

[2019-1 기초 빅데이터 프로그래밍]

기초빅데이터프로그래밍 기말 프로젝트

30 개국 인구, 지리, 경제 데이터 크롤링 및 시각화

World demographic, economic, geographic data
visualization

이름: 박보성 PARK BOSUNG

2019-6-23

프로젝트 기간: 2019.05.25~2019.06.13


학번: 20160796

I. 프로젝트 개요 및 목표

가) 프로젝트 개요

전세계 각종 데이터를 모아 시각화 하여 한 눈에 확인할 수 있는 크롤링 코드 및 시각화 코드를 제작한다. 총 30 개 국가를 예시적으로 보여주며 필요 시 나라이름만 추가하면 해당 나라 데이터를 확인할 수 있다. 데이터는 CIA World Factbook 의 업데이트 된 자료를 반영한 indexmundi.com 에서 가져온다.

[Table 1] Indexmundi.com webpage

	
United States Government Profile 2018	
Home > Factbook > Countries > United States	
Country name	conventional long form: United States of America conventional short form: United States abbreviation: US or USA etymology: the name America is derived from that of Amerigo VESPUCCI (1454-1512) - Italian explorer, navigator, and cartographer - using the Latin form of his name, Americus, feminized to America
Government type	constitutional federal republic
Capital	name: Washington, DC geographic coordinates: 38 53 N, 77 02 W time difference: UTC-5 (during Standard Time) daylight saving time: +1hr, begins second Sunday in March; ends first Sunday in November note: the 50 United States cover six time zones
Administrative divisions	50 states and 1 district*: Alabama, Alaska, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, District of Columbia*, Florida, Georgia, Hawaii, Idaho, Illinois, Indiana, Iowa, Kansas, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, Montana, Nebraska, Nevada, New Hampshire, New Jersey, New Mexico, New York, North Carolina, North Dakota, Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South Carolina, South Dakota, Tennessee, Texas, Utah, Vermont, Virginia, Washington, West Virginia, Wisconsin, Wyoming
Dependent areas	American Samoa, Baker Island, Guam, Howland Island, Jarvis Island, Johnston Atoll, Kingman Reef, Midway Islands, Navassa Island, Northern Mariana Islands, Palmyra Atoll, Puerto Rico, Virgin Islands, Wake Island note: from 18 July 1947 until 1 October 1994, the US administered the Trust Territory of the Pacific Islands; it entered into a political relationship with all four political entities: the Northern Mariana Islands is a commonwealth in political union with the US (effective 3 November 1986); the Republic of the Marshall Islands signed a Compact of Free Association with the US (effective 21 October 1986); the Federated States of Micronesia signed a Compact of Free Association with the US (effective 3 November 1986); Palau concluded a Compact of Free Association with the US (effective 1 October 1994)

Source: indexmundi.com

나) 프로젝트 목표

최근 다양한 글로벌 프로젝트들이 이루어지면서 각 국가의 다양한 데이터 수집 뿐 아니라 국가 간 비교를 쉽게 할 수 있는 자료도 많이 필요한 상황이다. 하지만 개별 국가의 숫자 데이터를 확인하기는 쉽지만 다양한 나라를 한눈에 시각화 하여 정리한 자료는 부족하다. 유동적으로 이용될 수 있는 코드를 만들어 다양하게 이용될 수 있게 할 것이다. 또한, 수업시간에 배운 각종 정규식, 시각화 툴, 객체지향 프로그래밍, 예외처리, decorator 등을 다방면으로 적용하기 위해 노력했다.

II. 개발 범위 및 내용

가) 개발범위

- 텍스트 크롤링: selenium, beautifulsoup(bs4), re, time 이용하여 크롤링하였다. 또한 csv 를 열고 해당 열, 행의 데이터만 가져오기 위해 csv, iloc, pandas 함수도 이용하였다. Re 정규식 컴파일을 통해서 웹상에서 필요한 데이터를 가져왔다.
- 시각화: 시각화를 위해서는 수업시간에 다뤘던 matplotlib 이외에 plotly, seaborn 함수를 이용하였다. 훨씬 더 정교하고 심미성이 있는 그래프를 제작할 수 있다. 전세계 지도에 해당 국가의 연령별 분포를 더 직관적으로 보여주기 위해서 choropleth 를 이용하였으며 마우스를 갖다 대면 해당 국가의 특정 정보를 표시해주기 위해 plotly.graph_objs 의 go 함수를 6 개 그래프 모두에서 이용하였다. 또한, 더 시각화 시키기 적당한 자료형태를 만들기 위해서 pandas 로 데이터를 읽어서 transpose 시키고, iloc 등을 이용해 데이터 변환을 시키기도 했다.

나) 개발내용

총 3 가지 카테고리에서 각각 2 개의 데이터를 가져왔으며 데이터 종류는 다음과 같다.

- ▶Demographic: Age structure, Median Age
- ▶Geographic: Land usage, Type of Land(water, land)
- ▶Economic: GDP Composition, Unemployment Rate

각 데이터를 가져와서 시각화 시키는 것은 다음과 같은 3 가지 단계로 이루어진다.

① URL Collection
<p>Input file: countrynamcode.csv</p> <p>URL:</p> <ol style="list-style-type: none"> 1. 'https://www.indexmundi.com/'+str(mtitle)+'demographics_profile.html' 2. 'https://www.indexmundi.com/'+str(mtitle)+'age_structure.html' 3. 'https://www.indexmundi.com/'+str(mtitle)+'unemployment_rate.html' 4. 'https://www.indexmundi.com/'+str(mtitle)+'economy_profile.html' 5. 'https://www.indexmundi.com/'+str(mtitle)+'geography_profile.html' <p>→ 150 개의 url 을 가져온다.</p>
② Data Web Crawling
<p>File Name: Country Data Crawling(population, geography, economy).ipynb</p> <pre> class population_crawling: def median_age() def age_structure() class economy_crawling: def Unemployment() def GDP_composition() Class geography_crawling: def area_comp() def land_use() Checktime(Decorator) WrongDataRecord(Exception) </pre>
③ Data Visualization
<p>Input File</p> <ol style="list-style-type: none"> 1. median_age.csv 2. age_structure.csv 3. unemployment.csv 4. GDP_composition.csv 5. area_comp.csv 6. land_use.csv <p>Dataframe</p> <ol style="list-style-type: none"> 1. Age_structure 2. Age_structure1 3. Land_use2 4. Area_comp2 5. GDP_comp2 6. unemployment2

크롤링 한 데이터를 가장 잘 표현할 수 있는 시각화를 했다는 것이 이 프로젝트의 가장 큰 특징이다. 6 개 데이터에 대해 총 7 개의 그래프를 제작하였다.

Median_age.csv file _ pandas dataframe

	country name	total	male	female
0	japan	47.3	46.0	48.7
1	india	27.9	27.2	28.6
2	china	37.4	36.5	38.4
3	thailand	37.7	36.6	38.7
4	vietnam	30.5	29.4	31.7
5	cambodia	25.3	24.6	26.0
6	singapore	34.6	34.5	34.7
7	indonesia	30.2	29.6	30.8
8	canada	42.2	40.9	43.5
9	united_states	38.1	36.8	39.4
10	united_kingdom	40.5	39.3	41.7
11	germany	47.1	46.0	48.2

Age_structure.csv _ pandas

	country name	0-14	15-24	25-54	54-64	65+
0	japan	12.84	9.64	37.50	12.15	27.87
1	india	27.34	17.90	41.08	7.45	6.24
2	china	17.15	12.78	48.51	10.75	10.81
3	thailand	16.93	14.17	46.32	12.00	10.58
4	vietnam	23.55	16.23	45.56	8.55	6.12
5	cambodia	31.01	18.36	40.68	5.69	4.25
6	singapore	12.82	16.56	50.53	10.46	9.63
7	indonesia	25.02	16.99	42.40	8.58	7.01
8	canada	15.44	11.85	39.99	14.10	18.63
9	united_states	18.73	13.27	39.45	12.91	15.63
10	united_kingdom	17.53	11.90	40.55	11.98	18.04
11	germany	12.82	10.09	40.45	14.58	22.06
12	france	18.53	11.79	37.78	12.42	19.48

인구 관련 데이터는 모두 비율로 표시된다. Median age 는 각 국가별로 bar graph 로 비교할 것이고 age_structure 데이터는 고령화 사회를 판단하는 데에 있어서 가장 유의미하게 작용하는 65 세 이상 인구 데이터를 bar_graph 로 표시할 것이다. 또한 경제활동인구(25-54), 고령인구(65+), 청년인구(15-24)에 각각 해당하는 비율을 Choropleth Plo 을 이용해 전세계 지도에 그려서 색깔로 비교하면 한눈에 파악할 수 있을 것이다.

