1. Elementos básicos do Plotly

September 11, 2023

1 Introdução ao Plotly

O Plotly é uma biblioteca muito completa e moderna, notável por suas capacidades interativas.

Nesta seção do curso, aprenderemos a usar o Plotly para visualização de dados. Primeiramente aprenderemos os conceitos básicos por trás do funcionamento dos gráficos criados. Após isso, entenderemos como editar os elementos principais de nossos gráficos e, por último, aprenderemos os principais tipos de visulizações disponíveis.

1.1 Instalando o Plotly

Para instalar o Plotly, basta executar o seguinte comando:

```
pip install plotly==4.12.0
```

ou

```
pip3 install plotly==4.12.0
```

1.2 Adicionar suporte ao Jupyter Notebook

```
pip install "notebook>=5.3" "ipywidgets>=7.2"
```

1.3 Suporte ao Jupyter Lab

```
$ pip install jupyterlab "ipywidgets>=7.5"
```

1.4 O elemento Figure

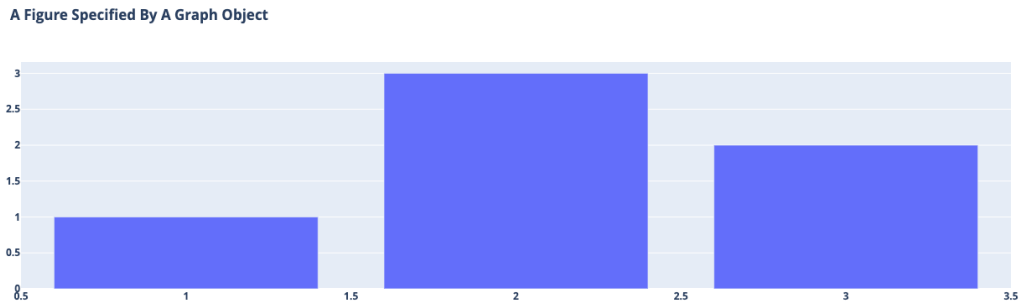
O elemento básico de criação de gráficos no Plotly se chama `Figure` e pode ser encontrado em `plotly.graph_objects.Figure`. Em sua construção, passamos um dicionário com até 3 chaves: - **data**: É a estrutura que conterá nossos dados, bem como o formato na qual os mesmos serão apresentados. Tais valores devem estar dispostos na forma de uma lista, internamente chamados de "traces". - **layout**: Responsável por controlar aspectos visuais de nosso gráfico. - **frames**: Atributo responsável por controlar eventuais animações que desejamos fazer com nossos gráficos.

Assim, o objeto Figure deve ter a seguinte estrutura:

```
Figure({     'data': ...     'layout': ...     'frames': ...     },
```

[1]:
```
import plotly
```

[10]:
```
import plotly.graph_objects as go

fig = go.Figure(
    data=[go.Bar(x=[1, 2, 3], y=[1, 3, 2])],
    layout=go.Layout(
        title=go.layout.Title(text="A Figure Specified By A Graph Object")
    )
)

fig.show()
```



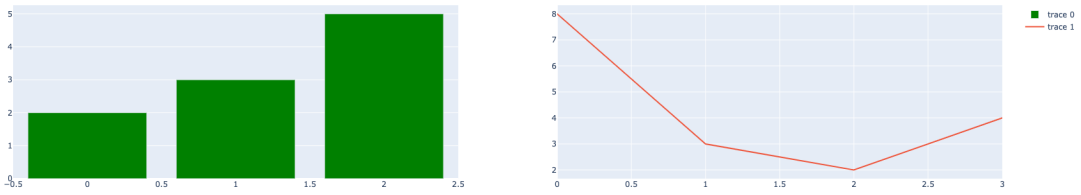
Também é possível criar um gráfico apenas passando um dicionário como atributo construtor para a classe Figure

[ ]:
```
import plotly.graph_objects as go

dict_of_fig = dict({
    "data": [{"type": "bar",
              "x": [1, 2, 3],
              "y": [1, 3, 2]}],
    "layout": {"title": {"text": "A Figure Specified By A Graph Object With A␣
 ↪Dictionary"}}
})

fig = go.Figure(dict_of_fig)

fig.show()
```

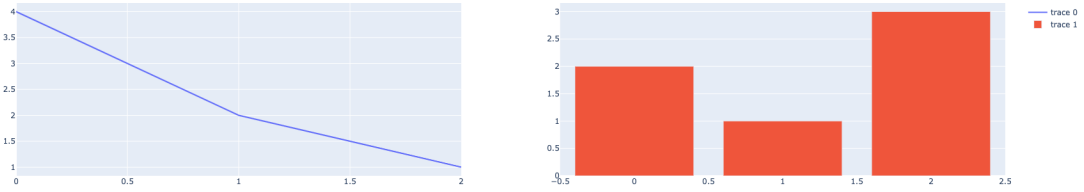
1.5 Criando Subplots

A função plotly.subplots.make_subplots () produz uma figura de objeto gráfico que é pré-configurada com uma grade de subplots aos quais os `traces` podem ser adicionados. A função `add_trace()` será discutida mais tarde com mais detalhes.

[16]:
```
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2)

fig.add_trace(go.Bar(y=[2, 3, 5], marker_color="green"), row=1, col=1)
fig.add_trace(go.Scatter(y=[8, 3, 2, 4], mode="lines"), row=1, col=2)

fig.show()
```



[18]:
```
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2)

fig.add_scatter(y=[4, 2, 1], mode="lines", row=1, col=1)
fig.add_bar(y=[2, 1, 3], row=1, col=2)

fig.show()
```

