Digital College

FORMAÇÃO EM

DATA ANALYTICS

MÓDULO 4:

GRÁFICOS COM PYTHON

UNIDADE 3:

<PYTHON PARA ANÁLISE DE DADOS>

Sumário

Sumário

1. Introdução

A visualização de dados desempenha um papel crucial na análise e na comunicação efetiva das informações contidas nos dados. Ela envolve a representação gráfica dos dados de forma clara e compreensível, permitindo que as pessoas identifiquem padrões, tendências e insights relevantes de maneira mais rápida e intuitiva.

Aqui estão alguns dos principais motivos pelos quais a visualização de dados é importante:

- Compreensão dos dados: os seres humanos têm uma capacidade natural de entender informações visuais com mais facilidade do que dados brutos ou números em forma tabular. A visualização de dados ajuda a simplificar a complexidade dos dados, tornando-os mais acessíveis e compreensíveis para qualquer pessoa, mesmo para aquelas sem conhecimento técnico avançado;
- Detecção de padrões e tendências: através de gráficos e visualizações, é possível identificar padrões, tendências e correlações nos dados que podem passar despercebidos em uma análise puramente numérica. As visualizações destacam características importantes dos dados e ajudam a revelar insights valiosos;
- Tomada de decisões informadas: a visualização de dados ajuda a embasar a tomada de decisões com base em evidências sólidas. Ao ver os dados representados graficamente, é mais fácil identificar oportunidades, tendências negativas, pontos fortes e fracos, permitindo que as decisões sejam respaldadas por informações claras e confiáveis;
- Comunicação efetiva: as visualizações de dados são ferramentas poderosas para comunicar informações complexas de maneira clara e concisa. Elas ajudam a transmitir mensagens e histórias de dados de forma impactante e engajadora, tornando mais fácil compartilhar insights e convencer outras pessoas sobre a importância de determinada informação ou curso de ação;
- Exploração e descoberta: as visualizações de dados possibilitam a exploração interativa desses elementos, permitindo que os usuários analisem diferentes perspectivas e detalhes específicos. Isso facilita a descoberta de informações ocultas, anomalias ou padrões inesperados, estimulando a curiosidade e a análise mais aprofundada dos dados.

A visualização de dados desempenha um papel crucial na análise e interpretação dos dados, auxiliando na compreensão, na tomada de decisões informadas e na comunicação eficaz dos insights extraídos. Ela capacita as pessoas a explorar, analisar e comunicar informações de maneira mais eficiente e impactante.

Os gráficos desempenham um papel fundamental na análise exploratória de dados. A análise exploratória de dados envolve a exploração inicial de um conjunto de dados para entender sua estrutura, suas características e seus possíveis insights.
Para identificação de distribuições, temos os gráficos (como histogramas e gráficos de densidades), que permitem visualizar alsso é crucial para entender a natureza dos dados e selecionar a abordagem adequada de análise estatística.

Na exploração de dados multidimensionais, às vezes, temos conjuntos de dados com várias variáveis. Nesses casos, os gráficos de dispersão de múltiplas variáveis, como matrizes de dispersão ou gráficos de pares, são úteis para identificar relações complexas entre as variáveis e visualizar clusters ou agrupamentos de dados.

Para comparação e resumo de dados, utilizamos gráficos de barras, gráficos de pizza e gráficos de caixa (boxplots) são ferramentas úteis para comparar categorias, resumir estatísticas e destacar diferenças entre grupos ou categorias. Esses gráficos ajudam a obter uma visão geral dos dados e a identificar diferenças significativas.

Visualização geoespacial é quando os dados estão relacionados a locais geográficos, como dados demográficos ou dados de vendas por região, os gráficos de mapas são essenciais. Eles permitem visualizar dados em um contexto geográfico e identificar padrões espaciais.
Os gráficos fornecem uma representação visual dos dados, permitindo uma compreensão mais rápida e intuitiva das informações. Eles ajudam os analistas de dados a explorarem e comunicarem insights de forma eficaz, destacando padrões, tendências e relações importantes. Portanto, é altamente recomendado o uso de gráficos na análise exploratória de dados.

