1. 데이터 시각화를 위한 plotly

R을 다루는 많은 교육코스나 서적에서 데이터의 시각화는 대부분 R base에서 제공하는 함수를 사용하거나 ggplot2 패키지를 사용하여는 방법을 위주로 설명한다. 이 두가지 방법은 데이터 시각화 결과가 우수한 편이기 때문에 많이 사용되고 있지만 정적(Static) 시각화이다. 정적 시각화는 최근 인포그래픽(Infographic)이라고 불리며 일반적으로 문서나 인쇄물에 많이 사용되고 웹에 게시되는 이미지로 사용된다. 그렇기 때문에 대부분 png, jpg, pdf 등의 벡터 혹은 픽셀 이미지 파일로 제공된다. 정적 데이터 시각화는 데이터 분석가의 의도에 맞춰 작성되기 때문에 데이터 분석가의 분석에 의존적일 수 밖에 없으며 독자의 의도에 따른 해석은 매우 제한될 수 밖에 없다.

이러한 제한점을 극복하기 위해 사용되는 데이터 시각화 방법이 동적(Dynamic) 시각화 혹은 인터랙티브(Interactive) 시각화라고 하는 방법이다. 이 동적 시각화는 시각화를 사용하는 사용자의 의도에 따라 데이터를 다각적 관점에서 살펴볼 수 있다는 점이 동적 시각화와 가장 크게 차이나는 점이다. 사용자의 의도에 따라 데이터가 동적으로 변동되어야 하기 때문에 데이터 시각화에 사용되는 매체는 인쇄물 형태 매체가 불가능하고 웹을 통해 제공한다. 따라서 일반적으로 동적 시각화는 웹 사이트에서 제공하는 대시보드(DashBoard)의 형태로 제공되는 것이 일반적이기 때문에 동적 시각화를 위해서는 동적 시각화 전용 패키지를 사용해 시각화 객체를 만드는 방법이외에 shiny 등의 패키지를 사용하여 대시보드를 만드는 것도 같이 익혀야 한다는 어려움이 따른다. R에서 동적 시각화를 위해 제공되는 패키지는 plotly, rbokeh, highcharter 등이 있다.

따라서 정적 시각화와 동적 시각화의 어느것이 더 효용성이 있는지를 단언할 수 없다. 데이터 시각화가 사용되는 매체, 데이터 시각화를 보는 대상, 데이터 시각화에서 보여주고자 하는 스토리에 따라서 정적 시각화를 사용해야 할 때와 동적 시각화를 사용해야 할 때를 적절히 선택해야 한다.

이번 장에서는 R에서 동적 시각화로 많이 사용되는 plotly 패키지를 사용하여 데이터를 시각화하는 방법을 알아본다.[[1]](#footnote-20)

# plotly란?

plotly는 오픈 소스인 JavaScript로 구현된 plotly.js를 기반으로 R에서 생성한 데이터 시각화 객체를 Javascript로 생성해주는 패키지이다. 따라서 plotly로 생성된 시각화는 결국 HTML 코드로 구현되고 이 코드는 웹브라우저 상에서 작동함으로써 사용자의 반응에 따른 데이터의 표현이 가능하다. plotly를 통해 생성된 데이터 시각화의 HTML은 R에서 JavaScript를 사용할 수 있게하는 htmlwidgets 프레임워크에서 동작하기 할 수 있어 HTML자체로 사용할 수도 있고 R Markdown이나 Shiny App, R-Studio, Jupiter Notebook 등에서 자유롭게 사용이 가능하다.

R에서 plotly를 사용한 인터랙티브 데이터 시각화를 만드는 방법은 두 가지이다. 첫 번째는 R에서 데이터 시각화에 가장 많이 사용하는 ggplot2를 사용하여 생성한 객체를 plotly 객체로 전환하는 방법이고 두 번쨰는 plotly 패키지에서 제공하는 함수들을 사용하여 plotly 객체를 직접 생성하는 방법이다.

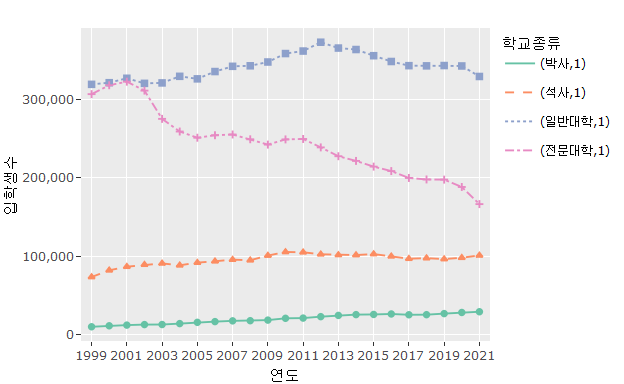
# ggplot 객체의 전환

기존의 R 사용자가 가장 쉽게 plotly를 사용한 인터랙티브 데이터 시각화를 생성하는 방법은 그동안 사용했던 ggplot2 패키지를 사용하여 생성했던 ggplot 객체를 plotly 객체로 전환하는 것이다. 이 방법은 plotly 패키지에서 제공하는 ggplotly()를 사용하면 간단히 전환된다.

ggplotly(p = ggplot2::last\_plot(), width = NULL, height = NULL, tooltip = “all”, dynamicTicks = FALSE, layerData = 1, originalData = TRUE, source = “A”, …)  
- p : plotly로 전환할 ggplot 객체  
- width : plotly 객체의 너비 설정  
- height : plotly 객체의 높이 설정  
- tooltip : plotly 객체에서 마우스의 위치에 따라 표시되는 툴팁의 문자열 설정  
- dynamicTicks : plotly 객체가 Zooming 될 때 눈금자(Tick)을 동적으로 재설정할 것인지를 설정하는 논리값  
- layerData : 레이어의 데이터를 리턴할지를 설정  
- originalData : 원천 데이터(original)를 리턴할지 스케일(scale)된 데이터를 리턴할지를 설정하는 논리값

