BIGDATA TERM PROJECT

인구 소멸 위기 지역 분석

울산대학교 UNIVERSITY OF ULSAN

IT융합학부
20194210 이재욱

CONTENTS

01 주제 선정

02 데이터 처리분석

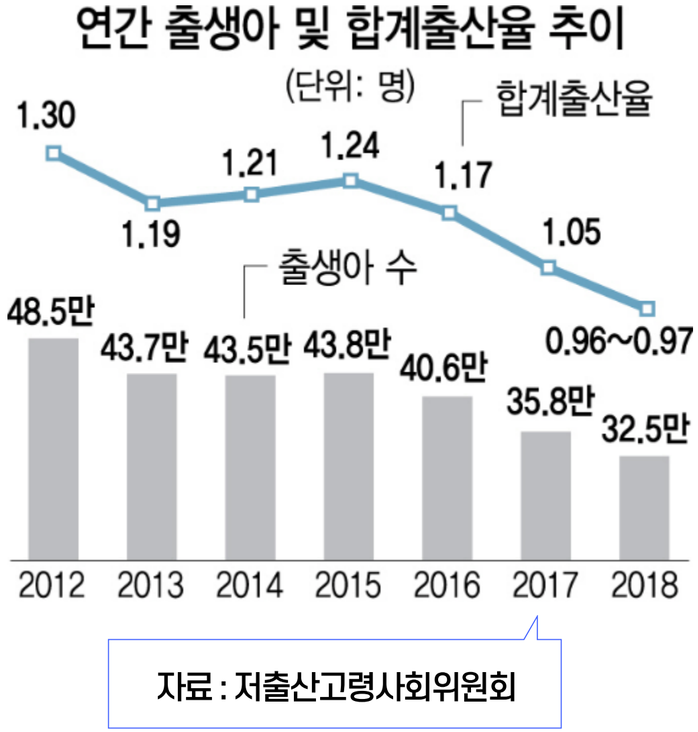
1. 데이터 확보
2. 데이터 전처리
3. 데이터 저장

03 시각화

1. 지역별 고유 ID 생성
2. 우리나라 지도 만들기
3. Folium

01 주제선정

저출산 고령화 시대



저출산 고령화 시대가 이루어지면서
'우리나라 인구 소멸 위기 지역' 이라는 주제로 분석을 수행,
인구 소멸 위기 지역을 시각화 하여 심각성을 느낄 수 있도록 주제를 선정.

