1. plotly 시각화 꾸미기와 사용하기

# plotly 시각화 꾸미기

## 레이아웃 설정 : layout()

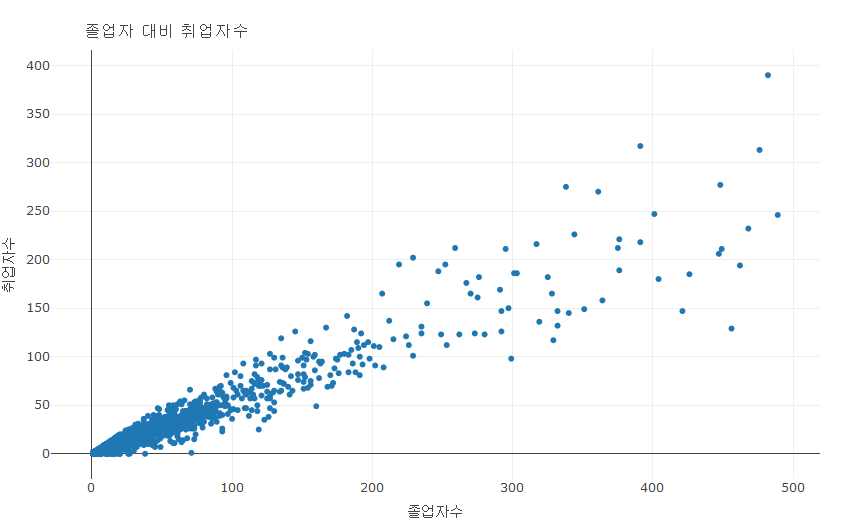
지금까지는 시각화를 만들기 위해 필요한 데이터를 trace로 표현하는 방법에 대해 알아보았다. 이렇게 추가된 trace들을 보다 효과적으로 시각화하기 위해서는 시각화의 제목, 범례, 여백, 크기, 폰트, 축 등에 대한 세부 설정을 해야 할 필요가 있다. plotly에서 이들을 설정하기 위해서는 layout()을 사용하여 설정할 수 있다. 이 설정들은 앞선 add\_trace()와 마찬가지로 list()를 사용하여 설정이 가능하다.

### 제목 설정 : title

plotly의 제목을 설정하는 type은 title이다. title의 세부 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| font | 제목 글꼴 설정 | 세부 속성들의 리스트 | color, family, size |
| pad | 제목 패딩(여백과 시각화 개체와의 거리) 설정 | 세부 속성들의 리스트 | b, l, r, t |
| text | 제목 문자열 설정 | 문자열 |  |
| x, y | 제목의 위치 | 0 ~ 1 사이의 수치 |  |
| xanchor, yanchor | X, Y방향의 정렬 방법 | ‘auto’, ‘left’, ‘right’, ‘center’ |  |
| xref, yref | x, y가 참조하는 범위 설정 | ‘container’, ‘paper’ |  |

## df\_취업률\_2000 데이터를 사용하는 plotly 객체 생성  
p\_layout\_1 <- df\_취업률\_2000 |>   
 plot\_ly()  
  
## markers 모드의 스캐터 trace 추가  
p\_layout\_1 <- p\_layout\_1 |>  
 add\_trace(type = 'scatter', mode = 'markers',   
 x = ~졸업자수, y = ~취업자수)  
  
p\_layout\_1 |>   
 ## layout의 title 세부 속성으로 시각화 제목 설정  
 layout(title = list(text = '졸업자 대비 취업자수', x = 0.1,   
 xanchor = 'left', yanchor = 'top'),  
 margin= margins)

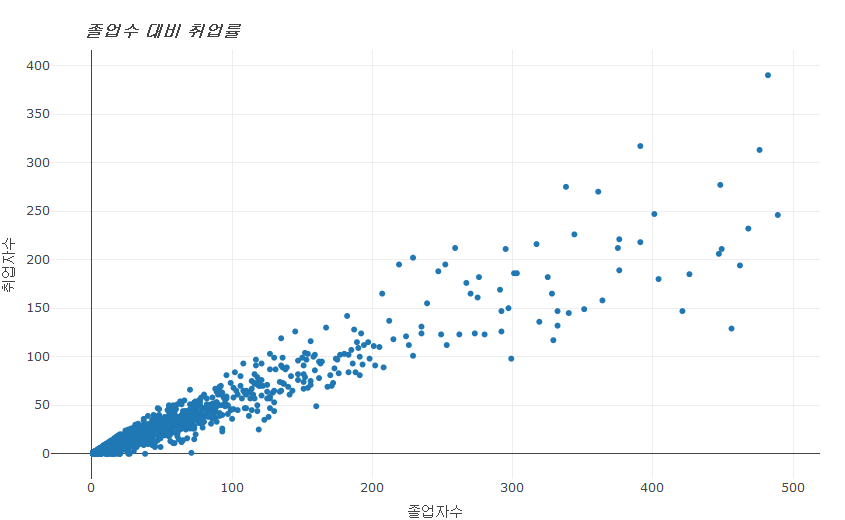


실행결과 3- . layout을 이용한 시각화 제목 설정

plotly에서는 제목과 같은 문자열을 꾸미기 위해 HTML 텍스트 태그를 지원하는데 전체 HTML 중 5가지만을 지원한다.

| HTML tab | 설명 |
| --- | --- |
| <b></b> | 볼드체 설정 |
| <i></i> | 이탤릭체 설정 |
| <br> | 줄 바꿈 설정 |
| <sup></sup> | 윗 첨자 설정 |
| <sub></sub> | 아래 첨자 설정 |
| <a href=‘url’></a> | 하이퍼링크 설정 |

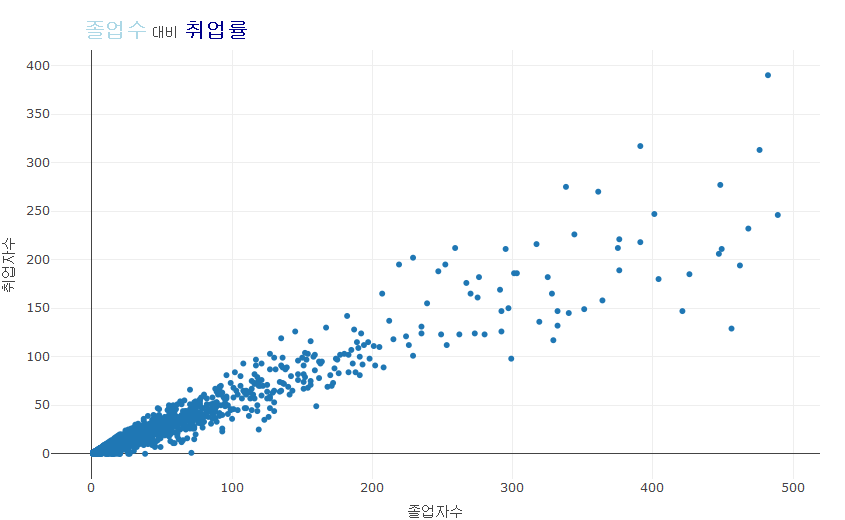
p\_layout\_1 |>   
 ## 문자열 강조를 위해 <b></b> 태그 사용  
 layout(title = list(text = "<b><i>졸업수 대비 취업률</b></i>",   
 x = 0.1, xanchor = 'left', yanchor = 'top'),  
 margin = margins  
 )



실행결과 3- . html 태그를 사용한 제목 설정

plotly가 HTML 텍스트 tag를 일부만 지원함으로써 문자열 스타일링에 한계가 있을듯 하지만 ’’을 사용하는 HTML inline 속성을 지원하기 때문에 CSS의 스타일을 사용하여 문자열의 세부 설정이 가능하다. 다음은 HTML의 inline 속성을 사용하여 제목을 설정하는 코드이다.

p\_layout\_1 |>   
 ## <span></span>을 사용한 제목 설정  
 layout(title = list(text = "<span style = 'font-size:15pt'><span style = 'color:lightblue;font-weight:bold;'> 졸업수</span><span style = 'font-size:10pt'> 대비</span> <span style = 'color:darkblue;font-weight:bold;'>취업률</span></span>",   
 x = 0.1, xanchor = 'left', yanchor = 'top'),   
 margin = margins  
 )



실행결과 3- . CSS 스타일의 제목 설정

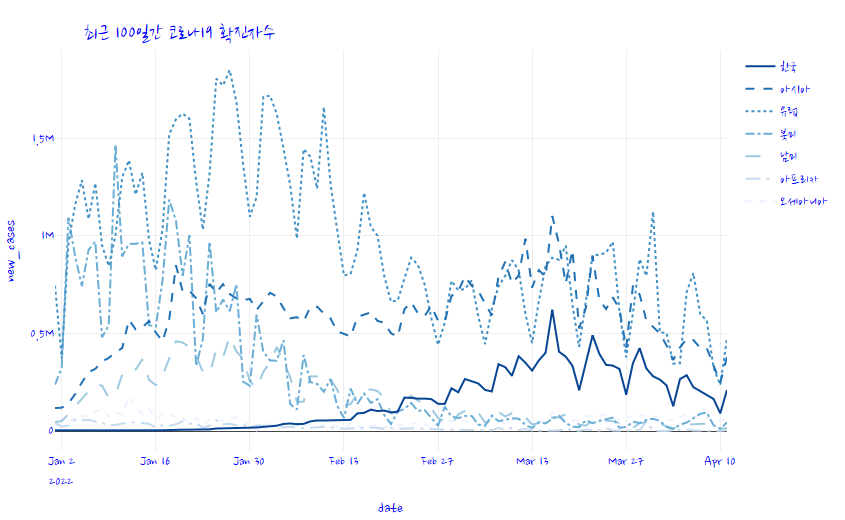
### 폰트 설정 : font

layout()에서 플롯 전체적인 문자열의 설정과 관련된 type은 font이다. font는 다음과 같은 세 개의 세부 type을 설정할 수 있다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| color | 플롯 전체 문자 색 설정 | 컬러 값 |  |
| family | 플롯 전체 문자 폰트 설정 | 폰트명 |  |
| size | 플롯 전체 문자 크기 설정 | 1이상의 수치값 |  |

전체적인 폰트의 설정은 다음과 같이 설정할 수 있다. ggplot2에서는 한글 폰트의 설정에 다소 어려움이 있었지만 plotly에서는 family type에 바로 폰트 이름을 설정하면 쉽게 한글 폰트를 사용할 수 있다.

p\_layout\_2 <- df\_covid19\_100 |>  
 ## mode가 lines인 스캐터 trace plotly 객체 생성  
 plot\_ly(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~new\_cases, linetype = ~location)  
  
p\_layout\_2 |>   
 layout(title = list(text = '최근 100일간 코로나19 확진자수', x = 0.1,  
 xanchor = 'left', yanchor = 'top'),  
 ## font의 세부 속성 설정  
 font = list(family = "나눔손글씨 붓", color = 'blue',   
 size = 15),   
 margin = margins  
 )



실행결과 3- . layout을 사용한 폰트 설정

### 축 설정 : xaxis, yaxis

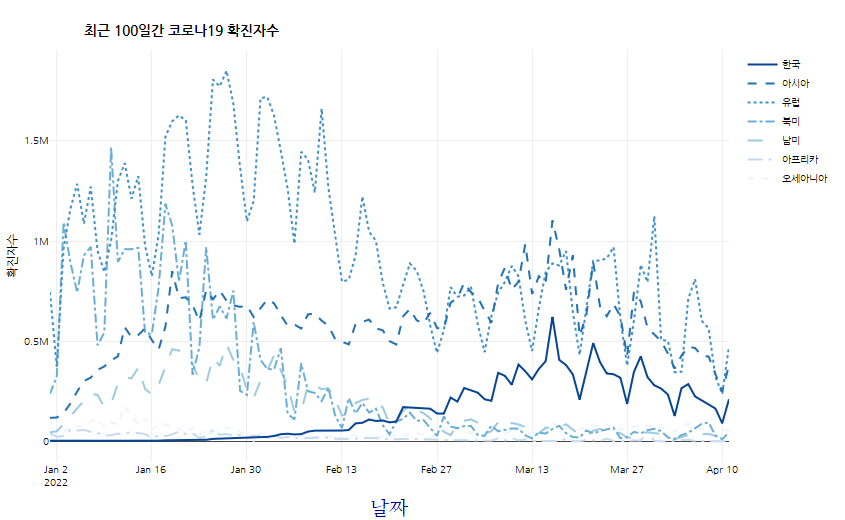
plotly에서 축은 trace type에 따라 매우 다른 이름으로 불린다. X, Y축을 사용하는 2차원의 데카르트 좌표계에서는 ‘axis’, 3차원 trace에서는 ‘scene’, 극 좌표계에서는 ‘polar’, 삼각축 좌표계(ternary)에서는 ‘ternary’, 지형(Geo) 좌표계에서는 ‘geo’, Mapbox 좌표계에서는 ‘mapbox’, 색 좌표계에서는 ’coloraxis’로 사용된다. 이 절에서는 이 중 2차원 데카르트 좌표계에서 사용되는 xaxis와 yaxis를 위주로 설명하겠다.

