 주소의 좌표 구하기

- 1 Geocoder-Xr을 실행한 후 [입력 파일]을 클릭해 CoffeeBean_2.csv로 선택
- 2 좌표를 구할 주소가 있는 [주소필드]를 address2로 설정,
- 3 결과를 저장할 파일 경로를 나타내는 [결과 SHP 파일]에 '9장_data/CB_geo.shp'로 입력.
- 4 결과를 CSV 파일 형태로도 저장'을 체크해서 선택
- 5 <시작> 버튼을 클릭



그림 9-19 좌표 구하기 1 - 필요 항목 설정

참고 : 데이터 과학 기반의 파이썬 빅데이터 분석

주소를 이용해 위도 경도 구하기

인하공전 컴퓨터 정보 과

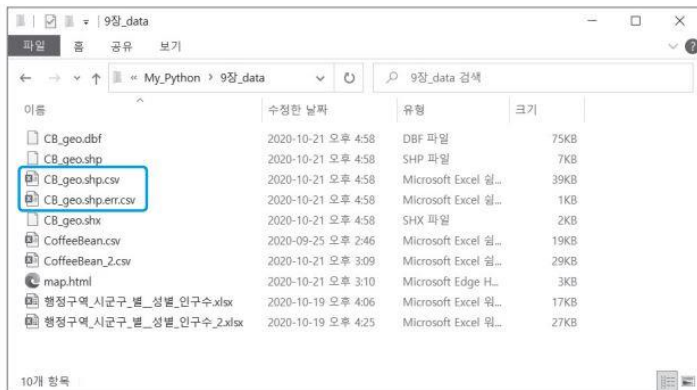


그림 9-20 좌표 구하기 2 - 주소의 좌표 변환 원료 후 생성된 파일

No	store	address	phone	address2	경도	위도	상태
1	1	학동역 DT 서울시 강	02-3444-9	서울특별시	127.032	37.5147	정좌표
2	2	수서점 서울시 강	02-3412-2	서울특별시	127.103	37.4873	정좌표
3	3	차병원점 서울시 강	02-538-76	서울특별시			
4	4	강남대로점 서울시 서	02-588-57	서울특별시			
5	5	메가박스점 서울 강남	02-6002-3	서울특별시			
6	6	압구정에스 서울시 강	02-541-58	서울특별시			
7	7	강남에스점 서울시 서	02-593-50	서울특별시	?	?	실패
8	23	논현팩스E 서울시 강	02-513-38	서울특별시	?	?	실패
9	50	순화점 서울시 중	02-2220-8	서울특별시	?	?	실패
10	60	신촌점 서울시 서	02-363-55	서울특별시	?	?	실패
11	87	무교점 서울시 중	02-753-23	서울특별시	?	?	실패
12	164	고대참살0 서울시 성	02-923-77	서울특별시	?	?	실패
13	174	강서그랜드 서울시 강	02-2699-7	서울특별시	?	?	실패

그림 9-21 변환된 주소 좌표를 나타내는 CB_geo.shp.csv와 변환을 하지 못한 항목을 모아둔 CB_geo.shp.err.csv

```
!pip install geopandas
```

```
import geopandas as gdp
```

```
pth=gpd.datasets.get_path('naturalearth_lowres')
```

```
world=gpd.GeoDataFrame.from_file(pth)
```

```
print(world.head())
```

```
world.plot(figsize=(12,12))
```

