VI. 지수와 지도의 시각화

이번 장에서는 지금까지 살펴본 시각화외에도 plotly 에서 추가적으로 제공하는 시각화 중 지수(index)의 시각화와 지도의 시각화를 살펴보도록 하겠다.

지수(Index)는 데이터 들을 특징일 종합하여 하나의 수치로 표현한 것을 말한다. 평균, 표준편차와 같은 수치도 가장 흔하게 만날 수 있는 지수 중에 하나이고 매일 매일 주식 시장 전체의 가격 변동을 종합한 KOPIS 나 KOSDAQ 와 같은 주가 종합 지수도 있다. 이렇게 하나의 수치를 시각화로 표현하는 방법을 plotly 에서 제공한다.

지도(Map)의 시각화는 매우 흔하게 사용하는 시각화로 보통 행정 구역과 같은 공간 단위별로 수집된 데이터에 대해 무늬나 색상의 단계로 데이터를 표현하는 단계 구분도(Choropleth map)나 지도위에 위경도 좌표를 지닌 객체들을 표현하는 분산 맵(Scatter map) 등이 사용된다.

1. 인디케이터(indicator)

인디케이터의 사전적 의미는 '가리키는 것'이다. 특정한 지시, 징후, 징표를 가리키는 것이라는 의미가 적절할 것이다. plotly 에서 인디케이터의 의미는 'value' 속성으로 지정된 단일 값을 시각화하는 것이다. 인디케이터의 속성은 다음의 네 가지 속성이 첫번째 레벨의 속성이다.

- value: 시각화 할 값
- mode: 인디케이터의 종류
- align: 수치와 델타 인디케이터의 정렬(left, center, right)
- domain: 인디케이터의 위치 범위

plotly 에서는 인디케이터를 시각화하기 위해 indicator 트레이스를 제공한다. indicator 트레이스는 인디케이터로 표현할 수치를 수치, 델타, 게이지의 세 가지 방법을 사용하여 시각화하는 것이다. 이 세가지 방법은 'mode' 속성을 통해 설정이 가능한데, 다음은 indicator 트레이스에서 사용하는 주요 속성이다.

속성			속성 설명	속성값(R 속성값)
title	align		제목의 가로 정렬 설정	("left" | "center" | "right")
	font	color	제목의 폰트 색상 설정	색상값
		family	제목의 폰트 이름 설정	폰트이름
		size	제목의 폰트 크기 설정	0이상의 수치
	text		제목 문자열 설정	문자열
visible			트레이스의 표시여부 설정	(True | False | "legendonly")
mode			value의 표시 모드 설정	"number", "delta", "gauge" 중 하나의 문자열이나 '+'를 사용한 조합
value			표시될 수치 설정	수치
align			박스 안 문자열의 가로 정렬 설정	("left" | "center" | "right")

1.1. 수치(number) 인디케이터

수치 인디케이터는 단일 수치값을 시각화하는 방법이다. 이 수치 인디케이터는 indicator 트레이스 속성 중에 'number' 속성에 의해 설정이 가능하다. 다음은 수치 인디케이터의 설정에 사용하는 'number'의 주요 속성이다.

속성			속성 설명	속성값(R 속성값)
number	font	color	수치 폰트 색상 설정	색상값
		family	수치 폰트 이름 설정	문자열
		size	수치 폰트 크기 설정	색상값
	prefix		수치 접두어 설정	문자열
	suffix		수치 접미어 설정	문자열
	valueformat		수치 값의 표시를 위한 d3 포맷 설정	문자열

수치 인디케이터는 전체 인디케이터에서 표현되는 부분 중에 수치값을 표현하는 부분으로, 수치 인디케이터가 단독으로 사용되기 보다는 뒤에 설명할 'delta'나 'guage'와 함께 전체 인디케이터를 구성하는 일부분으로 사용되는 경우가 일반적이다. 하지만 단순한 현재 수치를 표현할 때는 단독으로 사용될 수 있다.

다음은 주요 5 개국의 코로나 19 사망자수 선 그래프에 한국의 코로나 사망자수를 indicator 트레이스로 추가하는 R 과 python 코드이다. 한국의 코로나 사망자수를 그리고 나서 그래프 영역의 특정 부분(X 축의 0.4~0.6, Y 축의 0,85~0.95)에 수치 속성만을 가지는 indicator 트레이스를 추가하였다. indicator 트레이스의 'value'를 설정하기 위해 앞서 코로나 사망자수를 그리는데 사용했던 total_deaths_5_nations_by_day 데이터프레임에서 한국의 10 만명당 사망자수 데이터를 산출하고 이 데이터를 사용하여 indicator 트레이스를 생성하였다. indicator 트레이스의 위치를 설정하는데 'domain' 속성을 사용한다.

- R

R 에서 수치 인디케이터를 만들기 위해서는 indicator 트레이스를 만드는데, `add_trace(type = 'indicator', ...)`를 사용한다. 수치 인디케이터는 'mode' 속성을 "number"로 설정하고 'number' 속성을 설정하여 만들 수 있다.

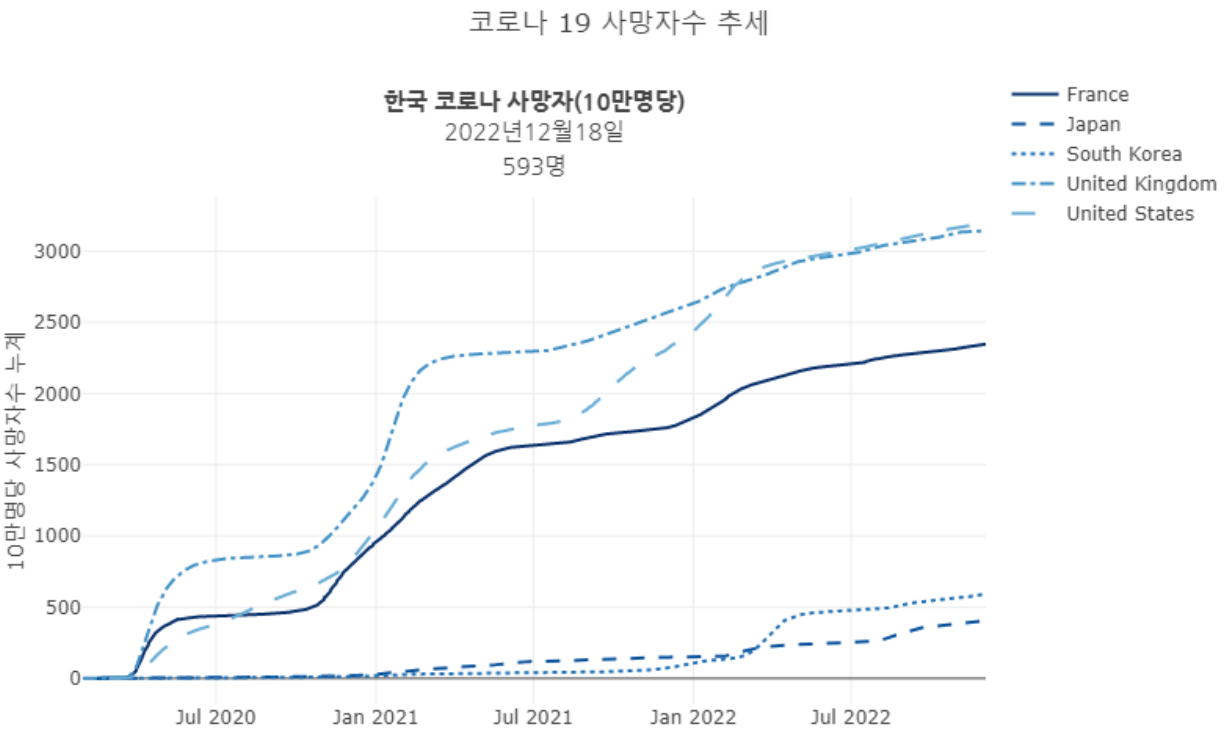
```
## indicator 트레이스를 위한 데이터 전처리
number_KOR <- total_deaths_5_nations_by_day |>
  filter(date == max(date), iso_code == 'KOR') |>
  select(total_deaths_per_million) |> pull()

fig <- total_deaths_5_nations_by_day |>
  plot_ly() |>
  ## scatter 트레이스 생성
  add_trace(type = 'scatter', mode = 'lines',
            x = ~date, y = ~total_deaths_per_million ,
            linetype = ~location, connectgaps = T) |>
  ## layout 의 제목, 축제목, 여백 속성 설정
  layout(title = list(text = '코로나 19 사망자수 추세', pad = list(b = 5)),
         xaxis = list(title = ''),
         yaxis = list(title = '10 만명당 사망자수 누계', domain = c(0, 0.8)),
         margin = margins_R)

## number 모드의 indicator 트레이스 추가
fig |> add_trace(type = 'indicator', mode = 'number', value = number_KOR,
            title = list(text = paste0('<b>한국 코로나 사망자(10 만명당)</b>\n', ye
ar(today()), '년', month(today()), '월', day(today()), '일'),
                         font = list(family = '나눔고딕', size = 15)
                         ),
            ## number 속성 설정
            number = list(font = list(family = '나눔고딕', size = 15),
                          suffix = '명'),
            domain = list(x = c(0.4, 0.6), y = c(0.8, 0.9)))
```

- python

python 에서 수치 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하고, `add_trace()`에 `plotly.graph_objects.Indicator()`를 사용한다. 수치 인디케이터는 'mode' 속성을 "number"로 설정하고 'number' 속성을 설정하여 만들 수 있다.

```
from datetime import datetime, timedelta

## indicator 트레이스를 위한 데이터 전처리
today = datetime.today()
```

```
number_KOR = total_deaths_5_nations_by_day.loc[(total_deaths_5_nations_by_day['d
ate']==total_deaths_5_nations_by_day['date'].max()) & (total_deaths_5_nations_by
_day['iso_code']=='KOR')]
number_KOR = number_KOR.loc[:,'total_deaths_per_million'].values[0]

fig = go.Figure()

## scatter 트레이스 생성
for location, group in total_deaths_5_nations_by_day.groupby('location'):
    fig.add_trace(go.Scatter(
        mode = 'lines',
        x = group['date'], y = group['total_deaths_per_million'],
        line = dict(dash = nations[location]),
        name = location, connectgaps = True))

fig.update_layout(title = dict(text = '코로나 19 사망자수 추세', x = 0.5),
                  xaxis = dict(title = ''),
                  yaxis = dict(title = '10 만명당 사망자수 누계', domain = (0, 0.
8)))

## number 모드의 indicator 트레이스 추가
fig.add_trace(go.Indicator(mode = 'number', value = number_KOR,
    title = dict(text = '<b>한국 코로나 사망자(10 만명당)</b><br>' + str(today.yea
r) + '년' + str(today.month) + '월' + str(today.day) + '일\n',
                 font = dict(family = '나눔고딕', size = 15)),
    ## number 속성 설정
    number = dict(font = dict(family = '나눔고딕', size = 15),
                  suffix = '명'),
    domain = dict(x = (0.4, 0.6), y = (0.8, 0.9))))

fig.show()
```



실행결과 VI-1. 수치 인디케이터가 포함된 R 의 선 그래프

1.2. 델타(delta) 인디케이터

델타 인디케이터는 일반적으로 수치 인디케이터와 병행하여 사용하여 수치 인디케이터에 표현된 값이 비교 기준값과의 차이를 기호와 증감값을 사용하여 보여주는 인디케이터를 말한다. 델타 인디케이터에는 수치 인디케이터에 비해 다음과 같은 몇 가지 요소들이 필요하다.

- reference : 값의 증감을 비교할 기준 값
- relative : 값의 증감이 절대적인지, 상대적인지를 결정
- increasing : 값이 증가할 경우 표시할 속성
- decreasing : 값이 감소할 경우 표시할 속성

다음은 델타 인디케이터의 설정에 사용하는 'delta'의 주요 속성이다.

