부록 1. plotly.express

## plolty.express 란

python에서 plolty를 만드는 데에는 plotly 라이브러리의 graph\_objects 모듈을 사용하는 방법과 express 모듈을 사용하는 두 가지 방법으로 만들 수 있다. plotly제작사에서도 plotly의 생성 원리를 파악하고 상세한 기능을 사용하기 위해서는 graph\_objects를 사용하여 만드는 것을 권고하고 있다. 하지만 graph\_objects 모듈을 사용하여 plotly를 만들다 보면 코드도 매우 길어지지만 어디 한군데 괄호가 빠지는 경우나 잘못 표시되는 경우 매우 혼란스러워진다. 그래서 plotly 제작사에서는 plotly를 보다 쉽게 만들 수 있는 모듈인 express 모듈을 제공한다.

plolty.graph\_objects는 몇 개의 필수 함수(Figure(), add\_trace(), update\_layout())와 속성값들의 딕셔너리로 사용하지만 plotly.express는 각각의 트레이스에 대한 함수들로만 사용이 가능하다. 따라서 plotly.express는 만들고자 하는 트레이스의 함수와 해당 함수에서 제공하는 매개변수의 리스트를 잘 알아두는 것이 핵심이다.

plotly.express 함수는 시각화를 위한 함수와 데이터, 색상, 추세선을 위한 서브모듈 함수로 구성되어 있다.

## plotly.express의 장단점

plolty 홈페이지에는 plolty.express는 plotly.graph\_objects에 비해 다음과 같은 장점이 있다고 나와 있다.

1. 하나의 함수에서 data와 layout을 모두 설정

pltoly.graph\_objects에서 전체 plotly 시각화를 완성하기 위해 add\_trace()와 update\_layout()의 두 개의 함수를 사용해야 했지만 plotly.express에서는 하나의 함수에서 data 속성과 layout 속성을 모두 설정할 수 있다.

1. 활용성 높은 기본값 설정

plotly.express의 함수에서 설정된 기본값들은 사용자에게 적절한 기본값들이 설정되어 있다.

1. 다양한 입력 형태 가능

plotly.express 함수는 리스트나 딕셔너리, 긴 형식이나 넓은 형식의 Pandas DataFrames, numpy 배열, GeoPandas, GeoDataFrames에 이르기까지 다양한 형식의 입력을 사용할 수 있다.

1. 트레이스와 레이아웃의 자동 연결

plotly.express 함수는 트레이스가 만들어질 때 그 색상, 라인 타입 등에 적절한 범례나 색상 등을 자동으로 설정해 준다.

1. 자동 라벨링과 호버

plotly.express 함수는 입력 데이터인 DataFrame나 xarray를 기반으로 축, 범례, 컬러바 등의 라벨과 호버를 자동으로 설정하고 추가로 설정이 필요한 라벨이나 호버들을 설정을 할 수 있는 기능을 제공한다.

1. 스타일링

plotly.express 함수는 기본으로 제공되는 템플릿에 정의된 스타일을 통해 기본 스타일을 결정하고, category\_orders 및 color\_discrete\_map와 같은 이산형 변수를 사용하여 시각화를 꾸미는데 필요한 컨트롤을 지원한다.

1. 일관된 색상 처리

plotly.express 함수는 입력값에 따라 자동으로 연속형 또는 범주형 색상 팔레트를 설정한다.

1. 패싯 기능

plotly.express 함수는 row, facet\_col and facet\_col\_wrap 매개변수를 사용하여 열 방향, 행 방향, 행렬 방향의 패싯(서브플롯)을 생성한다.

1. pandas backend

plotly.express 함수는 matplotlib의 plot()과 같이 pandas의 backend 함수로써 기능할 수 있다.

1. 추세선 기능

plotly.express 함수는 내장된 기능을 사용하여 다양한 추세선을 자동적으로 그려준다.

그러나 필자가 보기에는 plotly.express는 다음의 특징이 가장 큰 특징인 듯 하다.

첫 번째는 plotly 시각화에서 사용할 데이터를 미리 바인딩해서 열 이름을 사용하기 쉽다는 점이 가장 큰 특징이다. 이는 R에서의 plotly를 사용하는 것과 매우 유사한데 각각의 plolty.express의 함수에서 사용하는 데이터프레임이나 pasdas series를 미리 설정해주고 이 데이터프레임의 열을 해당 plotly 객체에서 사용할 때는 단순히 열 이름만을 설정함으로써 사용이 가능하다는 점이다.

두 번째는 plotly 시각화에서 데이터의 시각적 구분이 필요한 색상, 심볼, 라인타입 등의 시각적 요소들의 매핑이 매우 직관적이고 간편하게 설정할 수 있다는 것이다. 특히 plotly.graph\_objects에서 데이터의 그룹에 따른 색상의 설정에 사용하는 ‘color’ 속성은 배열형태의 데이터 매핑이 불가능해서 for 루핑을 사용해야만 했지만 plolty.express에서는 배열 형태의 데이터 매핑이 가능해서 데이터를 그룹화 하는 열만 매핑해주면 해당 그룹별로 색상의 설정이 가능하다.

세 번째는 서브 플롯을 바로 설정할 수 있다는 것이다. 이것은 plolty.express에서는 패싯이라는 이름으로 설정하는데 패싯으로 서브플롯화 하기 위한 변수를 설정해주면 서브플롯들을 하나하나 만들지 않아도 자동으로 서브플롯들이 만들어진다는 것이다.

네 번째는 plotly.express는 초기화 과정이 포함되어 있다는 것이다. plolty.graph\_objects는 반드시 처음에 Figure()를 사용하여 초기화해서 plolty 객체를 생성하고 여기에 트레이스들을 추가하는 형태로 사용하지만 plotly.express는 초기화 과정을 포함하기 때문에 Figure()를 사용하지 않고 바로 사용한다.

다섯 번째는 data 속성과 layout 속성을 하나의 함수에서 설정이 가능하다는 것이다. plolty.graph\_objects는 add\_trace()와 update\_layout()의 두 개의 함수를 사용해야 전체 시각화를 완성할 수 있지만 plotly.express의 함수는 그 함수의 매개변수에 data 속성에 해당하는 매개변수와 layout 속성에 해당하는 매개변수를 모두 포함하고 있다. 그러나 그 반면에 몇 가지 단점이 존재한다.

첫 번째 단점은 모든 트레이스를 제공하지 않는다는 것이다. plotly.graph\_objects에서는 40여 개의 트레이스를 제공하지만 plotly.express는 이 모든 트레이스를 다 지원하지는 않는다.

현재(version: 5.11.0) plotly.express에서 생성할 수 있는 트레이스는 다음과 같다.

* Basics: scatter, line, area, bar, funnel, timeline
* Part-of-Whole: pie, sunburst, treemap, icicle, funnel\_area
* 1D Distributions: histogram, box, violin, strip, ecdf
* 2D Distributions: density\_heatmap, density\_contour
* Matrix or Image Input: imshow
* 3-Dimensional: scatter\_3d, line\_3d
* Multidimensional: scatter\_matrix, parallel\_coordinates, parallel\_categories
* Tile Maps: scatter\_mapbox, line\_mapbox, choropleth\_mapbox, density\_mapbox
* Outline Maps: scatter\_geo, line\_geo, choropleth
* Polar Charts: scatter\_polar, line\_polar, bar\_polar
* Ternary Charts: scatter\_ternary, line\_ternary

두 번째 단점은 data와 layout에서 제공하는 모든 속성들을 설정할 수 없다는 것이다. 이것이 필자가 느끼기에 가장 큰 단점인데 plotly.express에서 제공하는 ‘layout’ 속성은 약 10개 정도에 불과하여 시각화를 정교하게 꾸미는 데는 매우 제한점이 있다. 그래서 어쩔 수 없이 update\_layout()과 병행하여 사용할 수밖에 없다.