앞 장에서 생성했던 ggplot 객체를 plotly 객체로 전환하는 코드는 다음과 같다.

if(!require('plotly')) {  
 install.packages('plotly')  
 library(plotly)  
}  
  
ggplotly <-df\_입학자\_long |>   
 ## 지역이 전체, 학교종류가 '전문대학', '일반대학', '석사', '박사'인 데이터 필터링  
 filter(지역 == '전체', 학교종류 %in% c('전문대학', '일반대학', '석사', '박사')) |>  
 ## x축이 연도로 매핑된 ggplot 객체 생성  
 ggplot(aes(x = 연도, y = 입학생수, color = 학교종류 )) +  
 ## y축이 입학생수, group, color, linetype이 학교종류로 매핑된 geom\_line 레이어 생성  
 geom\_line(aes(group = 학교종류, linetype = 학교종류)) +  
 ## y축이 입학생수, group, color이 학교종류로 매핑된 geom\_point 레이어 생성  
 geom\_point(aes(shape = 학교종류), show.legend = FALSE) +  
 ## x축의 눈금을 1999부터 2021까지 2씩 증가한 수치로 설정  
 scale\_x\_discrete(breaks = c(seq(from = 1999, to = 2021, by = 2))) +  
 ## y축에 표현된 라벨을 scales 패키지의 comma함수를 적용(scales 패키지 설치 필요)  
 scale\_y\_continuous(labels = scales::comma) +   
 ## 색 설정을 RColorBrewer 패키지의 'Accent'로 설정  
 scale\_color\_brewer(palette = 'Set2', labels = c('박사', '석사', '일반대학', '전문대학'))  
  
ggplotly(ggplotly)



앞의 plotly 객체에서 보면 ggplot 객체를 정확히 전환하지는 못한다.[[2]](#footnote-22) 그러나 ggplotly()는 ggplot2패키지를 사용해서 생성한 대부분의 ggplot 객체를 전환할 수 있다. 게다가 ggplot2 패키지를 확장해서 사용하게 해주는 ggforce, GGally와 같은 확장 패키지로 생성된 객체도 변환이 가능하다는 장점이 있다. 따라서 plotly 객체를 다루는 방법을 잘 익히면 기존에 생성했던 ggplot 객체의 시각화를 재활용 할 수있게 된다. 그리고 plotly를 사용하다보면 느끼겠지만 ggplot2의 통계 요소와 분할 요소의 몇몇 요소들은 plotly보다는 ggplot2가 훨씬 편리할 떄가 있다. 이러한 경우 ggplot2와 plotly를 적절히 혼용하면 매우 좋은 결과를 얻을 수 있다.

# plotly 구성

plotly 객체는 pltoly.js를 지원하는 스키마로 표현된다. 사실 ggplot2이던 plotly이던 각각의 시각화 객체는 R에서 특별하게 정의된 데이터 구조로 표현된다. 이 데이터 구조가 R의 그래픽 엔진을 통해 이미지로 표현되는 것이다. 따라서 plotly 객체도 우리가 눈으로 보기에는 이미지로 보이지만 R에서는 plotly.js에서 지원하는 데이터 구조로 표현된 데이터 객체인 것이다.

plotly 객체를 생성하기 위한 코드는 plot\_ly()를 사용한 plotly 객체 생성, add\_trace()를 사용한 trace 추가, layout()을 사용한 레이어 설정의 세 부분으로 구성된다.

## plotly 객체 생성 : plot\_ly()

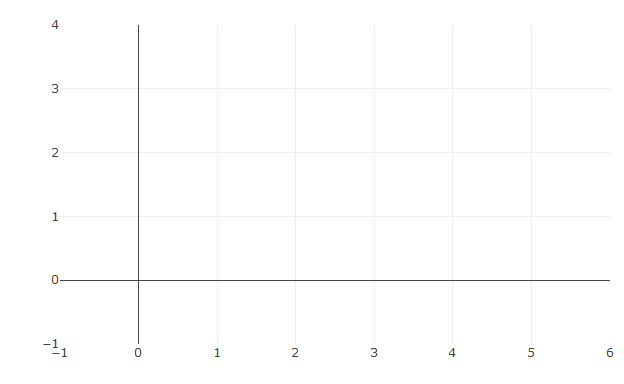
설명한바와 같이 plotly는 보여지기에 그래픽으로 보여지지만 내부적으로는 데이터 구조형태로 표현된다. 따라서 plotly객체를 표현하는 데이터 구조(스키마)를 생성하기 위해서는 제일 먼저 plotly 객체를 생성하는 기본 스키마를 정의하기 위한 초기화 함수가 필요하다. plotly 객체를 시작하기 위한 초기화 함수는 plot\_ly()이다.

plot\_ly()는 plotly.js에서 정의된 plotly 객체 스키마를 생성하는 함수이다. 사용자가 직접 plotly.js형태의 스키마 객체를 타이핑하여 생성하는 것은 어려움이 따르기 때문에 plotly 객체 스키마 생성, 기본 스키마 속성 설정 등을 지원하는 함수이다. 이 방법은 ggplot객체를 생성하기 위해 ggplot()를 사용하여 ggplot 객체를 생성하는 것과 동일한 방법이고 이 방법에서 영감을 받았다고 한다.