속성			속성 설명	속성값(R 속성값)
delta	decreasing	color	하락 색상 설정	색상값
		symbol	하락 심볼 설정	문자열
	font	color	하락 폰트 색상 설정	색상값
		family	하락 폰트 이름 설정	폰트이름
		size	하락 폰트 크기 설정	0이상의 수치
	increasing	color	상승 색상 설정	색상값
		symbol	상승 심볼 설정	문자열
	position		수치와 관련한 심볼 위치 설정	("top" | "bottom" | "left" | "right")
	prefix		델타의 접두어 설정	문자열
	reference		델타를 계산할 관련 값 설정	수치
	relative		상대적 변화 표시 여부 설정	논리값
	suffix		델타의 접미어 설정	문자열
	valueformat		델타 값의 표시를 위한 d3 포맷 설정	문자열

- R

R 에서 델타 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하는데, `add_trace(type = 'indicator', ...)`를 사용한다. 수치 인디케이터는 'mode' 속성을 "delta"로 설정하고 'delta' 속성을 설정하여 만들 수 있다.

```
number1_KOR <- total_deaths_5_nations_by_day |>
  filter(date == max(date)-1, iso_code == 'KOR') |>
  select(total_deaths_per_million) |> pull()

fig <- total_deaths_5_nations_by_day |> plot_ly() |>
  ## scatter 트레이스 생성
  add_trace(type = 'scatter', mode = 'lines',
            x = ~date, y = ~total_deaths_per_million ,
            linetype = ~location, connectgaps = T) |>
  ## layout 의 제목, 축제목, 여백 속성 설정
  layout(title = list(text = '코로나 19 사망자수 추세', pad = list(b = 5)),
         xaxis = list(title = ''),
         yaxis = list(title = '10 만명당 사망자수 누계', domain = c(0, 0.8)),
         margin = margins_R)

## number+delta 모드의 indicator 트레이스 추가
fig |> add_trace(type = 'indicator', mode = 'number+delta', value = number_KOR,

            title = list(text = paste0('<b>한국 코로나 사망자(10 만명당)</b>\n', ye
ar(today()), '년', month(today()), '월', day(today()), '일'),
                         font = list(family = '나눔고딕', size = 15)),
            number = list(font = list(family = '나눔고딕', size = 15),
                          suffix = '명'),
            ## delta 속성 설정
```

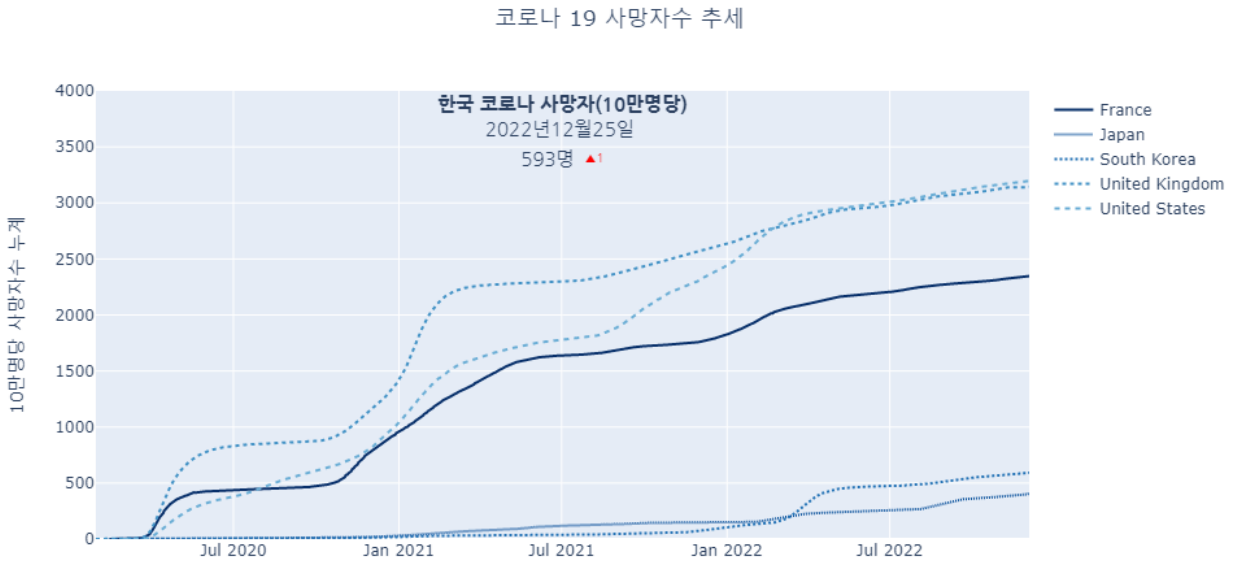
```
            delta = list(reference = number1_KOR, position = 'right',
                         increasing = list(color = 'red'),
                         decreasing = list(color = 'blue'),
                         font = list(family = '나눔고딕', size = 10)),
            domain = list(x = c(0.4, 0.6), y = c(0.8, 0.9)))
```

- python

python 에서 델타 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하고, `add_trace()`에 `plotly.graph_objects.Indicator()`를 사용한다. 델타 인디케이터는 'mode' 속성을 "delta"로 설정하고 'delta' 속성을 설정하여 만들 수 있다.

```
number1_KOR = total_deaths_5_nations_by_day.loc[(total_deaths_5_nations_by_day['
date']==total_deaths_5_nations_by_day['date'].max()- timedelta(days = 1)) & (tot
al_deaths_5_nations_by_day['iso_code']=='KOR')]
number1_KOR = number1_KOR.loc[:,'total_deaths_per_million'].values[0]

fig = go.Figure()
for location, group in total_deaths_5_nations_by_day.groupby('location'):
    fig.add_trace(go.Scatter(mode = 'lines',
        x = group['date'], y = group['total_deaths_per_million'],
        line = dict(dash = nations[location]),
        name = location, connectgaps = True))

fig.update_layout(title = dict(text = '코로나 19 사망자수 추세', x = 0.5),
                  xaxis = dict(title = ''),
                  yaxis = dict(title = '10 만명당 사망자수 누계', domain = (0, 0.
8)))

## number+delta 모드의 indicator 트레이스 추가
fig.add_trace(go.Indicator(
    mode = 'number+delta', value = number_KOR,
    title = dict(text = '<b>한국 코로나 사망자(10 만명당)</b><br>' + str(today.yea
r) + '년' + str(today.month) + '월' + str(today.day) + '일\n',
                 font = dict(family = '나눔고딕', size = 15)),
    number = dict(font = dict(family = '나눔고딕', size = 15),
                  suffix = '명'),
    ## delta 속성 설정
    delta = dict(reference = number1_KOR, position = 'right',
                 increasing = dict(color = 'red'),
                 decreasing = dict(color = 'blue'),
                 font = dict(family = '나눔고딕', size = 10)),
    domain = dict(x = (0.4, 0.6), y = (0.8, 0.9))))
```

```
fig.show()
```



실행결과 VI-2. 델타 인디케이터가 표시된 python 선 그래프

1.3. 게이지(gauge) 인디케이터

게이지 인디케이터는 둥근 반원의 형태의 틀에 둥글게 그려지는 막대의 각도에 의해 최소값과 최대값 사이에서의 현재의 지수를 표현할 때 사용하는 시각화이다. 이 시각화를 구성하는데에는 주 표현값인 'value' 값 이외에 'steps', 'threshold', 'delta'의 속성을 사용한다. 게이지 인디케이터는 indicator 트레이스에 'mode' 속성에 "gauge"를 포함하여 만들 수 있다. "gauge"만을 사용하거나, "number+gauge"를 사용하여 수치와 함께 게이지를 사용하거나, "number+gauge+delta"를 사용하여 수치, 게이지, 증감 기호를 모두 사용할 수도 있다.

다음은 게이지 인디케이터의 설정에 사용하는 'gauge'의 주요 속성이다.

속성				속성 설명	속성값(R 속성값)
gauge	axis	dtick		게이지 축의 눈금 간격 설정	수치나 좌표상의 변량값
		nticks		게이지 축의 눈금 개수 설정	0이상의 정수
		range		게이지 축의 범위 설정	list
		showticklabels		게이지 축의 눈금 라벨 표시여부 설정	논리값
		showtickprefix		게이지 축의 눈금 접두어 표시여부 설정	("all" | "first" | "last" | "none")
		showticksuffix		게이지 축의 눈금 접미어 표시여부 설정	("all" | "first" | "last" | "none")
		tick0		게이지 축의 눈금 시작값 설정	수치나 좌표상의 변량값
		tickcolor		게이지 축의 눈금 색상 설정	색상값
		tickfont	color	게이지 축의 눈금 폰트 색상 설정	색상값
			family	게이지 축의 눈금 폰트 이름 설정	폰트이름
			size	게이지 축의 눈금 크기 설정	0이상의 수치
		ticktext		tickvals'와 관련한 눈금 레벨 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
		tickvals		눈금으로 표시될 값 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
	bar	color		게이지 내부 막대의 배경색 설정	색상값
		line	color	게이지 내부 막대의 외곽선 색 설정	0이상의 수치
			width	게이지 내부 막대의 외곽선 두께 설정	0이상으 | 수치
		thickness		전체 게이지 두께에 비례한 내부 막대의 두께 설정	0부터 1사이의 수치
	bgcolor			게이지 배경색 설정	색상값
	bordercolor			게이지 외곽선 색상 설정	색
	borderwidth			게이지 외곽선 두께 설정	0이상의 수치
	shape			게이지의 모양 설정	("angular" | "bullet")
	steps	color		게이지의 단계 색상 설정	색상값
		line	color	게이지의 단계 외곽선 색상 설정	색상값
			width	게이지의 단계 두께 설정	0이상의 수치

다음의 코드는 게이지 인디케이터를 만드는 R과 python의 코드이다. 게이지 인디케이터의 'value'로 데이터의 마지막날의 10만명당 사망자수를 표시하고, 임계치는 해당 국가의 10만명당 사망자수 최대값으로 설정하였다. 최소값은 0, 최대값은 임계의 1.2배 값으로 설정하고 step은 최대값의 50%와 75%로 설정하였다.

- R

R 에서 게이지 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하는데, `add_trace(type = 'indicator', ...)`를 사용한다. 수치 인디케이터는 'mode' 속성을 "gauge"로 설정하고 'gauge' 속성을 설정하여 만들 수 있다.

```
## 게이지 indicator 트레이스를 위한 데이터 전처리
max_deaths_per_million_by_day <- total_deaths_5_nations_by_day |> group_by(locat
ion) |>
  summarise(최대사망자 = max(new_deaths_per_million, na.rm = TRUE))

deaths_per_million_in_lateast <- total_deaths_5_nations_by_day |> group_by(locat
ion) |>
  filter(is.na(new_deaths_per_million) == FALSE) |>
  filter(date == max(date)) |>
  select(iso_code, date, new_deaths_per_million)

df_gauge <- left_join(max_deaths_per_million_by_day, deaths_per_million_in_latea
```

```
st, by = 'location') |> arrange(location)

## 한국 게이지 인디케이터 생성
fig_gauge <- df_gauge |> plot_ly() |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[3,
1]),
            domain = list(row = 1, column = 1), value = pull(df_gauge[3, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[3, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[3, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[3, 2])*1.2*0.5, pull(df_gauge[3,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[3, 2])*1.2*0.75, pull(df_gauge[3,
2])*1.2), color = "gray")),
              threshold = list(line = list(color = 'white'),
                value = pull(df_gauge[3, 2])),
              bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 프랑스 게이지 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[1,
1]),
            domain = list(row = 0, column = 0), value = pull(df_gauge[1, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[1, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[1, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[1, 2])*1.2*0.5, pull(df_gauge[1,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[1, 2])*1.2*0.75, pull(df_gauge[1,
2])*1.2), color = "gray")),
              threshold = list(line = list(color = 'white'),
              value = pull(df_gauge[1, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 일본 게이지 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[2,
1]),
            domain = list(row = 0, column = 2), value = pull(df_gauge[2, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[2, 2])*1.2)),
              steps = list(
```

```
                list(range = c(0, pull(df_gauge[2, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[2, 2])*1.2*0.5, pull(df_gauge[2,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[2, 2])*1.2*0.75, pull(df_gauge[2,
2])*1.2), color = "gray")),
              threshold = list(line = list(color = 'white'),
              value = pull(df_gauge[2, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 영국 게이지 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[4,
1]),
            domain = list(row = 2, column = 0), value = pull(df_gauge[4, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[4, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[4, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[4, 2])*1.2*0.5, pull(df_gauge[4,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[4, 2])*1.2*0.75, pull(df_gauge[4,
2])*1.2), color = "gray")),
              threshold = list(line = list(color = 'white'),
              value = pull(df_gauge[4, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 미국 게이지 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[5,
1]),
            domain = list(row = 2, column = 2), value = pull(df_gauge[5, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[5, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[5, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[5, 2])*1.2*0.5, pull(df_gauge[5,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[5, 2])*1.2*0.75, pull(df_gauge[5,
2])*1.2), color = "gray")),
              threshold = list(line = list(color = 'white'),
              value = pull(df_gauge[5, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))
```

```
fig_gauge |> layout(grid=list(rows=3, columns=3),
                    margin = margins_R,
                    title = '10 만명당 사망자수(최근 공식발표 기준)')
```