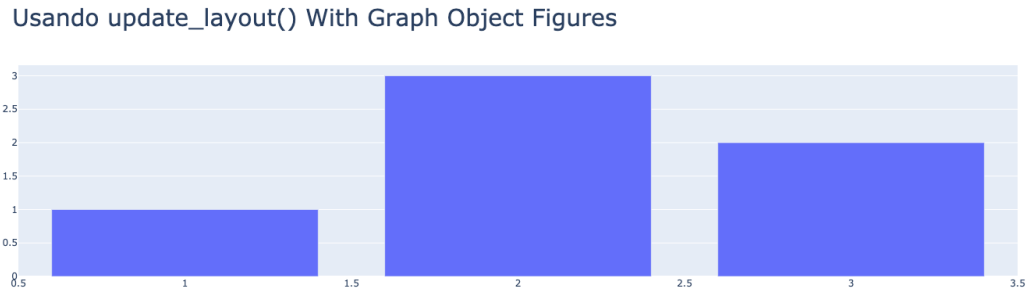


1.6 Atualizando Figuras

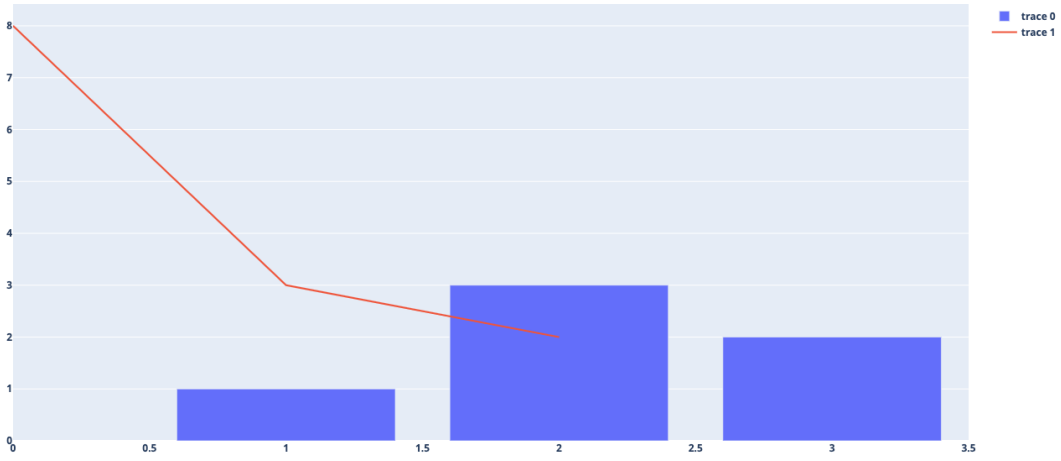
Podemos atualizar atributos do elemento `layout` com o método `update_layout()`

[19]:
```
import plotly.graph_objects as go

fig = go.Figure(data=go.Bar(x=[1, 2, 3], y=[1, 3, 2]))
```

```
fig.update_layout(title_text="Usando update_layout() With Graph Object Figures",
                  title_font_size=30)

fig.show()
```



[24]:
```
import plotly.graph_objects as go

fig = go.Figure(
    data=[
        go.Bar(x=[1, 2, 3], y=[1, 3, 2]),
        go.Scatter(y=[8, 3, 2], mode="lines")
    ]
)

fig.update_layout(height=700)
fig.show()
```



1.7 Atualizando propriedades

[1]:
```
import pandas as pd
```

[15]:
```
df = pd.DataFrame()
```

[16]:
```
df["Municípios"] = ["a", "b", "c", "d"]
df["teste"] = [1, 2, 3, 4]
```

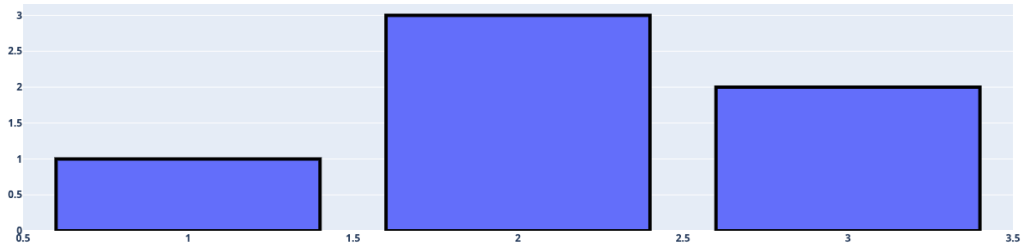
[18]:
```
df[df["Municípios"].isin(["a", "b"])]
```

[18]:

	Municípios	teste
0	a	1
1	b	2

[ ]:
```
df_municipios = df.loc[df["Mu"]]
```

[25]:
```
import plotly.graph_objects as go

fig = go.Figure(data=go.Bar(x=[1, 2, 3], y=[1, 3, 2]))

fig.data[0].marker.line.width = 4
fig.data[0].marker.line.color = "black"

fig.show()
```



Mais informações sobre como renderizar sua imagem podem ser encontrados aqui: https://plotly.com/python/renderers/

2. Gráficos Básicos

September 11, 2023

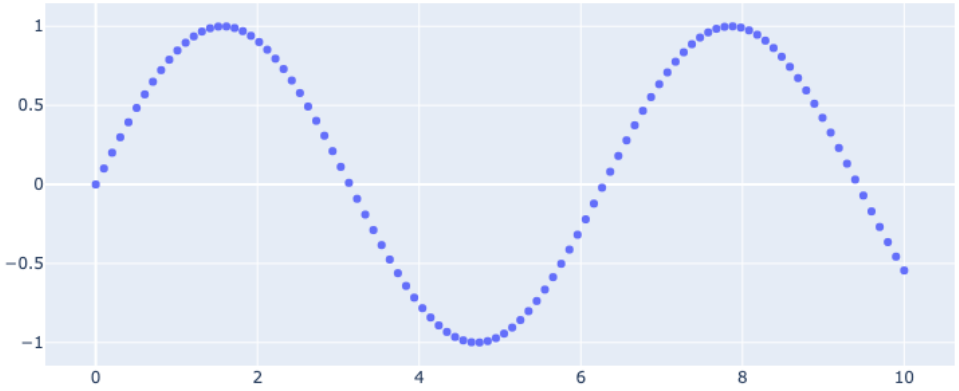
1 Gráficos básicos no Plotly

Nessa seção aprenderemos como criar alguns dos gráficos mais básicos disponíveis na biblioteca.