layout()에서 X축의 설정과 관련된 type은 xaxis이다. xaxis에서 제공하는 주요 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| title | 축 이름 설정 | 세부 속성들의 리스트 | font(color, family, size), standoff, text |
| type | 축 유형 설정 | ‘linear’, ‘log’, ‘date’, ‘category’, ‘multicategory’ |  |
| range | 축 범위 설정 | 리스트 |  |
| rangemode | 축 범위의 특성 설정 | ‘normal’, ‘tozero’, ‘non-negative’ |  |
| linecolor | 축 선 색 설정 | 컬러 값 |  |
| linewidth | 축 선 두께 설정 | 0이상의 수치값 |  |
| zeroline | 0 선 표시 설정 | 논리값 |  |
| zerolinecolor | 0선 색 설정 | 컬러 값 |  |
| zerolinewidth | 0 선 두께 설정 | 0이상의 수치값 |  |
| position | 축 선 위치 설정 | 0 ~1까지 수치값 |  |
| gridcolor | 눈금선 색 설정 | 컬러 값 |  |
| gridwidth | 눈금선 두께 설정 | 0이상의 수치값 |  |
| autotypenumber | 축에 설정되는 값이 문자형일때 숫자형으로 변환할지를 결정 | ‘convert types’, ‘strict’ |  |
| ticks | 눈금자를 어느쪽으로 그릴지 설정 | ‘outside’, ‘inside’ |  |
| tickangle | 눈금자의 각도 설정 | 각도 값 |  |
| ticklen | 눈금자의 길이 설정 | 0이상의 수치값 |  |
| tickcolor | 눈금자의 색 설정 | 컬러 값 |  |
| tick0, dtick | 눈금자의 시작점, 간격 설정 | 수치 또는 카테고리 좌표 문자열 |  |
| nticks | 눈금자 개수 설정 | 0이상의 수치값 |  |
| tickvals, ticktext | 눈금자의 위치, 표시문자 설정 | 데이터 프레임열, 리스트, 벡터 |  |
| showgrid, showline, showticklabel | 눈금선, 축 선, 눈금자 라벨의 표시 여부 설정 | 논리값 |  |
| tickprefix, ticksuffix | 눈금자 라벨의 접두어, 접미어 설정 | 문자열 |  |

축 설정에 가장 많이 사용되는 속성중에 하나는 축 제목이다. 축 제목은 title 속성을 사용하여 설정이 가능한데 축 제목은 설정하는 두 가지 방법이 있다. 첫 번째는 title에 바로 문자열을 설정함으로써 해당 축의 제목을 설정하는 방법이다. 이 방법은 이전 버전의 plotly에서 사용되던 방법인데 이 방법을 사용하면 축 제목의 폰트 패밀리, 색, 크기를 설정할 수 없다. 만약 세부 설정을 해야한다면 title을 list()로 세부 속성 값들을 리스트로 묶어 설정해야 한다. 첫 번쨰 방법은 쉽게 사용이 가능하지만 현재 버전의 plotly에서는 권장되지 않기 때문에 가급적 두번쨰 방법을 사용하는 것이 좋을 듯 하다.

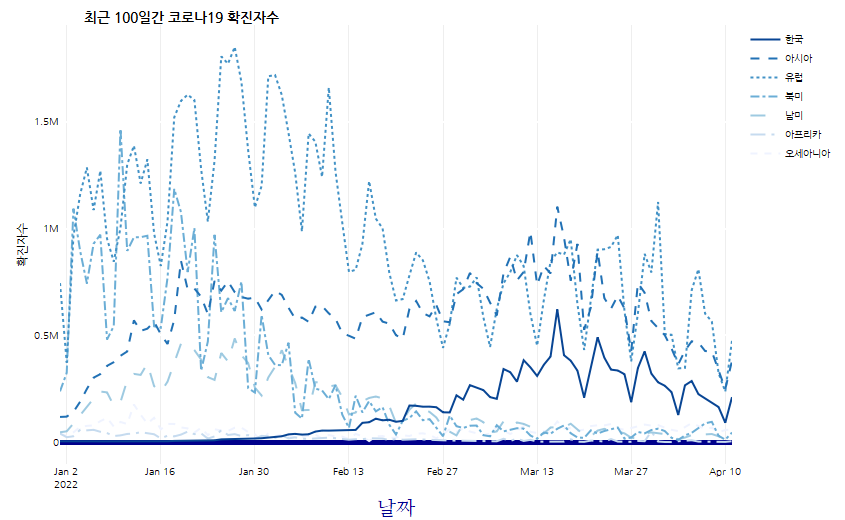
p\_layout\_2 |>   
 layout(title = list(text = '<b>최근 100일간 코로나19 확진자수</b>', x = 0.1,  
 xanchor = 'left', yanchor = 'top', size = 15),  
 font = list(family = "나눔고딕", color = 'black', size = 10),   
 ## xaxis 제목 속성 설정 - 정상적 방법  
 xaxis = list(title = list(text = '날짜',   
 font = list(family = '나눔명조',   
 size = 20, color = 'darkblue'))  
 ),  
 ## yaxis 제목 속성 설정 - 약식 방법  
 yaxis = list(title = '확진자수'),   
 margin = margins  
 )



실행결과 3- . layout을 사용한 축 제목 설정

다음의 예는 Y축의 grid와 zeroline의 속성을 설정하는 예이다.

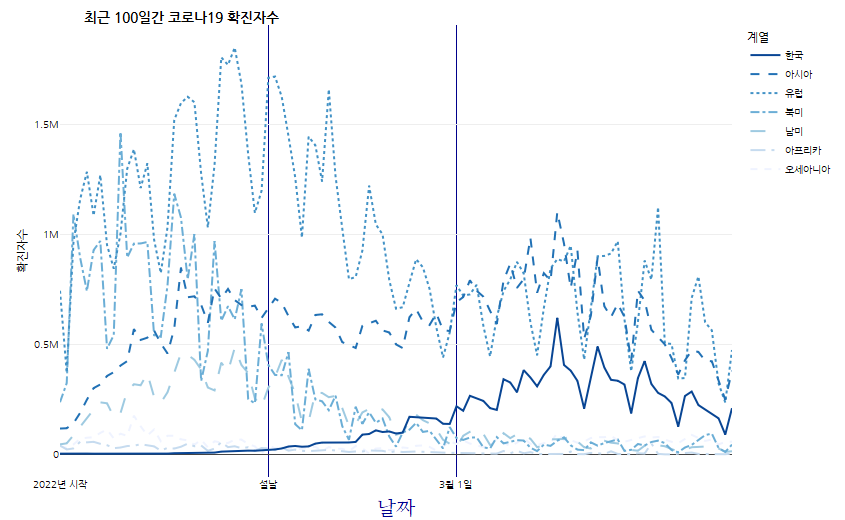
p\_layout\_2 |>   
 layout(title = list(text = '<b>최근 100일간 코로나19 확진자수</b>',  
 x = 0.1, xanchor = 'left', yanchor = 'top', size = 15),  
 font = list(family = "나눔고딕", color = 'black', size = 10),  
 xaxis = list(title = list(text = '날짜',   
 font = list(family = '나눔명조',   
 size = 20, color = 'darkblue')  
 )  
 ),  
 yaxis = list(title = '확진자수',   
 ## grid 속성 설정  
 gridcolor = 'white', gridwidth = 2,   
 ## zeroline 속성 설정  
 zerolinewidth = 5, zerolinecolor = 'darkblue')  
 )



실행결과 3- . layout을 사용한 grid와 zeroline 설정

X축과 Y축의 눈금선은 plotly에서 자동적으로 계산하여 가장 적절한 눈금선의 개수와 간격, 라벨을 설정한다. 그러나 사용자가 직접 원하는 눈금선을 설정하기 위해서는 ticktext와 tickvals를 사용하여 설정할 수 있다. 여기서 주의해야할 점은 ticktext와 tickvals는 모두 벡터나 리스트로 여러 개의 눈금선을 설정할 수 있는데 두 벡터의 길이가 같아야 한다는 것이고 tickvals에 설정되는 벡터는 X축에 매핑되는 데이터 변수의 데이터 타입과 동일한 타입이어야 한다는 점이다. 물론 이 두가지 조건이 충족되지 않는다고 해도 에러를 내지는 않지만 사용자가 원하는 대로 표시되지는 않을 것이다.

p\_layout\_2 |>   
 layout(title = list(text = '<b>최근 100일간 코로나19 확진자수</b>',   
 x = 0.1, xanchor = 'left', yanchor = 'top', size = 15),  
 font = list(family = "나눔고딕", color = 'black', size = 10),  
 legend = list(title = list(text = '계열', side = 'top')),  
 xaxis = list(title = list(text = '날짜',   
 font = list(family = '나눔명조',   
 size = 20, color = 'darkblue')),   
 ## tickmode 설정  
 tickmode = 'array',  
 ## ticktext를 벡터로 설정  
 ticktext = c('2022년 시작', '설날', '3월 1일'),  
 ## tickvals를 벡터로 설정  
 tickvals = c('2022-01-01', '2022-02-01', '2022-03-01'),   
 ## grid의 설정  
 gridwidth = 1, gridcolor = 'darkblue'  
 ),  
 yaxis = list(title = '확진자수')  
 )



실행결과 3- . layout을 사용한 눈금라벨 설정

### 범례 설정 : legend

layout()에서 범례 설정과 관련된 type은 앞서 설명한 showlegend와 legend이다.

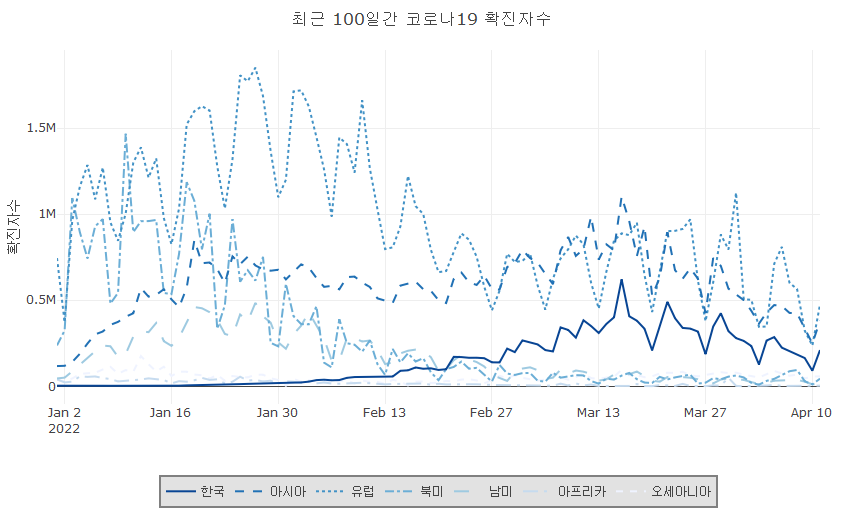
showlegend는 add\_trace()에서도 설정이 가능한 type인데 add\_trace()에서 설정하면 해당 trace만을 범례에서 제거하고 layout()에서 설정하면 전체 범례를 제가하게 된다.

legend는 범례의 세부 설정을 위한 세부 type의 list로 설정된다. legend로 설정이 가능한 주요 세부 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| bgcolor, bordercolor | 범례의 배경색, 외곽선 색 설정 | 컬러 값 |  |
| borderwidth | 범례를 둘러싸는 외곽선 색, 두께 설정 | 컬러 값, |  |
| font | 범례의 문자열에 적용되는 문자 속성 설정 | 0 이상의 수치 | color, family, size |
| groupclick | 범례 아이템을 클릭할 때 범례 그룹의 동작 설정 | ‘toggleitem’, ‘togglegroup’ |  |
| grouptitlefont | 범례 아이템 그룹의 제목과 제목 설정 | 세부 속성들의 리스트 | color, family, size |
| itemclick | 범례 아이템을 클릭 할 때의 동작 설정 | ‘toggle’, ‘toggleothers’, FALSE |  |
| itemdoubleclick | 범례 아이템을 더블 클릭 할 때의 동작 설정 | ‘toggle’, ‘toggleothers’, FALSE |  |
| orientation | 범례 표시 방향의 설정 | ‘v’, ‘h’ |  |
| title | 범례 제목 설정 | 문자열 | font(color, family, size), side, text |
| traceorder | 범례 순서 설정 | ‘reversed’, ‘grouped’, ‘normal’의 조합(’+’) |  |

다음의 예는 범례를 아래 쪽으로 배치하고 범례 방향을 수평으로 범례 배경색, 범례 외곽선 선과 색을 설정하는 예이다.

p\_layout\_2 |>  
 layout(title = '최근 100일간 코로나19 확진자수',   
 xaxis = list(title = list(text = NULL)),   
 yaxis = list(title = list(text = '확진자수')),   
 margin = margins,   
 ## legend 세부 속성을 사용한 범례 설정  
 legend = list(  
 ## 범례 배경색, 외곽선 색 설정  
 bgcolor = "#E2E2E2", bordercolor = 'gray',  
 ## 범례 외곽선 두께, 표시 방향 설정  
 borderwidth = 2, orientation = 'h',   
 ## X, Y 위치 설정과 위치 정렬 방법 설정  
 x = 0.5, y = -0.2, xanchor = 'center')  
 )