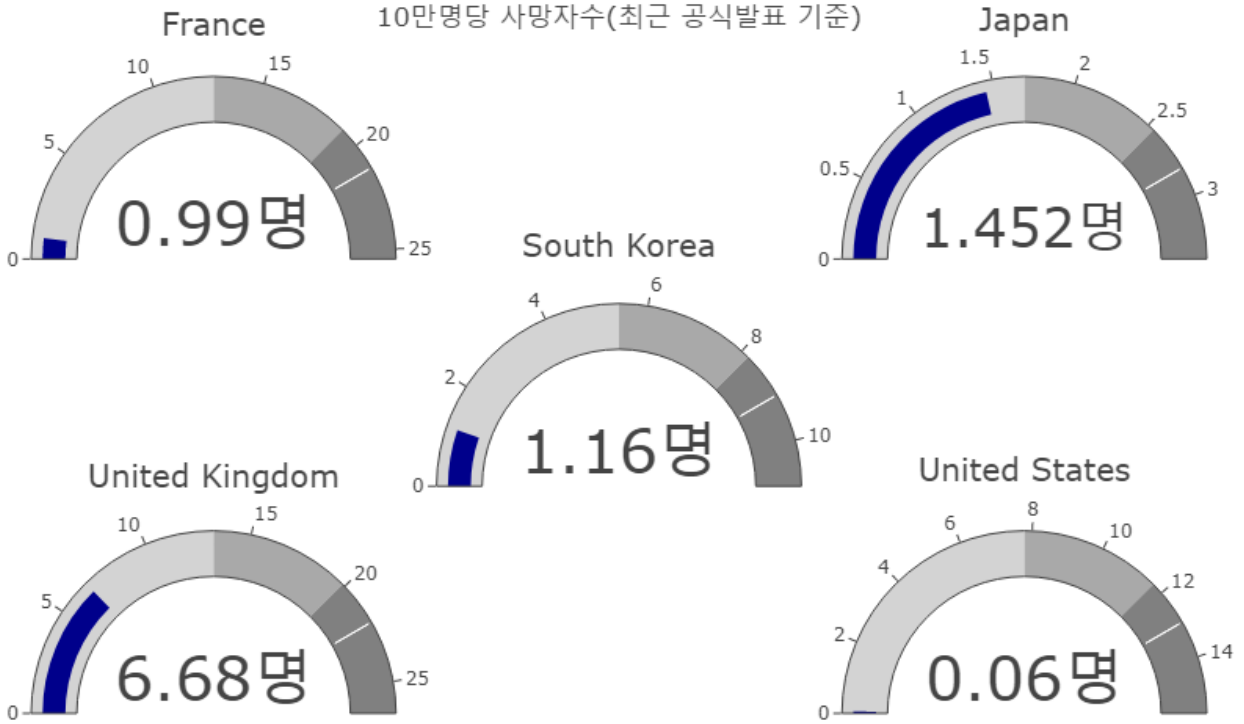
- python

python 에서 게이지 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하고, `add_trace()`에 `plotly.graph_objects.Indicator()`를 사용한다. 게이지 인디케이터는 'mode' 속성을 "gauge"로 설정하고 'gauge' 속성을 설정하여 만들 수 있다.

```
## 게이지 indicator 트레이스를 위한 데이터 전처리
deaths_per_million_in_lateast = total_deaths_5_nations_by_day[total_deaths_5_nat
ions_by_day['new_deaths_per_million'].isna() == False]
deaths_per_million_in_lateast = pd.merge(deaths_per_million_in_lateast.groupby('
location')['date'].max(), deaths_per_million_in_lateast, on = ("location", 'date
'))[['iso_code', 'location', 'date', 'new_deaths_per_million']]
df_gauge = pd.merge(deaths_per_million_in_lateast, total_deaths_5_nations_by_da
y.groupby('location')['new_deaths_per_million'].max().reset_index(), on = 'locat
ion').sort_values('location')
df_gauge.columns = ('iso_code', 'location', 'date', '최근사망자', '최대사망자')

fig = go.Figure()

## 한국 게이지 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[2, 1],
            domain = dict(row = 1, column = 1), value = df_gauge.iloc[2, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[2, 4]*1.2)),
              steps = [
                dict(range = (0, (df_gauge.iloc[2, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[2, 4])*1.2*0.5, (df_gauge.iloc[2,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[2, 4])*1.2*0.75, (df_gauge.iloc[2,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'),
                value = df_gauge.iloc[2, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 프랑스 게이지 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[0, 1],
            domain = dict(row = 0, column = 0), value = df_gauge.iloc[0, 3],
```

```
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[0, 4]*1.2)),
              steps = [
                dict(range = (0, (df_gauge.iloc[0, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[0, 4])*1.2*0.5, (df_gauge.iloc[0,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[0, 4])*1.2*0.75, (df_gauge.iloc[0,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'),
                value = df_gauge.iloc[0, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 일본 게이지 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[1, 1],
            domain = dict(row = 0, column = 2), value = df_gauge.iloc[1, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[1, 4]*1.2)),
              steps = [
                dict(range = (0, (df_gauge.iloc[1, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[1, 4])*1.2*0.5, (df_gauge.iloc[1,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[1, 4])*1.2*0.75, (df_gauge.iloc[1,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'),
                value = df_gauge.iloc[1, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 영국 게이지 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[3, 1],
            domain = dict(row = 2, column = 0), value = df_gauge.iloc[3, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[3, 4]*1.2)),
              steps = [
                dict(range = (0, (df_gauge.iloc[3, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[3, 4])*1.2*0.5, (df_gauge.iloc[3,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[3, 4])*1.2*0.75, (df_gauge.iloc[3,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'),
```

```
                value = df_gauge.iloc[3, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 미국 게이지 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[4, 1],
            domain = dict(row = 2, column = 2), value = df_gauge.iloc[4, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[4, 4]*1.2)),
              steps = [
                dict(range = (0, (df_gauge.iloc[4, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[4, 4])*1.2*0.5, (df_gauge.iloc[4,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[4, 4])*1.2*0.75, (df_gauge.iloc[4,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'),
                value = df_gauge.iloc[4, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

fig.update_layout(grid=dict(rows=3, columns=3),
              title = dict(text = '10 만명당 사망자수(최근 공식발표 기준)', x = 0.
5))

fig.show()
```



실행결과 VI-3. R 의 게이지 인디테이터

1.4. 불릿(bullet) 인디케이터

불릿 인디케이터는 막대 그래프를 변형하여 특정 값의 현재 상태를 표시하는 시각화의 방법이다. 온도계나 진행률의 표시에 사용되는 방식을 차용하여 만드는 불릿 인디케이터는 둥글게 표시되는 게이지 인디케이터의 대용으로 대시보드에서 많이 사용된다. 불릿 인디케이터는 단계를 나타내는 질적 단계(step)와 양적 막대(value), 임계치의 표시의 세 가지 요소로 구성한다. 단계는 일반적으로 배열의 형태로 단계 값들을 지정하여 정의하고, 막대는 표시할 수치의 현재 값을 나타내고, 임계값은 수치의 목표치나 한계치에 대한 목표 지점을 표시함으로서 나타낸다.

불릿 인디케이터는 indicator 트레이스를 사용하여 만들수 있고 게이지 인디케이터의 속성

다음의 코드는 불릿 인디케이터를 만드는 R과 python의 코드이다. 불릿 인디케이터의 'value', 'threshold', 'range', 'steps'는 앞선 게이지 인디케이터와 동일하게 설정하였다.

- R

R 에서 불릿 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하는데, `add_trace(type = 'indicator', ...)`를 사용한다. 불릿 인디케이터는 'mode' 속성을

“gauge”로, ‘gauge’의 ’shape’ 속성을 “bullet”으로 설정하고 ‘gauge’ 속성을 설정하여 만들 수 있다.

```
## 한국 불릿 인디케이터 생성
fig_gauge <- df_gauge |> plot_ly() |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[3,
1]),
            domain = list(x = c(0.3,0.8), y = c(0.82, 0.9)),
            value = pull(df_gauge[3, 5]),
            gauge = list(axis = list(
              range = list(NULL, pull(df_gauge[3, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[3, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[3, 2])*1.2*0.5, pull(df_gauge[3,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[3, 2])*1.2*0.75, pull(df_gauge[3,
2])*1.2), color = "gray")),
              shape = "bullet",
              threshold = list(
                line = list(color = 'white'), value = pull(df_gauge[3, 2])),
              bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 프랑스 불릿 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[1,
1]),
            domain = list(x = c(0.3,0.8), y = c(0.62, 0.7)),
            value = pull(df_gauge[1, 5]),
            gauge = list(axis = list(
              shape = "bullet",
              range = list(NULL, pull(df_gauge[1, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[1, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[1, 2])*1.2*0.5, pull(df_gauge[1,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[1, 2])*1.2*0.75, pull(df_gauge[1,
2])*1.2), color = "gray")),
              shape = "bullet",
              threshold = list(
              line = list(color = 'white'), value = pull(df_gauge[1, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 일본 불릿 인디케이터 생성
```

```r
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[2,
1]),
            domain = list(x = c(0.3,0.8), y = c(0.42, 0.5)),
            value = pull(df_gauge[2, 5]),
            gauge = list(axis = list(
              shape = "bullet",
              range = list(NULL, pull(df_gauge[2, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[2, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[2, 2])*1.2*0.5, pull(df_gauge[2,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[2, 2])*1.2*0.75, pull(df_gauge[2,
2])*1.2), color = "gray")),
              shape = "bullet",
              threshold = list(
              line = list(color = 'white'), value = pull(df_gauge[2, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 영국 불릿 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[4,
1]),
            domain = list(x = c(0.3,0.8), y = c(0.22, 0.3)),
            value = pull(df_gauge[4, 5]),
            gauge = list(axis = list(
              shape = "bullet",
              range = list(NULL, pull(df_gauge[4, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[4, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[4, 2])*1.2*0.5, pull(df_gauge[4,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[4, 2])*1.2*0.75, pull(df_gauge[4,
2])*1.2), color = "gray")),
              shape = "bullet",
              threshold = list(
              line = list(color = 'white'), value = pull(df_gauge[4, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

## 미국 불릿 인디케이터 생성
fig_gauge <- fig_gauge |>
  add_trace(type = 'indicator', mode = "gauge+number", title = pull(df_gauge[5,
1]),
            domain = list(x = c(0.3,0.8), y = c(0.02, 0.1)),
            value = pull(df_gauge[5, 5]),
```

```
            gauge = list(axis = list(
              shape = "bullet",
              range = list(NULL, pull(df_gauge[5, 2])*1.2)),
              steps = list(
                list(range = c(0, pull(df_gauge[5, 2])*1.2*0.5), color = "lightg
ray"),
                list(range = c(pull(df_gauge[5, 2])*1.2*0.5, pull(df_gauge[5,
2])*1.2*0.75), color = "darkgray"),
                list(range = c(pull(df_gauge[5, 2])*1.2*0.75, pull(df_gauge[5,
2])*1.2), color = "gray")),
              shape = "bullet",
              threshold = list(
              line = list(color = 'white'), value = pull(df_gauge[5, 2])),
            bar = list(color = "darkblue")),
            number = list(suffix = '명'))

fig_gauge |> layout(margin = margins_R,
                    title = '10 만명당 사망자수(최근 공식발표 기준)')
```