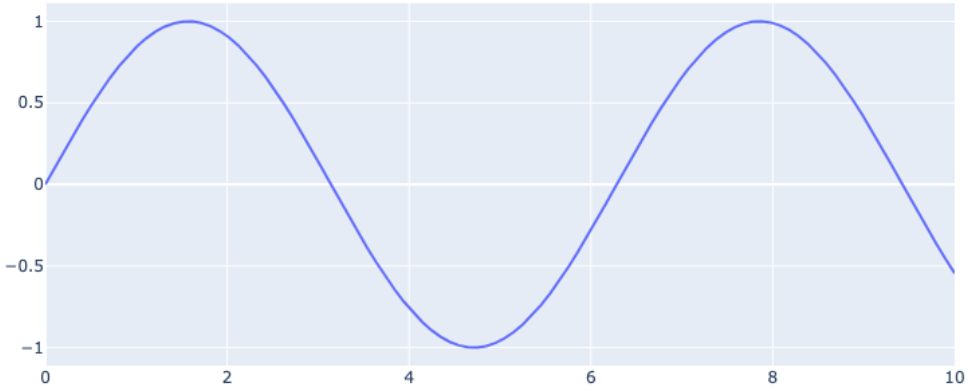
1.1 Scatter Plots

Referência: https://plotly.com/python/line-and-scatter/

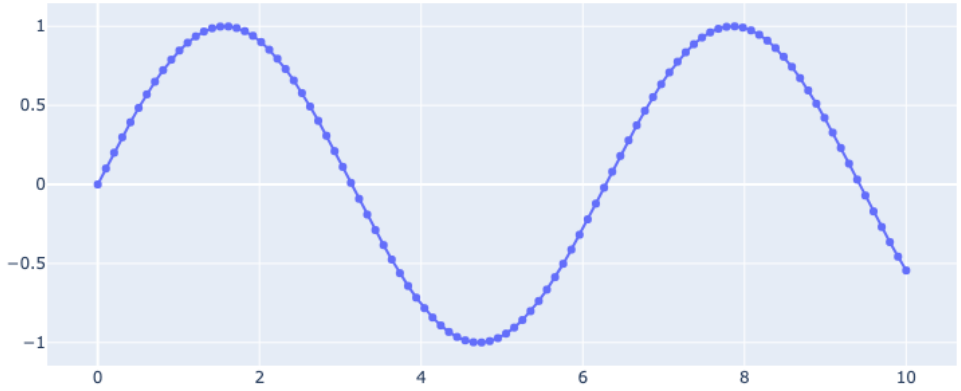
[1]:
```
import plotly.graph_objects as go
import numpy as np

t = np.linspace(0, 10, 100)
y = np.sin(t)

fig = go.Figure(data=go.Scatter(x=t, y=y, mode='markers'))
fig.show()
```



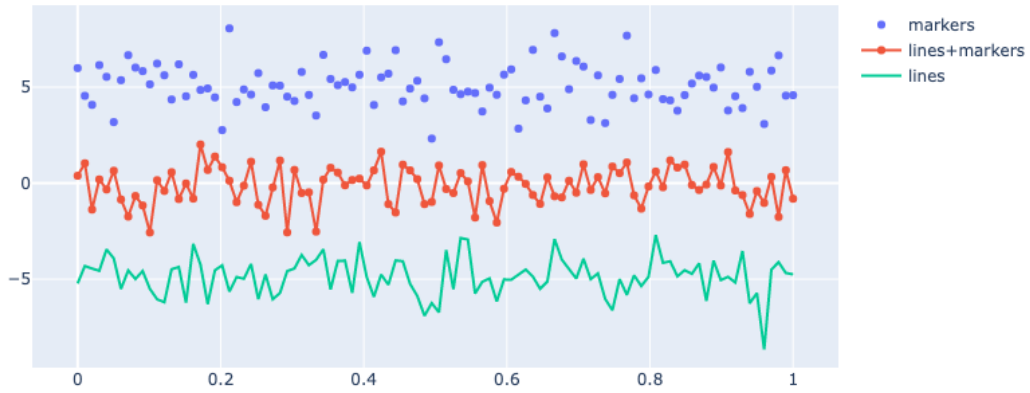
[3]:
```
t = np.linspace(0, 10, 100)
y = np.sin(t)

fig = go.Figure(data=go.Scatter(x=t, y=y, mode='lines'))
fig.show()
```



[12]:
```
t = np.linspace(0, 10, 100)
y = np.sin(t)

fig = go.Figure(data=go.Scatter(x=t, y=y, mode='lines+markers'))
fig.show()
```

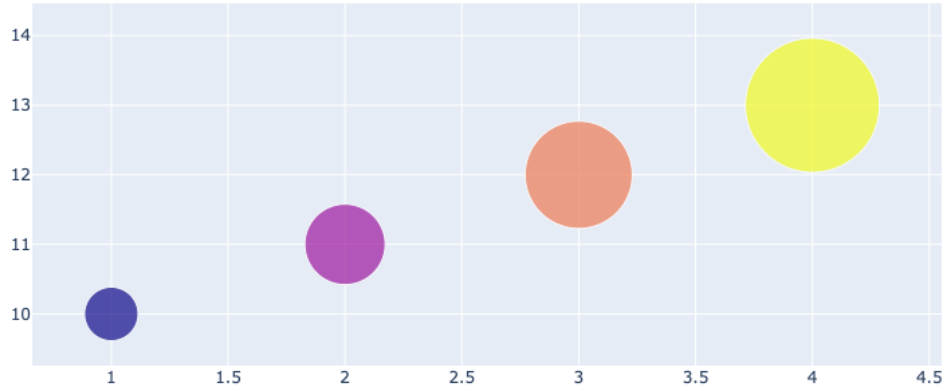


[4]:
```
N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N) + 5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N) - 5

fig = go.Figure()

# Add traces
fig.add_trace(go.Scatter(x=random_x, y=random_y0,
                    mode='markers',
                    name='markers'))

fig.add_trace(go.Scatter(x=random_x, y=random_y1,
                    mode='lines+markers',
                    name='lines+markers'))

fig.add_trace(go.Scatter(x=random_x, y=random_y2,
                    mode='lines',
                    name='lines'))

fig.show()
```



