텍스트 마이닝과 데이터 마이닝

Part 08. 데이터 마이닝 실습

정 정 민

programmers

Chapter 21. 글로벌 인구 통계 추세 데이터 실습

1. 글로벌 인구 통계 추세 데이터
2. 지도 그래프 활용하기
3. Bar Plot 살펴보기

글로벌 인구 통계 추세 데이터

강의 실습 코드 : 링크

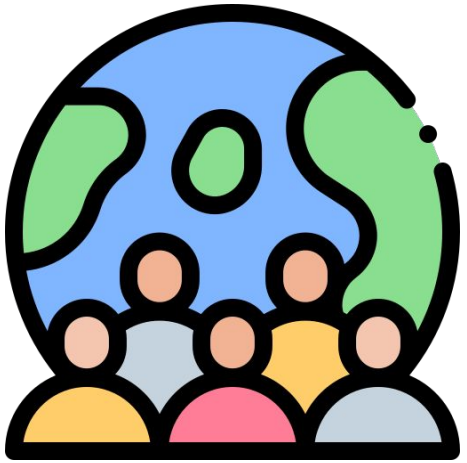
글로벌 인구 통계 추세 데이터 (WPP2022 Demographic Indicators)

- 이번 실습에서 사용할 데이터로 Kaggle의 공개 데이터 (링크)
  - 다운로드 받아주세요!

- 1950년부터 2020년까지 세계적인 인구 통계적 추세 정보를 담고 있음
- 인구수(전체, 남, 여), 출생률, 출산률, 평균 출산 나이, 기대수명, 사망률 등 다양한 항목이 존재

- 총 54개의 통계적 항목이 존재
- 각 항목에 대한 구체적인 설명은 여기를 참고!
  - Kaggle 데이터에는 축약어로 표시
  - 원래 항목의 이름과 축약어끼리의 비교 필요

- 혹은 설명 데이터 다운로드 가능 : 링크



타겟 데이터 선정

- 모든 항목의 추세를 보기에는 너무 많은 양
- 특정 년도 1월의 전체 인구수를 나타내는 항목 (“TPopulation1Jan”) 을 기준으로 선택

- 지역의 경우도 단순 나라의 구성만 선택
  - 그 외에 대륙, 소득 기준 그룹, 개발 도상국 등이 존재
  - 총 237개국 선택

```
import pandas as pd

file_path = 'WPP2022_Demographic_Indicators.csv'
data = pd.read_csv(file_path)

data = data[data['TPopulation1Jan'].notnull()]
country_data = data[data['LocTypeName']=='Country/Area']
```

지역의 위도 경도 추가

- 이번 실습에서는 지도를 그리면서 시각화를 할 예정
- 지도를 그리려면 각 지역의 위도와 경도 정보가 필요

- 일반적으로 나라와 관련된 데이터에는 들어있지만
- 이번 데이터는 그렇지 않아 외부 패키지(geopy)를 통해 추출할 예정

- 나라 이름을 입력하면 위도와 경도를 반환
- API의 형태이므로
  - 시간이 오래 걸리고
  - 너무 빠른 호출 금지
  - IP당 호출 횟수 제한 존재

- 위와 같은 문제가 있어서 이미 만들어둔 파일 제공 : 링크

```
# 방법 1
geolocator = Nominatim(user_agent="geoapiExercises")
location_coordinates = {}

for location in all_countries:
    loc = geolocator.geocode(location)
    location_coordinates[location] = (loc.latitude, loc.longitude)

# 방법 2
with open('location_coordinates.json', 'r') as f:
    location_coordinates = json.load(f)
```

지도 그래프 활용하기

특정 해의 인구 관련 지도 그래프 그리기

- 2020년의 인구 수치를 기반으로 지리적 데이터를 지도에 시각화

- folium이라는 파이썬 패키지 활용
  - 기본 지도를 생성
  - 그 위에 다양한 마커와 레이어를 추가할 수 있음
  - 약간의 인터랙티브 요소도 추가 가능

```
import folium
m = folium.Map(location=[20, 0], zoom_start=2)
m
```

지도에 인구수 통계 값 표시 (원 활용)

- 지도 위에 원을 활용해 인구수를 시각적으로 표시
- 상대적인 인원수에 따라 원의 크기를 조절

- 원을 지도위에 그리기 위해 필요한 정보 제공
  - 위치 : 경도 & 위도
  - 원의 크기
    - 어떻게 보여줘야 좋을까요?