Data structure : GeoSeries or GeoDataFrame

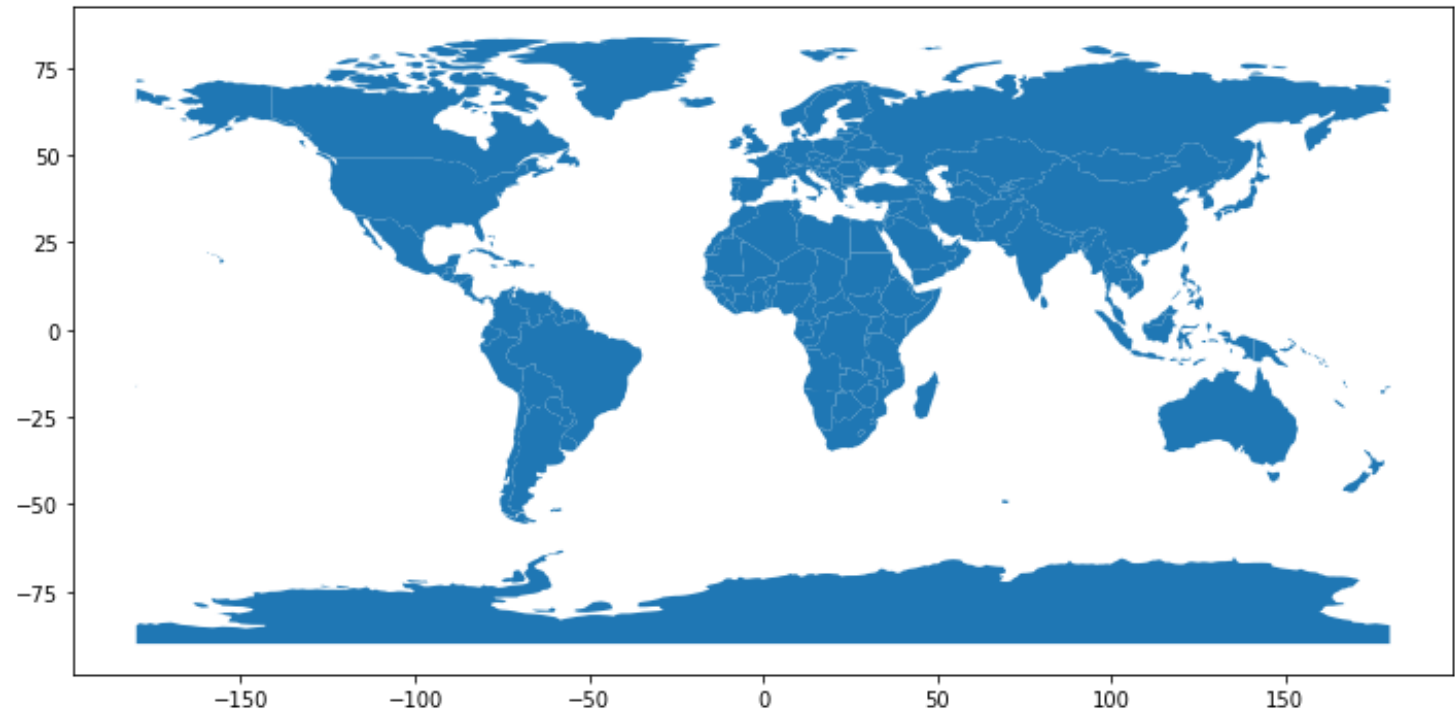
datasets : 'naturalearth_cities' : 세계 도시

 'naturalearth_lowres' : 국가

 'nybb' : 뉴욕 지도

	pop_est	continent	name	iso_a3	gdp_md_est	₩
0	920938	Oceania	Fiji	FJI	8374.0	
1	53950935	Africa	Tanzania	TZA	150600.0	
2	603253	Africa	W. Sahara	ESH	906.5	
3	35623680	North America	Canada	CAN	1674000.0	
4	326625791	North America	United States of America	USA	18560000.0	

	geometry
0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...