1.1.1 Gráficos bolha

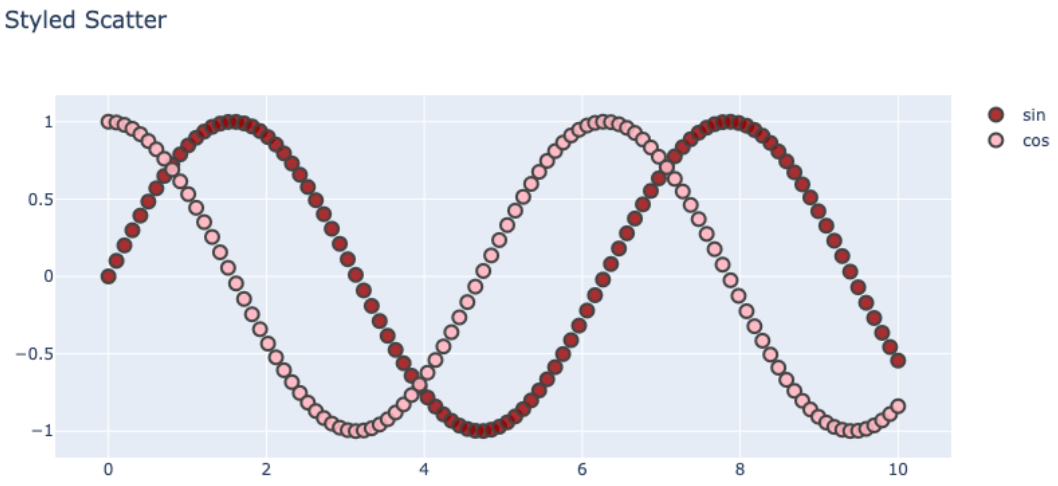
[5]:
```
import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(
    x=[1, 2, 3, 4],
    y=[10, 11, 12, 13],
    mode='markers',
    marker=dict(size=[40, 60, 80, 100],
                color=[0, 1, 2, 3]),
    hovertemplate="R$ %{y} - %{marker.size}",
    text=["item A", "item B", "item C", "item D"]
))

fig.show()
```



1.1.2 Estilizando Scatter Plots

[10]:
```
import plotly.graph_objects as go
import numpy as np


t = np.linspace(0, 10, 100)
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=t, y=np.sin(t),
    name='sin',
    mode='markers',
    marker_color='rgba(152, 0, 0, .8)'
))

fig.add_trace(go.Scatter(
    x=t, y=np.cos(t),
    name='cos',
    marker_color='rgba(255, 182, 193, .9)'
))

fig.update_traces(mode='markers', marker_line_width=2, marker_size=10)
fig.update_layout(title='Styled Scatter',
                  yaxis_zeroline=False, xaxis_zeroline=False)
```

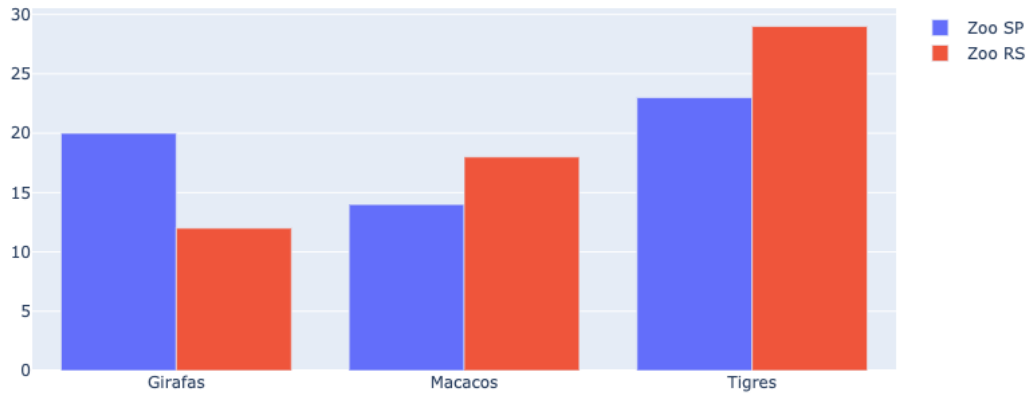
```
fig.show()
```



1.2 Bar Charts

Referência: https://plotly.com/python/bar-charts/

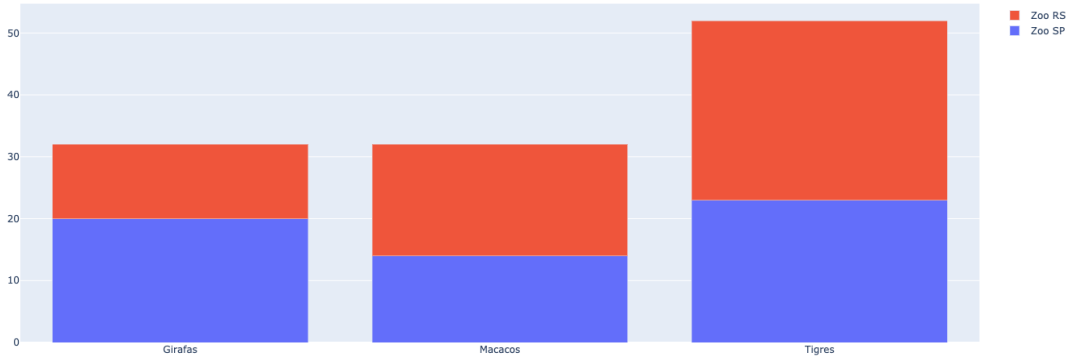
[15]:
```
import plotly.graph_objects as go
animais=['Girafas', 'Macacos', 'Tigres']

fig = go.Figure(data=[
    go.Bar(name='Zoo SP', x=animais, y=[20, 14, 23]),
    go.Bar(name='Zoo RS', x=animais, y=[12, 18, 29])
])
# Change the bar mode
fig.update_layout(barmode='group')
fig.show()
```



1.2.1 Empilhando barras

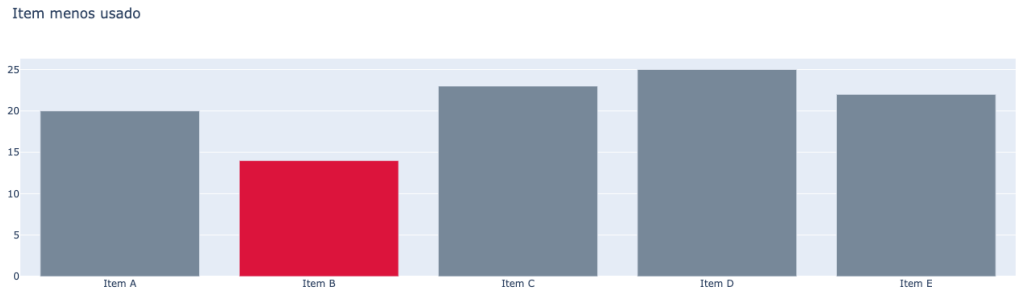
Também é possível apresentar os dados de forma empilhada.