Area_use .csv_ pandas

	country name	agriculture_land	forest	other	total
0	japan	12.5	68.5	19	100
1	india	60.5	23.1	16.4	100
2	china	54.7	22.3	23	100
3	thailand	41.2	37.2	21.6	100
4	vietnam	34.8	45.0	20.2	100
5	cambodia	32.1	56.5	11.4	100
6	singapore	1.0	3.3	95.7	100
7	indonesia	31.2	51.7	17.1	100
8	canada	6.8	34.1	59.1	100
9	united_states	44.5	33.3	22.2	100
10	united_kingdom	71.0	11.9	17.1	100
11	germany	48.0	31.8	20.2	100
12	france	52.7	29.2	18.1	100
13	italy	47.1	31.4	21.5	100

Land_comp.csv_pandas

	country name	total	land	water
0	japan	377,915	364,485	13,430
1	india	3,287,263	2,973,193	314,070
2	china	9,596,960	9,326,410	270,550
3	thailand	513,120	510,890	2,230
4	vietnam	331,210	310,070	21,140
5	cambodia	181,035	176,515	4,520
6	singapore	719.2	709.2	10
7	indonesia	1,904,569	1,811,569	93,000
8	canada	9,984,670	9,093,507	891,163
9	united_states	9,833,517	9,147,593	685,924
10	united_kingdom	243,610	241,930	1,680
11	germany	357,022	348,672	8,350
12	france	643,801	640,427	3,374
13	italy	301,340	294,140	7,200

실업률 데이터는 line graph 로 그려 그 변화율을 각 국가별로 비교하기 위해서 time series 로 수집하였다. 모든 국가에서 실업률 데이터를 제공해주지는 않기 때문에 해당하는 년도에 데이터가 모두 있는 국가에 대해서만 크롤링을 했다.

Unemployment.csv _pandas dataframe

	country name	2006	2007	2008	2009	2010	2011	2012	2013	2014
0	spain	8.4	8.2	11.0	17.0	19.0	21.0	24.0	26.	24.0
1	philippines	7.9	7.3	7.4	7.4	7.3	7.0	6.9	7.0	6.8
2	south_korea	3.4	3.2	3.1	3.6	3.7	3.4	3.2	3.1	3.5
3	mexico	3.5	3.6	3.8	5.3	5.2	5.1	4.8	4.9	4.7
4	austria	4.7	4.4	3.8	4.7	4.4	4.1	4.3	4.9	5.0
5	poland	13.0	9.6	7.1	8.1	9.6	9.6	10.0	10.	8.9
6	portugal	7.6	7.9	7.5	9.4	10.0	12.0	15.0	16.	13.0
7	egypt	10.0	9.2	8.6	9.3	9.2	10.0	12.0	13<	13.0
8	denmark	3.9	3.7	3.4	5.9	7.4	7.5	7.5	7.0	6.5
9	australia	4.7	4.3	4.2	5.5	5.2	5.0	5.2	5.6	6.0

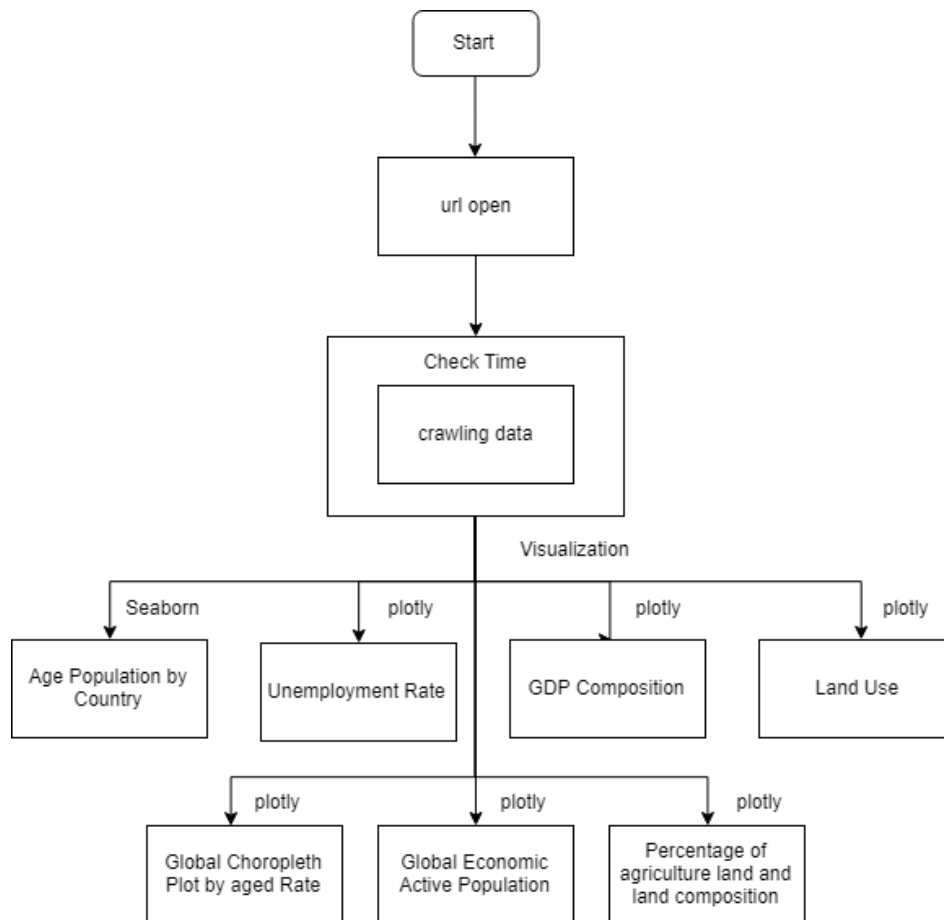
GDP 비율은 6 가지 항목(household consumption, government consumption, investment in fixed capital, investment in inventories, exports of goods and services, imports of goods and services) 으로 이루어지며 이 항목 비율을 모두 더하면 100%가 된다. 한눈에 파악하기 위해서는 각 국가별로 pie chart 를 그리면 적당할 것이라고 판단했다.

GDP_composition.csv _pandas dataframe

	country name	household consumption	government consumption	investment in fixed capital	investment in inventories	exports of S&G	imports of S&G
0	japan	55.9	19.5	23.5	0.2	17.8	-16.8
1	india	58.7	11.6	27.5	4.0	18.4	-20.2
2	china	39.1	14.6	43.3	1.1	19.6	-17.7
3	thailand	50.1	17.0	24.2	-7.0	70.4	-54.7
4	vietnam	68.5	6.6	24.8	2.9	98.6	-101.4
5	cambodia	76.4	5.4	22.0	0.9	62.8	-67.4
6	singapore	34.7	11.4	23.5	1.9	179.2	-150.6
7	indonesia	57.5	8.9	32.1	0.7	19.2	-18.4
8	canada	58.1	20.9	22.8	0.3	31.4	-33.6
9	united_states	69.1	17.2	16.3	0.3	12.2	-15.1
10	united_kingdom	65.3	19.0	16.6	0.7	30.1	-31.7
11	germany	53.7	19.9	20.1	-1.0	47.3	-40.0
12	france	54.8	23.5	22.0	1.3	30.3	-32.0
13	italy	60.2	18.7	17.2	0.1	31.8	-28.0

III. 개발 과정

본 프로젝트에 대한 flow chart 는 다음과 같다.