02 데이터 처리 분석

1. 데이터 확보

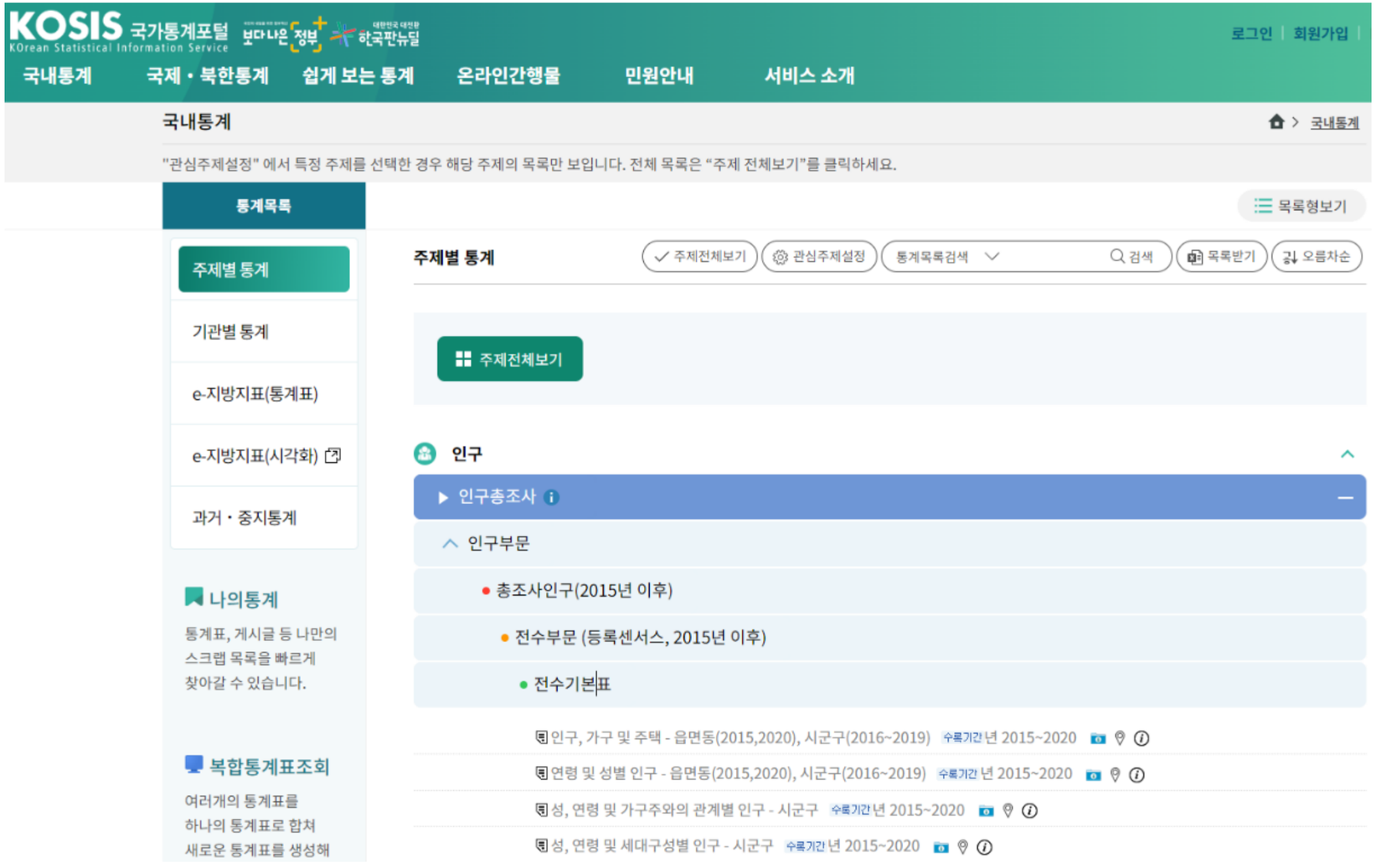
2. 데이터 전처리

3. 데이터 저장

인구 데이터

국가통계포털 (http://kosis.kr/index/index.do)에서

인구 > 인구총조사 > 인구부문 > 총조사인구 > 전수부문 > 전수기본표 > 연령 및 성별 인구에서 인구데이터 다운로드 (최신 2020)



02 데이터 처리 분석

1. 데이터 확보

2. 데이터 전처리

3. 데이터 저장

인구 소멸 위기 지역 정의

65세 이상 노인 인구와 20~39세 여성 인구를 비교하여

20~30대 여성 인구가 노인 인구의 절반(50%)미만인 경우 -> 인구 소멸 위기 지역,

20~30대 여성 인구가 노인 인구의 20%미만인 경우 -> 인구 소멸 고위험 지역으로 정의

연령대 구분

20~39세와 65세 이상으로 연령대를 구분

```
df['20~39세'] = df['20 - 24세'] + df['25 - 29세'] + df['30 - 34세'] + df['35 - 39세']
df['65세이상'] = df['65 - 69세'] + df['70 - 74세'] + df['75 - 79세'] + df['80세 이상']
df = df[['광역시도','시군구','구분','인구수','20~39세','65세이상']]
df.head()
```

	광역시도	시군구	구분	인구수	20~39세	65세이상
6	서울특별시	종로구	계	144866	43061	25597
7	서울특별시	종로구	남자	70613	21327	11361
8	서울특별시	종로구	여자	74253	21736	14236
9	서울특별시	중구	계	121520	38455	21878
10	서울특별시	중구	남자	59536	18864	9600

02 데이터 처리 분석

1. 데이터 확보

2. 데이터 전처리

3. 데이터 저장

그룹핑

pivot_table을 만들어서 시군구 단위로 grouping

광역시도, 시도를 index로 두고,

구분으로 세로를 첫 번째 컬럼을 설정,

value에 인구수, 20~39세, 65세 이상으로 정리

```
pop = pd.pivot_table(df,
                     index=['광역시도','시군구'], columns=['구분'],
                     values=['인구수','20~39세','65세이상'])
pop.head()
```

		20~39세			65세이상			인구수		
	구분	계	남자	여자	계	남자	여자	계	남자	여자
광역시도	시군구									
강원도	강릉시	47006	25199	21807	43830	18604	25227	211643	105025	106618
	고성군	5217	3175	2042	7755	3237	4519	26792	13864	12928
	동해시	18727	10469	8258	17572	7624	9949	89814	45572	44242
	삼척시	13583	7469	6117	16077	6766	9311	65623	33275	32348
	속초시	18080	9909	8171	15170	6345	8826	81497	40312	41185

인구 소멸 비율 계산

20~39세 여성 / 65세 이상 노인

```
# 인구 소멸비율 계산
pop['소멸비율'] = pop['20~39세','여자'] / pop['65세이상','계']
pop.head()
```

		20~39세			65세이상			인구수			소멸비율
	구분	계	남자	여자	계	남자	여자	계	남자	여자	
광역시도	시군구										
강원도	강릉시	47006	25199	21807	43830	18604	25227	211643	105025	106618	0.497536
	고성군	5217	3175	2042	7755	3237	4519	26792	13864	12928	0.263314
	동해시	18727	10469	8258	17572	7624	9949	89814	45572	44242	0.469952
	삼척시	13583	7469	6117	16077	6766	9311	65623	33275	32348	0.380481
	속초시	18080	9909	8171	15170	6345	8826	81497	40312	41185	0.538629

02 데이터 처리 분석

1. 데이터 확보

2. 데이터 전처리

3. 데이터 저장

소멸위기지역 분석과 출력

소멸위기지역유무를 boolean으로 지정해둠

```
pop['소멸위기지역'] = pop.소멸비율 < 0.5
pop['소멸위기고위험지역'] = pop.소멸비율 < 0.2
pop.head()
```