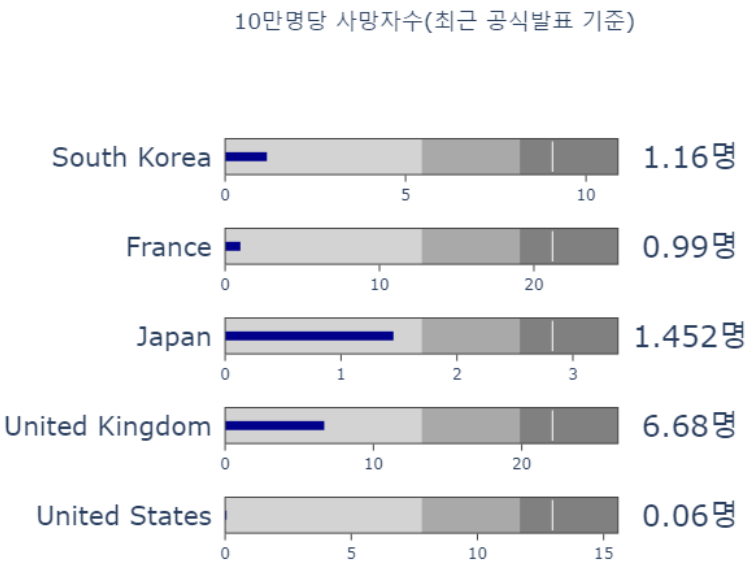
- python

python 에서 불릿 인디케이터를 만들기 위해서는 indicator 트레이스를 사용하고, `add_trace()`에 `plotly.graph_objects.Indicator()`를 사용한다. 불릿 인디케이터는 'mode' 속성을 "gauge"로, 'gauge'의 'shape' 속성을 "bullet"로 설정하고 'gauge' 속성을 설정하여 만들 수 있다.

```
fig = go.Figure()

## 한국 불릿 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[2, 1],
            domain = dict(x = (0.3,0.8), y = (0.82, 0.9)),
            value = df_gauge.iloc[2, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[2, 4]*1.2)),
              shape = "bullet",
              steps = [
                dict(range = (0, (df_gauge.iloc[2, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[2, 4])*1.2*0.5, (df_gauge.iloc[2,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[2, 4])*1.2*0.75, (df_gauge.iloc[2,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'), value = df_gauge.iloc[2, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))
```

```
## 프랑스 불릿 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[0, 1],
            domain = dict(x = (0.3,0.8), y = (0.62, 0.7)),
            value = df_gauge.iloc[0, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[0, 4]*1.2)),
              shape = "bullet",
              steps = [
                dict(range = (0, (df_gauge.iloc[0, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[0, 4])*1.2*0.5, (df_gauge.iloc[0,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[0, 4])*1.2*0.75, (df_gauge.iloc[0,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'), value = df_gauge.iloc[0, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 일본 불릿 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[1, 1],
            domain = dict(x = (0.3,0.8), y = (0.42, 0.5)),
            value = df_gauge.iloc[1, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[1, 4]*1.2)),
              shape = "bullet",
              steps = [
                dict(range = (0, (df_gauge.iloc[1, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[1, 4])*1.2*0.5, (df_gauge.iloc[1,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[1, 4])*1.2*0.75, (df_gauge.iloc[1,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'), value = df_gauge.iloc[1, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 영국 불릿 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[3, 1],
            domain = dict(x = (0.3,0.8), y = (0.22, 0.3)),
            value = df_gauge.iloc[3, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[3, 4]*1.2)),
```

```
              shape = "bullet",
              steps = [
                dict(range = (0, (df_gauge.iloc[3, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[3, 4])*1.2*0.5, (df_gauge.iloc[3,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[3, 4])*1.2*0.75, (df_gauge.iloc[3,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'), value = df_gauge.iloc[3, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

## 미국 불릿 인디케이터 생성
fig.add_trace(go.Indicator(
    type = 'indicator', mode = "gauge+number", title = df_gauge.iloc[4, 1],
            domain = dict(x = (0.3,0.8), y = (0.02, 0.1)),
            value = df_gauge.iloc[4, 3],
            gauge = dict(axis = dict(
              range = (0, df_gauge.iloc[4, 4]*1.2)),
              shape = "bullet",
              steps = [
                dict(range = (0, (df_gauge.iloc[4, 4])*1.2*0.5), color = "lightg
ray"),
                dict(range = ((df_gauge.iloc[4, 4])*1.2*0.5, (df_gauge.iloc[4,
4])*1.2*0.75), color = "darkgray"),
                dict(range = ((df_gauge.iloc[4, 4])*1.2*0.75, (df_gauge.iloc[4,
4])*1.2), color = "gray")],
              threshold = dict(
                line = dict(color = 'white'), value = df_gauge.iloc[4, 4]),
              bar = dict(color = "darkblue")),
            number = dict(suffix = '명')))

fig.update_layout(grid=dict(rows=3, columns=3),
              title = dict(text = '10 만명당 사망자수(최근 공식발표 기준)', x = 0.
5))

fig.show()
```



실행결과 VI-4. python 의 불릿 인디케이터

2. 지도의 시각화

지도의 시각화는 ~~plotly 에서 쉽게 할수 있는 매우 강력한 시각화 방법이다.~~ 사실 지도의 시각화는 엑셀이나 파워포인트, 포토샵이나 일러스트 같은 대중적으로 많이 사용되는 툴에서도 가능하겠지만 매우 번거로운 작업임에 틀림없다. 또 단순한 지도의 시각화라면 다소의 시간과 많은 정신적 스트레스를 감소하고라도 할 수 있겠지만 만약 200 개가 넘는 시군구 지역별 ~~시각화를~~ 해야한다면 이는 단순히 시간과 정신적 스트레스과 바꿀수 있는 작업이 아닐 것이다. 이때는 전문가를 찾아 의뢰하는 것이 해결책이겠지만 항상 전문가들은 바쁘기때문에 시간이 많이 소요되고 비싸다. ~~plolty~~ 는 지도의 시각화에 생각보다 많은 기능이 지원된다.

지도를 시각화하기 위해서는 먼저 몇가지 알아야하는 개념이 있다.

2.1. 지도 시각화의 기초 개념

2.1.1. 지형 데이터

지형 데이터는 지도를 표현하기 위한 지형의 경계선에 대한 폴리곤 정보를 가지고 있는 데이터를 말한다. 지형 데이터는 다음의 두 가지 형태의 데이터 파일로 제공된다.

- shape 파일

Shape 파일은 대부분의 GIS(Geographical Information System)에서 사용되는 지형 벡터 데이터 파일 포맷이다. 확장자를 'shp'로 설정하는데 선, 점, 다각형(Polygon)으로 지형을 벡터의 형태로 표현하는 텍스트 파일이다. 하지만 Shape 파일은 보통'*.shp', '*.shx', '*.dbf', '*.kml', '*.prj' 등의 파일이 한 세트처럼 제공되는 것이 일반적이다. 그렇다고 이 네개의 파일을 모두 쓰는 것은 아니고 사용하는 응용에 따라 선택하여 사용할 수 있다.

- geojson 파일

GeoJSON 은 위치정보를 갖는 점을 기반으로 체계적으로 지형을 표현하기 위해 설계된 개방형 공개 표준 형식이다. 지형 정보를 자바스그립트 오브젝트 노테이션(JAVA Script Object Notation, JSON)을 활용하여 사용하는 파일 포맷으로 OpenLayers, Leaflet, MapServer, Geoforge 소프트웨어, GeoServer, GeoDjango, GDAL, Safe Software FME 등 많은 매핑 및 GIS 소프트웨어 패키지에서 지원하고있다[1]

geojson 파일의 확장자는 '*.json'으로 붙는다. geojson 파일은 앞서 설명한 Shape 파일에서부터 변환하여 생성할 수 있다.

geojson 파일은 Shape 파일에 비해 파일 사이즈가 작고 처리속도도 Shape 파일보다 매우 빠르다는 장점이 있다.

2.1.2. 지도 투영법

지구는 3 차원의 구형태이지만 우리가 보는 지도는 2 차원의 평면이다. 따라서 3 차원을 2 차원에 표현해야 하기 때문에 이 표현의 방법이 여러가지인데 이를 투영법(Projection)이라고 한다. plotly 에서는 이 투영법에 따라 지도가 화면에 표현되는

---

[1] https://ko.wikipedia.org/wiki/GeoJSON

평태가 달라진다. plolty 에서 지원하는 투영법은 정각도법(equirectangular), 원통도법(mercator), 정사도법(orthographic) 등 총 22 개의 투영법을 지원한다.

2.1.3. 좌표계의 종류

일반적으로 지도의 좌표라고 하면 북위, 남위와 같이 적도를 기준으로 북쪽으로 얼마나 많이 떨어졌는지, 남쪽으로 얼마나 많이 떨어졌는지를 표시하는 위도(latitude)와 동경, 서경과 같이 영국 그리니치 천문대를 기준으로 동쪽으로의 위치, 서쪽으로의 위치를 표시하는 경도(longitude)를 생각한다. 이를 좌표계라고 하는데 이 좌표계도 몇 가지 종류가 있다. 이 좌표계가 어떻게 설정되어 있는가에 따라 같은 위도와 경도의 위치도 달라질 수 있다.

- WGS84(EPSG:4326): 미국에서 군사용으로 GPS 시스템를 이용하면서 만든 좌표계(경도와 위도)
- Bessel 1841(EPSG:4004): 3 차원 원형 지구를 2 차원으로 투영한 결과에 대한 좌표계로 우리나라에서 많이 사용되는 좌표계
- GRS80 UTM-K(EPSG:5179): 한반도 전체를 하나의 좌표계로, 전국을경도 127.5 를 기준으로 설계한 지도로 네이버에서 사용
- GRS80 중부원점(EPSG:5181): 과거 지리원 좌표계를 타원체로 수정한 좌표계. Kakao, 공공데이터포탈에서 사용
- Web mercator projection(Pseudo-Mercator, EPSG:3857) : WGS84 타원체의 장반경을 반지름으로 하는 좌표계로 구글, 빙, 야후 등 에서 사용
- Albers projection(EPSG:9822) : 미국 지질 조사국에서 사용하는 좌표계

2.2. 지도의 데이터 표현

plotly 는 지도를 그리기 위해 사용하는 패키지가 아니다. 즉 plotly 에서의 지도의 사용은 지역적 데이터를 표현하기 지도를 사용하여 시각화하기 위해 사용하는 것이다. 그렇다면 데이터를 어떻게 지도위에 표현할 것인가가 가장 중요하다. 지도위에 데이터를 표현하는 방법은 지역의 색을 사용하여 각각의 지역의 특성을 시각화하는 단계 구분도(Choropleth Map)과 지도의 특정 위치에 점을 표기하는 방법(Scatter Map)이 많이 사용된다.

2.2.1. 지도 트레이스

plotly 에서 지도를 그리기 위해서는 앞서 설명한 두 가지 지형 데이터 타입(shape, geojson)을 사용하는 방법과 인터넷에서 제공하는 지도를 사용하여 그리는 두 가지 방법을 제공한다.

이 두 가지 기본적 지도 생성 방법을 사용하여 다음의 다섯 가지 지도 관련 트레이스를 사용하여 목적에 맞는 지도를 만들 수 있다.

- scattergeo 트레이스 : geojson 지형 정보 위에 위도와 경도로 표현된 scatter 트레이스의 점이나 선을 표시하는 트레이스
- choropleth 트레이스 : geojson 지형 정보의 지형 다각형(polygon)에 'z'로 매핑되는 내부 색을 통해 데이터를 표현하는 트레이스
- scattermapbox 트레이스 : mapbox.com 에서 제공하는 맵위에 위도와 경도로 표현된 scatter 트레이스의 점이나 선을 표시하는 트레이스
- choroplethmapbox 트레이스 : mapbox.com 에서 제공하는 맵위에 지형 다각형(polygon)에 'z'로 매핑되는 내부 색을 통해 데이터를 표현하는 트레이스
- densitymapbox 트레이스 : 색상을 사용하여 위도, 경도 좌표와 'z'에 매핑된 값에 따른 밀도 커널 데이터를 표현하는 트레이스

plotly 의 지도 트레이스를 보면 scattergeo 트레이스와 choropleth 트레이스는 geoJSON 이나 shape 파일의 외곽선 지형 기반 지도(plotly geo map)를 사용하는 트레이스이고 scattermapbox, choroplethmapbox, densitymapbox 트레이스는 mapbox.com 에서 제공하는 타일 기반 지도를 사용하는 트레이스이다.

지금까지 대부분의 트레이스는 R 과 python 이 거의 유사한 형태로 사용이 되었다. 하지만 choropleth 트레이스는 R 과 python 의 방법상 차이가 크다. 그래서 R 과 python 을 따로 구분하여 설명하도록 하는데, 단계 구분도(choropleth)와 맵박스 산점도(scattermapbox)를 그리는 방법을 설명한다.

위의 두 가지 지도를 그리기 위해 두 가지 추가적인 데이터를 사용한다. 첫 번째 데이터는 전국의 대학 충원률 데이터이다.[2] 두 번째 데이터는 서울의 일부 대학의 위도, 경도 데이터이다.[3]

2.2.2. python 의 지도 그리기

2.2.2.1. scattergeo 트레이스

scattergeo 트레이스는 plolty 에서 지원하는 지형 외곽선 기반 지도를 기반으로 그 위에 다양한 데이터를 표현하는 지도 트레이스이다. plotly 에서 지원하는 지형 외곽선 기반의 지도(plotly geo map)는 Natual Earth(https://www.naturalearthdata.com/)에서에서){.uri} 제공하는 'physical(물리적)', 'cultural(문화적)'인 기본 지도를 지원하고 있다. 그래서 트레이스 관련 함수를 호출만으로도 해안선과 육지가 나타나는 세계 지도와 같은 기반 맵을 만들 수 있다. 또 이 기반 맵의 선이나 영역 설정등에 대한 다양한 속성을 설정할 수도 있고 해안선, 육지, 호수, 강 등을 표시할 수도 있다.

다음은 scattergeo 트레이스에서 사용하는 주요 속성이다.

속성	속성 설명	속성값(R 속성값)
lat	위도 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
geojson	이 트레이스에 관련한 GeoJSON 데이터 설정	수치나 좌표상의 문자열
featureidkey	'locations' 배열에 포함된 아이템과 매칭하기 위한 GeoJSON features의 키 설정	문자열
locations	location ID나 names에 대한 좌표 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
lon	경도 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
geo	서브플롯에서 참조할 지형 좌표의 레퍼런스 값 설정	subplotid
fill	색상을 채울 영역의 설정	("none" | "toself")
fillcolor	영역을 채울 색상의 설정	color
locationmode	지도 위에 표현되는 지역을 위한 locations와 매칭하는 지역 설정	("ISO-3" | "USA-states" | "country names" | "geojson-id")

---

[2] 해당 파일은 교육통계 홈페이지(https://kess.kedi.re.kr)에서에서){.uri} 다운로드 받을 수 있으며 필자의 블로그에서도 다운받을 수 있다.

