Título: Explorando a Biblioteca Plotly em Python para Visualização de Dados em Ciência de Dados

Derik B. Pimentel, Otávio Jose dos Santos, Sergio Assunção Monteiro

Ciência da Computação - Centro Universitário Carioca (UniCarioca)

Resumo

Este artigo tem como objetivo apresentar e explorar a biblioteca Plotly em Python, destacando suas funcionalidades e aplicações dentro do campo da Ciência de Dados. Serão fornecidos exemplos práticos de uso da biblioteca para criar visualizações de dados interativas e informativas. Além disso, serão discutidas as situações mais adequadas para utilizar cada tipo de gráfico disponível na Plotly.

Palavras chave: plotly; python; visualização de dados; ciência de dados; gráficos interativos; plotly para análise de dados.

Abstract

This article aims to present and explore the Plotly library in Python, highlighting its features and applications within the field of Data Science. Practical examples of using the library to create interactive and informative data visualizations will be provided. In addition, the most suitable situations for using each type of graph available in Plotly will be discussed.

Keywords: plotly; python; data visualization; data science; interactive graphics; plotly for data analysis.

Introdução

A visualização de dados desempenha um papel fundamental na análise e interpretação de dados, bem como na comunicação eficaz de informações relevantes. A biblioteca Plotly em Python se destaca como uma ferramenta poderosa e versátil para criar gráficos interativos e atrativos, oferecendo uma ampla gama de possibilidades para explorar e apresentar dados no campo da Ciência de Dados. Este artigo tem como objetivo explorar as funcionalidades essenciais da Plotly, proporcionando uma introdução clara e concisa para aqueles que desejam começar a utilizar essa ferramenta impressionante.

Ao longo deste artigo, vamos analisar de perto as principais características da Plotly e discutir sua aplicação em diferentes cenários. Além disso, apresentaremos exemplos de gráficos de fácil compreensão, destacando quando cada tipo é apropriado para uso. Ao compreender a versatilidade da biblioteca.

1. Fundamentos da Biblioteca Plotly

Nesta seção, vamos apresentar os conceitos básicos da biblioteca Plotly, incluindo:

- Instalação e configuração.

O Plotly não vem embutido com Python, e para instalá-lo no Python é fácil. Ele está disponível no pip, então, para instalá-lo, basta digitar 'pip install plotly' e executar. Feito isso, ele já estará disponível para uso.

- Estrutura de dados fundamentais (Figuras e Traços).

Existem três módulos principais no Plotly. Eles são:

- plotly.plotly
- plotly.graph.objects
- plotly.tools

O módulo 'plotly.plotly' atua como a interface entre a máquina local e o Plotly. Ele contém funções que requerem uma resposta do servidor do Plotly.

O módulo 'plotly.graph_objects' contém os objetos (Figura, layout, dados e a definição dos gráficos como gráfico de dispersão, gráfico de linha) que são responsáveis por criar os gráficos. A Figura pode ser representada como dict ou como instâncias de 'plotly.graph_objects.Figure' e são serializados como JSON antes de serem transmitidos para 'plotly.js'. Considere o exemplo das figuras 1 e 2.

Nota: o módulo 'plotly.express' pode criar a Figura inteira de uma vez. Ele usa o 'graph_objects' internamente e retorna a instância 'graph_objects.Figure'.

```
[ ]  import plotly.express as px


     # Criando a instância da Figura
     fig = px.line(x=[1,2, 3], y=[1, 2, 3])

     # imprimindo a instância da figura
     print(fig)
```

Figura 1

```
Figure({
    'data': [{'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array([1, 2, 3]),
              'xaxis': 'x',
              'y': array([1, 2, 3]),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'x'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'y'}}}
})
```

Figura 2

As figuras são representadas como árvores onde o nó raiz tem três atributos da camada superior (dados, layout e quadros) e os nós nomeados chamados 'atributos'. Considere o exemplo da figura 2, ‘layout.legend’ é um dicionário aninhado onde a legenda é a chave dentro do dicionário cujo valor também é um dicionário.

O módulo ‘plotly.tools’ contém várias ferramentas nas formas das funções que podem aprimorar a experiência do Plotly.

2. Tipos de Gráficos na Plotly

Na biblioteca Plotly em Python, podemos criar uma ampla variedade de tipos de gráficos para visualizar os dados. Cada tipo de gráfico tem suas próprias características e é mais apropriado para diferentes tipos de dados e situações. Abaixo, alguns dos tipos de gráficos mais comuns na Plotly (para o uso em ciência de dados) e alguns exemplos de quando é apropriado usá-los:

- Gráficos de dispersão (Scatter Plots).

  Um gráfico de dispersão é um conjunto de pontos pontilhados para representar partes individuais de dados nos eixos horizontal e vertical. Um gráfico no qual os valores de duas variáveis são plotados ao longo do eixo X e do eixo Y, o padrão dos pontos resultantes revela uma correlação entre eles.

  Para criar um gráfico de dispersão simples, utiliza-se o código mostrado na figura 3:

```python
import plotly.express as px

# usando o conjunto de dados da íris
df = px.data.iris()

# traçando o gráfico de dispersão
fig = px.scatter(df, x="species", y="petal_width")

# mostrando o gráfico
fig.show()
```

Figura 3

A variável ‘fig’ armazena o objeto do gráfico, utilizando a função ‘px.scatter()’. Dentro dessa função são passados os argumentos para geração do gráfico, como o data frame e os eixos X e Y.

Para exibir um gráfico no Plotly, é necessário passar todos os dados na forma de uma lista. Neste exemplo, é utilizado um conjunto de dados pré-existente da íris. Onde a variável df recebe os dados, seguindo a boa prática de passar os dados para dentro de uma variável.