		20~39세			65세이상			인구수			소멸비율	소멸위기지역	소멸위기고위험지역
	구분	계	남자	여자	계	남자	여자	계	남자	여자			
광역시도	시군구												
강원도	강릉시	47006	25199	21807	43830	18604	25227	211643	105025	106618	0.497536	True	False
	고성군	5217	3175	2042	7755	3237	4519	26792	13864	12928	0.263314	True	False
	동해시	18727	10469	8258	17572	7624	9949	89814	45572	44242	0.469952	True	False
	삼척시	13583	7469	6117	16077	6766	9311	65623	33275	32348	0.380481	True	False
	속초시	18080	9909	8171	15170	6345	8826	81497	40312	41185	0.538629	False	False

```
# 인구소멸 위기지역
crisis_region = pop[pop.소멸위기지역].index.get_level_values(1)
print(crisis_region)
```

```
Index([' 강릉시', ' 고성군', ' 동해시', ' 삼척시', ' 양구군', ' 양양군', ' 영월군', ' 인제군', ' 정선군',
       ' 철원군',
       ...
       ' 청양군', ' 태안군', ' 홍성군', ' 괴산군', ' 단양군', ' 보은군', ' 영동군', ' 옥천군', ' 음성군',
       ' 제천시'],
      dtype='object', name='시군구', length=102)
```

```
# 인구소멸위기 고위험지역
high_crisis_region = pop[pop.소멸위기고위험지역].index.get_level_values(1)
print(high_crisis_region)
```

```
Index([' 남해군', ' 산청군', ' 의령군', ' 하동군', ' 합천군', ' 군위군', ' 봉화군', ' 영덕군', ' 영양군',
       ' 의성군', ' 청도군', ' 청송군', ' 고흥군', ' 곡성군', ' 보성군', ' 신안군', ' 함평군', ' 임실군',
       ' 부여군', ' 서천군', ' 청양군', ' 보은군'],
      dtype='object', name='시군구')
```

02 데이터 처리 분석

1. 데이터 확보

2. 데이터 전처리

3. 데이터 저장

인구소멸 위기/고위험 지역 출력

다단으로 구성된 MultiIndex 초기화 후, 전처리가 완료된 데이터프레임을 csv파일로 저장

```
tmp_col = [pop.columns.get_level_values(0)[i] + pop.columns.get_level_values(1)[i]
           for i in range(len(pop.columns.get_level_values(0)))]
pop.columns = tmp_col
pop.reset_index(inplace=True)
pop['시군구'] = pop.시군구.apply(lambda x: x.strip())
pop.head()
```

	광역시도	시군구	20~39세계	20~39세남자	20~39세여자	65세이상계	65세이상남자	65세이상여자	인구수계	인구수남자	인구수여자	소멸비율	소멸위기지역	소멸위기고위험지역
0	강원도	강릉시	47006	25199	21807	43830	18604	25227	211643	105025	106618	0.497536	True	False
1	강원도	고성군	5217	3175	2042	7755	3237	4519	26792	13864	12928	0.263314	True	False
2	강원도	동해시	18727	10469	8258	17572	7624	9949	89814	45572	44242	0.469952	True	False
3	강원도	삼척시	13583	7469	6117	16077	6766	9311	65623	33275	32348	0.380481	True	False
4	강원도	속초시	18080	9909	8171	15170	6345	8826	81497	40312	41185	0.538629	False	False

```
pop.to_csv('./data/1.시군구_전처리완료.csv', encoding='euc-kr', index=False)
```

강릉시, 고성군, 동해시, 삼척시는 소멸위기지역이나, 속초시는 소멸위기지역이 아니라는 것을 확인할 수 있음

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

지도 시각화를 위해 지역별 고유 ID 만들기

울산, 부산 등 광역시의 구(자치구)도 있으나, 수원시, 화성시 등에 있는 구(행정구)도 있음.

