Smarcle data study

- 19대 대선 결과 분석 -

데이터 분석 스터디 1팀
강인영, 노지민, 유정수, 정유찬

폰트 설치 및 라이브러리 불러오기

- 폰트 설치

```
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

```
import matplotlib.pyplot as plt

plt.rc('font', family='NanumBarunGothic')
```

- 라이브러리 불러오기

```
## 라이브러리 불러오기
import pandas as pd          # 판다스
import numpy as np           # numpy

import platform              # platform
import matplotlib.pyplot as plt      # matplotlib.pyplot (데이터 시각화를 위한 라이브러리)

%matplotlib inline
```

selenium

- 크롤링을 위한 selenium 설치

```
!pip install selenium
!apt-get update
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin

# -*- coding: UTF-8 -*-
import time
from selenium import webdriver

#Colab에선 웹브라우저 창이 뜨지 않으므로 별도 설정한다.

options = webdriver.ChromeOptions()
options.add_argument('--headless')        # Head-less 설정
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver', options=options)
```

데이터 크롤링

- 크롤링을 위한 selenium 설치

```
driver.get("http://info.nec.go.kr/main/showDocument.xhtml?electionId=0000000000&topMenuId=VC&secondMenuId=VCCP09") # 해당 링크 크롤링 시작
from selenium.webdriver.common.by import By
```

```
driver.find_element(By.ID, "electionType1").click() # 해당 링크로 접속한 페이지에서 '대통령선거'라는 ID 위치를 찾고 클릭
```

```
driver.find_element(By.ID, "electionName").send_keys("제19대") # 그리고 나타나는 선택항목에서 '제19대'를 선택
```

```
driver.find_element(By.ID, "electionCode").send_keys("대통령선거") # 그리고 나타나는 선택에서 '대통령선거'를 선택
```

데이터 크롤링

- 시도 항목을 가져와 리스트로 바꾸기

```
#이제 나타나는 시도 항목에서 선택 부분의 XPath를 찾고, 해당 리스트를 가져옵니다
sido_list_raw = driver.find_element(By. XPATH, """//*[@id="cityCode"]""")
sido_list = sido_list_raw.find_elements(By.TAG_NAME, "option")
sido_names_values = [option.text for option in sido_list]
sido_names_values = sido_names_values[2:]
sido_names_values
```

```
['서울특별시',
 '부산광역시',
 '대구광역시',
 '인천광역시',
 '광주광역시',
 '대전광역시',
 '울산광역시',
 '세종특별자치시',
 '경기도',
 '강원도',
 '충청북도',
 '충청남도',
 '전라북도',
 '전라남도',
 '경상북도',
 '경상남도',
 '제주특별자치도']
```

데이터 크롤링

- 득표수 데이터 수집 및 move_sido, append_data 함수 선언

```
## 득표수 데이터 수집
import re

def get_num(tmp):
    return float(re.split('₩(', str(tmp))[0].replace(',',''))
```

```
## 광역시도 이름을 리스트에 전송하고 검색 버튼을 누르는 move_sido 함수 선언
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

wait = WebDriverWait(driver, 10)

def move_sido(name):
    element = driver.find_element(By.ID,"cityCode")
    element.send_keys(name) ## 원소값으로 전달된 값을 key로 사용
    make_xpath = """//*[@id="searchBtn"]"""
    wait.until_not(EC.element_to_be_clickable((By.XPATH,make_xpath))) ## XPATH에서 클릭할 수 있는 element를 찾을 때까지 계속 검색
    driver.find_element(By.XPATH, make_xpath).click()
```

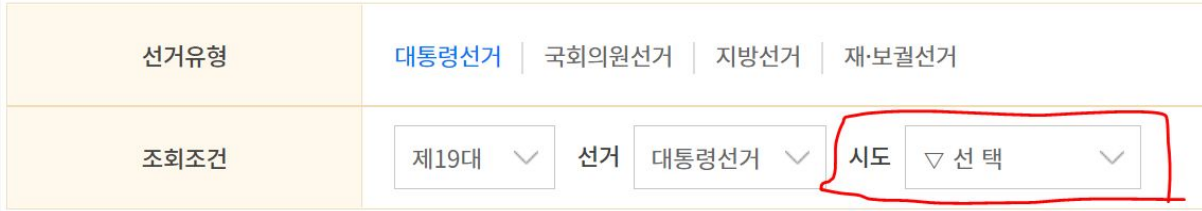
```
##  빈 내용으로 미리 준비한 DataFrame에 append 명령으로 읽은 데이터를 하나씩 추가하는 append_data 함수 선언
def append_data(df, sido_name, data):
    for each in df[0].values[1:]:
        data['광역시도'].append(sido_name)
        data['시군'].append(each[0])
        data['pop'].append(get_num(each[2]))
        data['moon'].append(get_num(each[3]))
        data['hong'].append(get_num(each[4]))
        data['ahn'].append(get_num(each[5]))
```