Por fim, a função ‘fig.show()’ exibirá o gráfico, podendo observar o resultado na figura 4:

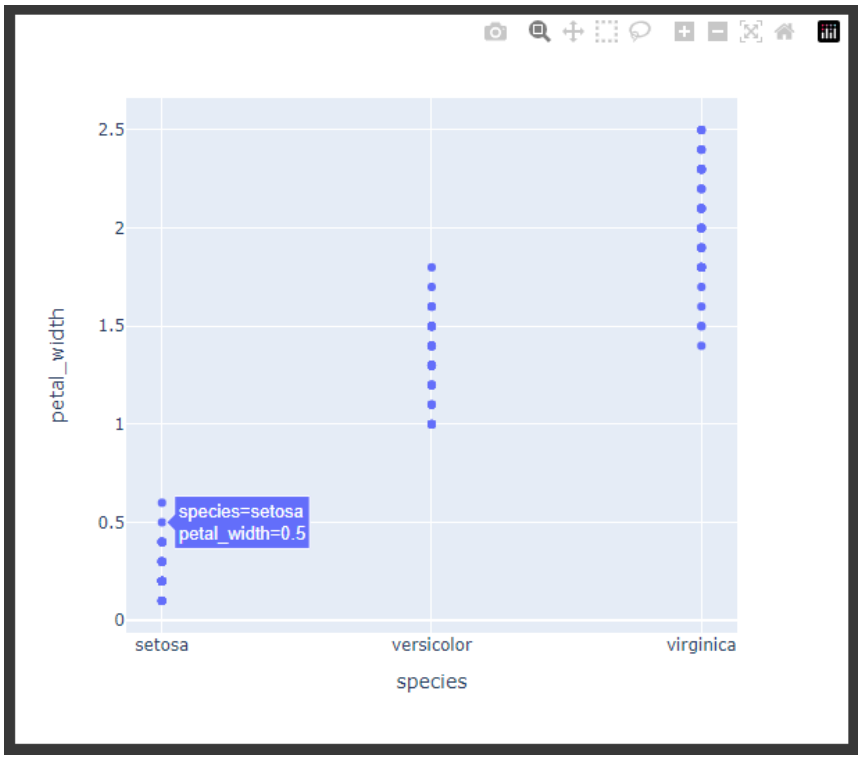

Figura 4

Alguns dos seus principais usos são:

- Análise de Correlação: Um gráfico de dispersão é uma maneira eficaz de visualizar a relação entre duas variáveis quantitativas. Ele pode ajudar a determinar se existe uma correlação positiva, negativa ou nenhuma correlação entre as variáveis. Por exemplo, você pode usar um gráfico de dispersão para visualizar a relação entre a idade e o salário de um grupo de pessoas.
- Identificação de Outliers: Os gráficos de dispersão podem destacar valores discrepantes (outliers) em um conjunto de dados. Valores que estão muito afastados da tendência geral podem ser identificados visualmente em um gráfico de dispersão.
- Visualização de Distribuição: Além de mostrar a relação entre duas variáveis, um gráfico de dispersão também pode ser usado para visualizar a distribuição dos dados. Isso pode ser útil para identificar agrupamentos de dados ou padrões em uma nuvem de pontos.
- Modelagem e Previsão: Gráficos de dispersão são frequentemente usados em análises de regressão para ajudar a escolher um modelo adequado. Você pode ajustar uma linha de regressão aos dados no gráfico de dispersão para criar um modelo preditivo.

- Gráficos de barras (Bar Charts).

  Os gráficos de barras são uma representação visual de dados que utiliza barras retangulares para mostrar a comparação entre diferentes categorias de dados. Uma das variáveis representa as categorias, enquanto a outra representa a medida associada a essas categorias. As barras podem ser dispostas horizontal ou verticalmente, dependendo da preferência ou da necessidade de apresentação dos dados. Características Principais:

  - Eixo Horizontal (X): Representa as categorias ou rótulos dos dados.
  - Eixo Vertical (Y): Representa a medida associada às categorias.
  - Barras: As barras retangulares são desenhadas a partir do eixo horizontal (ou vertical) para representar o valor associado a cada categoria.
  - Altura ou Comprimento das Barras: Indica a magnitude da medida para cada categoria.
  - Espaçamento: As barras geralmente são separadas por espaços iguais para evitar a confusão entre as categorias.

Para criar um gráfico de barras usando a biblioteca Plotly em Python, primeiro, você precisa instalá-la se ainda não tiver feito isso. Aqui está um exemplo de código para criar um gráfico de barras verticais simples usando a biblioteca Plotly:

```
import plotly.express as px

# Dados de exemplo
categorias = ['A', 'B', 'C', 'D', 'E']
valores = [4, 7, 1, 3, 9]

# Criando o gráfico de barras
fig = px.bar(x=categorias, y=valores, labels={'x':'Categorias', 'y':'Valores'},
             title='Exemplo de Gráfico de Barras com Plotly')

# Exibindo o gráfico
fig.show()
```

Figura 5

Neste exemplo, o eixo horizontal (X) representa as categorias A, B, C, D e E, enquanto o eixo vertical (Y) representa os valores correspondentes 4, 7, 1, 3 e 9. `categorias` e `valores` são listas que contêm os rótulos das categorias e os valores associados, respectivamente.

px.bar( )`: Cria um gráfico de barras usando Plotly Express, onde `x` recebe as categorias e `y` recebe os valores. É possível personalizar o gráfico de acordo com suas necessidades, modificando rótulos, cores, tamanhos e outros atributos usando os parâmetros adequados.

`labels={‘x’: ‘Categorías’, ‘y’: ‘Valores’}`: Define os rótulos para os eixos X e Y.

`title=’Exemplo de Gráfico de Barras com Plotly’`: Define o título do gráfico.

`fig.show( )`: Exibe o gráfico.