> > 1.1. Bibliotecas gráficas em Python

Aqui estão algumas bibliotecas populares para criação de gráficos em Python:

- Matplotlib: é uma das bibliotecas mais antigas e amplamente utilizadas para a geração de gráficos em Python. Ela oferece uma ampla gama de opções de plotagem e pode ser usada para criar gráficos estáticos, gráficos interativos e visualizações animadas;
- Seaborn: é uma biblioteca de visualização de dados baseada no Matplotlib. Ela fornece uma interface de alto nível para criar gráficos estatísticos atraentes e informativos. O Seaborn é frequentemente usado em combinação com o Pandas para análise de dados;
- Plotly: é uma biblioteca de gráficos interativos que oferece uma ampla gama de visualizações, desde gráficos de linhas simples até gráficos de dispersão 3D e mapas de calor. O Plotly pode ser usado em ambientes web e desktop;
- Bokeh: é uma biblioteca de visualização interativa que se concentra na geração de gráficos interativos para a web. Ela permite criar gráficos interativos complexos com facilidade e se integra bem com outras bibliotecas Python, como NumPy e Pandas;
- ggplot: inspirada na biblioteca R ggplot2, o ggplot oferece uma sintaxe semelhante para criar gráficos em Python. Ele fornece uma forma concisa e elegante de criar gráficos estatísticos usando uma gramática de gráficos intuitiva.

Essas são apenas algumas das bibliotecas de gráficos disponíveis em Python. Cada uma tem suas próprias vantagens e recursos distintos, por isso é recomendável explorá-las e escolher aquela que melhor atenda às suas necessidades e preferências.

2. Biblioteca Matplotlib

Matplotlib é uma biblioteca popular para criação de gráficos em Python. Ela fornece uma ampla gama de funcionalidades para criar visualizações estáticas, interativas e animadas. Aqui estão alguns conceitos e teorias importantes relacionados à biblioteca Matplotlib:

- Figuras e eixos: no Matplotlib, uma figura é uma área de desenho em branco onde os gráficos são criados. Dentro da figura, podemos criar um ou mais eixos (também conhecidos como subplots), que são áreas retangulares onde os elementos do gráfico são colocados;

- API de plotagem: o Matplotlib oferece duas interfaces para criação de gráficos – a API orientada a objetos e a API de plotagem (também chamada de Pyplot). A API de plotagem é uma interface mais simples e fácil de usar, que permite criar gráficos com apenas algumas linhas de código. Ela é geralmente importada como plt e a maioria das funções de plotagem são chamadas a partir desse objeto;

- Tipos de gráficos: o Matplotlib oferece uma variedade de tipos de gráficos, incluindo gráficos de linha, gráficos de dispersão, gráficos de barras, gráficos de pizza, histogramas, entre outros. Cada tipo de gráfico tem sua própria função correspondente no Matplotlib;

- Personalização dos gráficos: o Matplotlib permite personalizar praticamente todos os aspectos dos gráficos. É possível definir cores, estilos de linha, marcadores, títulos, legendas, rótulos dos eixos, escalas, entre outros. A biblioteca oferece uma ampla gama de opções para controlar a aparência visual dos gráficos;