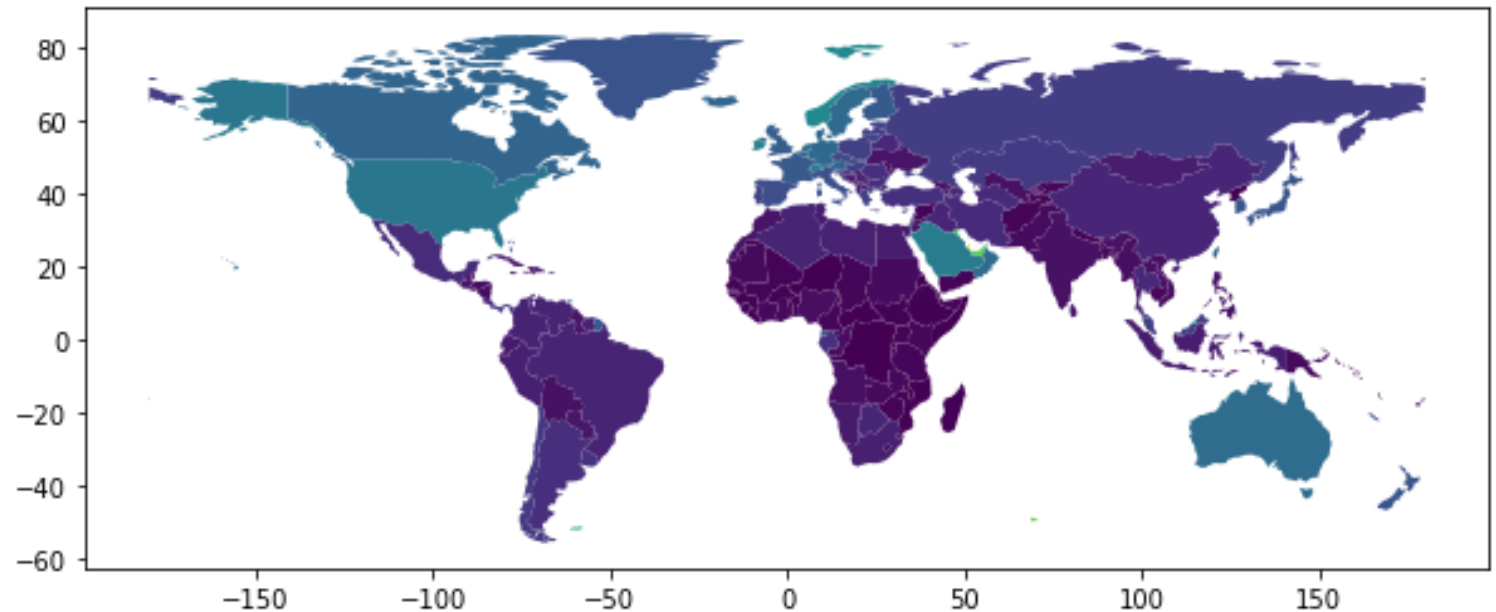
Choropleth Maps : `plot(column=`)

Plot by GDP per capita

```
world = world[(world.pop_est>0) & (world.name!="Antarctica")]
```

```
world['gdp_per_cap'] = world.gdp_md_est / world.pop_est # 1인당 소득
```

```
world.plot(column='gdp_per_cap');
```