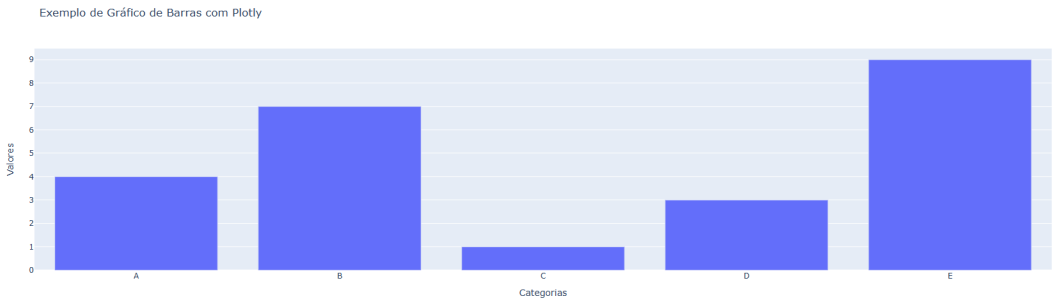

Figura 6

Alguns dos seus principais usos são:

- Comparação de Categorias: O gráfico de barras do Plotly permite comparar categorias de dados de maneira eficaz. Você pode criar gráficos de barras verticais ou horizontais para representar as diferenças entre categorias. Isso é

útil para comparar dados, como vendas de produtos, resultados de pesquisas ou desempenho de diferentes equipes ou departamentos.

- ◆ Visualização de Dados Categóricos: O Plotly é particularmente útil quando se trata de dados categóricos, onde as categorias são representadas no eixo x (horizontal) e as alturas das barras correspondem às quantidades ou valores associados a essas categorias. Isso é comum em visualizações de dados de pesquisa e análises de mercado.
- ◆ Gráficos de Barras Agrupadas ou Empilhadas: O Plotly oferece suporte para gráficos de barras agrupadas ou empilhadas, o que permite comparar múltiplas variáveis ou subcategorias em uma única visualização. Você pode usar isso para entender as contribuições relativas de diferentes subcategorias para o total.

- Gráficos de pizza (Pie Charts).

Este tipo de gráfico divide um todo em partes proporcionais para mostrar a contribuição relativa de cada categoria.

```
import pandas as pd
import plotly.express as px

# Criando um DataFrame com Pandas
dados = {'Categoria': ['Vermelho', 'Verde', 'Roxo', 'Azul'],
         'Contagem': [656, 789, 300, 946]}
df = pd.DataFrame(dados)

# Criando o gráfico de pizza
fig = px.pie(df, values='Contagem', names='Categoria',
             title='Exemplo de gráfico de Pizza')

# Exibindo o gráfico
fig.show()
```

Figura 7

Neste exemplo mostrado na figura 7, quatro categorias foram estabelecidas e associadas a valores distintos, um para cada categoria. Esse cenário permite o cálculo das proporções individuais de cada categoria em relação ao conjunto total, considerando-as como componentes de um todo unificado. Esse procedimento contribui para uma representação visual clara das contribuições relativas de cada categoria no contexto geral. Como pode ser visto a seguir no resultado do gráfico da figura 8:

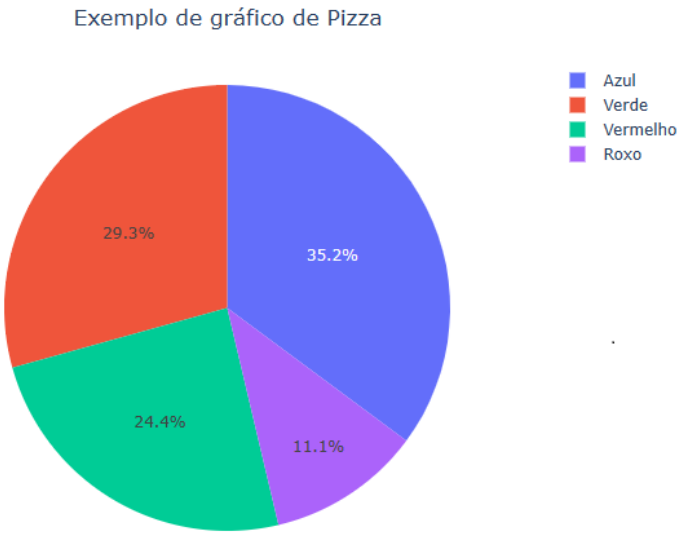

Figura 8

Alguns dos seus principais usos são:

- Representação da Composição Percentual: Gráficos de pizza são eficazes para mostrar a contribuição relativa de diferentes categorias para um todo. Cada fatia da pizza representa uma categoria e seu tamanho é proporcional à porcentagem que essa categoria representa em relação ao todo.
- Visualização de Partes de um Todo: Eles são úteis quando se deseja mostrar como uma quantidade se divide em partes, como a alocação de um orçamento para diferentes despesas, a composição de uma carteira de investimentos ou a distribuição de votos em uma eleição.
- Apresentações Simples: Em apresentações ou relatórios onde a simplicidade visual é fundamental, um gráfico de pizza pode ser útil para ilustrar conceitos complexos de uma maneira fácil de entender.

- Gráficos de linha (Line Charts).