IV. 개발 결과

가) Crawling Code

CountryNameCode.csv

```

1 malaysia
2 japan
3 india
4 china
5 thailand
6 vietnam
7 cambodia
8 singapore
9 indonesia
10 canada
11 united_states
12 united_kingdom
13 germany
14 france
15 italy
16 spain
17 philippines
18 south_korea
19 mexico
20 australia
21 poland
22 portugal
23 egypt
24 denmark
25 australia
26 cambodia
27 belgium
28 sudan
29 russia
30 norway

```

```

class population_crawling:
    def age_structure():
        with open('age_structure.csv', 'w', encoding='UTF-8', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(['country name', '0-14', '15-24', '25-54', '54-64', '65+'])

        pf = pandas.read_csv('CountryNameCode.csv', engine='python')

        for i in range(30):
            mtitle = pf[pf.columns[0]].iloc[i]

            url = 'https://www.indexmundi.com/' + str(mtitle) + '/age_structure.html'

            print(url)

            Age0_14 = re.compile(r'.0-14 years: </strong>(.*?)<br><strong>15.*')
            Age15_24 = re.compile(r'.15-24 years: </strong>(.*?)<br><strong>25.*')
            Age25_54 = re.compile(r'.25-54 years: </strong>(.*?)<br><strong>55.*')
            Age54_64 = re.compile(r'.55-64 years: </strong>(.*?)<br><strong>65.*')
            Age65 = re.compile(r'.65 years and over: </strong>(.*?)<br>.*')

            driver = webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
            driver.get(url)
            soup = BeautifulSoup(driver.page_source, "html.parser")

            link = soup.find_all("div", {"class": "c"})
            print(link)

```

CountryNameCode.csv 에 크롤링하고자 하는 국가들의 이름을 모두 저장한다. 해당 url 의 구조가 http://... + (country name) + /data type.html 이기 때문에 country name 자리에 csv 의 국가이름을 하나씩 가져와서 바꿔 넣어주면 변경된 해당 국가의 url 에 접속 가능하다. 또한 웹의 개발자도구에서 확인할 수 있는 코드를 참고하여 정규식 표현을 작성하여 해당 데이터만 가져온다. (.*?)을 이용하여 group 으로 가져오는 것이 가장 적절하다고 판단했다.

Regular expression으로 median age 가져오기

```
print(url)
total = re.compile(r'.<strong>total: </strong>(.*?)<br/><strong>male.*')
male=re.compile(r'.<strong>male: </strong>(.*?)<br/><strong>female.*')
female=re.compile(r'.<strong>female: </strong>(.*?)<br/>.*')

driver=webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
driver.get(url)
soup = BeautifulSoup(driver.page_source, "html.parser")

link = soup.find_all("div", {"class":"col-md-8 c"})
print("=====")
```

Time series 로 가져오기 위해서 2006-2014 사이의 실업률 데이터를 정규식을 이용해 가져왔다. 이후 각 데이터도 모두 정규식을 이용하여 적절하게 데이터를 추출하였다.

Regular Expression으로 unemployment 2006-2014 가져오기

```
print(url)
unemp_2006=re.compile(r'.2006</td><td>(.*?)</td><td>.*')
unemp_2007=re.compile(r'.2007</td><td>(.*?)</td><td>.*')
unemp_2008=re.compile(r'.2008</td><td>(.*?)</td><td>.*')
unemp_2009=re.compile(r'.2009</td><td>(.*?)</td><td>.*')
unemp_2010=re.compile(r'.2010</td><td>(.*?)</td><td>.*')
unemp_2011=re.compile(r'.2011</td><td>(.*?)</td><td>.*')
unemp_2012=re.compile(r'.2012</td><td>(.*?)</td><td>.*')
unemp_2013=re.compile(r'.2013</td><td>(.*?)</td><td>.*')
unemp_2014=re.compile(r'.2014</td><td>(.*?)</td><td>.*')

driver=webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
driver.get(url)
soup = BeautifulSoup(driver.page_source, "html.parser")

link = soup.find_all("table", {"class":"tblrank"})
```

Regular Expression으로 6개 항목에 대한 GDP 이용 데이터 가져오기

```
print(url)
HS_consumption = re.compile(r'.household consumption: </strong>(.*?)<br/><strong>government.*')
Gov_consumption=re.compile(r'.government consumption: </strong>(.*?)<br/><strong>investment in fixed capital.*')
inv_fixedcapital=re.compile(r'.investment in fixed capital: </strong>(.*?)<br/><strong>investment in inventories.*')
inv_inventories=re.compile(r'.inventories: </strong>(.*?)<br/><strong>exports of goods and services.*')
exports = re.compile(r'.exports of goods and services: </strong>(.*?)<br/><strong>imports.*')
imports = re.compile(r'.imports of goods and services: </strong>(.*?)<br/>.*')

driver=webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
driver.get(url)
soup = BeautifulSoup(driver.page_source, "html.parser")

link = soup.find_all("div", {"class":"col-md-8 c"})
```

Regular Expression으로 land, water 비중 가져오기 (*total, water, land*)

```
print(url)
total = re.compile(r'.total: </strong>(.*?)<br/><strong>land.*')
land=re.compile(r'.land: </strong>(.*?)<br/><strong>water.*')
water=re.compile(r'.water: </strong>(.*?)<br/>.*')

driver=webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
driver.get(url)
soup = BeautifulSoup(driver.page_source, "html.parser")

link = soup.find_all("div", {"class":"col-md-8 c"})
```

Regular Expression으로 land_use data 가져오기 (*agriculture, forest, other*)

```
agri=re.compile(r'.agricultural land: </strong>(.*?)<br/>arable.*')
forest=re.compile(r'.forest: </strong>(.*?)<br/><strong>other.*')
other=re.compile(r'.other: </strong>(.*?)<br/></td></tr>.*')

driver=webdriver.Chrome("C:/Users/ATIV/Downloads/chromedriver_win32/chromedriver")
driver.get(url)
soup = BeautifulSoup(driver.page_source, "html.parser")

link = soup.find_all("div", {"class":"col-md-8 c"})
print("=====")
```

숫자를 이용해 visualize 를 해야 하기 때문에 숫자 부분만 가져와야 한다. 데이터를 긁은 후 정제하기 보다는 긁기 전부터 특수기호, 앞뒤로 붙은 단어들은 제거를 한 후 가져왔다.

Crawling 에서 필요한 부분의 데이터 가공하여 가져오기 (*%, \$ 제거*)


```

for posts in link:
    time.sleep(0.33)
    z=str(posts)
    zz=str(posts).split(' ')
    str_index=0
    count = 0
    for aaa in zz:
        #index = 0
        str_index+=len(aaa)
        if word in aaa:
            count+=1
            if(count == num):

                index=aaa.find(word[len(word)-1])
                print(z[str_index-len(aaa)+len(word):])
                print(z[str_index:])
                break
        str_index+=1

a=Age0_14. search(z). group(1). strip()
b=Age15_24. search(z). group(1). strip()
c=Age25_54. search(z). group(1). strip()
d=Age54_64. search(z). group(1). strip()
e=Age65. search(z). group(1). strip()
Field1=mtitle
Field3=a. split() [0][:len(a. split() [0])-1]
Field4=b. split() [0][:len(b. split() [0])-1]
Field5=c. split() [0][:len(c. split() [0])-1]
Field6=d. split() [0][:len(d. split() [0])-1]
Field7=e. split() [0][:len(e. split() [0])-1]

writer.writerow([Field1,Field3,Field4,Field5, Field6, Field7])

```

정해진 항목별로 데이터 정리 예시 _ age_structure.csv

	A	B	C	D	E	F
1	country name	0-14	15-24	25-54	54-64	65+
2	japan	12.84	9.64	37.5	12.15	27.87
3	india	27.34	17.9	41.08	7.45	6.24
4	china	17.15	12.78	48.51	10.75	10.81
5	thailand	16.93	14.17	46.32	12	10.58
6	vietnam	23.55	16.23	45.56	8.55	6.12
7	cambodia	31.01	18.36	40.68	5.69	4.25
8	singapore	12.82	16.56	50.53	10.46	9.63
9	indonesia	25.02	16.99	42.4	8.58	7.01
10	canada	15.44	11.85	39.99	14.1	18.63
11	united_states	18.73	13.27	39.45	12.91	15.63
12	united_kingdom	17.53	11.9	40.55	11.98	18.04
13	germany	12.82	10.09	40.45	14.58	22.06
14	france	18.53	11.79	37.78	12.42	19.48
15	italy	13.65	9.66	42.16	12.99	21.53
16	spain	15.38	9.58	44.91	12.14	17.98
17	philippines	33.39	19.16	36.99	5.97	4.49
18	south_korea	13.21	12.66	45.52	14.49	14.12
19	mexico	26.93	17.54	40.81	7.64	7.09
20	austria	14.01	11.07	42.42	13.23	19.26
21	poland	14.76	10.7	43.48	14.21	16.86
22	portugal	15.34	11.36	41.72	12.18	19.4
23	egypt	33.29	18.94	37.6	5.95	4.22
24	denmark	16.41	13.08	38.76	12.52	19.23
25	australia	17.8	12.79	41.45	11.83	16.14
26	cambodia	31.01	18.36	40.68	5.69	4.25
27	belgium	17.16	11.34	40.05	12.86	18.58
28	sudan	38.68	21.04	32.77	4.24	3.27
29	ruissia	17.12	9.46	44.71	14.44	14.28
30	norway	18	12.58	41.01	11.71	16.71
31	saudi_arabia	26.1	18.57	46.86	5.03	3.44

Pandas 로 데이터를 쉽게 정리하기
 위해서 처음부터 각 항목별로
 정리하여 csv 파일을 생성하였다.
 6 개의 csv 파일 모두 Pandas 로
 불러서 바로 시각화 하기 쉬운
 형태이다. Index(항목이름) 도 따로
 설정해주기 보다 바로 설정하여
 이용할 수 있게 했다.