실행결과 3- . layout을 사용한 범레 설정

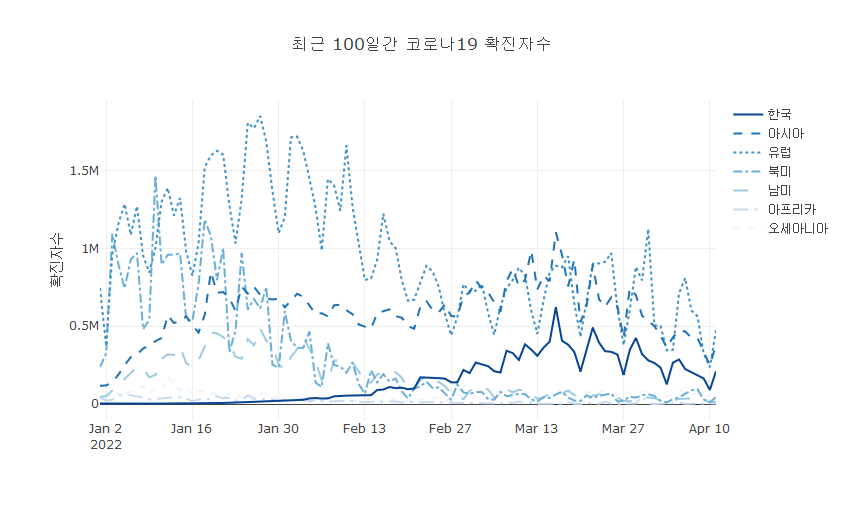
### 여백 설정 : margin

layout()에서 여백의 설정과 관련된 type은 margin이다. margin으로 설정되는 여백은 플롯이 표시되는 면적과 전체 플롯 경계선과의 여백을 말한다. 여백 설정에 사용되는 세부 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| autoexpand | 여백 설정 값을 사용할지 설정 | 논리값 |  |
| b | 아래쪽 여백 설정 | 0이상의 수치값 |  |
| l | 왼쪽 여백 설정 | 0이상의 수치값 |  |
| r | 오른쪽 여백 설정 | 0이상의 수치값 |  |
| t | 위쪽 여백 설정 | 0이상의 수치값 |  |
| pad | 플로팅 지역과 축과의 여백 설정 | 0이상의 수치값 |  |

앞선 plot\_ly()와 add\_trace()의 예를 보일 때 이미 layout()의 margin 속성이 사용되었다. 특히 제목이 설정된 plotly 객체에서는 이 여백을 적절히 설정해 주어야 전체 시각화가 보기 좋아진다.

p\_layout\_2 |>  
 layout(title = '최근 100일간 코로나19 확진자수',   
 xaxis = list(title = list(text = NULL)),   
 yaxis = list(title = list(text = '확진자수')),   
 ## margin 세부 속성을 사용한 여백 설정  
 margin = list(t = 100, b = 100, l = 100, r = 100)  
 )

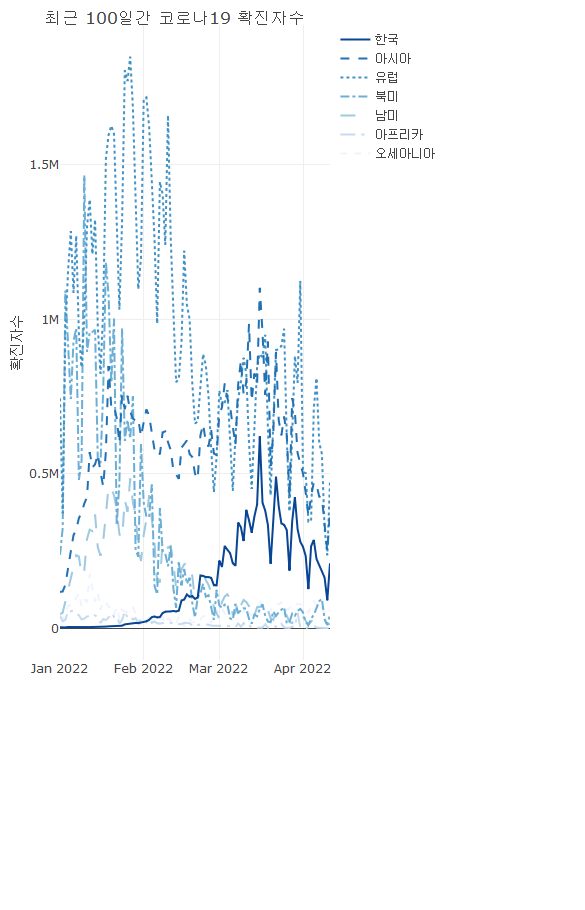


실행결과 3- . layout을 사용한 여백 설정

### 크기 설정 : size

layout()에서 전체 플롯 크기의 설정을 위해서는 autosize, height, width의 세 가지 type이 사용된다. autosize는 사용자가 정의하지 않은 레이아웃의 너비 또는 높이를 자동적으로 설정하는 논리값을 설정한다. height와 width는 전체 플롯 레이아웃의 높이와 너비를 설정하는 type으로 기본값이 450과 700이다.

p\_layout\_2 |>   
 layout(title = list(text = '최근 100일간 코로나19 확진자수',   
 x = 0.1, xanchor = 'left', yanchor = 'top'),  
 xaxis = list(title = list(text = NULL)),   
 yaxis = list(title = list(text = '확진자수')),   
 ## width, height를 사용한 시각화 크기 설정  
 width = 450, height = 700)



실행결과 3- . layout을 사용한 시각화 크기 설정

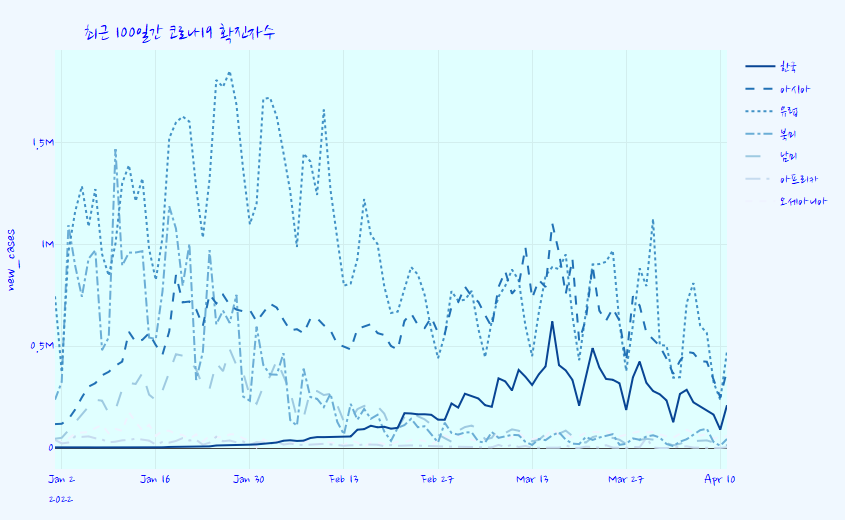
### 색 설정 : colors

layout()에서 플롯의 배경색이나 플롯에서 사용되는 전반적인 색 스케일을 설정하는 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| paper\_bgcolor | 플롯이 그려지는 용지의 배경색을 설정 | 컬러 값 |  |
| plot\_bgcolor | x축과 y축 사이의 플로팅 영역의 배경색 설정 | 컬러 값 |  |
| colorscale | 컬러 스케일의 설정 | 세부 속성들의 리스트 | diverging, sequential, sequentialminus |
| colorway | 기본 컬러 벡터 설정 | 컬러 리스트 |  |

plotly에서 사용하는 색의 설정은 색 이름 설정, 16진수로 설정된 RGB값 설정, rgb() 함수를 사용한 RGB값의 설정 세 가지 방법이 주로 사용된다. 이외에도 색조, 채도, 명도(lightness)를 사용하는 hsl()을 사용하는 방법, 색조, 채도, 명도(value)를 사용하는 hsv()를 사용하는 방법도 있다. plotly에서 사용이 가능한 색 이름은 W3.org에서 제공하는 CSS 색 이름을 사용한다.[[1]](#footnote-26)

p\_layout\_2 |>   
 layout(title = list(text = '최근 100일간 코로나19 확진자수', x = 0.1,   
 xanchor = 'left', yanchor = 'top'),  
 font = list(family = "나눔손글씨 붓", color = 'blue',   
 size = 15),  
 xaxis = list(title = list(text = NULL)),   
 ## paper\_bgcolor, plot\_bgcolor의 속성 설정  
 paper\_bgcolor = 'AliceBlue', plot\_bgcolor = 'LightCyan',  
 margin = margins  
 )



실행결과 3- . layout을 사용한 배경색 설정

### 주석 설정 : annotations

시각화 그래프를 만들때 시각화를 만드는 목적에 따라 시각화 그래프에 데이터에 대한 추가적인 설명이 들어가는 경우가 많다. 이와 같이 시각화를 만드는 사용자가 강조하고자 하거나 추가적으로 설명하고자 할때 사용하는 것이 주석(annotation)이다. plotly에서는 layout()에서 선과 화살표, 문자열을 사용하여 주석을 넣는 것이 가능하고 add\_annotation()을 사용할 수도 있다. layout()을 사용할 때는 annotations type을 사용하여 설정한다.

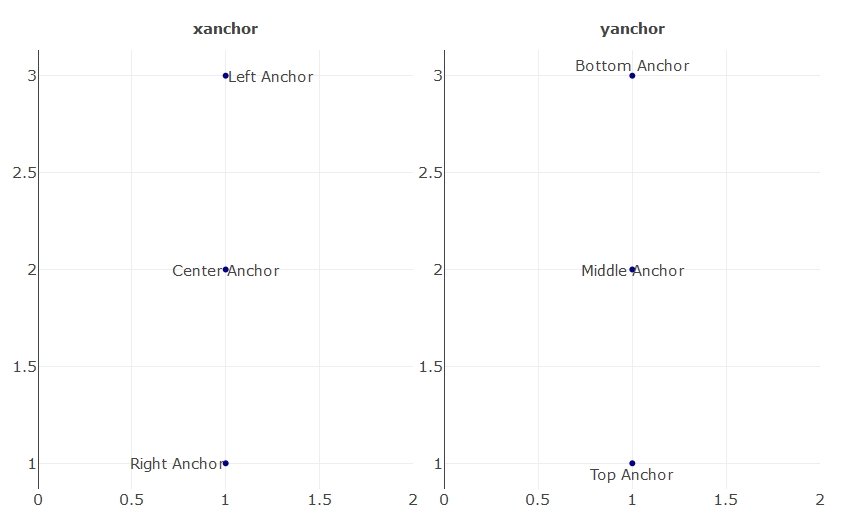
| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| text | 주석으로 표시할 문자열을 설정 | 문자열 |  |
| x, y | 주석을 달아야 할 X, Y축 좌표 설정 | 수치 또는 카테고리 좌표 문자열 |  |
| ax, ay | 주석의 화실표의 끝점 위치 | 수치 또는 카테고리 좌표 문자열 |  |
| xanchor, yanchor | X, Y방향의 정렬 방법 | ‘auto’, ‘left’, ‘center’, ‘right’ |  |
| xref, yref | x, y가 참조하는 범위 설정 | ‘x’, ‘y’, ‘paper’ |  |
| arrowcolor | 화살촉의 색 설정 | 컬러 값 |  |
| arrowhead | 화살촉의 모양 설정 | 0 ~ 8까지의 수치값 |  |
| arrowside | 화살촉 위치 설정 | ‘end’, ‘start’, ‘none’의 조합(’+’) |  |
| arrowsize | 화살촉의 크기 설정 | 0.3보다 큰 수치값 |  |
| arrowwidth | 화살촉의 너비 설정 | 0.1보다 큰 수치값 |  |
| bgcolor, bordercolor | 주석의 배경색, 주석 외곽선 색 설정 | 컬러값 |  |
| borderpad, borderwidth | 패딩(주석과 외곽선까지의 여백), 주석 외곽선 너비 설정 | 0보다 큰 수치값 |  |
| font | 주석의 문자 형태 설정 | 세부 속성들의 리스트 | color, family, size |

text는 주석으로 표시할 문자열을 설정하는 type이다. text는 주석의 표시에 있어 가장 중요한 type이고 필수적인 type이다. text는 문자열을 꾸밀수 있는 HTML 태그 중 줄 바꿈(), 굵은 글씨(), 기울임 글씨(), 하이퍼링크()를 사용할수 있고 , , 도 사용이 가능하다. 특히 을 사용하면 CSS 스타일을 사용할 수 있어 문자열을 세부적으로 꾸밀 수 있다.

x, y는 주석을 설정할 X축과 Y축의 좌표를 설정하는 type이다. 주석은 x와 y에 변수를 매핑해서 사용하거나 하나하나 표시할 수 있다. 따라서 x, y에 문자열이 표시될 좌표를 설정하거나 벡터나 변수를 매핑하여 주석을 표시할 수 있다.

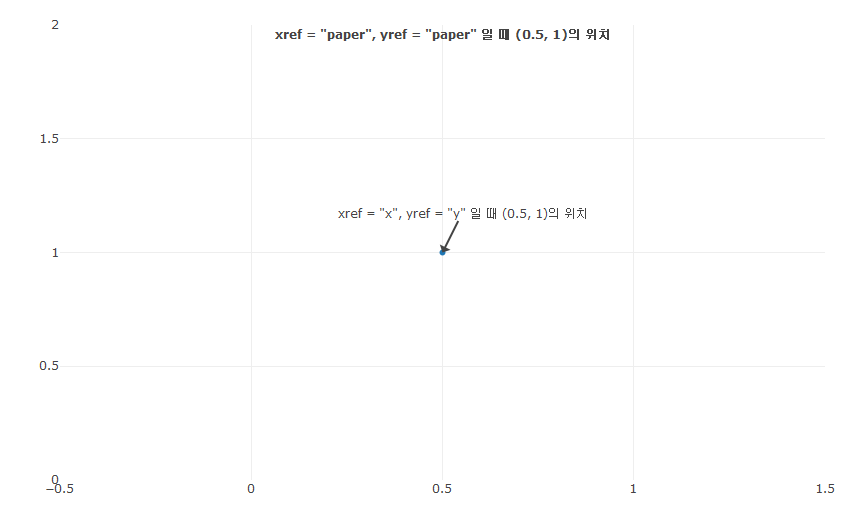
ax와 ay는 x, y에 설정된 좌표에 대한 주석이 실제로 위치할 좌표의 상대적 위치를 설정한다. 이 상대적 위치는 화살표로 표시되는데 ax는 x에 설정된 좌표에서의 거리를 설정하고 ay는 y에 설정된 좌표에서의 거리를 설정한다. ay 설정에 하나 주의할 점은 Y축의 양의 방향은 아래에서 위로 향하지만 ay의 양의 방향은 위에서 아래를 향하기 때문에 아래쪽에 위치한다.

x, y, ax, ay에 의해 결정된 좌표에 text로 설정한 문자열이 표시된다. 이 문자열의 위치 정렬을 설정하는 type이 xanchor와 yanchor이다. xanchor는 ‘auto’, ‘left’, ‘center’, ‘right’ 중에 하나의 값을 가지는데 x에 설정된 좌표값이 전체 문자열의 왼쪽인지, 중앙인지, 오른쪽인지를 설정한다. 또 yanchor는 ‘auto’, ‘top’, ‘middle’, ‘bottom’ 중에 하나의 값을 가지고 y에 설정된 자표값이 전체 문자열의 위쪽인지, 중앙인지, 아래쪽인지를 설정한다.