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”)  
- p : plotly로 시각화할 데이터프레임  
- … : type에서 설정하는 trace의 종류에 따라 설정할 수 있는 속성 설정  
- type : trace 타입 설정  
- name : plotly 객체의 trace name 속성 설정  
- color : ‘fill-color’ 속성으로 매핑될 색 값(value) 설정  
- colors : ‘fill-color’에 매핑될 colorbrewer2.org의 팔레트 이름이나 16진수의’#RRGGBB’형태로 표현된 색의 벡터(vector) 설정  
- alpha : color에서 설정된 색의 투명도 값(value) 설정  
- stroke : ‘stroke-color’(외곽선 색) 속성으로 매핑될 색 값(value) 설정  
- strokes : ‘stroke-color’(외곽선 색)에 매핑될 colorbrewer2.org의 팔레트 이름이나 16진수의 ‘#RRGGBB’형태로 표현된 색 벡터(vector) 설정/ - alpha-stroke : stroke(외곽선)에 적용될 alpha 값(value) 설정  
- size : ’fill-size’에 매핑될 크기값(value) 설정  
- sizes : size에 매핑될 수치 벡터(vector) 설정  
- span : ’stroke-size’(외곽선 두께)에 매핑될 두께 값(value) 설정  
- spans : ‘stroke-size’(외곽선 두께)에 매핑될 두께 벡터(vector) 설정  
- symbol : 점 표현에 사용되는 도형 번호(pch)나 도형 이름 값(value) 설정  
- symbols : 점 표현에 사용되는 도형 번호(pch)나 도형 이름 벡터(vector) 설정  
- linetype : 라인 타입의 설정에 사용되는 번호나 라인 타입 값(value) 설정  
- linetypes : 라인 타입의 설정에 사용되는 번호나 라인 타입 벡터(vector) 설정  
- split : 다중 traces를 생성시 사용하는 값 설정  
- frame : 애니메이션 프레임 생성시 사용할는 값 설정  
- width : 플롯의 너비(픽셀) 설정  
- height : 플롯의 높이(픽셀) 설정

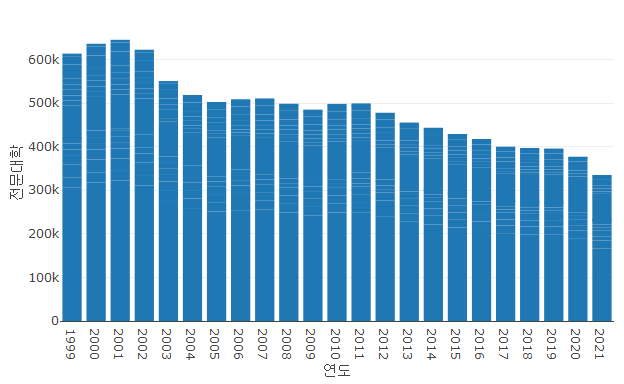
이처럼 plot\_ly()의 함수에서는 많은 매개변수가 사용된다. 특히 ’…’으로 표기된 부분은 type 매개변수에서 설정하는 trace 타입에 따라 설정 내용이 매우 달라진다. ggplot2에서 geom\_\*()를 사용하여 기하 요소 레이어를 하나 하나 설정하면서 전체 시각화를 완성하는 것과 유사하게 plotly에서는 여러개의 trace를 추가함으로써 전체 시각화를 완성해나갈수 있다. 다만 ggplot2의 ggplot()에서는 데이터 요소와 미적요소만을 설정할 수 있었지만 plotly에서는 데이터, trace, trace에 따른 속성까지 설정이 가능하므로 plot\_ly()만가지고도 시각화를 완성할 수 있다. 반면 동시에 여러개의 trace가 포함되는 시각화에서는 뒤에서 설명할 add\_trace()나 add\_\*()를 사용하여 trace를 추가하게 되는데 plot\_ly()에서 설정한 속성들을 상속받게 된다. 이 과정에서 원치않는 속성의 상속을 방지하기 위해 plot\_ly()에 매개변수를 넣지않고 단순히 plotly 객체의 초기화 명령으로 사용도 가능하다.

df\_입학자\_long |> plot\_ly()



plot\_ly()의 매개변수를 설정할 때 꼭 알아 두어야 하는 것은 변수를 매핑하는 방법과 값을 설정하는 방법이 다르다는 것이다. ggplot2에서는 변수를 매핑하기 위해서 aes()를 사용함으로써 변수를 매핑하였고 값을 실정하기 위해서는 aes() 밖에 선언함으로써 설정이 가능하였다. plotly에서는 aes() 대신 ~를 사용하여 변수를 매핑한다.

df\_입학자 |> plot\_ly(x = ~연도, y = ~전문대학)



앞선 plot\_ly()코드에서는 X축과 Y축을 지정하는 매개변수인 x, y에 ~를 사용하여 각각의 변수를 매핑하였고 색을 설정하는 매개변수인 color에는 I()를 사용하여 ’red’를 설정하였다. 사실 plotly 객체에서 가장 중요한 것은 trace의 종류이다. plot\_ly()만을 사용하여 그래프를 완성할 때 trace의 종류를 생략하면 plotly에서 데이터를 파악하여 가장 좋은 trace를 설정해준다.

## trace 설정 : add\_trace()

trace는 plotly 객체에서 데이터를 표현하는 방법을 설정하는 과정이다. ggplot2에서는 geom\_\*()을 사용하여 점, 선, 막대 등의 기하 요소로 표현하였지만 plotly 객체에서는 trace라는 이름으로 각각의 데이터 표현을 추가한다. plotly는 40가지 이상의 trace를 제공하는데 plotly 객체는 각각 고유한 하나 이상의 trace가 포함되어야 한다. 또 각각의 trace에는 해당 trace의 세부 속성을 설정을 위한 type을 가지고 있다.

이 trace와 type을 설정하는 방법은 add\_trace()를 사용하여 trace와 trace에 속한 type 속성을 설정하는 방법과 add\_marker(), add\_lines()등과 같이 add\_trace()에서 파생된 래핑 함수를 사용하는 방법이 있다.

add\_scatter()의 주요 사용법은 다음과 같다.

add\_trace(p, …, data = NULL, inherit = TRUE)  
- p : plot\_ly()로 생성한 plotly 객체  
- … : 스캐터 trace에 설정할 수 있는 속성 설정  
- data : 시각화할 데이터프레임  
- inherit : plot\_ly()에 설정된 속성 type을 상속할지를 결정하는 논리값

먼저 trace의 종류를 설명하도록 하겠다.

