## 색상 설정

데이터 시각화에서 데이터의 표현을 위해 가장 중요한 것은 데이터를 어떠한 도형을 사용하여 구조적으로 표현할 것인지이다. 하지만 그 데이터 시각화를 보다 효과적으로 사용하기 위해서 가장 효율적으로 사용되는 것이 데이터를 표현하는 색상이다. 색상을 어떻게 쓰느냐에 따라서 청중들을 데이터 시각화에서 전달하고자 하는 핵심에 빠르게 집중시킬 수도 있고 전혀 집중하지 못하게 할 수도 있다. 따라서 데이터 트레이스에 원하는 색상을 정확히 사용하는 방법을 익히는 것이 매우 중요하다. 데이터 시각화에서 사용하는 색상은 크게 연속형 색상과 이산형 색상으로 나눌 수 있다.

### 연속헝 색상

연속형 색상은 일반적으로 연속형 변수에 매핑되어 색상간의 단계 구분없이 색상의 전반적인 흐름을 사용하여 데이터의 변량 값을 표현한다. plotly에서 연속형 색상을 표현하는 방법으로 다음의 네 가지 컨셉을 사용한다.

* color scale : 전체 색상의 범위와 데이터를 0에서 1까지의 범위로 정규화하고 데이터에 매핑되는 색상으로 데이터를 표현하는 방법이다. 따라서 보통의 경우 0에 해당하는 색상과 1에 해당하는 색상을 설정하고 이 사이의 색상들을 보간(interpolate)하여 색상을 생성하고 여기에 데이터를 매핑하여 표현할 색상을 결정한다.
* color ranges : colorscale의 색상 범위인 0~1에 매핑될 데이터 값의 범위를 설정함으로써 색상을 매핑하는 방법이다. 기본적으로 color range는 기본적으로 매핑되는 데이터의 최솟값과 최댓값으로 설정되지만 ‘cmin’, ‘cmid’, ‘cmax’ 속성값을 사용해 설정할 수 있다.
* colorbar : plolty에서는 연속형 색상의 범례를 표현하는 방식으로 컬러바를 사용한다. 이 컬러바는 전체적인 colorscale을 눈금 레이블과 눈금 표시를 사용하는 색상 범례를 말한다. 컬러바는 레이아웃의 ‘colorbar’ 속성을 사용하여 설정하거나 scatter 트레이스 ‘marker’ 속성의 ‘colorbar’ 속성으로 설정이 가능하다.
* color axes : 컬러 축은 위의 세 가지 컬러 컨셉을 데이터와 연결한다. 기본적으로 컬러로 표현될 수 있는 속성은 그 자체적인 색상 축으로 설정되어 연결된다.

색상을 설정할 때 R과 python에 몇 가지 차이점이 있다.

R에서는 트레이스 설정에 사용하는 ‘data’ 속성의 하위 리프 노드 속성으로 ‘color’를 사용할 수 있지만 python에서는 데이터가 실제로 표시되는 ’data’ 속성의 하위 속성인 ‘marker’ 속성의 하위 속성으로 ‘color’를 설정한다는 것이다. 또 R에서는 ’data’ 속성의 하위 리프 노드 속성으로 ‘colors’를 사용하여 색상 팔레트를 설정하지만 python에서는 ’colors’ 속성이 존재하지 않는다.

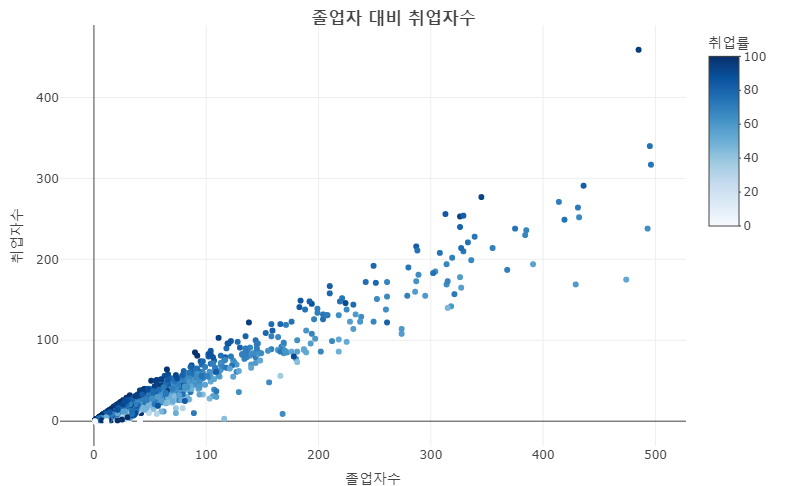
반면 R에서 색상의 설정시 특정 색상으로 지정할 때는 색상명을 바로 쓸 수 없고 I()를 사용하여 색상을 설정하여야 한다.