```
election_result_raw = {'광역시도' : [],
                       '시군' : [],
                       'pop' : [],
                       'moon' : [],
                       'hong' : [],
                       'ahn' : [] }
```

데이터 크롤링

- 데이터를 찾고 리스트에 추가 및 저장

```
## Beautiful Soup 라이브러리를 이용해 padnas의 read_html를 읽기
from bs4 import BeautifulSoup

## move_sido, append_data 함수를 통해 데이터를 찾고 추가
for each_sido in sido_names_values:
    move_sido(each_sido)

    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find('table')

    df = pd.read_html(str(table))

    append_data(df, each_sido, election_result_raw)
```



데이터 크롤링

- 리스트 출력 및 selenium 사용 종료

```
## 리스트 출력
election_result = pd.DataFrame(election_result_raw, columns=['광역시도', '시군', 'pop', 'moon','hong','ahn'])
election_result
```

	광역시도	시군	pop	moon	hong	ahn
0	서울특별시	종로구	102566.0	42512.0	22325.0	22313.0
1	서울특별시	중구	82852.0	34062.0	17901.0	19372.0
2	서울특별시	용산구	148157.0	58081.0	35230.0	32109.0
3	서울특별시	성동구	203175.0	86686.0	40566.0	45674.0
4	서울특별시	광진구	240030.0	105512.0	46368.0	52824.0
...	...	...	...	...	...	...
245	경상남도	산청군	24513.0	6561.0	12544.0	2753.0
246	경상남도	거창군	41325.0	11256.0	19976.0	4923.0
247	경상남도	합천군	33021.0	7143.0	19699.0	3077.0
248	제주특별자치도	제주시	273163.0	125717.0	48027.0	55971.0
249	제주특별자치도	서귀포시	101296.0	43776.0	20036.0	21890.0

250 rows × 6 columns

```
election_result.to_csv('../data/05. election_result.csv', encoding='utf-8', sep=',')
```

```
driver.close() # 현재 selenium webdriver가 활성화되어 있는 화면을 종료
```

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 데이터 불러오기 & 광역시도 이름 정리

```
election_result = pd.read_csv('../data/05. election_result.csv', encoding='utf-8',
                              index_col=0)
election_result.head()
```

	광역시도	시군	pop	moon	hong	ahn
0	서울특별시	종로구	102566.0	42512.0	22325.0	22313.0
1	서울특별시	중구	82852.0	34062.0	17901.0	19372.0
2	서울특별시	용산구	148157.0	58081.0	35230.0	32109.0
3	서울특별시	성동구	203175.0	86686.0	40566.0	45674.0
4	서울특별시	광진구	240030.0	105512.0	46368.0	52824.0
...	...	...	...	...	...	...
245	경상남도	산청군	24513.0	6561.0	12544.0	2753.0
246	경상남도	거창군	41325.0	11256.0	19976.0	4923.0
247	경상남도	합천군	33021.0	7143.0	19699.0	3077.0
248	제주특별자치도	제주시	273163.0	125717.0	48027.0	55971.0
249	제주특별자치도	서귀포시	101296.0	43776.0	20036.0	21890.0

250 rows × 6 columns