Get_population() 함수에서 age structure, median_age 데이터를 크롤링하게 되며
 get_unemp() 함수에서 실업률 데이터를 따로 저장한다. 이때 자주 발생하는 attribute
 error 와 index error 에 대해서 예외처리를 해주었다. Get_GDP(), get_GEO() 함수에서는

각각 GDP composition, 지리 데이터를 긁어오게 되며 예외가 발생했을 때 “에러가 발생했습니다. 크롤링을 다시 시작해주세요” 메시지가 뜨게 했다. Checktime decorator 는 각각의 크롤링 함수에 적용하였으며 총 크롤링 시간을 초단위로 계산하게 했다.

예외처리와 실행시간

```

@CheckTime
def get_population():
    population_crawling.age_structure()
    population_crawling.median_age()

@CheckTime
def get_unemp():
    try:
        economy_crawling.Unemployment()
    except AttributeError:
        pass

    except IndexError:
        raise WrongDataRecord("CountryNameCode.csv 파일의 나라 개수를 다시 확인해주세요")

    except:
        pass

@CheckTime
def get_GDP():
    try:
        economy_crawling.GDP_composition()
    except:
        raise WrongDataRecord("에러가 발생했습니다. 크롤링을 다시 시작해주세요")

@CheckTime
def get_GEO():
    try:
        geography_crawling.area_comp()
        geography_crawling.land_use()
    except:
        raise WrongDataRecord("에러가 발생했습니다. 크롤링을 다시 시작해주세요")

```

```

get_GEO()
https://www.indexmundi.com/egypt/geography_profile.html
=====
https://www.indexmundi.com/denmark/geography_profile.html
=====
https://www.indexmundi.com/australia/geography_profile.html
=====
https://www.indexmundi.com/cambodia/geography_profile.html
=====
https://www.indexmundi.com/belgium/geography_profile.html
=====
https://www.indexmundi.com/sudan/geography_profile.html
=====
https://www.indexmundi.com/russia/geography_profile.html
=====
https://www.indexmundi.com/norway/geography_profile.html
=====
https://www.indexmundi.com/saudi_arabia/geography_profile.html
=====
실행시간 : 1101.619479894638

```

나) Visualization _Seaborn

- Demographic_ Age Structure, Median_age Visualization

Seaborn안에서 Palette 이용하여 graph color 다양하게 설정

```

import matplotlib.pyplot as plt
# 주피터에 포함시키기
%matplotlib inline
import seaborn as sns

pkmn_type_colors = [
    '#78C850', # japan
    '#F08030', # india
    '#6890F0', # china
    '#A88820', # thailand
    '#A8A878', # vietnam
    '#A040A0', # cambodia
    '#F08030', # singapore
    '#E0C068', # indonesia
    '#E090AC', # canada
    '#C03028', # united_states
    '#F08030', # united_kingdom
    '#F08030', # germany
    '#705090', # france
    '#908080', # spain
    '#7030F0', # philippines
    '#50C070', # south_korea
    '#10C0A0', # mexico
    '#6040A0', # poland
    '#502060', # portugal
    '#60A070', # egypt
    '#50C070', # denmark
    '#00C0FF', # australia
    '#50C070', # cambodia
    '#E0C0A0', # belgium
    '#50C070', # sudan
    '#E050A0', # russia
    '#D00080', # norway
    '#E05050', # saudi_arabia

```

```

sns.set(style="ticks")
plt.xticks(rotation=-40)
age_structure1 = age_structure.sort_values(by='65+', ascending = False)
age_graph1 = sns.barplot(x='country_name', y='65+', data=age_structure1, palette=pkmn_type_colors)
age_graph1.figure.set_size_inches(20, 9)
age_graph1.axes.set_title('Aged Population by Country', fontsize=34, color="b", alpha=0.5)
age_graph1.set_xlabel("Country Name", size = 20, color="r", alpha=0.5)
age_graph1.set_ylabel("Population % over 65", size = 20, color="g", alpha=0.5)
age_graph1.tick_params(labelsize=14, labelcolor="black")

```

▼ 각 국가마다 팔레트 안에 색 지정

Colour picker ▼ google color picker

Seaborn에서는 matplotlib 보다 훨씬 다양한 색상을 이용할 수 있다. Palette 기능을 이용하여 각 항목(나라)에 다른 색상을 설정할 수 있는데 이 때 이용하게 되는 색상 코드는 구글에서 color picker 를 검색하면 쉽게 찾을 수 있다. 30 개 국가에 대해 각각 다른 색상을 설정해서 이용했다.

Seaborn 모듈과 matplotlib 의 GridSpec 을 이용하여 sorted 된 Age Structure 데이터에 대한 bar graph 를 만들었으며 해당 코드는 다음과 같다.

Age Structure Bar Graph

```
import seaborn as sns
from matplotlib.gridspec import GridSpec

sns.set(style="whitegrid")
#sns.set_color_codes("Spectral")

plt.figure(2, figsize=(33,24))
the_grid = GridSpec(2, 2)

plt.subplot(the_grid[0, 1], title='Population % over 65+')
sns.barplot(x='country name', y='65+', data=age_structure1, palette=pkmn_type_colors)
plt.xticks(rotation=-50)
plt.subplot(the_grid[0, 0], title='%Age 25-54')
sns.barplot(x='country name', y='25-54', data=age_structure1, palette=pkmn_type_colors)
plt.xticks(rotation=-50)
plt.subplot(the_grid[1, 1], title='%Age 15-24')
sns.barplot(x='country name', y='15-24', data=age_structure1, palette=pkmn_type_colors)
plt.xticks(rotation=-50)
plt.subplot(the_grid[1, 0], title='%Age 54-64')
sns.barplot(x='country name', y='54-64', data=age_structure1, palette=pkmn_type_colors)
plt.xticks(rotation=-50)

plt.suptitle('Age Structure', fontsize=30, color="b", alpha=0.5)
age_graph1.tick_params(labelsize=9, labelcolor="blue")
```

Medan Age 는 total, male, female 3 가지 데이터가 제공되기 때문에 이를 subplot 으로 한눈에 비교할 수 있게 하였다. Subplot 을 이용할 때 matplotlib 을 이용하게 되며 기본적인 그래프를 그릴 때는 seaborn 모듈을 이용한다. 글씨체, 색상 등에 대해서는 도전적으로 다양하게 설정해보았다.

