<https://plotly.com/python/>

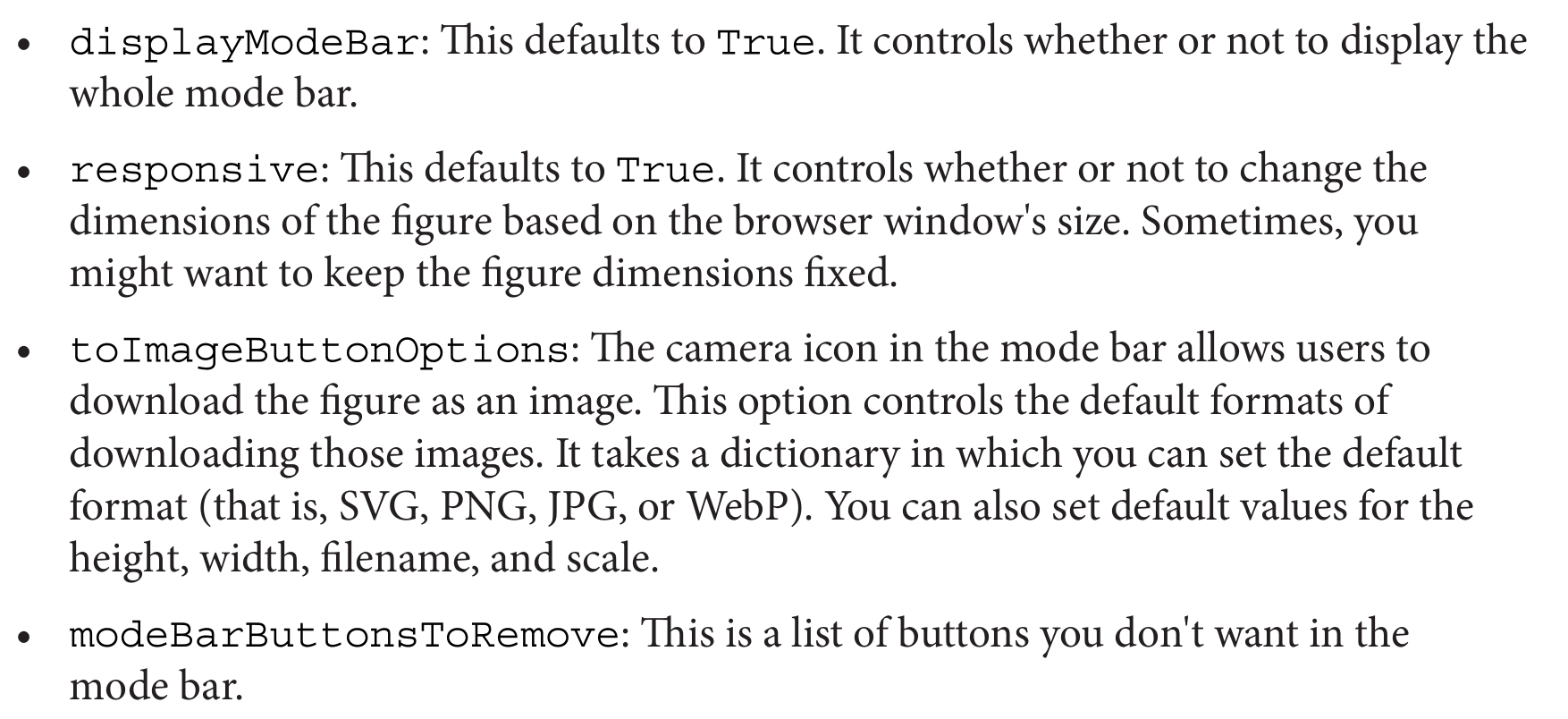
**显示图像**

fig.show(\*config={参数字典})

#config中的参数可用于控制显示图像中的元素、图像尺寸、按钮样式和功能等

config={‘displayModeBar’:True, ’responsive’:True, ‘toImageButtonOptions’: {'format': 'svg'},

’modeBarButtonsToRemove’:…}



**查看图像树结构**

fig.show(‘json’)

**保存为网页**

fig.write\_html(‘file.html’,config={'toImageButtonOptions':{'format': 'svg'} })

**保存图片**

fig.write\_image('file.svg',height=600, width=850)

EXPRESS模块

[Plotly Express](https://plotly.com/python/plotly-express/)

import plotly.express as px

fig=绘图命令

fig.show()

## 通用属性

title #标题

color #上色特征

size #尺寸特征

size\_min,size\_max #尺寸大小区间

symbol #形状特征

text #文本特征

pattern\_shape #形状填充特征

hover\_name #悬停显示特征

custom\_data #后续画图操作可能引用的数据列，用custom\_data[index]调用

facet\_col, facet\_row #列方向分面特征，行方向分面特征

facet\_col\_wrap, facet\_row\_wrap #分面列数，分面行数

range\_x,range\_y #x,y轴范围

labels={‘xcolname’:’xlab’,’ycolname’:’ylab’,’legendcolname’:’legend’} #坐标轴名/图例名

log\_x, log\_y #对数坐标轴

opacity #透明度

template=["plotly","plotly\_white","plotly\_dark","ggplot2","seaborn","simple\_white","none"]