실행결과 3- . anchor 설정에 따른 문자열 표시

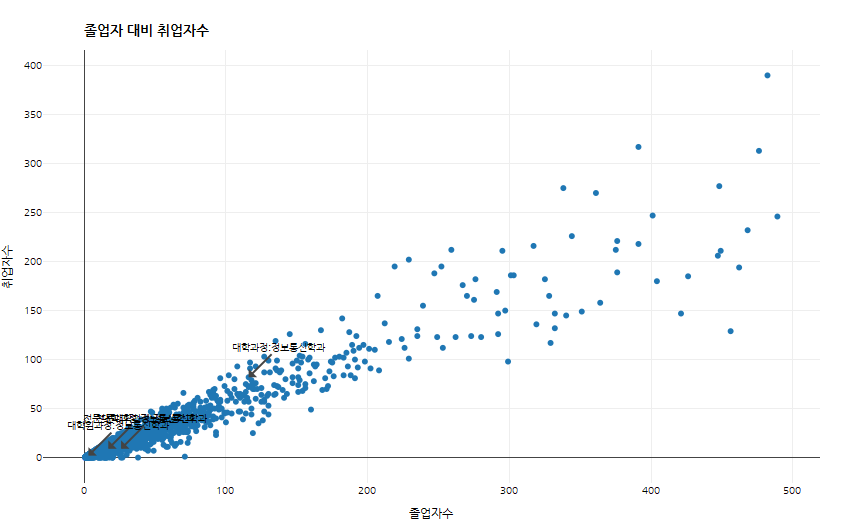
xref와 yref는 x, y, ax, ay의 절대 좌표와 상대 좌표의 참조 기준을 설정한다. xref, yref가 ‘x’, ’y’등의 축으로 설정되면 해당 축의 스케일에 맞게 x, y, ax, ay의 좌표와 거리를 계산하고 ’paper’로 설정되면 전체 플롯 영역을 0부터 1까지로 정규화한 상대적 거리로 계산된다.



실행결과 3- . xref, yref의 설정에 따른 위치 설정

다음의 예는 ‘졸업자 대비 취업자수’ 산점도에서 ’정보통신학과’의 위치를 확인하기 위한 예이다.

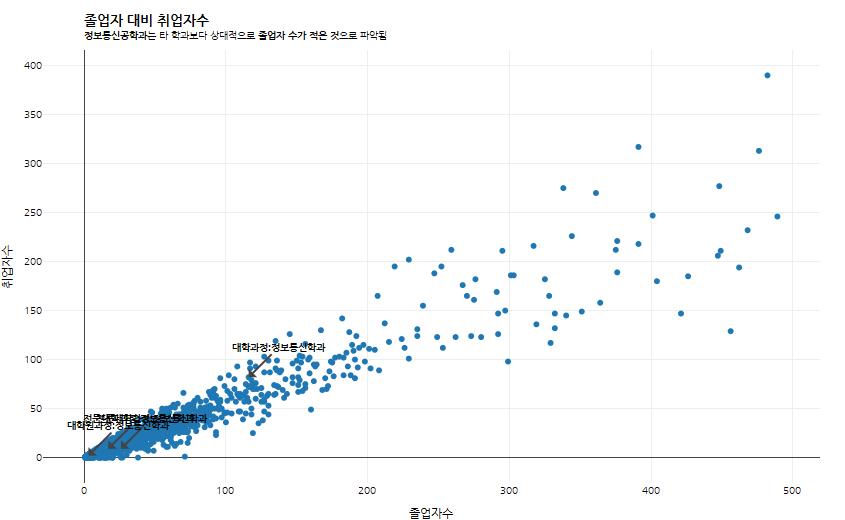
## 정보통신공학과 데이터만 필터링  
df\_annotation <- df\_취업률\_2000 |> filter(학과명 == '정보통신학과') |>   
 select(과정구분, 학과명, 졸업자수, 취업자수)  
  
  
p\_layout\_1 |>   
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>',   
 x = 0.1, xanchor = 'left', yanchor = 'top', size = 15),   
 font = list(family = "나눔고딕", color = 'black', size = 10),   
 ## 정보통신공학과의 과정구분을 text로 매핑  
 annotations = list(text = ~paste0(df\_annotation$과정구분, ':', df\_annotation$학과명),   
 ## X, Y축의 위치를 매핑  
 x = df\_annotation$졸업자수,   
 y = df\_annotation$취업자수,   
 ## 화살표의 위치를 설정  
 ax = 30, ay = -30),   
 margin = margins  
 )



실행결과 3- . 특정 아이템 주석 달기

앞의 예에서 시각화 어디엔가에는 정보통신학과 데이터의 특성을 설명해 주면 효율적일 수 있다. 이를 위해 다음과 같이 제목 아래에 주석을 붙여주고 플로팅 영역을 적절히 조절해주었다.

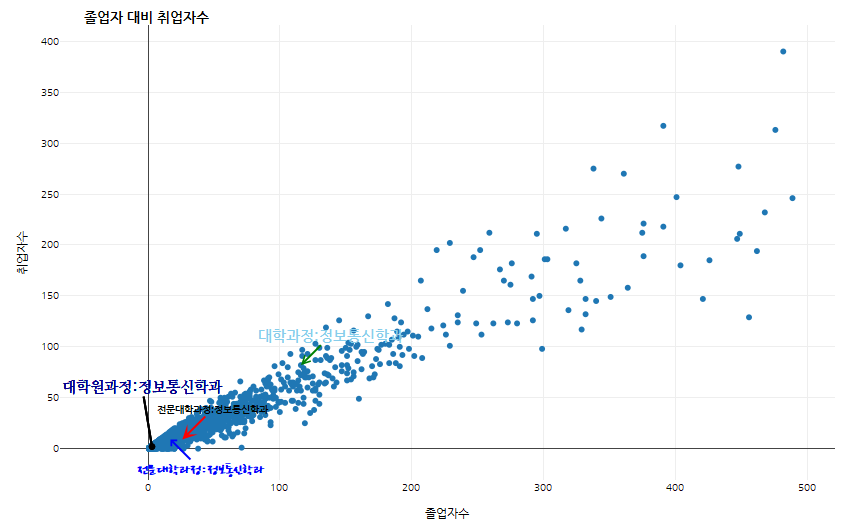
p\_layout\_1 |>   
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>', x = 0.1,   
 xanchor = 'left', yanchor = 'bottom', size = 15),   
 font = list(family = "나눔고딕", color = 'black', size = 10),   
 annotations = list(text = ~paste0('<b>', df\_annotation$과정구분, ':',   
 df\_annotation$학과명, '</b>'),   
 x = df\_annotation$졸업자수,   
 y = df\_annotation$취업자수,   
 ax = 30, ay = -30),   
 margin = margins) |>  
 add\_annotations(xref = 'paper', yref = 'paper',   
 text = '<b>정보통신공학과</b>는 타 학과보다 상대적으로 <b>졸업자 수가 적은 것</b>으로 파악됨',   
 x = 0.05, y = 1.05,  
 showarrow = FALSE)



실행결과 3- . 시각화 세부 설명 문구 추가

앞의 예에서 보면 아래쪽의 정보통신학과는 위치의 차이가 크지 않아 주석들이 겹쳐진다. 이를 수정하기 위해서는 각각의 주석들을 설정하여야 한다. 다음의 예는 각각의 주석을 설정하면서 주석의 화살표 형태를 다르게 설정하였다.

## 네개의 주석중에 세개는 layout의 annotations의 리스트로 설정  
p\_layout\_1 |>   
 layout(title = list(text = '<b>졸업자 대비 취업자수</b>', x = 0.1,   
 xanchor = 'left', yanchor = 'top', size = 15),   
 font = list(family = "나눔고딕", color = 'black', size = 10),   
 ## annotations 리스트 설정  
 annotations = list(  
 ## 첫 번째 주석의 annotations 세부 속성 설정  
 list(text = ~paste0('<b>', df\_annotation$과정구분[1], ':',   
 df\_annotation$학과명[1], '</b>'),   
 x = df\_annotation$졸업자수[1],   
 y = df\_annotation$취업자수[1],   
 ax = 30, ay = -30,   
 arrowhead = 3, arrowcolor = 'red',   
 arrowsize = 1.2  
 ),  
 ## 두 번째 주석의 annotations 세부 속성 설정  
 list(text = ~paste0('<b>', df\_annotation$과정구분[2], ':', df\_annotation$학과명[2], '</b>'),   
 x = df\_annotation$졸업자수[2],   
 y = df\_annotation$취업자수[2],   
 ax = 30, ay = 30,   
 arrowhead = 4, arrowcolor = 'blue',   
 arrowsize = 0.8,   
 font = list(family = '나눔손글씨 펜', size = 15, color = 'blue')  
 ),   
 ## 세 번째 주석의 annotations 세부 속성 설정  
 list(text = ~paste0('<b>', df\_annotation$과정구분[3], ':', df\_annotation$학과명[3], '</b>'),   
 x = df\_annotation$졸업자수[3],   
 y = df\_annotation$취업자수[3],   
 ax = 30, ay = -30,   
 arrowhead = 5, arrowcolor = 'green',   
 arrowsize = 0.8,   
 font = list(family = '나눔고딕', size = 15, color = 'skyblue')  
 )  
 )  
 ) |>  
 ## 네 번째 주석의 annotations을 add\_annotations()로 설정  
 add\_annotations(text = ~paste0('<b>', df\_annotation$과정구분[4], ':', df\_annotation$학과명[4], '</b>'),   
 x = df\_annotation$졸업자수[4],   
 y = df\_annotation$취업자수[4],   
 ax = -10, ay = -60,   
 arrowhead = 6, arrowcolor = 'black',   
 arrowsize = 0.8,   
 font = list(family = '나눔명조', size = 15, color = 'darkblue'))



실행결과 3- . 각각 주석의 세부 설정

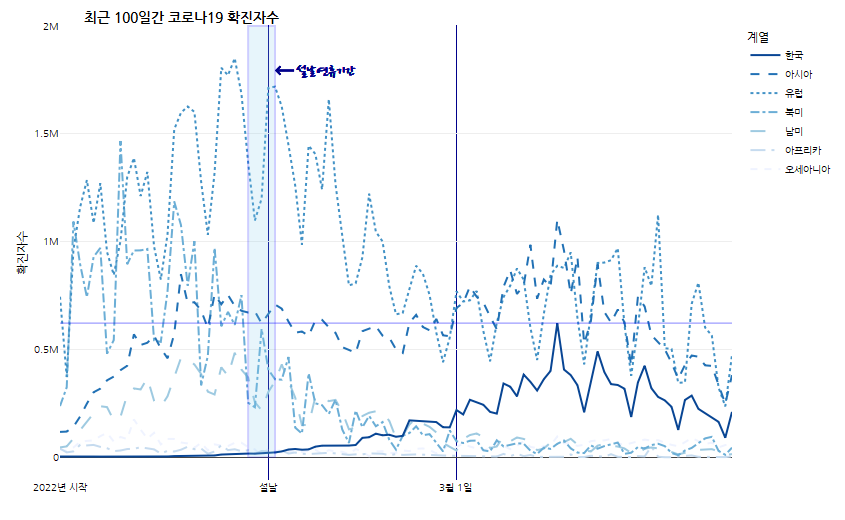
### 도형 설정 : shapes

데이터를 설명하기 위해 넣는 주석은 주로 문자열에 국한된다. 하지만 데이터를 설명하기 위해서는 선, 네모, 원 등의 다양한 도형을 사용할 필요가 있다. 이와 같이 데이터를 설명하기 위한 각종 도형을 사용자가 직접 그리기 위해서는 layout()의 shape type을 설정하여 생성할 수 있다. shape type에서 설정가능한 세부 type은 다음과 같다.

| 속성 | 설명 | 속성 값 | 세부 속성 |
| --- | --- | --- | --- |
| type | 그려질 도형의 형태 설정 | ‘circle’, ‘rect’, ‘path’, ‘line’ |  |
| name | 그려질 도형의 이름 설정 | 문자열 |  |
| x0, x1, y0, y1 | 도형의 X, Y축 시작점(x0, y0)과 끝점(x1, y1) 설정 | 수치 또는 카테고리 좌표 문자열 |  |
| xref, yref | 주석 도형이 그려지는 좌표의 참조 대상 설정 | ‘paper’, X축 또는 Y축 이름 |  |
| xsizemode, ysizemode | X, Y축데 따른 크기 조절 모드 설정 | ‘scaled’, ‘pixel’ |  |
| layer | 그려지는 도형 레이어의 위치를 설정 | ‘above’, ‘below’ |  |
| line | 선 도형의 세부 설정 | 세부 속성들의 리스트 | color, dash, width |
| filcolor | 도형 내부 색 설정 | 컬러 값 |  |