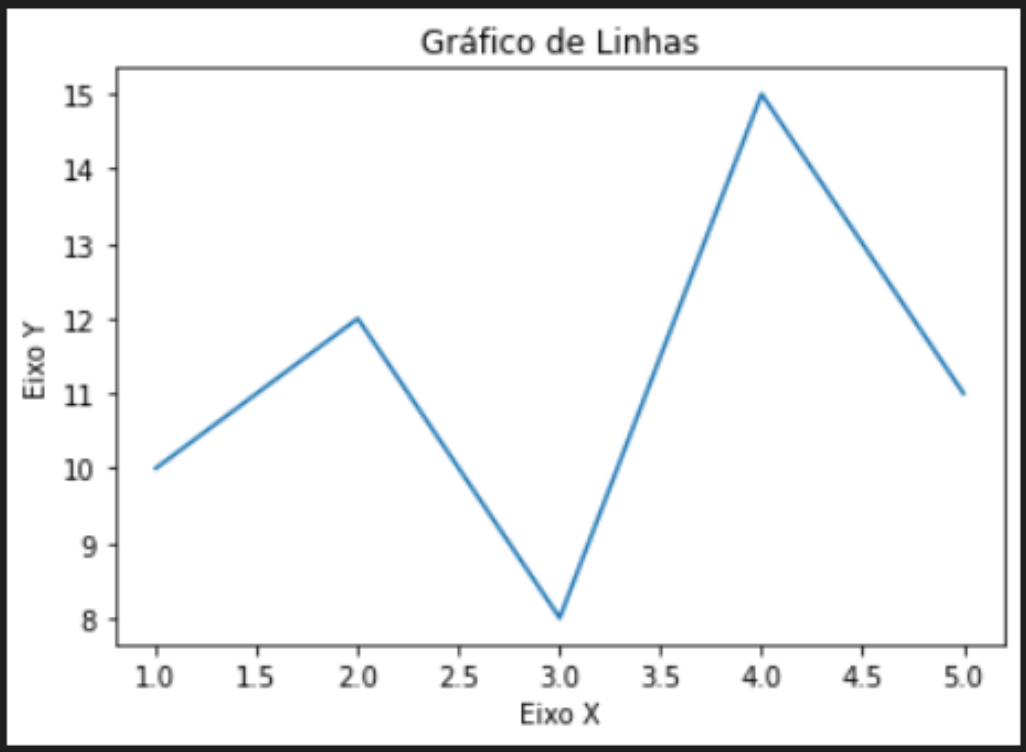
- Anotações e texto: o Matplotlib permite adicionar anotações, textos explicativos e rótulos em diferentes partes dos gráficos. Isso pode ser feito usando as funções text(), annotate(), xlabel(), ylabel(), entre outras;

- Exportação de gráficos: o Matplotlib permite exportar os gráficos criados em vários formatos, como PNG, PDF, SVG, EPS, entre outros. Isso é útil para salvar os gráficos em arquivos para uso posterior ou para incorporá-los em documentos e relatórios.

Esses são apenas alguns dos conceitos e teorias fundamentais relacionados à biblioteca Matplotlib. À medida que você utiliza o Matplotlib em seus projetos e explora sua documentação, você encontrará uma infinidade de opções e recursos adicionais para criar gráficos personalizados e expressivos.