해당 값을 하나만 출력하도록 unique( ) 함수를 사용하여 처리.

```
pop.시군구.unique()
```

```
array(['강릉시', '고성군', '동해시', '삼척시', '속초시', '양구군', '양양군', '영월군', '원주시',
       '인제군', '정선군', '철원군', '춘천시', '태백시', '평창군', '홍천군', '화천군', '횡성군',
       '가평군', '과천시', '광명시', '광주시', '구리시', '군포시', '권선구', '기흥구', '김포시',
       '남양주시', '단원구', '덕양구', '동두천시', '동안구', '만안구', '부천시', '분당구', '상록구',
       '수정구', '수지구', '시흥시', '안성시', '양주시', '양평군', '여주시', '연천군', '영통구',
       '오산시', '의왕시', '의정부시', '이천시', '일산동구', '일산서구', '장안구', '중원구', '처인구',
       '파주시', '팔달구', '평택시', '포천시', '하남시', '화성시', '거제시', '거창군', '김해시',
       '남해군', '마산합포구', '마산회원구', '밀양시', '사천시', '산청군', '성산구', '양산시', '의령군',
       '의창구', '진주시', '진해구', '창녕군', '통영시', '하동군', '함안군', '함양군', '합천군',
       '경산시', '경주시', '고령군', '구미시', '군위군', '김천시', '남구', '문경시', '봉화군', '북구',
       '상주시', '성주군', '안동시', '영덕군', '영양군', '영주시', '영천시', '예천군', '울릉군',
       '울진군', '의성군', '청도군', '청송군', '칠곡군', '광산구', '동구', '서구', '달서구', '달성군',
       '수성구', '중구', '대덕구', '유성구', '강서구', '금정구', '기장군', '동래구', '부산진구',
       '사상구', '사하구', '수영구', '연제구', '영도구', '해운대구', '강남구', '강동구', '강북구',
       '관악구', '광진구', '구로구', '금천구', '노원구', '도봉구', '동대문구', '동작구', '마포구',
       '서대문구', '서초구', '성동구', '성북구', '송파구', '양천구', '영등포구', '용산구', '은평구',
       '종로구', '중랑구', '세종시', '울주군', '강화군', '계양구', '남동구', '미추홀구', '부평구',
       '연수구', '옹진군', '강진군', '고흥군', '곡성군', '광양시', '구례군', '나주시', '담양군',
       '목포시', '무안군', '보성군', '순천시', '신안군', '여수시', '영광군', '영암군', '완도군',
       '장성군', '장흥군', '진도군', '함평군', '해남군', '화순군', '고창군', '군산시', '김제시',
       '남원시', '덕진구', '무주군', '부안군', '순창군', '완산구', '완주군', '익산시', '임실군',
       '장수군', '정읍시', '진안군', '서귀포시', '제주시', '계룡시', '공주시', '금산군', '논산시',
       '당진시', '동남구', '보령시', '부여군', '서북구', '서산시', '서천군', '아산시', '예산군',
       '청양군', '태안군', '홍성군', '괴산군', '단양군', '보은군', '상당구', '서원구', '영동군',
       '옥천군', '음성군', '제천시', '증평군', '진천군', '청원구', '충주시', '흥덕구'],
      dtype=object)
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

광역시가 아니면서 구를 갖고 있는 시와 그 행정구를 '딕셔너리형'으로 선언

```
# 고성군 - 고성(강원), 고성(경남)
# 광역시 - 서울 용산, 서울 서대문, 대전 서구, 대전 유성, 세종
# 행정구 - 수원 장안, 용인 수지, 고양 일산동, 창원 합포, 창원 회원
tmp_gu_dict = {
    '수원': ['장안구', '권선구', '팔달구', '영통구'],
    '성남': ['수정구', '중원구', '분당구'],
    '안양': ['만안구', '동안구'],
    '안산': ['상록구', '단원구'],
    '고양': ['덕양구', '일산동구', '일산서구'],
    '용인': ['처인구', '기흥구', '수지구'],
    '청주': ['상당구', '서원구', '흥덕구', '청원구'],
    '천안': ['동남구', '서북구'],
    '전주': ['완산구', '덕진구'],
    '포항': ['남구', '북구'],
    '창원': ['의창구', '성산구', '진해구', '마산합포구', '마산회원구']
}
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

반복문에서 조건문을 사용하여 데이터 정리

지도 시각화에 사용하기 위해 위 과정에서 만들어진 행정구역의 고유한 이름을 ID로 지정

'광역시', '특별시', '자치시'로 끝나지 않으면 일반 시 혹은 군으로 봄

'세종특별자치시'는 그냥 '세종'으로 처리

나머지는 광역시도 앞 에서 두 글자(서울특별시)와 시도에서 두 글자인 경우 모두, 아니면 앞 두 글자만 선택하면서 고유 ID 생성

```
metro_list = ['서울특별시','부산광역시','대구광역시','인천광역시','대전광역시','광주광역시','울산광역시']
si_name = [None] * len(pop)

for i in pop.index:
    if pop.광역시도[i] in metro_list:
        if len(pop.시군구[i]) == 2:
            si_name[i] = pop.광역시도[i][:2] + ' ' + pop.시군구[i]
        else:
            si_name[i] = pop.광역시도[i][:2] + ' ' + pop.시군구[i][:-1]     # 긴 구 이름에서 '구' 제외
    else:
        if pop.시군구[i][:-1] == '고성':
            if pop.광역시도[i] == '강원도':
                si_name[i] = '고성(강원)'
            else:
                si_name[i] = '고성(경남)'
        else:
            si_name[i] = pop.시군구[i][:-1]

        for key, values in tmp_gu_dict.items():
            if pop.시군구[i] in values:
                if len(pop.시군구[i]) == 2:
                    si_name[i] = key + ' ' + pop.시군구[i]
                elif pop.시군구[i] in ['마산합포구', '마산회원구']:
                    si_name[i] = key + ' ' + pop.시군구[i][2:-1]
                else:
                    si_name[i] = key + ' ' + pop.시군구[i][:-1]
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

결과 확인 및 저장

만들어진 행정 구역의 고유한 이름을 'ID' 로 지정

지도시각화에서 의미 없는 컬럼들 제거

csv 파일로 저장