다음의 예는 최근 100일간의 코로나19 확진자수의 그래프에 주요한 시간적 이벤트를 표시한 그래프이다. ‘rect’ 타입의 shape로 설날 연휴기간을 표시하였고 가로선으로 한국 확진자의 최대치를 그려주었다.

p\_layout\_2 |>   
 layout(title = list(text = '<b>최근 100일간 코로나19 확진자수</b>',   
 x = 0.1, xanchor = 'left', yanchor = 'top', size = 15),  
 font = list(family = "나눔고딕", color = 'black', size = 10),  
 legend = list(title = list(text = '계열', side = 'top')),  
 xaxis = list(title = list(text = NULL),   
 tickmode = 'array',  
 ticktext = c('2022년 시작', '설날', '3월 1일'),  
 tickvals = c('2022-01-01', '2022-02-01', '2022-03-01'),   
 gridwidth = 1, gridcolor = 'darkblue'  
 ),  
 yaxis = list(title = '확진자수'),   
 shapes = list(  
 ## rect 타입을 위한 shapes 세부 속성 설정  
 list(type = 'rect',  
 x0 = as.Date('2022-01-29'), x1 = as.Date('2022-02-02'),  
 y0 = 0, y1 = 2000000, fillcolor = "skyblue",  
 line = list(color = "blue"), opacity = 0.2),   
 ## line 타입을 위한 shapes 세부 속성 설정  
 list(type = 'line', x0 = 0, x1= 1,   
 y0 = max(df\_covid19\_100[df\_covid19\_100$iso\_code == 'KOR', 6]),   
 y1 = max(df\_covid19\_100[df\_covid19\_100$iso\_code == 'KOR', 6]),   
 xref = 'paper', yref = 'y',   
 line = list(color = 'blue'), opacity = 0.3  
 )  
 ),  
 ## 주석 설정  
 annotations = list(text = '<b>설날연휴기간</b>',   
 x = '2022-02-02', xref = 'x',   
 y = 0.9, yref = 'paper',   
 ax = 50, ay = 0,   
 arrowhead = 4, arrowcolor = 'darkblue',   
 arrowsize = 0.8,   
 font = list(family = '나눔손글씨 펜',   
 size = 15, color = 'darkblue')  
 )  
 )



실행결과 3- . layout을 사용한 도형 설정

## 서브 플롯 : subplot()

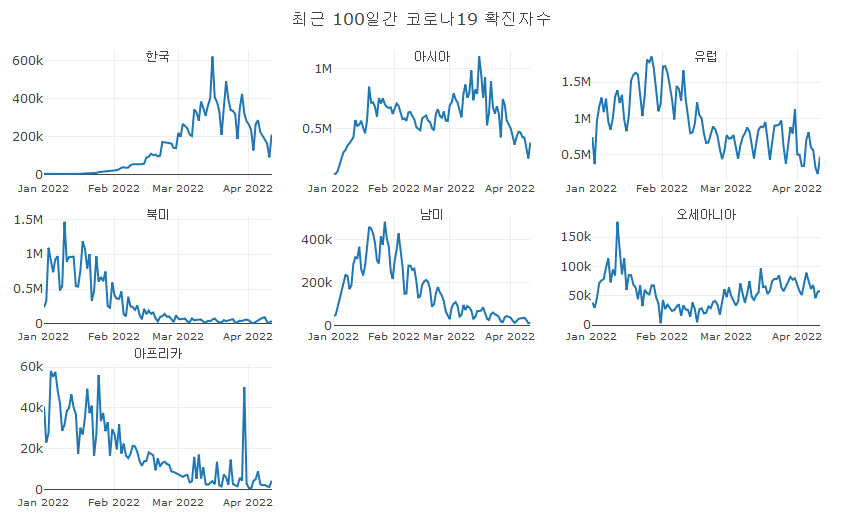
데이터를 시각화할 때 하나의 시각화에 여러 개의 데이터를 표시하는 경우 각각의 데이터 trace들이 너무 많아지거나 일부 구간에 중복되면 데이터의 정확한 확인이 어렵다. 이럴때는 각각의 데이터 trace들을 따로 떼서 작은 시각화를 여러개 그려줌으로써 해결할 수 있다. 이렇게 여러개의 plotlt 객체를 모아서 하나의 plotly 시각화로 만들어주는 함수가 subplot()이다. 앞서 plotly의 설명 과정에서 여러개의 시각화가 모여서 그려진 사례는 모두 subplot()과 ’annotation’을 사용해서 만들었다.

subplot()의 주요 사용법은 다음과 같다.

subplot(…, nrows = 1, widths = NULL, heights = NULL, margin = 0.02, shareX = FALSE, shareY = FALSE, titleX = shareX, titleY = shareY, which\_layout = “merge”)  
- … : plotly나 ggplot2 객체의 이름, 리스트, tibble  
- nrows : subplot의 행의 개수  
- widths : 각각의 subplot의 0부터 1사이의 상대적 열 너비  
- heights : 각각의 subplot의 0부터 1사이의 상대적 열 높이  
- margin : 0부터 1사이의 단일 수치나 4개의 수치(왼쪽, 오른쪽, 위, 아래)의 여백  
- shareX : X축을 공유할지에 대한 논리값  
- shareY : Y축을 공유할지에 대한 논리값  
- titleX : X축의 제목을 남길지에 대한 논리값  
- titleY : Y축의 제목을 남길지에 대한 논리값  
- which\_layout : 어떤 플롯의 레이아웃에 적용할지 결정, 기본값은 “merge”로 뒤 플롯의 레이아웃 옵션이 앞 옵션보다 우선함을 의미

subplot()은 subplot으로 그릴 각각의 plotly 객체나 ggplot2를 먼저 생성하고 이들 객체들을 subplot()을 사용하여 묶어 그려주는 형태로 사용된다. 이 과정에서 shareX, shareY를 사용하여 X, Y축의 공유를 설정하거나 nrows를 사용하여 subplot()의 행 개수를 설정할 수 있다. 또 각각의 서브 플롯이 겹쳐지지 않도록 margin을 설정할 수 있다.

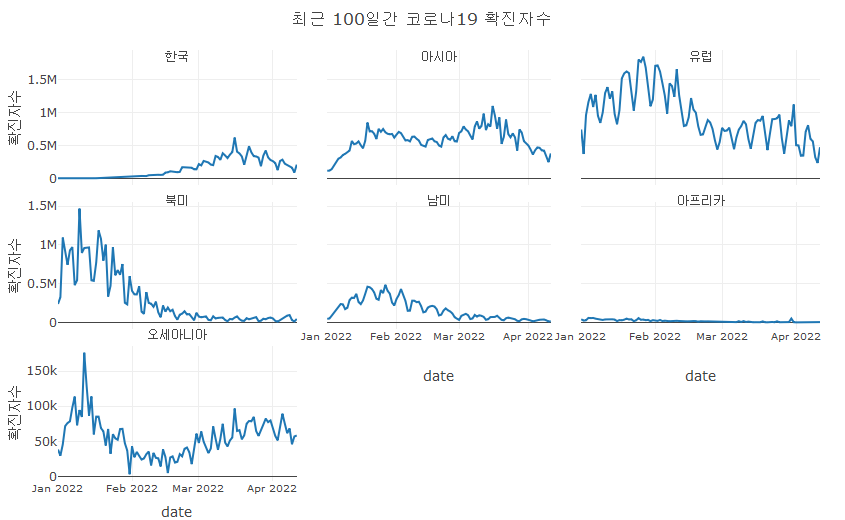
## 서브플롯 생성을 위한 기본 plotly 객체 생성  
p\_line\_wide <- df\_covid19\_100\_wide |> plot\_ly()  
  
## 첫 번째 서브 플롯 생성  
p1 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_한국, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 두 번째 서브 플롯 생성  
p2 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_아시아, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 세 번째 서브 플롯 생성  
p3 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_유럽, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 네 번째 서브 플롯 생성  
p4 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_북미, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 댜섯 번째 서브 플롯 생성  
p5 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_남미, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 여섯 번째 서브 플롯 생성  
p6 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,  
 y = ~확진자\_오세아니아, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 일곱 번째 서브 플롯 생성  
p7 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_아프리카, color = I('#1f77b4')) |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## subplot 설정  
subplot(  
 p1 |>   
 ## layout으로 첫 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 ## 주석의 위치와 주석 text 설정  
 x = 0.5 , y = 1.02, text = "한국",  
 ## 화살표는 표시하지 않음  
 showarrow = F,   
 ## X, Y의 참조 위치는 'paper'  
 xref='paper', yref='paper',  
 ## X방향 정렬은 'center'  
 xanchor = 'center')),  
 p2 |>   
 ## layout으로 두 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "아시아",  
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 p3 |>   
 ## layout으로 세 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "유럽",  
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 p4 |>   
 ## layout으로 네 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "북미",   
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 p5 |>   
 ## layout으로 다섯 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "남미",  
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 p6 |>   
 ## layout으로 여섯 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "오세아니아",  
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 p7 |>   
 ## layout으로 일곱 번째 서브 플롯의 주석 표시   
 layout(annotations = list(  
 x = 0.5 , y = 1.02, text = "아프리카",   
 showarrow = F, xref='paper', yref='paper',  
 xanchor = 'center')),  
 ## 서브플롯은 3개의 열로 설정  
 nrows = 3,  
 ## 서브플롯간의 여백 설정  
 margin = 0.04) |>   
 ## 범례는 제거  
 layout(showlegend = FALSE,  
 ## 전체 제목 설정  
 title = '최근 100일간 코로나19 확진자수',  
 ## 전체 여백 설정  
 margin = margins)



실행결과 3- . 서브 플롯 생성

앞의 subplot()에서의 예는 각각의 plotly 객체를 각각 만든 후에 이들 plotly 객체들을 subplot()으로 하나의 플롯으로 만들어 주었다. 서브 플롯이 몇개 되지 않을때는 이렇게 각각 만든 plotly 객체를 묶어주는 것이 가능하겠지만 서브 플롯이 많을 때는 이러한 작업이 매우 어려워진다. 특히 긴 형태의 데이터프레임의 경우 앞서와 같이 서브 플롯을 만들어 주기 위해서는 각각의 서브 플롯마다 데이터를 필터링하여 만들어야하기 때문에 매우 번거로운 작업이 수반된다. 이럴 경우는 group\_by()로 데이터프레임을 그룹화하고 .을 사용하여 각각의 그룹화된 데이터 그룹별로 plotly 객체를 만드는 방식으로 간단히 서브 플롯을 만들어 줄 수 있다. 이 때 do를 사용하여 각각의 그룹화된 데이터 그룹별 plotly 객체를 만드는 코드를 적용하여야 한다.

## 긴 형태의 100일 코로나19 데이터에서  
df\_covid19\_100 |>  
 ## 국가명으로 그룹화  
 group\_by(location) |>  
 ## 그룹화한 각각의 데이터 그룹들에 적용할 코드 설정  
 do(  
 ## 각 그룹화한 데이터를 사용해 plotly 객체 생성   
 p = plot\_ly(.) |>   
 ## line 모드의 스캐터 trace 추가  
 add\_trace(type = 'scatter', mode = 'lines',  
 ## X, Y축에 변수 매핑, color를 설정  
 x = ~date, y = ~new\_cases, color = I('#1f77b4')) |>  
 ## layout으로 X, Y축을 설정  
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')),  
 ## 국가명을 주석으로 표시  
 annotations = list(x = 0.5 , y = 1.02, text = ~location,   
 showarrow = F, xref='paper',   
 yref='paper', xanchor = 'center'))  
 ) |>  
 ## 생성된 plotly 객체들을 subplot 생성  
 subplot(nrows = 3, shareX = TRUE, shareY = TRUE) |>  
 ## 생성된 subplot의 layout 설정  
 layout(showlegend = FALSE,   
 title = '최근 100일간 코로나19 확진자수',  
 margin = margins)

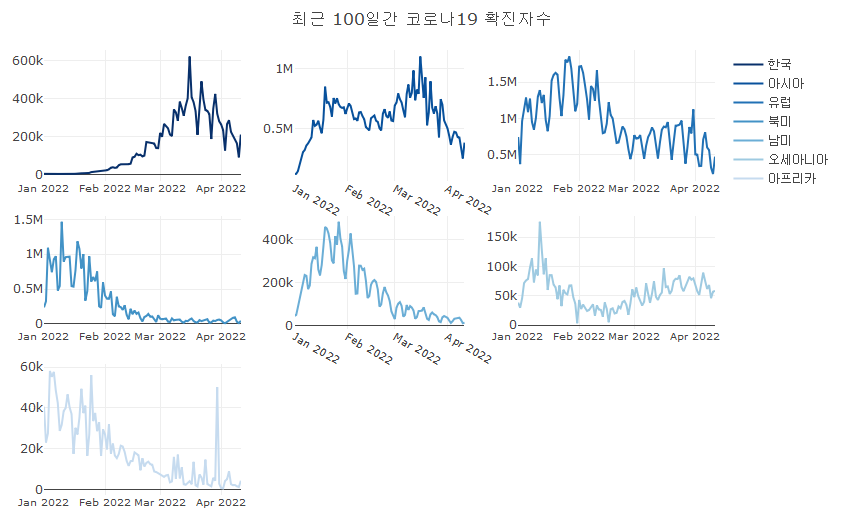


실행결과 3- . group\_by()를 사용한 subplot 생성

### 서브 플롯 범례 설정

앞선 예에서는 서브 플롯으로 사용되는 플롯들의 trace의 색을 강제로 동일하게 맞추고 annotations를 사용하여 각각의 플롯의 제목을 설정함으로써 각 플롯 데이터를 설명하였다. 하지만 일반적으로 이들을 범례로써 표현하는데 subplot()에 표시되는 범례는 세부 플롯마다 범례가 표시되는 것이 아니라 전체 플롯의 범례로 모아서 하나의 범례로 표시된다. 모아진 범례를 표시하거나 표시하지 않는 것을 결정하는 subplot()에서 결정하지 않고 layout()의 showlegend 속성을 사용해 결정할 수 있다. 또 다른 방법은 show\_legend()와 hide\_legend() 함수를 사용하면 범례의 사용을 결정할 수 있다.