세 번째 단점은 서브 플롯의 구성에 문제가 있다는 것이다. 물론 패싯 기능을 사용하면 서브 플롯을 만드는 것은 plotly.graph\_objects보다는 쉽지만 패싯은 가로와 세로로 동일한 크기의 행렬형 구조로만 만들어지기 때문에 서브 플롯에서 사용했던 다양한 구성 기능이라든지 여러 트레이스를 병합해서 사용하는 기능 등은 구현할 수 없다.

## plotly.express 설치와 로딩

express 모듈은 plotly 라이브러리에 포함되었기 때문에 따로 설치할 필요는 없고 python에서 ’px’라는 별칭으로 임포트 시키는 것이 일반적이다.

plotly.express는 다음과 같이 로딩할 수 있다.

import plotly.express as px

## 함수 설명, 주요 매개변수와 사용 예[[1]](#footnote-1)

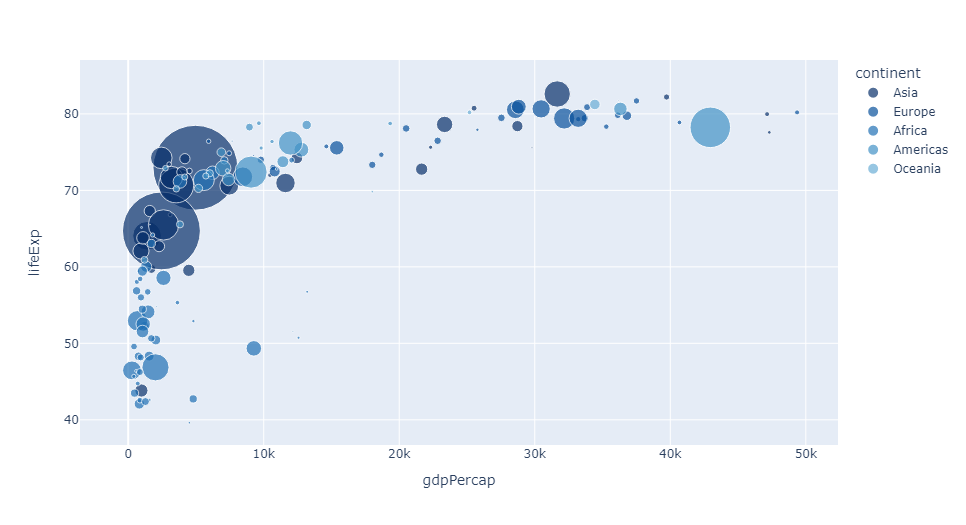
### plotly.express.scatter : 2차원 scatter 트레이스 생성

plotly.express.scatter(data\_frame=None, x=None, y=None, color=None, symbol=None, size=None, hover\_name=None, hover\_data=None, custom\_data=None, text=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, error\_x=None, error\_x\_minus=None, error\_y=None, error\_y\_minus=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, orientation=None, color\_discrete\_sequence=None, color\_discrete\_map=None, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, symbol\_sequence=None, symbol\_map=None, opacity=None, size\_max=None, marginal\_x=None, marginal\_y=None, trendline=None, trendline\_options=None, trendline\_color\_override=None, trendline\_scope=‘trace’, log\_x=False, log\_y=False, range\_x=None, range\_y=None, render\_mode=‘auto’, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| data\_frame | DataFrame, array 유형 객체, dict | 트레이스 생성에 사용되는 dataframe, pandas Series 또는 array와 유사한 오브젝트 설정. 여기에 설정된 데이터 열을 사용할때는 열 이름만으로 사용 가능 |
| x, y | str, int, Series, array 유형 객체 | X, Y축에 매핑에 사용될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| color, symbol, size | str, int, Series, array 유형 객체 | 색상, 심볼, 크기 매핑에 사용될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| hover\_name | str, int, Series, array 유형 객체 | 호버에 볼드체로 표현되는 호버 이름으로 매핑될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| hover\_data | str list, int Series, array 유형 객체, or dict | 호버를 표시할지 여부, 호버에 표시되는 d3 format 문자열 등 호버에 표현할 데이터와 관련된 데이터 |
| text | str or int or Series or array 유형 객체 | 시각화의 text로 매핑될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| facet\_row, facet\_col | str, int, Series, array 유형 객체 | 행 방향, 열 방향의 패싯 개수로 사용될 정수나 문자열 |
| facet\_col\_wrap | int | 열 방향 패싯의 최대 갯수로 사용될 정수 |
| facet\_row\_spacing, facet\_col\_spacing | 0에서 1사이의 float | 패싯의 행 방향, 열 방향의 간격으로 사용될 0부터 1사이의 수치 |
| category\_orders | str 키와 str 값 list로 구성된 dict | 이산형 변수의 순서를 설정하는 딕셔너리나 리스트 |
| labels | str 키와 str 값으로 구성된 dict | 축 제목, 범례 항목, 호버에 사용되는 라벨 설정 |
| orientation | h'나 'v' | 시각화의 방향을 설정하는 'v' 또는 'h' |
| color\_discrete\_sequence | str list | 이산형 색상의 리스트 |
| color\_discrete\_map | str 키와 str 값으로 구성된 dict | 이산형 변수와 색상이 매핑되는 변수값과 색상의 딕셔너리 |
| color\_continuous\_scale | str list | 연속형 색상 스케일 |
| symbol\_sequence | str list | 심볼값에 매핑될 샘볼 리스트 |
| opacity | float | 투명도를 설정하는 0과 1사이의 수치 |
| trendline | 'ols', 'lowess', 'rolling', 'expanding', 'ewm' | 추세선을 그리는 방식의 문자열 |
| range\_x, range\_y | 두 수치를 가지는 list | X, Y축의 범위 설정 |
| template | str, dict, plotly.graph\_objects.layout.Template 인스턴스 | 적용할 template 이름 |
| title | str | 시각화 제목으로 설정할 문자열 |
| width, height | int | 시각화의 전체 너비, 높이를 설정할 정수 |

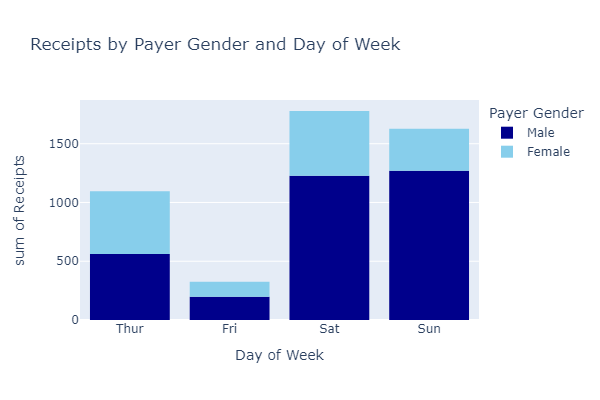
#### 기본 scatter plot

import plotly.express as px  
df = px.data.gapminder()  
  
fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",  
 size="pop", color="continent", hover\_name="country",   
 size\_max=60)  
fig.show()