#主题

width=400, height=400 #图宽图高

marginal\_x,marginal\_y="rug","box","violin" #添加辅图 (适用于描述分布的图形如直方、散点)

animation\_frame #动态动画列

color\_discrete\_map={"level1": "blue", " level2": "green", " level3": "red"} #离散图例颜色

**删除动画播放功能**

fig["layout"].pop("updatemenus") # optional, drop animation buttons

## 散点图

**二维散点图**

fig=px.scatter(data\_frame=data,x=’col1’,y=’col2’,

hover\_data, error\_x, error\_y,

marginal\_x,marginal\_y=[‘histogram’,’rug’,’box’,’violin’,],

trendline=[‘ols’,’lowess’],

size\_max)

#hover\_data是额外添加到点tooltip信息中的列

#error\_x和error\_y是各点在x,y方向的误差区间半径列

#marginal\_x,marginal\_y用于在x,y轴边绘制图表

**三维散点图**

px.scatter\_3d(data,x,y,z)

## 折线图

**二维折线图**

fig=px.line(df,x,y,markers=True)

#markers设置是否在折线图上显示数据点

#在绘制时间序列时，将时间列转化为Datetime类型

**三维折线图**

px.line\_3d(data, x,y,z)

## 面积图

px.area()

## 条形图

**分堆条形图**

px.bar(df,color='col1', color=上色特征, barmode='group')

**堆叠条形图**

px.bar(df,color='col1', color=上色特征)

**聚合条形图(使用hist实现)**

px.histogram(df,x,y,\*histfunc=’sum’/’avg’/’count’/’min’/’max’)

#x,y均有输入时，对y进行聚合

**聚合堆叠条形图**

px.histogram(df, x="col1", y="col2", color="col3", pattern\_shape="col3")

**水平方向聚合条形图**

px.histogram(df, y="col1")

**顺序排列**

update\_xaxes(categoryorder='total ascending'/'total descending')

## 直方图

[Histograms (plotly.com)](https://plotly.com/python/histograms/)

fig = px.histogram(df, x="col1",\*nbins)

**调节间距：**fig.update\_layout(bargap=0.2)

**设置因素水平顺序：**px.histogram(df, x="col1", category\_orders=dict(col1=[level1,level2,…]))

**密度/概率直方图：**px.histogram(df, x="col1",histnorm='percent'/'probability'/'density'/ 'probability density')

**显示数据标签：**参数text\_auto=True

## 箱型图/小提琴图

[Box Plots (plotly.com)](https://plotly.com/python/box-plots/)

fig = px.violin(df,x,y)

fig=px.box(df,x,y)

## Pairsplot

[Splom (plotly.com)](https://plotly.com/python/splom/)

fig = px.scatter\_matrix(df, dimensions=要显示的列)

**隐藏对角线**

fig.update\_traces(diagonal\_visible=False)

## Alluvial Plot

[Parallel Categories Diagram (plotly.com)](https://plotly.com/python/parallel-categories-diagram/)

## 蠕虫图

[Strip Charts (plotly.com)](https://plotly.com/python/strip-charts/)

fig = px.strip(df,x,y,labels,width=400,height=400)

## ECDF

[Ecdf Plots (plotly.com)](https://plotly.com/python/ecdf-plots/)

## 热力图

[2d Histogram (plotly.com)](https://plotly.com/python/2D-Histogram/)

## 分层饼图

[Sunburst Charts (plotly.com)](https://plotly.com/python/sunburst-charts/)

## 树状图

[Treemaps (plotly.com)](https://plotly.com/python/treemaps/)

[Icicle Charts (plotly.com)](https://plotly.com/python/icicle-charts/)

fig = px.treemap(df,path=[‘col1’,’col2’,…], values="col")

## 漏斗图

[Funnel Charts (plotly.com)](https://plotly.com/python/funnel-charts/)

## 地理散点图

[Scattermapbox (plotly.com)](https://plotly.com/python/scattermapbox/) #小地图

[Scatter Plots On Maps (plotly.com)](https://plotly.com/python/scatter-plots-on-maps/) #大地图

## 地理热力图

<https://plotly.com/python/maps/> #地理图galary

[Choropleth Maps (plotly.com)](https://plotly.com/python/choropleth-maps/)

fig = px.choropleth\_mapbox(df, geojson=counties, locations, color,

color\_continuous\_scale="Viridis",

range\_color=(0, 12),mapbox\_style="carto-positron",

zoom=3, center = {"lat": 37.0902, "lon": -95.7129},

opacity=0.5)

#geojson传入一个对地理元素多边形进行编码的geojson类型文件

#locations通常为对应地理元素多边形的id列

#color为通过颜色显示在地理图上的值列

## 添加动画效果

[Animations (plotly.com)](https://plotly.com/python/animations/)