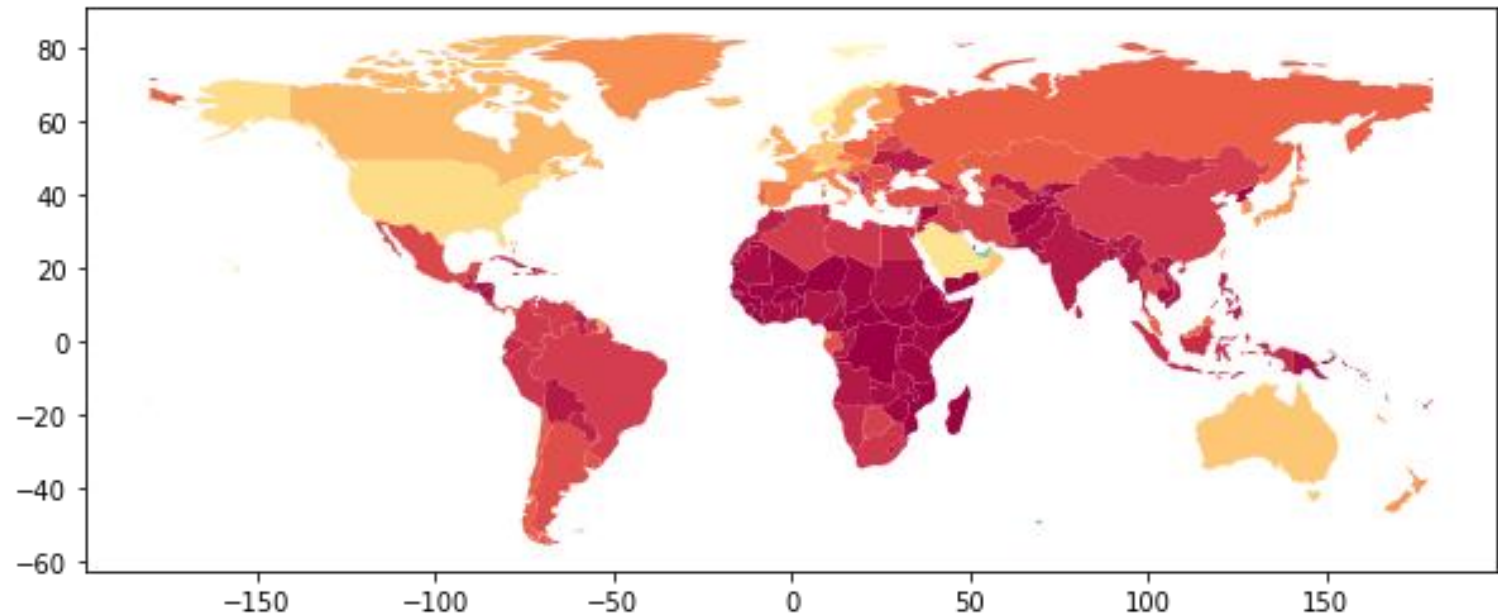


Choropleth Maps : `plot(column=`)

```
world = world[(world.pop_est>0) & (world.name!="Antarctica")]
```

```
world['gdp_per_cap'] = world.gdp_md_est / world.pop_est
```

```
world.plot(column='gdp_per_cap',cmap='Spectral',figsize=(10,10));
```