Median Age_Subplots

```
import seaborn as sns
from matplotlib.gridspec import GridSpec

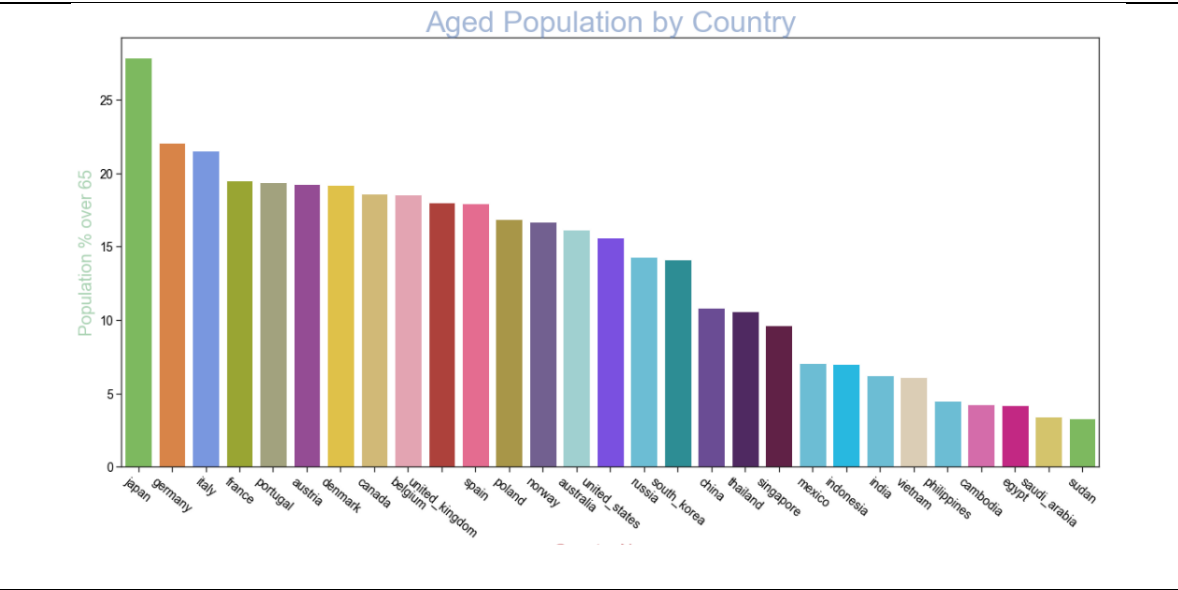
sns.set(style="whitegrid")
#sns.set_color_codes("Spectral")

plt.figure(3, figsize=(20,15))
the_grid = GridSpec(3, 3)
median_age1 = median_age.sort_values(by='total', ascending = False)

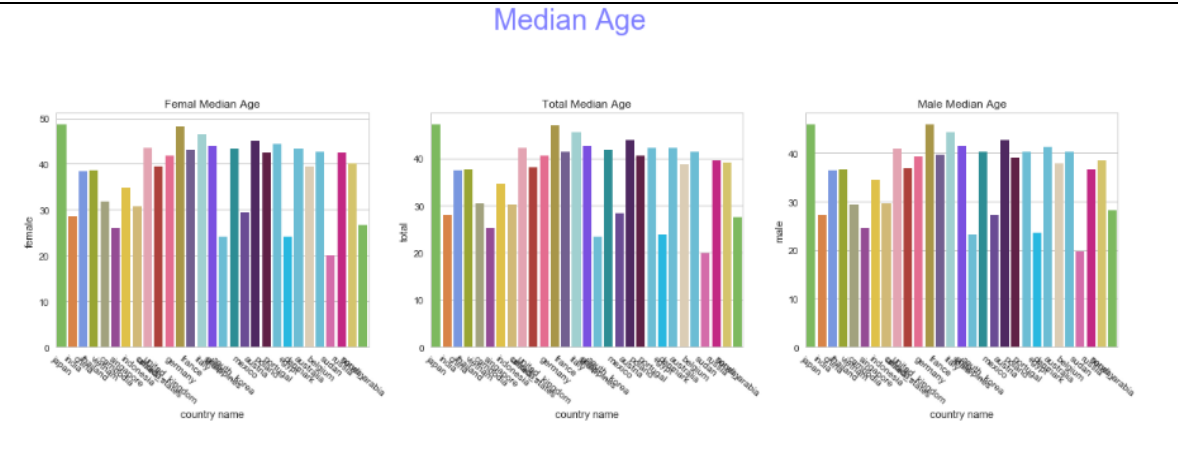
plt.subplot(the_grid[0, 1], title='Total Median Age')
sns.barplot(x='country name', y='total', data=median_age1, palette=pkmn_type_colors)
plt.xticks(rotation=-40)
plt.subplot(the_grid[0, 0], title='Femal Median Age')
sns.barplot(x='country name', y='female', data=median_age1, palette=pkmn_type_colors)
plt.xticks(rotation=-40)
plt.subplot(the_grid[0, 2], title='Male Median Age')
sns.barplot(x='country name', y='male', data=median_age1, palette=pkmn_type_colors)
plt.xticks(rotation=-40)

plt.suptitle('Median Age', fontsize=30, color="b", alpha=0.5)
```

Aged Population by Country (population % over 65_sorted data)

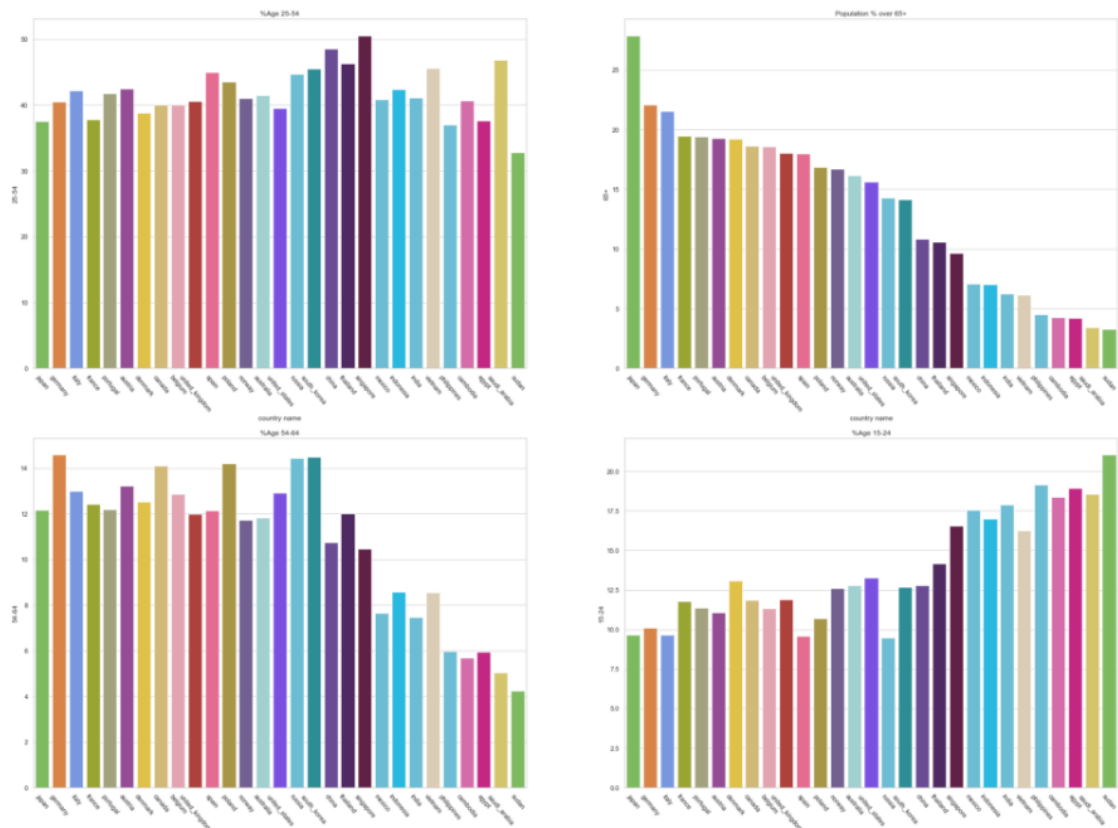


Median Age by Country



Subplots of Age Structure Data (25-54%, 65+, 54-64%, 15-24%)

Age Structure



다) Visualization_Plotly

- Economic Data_ Unemployment Rate Visualization

Plotly 모듈을 이용하여 더 다양한 그래프를 그렸다. Plotly는 해당 사이트에 가입하여 다음 그림처럼 계정을 연동시켜야 이용할 수 있다. Jupyter 상에서 그린 그래프도 홈페이지에 들어가면 모두 저장되어 있으며 수정도 할 수 있다. Line graph로 표현하기 편하게 하기 위해서 unemployment 데이터를 transpose 시킨 상태로 이용하였으며 2행부터 저장하는 등의 작업을 진행했다.

각각의 trace에 각 국가의 데이터를 저장하였으며 색상 등의 설정도 다 정했다. 마우스를 갖다 댔을 때 해당 국가들의 데이터를 바로 확인할 수 있는 legend를 띄우기 위해서 plotly의 go 기능을 이용했다.