[19]:
```
import plotly.graph_objects as go
animais=['Girafas', 'Macacos', 'Tigres']

fig = go.Figure(data=[
    go.Bar(name='Zoo SP', x=animais, y=[20, 14, 23]),
    go.Bar(name='Zoo RS', x=animais, y=[12, 18, 29])
])

# Change the bar mode
fig.update_layout(barmode='stack', height=600)
fig.show()
```



1.2.2 Estilizando barras individualmente

[23]:
```
colors = ['lightslategray',] * 5
colors[1] = 'crimson'

fig = go.Figure(data=[go.Bar(
    x=['Item A', 'Item B', 'Item C',
       'Item D', 'Item E'],
    y=[20, 14, 23, 25, 22],
    marker_color=colors
)])

fig.update_layout(title_text='Item menos usado')
```



1.3 Pie Charts

[26]:
```
labels = ['Oxigênio','Hidrogênio','Gás Carbônico','Nitrogênio']
values = [4500, 2500, 1053, 500]

fig = go.Figure(data=[go.Pie(labels=labels, values=values)])
fig.show()
```

1.3.1 Estilizando gráficos de pizza

[27]:
```
import plotly.graph_objects as go
colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']

fig = go.Figure(data=[go.Pie(labels=['Oxigênio','Hidrogênio','Gás␣
 ↪Carbônico','Nitrogênio'],
                             values=[4500,2500,1053,500])])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000',␣
 ↪width=2)))
fig.show()
```

1.3.2 Destacando elementos individuais do gráfico

[28]:
```
labels = ['Oxigênio','Hidrogênio','Gás Carbônico','Nitrogênio']
values = [4500, 2500, 1053, 500]

# pull is given as a fraction of the pie radius
fig = go.Figure(data=[go.Pie(labels=labels, values=values, pull=[0, 0, 0.2,␣
↪0])])
fig.show()
```

1.4 Dúvidas

Acesse: https://plotly.com/python/basic-charts/

3. Gráficos estatísticos

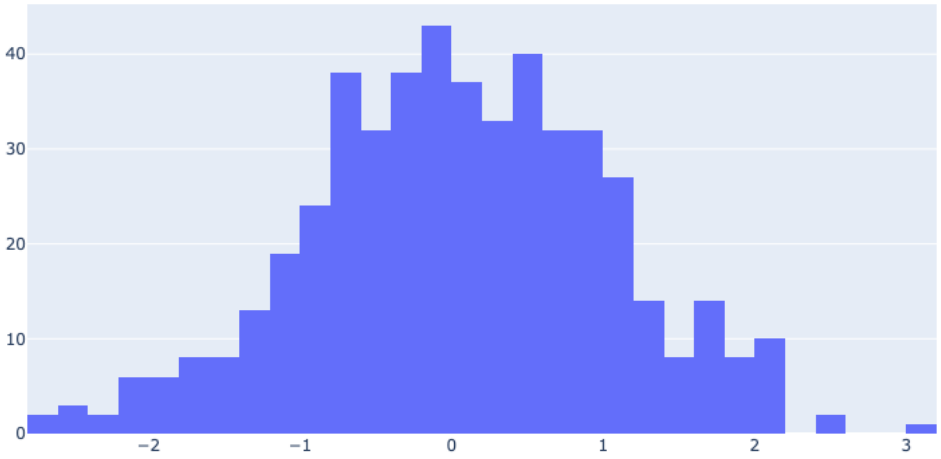
September 11, 2023

1 Gráficos estatísticos

Nessa seção aprenderemos como criar gráficos estatísticos no Plotly. Referência: https://plotly.com/python/statistical-charts/

1.1 Histogramas

[5]:
```
import plotly.graph_objects as go

import numpy as np
np.random.seed(1)

x = np.random.randn(500)

fig = go.Figure(data=[go.Histogram(x=x)])
fig.update_layout(height=500)
fig.show()
```

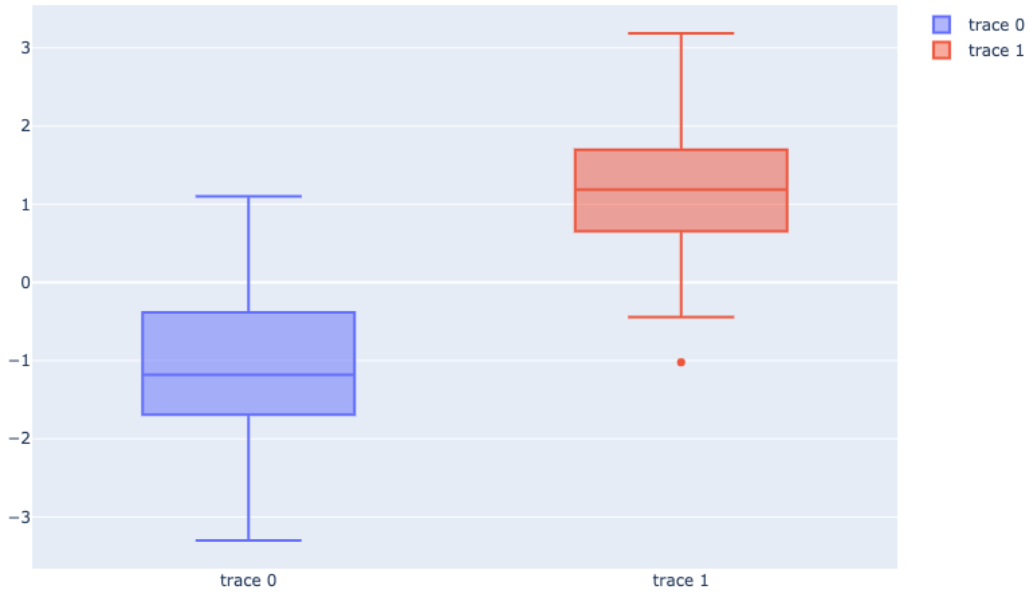


1.1.1 Sobreposição de histogramas

[ ]:
```
import plotly.graph_objects as go

import numpy as np

x0 = np.random.randn(500)
# Adicione 1 para deslocar a média da distribuição x0
x1 = np.random.randn(500) + 1

fig = go.Figure()
fig.add_trace(go.Histogram(x=x0))
fig.add_trace(go.Histogram(x=x1))

# Sobreposição
fig.update_layout(barmode='overlay')

# Reduz a opacidade para que possamos ver ambos histogramas
fig.update_traces(opacity=0.75)
fig.show()
```