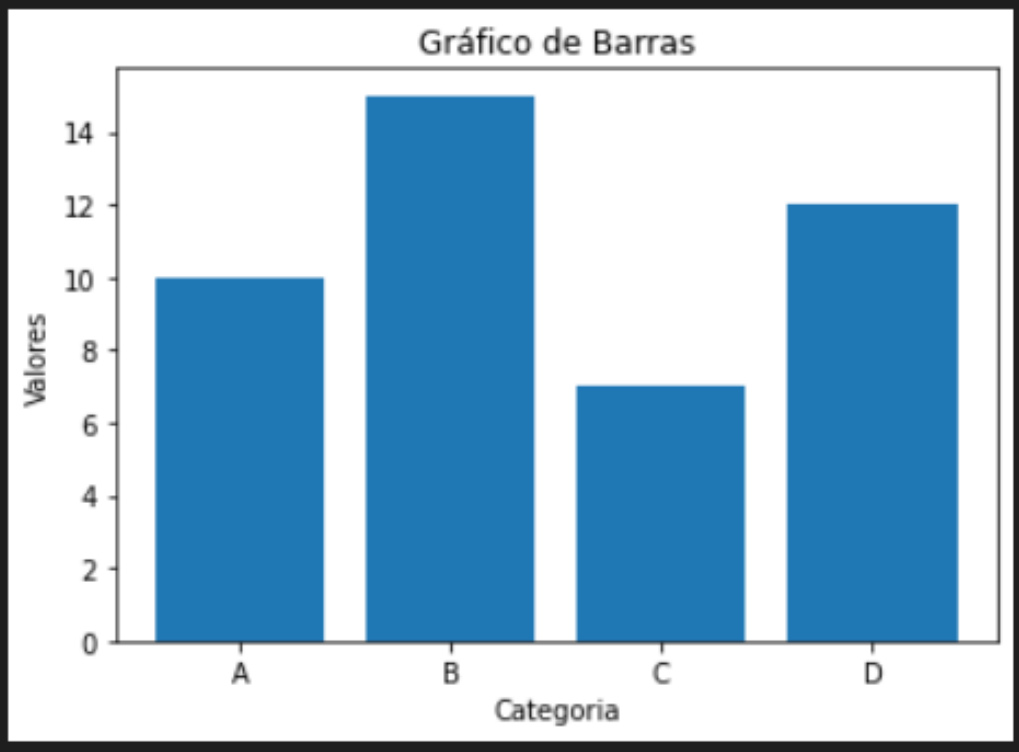
> > 2.1. Gráfico de linhas com Matplotlib

```python
import matplotlib.pyplot as plt

# Dados de exemplo
x = [1, 2, 3, 4, 5]
y = [10, 12, 8, 15, 11]

# Criar o gráfico de linhas
plt.plot(x, y)

# Adicionar rótulos e título
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linhas')

# Exibir o gráfico
plt.show()
```



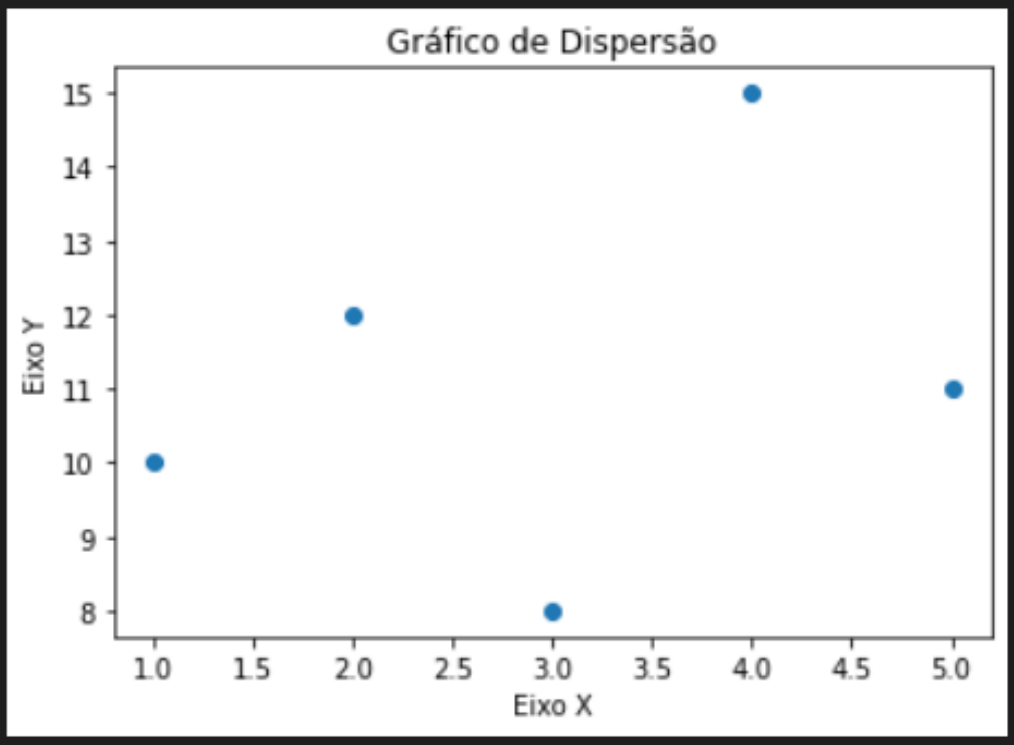
> > 2.2. Gráfico de barras com Matplotlib

```python
import matplotlib.pyplot as plt

# Dados de exemplo
x = ['A', 'B', 'C', 'D']
y = [10, 15, 7, 12]

# Criar o gráfico de barras
plt.bar(x, y)

# Adicionar rótulos e título
plt.xlabel('Categoria')
plt.ylabel('Valores')
plt.title('Gráfico de Barras')

# Exibir o gráfico
plt.show()
```