### 스캐터(scatter) trace

스캐터 trace는 점, 선, 문자를 X, Y 축 좌표의 위치를 사용해 시각화를 하는 형태의 trace를 모두 말한다. 따라서 스캐터 trace를 통해 생성 가능한 시각화는 산점도, 선 그래프, 텍스트 차트, 풍선 차트 등이다. 이 스캐터 trace는 add\_trace()의 type 매개변수에 ’scatter’를 설정함으로써 사용할 수 있고 add\_\*() 함수(add\_markers(), add\_lines(), add\_paths(), add\_segments(), add\_ribbons())를 사용할 수 있다.

#### add\_markers()

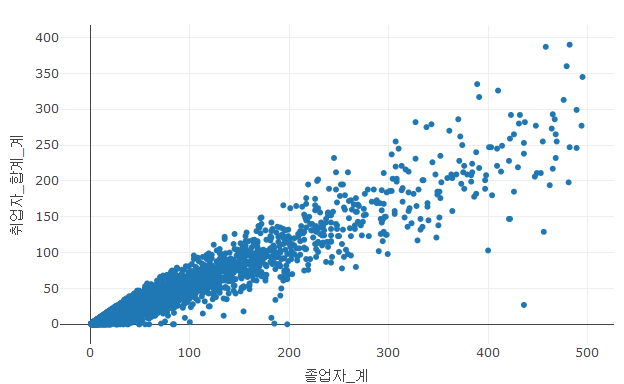
add\_markers()는 X, Y축의 값으로 표현가능한 좌표에 점을 사용하여 데이터를 시각화하는 방법이다. 데이터 시각화의 한 종류인 산점도를 만들때 사용되는 trace로 주요 사용법은 다음과 같다.

add\_trace(p, type = ‘sactter’, mode = ‘markers’, marker = NULL, …, data = NULL, inherit = TRUE)  
add\_markers(p, x = NULL, y = NULL, z = NULL, …, data = NULL, inherit = TRUE)  
- p : plot\_ly()로 생성한 plotly 객체  
- type : trace 타입을 설정, add\_markers()는 ‘scatter’로 설정 - mode : ’scatter’ trace 중 어떤 도형을 사용할지 설정, add\_markers()는 ’markers’를 설정 - marker : marker의 type 설정을 위한 매개변수 list 설정 - … : 스캐터 trace의 markers 모드에 설정할 수 있는 속성 설정  
- data : 시각화할 데이터프레임  
- inherit : plot\_ly()에 설정된 속성 type을 상속할지를 결정하는 논리값  
- x : X축에 매핑할 변수를 ~로 설정 - y : Y축에 매핑할 변수를 ~로 설정 - z : Z축에 매핑할 변수를 ~로 설정

add\_trace()로 스캐터 trace를 사용하려면 먼저 trace가 스캐터 trace라는 것을 type = 'scatter'로 전달하여야 한다. 다음으로 앞서 설명한 바와 같이 스캐터 trace는 점, 선, 문자를 사용하기 때문에 이 중 어떤 도형을 사용할지를 mode 매개변수를 통해 전달한다. 점을 사용한 산점도는 mode = 'markers'를 설정한다.

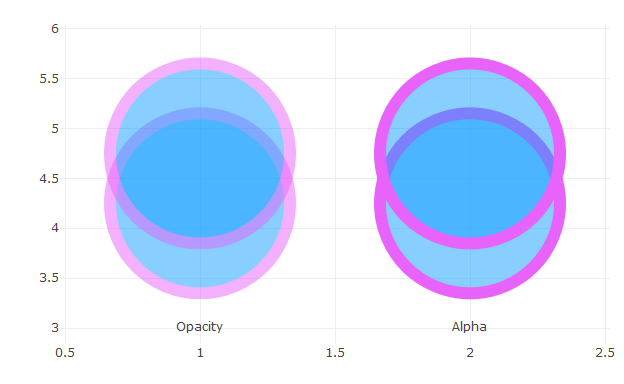
trace의 type과 mode가 설정되면 먼저 X, Y, Z 축으로 매핑할 변수를 설정한다. plotly는 3차원 그래픽을 지원하기 때문에 Z축을 설정할 수 있다. 동적 시각화는 사용자가 시각화를 자신이 원하는 방향으로 설정하여 관찰할 수 있기 때문에 3차원 효과가 효율적일 수 있다. 하지만 일반적으로 데이터 시각화에서는 3차원 을 사용하는 것은 크게 효과적이지 않다고 알려져 있기 때문에 3차원의 활용은 주의할 필요가 있다.

p <- df\_취업통계 |> filter(졸업자\_계 < 500) |> plot\_ly()  
  
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계)



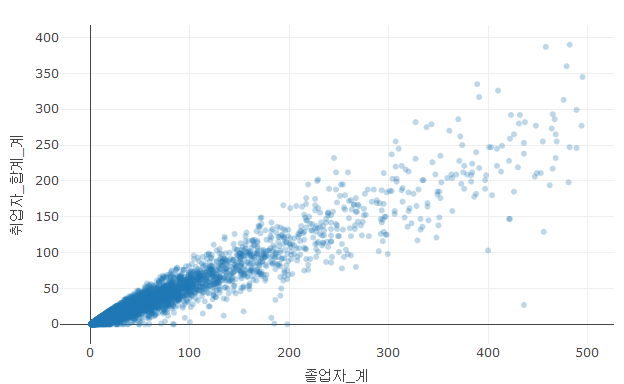
##### alpha, opacity

alpha와 opacity는 markers로 표현되는 점의 투명도를 설정하는 속성이다. alpha는 점의 내부 색상에만 적용되는 투명도이고 opacity는 점과 외곽선에 모두 적용되는 투명도이다.