```
sido_candi =  election_result['광역시도']
sido_candi = [name[:2] if name[:2]
                              in ['서울','부산','대구','광주','인천','대전','울산']
                          else '' for name in sido_candi]
```

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 광역시군 ID 줄이기 위한 함수 (ex) 중랑구 -> 중랑) 정의 & 광역시군 ID 정리

```
def cut_char_sigu(name):
    return name if len(name)==2 else name[:-1]
```

```
import re

sigun_candi = ['']*len(election_result)

for n in election_result.index:
    each = election_result['시군'][n]
    if each[:2] in ['수원', '성남','안양','안산','고양',
                    '용인','청주','천안','전주','포항','창원']:
        sigun_candi[n] = re.split('시', each)[0]+' '+ ##
                                                cut_char_sigu(re.split('시', each)[1])
    else:
        sigun_candi[n] = cut_char_sigu(each)

sigun_candi
```

```
Out[5]: ['종로',
         '중구',
         '용산',
         '성동',
         '광진',
         '동대문',
         '중랑',
         '성북',
         '강북',
         '도봉',
         '노원',
         '은평',
         '서대문',
         '마포',
         '양천',
         '강서',
         '구로',
         '금천',
         '영등포',
```

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 행정구 ID 정리 (ex) 만안 -> 안양 만안)

In [6]:
```
ID_candi = [sido_candi[n]+' '+sigun_candi[n] for n in range(0,len(sigun_candi))]

ID_candi = [name[1:] if name[0]==' ' else name for name in ID_candi]
ID_candi = [name[:2] if name[:2]=='세종' else name for name in ID_candi]

ID_candi
```

Out[6]:
```
['서울 종로',
 '서울 중구',
 '서울 용산',
 '서울 성동',
 '서울 광진',
 '서울 동대문',
 '서울 중랑',
 '서울 성북',
 '서울 강북',
 '서울 도봉',
 '서울 노원',
 '서울 은평',
 '서울 서대문',
 '서울 마포',
 '서울 양천',
 '서울 강서',
 '서울 구로',
 '서울 금천',
 '서울 영등포',
```

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 앞서 지정한 ID_candi를 election_result에 합치기

```
In [7]: election_result['ID'] = ID_candi
        election_result.head(10)
```

Out[7]:

	광역시도	시군	pop	moon	hong	ahn	ID
0	서울특별시	종로구	102566.0	42512.0	22325.0	22313.0	서울 종로
1	서울특별시	중구	82852.0	34062.0	17901.0	19372.0	서울 중구
2	서울특별시	용산구	148157.0	58081.0	35230.0	32109.0	서울 용산
3	서울특별시	성동구	203175.0	86686.0	40566.0	45674.0	서울 성동
4	서울특별시	광진구	240030.0	105512.0	46368.0	52824.0	서울 광진
5	서울특별시	동대문구	236092.0	98958.0	51631.0	53359.0	서울 동대문
6	서울특별시	중랑구	265706.0	111450.0	56545.0	62778.0	서울 중랑
7	서울특별시	성북구	295866.0	129263.0	57584.0	66518.0	서울 성북
8	서울특별시	강북구	210614.0	89645.0	42268.0	51669.0	서울 강북
9	서울특별시	도봉구	229233.0	94898.0	47461.0	55600.0	서울 도봉

=> 지도를 그리기 위한 기초 작업

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 각 후보의 득표수에서 투표지수를 나눠 각각의 득표율 계산

In [8]:
```
election_result[['rate_moon','rate_hong','rate_ahn']] = ₩
            election_result[['moon','hong','ahn']].div(election_result['pop'],  axis=0)
election_result[['rate_moon','rate_hong','rate_ahn']] *= 100
election_result.head()
```

Out [8]:

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
0	서울특별시	종로구	102566.0	42512.0	22325.0	22313.0	서울 종로	41.448433	21.766472	21.754773
1	서울특별시	중구	82852.0	34062.0	17901.0	19372.0	서울 중구	41.111862	21.605996	23.381451
2	서울특별시	용산구	148157.0	58081.0	35230.0	32109.0	서울 용산	39.202333	23.778829	21.672280
3	서울특별시	성동구	203175.0	86686.0	40566.0	45674.0	서울 성동	42.665682	19.966039	22.480128
4	서울특별시	광진구	240030.0	105512.0	46368.0	52824.0	서울 광진	43.957839	19.317585	22.007249

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 각 후보 별 높은 비율로 득표한 지역 확인