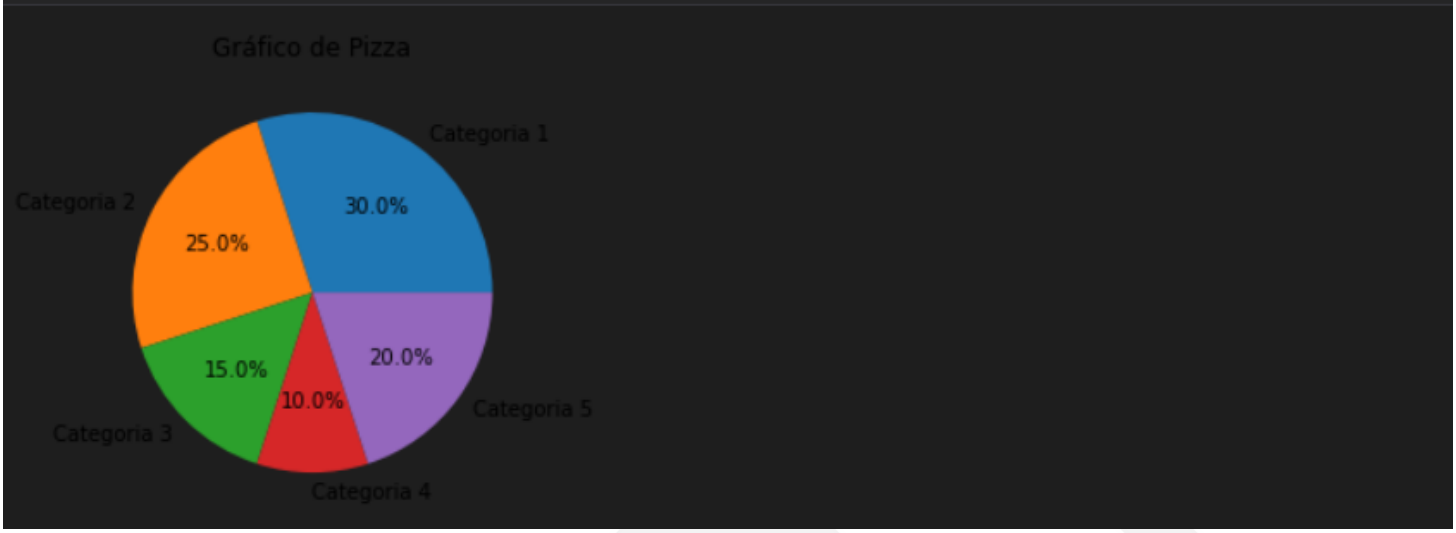
> > 2.3. Gráfico de dispersão com Matplotlib

```python
import matplotlib.pyplot as plt

# Dados de exemplo
x = [1, 2, 3, 4, 5]
y = [10, 12, 8, 15, 11]

# Criar o gráfico de dispersão
plt.scatter(x, y)

# Adicionar rótulos e título
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Dispersão')

# Exibir o gráfico
plt.show()
```