  Os gráficos de linhas são usados para mostrar tendências ao longo do tempo ou em uma sequência de pontos de dados.

```python
import pandas as pd
import plotly.express as px

# Criando um DataFrame com Pandas
dados = {'dias': ['Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sab', 'Dom'],
         'ligacoes': [68, 53, 50, 41, 70, 12, 5]}
df = pd.DataFrame(dados)

# Criando o gráfico de linha
fig = px.line(df, x='dias', y='ligacoes', title='Ligações da Semana' )

# Mudando o nome de exibição dos eixos X e Y
fig.update_layout(
    xaxis_title='Dias da semana',
    yaxis_title='Quantidade de ligações')

# Exibindo o gráfico
fig.show()
```

Figura 9

Neste exemplo, apresentamos a construção de um gráfico que incorpora sete conjuntos distintos de dados, correspondentes a diferentes dias (eixo X), cada um associado a seus respectivos valores, representando ligações (eixo Y). Essa representação possibilita a criação de uma linha conectando sete pontos diversos, oferecendo uma visualização da variação nos valores ao longo do tempo. A seguir mostramos o gráfico gerado por este código na figura 10:

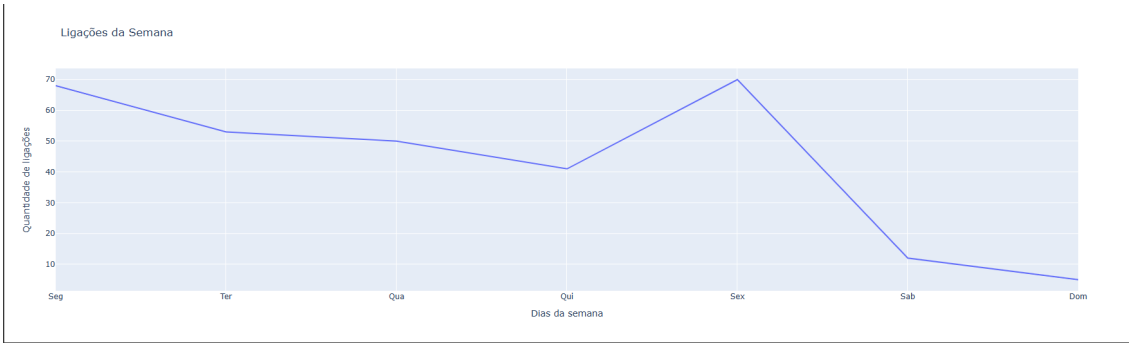

Figura 10

Alguns dos seus principais usos são:

- Visualização de Tendências Temporais: Gráficos de linhas são especialmente eficazes para mostrar tendências ao longo do tempo. Eles são comumente usados para representar séries temporais, como dados financeiros diários, métricas de desempenho ao longo dos meses ou anos, ou qualquer conjunto de dados onde a dimensão temporal seja relevante.
- Análise de Séries Temporais: Os gráficos de linhas são cruciais na análise de séries temporais, permitindo a detecção de padrões sazonais, ciclos e

tendências em dados ao longo do tempo. Eles podem ser usados em previsões e análises de séries temporais.

- Monitoramento de Desempenho: Empresas frequentemente usam gráficos de linhas para monitorar o desempenho de métricas-chave, como receita, lucro, tráfego de site ou engajamento de mídia social ao longo do tempo. Isso ajuda a identificar tendências de crescimento ou declínio.
- Visualização de Dados Contínuos: Gráficos de linhas também são adequados para representar dados contínuos, como temperatura ao longo de um dia ou níveis de poluição ao longo de uma rota.

- Gráficos de caixa (Box Plots).

  O gráfico de caixa mostra a mediana, quartis, valores mínimos e máximos de um conjunto de dados de maneira compacta e fácil de entender.

```
import plotly.tools as tls
import plotly.graph_objects as go

# Dados de exemplo
dados1 = [1, 2, 2, 3, 3, 3, 4, 4, 5]
dados2 = [2, 3, 3, 4, 4, 5, 5, 6, 6]

# Criar os traces (caixas) para os dados
trace1 = go.Box(y=dados1, name='Dados 1', boxpoints='all', jitter=0.3, pointpos=-1.8)
trace2 = go.Box(y=dados2, name='Dados 2', boxpoints='all', jitter=0.3, pointpos=-1.8)

# Layout do gráfico
layout = go.Layout(title='Gráfico de Caixa (Box Plot)',
                   yaxis=dict(title='Valores'),
                   boxmode='group')

# Criar a figura
fig = go.Figure(data=[trace1, trace2], layout=layout)

# Exibir o gráfico
fig.show()
```

Figura 11

No exemplo da figura 11, estão definidos dois conjuntos de dados de exemplo que serão utilizados para criar o gráfico de caixa. São criados dois traces (caixas) utilizando a função `go.Box`. Cada trace representa um conjunto de dados e possui algumas opções de configuração, como a posição dos pontos (`boxpoints`), o deslocamento dos pontos (`jitter`), e a posição dos pontos em relação às caixas (`pointpos`).

Também configura-se o layout do gráfico, incluindo o título (`title`), o título do eixo y (`yaxis`), e o modo de agrupamento das caixas (`boxmode`). Para finalizar, utilizando a função `go.Figure`, cria a figura do gráfico combinando os traces

(caixas) e o layout configurado, e executa a função `fig.show()` para exibir o gráfico em uma janela separada. Podemos ver o resultado na figura 12.