```
In [9]: election_result.sort_values(['rate_moon'], ascending=[False]).head(10)
```

Out[9]:

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
182	전라남도	순천시	181451.0	122595.0	4525.0	40429.0	순천	67.563695	2.493786	22.280946
166	전라북도	전주시먹진구	187921.0	125375.0	5183.0	40188.0	전주 먹진	66.716865	2.758074	21.385582
165	전라북도	전주시완산구	236383.0	157637.0	7003.0	50506.0	전주 완산	66.687114	2.962565	21.366173
175	전라북도	장수군	16079.0	10714.0	717.0	3353.0	장수	66.633497	4.459233	20.853287
184	전라남도	광양시	96384.0	63544.0	4100.0	20080.0	광양	65.927955	4.253818	20.833333
173	전라북도	진안군	18107.0	11918.0	819.0	3904.0	진안	65.819849	4.523113	21.560722
172	전라북도	완주군	62470.0	41057.0	2107.0	13897.0	완주	65.722747	3.372819	22.245878
168	전라북도	익산시	192208.0	123422.0	6470.0	45737.0	익산	64.212728	3.366145	23.795576
170	전라북도	남원시	55371.0	35539.0	1939.0	13854.0	남원	64.183417	3.501833	25.020317
63	광주광역시	광산구	248209.0	159119.0	3630.0	65402.0	광주 광산	64.106862	1.462477	26.349568

```
In [10]: election_result.sort_values(['rate_hong'], ascending=[False]).head(10)
```

Out[10]:

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
219	경상북도	군위군	17627.0	2251.0	11651.0	1939.0	군위	12.770182	66.097464	11.000170
220	경상북도	의성군	37855.0	5365.0	23790.0	4767.0	의성	14.172500	62.845067	12.592788
223	경상북도	영덕군	26125.0	3786.0	16314.0	3231.0	영덕	14.491866	62.445933	12.367464
247	경상남도	합천군	33021.0	7143.0	19699.0	3077.0	합천	21.631689	59.655976	9.318313
216	경상북도	고령군	22396.0	3754.0	13248.0	2600.0	고령	16.761922	59.153420	11.609216
213	경상북도	예천군	32124.0	5261.0	18863.0	4427.0	예천	16.377163	58.719338	13.780974
215	경상북도	청도군	30398.0	5323.0	17678.0	3654.0	청도	17.511020	58.155142	12.020528
221	경상북도	청송군	18418.0	3218.0	10669.0	2387.0	청송	17.472038	57.927028	12.960148
240	경상남도	창녕군	42878.0	10310.0	24464.0	3877.0	창녕	24.044965	57.054900	9.041933
212	경상북도	문경시	49113.0	8616.0	27832.0	6905.0	문경	17.543217	56.669314	14.059414

```
In [11]: election_result.sort_values(['rate_ahn'], ascending=[False]).head(10)
```

Out[11]:

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
196	전라남도	진도군	21189.0	10392.0	511.0	8855.0	진도	49.044315	2.411629	41.790552
201	전라남도	신안군	28950.0	14370.0	713.0	12000.0	신안	49.637306	2.462867	41.450777
193	전라남도	강진군	25175.0	12476.0	753.0	10152.0	강진	49.557100	2.991063	40.325720
195	전라남도	해남군	48351.0	25901.0	1158.0	18157.0	해남	53.568696	2.394987	37.552481
197	전라남도	영암군	36402.0	18999.0	825.0	13610.0	영암	52.192187	2.266359	37.388056
180	전라남도	목포시	145476.0	77896.0	2584.0	53303.0	목포	53.545602	1.776238	36.640408
59	광주광역시	동구	66287.0	37053.0	1308.0	23438.0	광주 동구	55.897838	1.973238	35.358366
192	전라남도	장흥군	27149.0	14821.0	636.0	9593.0	장흥	54.591329	2.342628	35.334635
190	전라남도	보성군	29967.0	16666.0	732.0	10514.0	보성	55.614509	2.442687	35.085260
198	전라남도	무안군	52516.0	29516.0	983.0	18052.0	무안	56.203824	1.871810	34.374286

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 전국 지도 그리기 위한 데이터 파일 불러오기