Plotly 계정 연결, unemployment 데이터 처리, scattered line graph

```
import plotly
plotly.tools.set_credentials_file(username='BOSUNG', api_key='pdpCFjaV9BMmS3s8Y5fz')
```

◀plotly 계정 연결

```
import csv
import pandas as pd

import plotly.plotly as py
import plotly.graph_objs as go
```

```
unemployment = pd.read_csv('unemployment.csv')
unemployment1 = unemployment.T
unemployment1.iloc[:, 3]
year = ['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014']
```

```
unemployment1 = unemployment.T
unemployment1.iloc[:, 3]
country = unemployment1.iloc[:, 0]
unemployment2 = unemployment1.iloc[:, 1:]
unemployment2
spain = unemployment2.iloc[:, 0]
philippines = unemployment2.iloc[:, 1]
south_korea = unemployment2.iloc[:, 2]
mexico = unemployment2.iloc[:, 3]
austria = unemployment2.iloc[:, 4]
poland = unemployment2.iloc[:, 5]
portugal = unemployment2.iloc[:, 6]
egypt = unemployment2.iloc[:, 7]
denmark = unemployment2.iloc[:, 8]
australia = unemployment2.iloc[:, 9]
unemployment
```

◀데이터 처리

```
trace6 = go.Scatter(
    x = year,
    y = portugal,
    name = 'portugal',
    line = dict(
        color = ('rgb(200, 800, 400)'),
        width = 4)
)

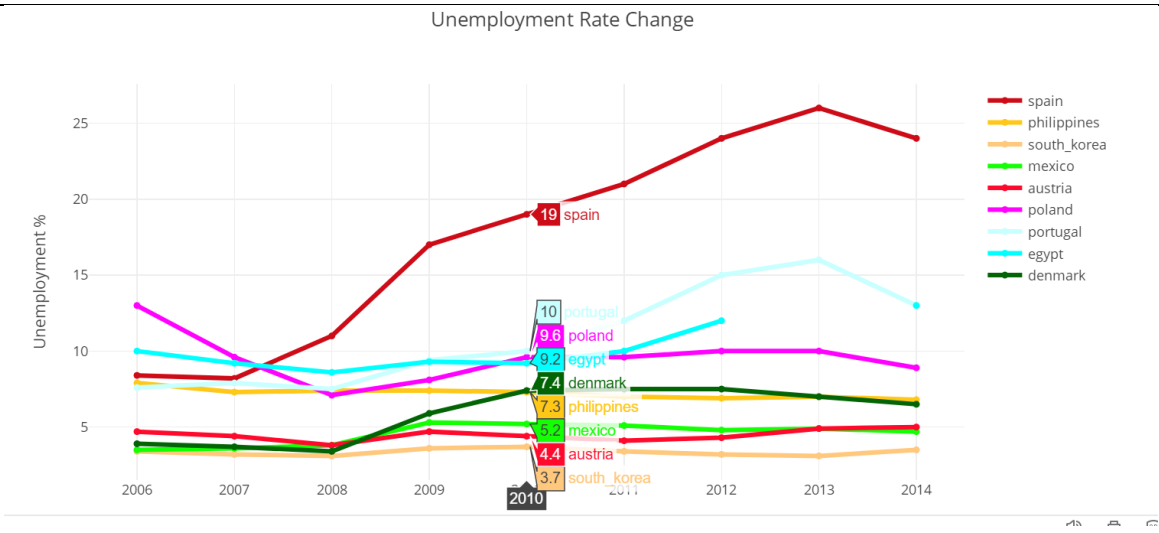
trace7 = go.Scatter(
    x = year,
    y = egypt,
    name = 'egypt',
    line = dict(
        color = ('rgb(2, 600, 400)'),
        width = 4)
)

trace8 = go.Scatter(
    x = year,
    y = denmark,
    name = 'denmark',
    line = dict(
        color = ('rgb(2, 100, 4)'),
        width = 4)
)

data = [trace0, trace1, trace2, trace3, trace4, trace5, trace6, trace7, trace8]
layout = dict(title = 'Unemployment Rate Change',
    xaxis = dict(title = 'Year'),
    yaxis = dict(title = 'Unemployment %'),
)

fig = dict(data=data, layout=layout)
py.iplot(fig, filename='legend-names')
```

Unemployment Rate Change Graph (Scattered line graph with legend)



- Economic Data _ GDP Composition Visualization

검색한 국가에 해당하는 GDP 정보 추출하게 설정

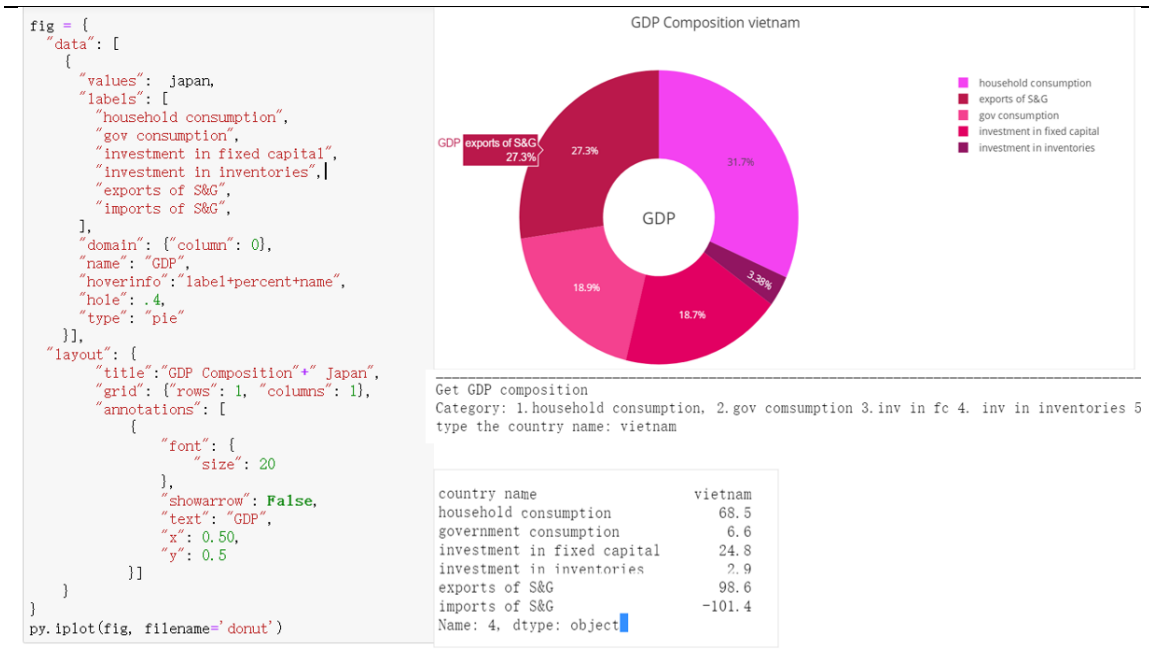
```
import pandas as pd
GDP_comp = pd.read_csv('GDP_composition.csv')
GDP_comp
GDP_comp1 = GDP_comp.T
GDP_comp2 = GDP_comp1.iloc[1:, ]

print("_____")
print("Get GDP composition")
print("Category: 1.household consumption, 2.gov consumption 3.inv in fc 4. inv in inventories 5. expoert of S&G 6.imports of S&G")

a = input("type the country name: ")
for i in range(0,30):
    if GDP_comp1.iloc[:, i][0] == a:
        print()
        print(GDP_comp1.iloc[:, i])
```

모든 국가의 데이터를 크롤링해서 한번에 보여주는 것은 가독성이 떨어질 것이라고 판단하여 데이터를 알고 싶은 국가를 input 으로 받아서 해당 국가 데이터를 보여주는 형식으로 코드를 짰다. 다음 예시는 vietnam 을 검색한 것이며 검색을 하면 vietnam 의 GDP 구성 정보와 함께 설정한 pie chart 형식으로 차트를 확인할 수 있다. 모두 마우스를 갖다 대면 해당 숫자, 항목이 표시되게 하였다.