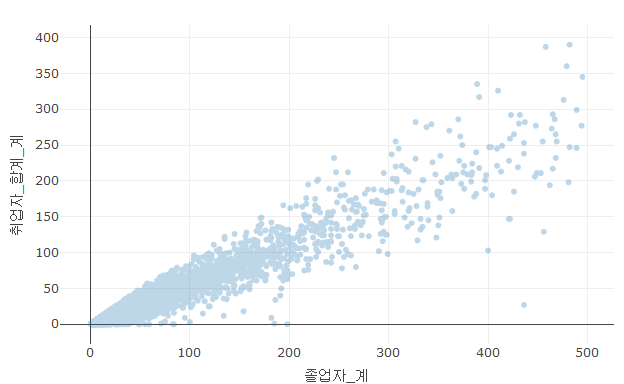


alpha와 opacity는 0부터 1까지의 수치로 설정하며 0은 완전 투명으로 보이지 않고 1은 불투명한 점을 표현한다. 이는 산점도의 표현시 오버플로팅을 방지하고 색으로 구분되는 데이터 그룹간의 표현에 효과적으로 사용된다.

p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, alpha = 0.3)



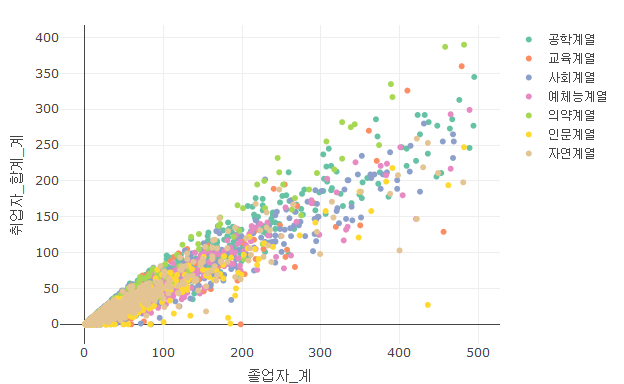
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, opacity = 0.3)



##### color, colors

color는 markers의 내부 색을 설정하는 속성이다. 내부 색을 설정할 때는 먼저 색을 변수에 매핑할지, 특정 색상으로 설정할 지를 결정해야 한다. 이를 설정하는 매개변수가 color이다. color에 변수를 ~를 사용하여 매핑하면 해당 변수의 값에 따라 색이 매핑되어 표현된다.

p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~대계열)

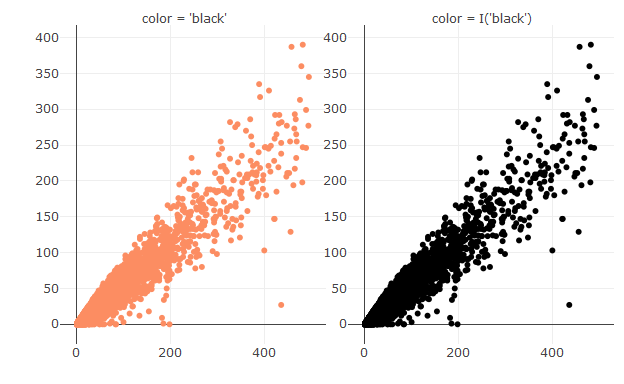


반면 특정한 색으로 설정할 때는 color에 특정 색 이름을 설정하면 모든 marker가 동일한 색으로 나타난다. 이 과정에서 하나 주의해야 할 것이 설정의 방법이다. 일반적인 변수 설정과 같이 색상명을 설정하면 다음과 같이 정확한 색상이 나타나지 않는다.

p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = 'black')

plotly에서 색의 사용은 기본적으로 매핑을 전제로 사용된다. 따라서 앞의 코드에서 color = 'black'으로 설정하는 것은 색을 검정색으로 지정하는 것이 아니고 ‘black’이라는 이름으로 매핑된 색을 불러온다는 의미이다. 하지만 미리 매핑되어 정의된 ’black’ 색 배열이 없기 때문에 plotly의 디폴트 색 팔레트를 사용하여 색이 설정된다. 자신이 원하는 색을 직접 설정하기 위해서는 ’asis’를 의미하는 I()를 사용하여 색을 설정하여야 한다.

p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = I('black'))



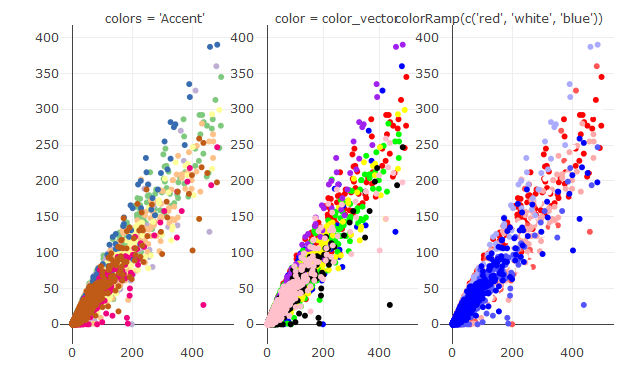
colors는 color에서 매핑된 변수에 따른 색의 스케일을 설정하는 매개변수이다. plotly에서는 색의 스케일을 설정하는 팔레트를 선정할 때 다음의 세 가지 방법을 사용한다.

첫 번째 방법은 RColorBrewer 패키지에서 제공하는 팔레트의 이름을 설정하는 방법이다. RColorBrewer 패키지는 R에서 가장 대중적으로 사용되는 색 팔레트를 제공하는 패키지로 ggplot2에서도 많이 사용된다. 이 패키지에서 제공하는 팔레트의 이름을 colors에 지정함으로써 해당 팔레트를 사용할 수 있다.

두 번째 방법은 사용할 색을 직접 지정하는 방법이다. 사용할 색의 이름을 가지는 문자열 벡터를 사용하여 직접 색을 지정한다.