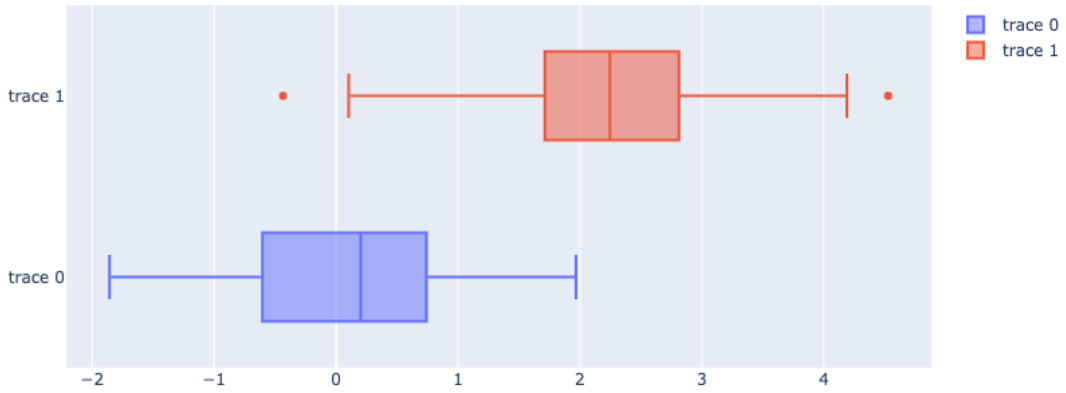
1.2 Boxplots

[7]:
```
import plotly.graph_objects as go
import numpy as np
np.random.seed(1)

y0 = np.random.randn(50) - 1
y1 = np.random.randn(50) + 1

fig = go.Figure()
fig.add_trace(go.Box(y=y0))
fig.add_trace(go.Box(y=y1))

fig.update_layout(height=600)
fig.show()
```



1.2.1 Plotando horizontalmente

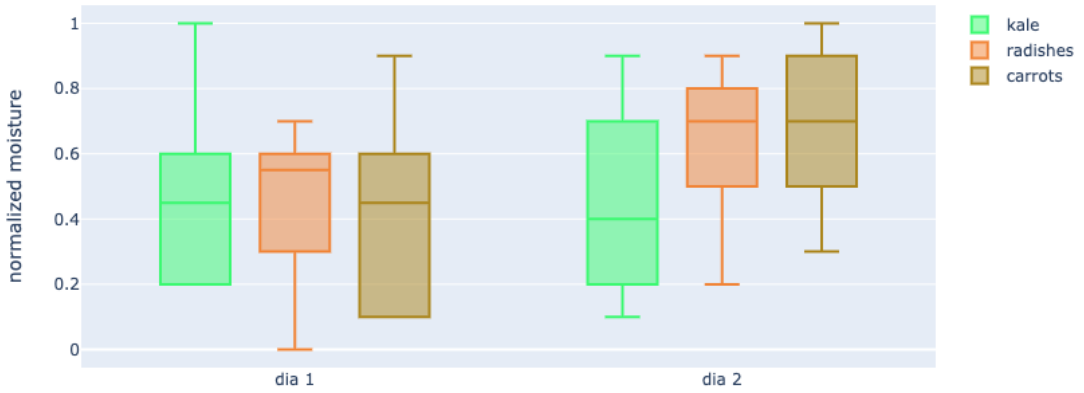
[8]:
```
x0 = np.random.randn(50)
x1 = np.random.randn(50) + 2 # shift mean

fig = go.Figure()

# Para plotar horizontalmente, basta substituir x por y
fig.add_trace(go.Box(x=x0))
fig.add_trace(go.Box(x=x1))

fig.show()
```



1.2.2 Agrupando boxes

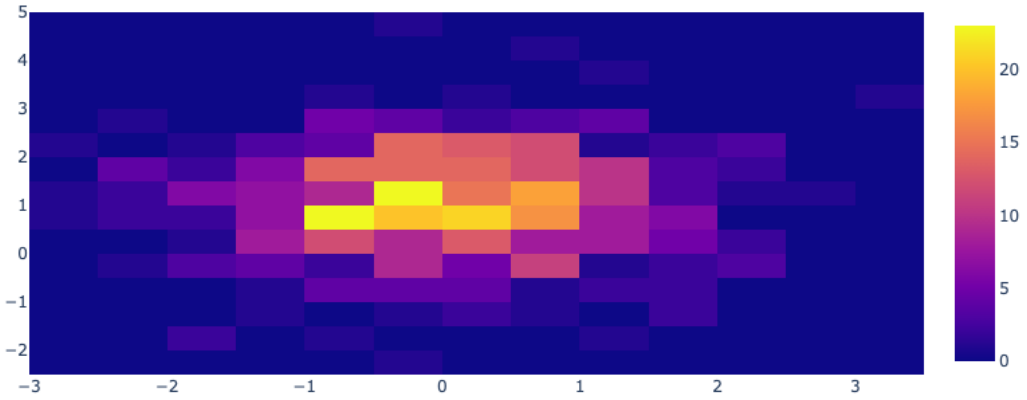
[14]:
```
import plotly.graph_objects as go

x = ['dia 1', 'dia 1', 'dia 1', 'dia 1', 'dia 1', 'dia 1',
     'dia 2', 'dia 2', 'dia 2', 'dia 2', 'dia 2', 'dia 2']

fig = go.Figure()

fig.add_trace(go.Box(
    y=[0.2, 0.2, 0.6, 1.0, 0.5, 0.4, 0.2, 0.7, 0.9, 0.1, 0.5, 0.3],
    x=x,
    name='couve',
    marker_color='#3AF970'
))
fig.add_trace(go.Box(
    y=[0.6, 0.7, 0.3, 0.6, 0.0, 0.5, 0.7, 0.9, 0.5, 0.8, 0.7, 0.2],
    x=x,
    name='rabanete',
    marker_color='#F18436'
))
fig.add_trace(go.Box(
    y=[0.1, 0.3, 0.1, 0.9, 0.6, 0.6, 0.9, 1.0, 0.3, 0.6, 0.8, 0.5],
    x=x,
    name='cenoura',
    marker_color='#AB851B'
))
```

```
fig.update_layout(
    yaxis_title='normalized moisture',
    boxmode='group' # Agrupe caixas de diferentes traces para cada valor de x
)
fig.show()
```