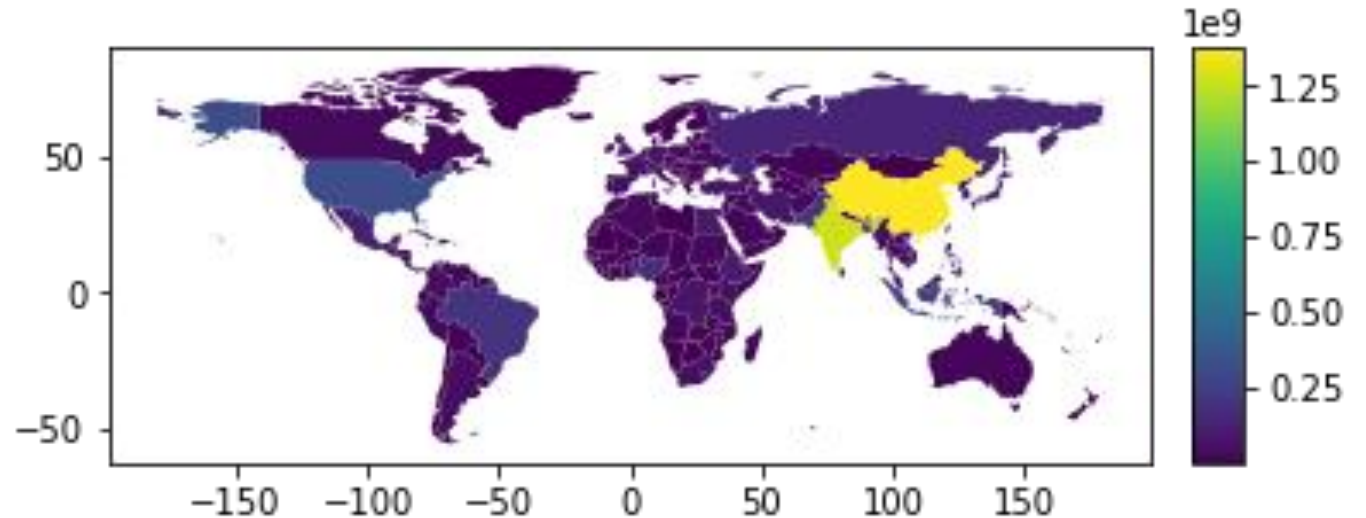

```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

fig, ax = plt.subplots(1, 1)

divider = make_axes_locatable(ax)

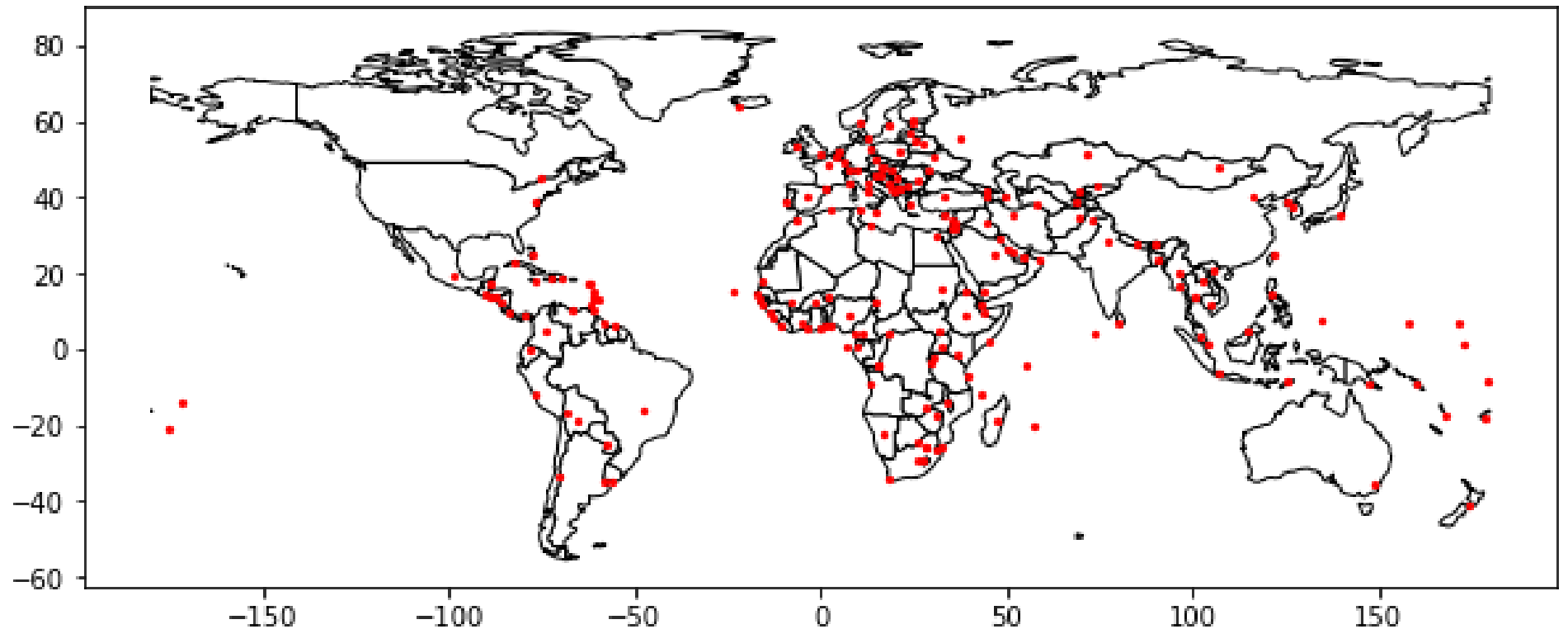
cax = divider.append_axes("right", size="5%", pad=0.1)

world.plot(column='pop_est', ax=ax, legend=True, cax=cax)
```



```
import geopandas as gpd
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
base = world.plot(color='white', edgecolor='black',figsize=(10,10))

cities.plot(ax=base, marker='o', color='red', markersize=5)
```



수고하셨습니다

jhmin@inhatec.ac.kr