```
draw_korea = pd.read_csv('../data/05. draw_korea.csv', encoding='utf-8',
                                                                    index_col=0)
draw_korea.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

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 지역별 좌표 정보를 가진 draw_korea 와 ID 시각화 대상인 election_result의 ID 일치 확인

  => 각 데이터를 하나의 집합으로 보고, 서로의 차집합을 구해 공집합인지 확인

```
set(draw_korea['ID'].unique()) - set(election_result['ID'].unique())
```

```
{'고성(강원)', '고성(경남)', '부천 소사', '부천 오정', '부천 원미', '창원 합포', '창원 회원'}
```

```
set(election_result['ID'].unique()) - set(draw_korea['ID'].unique())
```

```
{'고성', '부천', '창원 마산합포', '창원 마산회원'}
```

```
election_result[election_result['ID'] == '고성']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
125	강원도	고성군	18692.0	5664.0	6511.0	3964.0	고성	30.301733	34.833084	21.206933
233	경상남도	고성군	34603.0	9848.0	16797.0	4104.0	고성	28.459960	48.542034	11.860243

```
election_result.loc[125, 'ID'] = '고성(강원)'
election_result.loc[233, 'ID'] = '고성(경남)'

election_result[election_result['시군'] == '고성군']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
125	강원도	고성군	18692.0	5664.0	6511.0	3964.0	고성(강원)	30.301733	34.833084	21.206933
233	경상남도	고성군	34603.0	9848.0	16797.0	4104.0	고성(경남)	28.459960	48.542034	11.860243

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 나머지 ID도 재정의

```
election_result[election_result['광역시도'] == '경상남도']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ah
226	경상남도	창원시의창구	164047.0	60757.0	56887.0	22830.0	창원 의창	37.036337	34.677257	13.91674
227	경상남도	창원시성산구	153327.0	63717.0	42052.0	22923.0	창원 성산	41.556282	27.426350	14.95040
228	경상남도	창원시마산합포구	119281.0	35592.0	54488.0	14686.0	창원 마산합포	29.838784	45.680368	12.31210
229	경상남도	창원시마산회원구	136757.0	45014.0	56340.0	17744.0	창원 마산회원	32.915317	41.197160	12.97483
230	경상남도	창원시진해구	114779.0	41249.0	40049.0	17435.0	창원 진해	35.937759	34.892271	15.19006
231	경상남도	진주시	222813.0	73929.0	93751.0	26687.0	진주	33.179841	42.076091	11.97730
232	경상남									

```
election_result.loc[228, 'ID'] = '창원 합포'
election_result.loc[229, 'ID'] = '창원 회원'

election_result[election_result['광역시도'] == '경상남도']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
226	경상남도	창원시의창구	164047.0	60757.0	56887.0	22830.0	창원 의창	37.036337	34.677257	13.916743
227	경상남도	창원시성산구	153327.0	63717.0	42052.0	22923.0	창원 성산	41.556282	27.426350	14.950400
228	경상남도	창원시마산합포구	119281.0	35592.0	54488.0	14686.0	창원 합포	29.838784	45.680368	12.312103
229	경상남도	창원시마산회원구	136757.0	45014.0	56340.0	17744.0	창원 회원	32.915317	41.197160	12.974839
230	경상남도	창원시진해구	114779.0	41249.0	40049.0	17435.0	창원 진해	35.937759	34.892271	15.190061
231	경상남도	진주시	222813.0	73929.0	93751.0	26687.0	진주	33.179841	42.076091	11.977308
232	경상남도	통영시	82855.0	25477.0	36128.0	10738.0	통영	30.748899	43.603886	12.959990
233	경상남도	고성군	34603.0	9848.0	16797.0	4104.0	고성(경남)	28.459960	48.542034	11.860243
234	경상남도	사천시	71555.0	22370.0	32475.0	8350.0	사천	31.262665	45.384669	11.669345

```
set(draw_korea['ID'].unique()) - set(election_result['ID'].unique())
```

```
{'부천 소사', '부천 오정', '부천 원미'}
```

```
set(election_result['ID'].unique()) - set(draw_korea['ID'].unique())
```

```
{'부천'}
```

```
election_result[election_result['시군'] == '부천시']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn
85	경기도	부천시	543777.0	239697.0	100544.0	128297.0	부천	44.080018	18.489932	23.593679