```
pop['ID'] = si_name
del pop['20~39세남자']
del pop['65세이상남자']
del pop['65세이상여자']
pop.head()
```

	광역시도	시군구	20~39세계	20~39세여자	65세이상계	인구수계	인구수남자	인구수여자	소멸비율	소멸위기지역	소멸위기고위험지역	ID
0	강원도	강릉시	47006	21807	43830	211643	105025	106618	0.497536	True	False	강릉
1	강원도	고성군	5217	2042	7755	26792	13864	12928	0.263314	True	False	고성(강원)
2	강원도	동해시	18727	8258	17572	89814	45572	44242	0.469952	True	False	동해
3	강원도	삼척시	13583	6117	16077	65623	33275	32348	0.380481	True	False	삼척
4	강원도	속초시	18080	8171	15170	81497	40312	41185	0.538629	False	False	속초

```
pop.to_csv('./data/2.ID부여완료.csv', encoding='euc-kr', index=False)
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

ID로 나눈 참고 데이터 불러오기

만들어져 있는 파일을 다운로드 받아옴

```
draw_korea_raw = pd.read_excel('./data/draw_korea_raw.xlsx')
draw_korea_raw
```

각 지역별 위치 x , y 좌표

각 행정 구역의 화면상 좌표를 얻기 위해 인덱스를 재설정(reset_index)

변수 이름을 다시 설정

```
draw_korea_raw_stacked = pd.DataFrame(draw_korea_raw.stack())
draw_korea_raw_stacked.reset_index(inplace=True)
draw_korea_raw_stacked.rename(columns={'level_0':'y', 'level_1':'x', 0:'ID'},
                              inplace=True)

draw_korea_raw_stacked.head()
```

	y	x	ID
0	0	7	철원
1	0	8	화천
2	0	9	양구
3	0	10	고성(강원)
4	1	3	양주

```
draw_korea = draw_korea_raw_stacked
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

경계선, 지도 그리기

'광역시도' 를 구분하는 경계선 입력 (코드 생략)

itterows( ) : Pandas에서 반복을 효율적으로 처리하는 방법

splitlines( ) : 개행을 기준으로 문자열을 나눔

invert_yaxis( ) : y축을 반대로 변경

```
plt.figure(figsize = (8, 11))

# 지역 이름 표시
for idx, row in draw_korea.iterrows():

    # 광역시는 구 이름이 겹치는 경우가 많아서 시 단위 이름도 같이 표시!
    # (중구, 서구)
    if len(row['ID'].split()) == 2:
        dispname = '{}₩n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
    elif row['ID'][:2] == '고성':
        dispname = '고성'
    else:
        dispname = row['ID']

    # '서대문구', '서귀포시'와 같이 이름이 3자 이상인 경우에는 작은 글자로 표시
    if len(dispname.splitlines()[-1]) >= 3:
        fontsize, linespacing = 9.5, 1.5
    else:
        fontsize, linespacing = 11, 1.2

    # annotate(): 그래프에 화살표를 그린후, 그 화살표에 문자열을 출력하는 기능을 수행
    plt.annotate(dispname, (row['x'] + 0.5, row['y'] + 0.5), weight = 'bold',
                 fontsize = fontsize, ha = 'center', va = 'center',
                 linespacing = linespacing)


# '시도' 경계를 그려주기
for path in BORDER_LINES:
    ys, xs = zip(*path)
    plt.plot(xs, ys, c = 'black', lw = 1.5)

# ivert_yaxis()는 y축이 엑셀에서 0번이 시작하는 것과 matplotlib이 0이라고 인식하는 좌표가 서로 반대이기 때문에 사용!
plt.gca().invert_yaxis()
# plt.gca().set_aspect(1)

plt.axis('off')