다음은 졸업자수 대비 취업자수의 scatter 트레이스에 연속형 변수인 취업률을 색상 변수로 사용하는 R과 python 코드이다.

* R

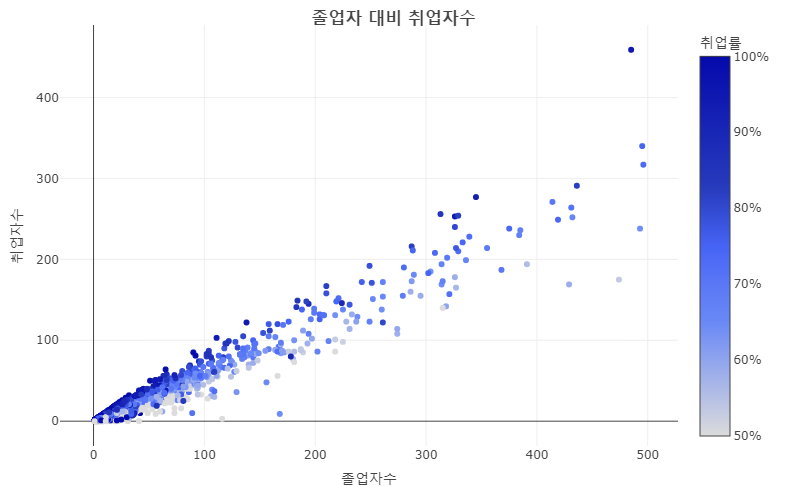
R에서 변수를 색상에 매핑하기 위해서는 ‘data’ 속성의 하위 속성인 ‘color’ 속성에 색상을 매핑할 데이터프레임 열이나 벡터를 설정한다. 데이터프레임 열을 매핑할 때는 ‘x’, ‘y’ 속성과 같이 “~”를 열 이름 앞에 붙여주어야 한다. 만약 전반적 컬러 스케일을 변경하고자 한다면 두 가지 방법을 사용할 수 있는데, 사용자가 직접 최솟값에 해당하는 색상과 최댓값에 해당하는 색상을 설정함으로서 변경할 수 있고 미리 정의된 팔레트를 사용하는 방법이 있다. 컬러 팔레트를 설정한다면 다음과 같이 ‘colors’ 속성에 색상 팔레트 설정하여 사용할 수 있다.

df\_취업률\_500 |>   
 filter(졸업자수 < 500) |>   
 plot\_ly() |>  
 add\_trace(type = 'scatter', mode = 'markers',   
 x = ~졸업자수, y = ~취업자수,   
 color = ~취업률, colors = 'Blues') |>   
 ## title 속성의 설정  
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>',   
 x = 0.5, xanchor = 'center', yanchor = 'top'))



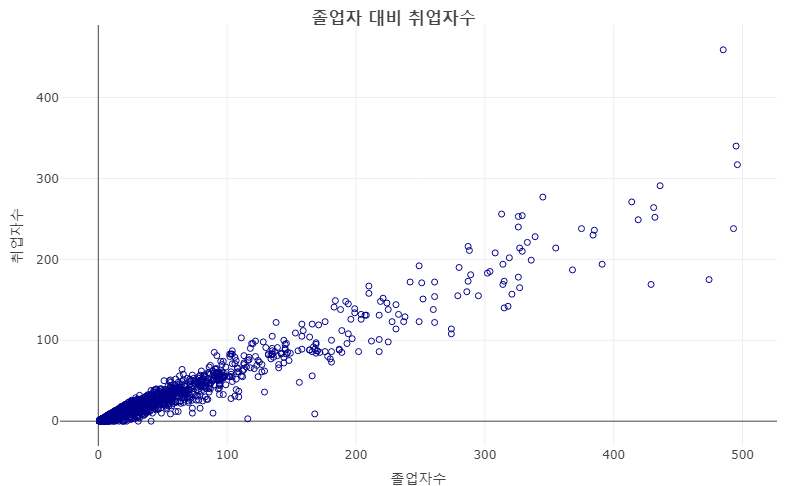
색상의 전반적 스케일 중 일부 스케일에만 관심이 있다면 color ranges를 설정하여 범위를 설정할 수 있다. ‘color’와 ’colors’ 속성외에 속성을 설정하기 위해서는 python과 같이 ‘marker’ 속성의 세부 속성으로 설정해야 한다. 다음은 ‘cmin’과 ’cmax’를 사용하여 color range를 50~100사이로 설정하는 R 코드이다. ’marker’ 내부에서 ‘colorscale’을 설정하면 ’marker’외부에서 ’colors’로 설정하는 것과 반대로 설정되기 때문에 ’reversescale’을 사용하여 색상 스케일을 반전시켰고 ’colorbar’의 ’title’을 사용하여 컬러바 제목과 ’ticksufix’를 사용하여 컬러바의 라벨에’%’ 붙여주었다.