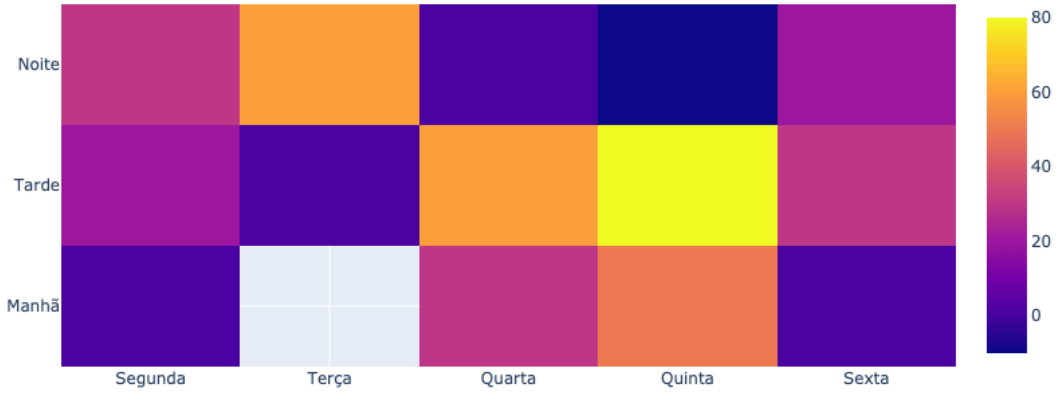


1.2.3 Histogram2d

[15]:
```
import plotly.graph_objects as go

import numpy as np
np.random.seed(1)

x = np.random.randn(500)
y = np.random.randn(500)+1

fig = go.Figure(go.Histogram2d(
        x=x,
        y=y
    ))
fig.show()
```



[16]:
```
fig = go.Figure(data=go.Heatmap(
                    z=[[1, None, 30, 50, 1], [20, 1, 60, 80, 30], [30, 60, 1,␣
 ↪-10, 20]],
                    x=['Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta'],
                    y=['Manhã', 'Tarde', 'Noite'],
                    hoverongaps = False))
fig.show()
```



[ ]:

4. Outros Exemplos

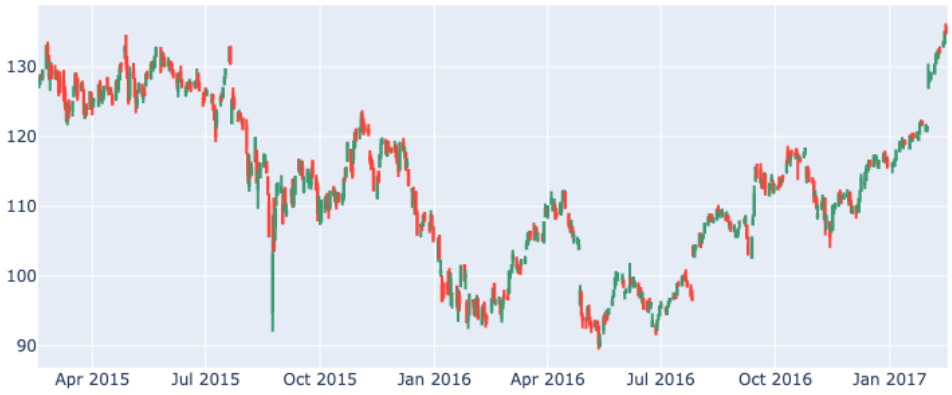
September 11, 2023

1 Gráficos alternativos

Nesta seção são apresentados alguns gráficos não convencionais. Lembrando que a biblioteca do Plotly é muito extensa e sempre se recomenda pesquisar na documentação da mesma.

1.1 Candlestick

[5]:
```
import plotly.graph_objects as go
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/
 ↪finance-charts-apple.csv')

fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['AAPL.Open'], high=df['AAPL.High'],
                low=df['AAPL.Low'], close=df['AAPL.Close'])
                     ])

fig.update_layout(xaxis_rangeslider_visible=False)
fig.show()
```

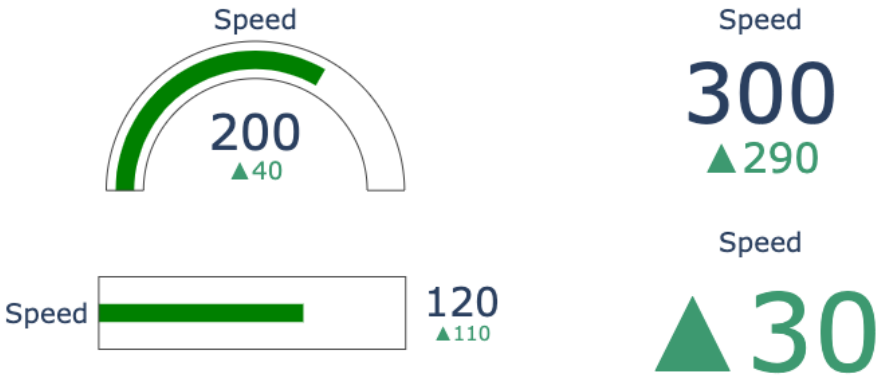


Por padrão, tal objeto vem com um `RangeSlider` embutido, permitindo o controle do eixo x.