세 번째는 colorRamp()나 scales::colour\_ramp()와 같은 색 보간 함수를 사용하는 방법이다. colorRamp()는 매개변수로 전달되는 색 벡터의 사이 색을 반환하는 함수를 만들어주는데 0부터 1까지의 값 범위내에 해당하는 색을 반환해준다.

p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~대계열, colors = 'Accent')  
  
color\_vector <- c('red', 'blue', 'green', 'yellow', 'purple', 'black', 'pink')  
  
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~대계열, colors = color\_vector)  
  
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~대계열, colors = colorRamp(c('red', 'blue')))

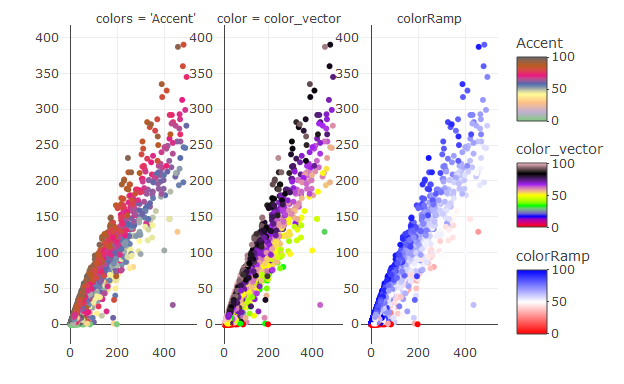


plotly의 색 설정에서 하나 더 알아야하는 것이 colorbar() 설정이다. color 매개변수에 매핑되는 변수가 이산형일 경우는 각각의 카테고리에 따라 색상이 설정되지만 연속형일 경우는 변수의 수치에 따라 색상이 변경된다. 이러한 경우 사용되는 색 스케일이 colorbar()이다.

colorbar(p, …, limits = NULL, which = 1)  
- p : plot\_ly()로 생성한 plotly 객체  
- … : 컬러바 설정을 위한 세부 속성  
- limits : 컬러바 범위 설정을 위한 수치 벡터  
- which : 다중 컬러바가 생성된 경우 컬러바 선택

컬러바 설정에 사용되는 세부 속성은 현재(22.03) 43개가 제공된다. 이에 대한 세부 속성과 설명은 plotly 매뉴얼을 참조하라.[[3]](#footnote-26)

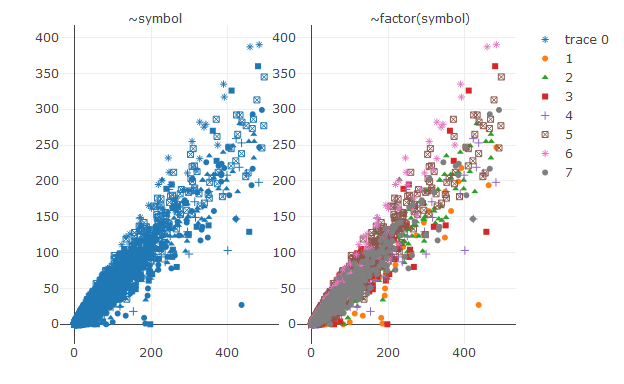
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~취업률\_계, colors = 'Accent')  
  
color\_vector <- c('red', 'blue', 'green', 'yellow', 'purple', 'black', 'pink')  
  
p) |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~취업률\_계, colors = color\_vector)  
  
p |>  
 add\_markers(x = ~졸업자\_계, y = ~취업자\_합계\_계, color = ~취업률\_계, colors = colorRamp(c('red', 'blue')))



##### symbol, symbols

symbol은 점의 형태를 설정하는 속성이다. 앞서 color와 같이 symbol도 매핑할 변수를 설정하는 매개변수이고 symbols는 변수 카테고리에 따라 설정하는 점의 형태를 설정하는 매개변수이다. color와 다른 점은 연속형 변수에 매핑되는 symbol은 단 하나의 카테고리로 분류된다는 점이다. 그렇기 때문에 좌표에 표시되는 점의 형태는 다르게 보이지만 범례에 표현되지는 않는다는 점이다. 다음의 예를 살펴보자.

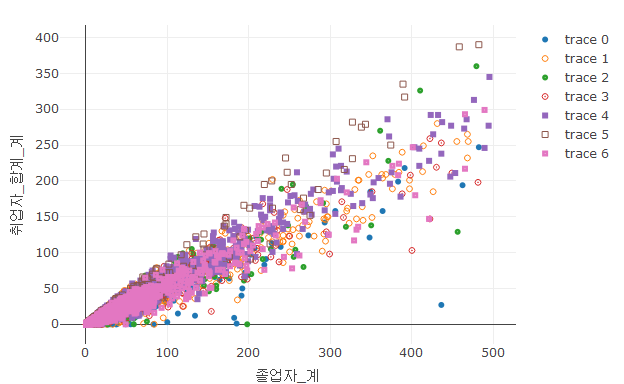
p1 <- df\_취업통계 |> filter(졸업자\_계 < 500) |> mutate(symbol = case\_when(  
 대계열 == '인문계열' ~ 1,   
 대계열 == '사회계열' ~ 2,  
 대계열 == '교육계열' ~ 3,  
 대계열 == '자연계열' ~ 4,  
 대계열 == '공학계열' ~ 5,  
 대계열 == '의약계열' ~ 6,  
 대계열 == '예체능계열' ~ 7  
)) |> plot\_ly()  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, symbol = ~symbol)  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, symbol = ~factor(symbol))



앞의 예에서 ‘symbol’ 열은 대계열을 수치로 표현한 열로 수치형 열이다. 수치형 열이기 때문에 연속형 변수로 취급되고 이 열을 변수로 바로 사용하면 왼쪽과 같이 점의 모형은 달리지지만 범례에는 ‘trace 0’로만 표현된다. 반면 오른쪽과 같이 ’symbol’ 열을 factor()를 사용하여 팩터로 전환하여 이산형 변수로 취급하면 각각의 카테고리를 구분하여 색까지 구분해주며 범례에서 각각의 카테고리를 표현한다.