df\_취업률\_500 |>   
 filter(졸업자수 < 500) |>   
 plot\_ly() |>  
 add\_trace(type = 'scatter', mode = 'markers',   
 x = ~졸업자수, y = ~취업자수,   
 marker = list(color = ~취업률, colorscale = 'Blues',   
 cmin = 50, cmax = 100,   
 colorbar = list(title = '취업률', ticksuffix = '%'),  
 reversescale = T)   
 ) |>   
 ## title 속성의 설정  
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>',   
 x = 0.5, xanchor = 'center', yanchor = 'top'))



앞서 설명했듯이 R에서 색상을 변수에 매핑하지 않고 단일 색상으로 설정하기 위해서는 색상명만으로 설정이 불가하고 I()를 사용해야 한다. 이 방법은 ‘color’외에 ’symbol’ 설정에도 동일하게 적용된다. 다만 ‘marker’ 속성의 하위 속성으로 ’color’를 설정할 떄는 I()를 사용하지 않아도 가능하다. 다음은 marker의 색상을 “darkblue”로 설정하는 R 코드이다.

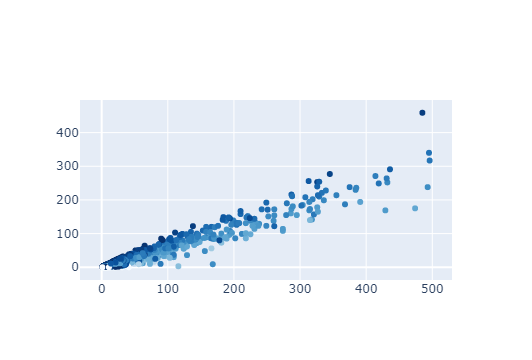
df\_취업률\_500 |>   
 filter(졸업자수 < 500) |>   
 plot\_ly() |>  
 add\_trace(type = 'scatter', mode = 'markers',   
 x = ~졸업자수, y = ~취업자수,   
 color = I('darkblue'), symbol = I('circle-open')) |>   
 ## title 속성의 설정  
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>',   
 x = 0.5, xanchor = 'center', yanchor = 'top'))



* python

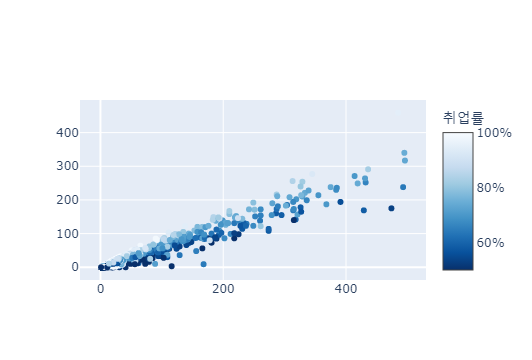
python에서 변수를 색상에 매핑하기 위해서는 ‘data’ 속성의 하위 속성인 ‘marker’ 속성의 하위 속성으로 ‘color’에 연속형 변수를 매핑한다. 만약 전반적 컬러 스케일을 변경하고자 한다면 두 가지 방법을 사용할 수 있는데, 사용자가 직접 최솟값에 해당하는 색상과 최댓값에 해당하는 색상을 설정함으로서 변경할 수 있고 미리 정의된 팔레트를 사용하는 방법이 있다. 컬러 팔레트를 설정한다면 다음과 같이 ’colors’ 속성에 색상 팔레트 설정하여 사용할 수 있다. 다음은 ‘color’와 ’colorscale’ 속성을 사용하여 ’marker’의 색상과 팔레트를 설정한 python 코드이다.

fig = go.Figure()  
  
fig.add\_trace({  
 'type' : 'scatter', 'mode' : 'markers',  
 'x': df\_취업률\_500['졸업자수'], 'y': df\_취업률\_500['취업자수'],   
 'marker' : {'color' : df\_취업률\_500['취업률'], 'colorscale' : 'Blues'}})  
   
fig.show()