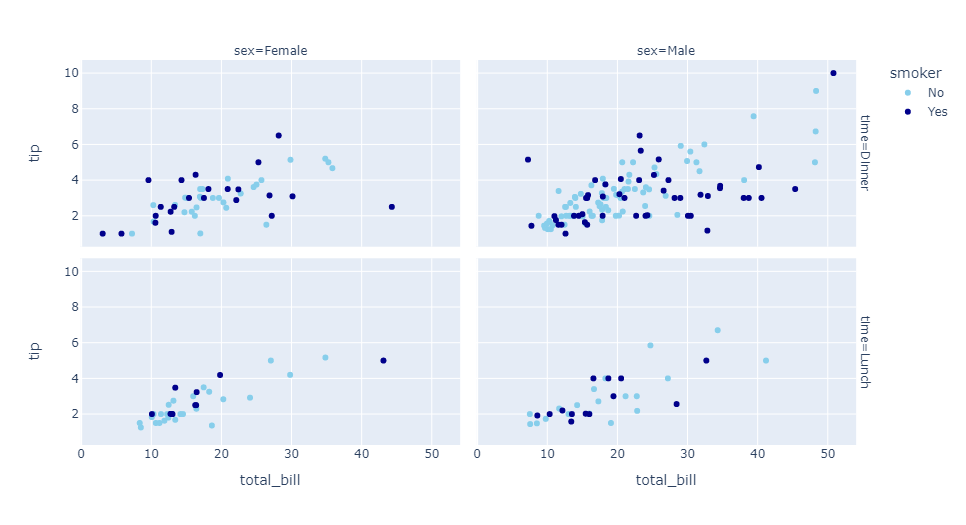
#### order, map 설정

import plotly.express as px  
df = px.data.tips()  
fig = px.histogram(df, x="day", y="total\_bill", color="sex",  
 title="Receipts by Payer Gender and Day of Week",  
 width=600, height=400,  
 labels={ # replaces default labels by column name  
 "sex": "Payer Gender", "day": "Day of Week", "total\_bill": "Receipts"  
 },  
 category\_orders={ # replaces default order by column name  
 "day": ["Thur", "Fri", "Sat", "Sun"], "sex": ["Male", "Female"]  
 },  
 color\_discrete\_map={ # replaces default color mapping by value  
 "Male": "darkblue", "Female": "skyblue"  
 }  
 )  
fig.show()



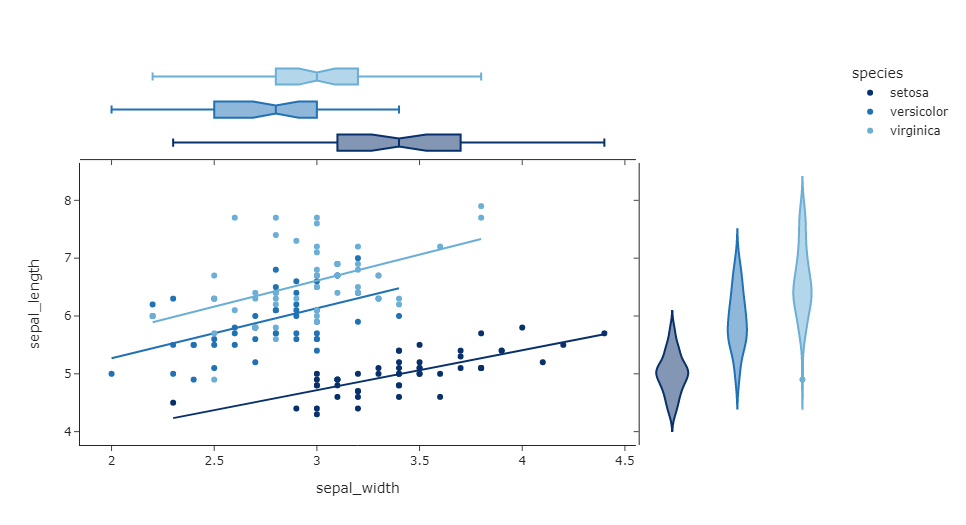
#### facet

import plotly.express as px  
df = px.data.tips()  
fig = px.scatter(df, x="total\_bill", y="tip", color="smoker", facet\_col="sex", facet\_row="time")  
fig.show()



#### trendline, marginal 설정

import plotly.express as px  
df = px.data.iris()  
fig = px.scatter(df, x="sepal\_width", y="sepal\_length", color="species", marginal\_y="violin",  
 marginal\_x="box", trendline="ols", template="simple\_white")  
fig.show()



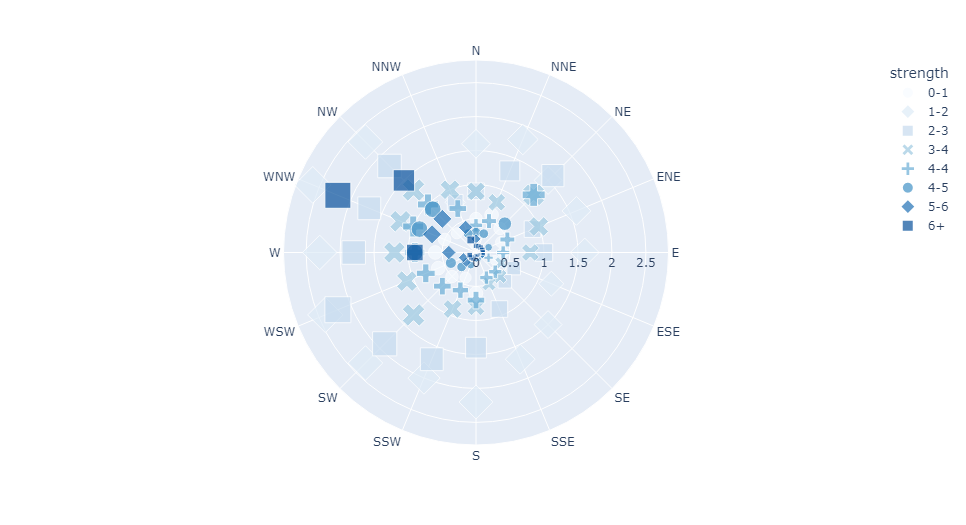
### plotly.express.scatter\_polar

plotly.express.scatter\_polar(data\_frame=None, r=None, theta=None, color=None, symbol=None, size=None, hover\_name=None, hover\_data=None, custom\_data=None, text=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, symbol\_sequence=None, symbol\_map=None, opacity=None, direction=‘clockwise’, start\_angle=90, size\_max=None, range\_r=None, range\_theta=None, log\_r=False, render\_mode=‘auto’, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| r | str, int, Series, array 유형 객체 | 표시되는 데이터의 반지름으로 사용될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| theta | str, int, Series, array 유형 객체 | 표시되는 데이터의 각도로 사용될 배열이나 data\_frame으로 설정된 dataframe, pandas Series의 열 이름 |
| direction | ‘counterclockwise'나 'clockwise' | 각도 축의 표시되는 방향 설정 |
| start\_angle | int | 각도 축이 시작되는 각도 설정 |
| range\_r | 두 수치를 가지는 list | r의 범위 설정 |
| range\_theta | 두 수치를 가지는 list | theta의 범위 설정 |

#### 기본 scatter\_polar plot

import plotly.express as px  
df = px.data.wind()  
fig = px.scatter\_polar(df, r="frequency", theta="direction",  
 color="strength", symbol="strength", size="frequency",  
 color\_discrete\_sequence=px.colors.sequential.Viridis)  
fig.show()