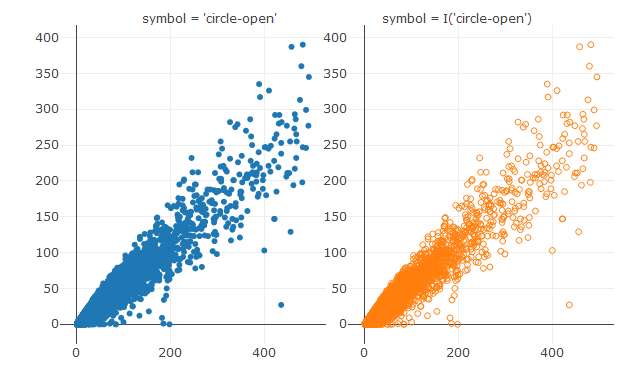
plotly에서는 0번부터 52번까지 총 53개의 symbol을 제공한다. 이 기본 53개의 도형 번호에 100을 더하면 내부가 빈 ’open’형 심볼, 200을 더하면 점이 찍힌 ’dot’형 심볼, 300을 더하면 내부가 비고 점이 찍힌 ’open-dot’형 심볼을 의미한다.

add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, symbol = ~factor(symbol), symbols = c('circle', 'circle-open', 'circle-dot', "circle-open-dot", "square", "square-open", "square-dot"), names = c('인문계열', '사회계열', '교육계열', '자연계열', '공학계열', '의약계열', '예체능계열'))



symbol에 변수를 매핑하지 않고 특정 심볼을 설정할 경우는 색의 설정과 같이 I()를 사용한다.

add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, symbol = 'circle-open')  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, symbol = I('circle-open'))



##### stroke, strokes

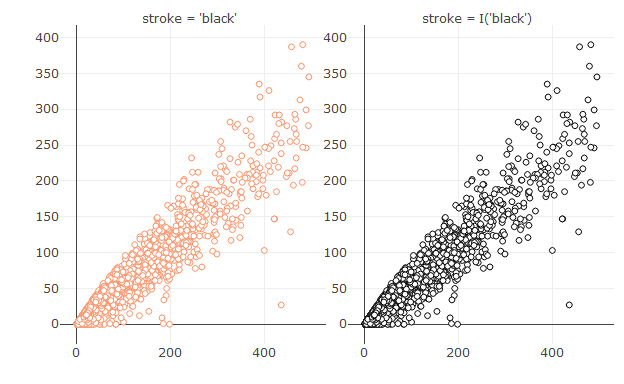
stroke는 점의 외곽선 색을 설정하는 속성이다. color의 설정은 점의 내부색과 외곽선 색을 동시에 설정하지만 외곽선 색을 따로 매핑해야할 경우에 stroke를 사용하여 매핑하거나 설정할 수 있다. stroke와 strokes는 color나 symbol과 같이 변수 매핑으로 사용되는 매개변수는 stroke이고 매핑된 변수의 카테고리에 해당하는 외곽선 색을 설정하는 매개변수는 strokes이다. strokes에 색을 설정하는 방법은 colors에서 사용한 세 가지 방법을 같이 사용할 수 있다 .

add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, stroke = ~대계열, strokes = 'Accent', color = I('white'))



stroke를 특정 값으로 설정하기 위해서는 I()를 사용한다.

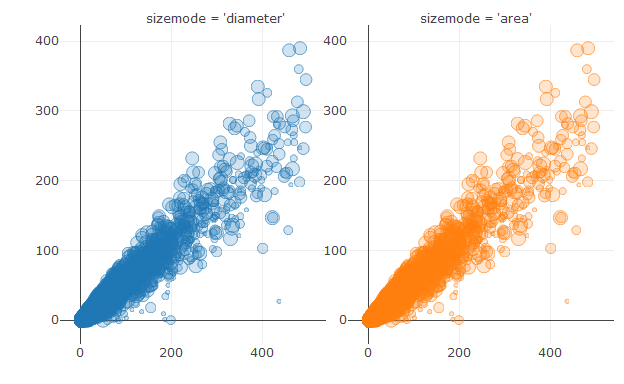
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, stroke = 'black', color = I('white'))  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, stroke = I('black'), color = I('white'))



##### size, sizes

size는 점의 크기를 설정하는 속성이다. size는 변수 매핑을 위한 매개변수이고 sizes는 변수에 매핑된 카테고리별로 크기를 설정하는 매개변수이다. 앞선 속성들과 달리 size는 sizemode 매개변수를 추가적으로 사용할 수 있다. sizemode는 점의 크기를 결정하는 기준을 설정하는 매개변수로 점의 크기를 지름으로 설정하는 ’diameter’와 면적으로 설정하는 ’area’의 두 가지가 있다.

add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, size = ~symbol, sizemode = 'diameter', alpha = 0.3)  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, size = ~symbol, sizemode = 'area', alpha = 0.3)