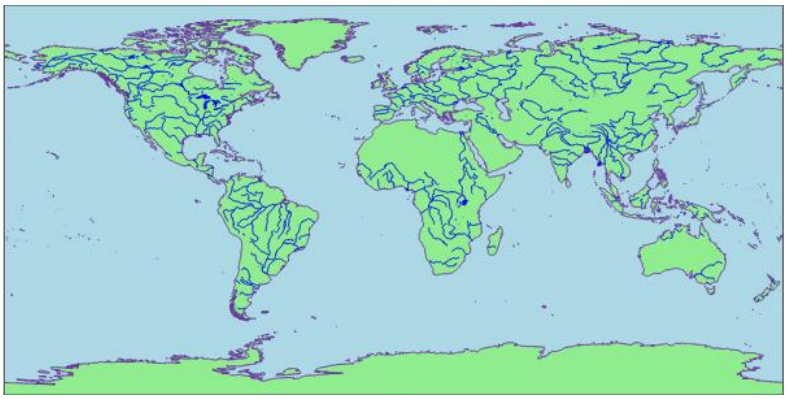
[3] 해당 파일은 필자의 블로그에서도 다운받을 수 있다.

지형 기반 지도의 레이아웃을 설정하는데 사용하는 속성은 'geo' 속성이다. 다음은 'geo'의 주요 속성이다.

속성				속성 설명	속성값(R 속성값)
geo	bgcolor			지도의 배경색 설정	색상값
	center	lat		지도의 중심 위도 설정	수치
		lon		지도의 중심 경도 설정	수치
	coastlinecolor			해안선 색 설정	색상값
	coastlinewidth			해안선 두께 설정	0이상의 수치
	countrycolor			국경선 색 설정	색상값
	countrywidth			국경선 두께 설정	0이상의 수치
	domain	column		플로팅 영역의 그리드에서 지도의 표시 위치 열 설정	정수
		row		플로팅 영역의 그리드에서 지도의 표시 위치 행 설정	정수
		x		플롯팅 영역에서 지도 표시 X범위 설정	리스트
		y		플롯팅 영역에서 지도 표시 Y범위 설정	리스트
	fitbounds			지도의 영역이 트레이스 데이터에 의해 자동설정되는지 여부 설정	(False | "locations" | "geojson")
	framecolor			지도 프레임 색상 설정	색상값
	framewidth			지도 프레임 두께 설정	0이상의 수치
	lakecolor			호수의 색상 설정	색상값
	landcolor			육지의 색상 설정	색상값
	lataxis	dtick		위도 축의 눈금 간격 설정	수치
		gridcolor		위도 축의 눈금자 색상 설정	색상값
		gridwidth		위도 축의 눈금자 두께 설정	0이상의 수치
		range		위도 축의 범위 설정	리스트
		showgrid		위도 축의 표시여부 설정	논리값
		tick0		위도 축의 시작 단위 설정	수치
	lataxis	dtick		경도 축의 눈금 간격 설정	수치
		gridcolor		경도 축의 눈금자 색상 설정	색상값
		gridwidth		경도 축의 눈금자 두께 설정	0이상의 수치
		range		경도 축의 범위 설정	리스트
		showgrid		경도 축의 표시여부 설정	논리값
		tick0		경도 축의 시작 단위 설정	수치
	oceancolor			바다 색상 설정	색상값
	projection	distance		위성 투영법에서 지구와 위성과의 거리 설정	1.001이상의 수치
		parallels		원뿔 투영법에서 위선 설정	리스트
		rotation	lat	자오선을 따른 회전 위도 설정	수치
			lon	위선을 따른 회전 경도 설정	수치
			roll	지도의 롤링	수치
		scale		지도의 줌인 또는 줌 아웃의 스케일 설정	0이상의 수치
		tilt		위성 투영법에서 경사 각도 설정	수치

다음은 scattergeo 트레이스로 물리적 지도를 그리는 python 코드이다.

```
## scattergeo 트레이스 생성
fig = go.Figure(go.Scattergeo())

## geo 속성 설정
fig.update_layout(geo = dict(
  resolution=50,
  showcoastlines=True, coastlinecolor='RebeccaPurple',
  showland=True, landcolor='LightGreen',
  showocean=True, oceancolor='LightBlue',
  showlakes=True, lakecolor='Blue',
  showrivers=True, rivercolor='Blue'),
  height=300, margin={'r':0,'t':0,'l':0,'b':0})

fig.show()
```

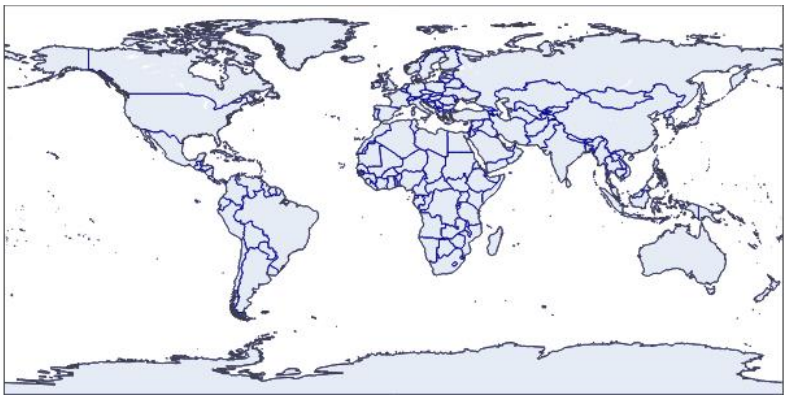


실행결과 VI-5. python 의 scattergeo 트레이스 물리적 세계지도

반면 문화(cultural) 기반 지도는 물리 기반 지도에 국가별 경계선과 국가내 지역 경계선을 포함한다.

다음은 문화적 지도를 그리는 python 코드이다.

```
## scattergeo 트레이스 생성
fig = go.Figure(go.Scattergeo())

## geo 속성 설정
fig.update_layout(geo = dict(
  resolution=50,
  showcountries=True, countrycolor='darkBlue'),
  height=300, margin={'r':0,'t':0,'l':0,'b':0})

fig.show()
```



실행결과 VI-6. python 의 scattergeo 트레이스 문화적 세계지도

2.2.2.2. choropleth 트레이스

단계 구분도(choropleth map)는 지역적으로 구분된 데이터를 지도에 표시하는 방법으로 가장 많이 사용되는 것은 색으로 지도의 지형에 데이터를 표시하는 방법이다. 각각의 다각형(Polygon)으로 구성되어 있는 지역의 내부 색을 데이터에 따라 다르게 함으로서 지역간 데이터의 차이를 시각화 하는 방법이다. 결국 색이 칠해진 다각형을 배치함으로 데이터를 표현하는 것이다.

python 에서 choropleth 트레이스를 추가하지 위해서는 `add_trace()`에 `plotly.graph_objects.Choropleth()`를 사용하여 추가한다.

다음은 choropleth 트레이스에서 사용되는 주요 속성들이다.

속성			속성 설명	속성값(R 속성값)
z			색상 값의 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
geojson			이 트레이스에 관련한 GeoJSON 데이터 설정	수치나 좌표상의 문자열
featureidkey			'locations' 배열에 포함된 아이템과 매칭하기 위한 GeoJSON features의 키 설정	문자열
locations			location ID나 names에 대한 좌표 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
text			각각의 위치에 표현될 텍스트 설정	문자열이나 문자열 배열
geo			서브플롯에서 참조할 지형 좌표의 레퍼런스 값 설정	subplotid
marker	line	color	지형 마커의 외곽선 색 설정	색상값이나 색상값 배열
		width	지형 마커의 외곽선 두께 설정	0이상의 수치나 수치 배열
	opacity		지형 마커의 투명도 설정	0부터 1사이의 수치나 수치 배열
reversescale			색상 값 스케일을 역변환	논리값
locationmode			지도 위에 표현되는 지역을 위한 locations와 매칭하는 지역 설정	("ISO-3" | "USA-states" | "country names" | "geojson-id")

choropleth 트레이스에서 사용하는 layout 속성은 scattergeo 트레이스에서 설명한 'geo' 속성을 사용한다.

단계 구분도를 그리기 위해서는 크게 두 가지 데이터가 필요하다.

첫 번째는 지형 데이터이다. plotly 에서는 이 지형 데이터를 geoJSON 으로 사용자가 직접 불러들인 지형 데이터를 사용할 수도 있고 plotly 에서 지원하는 지형데이터를 사용할 수도 있다. 하지만 plotly 에서 지원하는 지형 데이터는 앞에서 그려본 바와 같이 세계지도, 대륙별 지도, 미국 지도 정도의 의미가 있기 때문에 우리나라 지형에 사용하는데에는 적절치 않다. 우리나라 지형에 사용하기 위해서는 geoJSON 데이터를 사용해야 하는데 geoJSON 데이터는 각각의 feature 를 구분하기 위한 'id'를 가져야 하고 각 feature 의 특성은 'properties'로 인식되어야 한다.

두 번째는 지도위에 색상으로 표시할 데이터이다. 이 데이터는 지형 데이터의 feature 구분자와 매핑되는 필드를 가지고 있어야 한다.

이렇게 두 가지 데이터가 준비되면 geoJSON 데이터는 choropleth 트레이스의 'geojson' 속성에 설정된다. 또 두 개의 데이터를 연결하는 id 중 지형 데이터 쪽 id 는 'featureid' 속성으로 표시 데이터 쪽 id 는 'locations'에 설정함으로써 지형 데이터와 표시 데이터가 연결된다. 다음으로 지도위에 색상으로 단계를 구분하기 위한 데이터는 'z' 속성으로 설정한다.

여기서는 전국 시도별 대학 신입생 충원율의 단계 구분도를 그려보겠다.

plotly 에서는 미국 지도를 바로 지원하지만 한국 지도를 지원하지 않는다. 따라서 제일 먼저 한국 지형을 geoJSON 으로 불러 읽어들이고 이를 'geojson'으로 설정한다. 우리나라의 geoJSON 파일은 인터넷에서 쉽게 구할 수 있다. 파일을 다운 받을때는 시도 단위의 데이터인지 시군구 단위의 데이터인지를 잘 구분하여 다운해야 한다. geoJSON 파일의 데이터를 읽어오기 위해서 json 라이브러리를 임포트하고 `json.load()`를 사용하여 geoJSON 데이터를 불러온다.

다음으로 전국 대학 신입생 충원율 데이터를 불러오고 이 데이터 중 필요한 데이터 5 개를 필터링하고 열 이름을 적절히 설정한다.

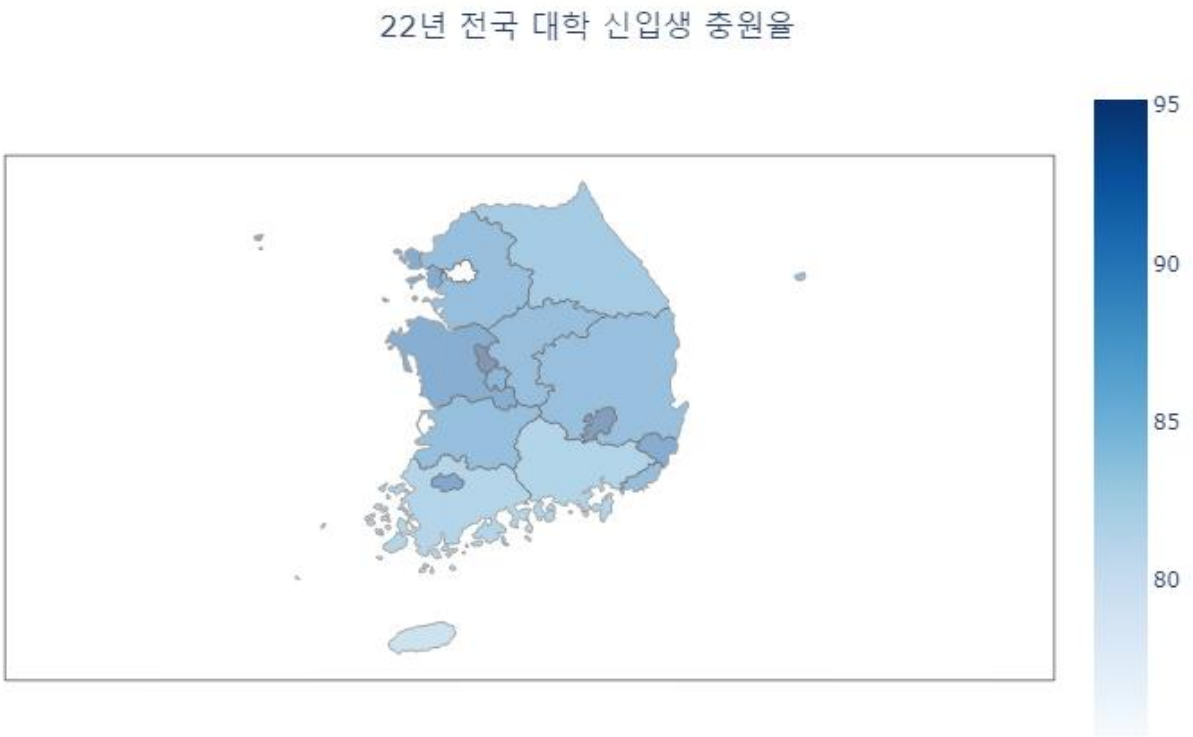
불러들여온 전국 대학 신입생 충원율 데이터에는 지역을 표현하는데 한글 시도명을 사용한다. 반면 geoJSON 데이터에서는 properties 의 CTPRVN_CD 속성으로 정의되는데 시도를 수치로 표현하고 있다. 그래서 이를 매칭시키기 위한 함수를 정의 하였고 이 함수를 사용하여 대학 신입생 충원율 데이터에 매핑된 수치 열을 만들어 주었다.