加参数 animation\_frame=作为动态轴的列, animation\_group=不变的列

GO模块

**初始化图像**

import plotly.graph\_objects as go

fig=go.Figure()

**添加图层**

fig.add\_trace(go.Figure(…))或fig.add\_scatter(),fig.add\_bar()等

**修改layout**

1. 通过链式引用属性来修改 ②fig.update\_layout(attribute=value)
2. 对应的fig.update\_Object函数来修改

## 散点图/折线图

fig = go.Figure(

data=go.Scatter(x, y, mode=['markers','lines+markers',’lines’],

marker\_color=颜色列,text=文本列,

marker=dict(size,color,colorscale=调色板,showscale=True),

line=dict(color,width))

)

#marker中的字典用于设置点的属性,line中的字典用于设置线的属性

## 蜡烛图

fig=go.Figure()

fig.add\_trace(go.Candlestick(x,open,high,low,close)))

layout，traces

**添加额外的散点、折线**

fig.add\_trace(go.Scatter(x,y,mode=['markers','lines+markers',’lines’],name=在图例中的名字))

**修改图元素尺寸(点大小,轮廓线等)**

fig.update\_traces(marker=dict(size=1), line=dict(width=2,color='DarkSlateGrey'))

**update\_layout函数**

fig.update\_layout(Object1={attr1:value1,…, attrn:valuen},

…,

Objectn={attr1:value1,…, attrn:valuen})

fig.update\_layout(

showlegend=False,

template='ggplot2', #模板名

updatemenus=[

dict(

type="buttons",

direction="down",

active=0,

x=1.2,

y=0.75,

buttons=buttons\_appliances),

dict(

direction="down",

active=0,

x=1.2,

y=0.95,

buttons=buttons\_type)],

annotations=[

dict(text="Select Appliance", showarrow=False,font=dict(size=14),x=1.2, y=0.83, yref="paper", align="left",xref='paper'),

dict(text="Select Plot Type", showarrow=False,font=dict(size=14),x=1.2, y=1.02, yref="paper", align="left",xref='paper')

]

)

**修改标题**

fig.update\_layout(

title={

'text': "title",

'y':0.95,

'x':0.46,

'xanchor': 'center',

'yanchor': 'top'})

**副标题**

fig.update\_layout(

title=go.layout.Title(text="Plot Title <br><sup>Plot Subtitle</sup>"))

**修改X轴**

fig.update\_xaxes(rangeslider\_visible=True,…)

fig.update\_layout(

xaxis = dict(

title\_text="xlab,

tickmode = 'array',

tickvals = position\_list,

ticktext = label\_list,

tickangle=角度,

tickfont=dict(family='Ariel', color, size),

rangeslider=dict(visible=True),

type=’log’)

**修改Y轴**

fig.update\_yaxes(rangeslider\_visible=True,…)

fig.update\_layout(

yaxis = dict(

title\_text="ylab",

tickmode = 'array',

tickvals = position\_list,

ticktext = label\_list,

tickangle=角度,

tickfont=dict(family='Ariel', color, size),

rangeslider=dict(visible=True),

type=’log’)

**同一设置标题和坐标轴标题**

fig.update\_layout(title=””, xaxis\_title="", yaxis\_title="")

**设置文本位置**

fig.update\_traces(textposition="bottom right")

**选择模板**

fig.update\_layout(template=["plotly","plotly\_white","plotly\_dark","ggplot2","seaborn","simple\_white","none"])

**plot背景色**

fig.update\_layout(paper\_bgcolor="#CCCCCC")

**隐藏图例**

fig.update\_layout(showlegend=False)

**加多条文本标注**

fig.update\_layout(

annotations=[

dict(text="Text1", showarrow=False,font=dict(size=14,color),x=1.2, y=1, yref="paper", align="left",xref='paper', textangle=0),

dict(text="Text2", showarrow=False,font=dict(size=14,color),x=0.2, y=1, yref="paper", align="left",xref='paper', textangle=0)]

)

**定义交互按钮**

buttons =[

dict(label='按钮名称',

method="update"/"restyle"/"relayout"/"animate",

args=[{'attr':value}]

)

]

* "restyle": modify data or data attributes
* "relayout": modify layout attributes
* "update": modify data and layout attributes; combination of "restyle" and "relayout"
* "animate": start or pause an animation)

**添加按钮**

fig.update\_layout(

updatemenus=[

dict(type="buttons",

direction="down",

active=0,

x=1.2,

y=0.75,

buttons=buttons)]

)

**添加下拉槽**

fig.update\_layout(

updatemenus=[

dict(direction="down",

active=0,

x=1.2,

y=0.95,

buttons=buttons)]

)

**查看默认hovertemplate**

fig.data[0].hovertemplate

**自定义tooltip格式**

import plotly.express as px

fig=px.line(data\_frame=pd.DataFrame({'X':x,'Y':y,'Z':z}),x='FPR',y='TPR',custom\_data=['z'])

fig.update\_traces(

hovertemplate="<br>".join([

"z: %{customdata[0]}",

"x: %{x}",

"y: %{y}"

])

)

fig.show()