## 첫 번째 서브 플롯 생성  
p1 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_한국, name = '한국') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 두 번째 서브 플롯 생성  
p2 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_아시아, name = '아시아') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 세 번째 서브 플롯 생성  
p3 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_유럽, name = '유럽') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 네 번째 서브 플롯 생성  
p4 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_북미, name = '북미') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 댜섯 번째 서브 플롯 생성  
p5 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_남미, name = '남미') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 여섯 번째 서브 플롯 생성  
p6 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,  
 y = ~확진자\_오세아니아, name = '오세아니아') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## 일곱 번째 서브 플롯 생성  
p7 <- p\_line\_wide |>   
 add\_trace(type = 'scatter', mode = 'lines', x = ~date,   
 y = ~확진자\_아프리카, name = '아프리카') |>   
 layout(title = list(title = NULL),  
 xaxis = list(tickfont = list(size = 10)),   
 yaxis = list(title = list(text = '확진자수')))  
  
## subplot 설정  
subplot(  
 p1, p2, p3, p4, p5, p6, p7,  
 ## 서브플롯은 3개의 열로 설정  
 nrows = 3,  
 ## 서브플롯간의 여백 설정  
 margin = 0.04) |>   
 ## 범례는 제거  
 layout(showlegend = TRUE,  
 ## 전체 제목 설정  
 title = '최근 100일간 코로나19 확진자수',  
 ## 전체 여백 설정  
 margin = margins)



실행결과 3- . 서브 플롯 생성

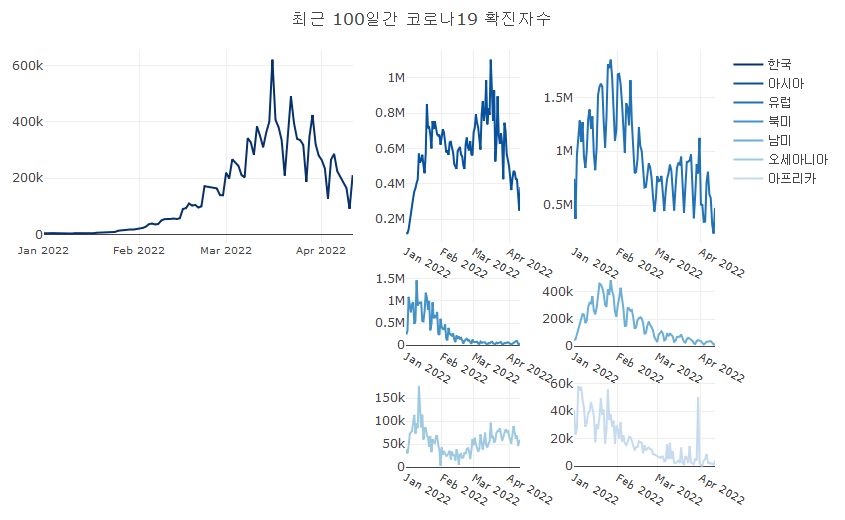
### 서브 플롯 위치 배치와 편집

앞서 살펴본 바와 같이 subplot()은 서브 플롯안에 포함되는 플롯들의 위치를 행과 열의 설정을 통해 2차원 형태로 배치할 수 있다. 행의 수를 설정하기 위해서는 nrows 속성을 사용하지만 열의 수를 설정할 수는 없다는 점을 주의해야 한다. 기본적으로 서브 플롯에 포함되는 플롯들의 크기는 서브 플롯에 설정된 행과 열의 수에 따라 전체 높이와 너비를 동일한 비율로 공유한다. 하지만 이 비율은 heights, widths를 사용하여 변경할 수 있는데 각각의 플롯의 크기를 변경함으로써 플롯들의 위치와 크기를 사용자가 원하는 대로 편집할 수 있다.

서브 플롯을 원하는 대로 배치하고 크기를 설정하기 위해서는 서브 플롯이 배치되는 방향을 잘 고려해야 한다. 앞서 살펴본 서브 플롯의 경우 p1부터 p7까지를 순서대로 subplot()의 매개변수로 설정했다. 그리고 nrows를 3으로 설정했기 때문에 7개의 플롯을 표시하기 위해서는 자동적으로 열의 수가 3으로 설정된다. 따라서 전체 플롯을 9개로 등분하고 각각의 매개변수 호출 순서대로 열 방향으로 배치하게 된다. 전체 플롯을 9등분 하기 때문에 2개의 등분은 빈 공간으로 남게 된다. 만약 마지막 행의 빈 공간을 적절히 사용하기 위해서는 plotly\_empty()를 사용해서 적절한 위치에 비어있는 plotly 객체를 넣어 주는 방법을 사용한다.

앞의 예에서는 1행에 p1(한국), p2(아시아), p3(유럽), 2행에 p4(북미), p5(남미), p6(오세아니아)가 배치되고 마지막 행에는 하나 남은 p7(아프리카)가 1열에 배치되며 나머지 2열은 비게 된다. 만약 p1 플롯을 좀 부각시키기 위해서는 다음과 같이 설정할 수 있다.

subplot(  
 p1, p2, p3, plotly\_empty(), p4, p5, plotly\_empty(), p6, p7,  
 ## 서브플롯은 3개의 열로 설정  
 nrows = 3,  
 ## 서브플롯간의 여백 설정  
 heights = c(0.5, 0.25, 0.25),   
 widths = c(0.5, 0.25, 0.25),   
 margin = 0.04) |>   
 ## 범례는 제거  
 layout(showlegend = TRUE,  
 ## 전체 제목 설정  
 title = '최근 100일간 코로나19 확진자수',  
 ## 전체 여백 설정  
 margin = margins)

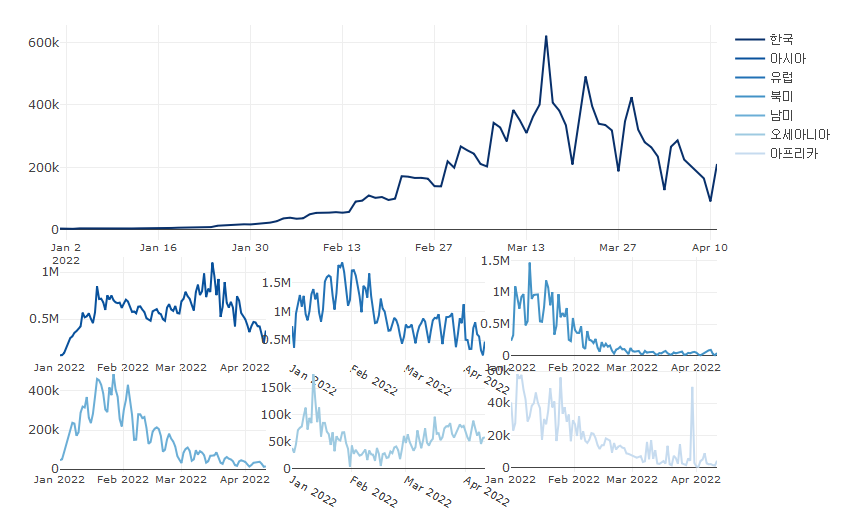


실행결과 3- . 서브 플롯의 크기와 배치 설정

앞의 서브 플롯을 보면 한국 플롯이 크게 표현되어 강조되는 효과를 내기는 했지만 같은 행, 같은 열에 있는 플롯들은 한쪽으로 길게 표시되어 보기에 어색해 보인다. 이를 해결하기 위해서는 subplot()을 중첩해서 해결할 수 있다. subplot()을 중첩하여 사용한다는 것은 subplot()으로 완성된 서브 플롯을 다시 subplot()으로 배열한다는 것이다.

다음은 한국을 위에 길게 표현하고 아래에 6개의 플롯을 2행으로 배열하는 예이다. 다음의 예에서는 p1(한국)과 p2부터 p7까지의 플롯을 subplot()으로 하나의 plotly로 생성하여 이 두개의 플롯을 다시 subplot()으로 붙여주는 코드이다.

subplot(  
 ## 아래의 nrows가 2이기 때문에 맨 위 열에 p1 하나를 위치시킴  
 p1,   
 ## subplot()으로 p2부터 p7까지를 묶어 하나의 플롯으로 만듬  
 subplot(p2, p3, p4, p5, p6, p7,  
 ## 서브플롯은 2개의 열로 설정함으로써 2행 3열 서브 플롯 생성  
 nrows = 2),   
 ## 전체 서브 플롯은 2열로 구성  
 nrows = 2  
)

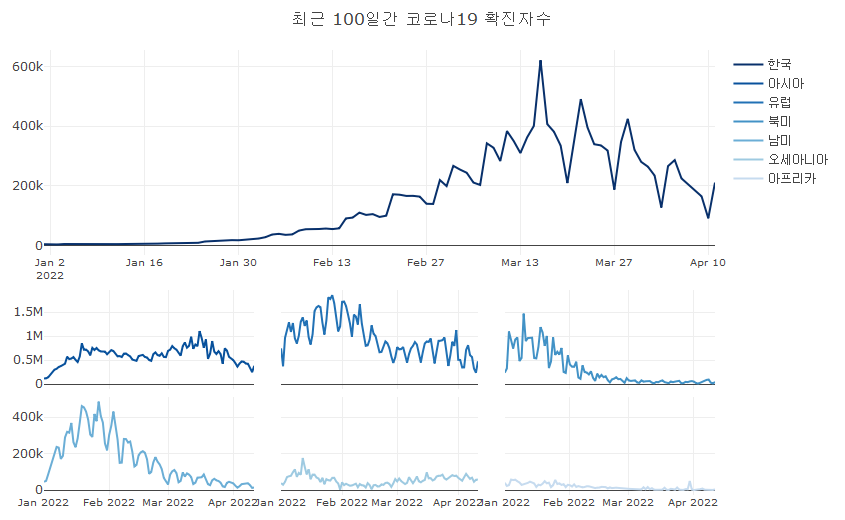


실행결과 3- . 크기와 배치를 수정한 서브 플롯

### 축 공유

subplot()으로 완성되는 시각화는 layout()으로 세부 속성의 설정이 가능하다. 하지만 subplot()에서 사용되는 속성값이 몇 개 있는데 그 중 꼭 알아두어야 하는 속성 값이 shareX와 shareY이다. 이 두 속성은 서브 플롯에 표시되는 각각의 플롯의 X, Y축을 공유할지를 결정하는 속성이다. 이 속성값이 shareX값이 TRUE로 설정되면 수직으로 같은 열에 표시된 서브플롯들의 X축은 모두 스케일로 고정된다. 하지만 각각의 열들의 X축은 모두 각각의 열에 적합한 축으로 표현되기 때문에 열 별로는 X축의 스케일이 달라질 수 있다. 마찬가지로 shareY가 TRUE로 설정되면 전체 플롯의 같은 행에 표현된 서브 플롯은 같은 스케일의 Y축을 공유한다.

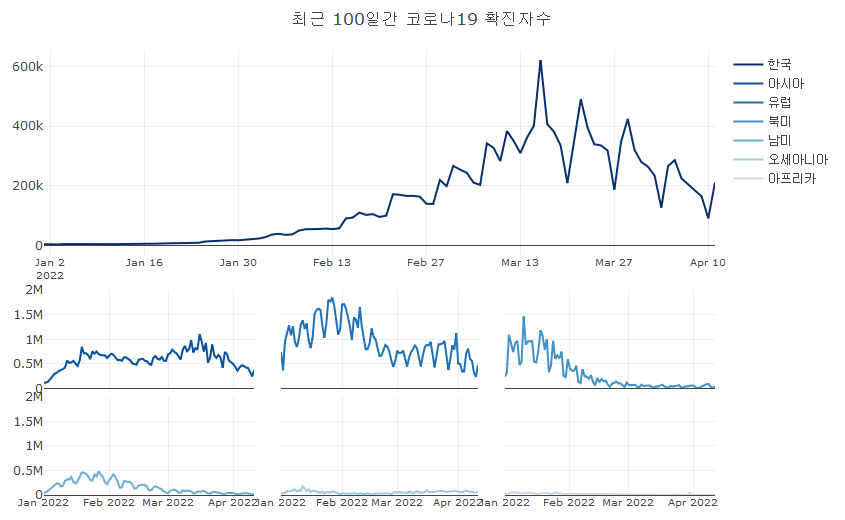
subplot(  
 p1,   
 subplot(p2, p3, p4, p5, p6, p7,  
 nrows = 2,   
 ## shareX, shareY를 TRUE로 설정하여 축 공유  
 shareX = TRUE, shareY = TRUE),   
 nrows = 2, margin = 0.04  
 ## shareX, shareY를 TRUE로 설정하여 축 공유  
 ) |>   
 layout(showlegend = TRUE,   
 title = '최근 100일간 코로나19 확진자수',  
 margin = margins)



실행결과 3- . shareX, shareY가 설정된 subplot

만약 모든 서브 플롯의 X, Y축의 스케일을 동일하게 표현하기 위해서는 각각의 서브 플롯의 X, Y축의 range 속성을 설정함으로써 가능하다.