색상의 전반적 스케일 중 일부 스케일에만 관심이 있다면 color ranges를 설정하여 범위를 설정할 수 있다. ‘color’와 ’colors’ 속성외에 속성을 설정하기 위해서는 python과 같이 ‘marker’ 속성의 세부 속성으로 설정해야 한다. 다음은 ‘cmin’과 ’cmax’를 사용하여 color range를 50~100사이로 설정하는 R 코드이다. ’marker’ 내부에서 ‘colorscale’을 설정하면 ’marker’외부에서 ’colors’로 설정하는 것과 반대로 설정되기 때문에 ’reversescale’을 사용하여 색상 스케일을 반전시켰고 ’colorbar’의 ’title’을 사용하여 컬러바 제목과 ’ticksufix’를 사용하여 컬러바의 라벨에’%’ 붙여주었다.

fig = go.Figure()  
  
fig.add\_trace({  
 'type' : 'scatter', 'mode' : 'markers',  
 'x': df\_취업률\_500['졸업자수'], 'y': df\_취업률\_500['취업자수'],   
 'marker' : {'color' : df\_취업률\_500['취업률'], 'colorscale' : 'Blues',   
 'cmin' : 50, 'cmax' : 100,  
 'colorbar' : {'title' : '취업률', 'ticksuffix' : '%'},  
 'reversescale' : True}})  
  
fig.show()



### 이산헝 색상

이산형 색상은 일반적으로 이산형 변수에 매핑되어 색상간의 단계 구분을 통해 데이터의 변량값을 표현한다. plotly에서 이산형 색상을 표현하는 방법으로 다음의 두 가지 컨셉을 사용한다.

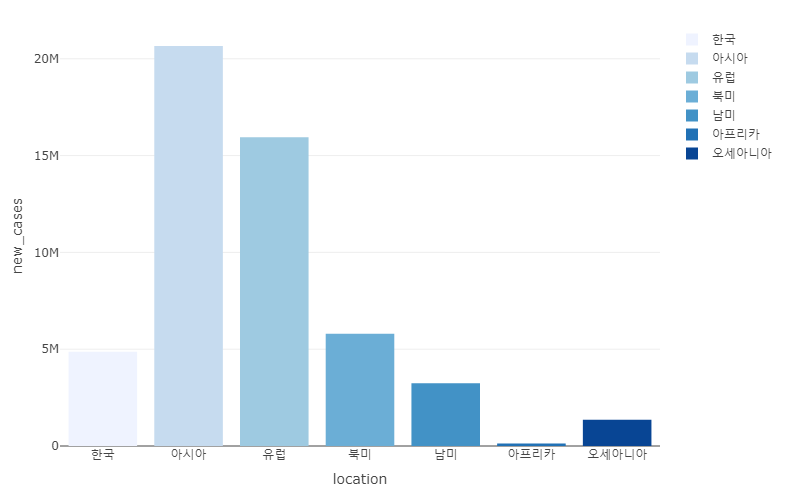
* color sequences : 컬러 시퀀스는 이산형 데이터에 각각 매핑되는 색상의 리스트를 설정하는 방법이다. 이 방법은 각각의 이산형 데이터에 1:1로 매칭되는 색상을 레이아웃의 ’colorway’에 리스트로 설정하는 방법과 plotly에서 제공하는 팔레트를 설정하여 표현하는 방법이 있다.
* legends : 이산형 색상은 앞서 연속형 색상에서 사용했던 컬러바를 사용하지 않고 색상 범레를 사용한다.

다음은 졸업자수 대비 취업자수의 scatter 트레이스에 연속형 변수인 취업률을 색상 변수로 사용하는 R과 python 코드이다.