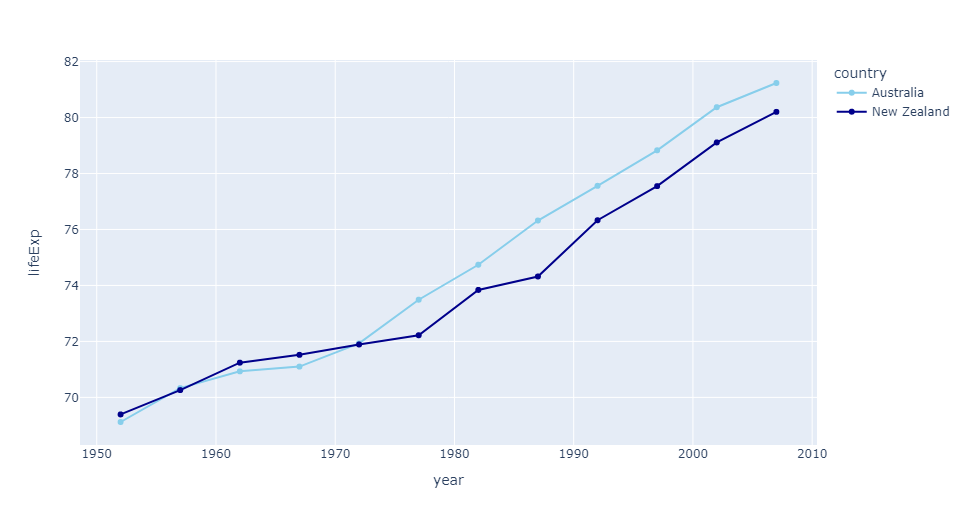
### plotly.express.line

plotly.express.line(data\_frame=None, x=None, y=None, line\_group=None, color=None, line\_dash=None, symbol=None, hover\_name=None, hover\_data=None, custom\_data=None, text=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, error\_x=None, error\_x\_minus=None, error\_y=None, error\_y\_minus=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, orientation=None, color\_discrete\_sequence=None, color\_discrete\_map=None, line\_dash\_sequence=None, line\_dash\_map=None, symbol\_sequence=None, symbol\_map=None, markers=False, log\_x=False, log\_y=False, range\_x=None, range\_y=None, line\_shape=None, render\_mode=‘auto’, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **line\_group** | str, int, Series, array 유형 객체 | 같은 라인으로 그려질 데이터 그룹화 변수 |
| **line\_dash** | str, int, Series, array 유형 객체 | 라인의 대시 패턴 설정 |
| **line\_dash\_sequence** | str list | 라인의 대시 순차 리스트 |
| **line\_dash\_map** | str 키와 str 값으로 구성된 dict | 라인의 대시 매핑 리스트 |
| **markers** | boolean | 라인에 마커를 표시할지 여부 설정 |
| **line\_shape** | linear' 이나 'spline' | 꺾이는 부분을 직선과 곡선 여부 설정 |

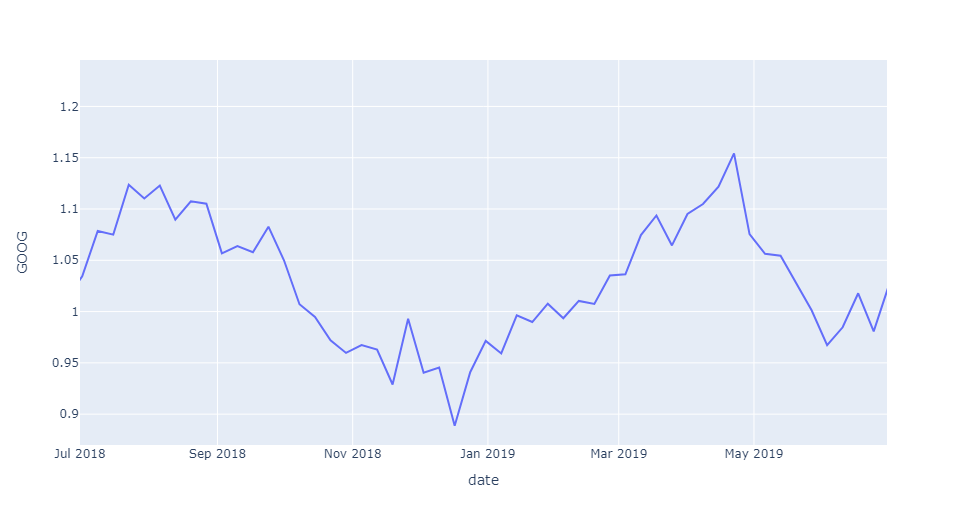
#### 기본 line plot

import plotly.express as px  
df = px.data.gapminder().query("continent == 'Oceania'")  
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True)  
fig.show()



#### 범위 설정

import plotly.express as px  
  
df = px.data.stocks()  
fig = px.line(df, x='date', y="GOOG", range\_x=['2018-07-01','2019-06-30'])  
fig.show()



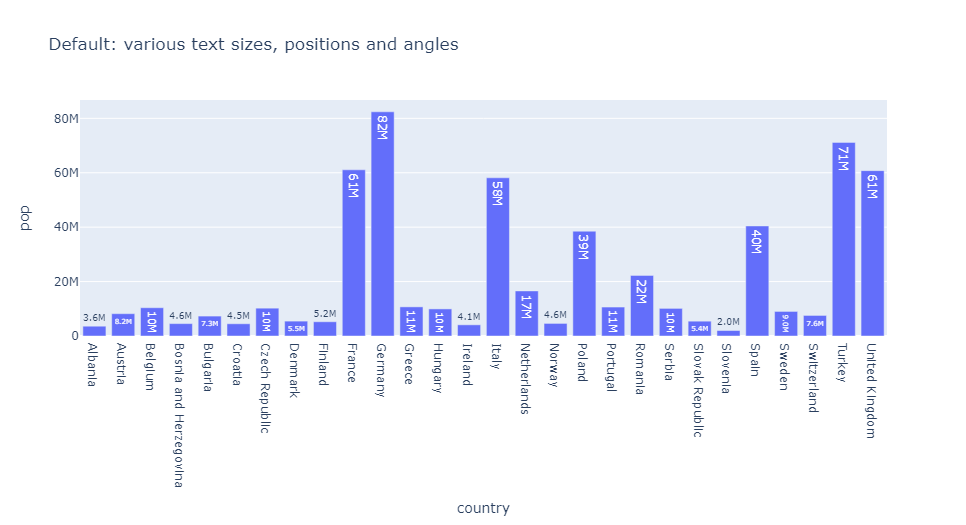
### plotly.express.bar

plotly.express.bar(data\_frame=None, x=None, y=None, color=None, pattern\_shape=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, hover\_name=None, hover\_data=None, custom\_data=None, text=None, base=None, error\_x=None, error\_x\_minus=None, error\_y=None, error\_y\_minus=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, color\_continuous\_scale=None, pattern\_shape\_sequence=None, pattern\_shape\_map=None, range\_color=None, color\_continuous\_midpoint=None, opacity=None, orientation=None, barmode=‘relative’, log\_x=False, log\_y=False, range\_x=None, range\_y=None, text\_auto=False, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **pattern\_shape** | str, int, Series, array 유형 객체 | 막대에 표시할 패턴 설정 |
| **base** | str, int, Series, array 유형 객체 | 막대의 기초값 설정 |
| **pattern\_shape\_sequence** | str의 list | 패턴의 순차 리스트 |
| **pattern\_shape\_map** | str 키와 str 값으로 구성된 dict | 패턴의 매핑 리스트 |
| **barmode** | group', 'overlay', 'relative' | 막대 표현 모드 설정 |
| **text\_auto** | boolean이나 str | 텍스트의 표현 방법 설정 |