plt.tight_layout()
plt.show()
```

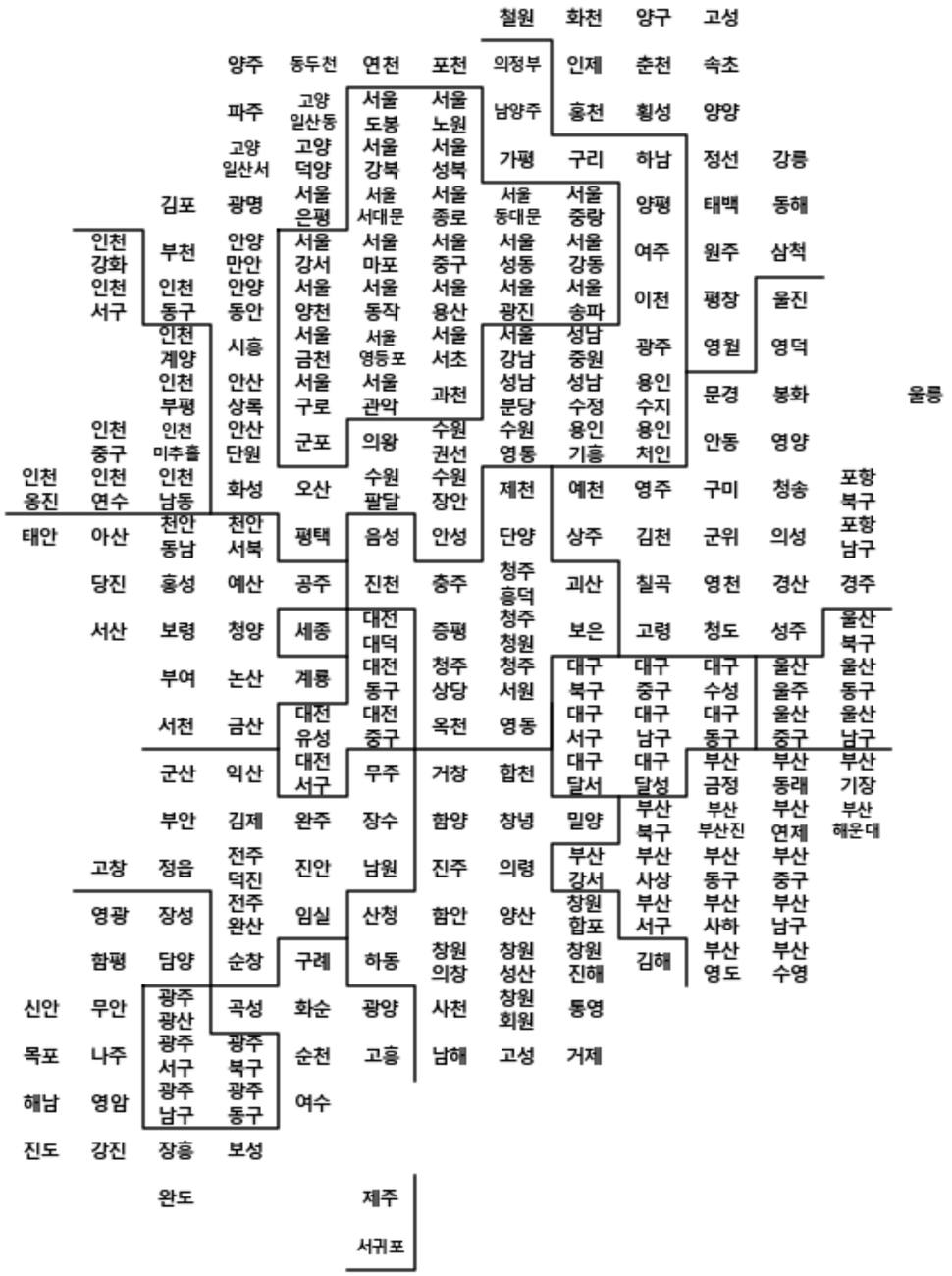
03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

경계선, 지도 그리기



03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

ID를 key로 merge

pop이랑 draw_korea의 ID 컬럼이 일치하다고 판단한 후 ID를 key로 merge

```
pop = pd.merge(pop, draw_korea, how = 'left', on = ['ID'])

pop.head()
```

	광역시도	시군구	20~39세계	20~39세여자	65세이상계	인구수계	인구수남자	인구수여자	소멸비율	소멸위기지역	소멸위기고위험지역	ID	y	x
0	강원도	강릉시	47006	21807	43830	211643	105025	106618	0.497536	True	False	강릉	3	11
1	강원도	고성군	5217	2042	7755	26792	13864	12928	0.263314	True	False	고성(강원)	0	10
2	강원도	동해시	18727	8258	17572	89814	45572	44242	0.469952	True	False	동해	4	11
3	강원도	삼척시	13583	6117	16077	65623	33275	32348	0.380481	True	False	삼척	5	11
4	강원도	속초시	18080	8171	15170	81497	40312	41185	0.538629	False	False	속초	1	10

데이터에 정보 그리기

np.isnan( ): 배열에 NaN(Not a Number) 포함 여부 확인 함수

np.ma.masked_where(condition, a, copy = True): a를 조건이 True인 값들에 대해서 '--'로 마스킹해서 array 형태로 반환

```
mapdata = pop.pivot_table(index = 'y', columns = 'x', values = '인구수계')
masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