choropleth 트레이스를 만들기 위한 데이터 전처리가 끝났으면 plotly 를 초기화하고 `plotly.graph_objects.Choropleth()`를 사용한다. 앞에서 불러들어온 geoJSON 데이터를

'geojson' 속성에 설정하고 충원율 데이터와 매칭할 키인 "properties.CTPRVN_CD"를 'featureidkey'에 설정한다. 이번에는 'featureidkey'에 매칭될 충원율 데이터의 열을 'locations'에 설정하고 색상으로 표현해 줄 데이터인 충원율 열을 'z' 속성으로 설정한다. 전체적 색상 스케일 속성인 'colorscale'을 푸른색 팔레트("blues")로 설정하고 지형의 경계선, 컬러바, 표현 텍스트, 호버를 설정하였다.

```
##  json 파일을 읽기 위한 라이브러리 로딩
import json

##  지역값 매핑을 위한 함수 정의
def cat(row):
    key = row['지역']
    value = {'강원' : '42','경기' : '41','경남' : '48','경북' : '47','광주' : '29
','대구' : '27','대전' : '30','부산' : '26','서울' : '11','세종' : '36','울산' : '3
1','인천' : '28','전남' : '46','전북' : '45','제주' : '50','충남' : '44','충북' : '
43'}.get(key)
    return value

## json 파일의 데이터를 불러들임
geometry = json.load(open('D:\\R\\git\\datavisualization\\plotly\\RnPy\\chap6\\T
L_SCCO_CTPRVN.json', encoding='utf-8'))

## 충원율 데이터 불러들임
df_충원율 = pd.read_excel("D:/R/git/datavisualization/plotly/RnPy/chap6/고등 주요
01-시도별 신입생 충원율(2010-2022)_220825y.xlsx",
                          sheet_name = 'Sheet1', skiprows=(6), header = 0)

df_충원율 = df_충원율.iloc[:, 0:5]
df_충원율.columns = ('연도', '지역', '정원내모집인원', '정원내입학생수', '신입생충원
율')
df_충원율['지역코드']= df_충원율.apply(cat, axis=1)
df_충원율 = df_충원율[(df_충원율['지역'] != '전국')&(df_충원율['연도'] == 2022)]

fig = go.Figure()

##  chropleth 트레이스 생성
fig.add_trace(go.Choropleth(
  geojson=geometry,
  featureidkey='properties.CTPRVN_CD',
  locations = df_충원율['지역코드'],
```

```
  z = df_충원율['신입생충원율'],
  colorscale="Blues",
  text = df_충원율['신입생충원율'],
  marker=dict(line=dict(width=1),
  opacity = 0.5),
  colorbar = dict(y = 0.5, yanchor = 'middle')))

fig.update_layout(
  title = dict(text = '22 년 전국 대학 신입생 충원율', x = 0.5),
  ## geo 속성 설정
  geo=dict(
    showframe = True, fitbounds = "locations",
    visible = False, center=dict(lon=126.98, lat=37.56)),
  autosize=False,
  margin = dict(t = 50, b = 25, l = 25, r = 25, pad=4, autoexpand=True))
```



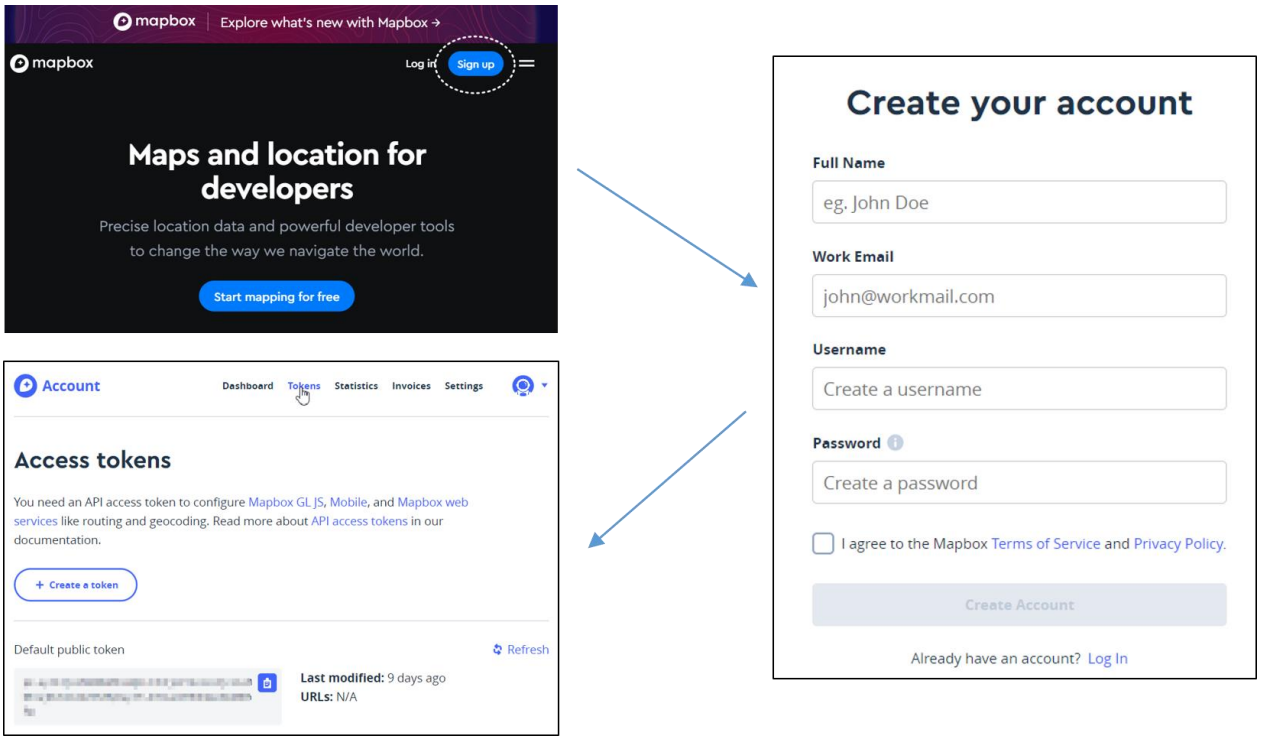
실행결과 VI-7. python 의 단계구분도

2.2.2.3. scattermapbox 트레이스

스캐터 맵은 지도 위에 위도, 경도로 표현된 데이터를 표시하여 그 상대적 위치를 시각화하는 지도이다. 우리가 네이버나 카카오에서 특정 장소를 찾을 때 쓰는 지도가 바로 스캐터 맵이다.

이 스캐터 맵을 맵박스에서 지원하는 지도를 기반으로 만들어 주는 것이 scattermapbox 트레이스이다.

mapbox.com 은 타일 기반의 지도를 제공하는 인터넷 서비스이다. mapbox.com 에서 제공하는 지도를 기반으로 해서 위도와 경도상의 위치에 특정 데이터를 점이나 선, 텍스트를 표시해 지도를 만들 수 있다. 맵박스로 구성한 mapbox.com 의 지도를 사용하기 위해서는 먼저 mapbox.com 에 가입하고 mapbox.com 의 지도를 가져오기 위한 액세스 토큰를 받아야 한다.



python 에서 scattermapbox 트레이스를 추가하지 위해서는 `add_trace()`에 `plotly.graph_objects.Scattermapbox()`를 사용하여 추가한다.

다음은 Scattermapbox 트레이스에서 사용되는 주요 속성들이다.

속성		속성 설명	속성값(R 속성값)
lat		위도 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
lon		경도 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
cluster	color	각각의 클러스터 단위에 따른 색상 설정	색상값이나 색상값 배열
	enabled	클러스터를 사용할지를 설정	논리값
	maxzoom	최대 줌 레벨의 설정	0부터 24사이의 수치
	opacity	마커의 투명도 설정	0부터 1사이의 수치
	size	클러스터 단위에 따른 크기 설정	0이상의 수치
	step	클러스터 단위에 몇 개의 점을 포함할지를 결정	-1이상의 수치
text		지도위에 표시할 위도와 경도 등의 텍스트 설정	문자열이나 문자열 배열
hovertext		지도위에 표시할 위도와 경도 등의 호버 텍스트 설정	문자열이나 문자열 배열
hoverinfo		호버에 표시될 정보의 설정	"lon", "lat", "text", "name" 를 '+'로 연결한 문자열, 또는 "all", "none", "skip"
marker	allowoverlap	마커가 오버랩될지 여부를 설정	논리값
	colorscale	마커의 컬러 스케일 설정	컬러스케일값
	symbol	https://www.mapbox.com/maki-icons/에서 제공하는 심볼 설정	심볼문자열
below		scattermapbox 트레이스의 레이어가 특정 레이어 밑으로 들어가는 경우 밑에 설정되는 레이어 id	레이어 id
fill		색상을 채울 영역의 설정	("none" | "toself")
fillcolor		영역을 채울 색상의 설정	color

Scattermapbox 트레이스에서 'layout'을 설정하기 위해서는 'mapbox' 속성을 사용하는데 다음은 'mapbox'의 주요 속성이다.

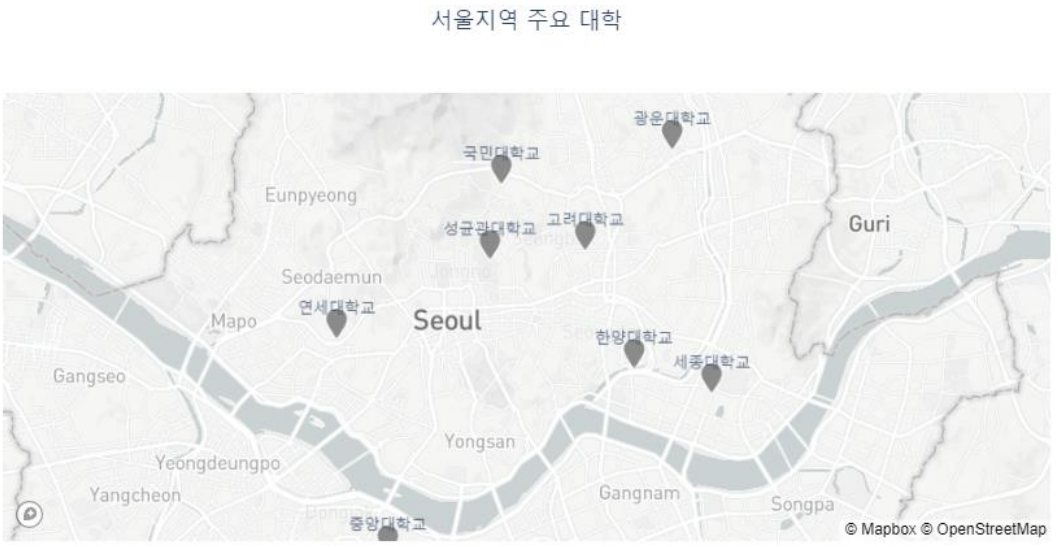
속성				속성 설명	속성값(R 속성값)
mapbox	accesstoken			맵박스 액세스 토큰 설정	문자열
	bearing			북쪽에서 시계 반대 방향으로 지도의 방위 각도 설정	수치
	bounds	east		지도의 최대 경도 설정	수치
		north		지도의 최대 위도 설정	수치
		south		지도의 최소 위도 설정	수치
		west		지도의 최소 경도 설정	수치
	center	lat		지도의 중심 위도 설정	수치
		lon		지도의 중심 경도 설정	수치
	domain	column		플로팅 영역의 그리드에서 지도의 표시 위치 열 설정	정수
		row		플로팅 영역의 그리드에서 지도의 표시 위치 행 설정	정수
		x		플로팅 영역에서 지도 표시 X범위 설정	리스트
		y		플로팅 영역에서 지도 표시 Y범위 설정	리스트
	layers	below		scattermapbox 트레이스의 레이어가 특정 레이어 밑으로 들어가는 경우 밑에 설정되는 레이어 id	레이어 id
		color		외상위 레이어 색상 설정	색상값
		coordinates		왼쪽 위부터 시계 방향 순서로 나열된 이미지 모서리에 대한 경도, 위도 좌표 설정	리스트
		fill	outlinecolor	레이어 외곽선색 설정	색상값
		line	dash	레이어 외곽선 타입 설정	list, numpy array, or Pandas series of numbers, strings, or datetimes. (dataframe column, list, vector)
			width	레이어 외곽선 두께 설정	수치
		maxzoom		최대 줌 레벨 설정	0부터 24사이의 수치
		minzoom		최소 줌 레벨 설정	0부터 24사이의 수치
		opacity		레이어 투명도 설정	0과 1사이의 수치
		source		레이어의 소스 설정	수치나 좌표상의 문자열
		sourceattribution		레이어 소스의 속성 설정	문자열
		sourcelayer		source가 "vector"이고 다중 레이어를 포함한 경우 어떤 레이어를 사용할지 설정	문자열
		sourcetype		레이어의 소스 타입 설정	("geojson" | "vector" | "raster" | "image")
		symbol	icon	https://www.mapbox.com/maki-icons/에서 제공하는 심볼 설정	심볼문자열
			iconsize	아이콘의 크기 설정	수치
			placement	아이콘과 텍스트와의 배치 설정	("point" | "line" | "line-center")
			text	심볼에 표시할 텍스트 설정	문자열
			textfont	color	심볼에 표시할 텍스트 색상 설정	색상값
				family	심볼에 표시할 텍스트 폰트 설정	폰트명
				size	심볼에 표시할 텍스트 크기 설정	수치
			textposition	심볼에 표시할 텍스트 위치 설정	("top left" | "top center" | "top right" | "middle left" | "middle center" | "middle right" | "bottom left" | "bottom center" | "bottom right")
		type		레이어 타입의 설정	("circle" | "line" | "fill" | "symbol" | "raster")
		visible		레이어 표시여부 설정	논리값
	pitch			지도의 경점 각도 설정	수치
	style			트레이스 레이어 아래에 그려지는 기본 지도의 스타일 설정	수치나 좌표상의 문자열
	zoom			지도의 줌 레벨 설정	수치