[11]:
```
import plotly.graph_objects as go
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/
 ↪finance-charts-apple.csv')

fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['AAPL.Open'], high=df['AAPL.High'],
                low=df['AAPL.Low'], close=df['AAPL.Close'])
                     ])

fig.update_layout(height=700)
fig.show()
```

1.2 Indicator

[17]:
```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Indicator(
    value = 200,
    delta = {'reference': 160},
    gauge = {
        'axis': {'visible': False}},
    domain = {'row': 0, 'column': 0}))

fig.add_trace(go.Indicator(
    value = 120,
    gauge = {
        'shape': "bullet",
        'axis' : {'visible': False}},
    domain = {'x': [0.05, 0.5], 'y': [0.15, 0.35]}))

fig.add_trace(go.Indicator(
    mode = "number+delta",
    value = 300,
    domain = {'row': 0, 'column': 1}))

fig.add_trace(go.Indicator(
    mode = "delta",
    value = 40,
    domain = {'row': 1, 'column': 1}))

fig.update_layout(
    grid = {'rows': 2, 'columns': 2, 'pattern': "independent"},
    template = {'data' : {'indicator': [{
        'title': {'text': "Speed"},
        'mode' : "number+delta+gauge",
         'delta' : {'reference': 10}}]
                          }})
```

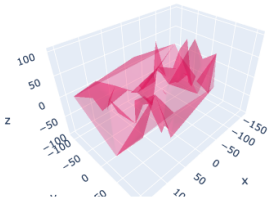


[ ]:
```
fig = go.Figure()

fig.add_trace(go.Indicator(
    value = 200,
    mode="gauge+number",
    delta = {'reference': 160},
    gauge = {
        'axis': {'visible': False}},
    title="Speed"))

fig.show()
```

1.3 Mesh3D

[2]:
```
import plotly.graph_objects as go
import numpy as np
np.random.seed(1)

N = 70

fig = go.Figure(data=[go.Mesh3d(x=(70*np.random.randn(N)),
                   y=(55*np.random.randn(N)),
                   z=(40*np.random.randn(N)),
                   opacity=0.5,
                   color='rgba(244,22,100,0.6)'
                  )])
```

```
fig.update_layout(
    width=700)

fig.show()
```



[ ]:

5. Plotly Express

September 11, 2023

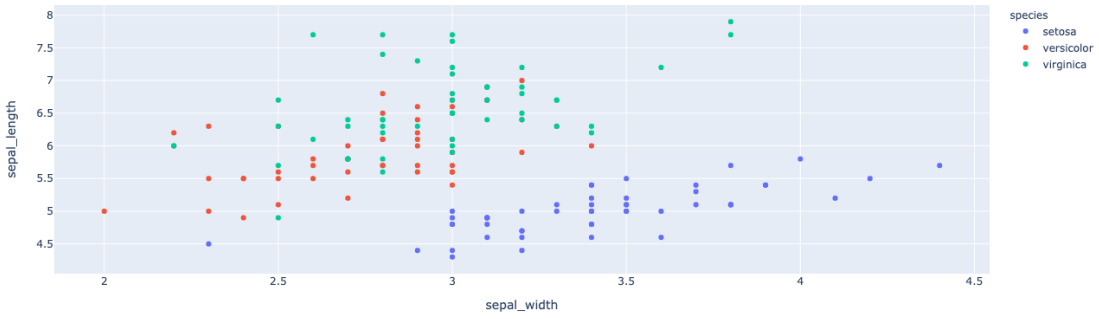
1 Plotly Express

O Plotly Express **(px)** é um módulo adicional da biblioteca Plotly capaz de criar os principais gráficos estudados até aqui com muito mais facilidade. Todas as funções do **(px)** se utilizam internamente do `Graphic Object` e retornam uma `Figure`.

Ao se instanciar o módulo, obtemos acesso instantâneo a uma grande variedade de funções de plot e dados para teste (acessíveis através do `px.data`). Praticamente todos os artigos presentes na documentação da biblioteca apresentam formas de reproduzir os gráficos utilizando o `px`.

1.1 Scatter Plots, Area, Line e Bar

[3]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```



[ ]:
```
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()
```

[ ]:
```
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group")
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group",␣
 ↪facet_row="time", facet_col="day",
       category_orders={"day": ["Thur", "Fri", "Sat", "Sun"], "time": ["Lunch",␣
 ↪"Dinner"]})
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter_matrix(df, dimensions=["sepal_width", "sepal_length",␣
 ↪"petal_width", "petal_length"], color="species")
fig.show()
```

[ ]:
```
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", log_x=True, size_max=60)
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year",␣
 ↪animation_group="country",
           size="pop", color="continent", hover_name="country",␣
 ↪facet_col="continent",
           log_x=True, size_max=45, range_x=[100,100000], range_y=[25,90])
fig.show()
```

1.2 Gráficos alternativos

[ ]:
```
import plotly.express as px

df = px.data.gapminder().query("year == 2007")
fig = px.sunburst(df, path=['continent', 'country'], values='pop',
                  color='lifeExp', hover_data=['iso_alpha'])
fig.show()
```

1.3 Gráficos estatísticos

[ ]:
```
import plotly.express as px
df = px.data.tips()
fig = px.histogram(df, x="total_bill", y="tip", color="sex", marginal="rug",␣
 ↪hover_data=df.columns)
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x="day", y="total_bill", color="smoker", notched=True)
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.iris()
fig = px.density_heatmap(df, x="sepal_width", y="sepal_length",␣
 ↪marginal_x="rug", marginal_y="histogram")
fig.show()
```

1.4 Mapas

[ ]:
```
import plotly.express as px
df = px.data.carshare()
fig = px.scatter_mapbox(df, lat="centroid_lat", lon="centroid_lon",␣
 ↪color="peak_hour", size="car_hours",
                  color_continuous_scale=px.colors.cyclical.IceFire,␣
 ↪size_max=15, zoom=10,
                  mapbox_style="carto-positron")
fig.show()
```

[ ]:
```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_mapbox(df, geojson=geojson, color="Bergeron",
                           locations="district", featureidkey="properties.
 ↪district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           mapbox_style="carto-positron", zoom=9)
fig.show()
```

[ ]:
```
import plotly.express as px
df = px.data.gapminder()
fig = px.choropleth(df, locations="iso_alpha", color="lifeExp",␣
 ↪hover_name="country", animation_frame="year", range_color=[20,80])
```

```
fig.update_layout(height=800)
fig.show()
```

1.5 Polar bar charts

[ ]:
```
import plotly.express as px
df = px.data.wind()
fig = px.bar_polar(df, r="frequency", theta="direction", color="strength",␣
↪template="plotly_dark",
            color_discrete_sequence= px.colors.sequential.Plasma_r)
fig.show()
```

[ ]:
```

```