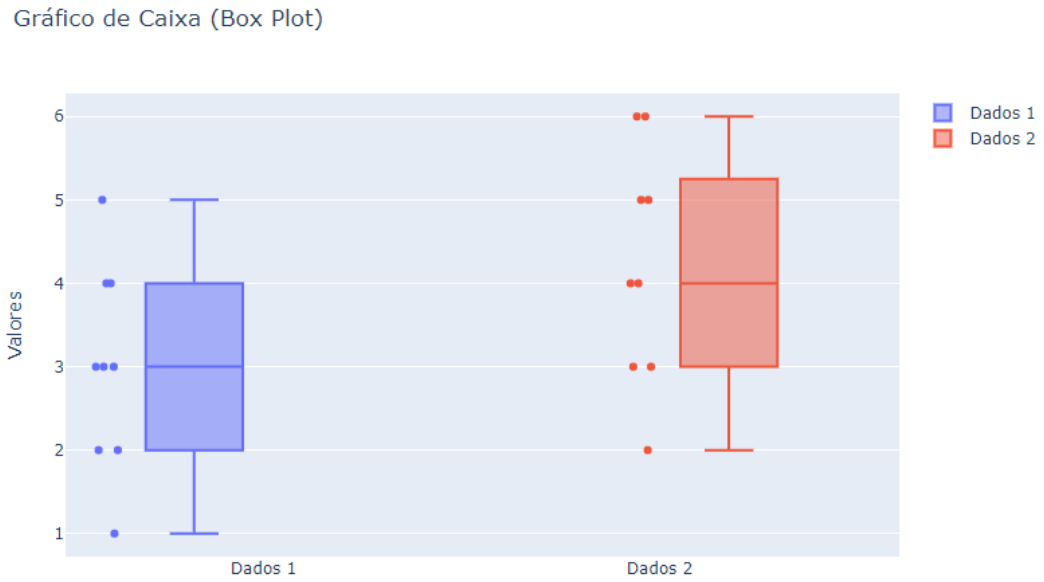

Figura 12

Alguns dos seus principais usos são:

- Resumo Estatístico: Os gráficos de caixa fornecem um resumo visual das estatísticas importantes de um conjunto de dados, como a mediana, os quartis (Q1 e Q3), a amplitude interquartil (IQR) e possíveis valores atípicos (outliers).
- Identificação de Outliers: Valores extremos ou outliers são facilmente identificados como pontos fora das "caixas" principais e geralmente são representados como pontos individuais nos gráficos de caixa.
- Análise de Dados Multivariados: Quando vários gráficos de caixa são empilhados ou agrupados, eles podem ser usados para visualizar a distribuição de diferentes variáveis em relação a uma variável categórica.

- Mapas de calor (Heatmaps).

  Exibe a relação entre duas variáveis categóricas, usando cores para representar a intensidade.

```python
import pandas as pd
import plotly.express as px
import numpy as np

# Criando um DataFrame com valores aleatórios utilizando Pandas e numpy
dados = pd.DataFrame({
    'Variavel1': np.random.randn(10),
    'Variavel2': np.random.randn(10),
})

# Criando o mapa de calor
fig = px.density_heatmap(dados,
                         x='Variavel1',
                         y='Variavel2',
                         title='Exemplo de Mapa de Calor')

# Exibindo o gráfico
fig.show()
```

Figura 13

Aqui foram colocadas variáveis aleatórias com a ajuda da biblioteca numpy em um dataframe. Depois utilizamos as variáveis nos eixos X e Y para gerar o gráfico da figura 14:

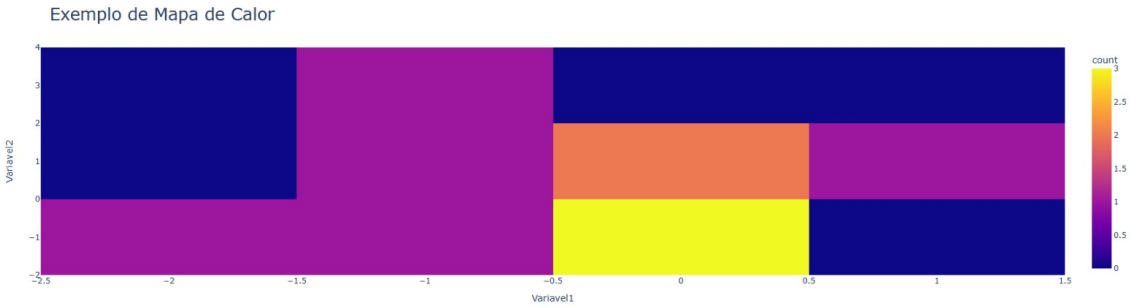

Figura 14

Alguns dos seus principais usos são:

- Simulações e Modelagem Numérica: Eles são usados em simulações computacionais para representar fenômenos tridimensionais complexos, como simulações climáticas, dinâmica de fluidos ou simulações de engenharia.
- Visualização de Dados Geoespaciais: Quando usado com dados geoespaciais, gráficos de superfície podem representar informações tridimensionais sobre um terreno ou uma região geográfica.
- Análise de Mercado Financeiro: Em finanças, eles podem ser usados para visualizar a superfície de preços de opções financeiras em relação a diferentes variáveis, como preço da ação e tempo até o vencimento.

- Gráfico de Superfície (3D Surface Plot).

Usado para visualizar funções tridimensionais em um espaço 3D.

```
import pandas as pd
import plotly.graph_objects as go
import numpy as np

# Criando os dados.
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.random.randn(100, 100)  # Valores sendo gerados de forma aleatória

df = pd.DataFrame({
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
})

# Criando o gráfico
fig = go.Figure(data=[go.Surface(x=df['X'].values.reshape(100, 100),
                                 y=df['Y'].values.reshape(100, 100),
                                 z=df['Z'].values.reshape(100, 100))])

# Atualizando o layout
fig.update_layout(scene=dict(
                    xaxis_title='Eixo X',
                    yaxis_title='Eixo Y',
                    zaxis_title='Eixo Z'),
                    title='Gráfico de Superfície 3D')

# Exibindo o gráfico
fig.show()
```

Figura 15

Assim como já vimos em exemplos anteriores, neste exemplo, primeiro geramos os dados utilizando a biblioteca numpy e alocamos em um dataframe com ajuda da biblioteca pandas. Assim como todos os outros gráficos, podemos gerar dados aleatoriamente ou utilizar dados já existentes para gerar os gráficos. Após isso utilizamos a função go.Figure do modulo graph_objects para gerar o gráfico de superfície (go.Surface()). Aqui podemos visualizar uma pequena diferença na escrita do código quando comparado com o modulo plotly.express. A seguir vemos o gráfico gerado na figura 16.