다음은 맵박스의 서울 지도 위에 서울의 주요 대학의 위치를 설정하는 python 코드이다.

앞서 언급했다시피 맵박스에서 제공되는 지도를 사용하기 위해서는 맵박스 토큰이 필요하다. 이 토큰은 'layout'의 'mapbox'의 'accesstoken'에 설정해야 맵박스 지도가 열린다. 맵박스에서 표시되는 지도는 대부분 'layout'의 'mapbox' 속성으로 설정한다. scattermapbox 트레이스에는 이 맵박스 지도위에 표시될 데이터만을 설정하는데 서울 주요 대학의 위도, 경도, 대학의 이름을 설정하였다.

```
##  대학 위경도 데이터 불러들임
df_univ = pd.read_excel("D:/R/git/datavisualization/plotly/RnPy/chap6/university.xlsx")
```

```
##  scattermapbox 트레이스 추가
fig = go.Figure(go.Scattermapbox(
        lat=df_univ['lat'], lon=df_univ['lon'],
        mode='markers+text',
        marker=dict(symbol ='marker', size=15, color='blue'),
        text=df_univ['학교명'], textposition = 'top center'))

fig.update_layout(title = dict(text = '서울지역 주요 대학', x = 0.5),
    autosize=True, hovermode='closest',
    ##  mapbox 속성 설정
    mapbox=dict(
        accesstoken= mapbox_access_token,
        bearing=0, center=dict(lon=126.98, lat=37.56),
        pitch=0, zoom=10, style="light"))

fig.show()
```



실행결과 VI-8. 맵박스를 사용한 R 의 scatter 지도

2.2.3. R 의 지도 그리기

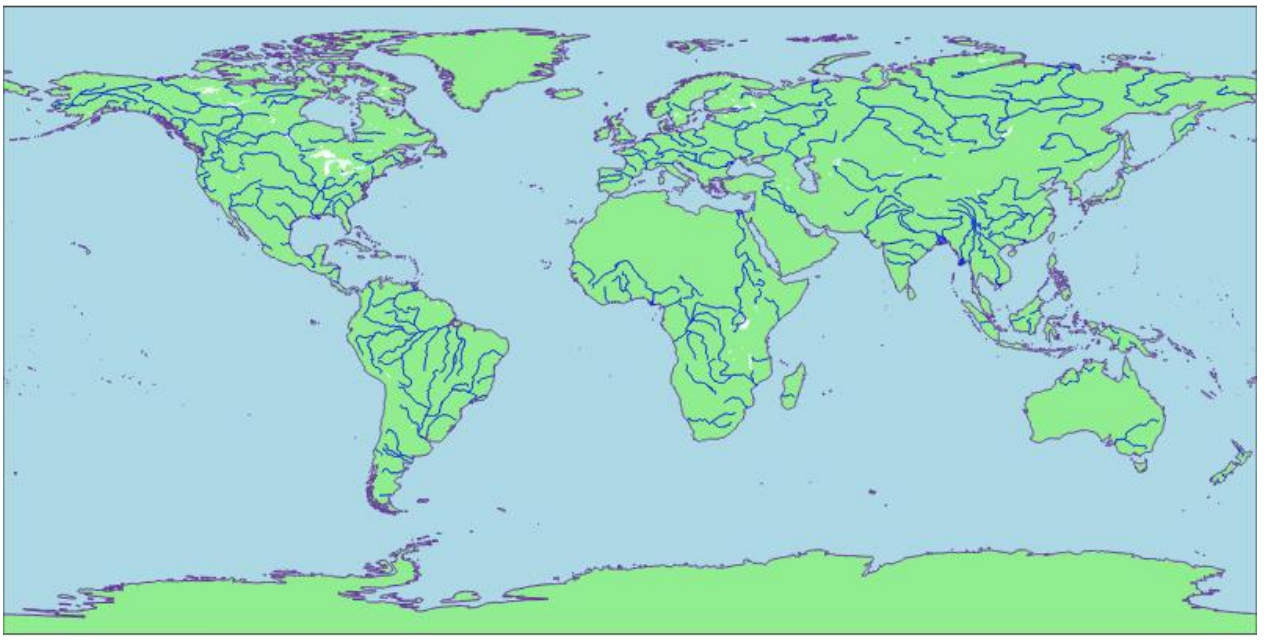
2.2.3.1. scattergeo 트레이스

R 에서도 plotly 에서 지원하는 기본 지도를 scattergeo 트레이스를 사용하여 그릴 수 잇다. scattergeo 트레이스에서 사용하는 속성과 'layout'의 설정을 위한 'geo' 속성은 python 과 동일하다.

다음은 scattergeo 트레이스로 물리적 지도를 그리는 R 코드이다.

- R

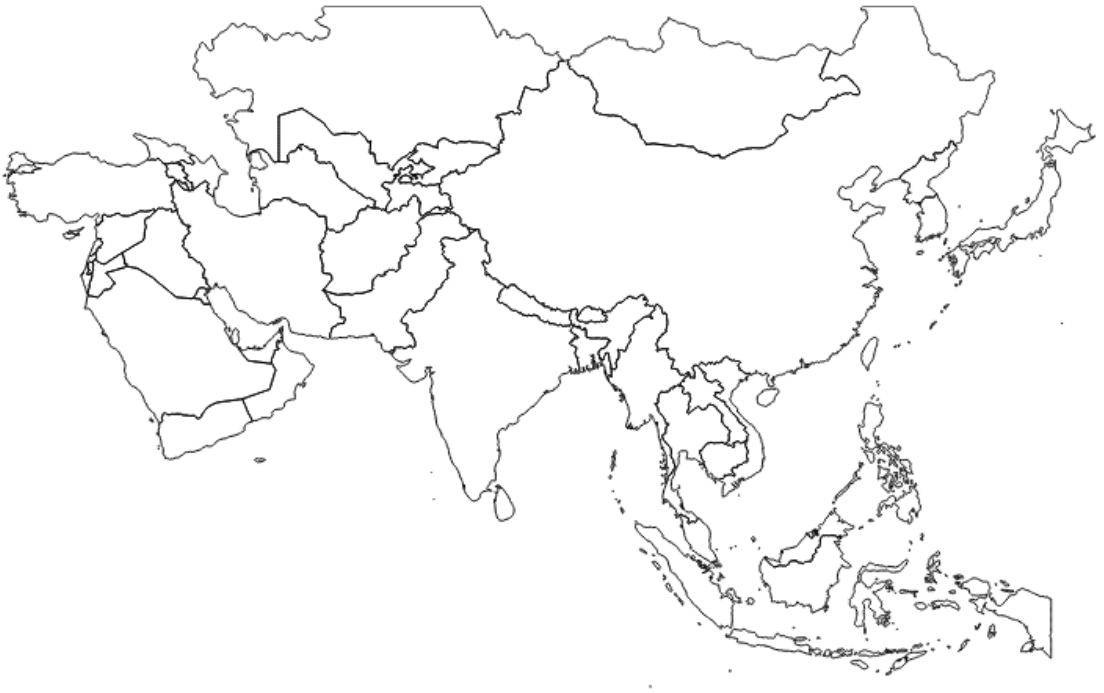
```
plot_ly() |>
  ##   scattergoe 트레이스 생성
  add_trace(type = 'scattergeo') |>
  layout(geo = list(resolution=50,
                    showcoastlines=TRUE, coastlinecolor='RebeccaPurple',
                    showland=TRUE, landcolor='LightGreen',
                    showocean=TRUE, oceancolor='LightBlue',
                    showlakes=TRUE, lakecolor='white',
                    showrivers=TRUE, rivercolor='Blue'),
         margin = list(r = 0, l = 0, t = 0, b = 0))
```



실행결과 VI-9. R 의 scattergeo 트레이스 물리적 세계지도

다음은 scattergeo 트레이스로 아시아 지역의 문화적 지도를 그리는 R 코드이다.

```
plot_ly() |>
  add_trace(type = 'scattergeo') |>
  layout(geo = list(resolution=50, scope = 'asia',
                    showcountries=TRUE, countrycolor="black"),
         margin = list(r = 0, l = 0, t = 0, b = 0))
```



실행결과 VI-10. python 의 scattergeo 트레이스 문화적 아시아 지도

2.2.3.2. 단계 구분도

R 에서 plotly 로 지도를 그리는 것은 지금까지의 plotly 객체를 만드는 방식과 조금 다른 방식으로 설명하도록 하겠다. 물론 python 과 같이 만들수도 있겠지만 이보다는 수월하면서 품질이 놓은 방법이 있기 때문에 이를 위주로 설명하도록 하겠다.

R 에서 지도를 그리기 위해서는 하나 알아두어야 하는 것이 `sf` 클래스 데이터이다. 이는 단지 plotly 에서만 사용하는 클래스가 아닌 R 에서 광범위하게 사용되는 지형 데이터 전용 데이터 클래스이다. `sf` 클래스는 R 의 `sf` 패키지에서 제공하는 클래스 데이터 타입으로 'simple feature'의 준말이다. `sf` 클래스는 지형 데이터를 R 에서 기본적으로 사용되는 data.frame 이나 tibble 형태로 저장하여 R 에서 다루기가 쉽게 사용되는 클래스로 R 을 종합적으로 관장하는 R consortium 에서 관리하는 데이터 타입이다. ~~그 때문에~~ R 에서 가장 쉽게 지형 데이터를

관리하고 처리할 수 있는 데이터 타입이기 때문에 `plolty` 에서도 `sf` 데이터 타입을 지원하도록 설계되어 있다.

그래서 R 에서는 plotly 의 지도 전용 트레이스를 사용하기 보다는 `sf` 클래스를 활용한 R 전용 함수를 사용하는 방법이 지도의 품질상 더 효과적이다. `sf` 클래스는 `plot_ly()`, `plot_geo()`와 `plot_mapbox()`로 사용이 가능하다.