- 추가적으로 원을 클릭 시 필요한 정보 제공
  - 기본 popup argment로 전달
  - 메시지를 str의 형태로 제공
  - 원하는 정보를 쉽게 띄울 수 있음

```
m = folium.Map(location=[20, 0], zoom_start=2)

for idx, row in country_data_2020.iterrows():
    radius = row['TPopulation1July']
    popup_message = f"{row['Location']}: {row['TPopulation1July']}"

    folium.Circle(
        location=[row['latitude'], row['longitude']],
        radius=radius,
        color='blue',
        fill=True,
        fill_color='blue',
        popup=popup_message
    ).add_to(m)
```

지도에 인구수 통계 값 표시 (히트맵 활용)

- 지도 위에 히트맵(HeatMap)을 덮어 씌워 시각적으로 표시 가능
- 색을 통해 인구수 표시 가능
  - 인구수가 많다면 붉을 계열
  - 적다면 푸른 계열

- Argment를 통해 Heatmap의 시각 정보를 변경할 수 있음
  - min_opacity : 투명도 설정
  - raidus : 데이터 포인트의 영향 반경
  - blur : 블러 효과, 값이 크면 부드럽게 표현
  - gradient : 히트맵의 색상을 사용자가 정의
- 지도의 크기, 범위에 따라 보여지는 영향력이 다름
- 최적의 시각화를 위해 여러번 테스트 진행

```
import folium
from folium.plugins import HeatMap

m = folium.Map(location=[20, 0], zoom_start=2)

heat_data = [[row['latitude'], row['longitude'], row['TPopulation1July']] \
             for idx, row in country_data_2020.iterrows()]

gradient = {0.0: 'blue', 1.0: 'red'}

HeatMap(heat_data,
        min_opacity=0.5,
        radius=30,
        blur=25,
        gradient=gradient).add_to(m)

m
```

시간에 따른 인구수 통계 값 확인

- 앞선 예시는 모두 2020년의 데이터만 본 결과물

- 파이썬 패키지인 IPyWidgets을 활용하면
- 사용자 인터랙션의 결과로 특정 값을 반환 받을 수 있는 위젯을 사용할 수 있음

- 필요한 위젯을 설정
  - 연도를 컨트롤 하는 위젯
  - 지도를 그리는 위젯

- 2020년 데이터가 아니라 전체 데이터를 호출

- 과정
  - 연도 위젯의 값이 변하면
  - 데이터에서 특정 연도의 데이터를 불러와서
  - 지도에 그림을 그리기!

```python
import ipywidgets as widgets

# 필요한 위젯 설정
year_slider = widgets.IntSlider()
map_output = widgets.Output()

# year_slider의 변화를 감지(observe)하고, (특히 value라는 속성이 변경되면)
# 변화가 있다면 on_year_change 함수를 실행
year_slider.observe(on_year_change, names='value')

# on_year_change 함수는 지도를 다시 그리는 update_map 함수 호출
# 여기에는 변경 사항 중 새로운 값(new)을 입력으로 넣어줌
def on_year_change(change):
    update_map(change['new'])

# 지도를 새로 그리는 함수
def update_map(year):
    # 지도 output 위젯 안에서
    with map_output:
        # 이전 지도를 지우고
        clear_output(wait=True)
        # 새로운 지도 생성
        ...
```