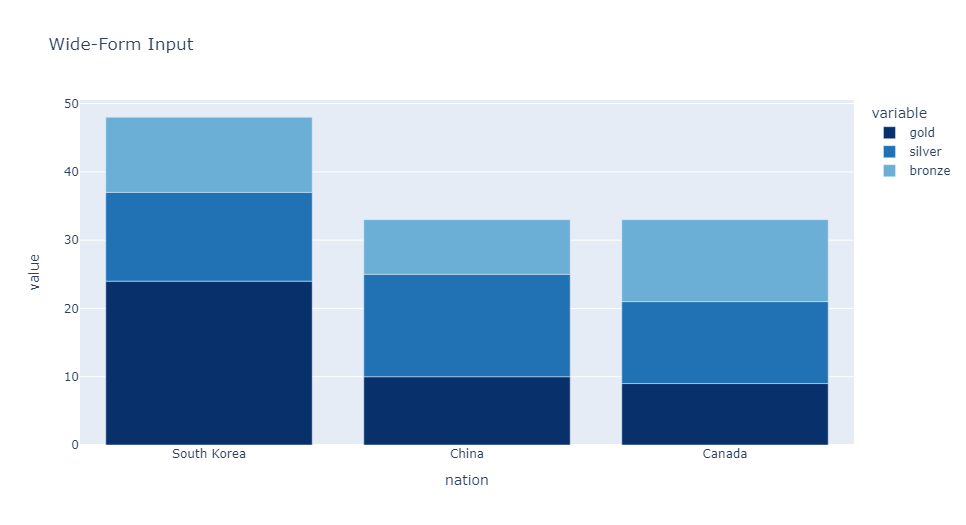
#### 기본 bar plot

import plotly.express as px  
  
df = px.data.gapminder().query("continent == 'Europe' and year == 2007 and pop > 2.e6")  
fig = px.bar(df, y='pop', x='country', text\_auto='.2s',  
 title="Default: various text sizes, positions and angles")  
fig.show()



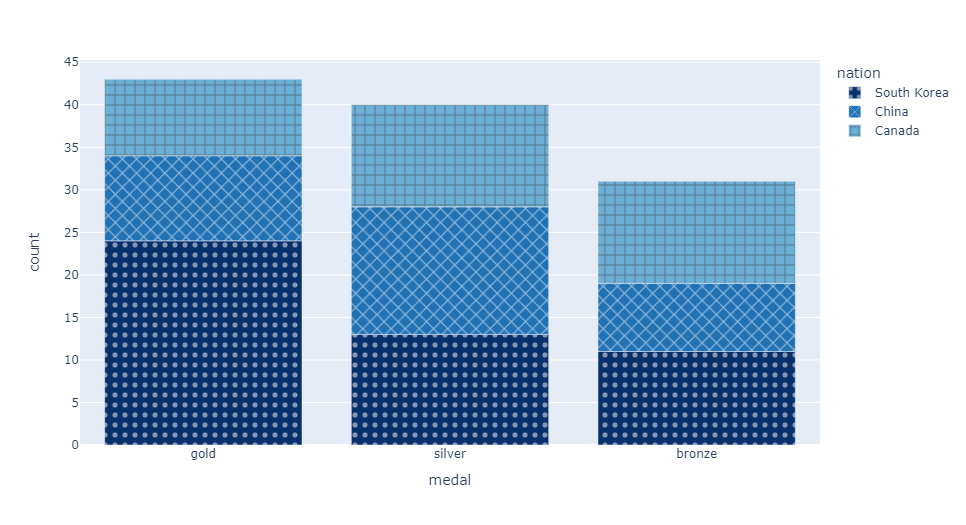
#### 넓은 형태 데이터프레임 사용

import plotly.express as px  
  
wide\_df = px.data.medals\_wide()  
  
fig = px.bar(wide\_df, x="nation", y=["gold", "silver", "bronze"], title="Wide-Form Input")  
fig.show()



#### 패턴 설정

import plotly.express as px  
df = px.data.medals\_long()  
  
fig = px.bar(df, x="medal", y="count", color="nation",  
 pattern\_shape="nation", pattern\_shape\_sequence=[".", "x", "+"])  
fig.show()



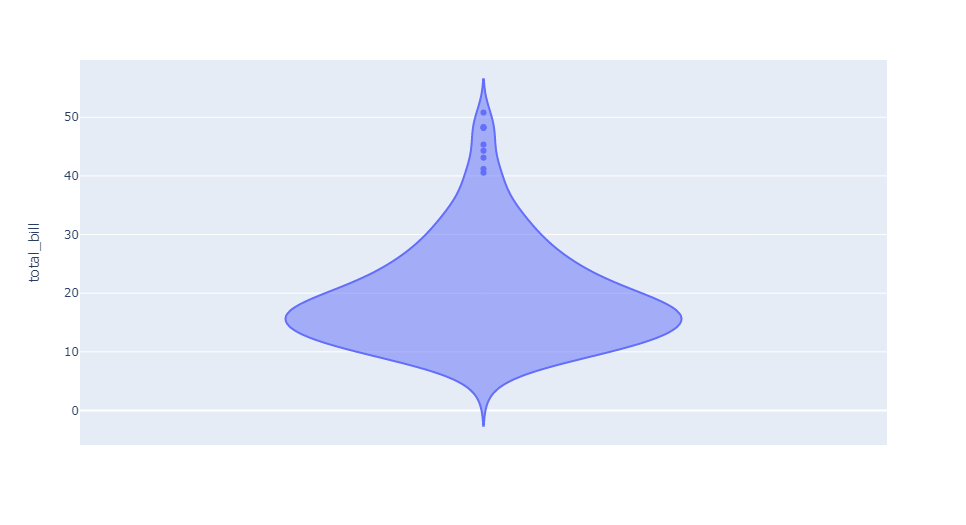
### plotly.express.violin

plotly.express.violin(data\_frame=None, x=None, y=None, color=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, hover\_name=None, hover\_data=None, custom\_data=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, orientation=None, violinmode=None, log\_x=False, log\_y=False, range\_x=None, range\_y=None, points=None, box=False, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **violinmode** | 'group' , 'overlay' | 바이올린 표시 모드 설정 |
| **points** | 'outliers', 'suspectedoutliers', 'all', or False | 아웃라이어 표시 모드 설정 |
| **box** | boolean | 바이올린 내부의 박스 표시 설정 |

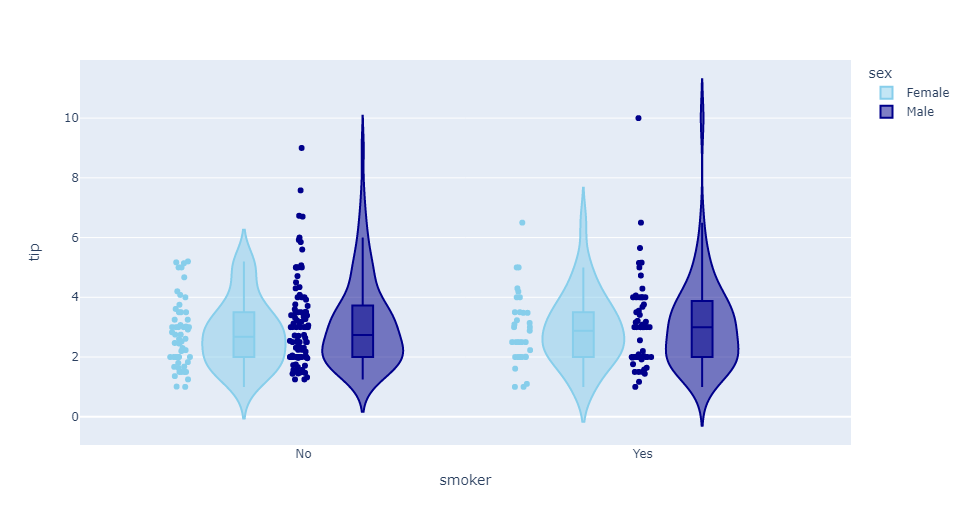
#### 기본 violin plot

import plotly.express as px  
  
df = px.data.tips()  
fig = px.violin(df, y="total\_bill")  
fig.show()