`plot_ly()`는 지금까지 plotly 객체를 초기화하는데 사용했던 바로 그 함수이다. `plot_ly()`에 바인딩되는 데이터프레임이 `sf` 클래스면 `plot_ly()`는 지형 데이터를 처리하도록 동작한다. `plot_ly()`에서 사용하는 주요 매개변수는 다음과 같다.

plot_ly(data = data.frame(), ..., type = NULL, name, color, colors = NULL, alpha = NULL, stroke, strokes = NULL, alpha_stroke = 1, size, sizes = c(10, 100), span, spans = c(1, 20), symbol, symbols = NULL, linetype, linetypes = NULL, split, frame, width = NULL, height = NULL, source = "A")

- data : 시각화에 사용될 지형 데이터(sf 타입)
- color : 내부에 채워지는 색상 매핑으로 사용될 데이터
- colors : color 에 사용될 색상 팔레트
- alpha : 투명도를 설정하는 색상의 알파 채널값
- stroke : 컬러와 유사하지만 색이 채워진 폴리곤에서의 외곽선 색
- split : 다중 트레이스로 구분될 값 설정

`plot_geo()`는 R 의 plotly 에서 지도 객체의 초기화를 위해 사용하는 함수로 `plot_ly()` 대신 사용할 수 있는 함수이다. `plot_geo()`는 plotly 의 주로 shape 파일이나 geoJSON 으로 지도를 그리고 데이터를 사용하여 지도를 그리는데 지도 트레이스 중 scattermap 트레이스로 지도를 그린다. `plot_geo()`의 매개변수는 'offline' 매개변수를 제외하고는 `plot_ly()`의 매개변수와 같다

plot_geo(data = data.frame(), ..., type = NULL, name, color, colors = NULL, alpha = NULL, stroke, strokes = NULL, alpha_stroke = 1, size, sizes = c(10, 100), span, spans = c(1, 20), symbol, symbols = NULL, linetype, linetypes = NULL, split, frame, width = NULL, height = NULL, source = "A", offline = FALSE)

- offline : 인터넷 연결 여부에 관계없이 지도를 볼 수 있도록 지형 데이터를 포함시킬지를 결정

`plot_mapbox()`는 mapbox.com 의 지도를 사용하여 지도 그리는 함수로 앞에서 언급했듯이 mapbox.com 의 API key 가 필요하다. `plot_mapbox()`는 지도 트레이스 중 scattermapbox 트레이스로 지도를 그린다. 사용법은 `plot_ly()`와 동일하다.

다음은 `ggplot2` 의 `geom_sf()`와 `plot_ly()`, `plot_geo()`, `plot_mapbox()`로 그린 우리나라의 지도이다.

실행결과 VI-10. R 전용 지도 생성 함수 사용 결과

그럼 R 에서 단계 구분도(choropleth)를 그리는 방법에 대해 알아본다.

앞서 설명했듯이 단계 구분도에서는 지형 데이터와 ~~지형에 표시될~~ 데이터, 두 가지 데이터가 필요하다.

R 에서 단계 구분도를 그리기 위해 중요한 것은 python 과는 달리 지형 데이터와 표시할 데이터를 가진 데이터프레임을 조인하여 하나의 객체로 만들어야 한다는 것이다. 이를 위해서 지역을 표현하는 지형 데이터의 키와 표현할 데이터의 지역 키를 일치하도록 전처리 해야한다는 것이다.

R 에서도 앞의 python 에서 그렸던 전국 시도별 대학 신입생 충원율의 단계 구분도를 동일하게 그려보겠다.

먼저 지도 생성에 필요한 우리나라 지형 데이터를 가져와야한다. 이에 사용한 우리나라의 지도 데이터는 `raster` 패키지의 `getData()`를 통해 제공되는 한국의 지형 데이터를 사용하였다. `getData()`는 특정 국가의 지형 데이터를 가져오는 함수인데, 'level'에 따라 지도의 지형 레벨을 결정할 수 있는데 한국 지형 데이터의 경우 'level = 1'이면 시도 단위의 지형 데이터를 의미한다. 이 데이터를 `sf` 데이터 타입으로 가져왔다.

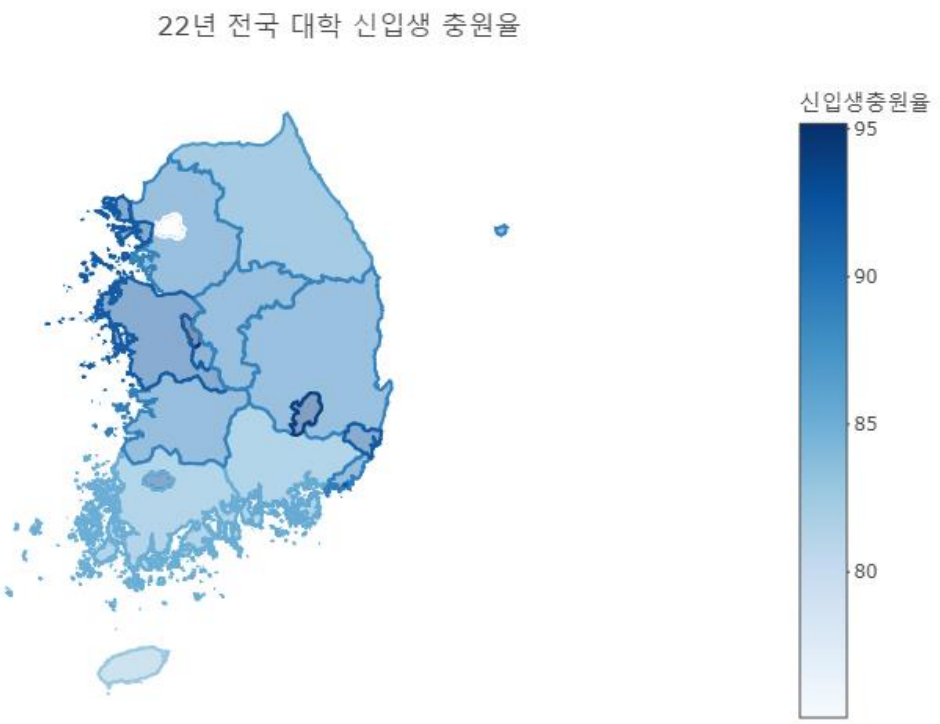
다음으로 지도에 표시할 데이터인 전국의 대학 신입생 충원율 데이터를 가져온다. 이 데이터 중 사용할 5 개의 열만을 선택해서 데이터를 저장하고 열 이름을 설정한다.

전국 대학 충원률 데이터에서는 지역이 한글로 표현되어 있고 지형 데이터에서는 영문 키로 저장되어 있기 때문에 지형 데이터의 키와 맞추기 위해 충원률 데이터의 지역을 지형 데이터의 영문 키로 변환하는 과정을 거쳤다.

이제 충원율이 포함된 `sf` 클래스 데이터프레임을 사용하여 단계 구분도를 그린다. 단계 구분도를 그리기 위해서 `plot_ly()`, `plot_geo()`, `plot_mapbox()` 중에 `plot_ly()`를 사용하였다.

그리고 전체 플롯의 제목과 여백을 설정하기 위해 `layout()`을 사용하였다.

다음은 한국지도에 전국의 대학 신입생 충원율을 색상으로 표시하는 단계구분도를 만드는 R 코드이다.



실행결과 VI-12. R 의 plot_ly()로 구현한 단계 구분도

2.2.3.3. 맵 박스를 사용한 지도

R 에서 스캐터 맵을 맵박스 지도 위에서 구현하는 것은 앞에서 설명한 `plot_mapbox()`로 맵박스로 지도를 그리고, 이 지도 위에 위도와 경도 축에 표현되는 scatter 트레이스를 그리는 방법으로 구현할 수 있다.

맵박스에서 맵을 불러들이는 과정은 크게 세 가지 방법이 있는데 이 세 가지 방법들은 각각의 레이어로 구성이 되는데 기본 레이어 외의 레이어에 속한 맵들의 순서는 'below' 속성을 통해 순서를 다시 구성할 수 있다.

첫 번째는 맵 레이어는 layout.mapbox.sytle 로 정의되는 기본 맵 레이어이다. 이 맵은 가장 낮은 수준의 레이어로써 필수적인 레이어의 맵이고 순서를 변경할 수 없다. 두 번째는 `plot_ly()`에 의해 불려지는 맵 레이어이다. 보통 이 레이어는 지도에 표시되는 트레이스에 관련한 레이어로 많이 사용된다. 세 번째는 layout.mapbox.layer 에 의해 불려지고 순서가 정해지는 레이어이다. 이 레이어는 raster 나 vector 등의 타입으로 설정되어 기본 맵이나 트레이스 맵의 보완적 맵으로 많이 사용된다.

트레이스 이름에 'mapbox'가 붙는 트레이스와 'layout'의 'mapbox' 속성들은 mapbox.com 에서 제공하는 Mapbox GL JS 오픈 소스 라이브러리를 사용한다. 'mapbox' 트레이스의 layout.mapbox.style 에서 호출할 때 API 인증이 필요한 지도를 사용하기 위해서는 'https://mapbox.com/'에서 무료 계정을 등록하고 Mapbox 액세스 토큰을 받아야 한다.

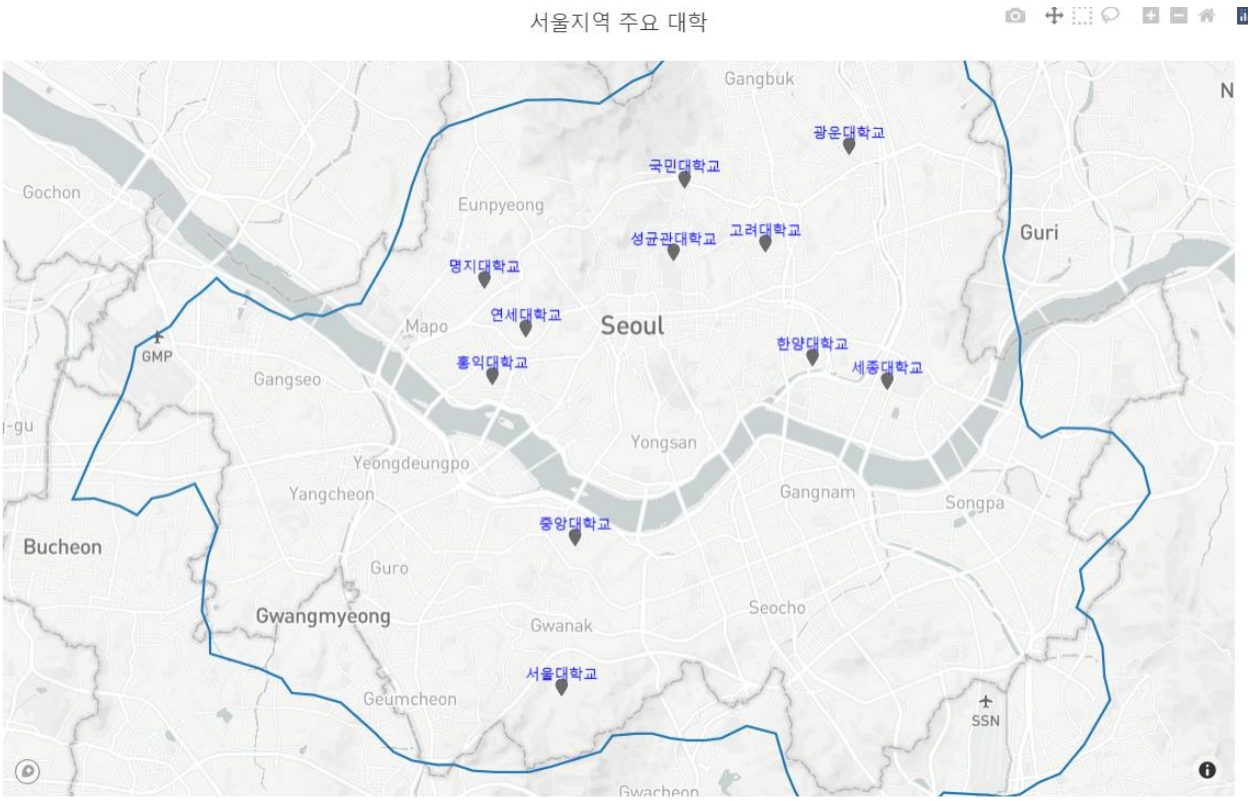
'white-bg', 'open-street-map', 'carto-positron', 'carto-darkmatter', 'stamen-terrain', 'stamen-toner', 'stamen-watercolor'의 몇 가지 맵을 제외하고는 대부분의 맵이 API 인증이 필요한데 API 키를 받으면 다음과 같이 설정할 수 있다.

```
##  맵박스 토큰 설정
Sys.setenv("MAPBOX_TOKEN" = mapboxToken)
```

다음은 서울의 주요 대학의 위도와 경도 데이터를 읽어들이고, `raster` 패키지에서 우리나라 지도를 시군구 레벨('level = 2')로 불러들였다.

이 지도위에 scatter 트레이스로 마커와 텍스트로 각 대학의 위치를 표시하였다.

```
##  대학의 위경도 데이터 불러들임
df_univ <- read_excel("D:/R/git/datavisualization/plotly/RnPy/chap6/university.xlsx",
                      col_types = c('text', 'numeric', 'numeric'))

plot_dat_seoul <- plot_dat |> filter(GID_1 == 'KOR.16_1')

plot_mapbox(plot_dat_seoul) |>
  add_trace(data = df_univ, type = 'scattermapbox', mode = 'markers+text',
            x = ~lon, y = ~lat,
            marker = list(size = 10, symbol = 'marker'),
            text = ~학교명, textposition = 'top center',
            textfont = list(color = 'blue')) |>
  layout(title = '서울지역 주요 대학',
         autosize=TRUE, hovermode='closest',
         mapbox=list(
           bearing=0, center=list(lon=126.98, lat=37.56),
           pitch=0, zoom=10, style="light"),
        margin = margins_R,
        showlegend = FALSE)
```



실행결과 VI-13. R 의 맵 박스를 사용한 지도