colormap

colormap을 완성하는 명령을 추가하여 drawKorea( ) 함수 생성

```
def drawKorea(targetData, blockedMap, cmapname):
    gamma = 0.75

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

    plt.figure(figsize=(9, 11))
    plt.pcolor(masked_mapdata, vmin=vmin, vmax=vmax, cmap=cmapname,
               edgecolor='#aaaaaa', linewidth=0.5)

    # 지역 이름 표시
    for idx, row in blockedMap.iterrows():
        # 광역시는 구 이름이 겹치는 경우가 많아서 시단위 이름도 같이 표시한다.
        #(중구, 서구)
        if len(row['ID'].split())==2:
            dispname = '{}₩n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
        elif row['ID'][:2]=='고성':
            dispname = '고성'
        else:
            dispname = row['ID']

        # 서대문구, 서귀포시 같이 이름이 3자 이상인 경우에 작은 글자로 표시한다.
        if len(dispname.splitlines()[-1]) >= 3:
            fontsize, linespacing = 10.0, 1.1
        else:
            fontsize, linespacing = 11, 1.

        annocolor = 'white' if row[targetData] > whitelabelmin else 'black'
        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 그린다.
    for path in BORDER_LINES:
        ys, xs = zip(*path)
        plt.plot(xs, ys, c='black', lw=2)

    plt.gca().invert_yaxis()

    plt.axis('off')

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

    plt.tight_layout()
    plt.show()
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

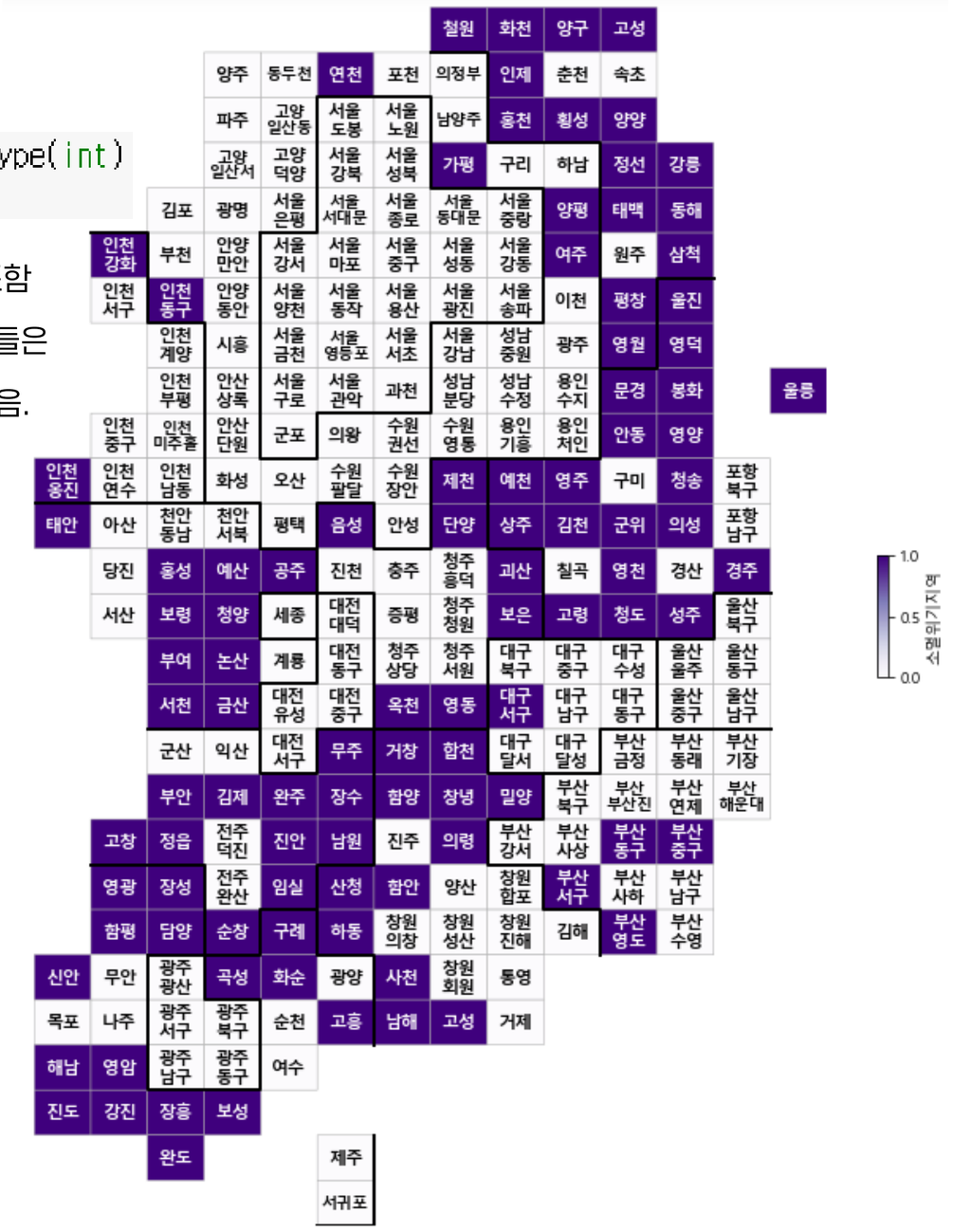
인구 소멸 지역 확인

```
pop['소멸위기지역'] = pop.소멸위기지역.astype(int)
drawKorea('소멸위기지역', pop, 'Purples')
```

광역시 중 부산은 동구, 중구, 서구, 영도가 위험지역에 포함

전체적으로 수도권과 일부 광역시를 제외한 나머지 지역들은

인구 밀집 현상이 심한 인구 소멸 위기 지역임을 알 수 있음.



03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium

Index 설정

pop 데이터에서 ID 컬럼을 index로 설정