## 서브 플롯들의 Y축 범위 설정  
p2 <- p2 |> layout(yaxis = list(range = c(0, 2000000)))  
  
p3 <- p3 |> layout(yaxis = list(range = c(0, 2000000)))  
  
p4 <- p4 |> layout(yaxis = list(range = c(0, 2000000)))  
  
p5 <- p5 |> layout(yaxis = list(range = c(0, 2000000)))  
  
p6 <- p6 |> layout(yaxis = list(range = c(0, 2000000)))  
  
p7 <- p7 |> layout(yaxis = list(range = c(0, 2000000)))  
  
subplot(  
 p1,   
 subplot(p2, p3, p4, p5, p6, p7,  
 nrows = 2,   
 shareX = TRUE, shareY = TRUE),   
 nrows = 2, margin = 0.04  
 ## shareX, shareY를 TRUE로 설정하여 축을 공유  
 ) |>   
 layout(showlegend = TRUE,   
 title = '최근 100일간 코로나19 확진자수',  
 margin = margins)



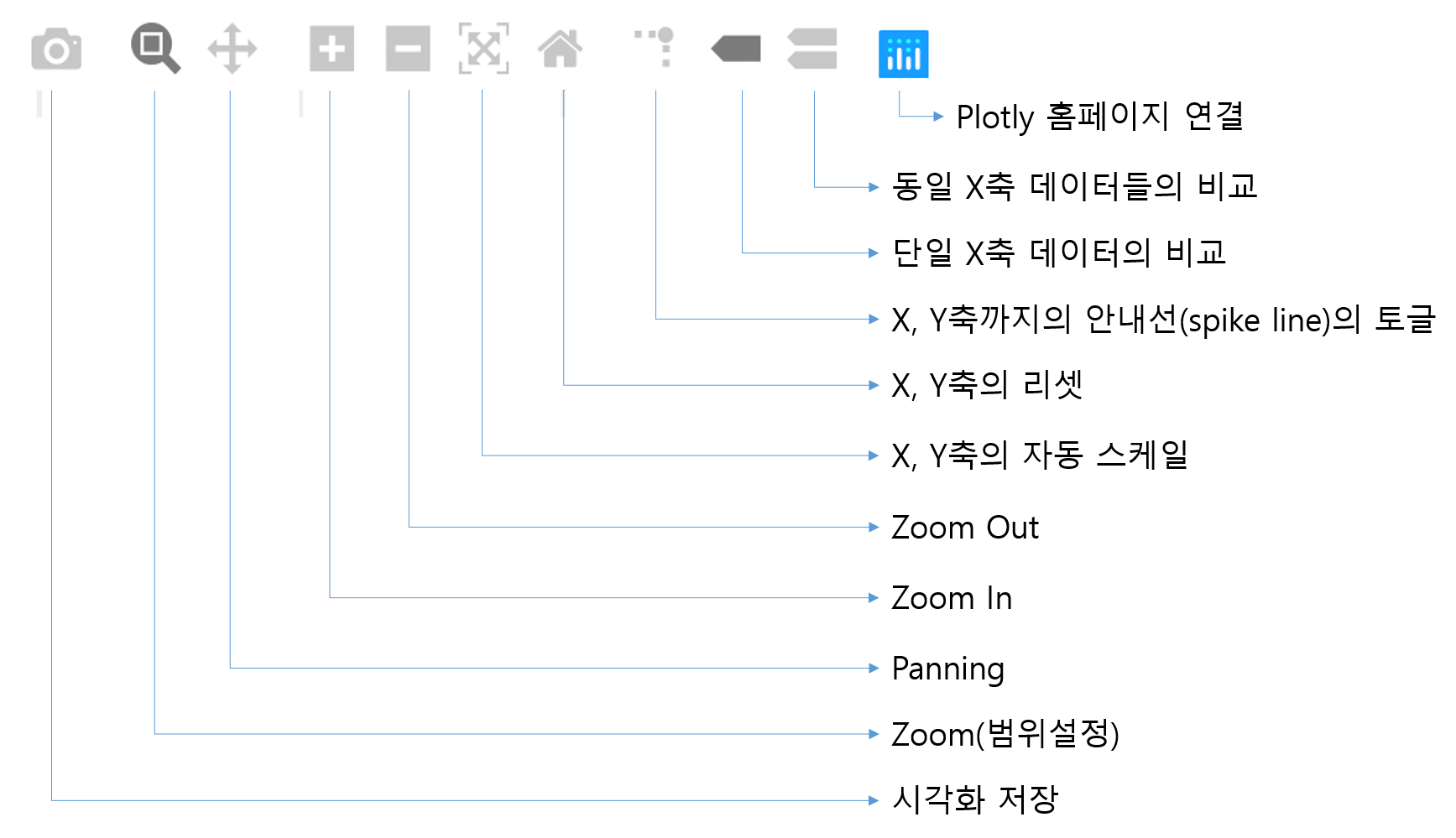
실행결과 3- . 축 범위가 설정된 subplot

# plotly 시각화 사용하기

ggplot2와 같은 정적 시각화는 시각화 그래프를 만든 이후에는 더 이상의 편집이 불가하다. 따라서 시각화를 일부를 변경하기 위해서는 다시 코딩해서 만들어야하는 불편함이 따른다. 특히 특정 위치의 데이터 값을 확인하거나 특정 구간 데이터를 줌인하기 위해서도 다시 코딩해야하는데 단지 한번의 데이터 확인을 위해 코드를 고쳐야 한다는 것은 큰 어려움이 따른다. 따라서 데이터 분석 과정에서 첫 번쨰로 수행하는 탐색적 데이터 분석(EDA : Exploratory Data Analysis)에 적절한 시각화 방법이 아니다. 반면 plotly와 같은 동적 시각화에서는 특징적 데이터 값의 확인, 줌인, 줌 아웃, 특정 데이터만의 표기 등과 같은 탐색적 데이터 분석에 필수적인 기능을 제공한다. 따라서 시각화를 완성한 이후에도 그 사용법을 잘 알아둘 필요가 있다. 당연히 plolty에도 완성된 시각화에서 제공하는 다양한 기능이 있다.

## modebar의 사용

plotly가 시각화 사용자와의 상호작용을 위한 주요 기능을 제공하는 메뉴가 ’modebar’이다. ’modebar’는 plotly가 실행되는 R-Studio나 웹 브라우저에서 오른쪽 상단에 나타나는 버튼 메뉴를 말한다. 기본적으로 설정되는 ’modebar’는 다음의 그림과 같이 8개의 기능을 버튼을 통해 제공한다.

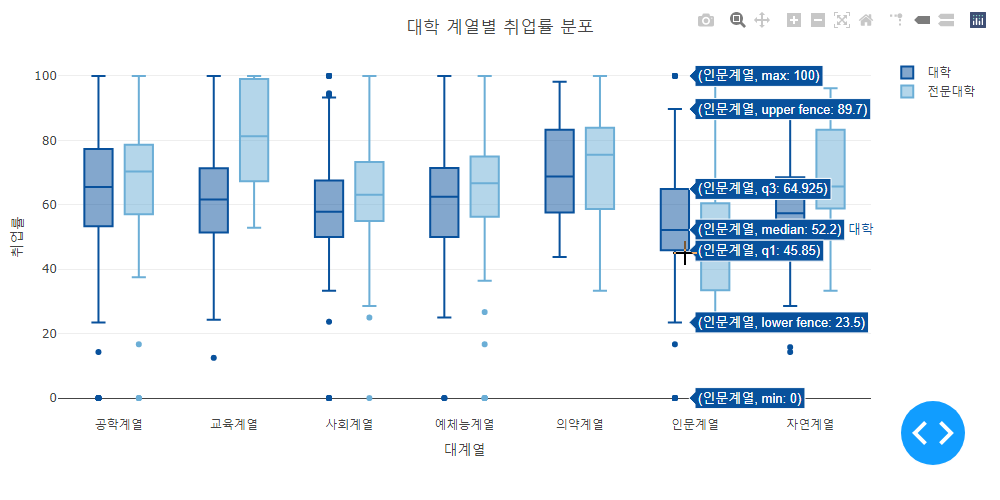


modebar 버튼과 기능

## 마우스를 사용한 데이터 확인

plotly 시각화에서 가장 쉽게 사용하는 기능은 마우스를 사용하여 해당 위치의 데이터 정보를 확인하는 기능이다. plotly 객체로 생성된 시각화 위에 표현된 각 trace들은 자체 데이터를 JSON의 형태로 포함하고 있기 때문에 마우스 포인터를 trace위에 위치시키면 해당 trace의 정보가 표시된다.

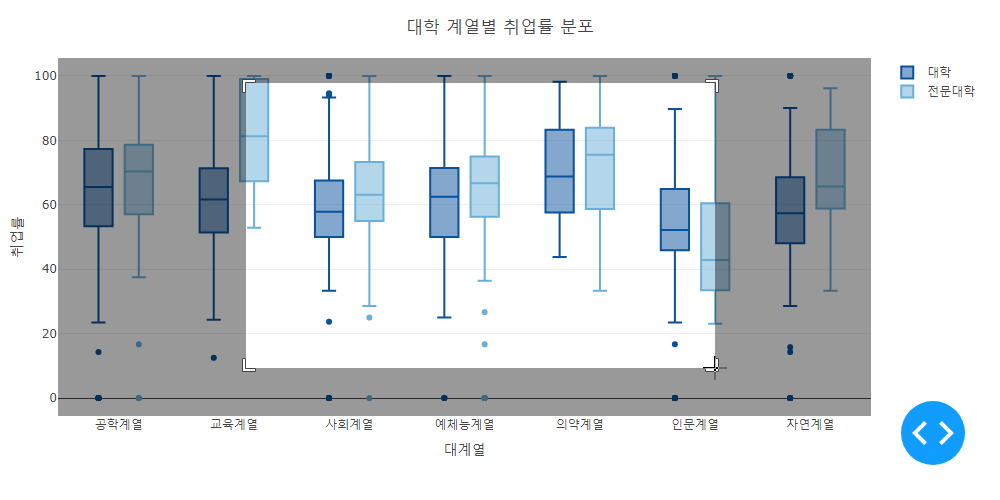
마우스 포인터를 사용하여 표시되는 기본 정보는 X, Y축에 매핑된 정보와 trace에 설정되거나 매핑된 name 속성이 표시된다. 이 정보를 변경하기 위해서는 두가지 방법이 있는데 앞선 plotly 시각화 생성에서 설명한 호버(hover)의 설정인 hoverinfo를 사용하는 방법과 hovertemplate를 사용하는 방법이다. hoverinfo(실행결과 2-\*)를 사용하는 방법이 보다 간단하고 쉬운 반면 hovertemplate(실행결과 2-\*)는 표시되는 정보를 세부적으로 설정할 수 있다는 장점이 있다.



박스 trace의 정보 표시

## 마우스 드래그를 통한 줌

plotly로 완성된 시각화에서 trace가 나타나는 plotting area에서 마우스를 클릭한 상태로 드래그하면 다음 그림과 같이 줌 인 영역을 선택할 수 있다. 이렇게 영역을 선택한 후에 마우스 클릭을 놓으면 해당 부분이 줌인 되어 표시되게 된다. 만약 다시 처음의 상태로 돌아가려면 모드바의 집 아이콘인 ‘Reset Axes’ 버튼을 사용한다.



마우스 드래그를 통한 줌

## 마우스 휠을 사용한 줌

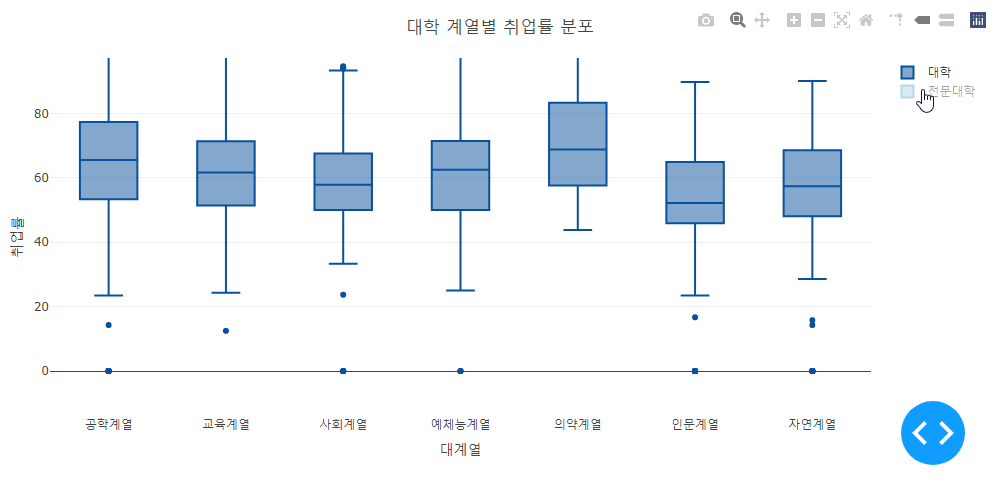
줌인과 줌아웃을 지원하는 많은 프로그램에서는 마우스 휠을 사용하는 프로그램이 많다. plotly도 마우스 휠을 사용하여 줌인과 줌아웃을 사용할 수 있는데 이 기능은 기본적으로 지원하지 않고 config()를 사용하여 설정해야 한다. config()는 plotly 시각화가 사용자와의 상호작용을 지원하는 기능을 조절하고 설정하는 다양한 속성을 조절하는 함수이다. config()에서 설정할 수 있는 주요 속성은 다음과 같다.[[2]](#footnote-43)