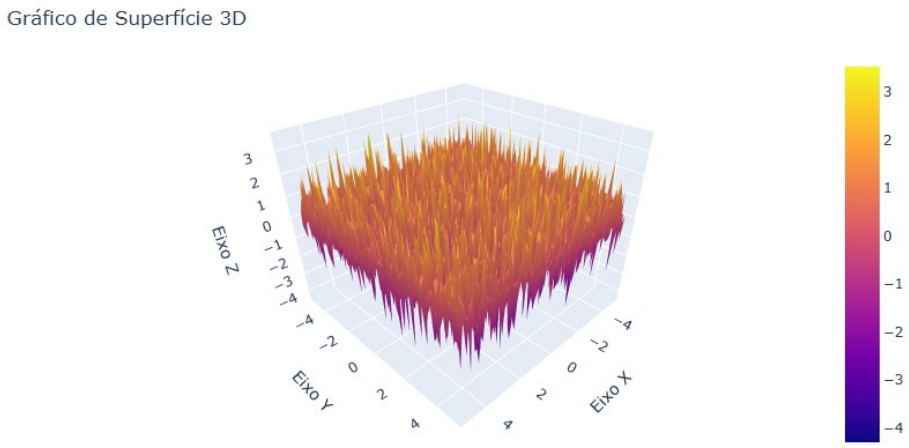

Figura 16

Alguns dos seus principais usos são:

- Visualização de Funções Matemáticas: Gráficos de superfície são comumente usados para representar funções matemáticas tridimensionais, como superfícies de resposta em otimização, curvas de nível, ou qualquer relação complexa entre três variáveis.
- Mapeamento Topográfico: Eles são úteis para representar mapas topográficos, mostrando elevações e relevo de uma região geográfica. Isso é valioso em geografia, geologia e planejamento urbano.
- Modelagem de Terrenos e Paisagens: Em jogos de vídeo, simulações geográficas ou modelagem de paisagens, gráficos de superfície são usados para criar terrenos realistas.
- Análise de Dados Experimentais: Gráficos de superfície podem ser usados para visualizar resultados de experimentos em que três variáveis estão sendo estudadas simultaneamente. Isso pode ser útil em física, química, biologia e outras ciências.

A escolha do tipo de gráfico certo depende da natureza dos dados e da mensagem que se deseja transmitir. É necessário sempre levar em consideração a clareza e a interpretabilidade dos gráficos ao escolher o tipo apropriado para análise. A melhor prática é sempre experimentar os diferentes tipos de gráficos para encontrar o que melhor se adapta aos dados e ao público-alvo.

3. Casos de Uso em Ciência de Dados

Nesta seção, vamos analisar exemplos de casos de uso da Plotly em projetos de Ciência de Dados. Para cada caso, apresentaremos códigos de exemplo e os tipos de gráficos mais adequados:

- Análise exploratória de dados.

  A Plotly é muito útil para criar visualizações interativas que facilitam a análise exploratória de dados. Um exemplo seria a criação de um gráfico de dispersão para explorar a relação entre duas variáveis. Conforme mostrado nas figuras 17 e 18.

```python
import plotly.express as px
import pandas as pd

# Gerar dados fictícios para exemplo
df = pd.DataFrame({
    'A': [1, 2, 3, 4, 5],
    'B': [2, 3, 4, 5, 6],
    'C': ['X', 'Y', 'X', 'Y', 'Z']
})

# Gráfico de dispersão interativo
fig = px.scatter(df, x='A', y='B', color='C', size='B', hover_data=['A'])

# Exibir o gráfico
fig.show()
```

Figura 17

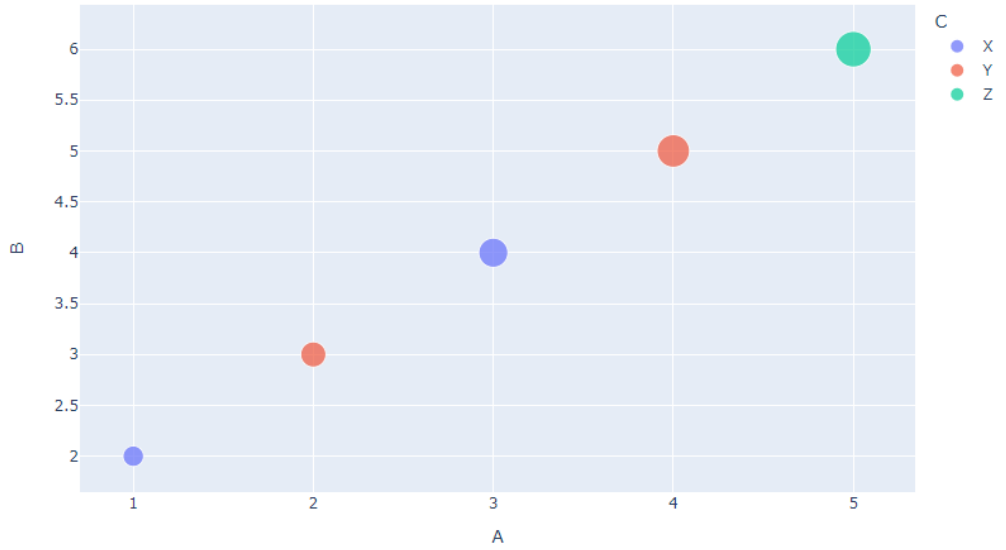

Figura 18

- Visualização de tendências ao longo do tempo.