```
ahn_tmp = election_result.loc[85, 'ahn']/3
hong_tmp = election_result.loc[85, 'hong']/3
moon_tmp = election_result.loc[85, 'moon']/3
pop_tmp = election_result.loc[85, 'pop']/3

rate_moon_tmp = election_result.loc[85, 'rate_moon']
rate_hong_tmp = election_result.loc[85, 'rate_hong']
rate_ahn_tmp = election_result.loc[85, 'rate_ahn']

election_result.loc[250] = [ahn_tmp, hong_tmp, moon_tmp, pop_tmp,
                            '경기도', '부천시', '부천 소사',
                            rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]
election_result.loc[251] = [ahn_tmp, hong_tmp, moon_tmp, pop_tmp,
                            '경기도', '부천시', '부천 오정',
                            rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]
election_result.loc[252] = [ahn_tmp, hong_tmp, moon_tmp, pop_tmp,
                            '경기도', '부천시', '부천 원미',
                            rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]
```

```
election_result.drop([85], inplace=True)
election_result[election_result['시군'] == '부천시']
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- 두 변수의 ID가 모두 일치!!

```
set(draw_korea['ID'].unique()) - set(election_result['ID'].unique())
```

```
set()
```

```
set(election_result['ID'].unique()) - set(draw_korea['ID'].unique())
```

```
set()
```

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

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn	y	x
0	서울특별시	종로구	102566.0	42512.0	22325	22313	서울 종로	41.448433	21.766472	21.754773	4	6
1	서울특별시	중구	82852.0	34062.0	17901	19372	서울 중구	41.111862	21.605996	23.381451	5	6
2	서울특별시	용산구	148157.0	58081.0	35230	32109	서울 용산	39.202333	23.778829	21.672280	6	6
3	서울특별시	성동구	203175.0	86686.0	40566	45674	서울 성동	42.665682	19.966039	22.480128	5	7
4	서울특별시	광진구	240030.0	105512.0	46368	52824	서울 광진	43.957839	19.317585	22.007249	6	7

데이터 가공 - 각 후보의 득표율과 지역 ID 정리하기

- election_rsult와 draw_korea 데이터 합치기 & 각 후부 사이의 득표율 차이 계산

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

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn	y	x
0	서울특별시	종로구	102566.0	42512.0	22325	22313	서울 종로	41.448433	21.766472	21.754773	4	6
1	서울특별시	중구	82852.0	34062.0	17901	19372	서울 중구	41.111862	21.605996	23.381451	5	6
2	서울특별시	용산구	148157.0	58081.0	35230	32109	서울 용산	39.202333	23.778829	21.672280	6	6
3	서울특별시	성동구	203175.0	86686.0	40566	45674	서울 성동	42.665682	19.966039	22.480128	5	7
4	서울특별시	광진구	240030.0	105512.0	46368	52824	서울 광진	43.957839	19.317585	22.007249	6	7

```
final_elect_data['moon_vs_hong'] = final_elect_data['rate_moon'] - \
                                   final_elect_data['rate_hong']
final_elect_data['moon_vs_ahn'] = final_elect_data['rate_moon'] - \
                                  final_elect_data['rate_ahn']
final_elect_data['ahn_vs_hong'] = final_elect_data['rate_ahn'] - \
                                  final_elect_data['rate_hong']
final_elect_data.head()
```

```
final_elect_data.sort_values(['moon_vs_hong'], ascending=[False]).head(10)
```