| 속성 | 설명 | 속성값 | 세부 속성 |
| --- | --- | --- | --- |
| staticPlot | 정적 시각화로 설정 | 논리값 |  |
| editable | 시각화를 편집할 수 있는지를 설정 | 논리값 |  |
| edit | editable이 TRUE인 경우 편집 가능한 세부 속성 | 세부 속성들의 리스트 | annotationPosition, annotationText, axisTitleText, legendPosition, legendText, titleText |
| scrollZoom | 마우스 휠을 줌으로 사용할지를 설정 | ‘cartesian’, ‘gl3d’, ‘geo’, ‘mapbox’, TRUE, FALSE |  |
| doubleClick | 마우스 더블 클릭을 어떤 기능으로 사용할지 설정 | FALSE, ‘reset’, ‘autosize’, ‘reset+autosize’ |  |
| showTips | 팁(호버)를 사용할지 설정 | 논리값 |  |
| displayModeBar | 모드바의 표시 방법 설정 | ‘hover’, TRUE, FALSE |  |
| modeBarButtonsToRemove | 모드바의 버튼 중 없앨 버튼 설정 | 버튼 ID 문자열 |  |
| modeBarButtonsToAdd | 모드바에 포함시킬 버튼 설정 | 버튼 ID 문자열 |  |
| toImageButtonOptions | 이미지 저장 버튼의 설정 |  |  |
| displaylogo | 모드바의 plotly 로고를 표시할지 설정 | 논리값 |  |

df\_취업률\_2000 |>   
 plot\_ly() |>   
 add\_trace(type = 'box', data = df\_취업률\_2000 |> filter(과정구분 == '대학과정'),  
 x = ~대계열, y = ~취업률, name = '대학') |>   
 add\_trace(type = 'box', data = df\_취업률\_2000 |> filter(과정구분 == '전문대학과정'),   
 x = ~대계열, y = ~취업률, name = '전문대학') |>   
 layout(boxmode = "group",   
 title = list(text = '대학 계열별 취업률 분포'),   
 margin = margins) |>  
 ## 마우스 휠을 줌으로 사용하기 위해 속성 설정  
 config(scrollZoom = TRUE)

## 축의 이동

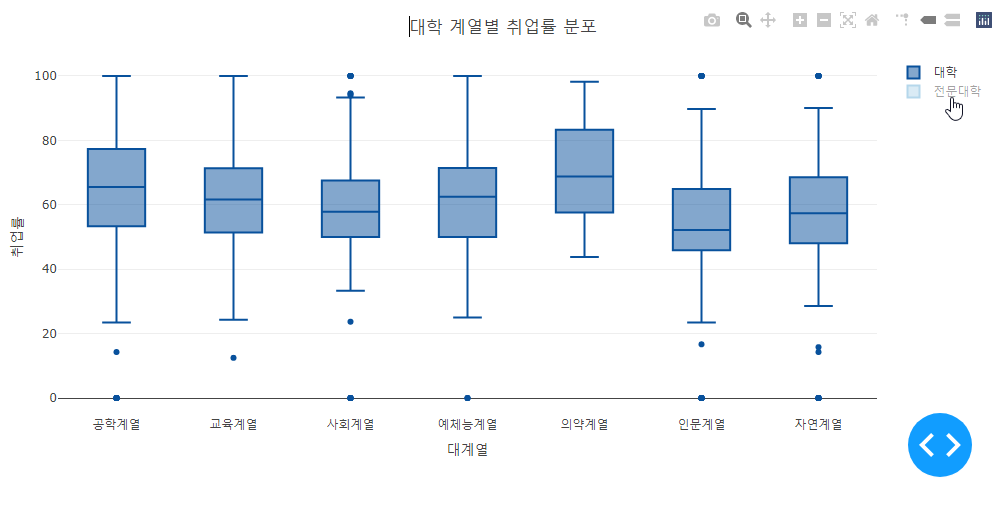
plotly 시각화에서 마우스를 사용한 기능중에 또 하나가 축의 이동이다. plotly의 플롯 영역(plotting area)에서 마우스를 클릭하고 드래그를 하면 줌인의 기능을 위한 사각형이 만들어지지만 X, Y축의 위치에서 마우스를 클릭하고 드래그하면 축을 이동시킬수 있다. 따라서 줌인한 후에 축을 이동시키면 사용자가 자세히 보기를 원하는 지역의 데이터를 확인할 수 있다.



축의 이동

## 범례의 사용

plotly에서 범례는 단순히 trace들의 이름과 표현방식을 매핑해주는 역할을 넘어 trace들의 표시를 조절할 수 있는 기능이 있다. 범례의 아이템을 클릭하면 해당 trace가 표시를 토글하는 역할을 한다. 따라서 여러 데이터 trace중에 특정한 trace만을 확인하기 위해서 해당 trace만 남기고 다른 trace의 표시를 꺼둘수 있는 기능이 있다. 또 legendgroup으로 그룹화된 범례는 해당 그룹의 아이템의 클릭만으로 그 그룹의 전체 trace를 켜거나 꺼둘 수 있다.



범례의 토글

# plotly 시각화 보여주기

R에서 만들어진 데이터 시각화를 보여주기 위해서 가장 많이 사용되는 방법은 jpg, pdf 형태의 이미지로 만들어서 인포그래픽이나 보고서로 만들어서 사용한다. 하지만 이렇게 사용되면 구지 동적 시각화를 만들 이유가 없다. 따라서 동적 시각화의 기능을 충분히 사용하기 위해서는 사용자와의 상호작용이 가능한 매체를 통해 제공해야 한다.

## 정적 이미지로 저장하기

plotly로 만든 시각화를 정적 이미지로 만들기 위해서는 앞서 설명한 모드바의 카메라 아이콘인 ‘download plot as a png’ 버튼을 사용하면 간단히 저장할 수 있다. 저장되는 이미지는 plotly의 다양한 기능을 사용하여 편집한 형태로 저장되기 때문에 하나의 plotly 시각화를 만들어서 탐색적 데이터 분석을 수행한 후 결과를 이미지로 저장할 수 있다. 또 config()의 속성을 사용하여 저장되는 파일 형식이나 이미지 사이즈를 설정할 수 있다. 저장되는 이미지 파일 형식은 픽셀 이미지 형식으로 많이 사용되는 ‘png’, ’jpeg’와 구글에서 개발한 ’webp’를 지원하며 벡터 이미지 형식으로 ’svg’를 지원한다.

save\_fig <- subplot(  
 p1,   
 subplot(p2, p3, p4, p5, p6, p7,  
 nrows = 2,   
 shareX = TRUE, shareY = TRUE),   
 nrows = 2, margin = 0.04  
 ## shareX, shareY를 TRUE로 설정하여 축을 공유  
 ) |>   
 layout(showlegend = TRUE,   
 title = '최근 100일간 코로나19 확진자수',  
 margin = margins) |>  
 config(  
 ## 이미지 저장 버튼의 세부 속성 설정  
 toImageButtonOptions = list(  
 ## 파일이름 설정  
 filename= '이미지',  
 ## 저장 포팻 설정  
 format = "svg",  
 ## 높이, 너비 설정  
 width = 800, height = 600  
 )  
 )

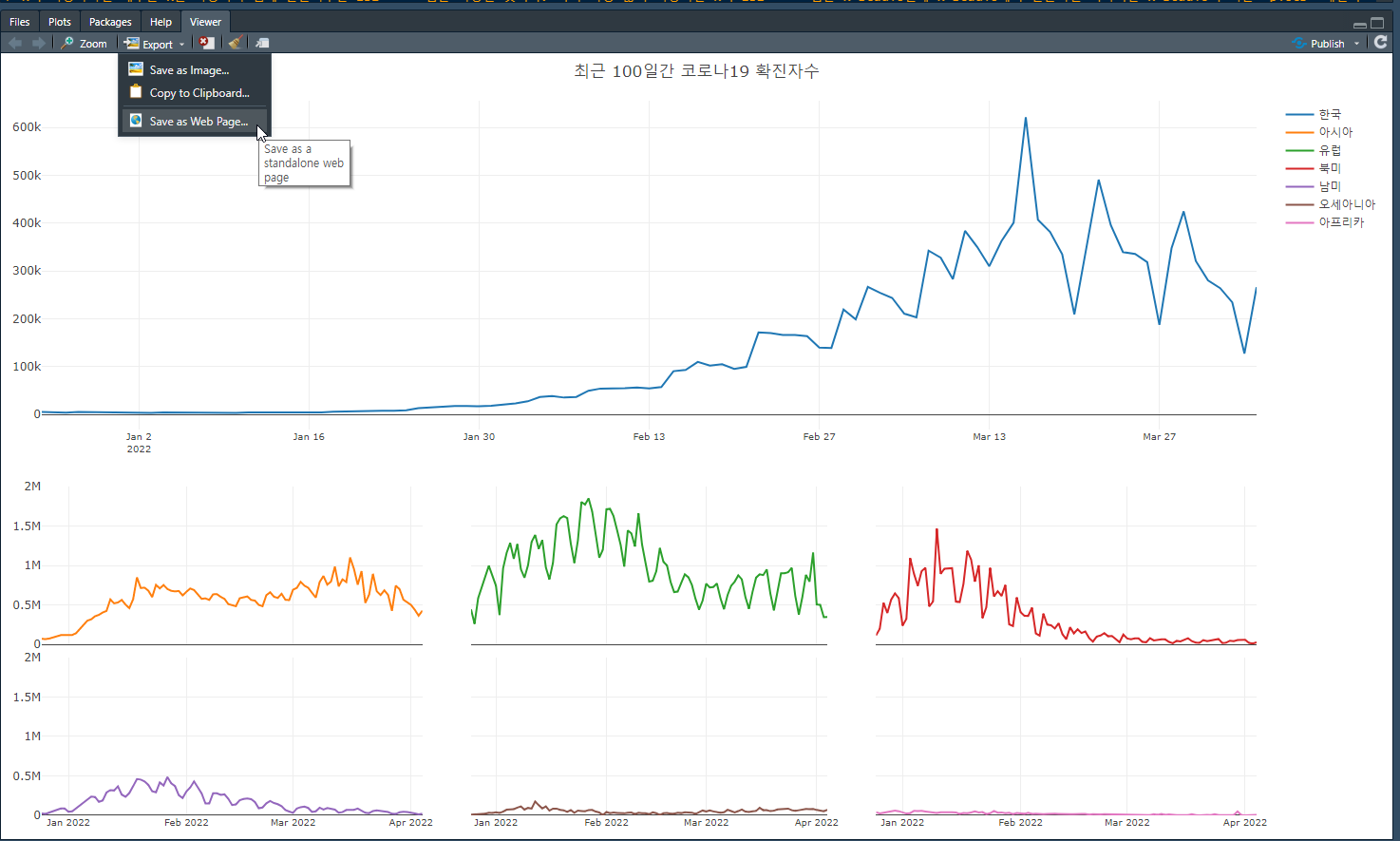
## R-Studio로 사용하기

R의 사용자라면 대부분 R을 사용하기 쉽게 만들어주는 IDE 프로그램을 사용할 것이다. 특히 가장 많이 사용하는 R의 IDE 프로그램은 R-Studio인데 R-Studio에서 만들어진 시각화는 R-Studio의 기본 ‘plots’ 패널이 출력된다. plotly 시각화의 모든 기능은 ‘plots’ 패널에서 사용이 가능하지만 그 크기가 한정적이어서 사용에 원활치는 않다. 따라서 R-Studio에서 효과적으로 사용하기 위해서는 ‘plots’ 패널의 zoom 버튼을 사용하여 ‘View Zoom’ 패널을 띄워 사용하면 plotly 시각화의 기능을 충실히 사용할 수 있다.

## HTML로 사용하기

앞서 두 가지 방법은 시각화를 만든 작성자 위주로 사용할 수 있는 방법이다. 하지만 동적 시각화의 효율적 사용을 위해서라면 여러 사람이 사용해야 할 것이다. 따라서 plotly 시각화를 HTML 파일로 저장하여 전달하면 웹 브라우저에 plotly 시각화를 띄워 사용할 수 있다.

plotly 시각화를 HTML 파일로 저장하는 방법은 두 가지가 있는데 앞서 설명한 R-Stutio ‘plot’ 패널의 export->Save as Web Page…메뉴를 사용하여 저장할 수 있다.



plot 패널의 save as web page

또 R 코드를 사용한다면 htmlwidgets::saveWidget()를 사용하여 다음과 같이 저장할 수 있다.

save\_fig |> htmlwidgets::saveWidget('save\_fig.html', selfcontained = F)

htmlwidgets::saveWidget()을 사용해서 생성된 HTML파일은 생각보다 사이즈가 크다. 보통 4~5MB정도의 크기로 생성되는데 이 파일에는 모든 자바스크립트와 CSS 정보가 담겨있다. 이정도 크기의 HTML파일은 블로그에 올릴 때 크기의 문제가 발생되는 경우가 있다. 따라서 plotly 시각화의 운영을 위한 최소한의 필수적 정보만을 담은 HTML파일을 만들기 위해서는 partial\_buldle()을 사용한다.

save\_fig1 <- save\_fig |>  
 partial\_bundle() |>  
 htmlwidgets::saveWidget('save\_fig1.html')

1. <https://www.w3schools.com/cssref/css_colors.asp> [↑](#footnote-ref-26)
2. <https://github.com/plotly/plotly.js/blob/master/src/plot_api/plot_config.js> [↑](#footnote-ref-43)