* R

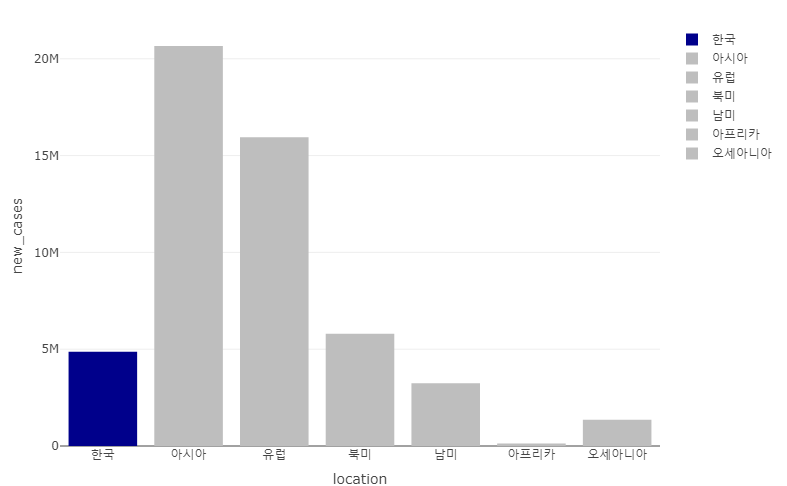
R에서 이산형 변수에 색상을 설정하기 위해서는 앞서 연속형 변수에서 사용했던 ‘color’ 속성과 ‘colors’ 속성을 사용할 수 있다.

df\_covid19\_100 |>  
 group\_by(location) |>  
 summarise(new\_cases = sum(new\_cases)) |>  
 plot\_ly() |>  
 add\_trace(type = 'bar',   
 x = ~location, y = ~new\_cases,   
 color = ~location, colors = 'Blues')



특히 ‘colors’ 속성은 이산형 변수에 매핑되는 이산형 색상은 이산형 변수의 각 변량에 매칭되는 색상을 설정할 수 있다. ‘colors’ 속성에 c()를 사용하여 이산형 변수의 변량 순서에 따른 색상을 문자열로 설정하면 각각의 변량에 매칭되어 색상이 표현된다. 이 방법은 다수의 이산형 변량중에 특정한 하나의 변량을 강조하고자 할 때 효율적으로 사용이 가능하다. 다음은 코로나19의 신규 확진자 중 한국 데이터를 강조한 막대그래프를 그리는 R 코드이다.

df\_covid19\_100 |>  
 group\_by(location) |>  
 summarise(new\_cases = sum(new\_cases)) |>  
 plot\_ly() |>  
 add\_trace(type = 'bar',   
 x = ~location, y = ~new\_cases,   
 color = ~location, colors = c('darkblue', 'gray', 'gray', 'gray', 'gray', 'gray', 'gray'))



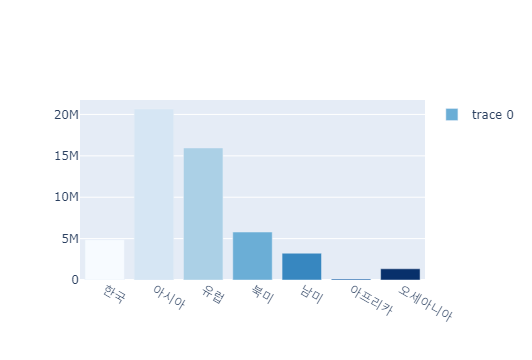
* python

python에서 이산형 색상을 설정하는 것은 R보다는 약간 번거롭다. 사실 plotly에서 이산형 색상의 설정은 연속형 색상의 설정과 기본적으로 다루는 방식이 다르다. 연속형 색상은 하나의 트레이스안에서 해당 트레이스의 데이터를 표현하는 방법으로 색상의 변화를 사용한다. 이것이 트레이스 리프 노드로 ‘color’를 설정하지 않고 ’marker’ 속성으로 ‘color’ 속성을 설정하는 이유이다. 그리고 ‘color’ 속성은 색상의 변화에 매핑되어야 하기 때문에 ‘color’ 속성에 할당되는 변수는 반드시 수치형 변수이어야 한다. 하지만 이산형 색상 설정에서는 각각의 이산형 변수들이 독립적 색상을 가지고 그 중간의 색상이 없기 때문에 plotly는 이산형 색상을 각각의 트레이스로 분리하여 설정한다. 앞서 ‘name’ 속성에서 데이터 그룹간의 트레이스를 분리하여 for 루프를 사용한 것과 유사한 방법인 것이다.

물론 이산형 색상도 연속형 변수와 같이 for 루프를 사용하여 트레이스를 분리하지 않고 ‘color’ 속성을 사용할 수 있다. 이렇게 사용하기 위해서는 먼저 이산형 변수를 수치형 변수로 변환하고 이 변수를 ’marker’의 ’color’에 설정해 줄 수 있다.