```
pop_folium = pop.set_index('ID')
pop_folium.head()
```

ID	광역시도	시군구	20~39세계	20~39세여자	65세이상계	인구수계	인구수남자	인구수여자	소멸비율	소멸위기지역	소멸위기고위험지역	y	x
강릉	강원도	강릉시	47006	21807	43830	211643	105025	106618	0.497536	1	False	3	11
고성(강원)	강원도	고성군	5217	2042	7755	26792	13864	12928	0.263314	1	False	0	10
동해	강원도	동해시	18727	8258	17572	89814	45572	44242	0.469952	1	False	4	11
삼척	강원도	삼척시	13583	6117	16077	65623	33275	32348	0.380481	1	False	5	11
속초	강원도	속초시	18080	8171	15170	81497	40312	41185	0.538629	0	False	1	10

json 파일 연결하여 Folium으로 표현

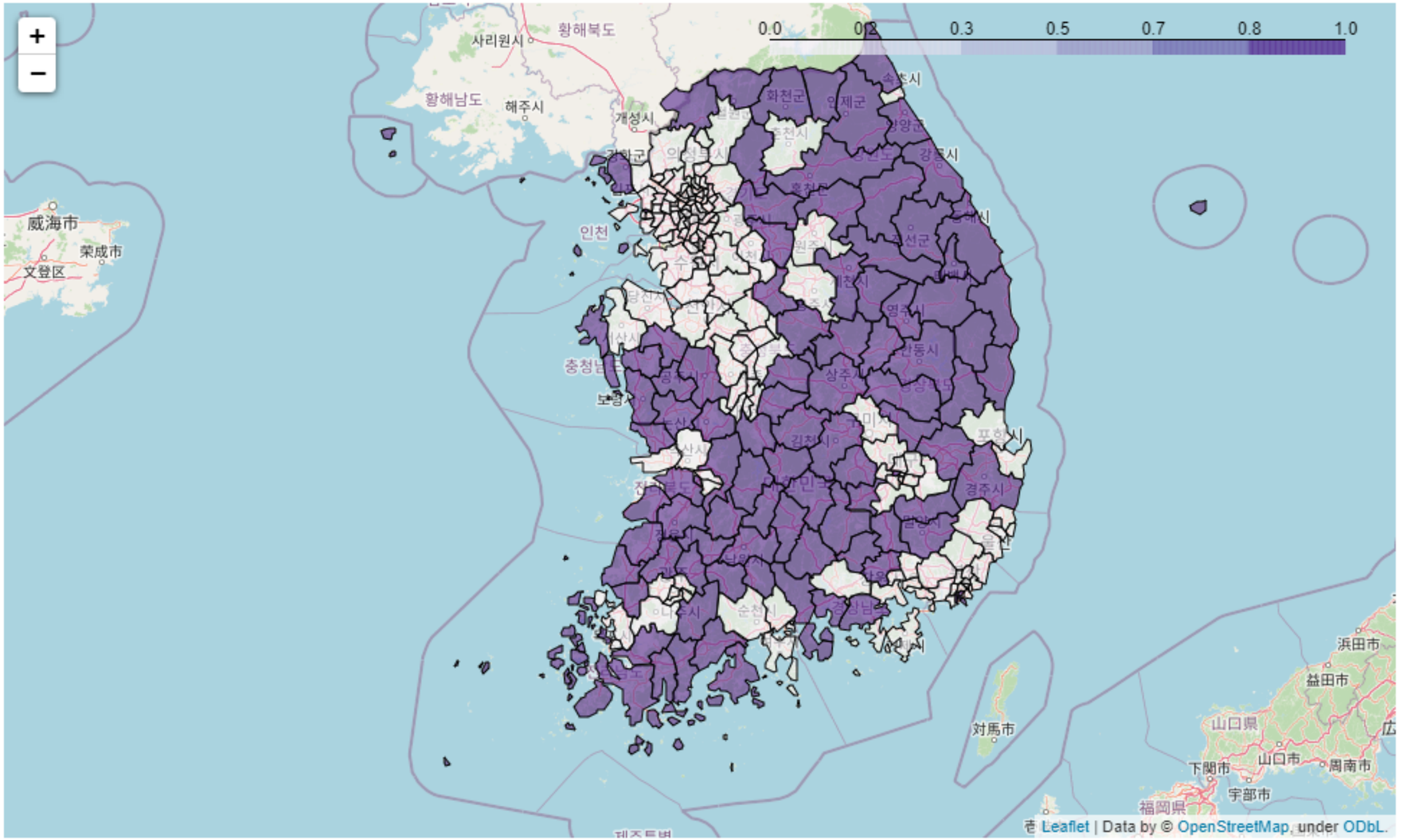
```
geo_path = './data/skorea_municipalities_geo_simple.json'
geo_str = json.load(open(geo_path, encoding = 'utf-8'))

map = folium.Map(location = [36.2002, 127.054], zoom_start = 7)
map.choropleth(geo_data = geo_str,
               data = pop_folium['소멸위기지역'],
               columns = [pop_folium.index, pop_folium['소멸위기지역']],
               fill_color = 'Purples',
               key_on = 'feature.id')

map
```

03 시각화

1. 지역별 고유 ID 생성

2. 우리나라 지도 만들기

3. Folium



감사합니다