#### 다중 violin plot

import plotly.express as px  
  
df = px.data.tips()  
fig = px.violin(df, y="tip", x="smoker", color="sex", box=True, points="all",  
 hover\_data=df.columns)  
fig.show()



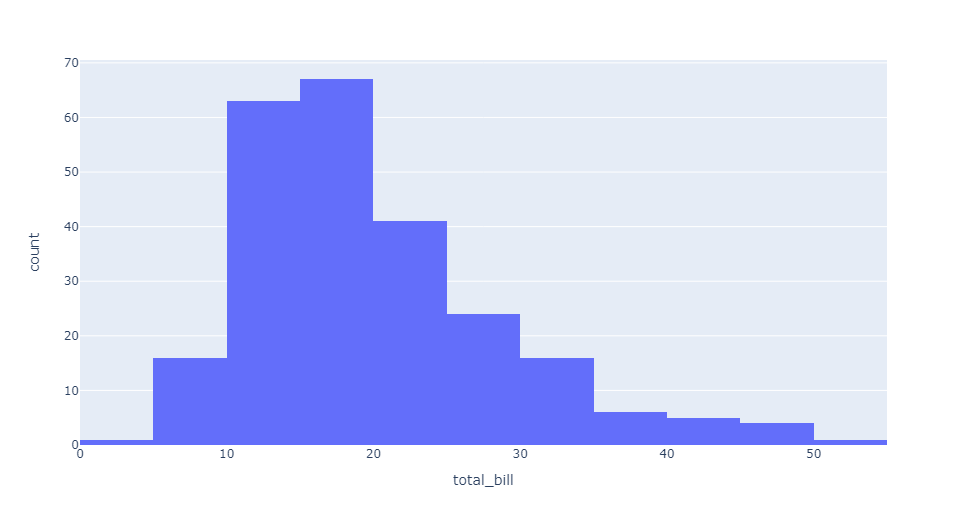
### plotly.express.histogram

plotly.express.histogram(data\_frame=None, x=None, y=None, color=None, pattern\_shape=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, hover\_name=None, hover\_data=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, pattern\_shape\_sequence=None, pattern\_shape\_map=None, marginal=None, opacity=None, orientation=None, barmode=‘relative’, barnorm=None, histnorm=None, log\_x=False, log\_y=False, range\_x=None, range\_y=None, histfunc=None, cumulative=None, nbins=None, text\_auto=False, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **marginal** | 'rug', 'box', 'violin', 'histogram' | 메인 플롯의 옆에 표시되는 서브 플롯 모드의 설정 |
| **barnorm** | 'fraction', 'percent' | 막대의 표준화 방법 설정 |
| **histnorm** | 'percent', 'probability', 'density', or 'probability density' | 히스토그램 표준화 방법 설정 |
| **histfunc** | 'count', 'sum', 'avg', 'min', or 'max' | 히스토그램 함수 설정 |
| **cumulative** | boolean | 누적값 적용 여부 설정 |
| **nbins** | [int](https://docs.python.org/3/library/functions.html#int) | bins의 개수 설정 |

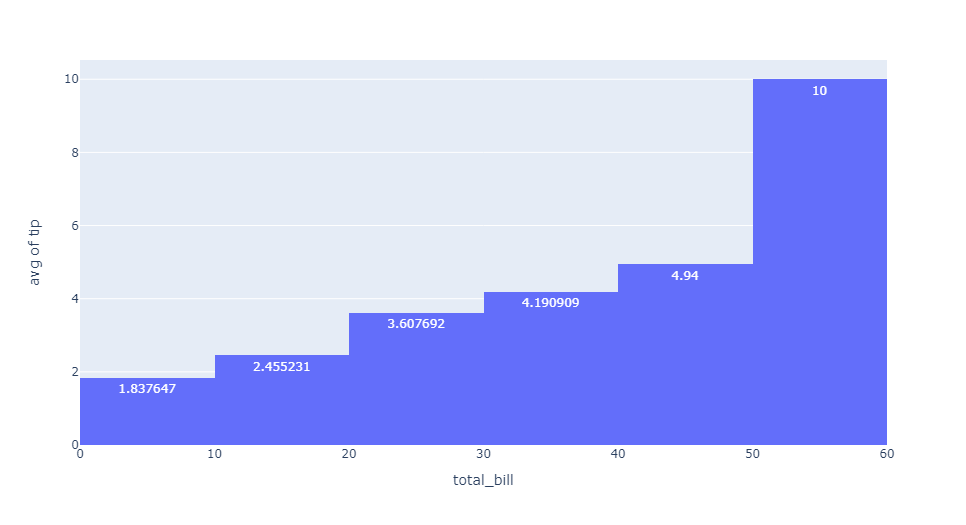
#### 기본 histogram plot

import plotly.express as px  
df = px.data.tips()  
fig = px.histogram(df, x="total\_bill", nbins=20)  
fig.show()



#### text가 표시된 평균 histogram

import plotly.express as px  
df = px.data.tips()  
fig = px.histogram(df, x="total\_bill", y="tip", histfunc="avg", nbins=8, text\_auto=True)  
fig.show()



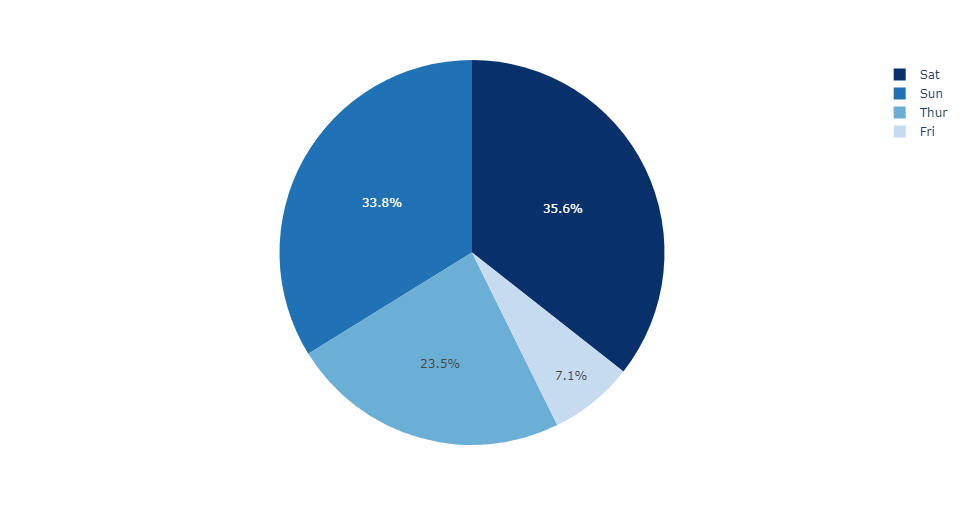
### plotly.express.pie

plotly.express.pie(data\_frame=None, names=None, values=None, color=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, color\_discrete\_sequence=None, color\_discrete\_map=None, hover\_name=None, hover\_data=None, custom\_data=None, category\_orders=None, labels=None, title=None, template=None, width=None, height=None, opacity=None, hole=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **values** | str, int, Series, array 유형 객체 | 섹터와 관련한 값 설정 |
| **hole** | [float](https://docs.python.org/3/library/functions.html#float) | 원 내부의 잘라낼 반지름 설정 |