Para visualizar tendências ao longo do tempo, um gráfico de linhas interativo pode ser eficaz. Vamos supor que estamos analisando o crescimento de vendas ao longo dos meses. Podemos observar nas figuras 19 e 20.

```python
import plotly.express as px
import pandas as pd

# Gerar dados fictícios para exemplo
df = pd.DataFrame({
    'Data': pd.date_range(start='2022-01-01', periods=12, freq='M'),
    'Vendas': [10, 15, 8, 12, 20, 25, 18, 22, 27, 30, 35, 28]
})

# Gráfico de linhas interativo
fig = px.line(df, x='Data', y='Vendas', markers=True, title='Tendência de Vendas ao Longo do Tempo')

# Exibir o gráfico
fig.show()
```

Figura 19

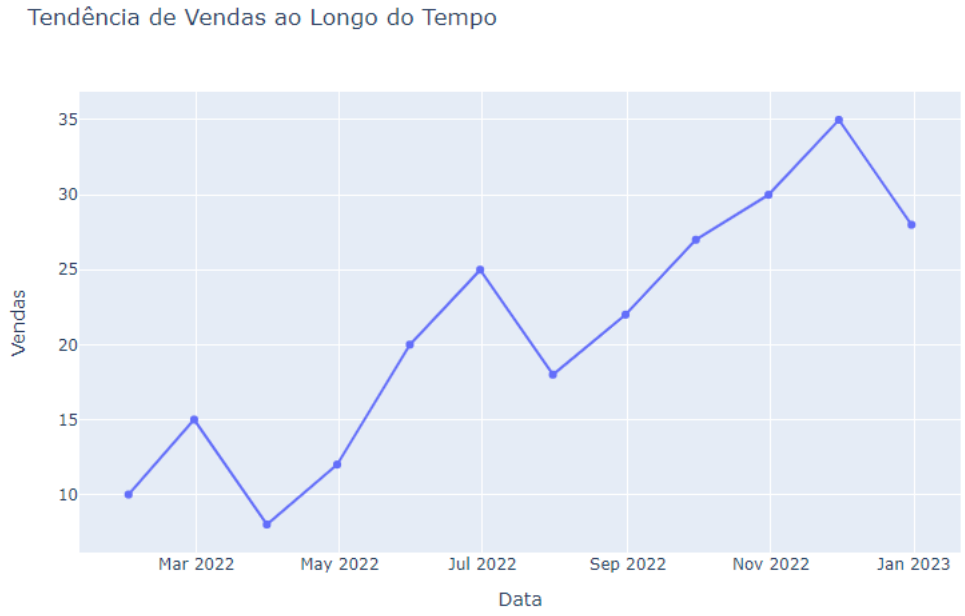

Figura 20

- Comparação de categorias.

  A Plotly é útil para criar gráficos de barras interativos para comparar categorias. Neste exemplo, compararemos a receita de diferentes produtos. Conforme mostrado nas figuras 21 e 22.

```python
import plotly.express as px
import pandas as pd

# Gerar dados fictícios para exemplo
df = pd.DataFrame({
    'Produto': ['A', 'B', 'C', 'D'],
    'Receita': [100, 150, 120, 200]
})

# Gráfico de barras interativo
fig = px.bar(df, x='Produto', y='Receita', text='Receita', title='Comparação de Receita por Produto')

# Exibir o gráfico
fig.show()
```

Figura 21

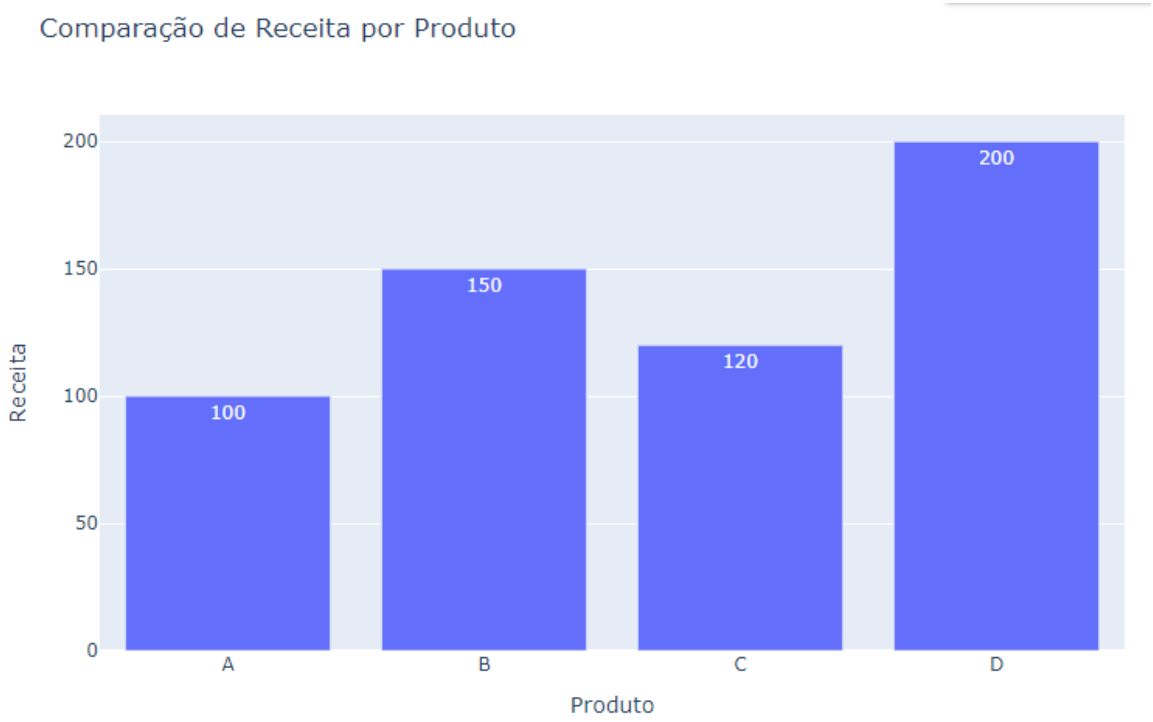

Figura 22

- Análise geoespacial.