Bar Plot 살펴보기

기본 Bar Plot 적용

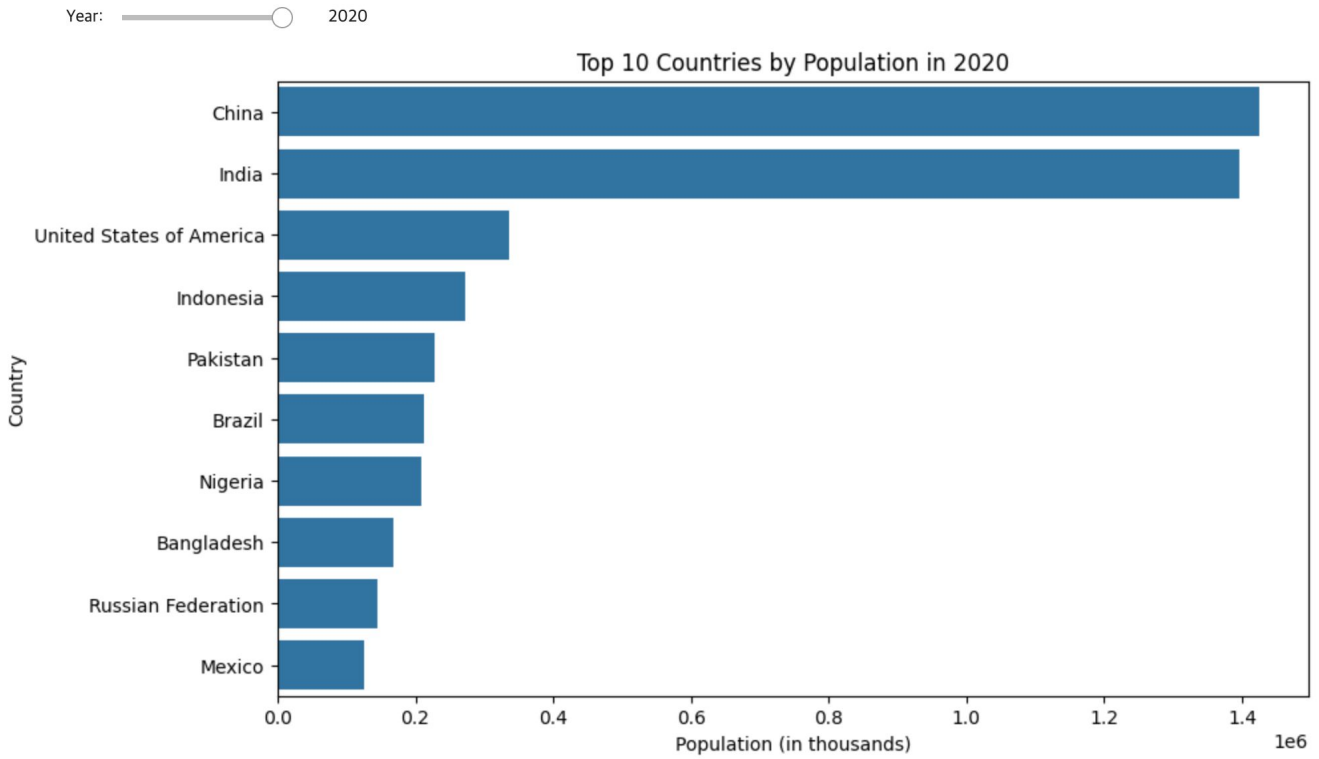
- 수의 양적 차이를 직관적으로 보여주는 그래프는 Bar Plot

- Bar Plot을 위젯 내부에 그려줌
- 연도 변화에 따른 서로 다른 데이터를 활용

- Matplotlib의 plt 말고도
- seaborn이라는 패키지를 활용할 수 있음

- Seaborn은 Matplotlib를 기반으로 함
- 기본 기능과 추가 개선된 기능을 제공

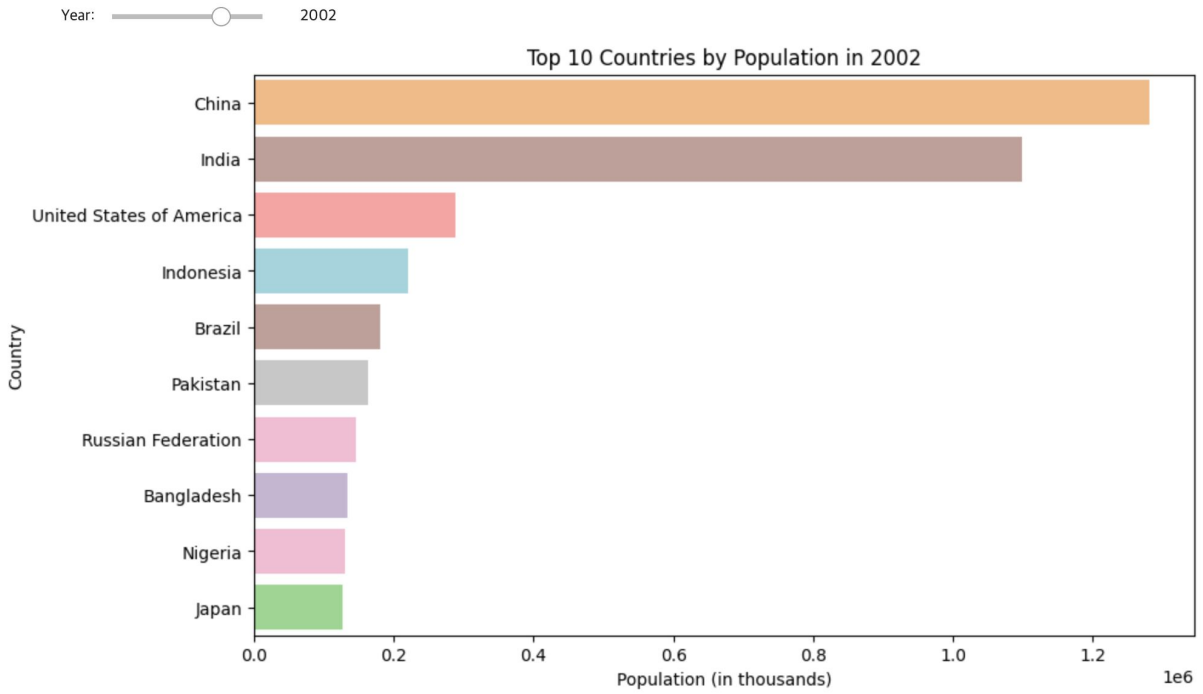
- 대신, 더욱 쉽고 직관적인 API 제공
- 또한 많은 스타일과 테마 제공



나라마다 고유한 색깔 지정

- 특정 나라의 어떤 항목의 순위는 연도 변화에 따라 다를 수 있음
- 이런 경우 나라마다 고유한 색을 지정하면 변화가 직관적으로 보일 수 있음

- 나라 수 만큼 색을 지정하고
- 이것을 나라 이름마다 지정 후 시각화 진행




```python
import ipywidgets as widgets

# 필요한 위젯 설정
year_slider = widgets.IntSlider()
bar_output = widgets.Output()

# 전체 데이터셋에서 고유한 나라들 추출
unique_countries = data_clean['Location'].unique()

# 고유한 나라들에 대한 색상 매핑 생성
colors = sns.color_palette("tab20", len(unique_countries))
country_colors = dict(zip(unique_countries, colors))

def update_map(year):
    with bar_output:
        ...
        # palette에 사용할 색 리스트를 지정
        sns.barplot(x='TPopulation1July', y='Location', data=top10_data,
                    palette=[top10_colors[country] \
                             for country in top10_data['Location']])
        ...
```

E.O.D