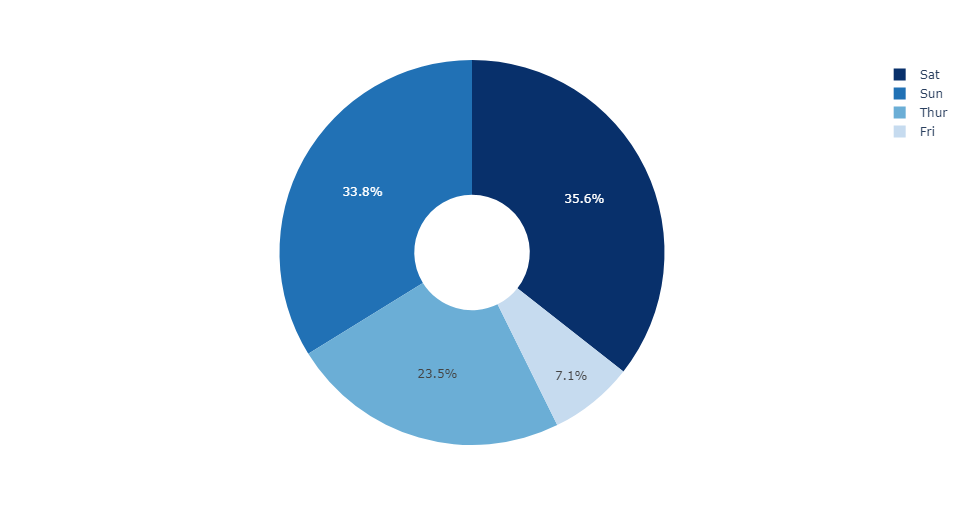
#### 기본 pie plot

import plotly.express as px  
df = px.data.tips()  
fig = px.pie(df, values='tip', names='day')  
fig.show()



#### 도넛 pie

import plotly.express as px  
df = px.data.tips()   
px.pie(df, values='tip', names='day', hole = 0.3)



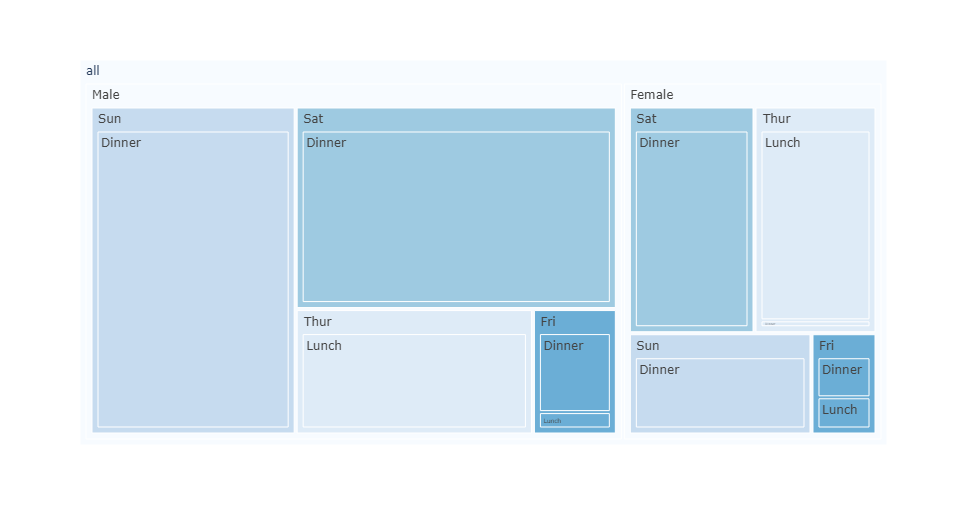
### plotly.express.treemap

plotly.express.treemap(data\_frame=None, names=None, values=None, parents=None, ids=None, path=None, color=None, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, color\_discrete\_sequence=None, color\_discrete\_map=None, hover\_name=None, hover\_data=None, custom\_data=None, labels=None, title=None, template=None, width=None, height=None, branchvalues=None, maxdepth=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| parents | str, int, Series, array 유형 객체 | sunburst와 treemap charts의 부모노드 설정 |
| path | [str이나 int의 list, Series, array 유형 객체](https://docs.python.org/3/library/functions.html#int) | 루트로부터 리프까지의 섹터 구조 설정 |
| branchvalues | ‘total’ , ‘remainder’ | 브랜치 표시 모드 설정 |
| maxdepth | [int](https://docs.python.org/3/library/functions.html#int) | 섹터의 최대 깊이 설정 |

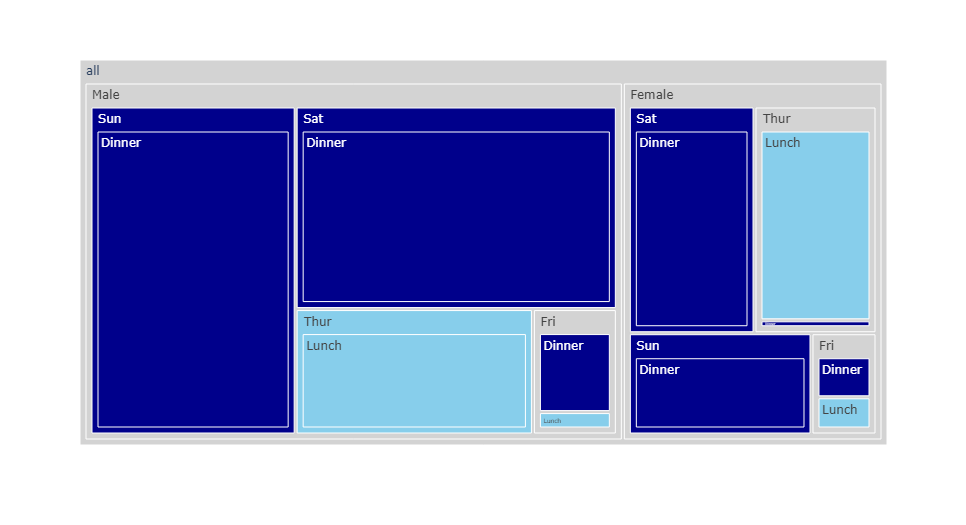
#### 기본 treemap plot

import plotly.express as px  
df = px.data.tips()  
fig = px.treemap(df, path=[px.Constant("all"), 'sex', 'day', 'time'],   
 values='total\_bill', color='day')  
fig.show()



#### 컬러 매핑

import plotly.express as px  
df = px.data.tips()  
fig = px.treemap(df, path=[px.Constant("all"), 'sex', 'day', 'time'],   
 values='total\_bill', color='time',  
 color\_discrete\_map={'(?)':'lightgrey', 'Lunch':'gold', 'Dinner':'darkblue'})  
fig.show()