  A Plotly também oferece suporte a visualizações geoespaciais. Vamos criar um mapa de calor interativo para visualizar a densidade de ocorrências em diferentes regiões. Observe as figuras 23 e 24.

```python
import plotly.express as px
import pandas as pd

# Gerar dados fictícios para exemplo
df = pd.DataFrame({
    'Cidade': ['A', 'B', 'C', 'D'],
    'Ocorrencias': [10, 5, 15, 8]
})

# Mapa de calor interativo
fig = px.density_mapbox(df, lat=[1, 2, 3, 4], lon=[1, 2, 3, 4], z='Ocorrencias',
                        radius=10, center=dict(lat=2.5, lon=2.5), zoom=1,
                        mapbox_style="stamen-terrain")

# Exibir o gráfico
fig.show()
```

Figura 23

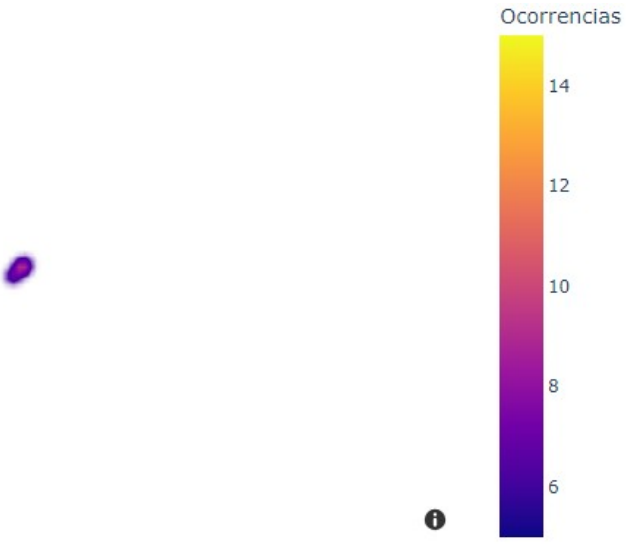

Figura 24

Esses são apenas exemplos básicos para ilustrar a aplicação da Plotly em diferentes casos de uso em projetos de Ciência de Dados. A biblioteca oferece uma variedade de gráficos e opções de personalização, permitindo a criação de visualizações poderosas e interativas para análise de dados.

4. Integração com Outras Bibliotecas

Ao longo dos exemplos explorados, não limitamos nossa abordagem apenas ao Plotly, mas também incorporamos outras bibliotecas muito reconhecidas do Python, como Pandas (Figuras 7, 9, 13 e 15) e NumPy (Figuras 13 e 15). A versatilidade do Plotly é evidente na sua integração eficiente com diversas outras bibliotecas populares, como exemplo a Matplotlib e outras que não foram abordadas nos exemplos apresentados. Essas integrações oferecem uma

ampla gama de possibilidades, elevando o nível de detalhes, complexidade e diversidade de informações disponíveis para visualização.

Conforme demonstrado, utilizamos funcionalidades do Pandas para criar e manipular dataframes, enquanto utilizamos o poder de cálculo e manipulação de dados do NumPy. Há várias outras maneiras de combinar essas duas bibliotecas com o Plotly, assim como também a oportunidade de integrar outras bibliotecas conforme necessário para atender às necessidades específicas de cada situação. Essa flexibilidade permite uma abordagem adaptável e eficaz na análise e visualização de dados em Python.

5. Considerações Finais

Ao decorrer deste artigo vimos uma introdução do que é a biblioteca Plotly, e como começar a utilizar a mesma pode ser simples e rápido. Vimos como realizar a instalação e importação de pacotes que precisamos para começar a usar.

Além disso, vimos alguns exemplos práticos de gráficos assim como discutimos situações em que cada tipo é mais adequado.

Destacamos também a versatilidade da Plotly ao mostrar a integração com outras bibliotecas, ampliando assim a capacidade de processamento e visualização de dados. Essa capacidade de integração não apenas enriquece a experiência do usuário, mas também proporciona uma gama mais ampla de possibilidades para análises detalhadas e representações visuais.

Referências

Acervo Lima. Tutorial do Python Plotly. Disponível em: <https://acervolima.com/tutorial-do-python-plotly/>. Acesso em: 12 de Nov. de 2023.

Análise de Dados com Python. Awari, 2023. Disponível em: <https://awari.com.br/analise-de-dados-com-python/>. Acesso em: 11 de Nov. de 2023.

Pandas. Disponível em: <https://pandas.pydata.org/docs/index.html/>. Acesso em: 12 de Nov. de 2023.

Paulo Vasconcellos. Como Criar Gráficos Interativos Utilizando Plotly e Python. Disponível em: <https://paulovasconcellos.com.br/como-criar-gr%C3%A1ficos-interativos-utilizando-plotly-e-python-3eb6eda57a2b/>. Acesso em: 12 de Nov. de 2023.

Plotly. Disponível em: <https://plotly.com/python/>. Acesso em: 12 de Nov. de 2023.

Python para Data Science: Explorando a Linguagem Python em Análise de Dados. Awari, 2023. Disponível em: <https://awari.com.br/python-para-data-science-explorando-a-linguagem-python-em-analise-de-dados/>. Acesso em: 11 de Nov. de 2023.

Conclusão

Este artigo serve como uma introdução à biblioteca Plotly em Python e como um guia prático para sua utilização em projetos de Ciência de Dados. Através de exemplos e discussões sobre diferentes tipos de gráficos e casos de uso, esperamos que tenha servido ao seu propósito e que venha a ajudar outras pessoas a conhecer e iniciar a utilização desta poderosa ferramenta.