GDP composition Pie Chart with Legend



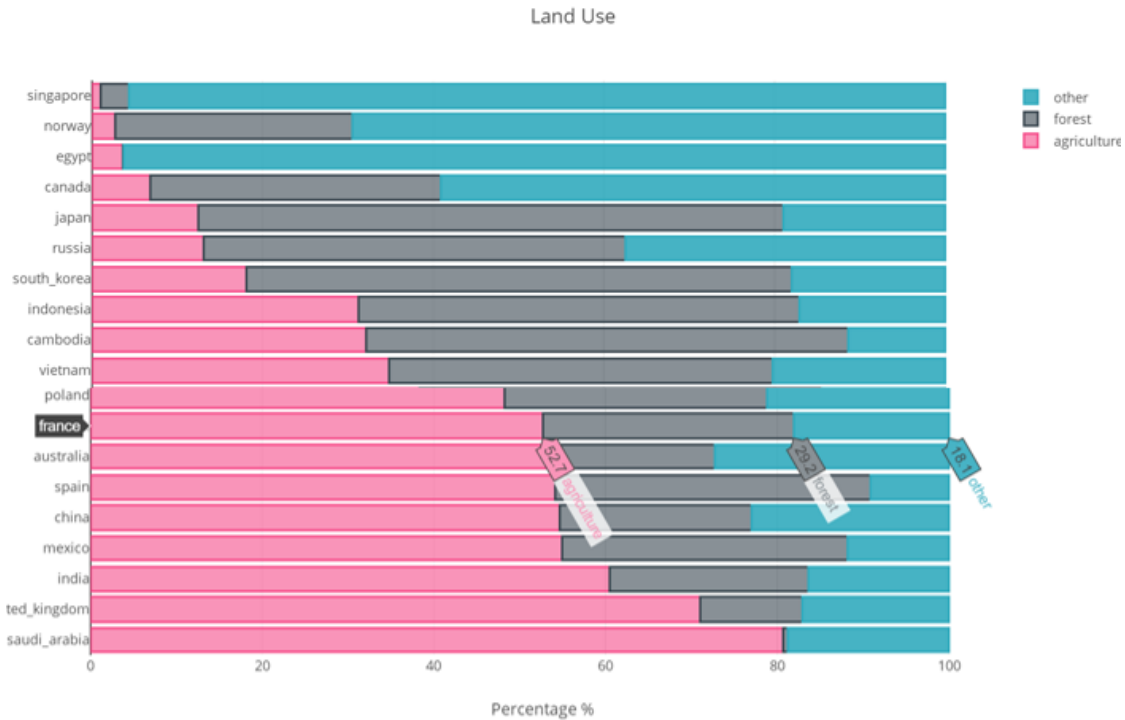
• Geographic Data _ Land Use Ratio Visualization

Land use 데이터는 각 국가의 땅의 agriculture, forest, etc 가 차지하는 비율을 나타낸 데이터이다. 다음과 같이 horizontal bar graph 를 제작했다.

Horizontal Bar Graph로 Land_Use 데이터 표현하는 코드



Land Use Data _Horizontal Bar Graph



- Geographic Data _ Area Composition Visualization
비율이 아닌 해당 면적 데이터만 제공해주기 때문에 비율을 계산하여서 기존 dataframe 에 열을 두개 추가하였다. 추가된 열을 이용하여 시각화를 진행했다.

비율을 표현하는 Columns를 추가

```
area_comp1['land%'] = area_comp1['land'].div(area_comp1['total'])*100
area_comp1['water%'] = area_comp1['water'].div(area_comp1['total'])*100

area_comp2 = area_comp1.sort_values(by='land%', ascending = False)
area_comp2
```

▲column 두개 추가

	country name	total	land	water	land%	water%
29	saudi_arabia	2149690.0	2149690.0	0.0	100.000000	0.000000
3	thailand	513120.0	510890.0	2230.0	99.565404	0.434596
12	france	643801.0	640427.0	3374.0	99.475925	0.524075
21	egypt	1001450.0	995450.0	6000.0	99.400869	0.599131
15	philippines	300000.0	298170.0	1830.0	99.390000	0.610000
20	portugal	92090.0	91470.0	620.0	99.326746	0.673254
10	united_kingdom	243610.0	241930.0	1680.0	99.310373	0.689627
23	australia	7741220.0	7682300.0	58920.0	99.238880	0.761120
25	belgium	30528.0	30278.0	250.0	99.181080	0.818920
17	mexico	1964375.0	1943945.0	20430.0	98.959975	1.040025
14	spain	505370.0	498980.0	6390.0	98.735580	1.264420

Plotly 상에서도 subplot 이 존재하기 때문에 한번에 두 개의 정보를 한꺼번에 다른 그래프로 확인할 수 있게 했다. 중요하다고 판단되는 농지비율과 육지 비율을 각각 bar graph, line graph 로 표현하였다.

Land_use, Area_comp 데이터를 한꺼번에 Subplot 이용하여 표현

```
y_agr = land_use2['agriculture_land']
y_land = area_comp2['land%']
x_agr = land_use2['country name']
x_land = area_comp2['country name']

trace0 = go.Bar(
    x=y_agr,
    y=x_agr,
    marker=dict(
        color='rgba(244, 66, 185, 0.6)',
        line=dict(
            color='rgba(244, 66, 185, 1.0)',
            width=1),
    ),
    name='argriculture land',
    orientation='h',
)
tracel = go.Scatter(
    x=y_land,
    y=x_land,
    mode='lines+markers',
    line=dict(
        color='rgba(244, 66, 185, 1.0)',
        width=1),
)
```

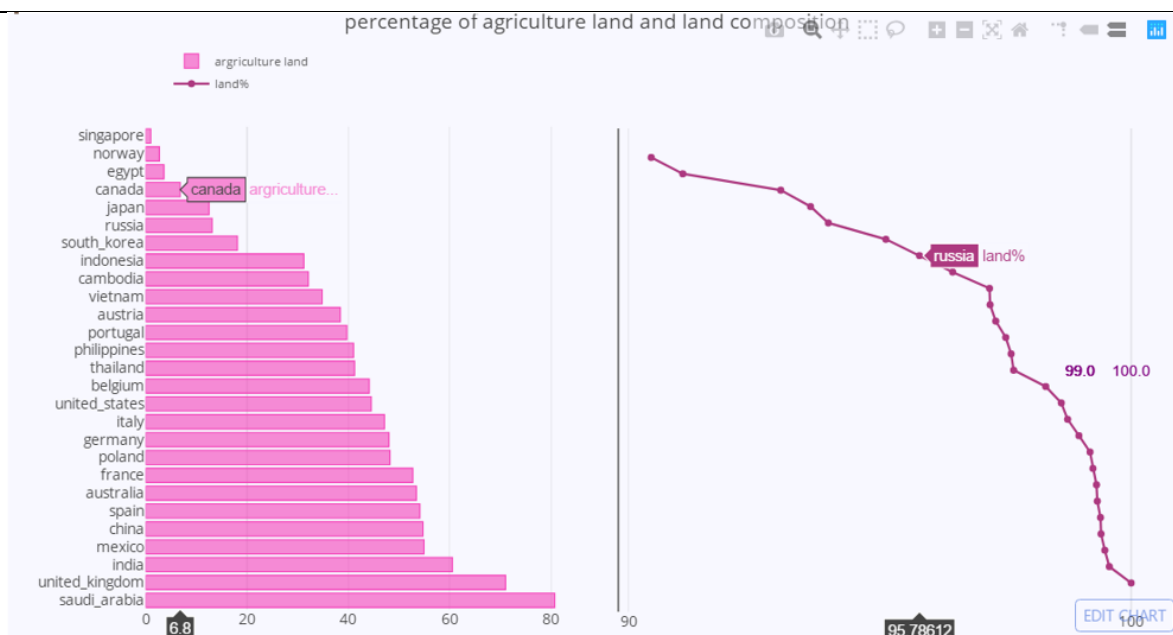
Subplot 두개로 다른 두 정보 한번에 볼 수 있게

```
# Creating two subplots
fig = tools.make_subplots(rows=1, cols=2, specs=[[0], [0]], shared_xaxes=True,
                          shared_yaxes=False, vertical_spacing=0.001)

fig.append_trace(trace0, 1, 1)
fig.append_trace(tracel, 1, 2)

fig['layout'].update(layout)
py.iplot(fig, filename='area composition and land_use')
```

Agriculture %, Land % subplots _ Bar Graph, Line Graph



- Demographic Data _ Age Structure Visualization

인구데이터를 seaborn 으로 표현했을 때 한꺼번에 많은 국가들의 정보를 확인할 수는 있으나 가독성이 떨어졌고 30 개국이나 되기 때문에 비교가 힘들었다.