다음은 하나의 트레이스에 이산형 색상을 설정하는 python 코드이다. 먼저 이산형 변수인 ‘location’ 열은 수치형 변수가 아니기 때문에 ‘cat()’ 함수를 만들어서 각각의 변량값에 매칭되는 수치형 열을 만든다. 다음으로 bar 트레이스의 막대그래프를 그리는데 ‘marker’ 속성의 ‘color’ 속성에 이산형 변수를 수치형으로 만든 열을 설정하고 색상 팔레트를 ‘Blues’ 팔레트로 설정하였다.

fig = go.Figure()  
  
## 이산형 변수를 수치형 변수로 변환하는 함수   
def cat(row):  
 key = row['location']  
 value = {'한국': 0 , '아시아' : 1, '유럽' : 2, '북미' : 3, '남미' : 4, '아프리카' : 5, '오세아니아' : 6}.get(key)  
 return value  
  
## 시각화를 위한 데이터 전처리  
temp = df\_covid19\_100.groupby('location').agg(new\_cases = ('new\_cases', 'sum')).reset\_index()  
  
## 이산형 변수 열을 사용해 수치형 변수열 생성  
temp['loc\_code']= temp.apply(cat\_1, axis=1)  
  
fig.add\_trace({  
 'type' : 'bar', ## bar 트레이스 설정  
 'x': temp['location'], 'y': temp['new\_cases'],   
 ## 이산형 변수를 변환한 수치형 변수를 color로 설정  
 'marker' : {'color' : temp['loc\_code'], 'colorscale' : 'Blues'},   
 'showlegend' : True  
})  
  
fig.show()

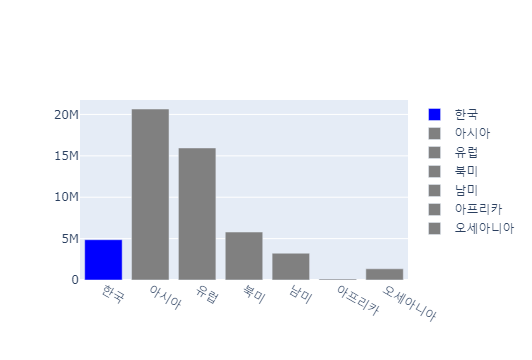


위의 결과를 보면 한국과 각 대륙의 bar 트레이스가 하나의 트레이스로 구성되어 있다. 그래서 범례가 ’trace 0’라고 명명되어 하나의 범례만 나타난다. 하지만 일반적인 사용자들은 범례에도 색상과 각각의 대륙명이 나오길 바랄 것이다. 이를 위해서는 각각의 대륙에 따르는 bar 트레이스를 그려주어야 한다.

그런데 이렇게 트레이스를 나누어줄 때 트레이스 색상을 각각 설정해야 한다. 각각의 색상 설정은 이산형 변수에 매칭되는 딕셔너리를 설정하여 사용하거나 update\_layout()의 ’colorway’를 사용하는 두 가지 방법이 있다.

다음은 for 루프를 사용하여 각각의 트레이스를 나누고 이산형 변수에 매칭되는 색상 딕셔너리를 사용하여 한국의 데이터를 강조하는 python 코드이다.

fig = go.Figure()  
  
## 이산형 변수와 색상을 매칭하는 딕셔너리 정의  
colors = {'한국': 'blue' , '아시아' : 'gray', '유럽' : 'gray', '북미' : 'gray', '남미' : 'gray', '아프리카' : 'gray', '오세아니아' : 'gray'}  
temp = df\_covid19\_100.groupby('location').agg(new\_cases = ('new\_cases', 'sum')).reset\_index()  
  
## 'location'의 그룹에 따라 for 루프  
for cat, group in temp.groupby('location'):  
 fig.add\_trace({  
 'type' : 'bar', ## bar 트레이스 설정  
 'x': group['location'], 'y': group['new\_cases'],   
 'name' : cat,  
 'marker' : {'color' : colors[cat]}  
 })  
  
fig.show()



이를 update\_layout()의 ’colorway’를 사용할 수 있는데 이를 사용한 python 코드는 다음과 같다.

fig = go.Figure()  
  
temp = df\_covid19\_100.groupby('location').agg(new\_cases = ('new\_cases', 'sum')).reset\_index()  
  
## 'location'의 그룹에 따라 for 루프  
for cat, group in temp.groupby('location'):  
 fig.add\_trace({  
 'type' : 'bar', ## bar 트레이스 설정  
 'x': group['location'], 'y': group['new\_cases'],   
 'name' : cat,  
 'marker' : {'color' : colors[cat]}  
 })  
  
fig.update\_layout({  
 'colorway' : ('blue', 'gray', 'gray', 'gray', 'gray', 'gray', 'gray')})  
   
fig.show()