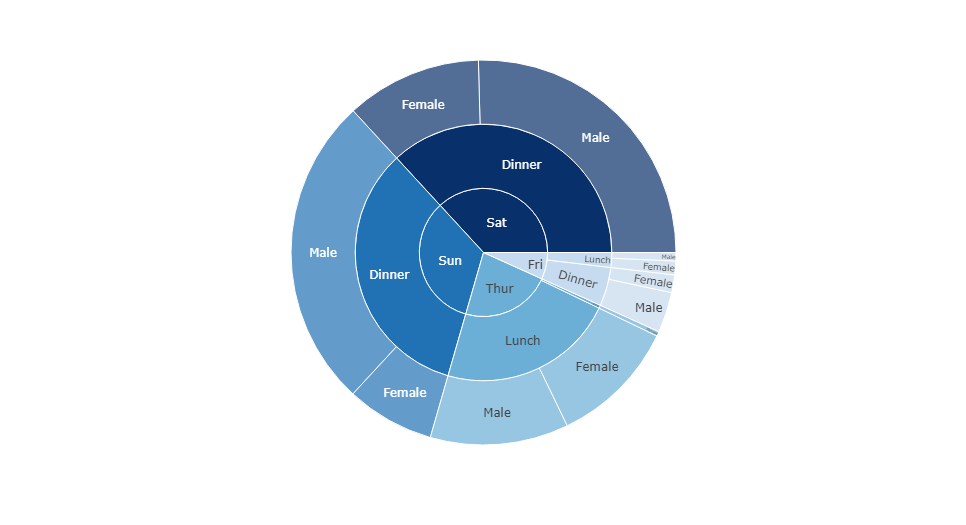


### plotly.express.sunburst

plotly.express.sunburst(data\_frame=None, names=None, values=None, parents=None, path=None, ids=None, color=None, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, color\_discrete\_sequence=None, color\_discrete\_map=None, hover\_name=None, hover\_data=None, custom\_data=None, labels=None, title=None, template=None, width=None, height=None, branchvalues=None, maxdepth=None)

#### 기본 sunburst plot

import plotly.express as px  
df = px.data.tips()  
fig = px.sunburst(df, path=['day', 'time', 'sex'], values='total\_bill')  
fig.show()

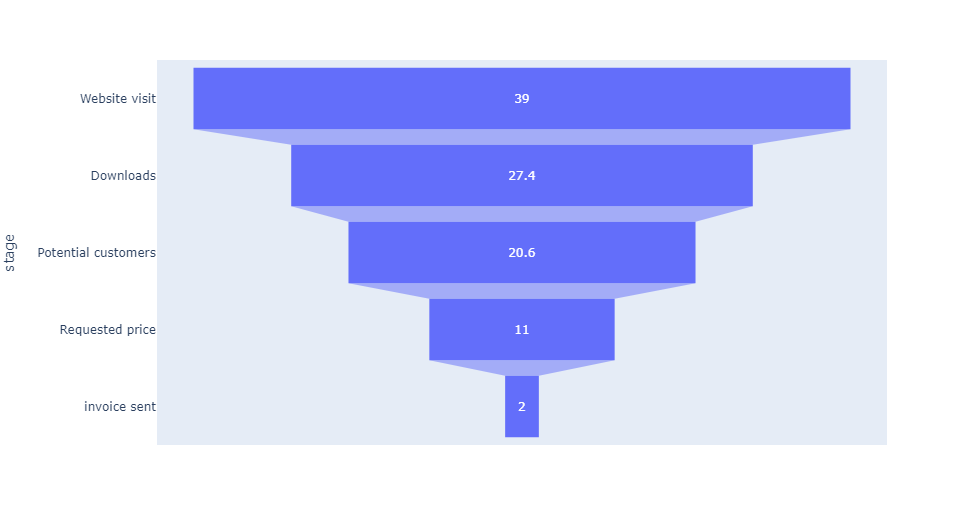


### plotly.express.funnel

plotly.express.funnel(data\_frame=None, x=None, y=None, color=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, hover\_name=None, hover\_data=None, custom\_data=None, text=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, opacity=None, orientation=None, log\_x=False, log\_y=False, range\_x=None, range\_y=None, title=None, template=None, width=None, height=None)

#### 기본 funnel plot

import plotly.express as px  
data = dict(  
 number=[39, 27.4, 20.6, 11, 2],  
 stage=["Website visit", "Downloads", "Potential customers", "Requested price", "invoice sent"])  
fig = px.funnel(data, x='number', y='stage')  
fig.show()



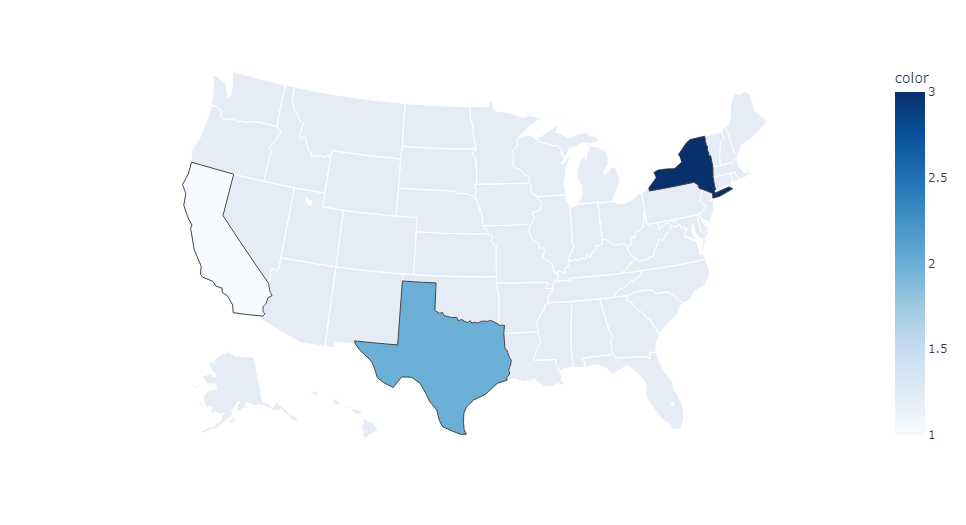
### plotly.express.choropleth

plotly.express.choropleth(data\_frame=None, lat=None, lon=None, locations=None, locationmode=None, geojson=None, featureidkey=None, color=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, facet\_row\_spacing=None, facet\_col\_spacing=None, hover\_name=None, hover\_data=None, custom\_data=None, animation\_frame=None, animation\_group=None, category\_orders=None, labels=None, color\_discrete\_sequence=None, color\_discrete\_map=None, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, projection=None, scope=None, center=None, fitbounds=None, basemap\_visible=None, title=None, template=None, width=None, height=None)

|  |  |  |
| --- | --- | --- |
| 매개변수 | 매개변수타입 | 설명 |
| **lat, lon** | str, int, Series, array 유형 객체 | 마크가 위치할 위도, 경도 설정 |
| **locations** | str, int, Series, array 유형 객체 | location mode에 의해 결정되는 위도, 경도 매핑 값 |
| **locationmode** | ‘ISO-3’, ‘USA-states’, ‘country names’ | locations의 위도, 경도를 해석할 방법 설정 |
| **geojson** | GeoJSON 포맷의 dict | ID를 포함한 Polygon feature collection |
| **featureidkey** | str | GeoJSON feature object에서의 locations에 전달되어 매칭될 경로 필드 설정 |
| **projection** | equirectangular', 'mercator', 'orthographic', 'natural earth', … | 프로젝션 방법의 설정 |
| **scope** | 'world', 'usa', 'europe', 'asia', 'africa', 'north america', or 'south america' | 지도에 표시될 범위 설정 |
| **center** | [dict](https://docs.python.org/3/library/stdtypes.html#dict) | 지도의 중심 위도, 경도 설정 |
| **basemap\_visible** | [bool](https://docs.python.org/3/library/functions.html#bool) | 기본 맵 표시 여부 설정 |

#### 기본 choropleth plot

import plotly.express as px  
  
fig = px.choropleth(locations=["CA", "TX", "NY"], locationmode="USA-states", color=[1,2,3], scope="usa")  
fig.show()



1. 앞선 함수에서 설명한 매개변수의 설명은 생략함 [↑](#footnote-ref-1)