	광역시도	시군	pop	moon	hong	ahn	ID	rate_moon	rate_hong	rate_ahn	y	x	moon_vs_hong	moon_v
181	전라남도	순천시	181451.0	122595.0	4525	40429	순천	67.563695	2.493786	22.280946	22	4	65.069909	45.2
165	전라북도	전주시덕진구	187921.0	125375.0	5183	40188	전주덕진	66.716865	2.758074	21.385582	18	3	63.958791	45.3
	전라	전주시					전주							

시각화

- 지역별로 경계선 그리기

```
BORDER_LINES = [
    [(5, 1), (5,2), (7,2), (7,3), (11,3), (11,0)], # 인천
    [(5,4), (5,5), (2,5), (2,7), (4,7), (4,9), (7,9),
     (7,7), (9,7), (9,5), (10,5), (10,4), (5,4)], # 서울
    [(1,7), (1,8), (3,8), (3,10), (10,10), (10,7),
     (12,7), (12,6), (11,6), (11,5), (12, 5), (12,4),
     (11,4), (11,3)], # 경기도
    [(8,10), (8,11), (6,11), (6,12)], # 강원도
    [(12,5), (13,5), (13,4), (14,4), (14,5), (15,5),
     (15,4), (16,4), (16,2)], # 충청북도
    [(16,4), (17,4), (17,5), (16,5), (16,6), (19,6),
     (19,5), (20,5), (20,4), (21,4), (21,3), (19,3), (19,1)], # 전라북도
    [(13,5), (13,6), (16,6)], # 대전시
    [(13,5), (14,5)], #세종시
    [(21,2), (21,3), (22,3), (22,4), (24,4), (24,2), (21,2)], #광주
    [(20,5), (21,5), (21,6), (23,6)], #전라남도
    [(10,8), (12,8), (12,9), (14,9), (14,8), (16,8), (16,6)], #충청북도
    [(14,9), (14,11), (14,12), (13,12), (13,13)], #경상북도
    [(15,8), (17,8), (17,10), (16,10), (16,11), (14,11)], #대구
    [(17,9), (18,9), (18,8), (19,8), (19,9), (20,9), (20,10), (21,10)], #부산
    [(16,11), (16,13)], #울산
    [(27,5), (27,6), (25,6)],
]
```

시각화

- 5장에서 활용한 drawKorea함수를 이용해 지도만들기

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

    whitelabelmin = 20.

    datalabel = targetData

    tmp_max = max([ np.abs(min(blockedMap[targetData])),
                                np.abs(max(blockedMap[targetData]))])
    vmin, vmax = -tmp_max, tmp_max

    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 np.abs(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)
```

시각화

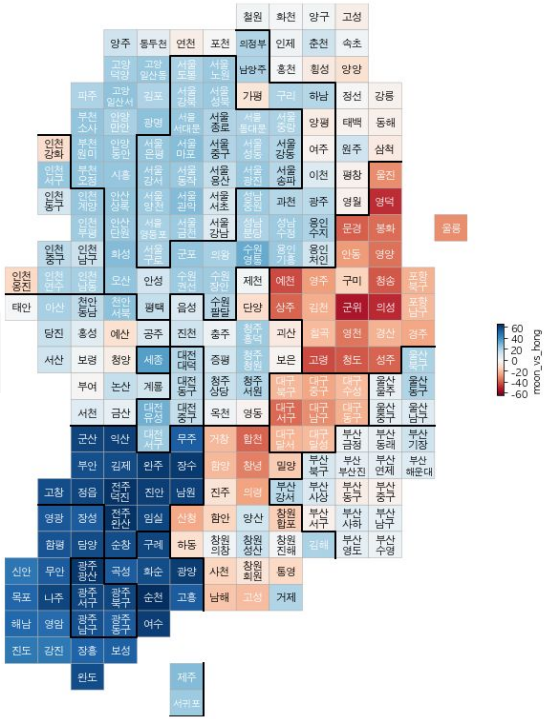
- 5장에서 활용한 drawKorea함수를 이용해 지도만들기