따라서 전세계 지도에 비율을 색상 강도로 표현하여 한눈에 비교 가능하게 만들었다. 중요하다고 판단되는 세 group 의 데이터를 이용하였으며 고령 인구는 핑크색, 청년인구는 파란색, 경제활동인구는 초록색으로 나타냈다. 색상이 진할수록 비율이 높은 것이며 낮을수록 비율이 낮은 것이다.

확실하게 비율이 높게 나타나는 나라들을 빨리 찾을 수 있었으며 이번에도 해당 국가에 마우스를 대면 국가 이름과 비율 정보가 바로 legend 로 나타나게 했다.

3가지 Age Group의 정보를 세계지도에 표시하기

```
scl = [
    [0.0, 'rgb(20, 0, 14)'],
    [0.2, 'rgb(79, 7, 61)'],
    [0.4, 'rgb(206, 30, 153)'],
    [0.6, 'rgb(239, 93, 195)'],
    [0.8, 'rgb(239, 184, 223)'],
    [1.0, 'rgb(252, 244, 250)']
]

meta_data = dict(type = 'choropleth',
    locations = age_structure['country name'],
    colorscale = scl,
    reversescale = True,
    locationmode = 'country names',
    z = age_structure['65+'],
    marker = dict(line=dict(color='black',
        width=1)),
    text = age_structure['country name'],
    hoverlabel = dict(bgcolor = '#e209ac',
        font=dict(family='Times New Roman',
            color='white')),
    colorbar = {'title': 'Aged Rate', 'nticks': 9})

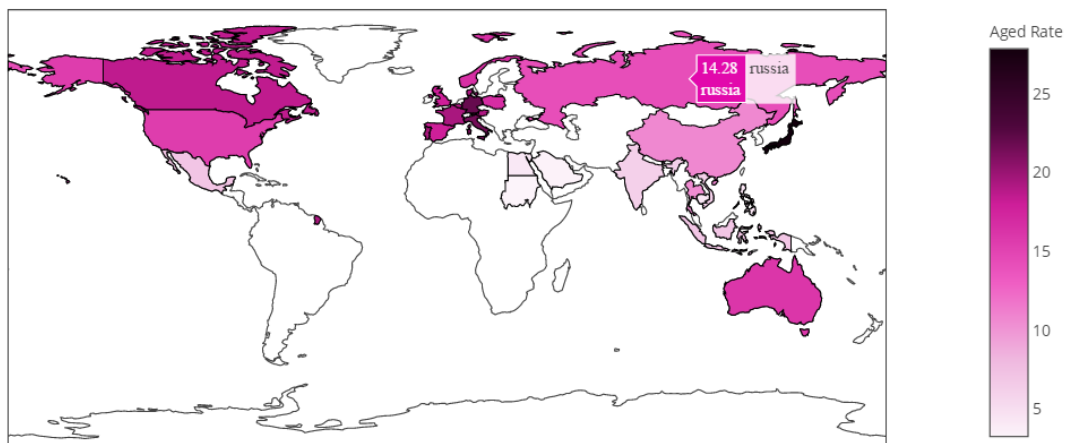
layout = dict(title = 'Global Choropleth Plot By Aged Rate(65+%)',
    geo = dict(showframe = True,
        projection = ['type': 'equiarectangular']))

My_choromap = go.Figure(data = [meta_data], layout=layout)
py.iplot(My_choromap)
```

	country name	0-14	15-24	25-54	54-64	65+
0	japan	12.84	9.64	37.50	12.15	27.87
1	india	27.34	17.90	41.08	7.45	6.24
2	china	17.15	12.78	48.51	10.75	10.81
3	thailand	16.93	14.17	46.32	12.00	10.58
4	vietnam	23.55	16.23	45.56	8.55	6.12
5	cambodia	31.01	18.36	40.68	5.69	4.25
6	singapore	12.82	16.56	50.53	10.46	9.63
7	indonesia	25.02	16.99	42.40	8.58	7.01
8	canada	15.44	11.85	39.99	14.10	18.63
9	united_states	18.73	13.27	39.45	12.91	15.63
10	united_kingdom	17.53	11.90	40.55	11.98	18.04
11	germany	12.82	10.09	40.45	14.58	22.06
12	france	18.53	11.79	37.78	12.42	19.48
13	italy	13.65	9.66	42.16	12.99	21.53
14	spain	15.38	9.58	44.91	12.14	17.98
15	philippines	33.39	19.16	36.99	5.97	4.49
16	south_korea	13.21	12.66	45.52	14.49	14.12
17	mexico	26.93	17.54	40.81	7.64	7.09

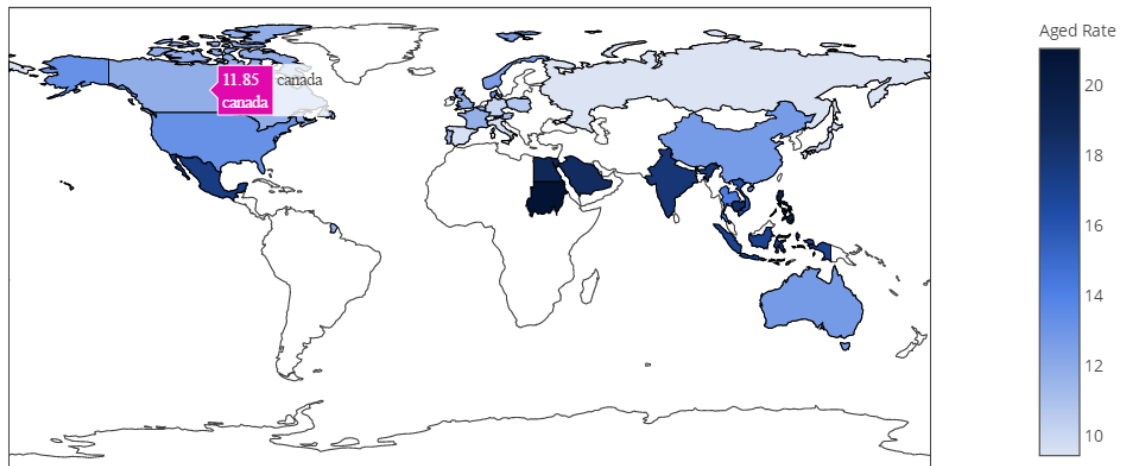
Choropleth Plot by Aged Ratio (Population over 65)

Global Choropleth Plot By Aged Rate(65+%)



Choropleth Plot by Young Age Group Ratio (Population 15–24%)

Global Choropleth Plot By Aged Rate(15-24%)



Choropleth Plot by Working Age Group Ratio (Population 25–54%)

Global Choropleth Plot / Economically Active Population (25-54%)



V. 결론

이번 프로젝트는 크롤링 기반의 시각화에 초점을 맞춰서 진행해 보았다. 수업시간에 다뤘던 matplotlib의 심미성, 다양성 측면에서 한계를 느껴서 가장 다양한 그래프를 제공하는 plotly를 중심으로 그래프를 만들었다.

최적화된 그래프를 이용하여 시각화를 시켜 다양한 색상을 적용해보니 숫자로만 표현되어 있던 너무나 많은 데이터가 한 눈에 비교 가능했다. 처음에 달성하고자 하는 프로젝트 목표를

이렸으며 이번 프로젝트뿐 아니라 계속해서 개선 및 확장해서 이용가능한 코드를 제작하였다는 것에도 의의가 있다. Countrydatacode.csv 에 크롤링/시각화 하여 확인하고 싶은 국가 이름만 추가하면 전세계 데이터를 다 적용 가능하다. 하지만 웹 상에서 정구식으로 표현한 표현들이 자주 변하기 때문에 웹페이지 업데이트에 따른 코드 수정도 주기적으로 해주어야 할 필요가 있다.

추후에 진행했던 이번 프로젝트를 좀 더 심화된 방향으로 발전시키고자 한다. 크롤링한 indexmundi.com 에서는 각 국가마다 거의 모든 공개 데이터를 제공하고 있기 때문에 politics/government 와 관련된 government type, constitution, national symbol 등의 데이터가 모두 확인 가능하다. 이후에는 숫자가 아닌 문자데이터를 수집하여 비슷한 성향의 국가들끼리 grouping 하여 분석하는 작업을 할 것이다. Visualization 은 plotly 모듈을 이용할 것이다.