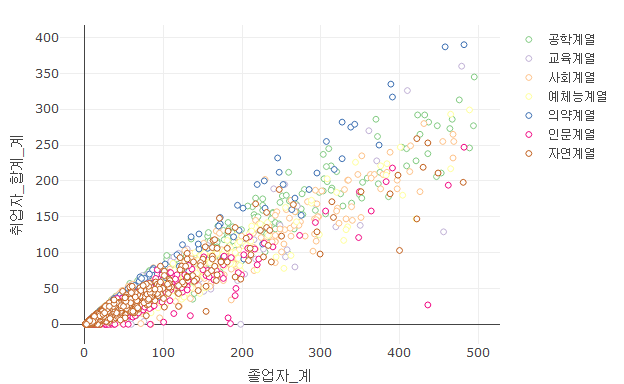


##### name

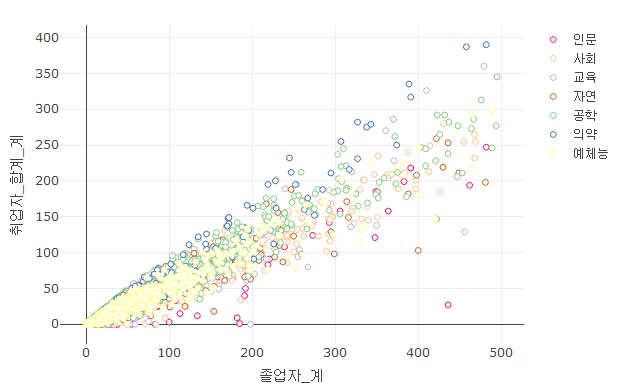
name은 plotly에서 추가되는 각각의 trace에 대한 이름을 설정한다. 이 이름은 범례 아이템과 마우스 포인터가 데이터 점에 위치할때 나타나는 호버(Hover)에 표기되는 이름이 된다.

앞서 본 예에서 color, symbol에 매핑된 변수의 카테고리는 자동적으로 범례 아이템으로 표기되었다. 하지만 stroke에 설정된 변수는 범례에 표현되지 않았다. 이러한 경우 name 매개변수에 범례에 사용할 변수를 매핑함으로써 범례를 설정할 수 있다.

add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, stroke = ~대계열, strokes = 'Accent', color = I('white'), name = ~대계열)



legend\_items <- df\_취업통계 |>   
 filter(졸업자\_계 < 500) |>   
 mutate(symbol = case\_when(  
 대계열 == '인문계열' ~ '인문',   
 대계열 == '사회계열' ~ '사회',  
 대계열 == '교육계열' ~ '교육',  
 대계열 == '자연계열' ~ '자연',  
 대계열 == '공학계열' ~ '공학',  
 대계열 == '의약계열' ~ '의약',  
 대계열 == '예체능계열' ~ '예체능')) |>  
 select(symbol) |>  
 pull()  
  
legend\_items <- fct\_relevel(legend\_items, '인문', '사회', '교육', '자연', '공학', '의약', '예체능')  
  
add\_markers(p1, x = ~졸업자\_계, y = ~취업자\_합계\_계, stroke = ~대계열, strokes = 'Accent', color = I('white'), name = legend\_items)



#### add\_lines()

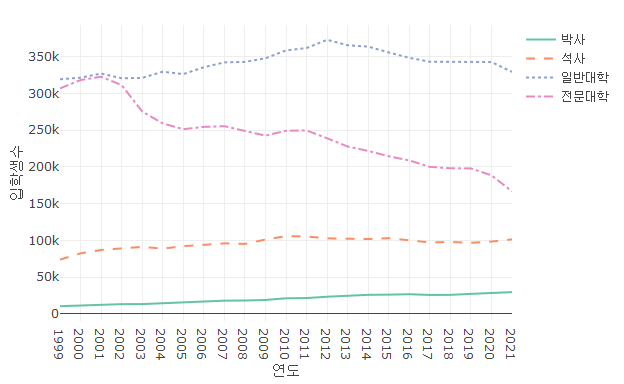
add\_lines()는 X, Y축의 값으로 표현가능한 좌표에 점들을 이어주는 선을 사용하여 데이터를 시각화하는 방법이다. 데이터 시각화의 한 종류인 선 그래프를 만들때 사용되는 trace로 주요 사용법은 다음과 같다.

add\_trace(p, type = ‘sactter’, mode = ‘lines’, line = NULL, …, data = NULL, inherit = TRUE)  
add\_markers(p, x = NULL, y = NULL, z = NULL, …, data = NULL, inherit = TRUE)  
- p : plot\_ly()로 생성한 plotly 객체  
- type : trace 타입을 설정, add\_markers()는 ‘scatter’로 설정 - mode : ’scatter’ trace 중 어떤 도형을 사용할지 설정, add\_markers()는 ’markers’를 설정  
- line : line의 type 설정을 위한 매개변수 list 설정  
- … : 스캐터 trace의 line 모드에 설정할 수 있는 속성 설정  
- data : 시각화할 데이터프레임  
- inherit : plot\_ly()에 설정된 속성 type을 상속할지를 결정하는 논리값  
- x : X축에 매핑할 변수를 ~로 설정  
- y : Y축에 매핑할 변수를 ~로 설정  
- z : Z축에 매핑할 변수를 ~로 설정

add\_trace()로 선 trace를 사용하려면 먼저 trace가 스캐터 trace라는 것을 type = 'scatter'로 전달하고 mode 매개변수에 mode = 'lines'를 설정한다.

trace의 type과 mode가 설정되면 먼저 X, Y, Z 축으로 매핑할 변수를 설정한다.선 그래프는 앞서 그려본 산점도와는 다른 것이 점들이 연결되는 데이터의 그룹 정보가 필요하다. plotly에서의 add\_lines()는 일반적으로 color나 linetype으로 매핑된 변수를 기준으로 그룹화하여 선을 연결한다.

p2 <- df\_입학자\_long |> filter(지역 == '전체', 학교종류 %in% c('전문대학', '일반대학', '석사', '박사')) |> plot\_ly()  
  
add\_lines(p2, x = ~연도, y = ~입학생수, color = ~ 학교종류, linetype = ~학교종류)



##### linetype, linetypes

linetype은

1. 데이터 시각화를 실습하기 위한 데이터는 앞 장의 ggplot2에서 사용하던 데이터를 사용한다. [↑](#footnote-ref-20)
2. 앞의 예에서 범례에 나타난 오류는 ggplotly()에서 다중 범례(Multiple Legend)의 변환 과정에서 발생하는 문제로 ggplot의 범례를 제거하고 plotly의 범례를 생성함으로써 해결가능하다. [↑](#footnote-ref-22)
3. <https://plotly.com/r/reference/#scatter-marker-colorbar> [↑](#footnote-ref-26)