```
# 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()
```

시각화

- drawKorea함수를 이용해 나타내기

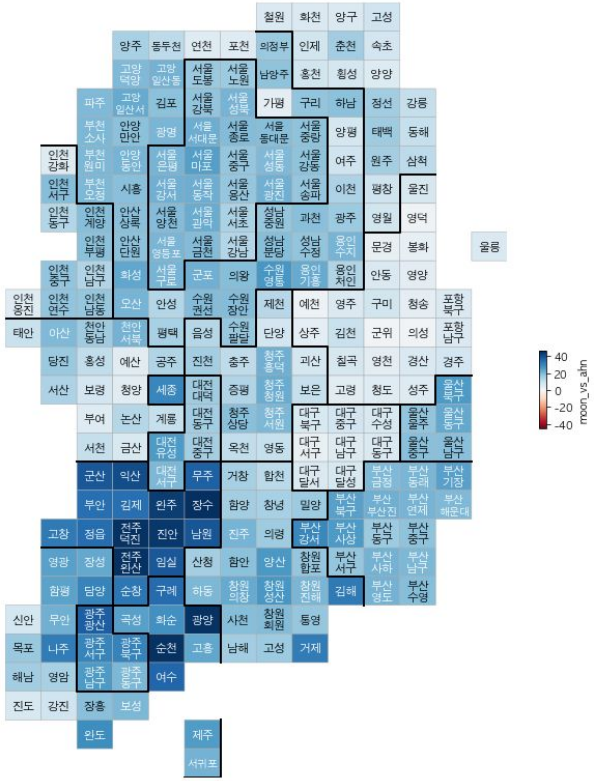
```
drawKorea('moon_vs_hong', final_elect_data, 'RdBu')
```



시각화

- drawKorea함수를 이용해 나타내기

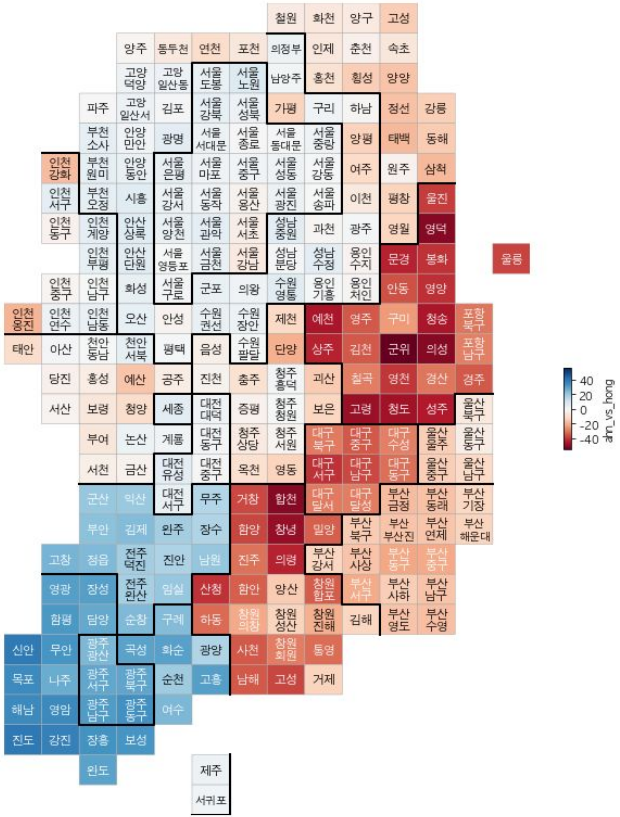
```
drawKorea('moon_vs_ahn', final_elect_data, 'RdBu')
```



시각화

- drawKorea함수를 이용해 나타내기

```
drawKorea('ahn_vs_hong', final_elect_data, 'RdBu')
```



시각화

- foilum을 import하기

```
import folium
import json
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
pop_folium = final_elect_data.set_index('ID')

del pop_folium['광역시도']
del pop_folium['시군']

pop_folium.head()
```

시각화

- 지도데이터를 불러와 output하기


```
geo_path = '../data/05. 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=6)
map.choropleth(geo_data = geo_str,
               data = pop_folium['moon_vs_hong'],
               columns = [pop_folium.index, pop_folium['moon_vs_hong']],
               fill_color = 'PuBu', #PuRd, YlGnBu
               key_on = 'feature.id')

map
```