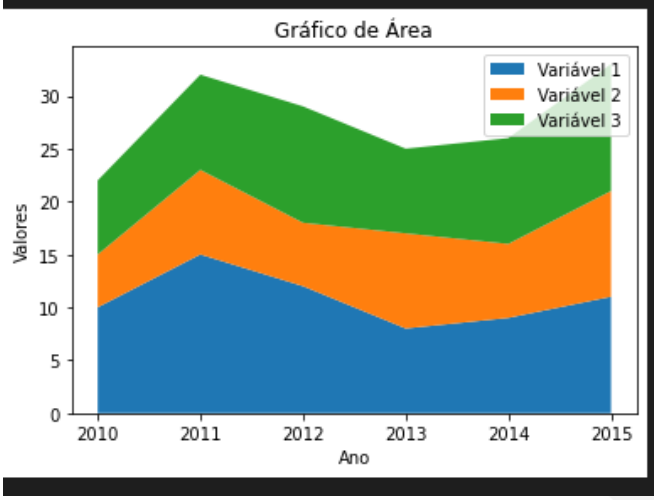
> > 2.4. Gráfico de pizza com Matplotlib

```python
import matplotlib.pyplot as plt

# Dados de exemplo
sizes = [30, 25, 15, 10, 20]
labels = ['Categoria 1', 'Categoria 2', 'Categoria 3', 'Categoria 4', 'Categoria 5']

# Criar o gráfico de pizza
plt.pie(sizes, labels=labels, autopct='%1.1f%%')

# Adicionar título
plt.title('Gráfico de Pizza')

# Exibir o gráfico
plt.show()
```



> > 2.5. Gráfico de área com Matplotlib

```python
import matplotlib.pyplot as plt

# Dados de exemplo
anos = [2010, 2011, 2012, 2013, 2014, 2015]
variavel1 = [10, 15, 12, 8, 9, 11]
variavel2 = [5, 8, 6, 9, 7, 10]
variavel3 = [7, 9, 11, 8, 10, 12]

# Criar o gráfico de área
plt.stackplot(anos, variavel1, variavel2, variavel3, labels=['Variável 1', 'Variável 2', 'Variável 3'])

# Adicionar rótulos e título
plt.xlabel('Ano')
plt.ylabel('Valores')
plt.title('Gráfico de Área')

# Adicionar legenda
plt.legend()

# Exibir o gráfico
plt.show()
```



Esses são apenas alguns exemplos básicos para demonstrar o uso do Matplotlib. A biblioteca oferece muitas outras opções de personalização, como a escolha de cores, estilos de linha, legendas, entre outros. Com o Matplotlib, você pode criar uma ampla variedade de gráficos para visualizar seus dados de maneira eficaz e informativa.

3. Biblioteca Seaborn

Seaborn é uma biblioteca de visualização de dados em Python, construída sobre o Matplotlib. Ela fornece uma interface de alto nível para criar gráficos estatísticos atraentes e informativos. Aqui estão alguns conceitos e teorias importantes relacionados à biblioteca Seaborn:

- Estilo de gráficos: o Seaborn possui estilos de gráficos predefinidos que melhoram a estética dos gráficos em comparação com o estilo padrão do Matplotlib. Os estilos podem ser alterados usando a função set_style(), permitindo personalizar a aparência dos gráficos para se adequarem ao contexto;

- Paletas de cores: o Seaborn oferece uma variedade de paletas de cores agradáveis para a criação de gráficos. Essas paletas são úteis para a distinção de categorias, a criação de mapas de calor e a representação de dados em várias dimensões. As paletas podem ser acessadas por meio da função color_palette() e configuradas usando a função set_palette();

- Plotagem estatística: o Seaborn é especialmente útil para plotagens estatísticas, como gráficos de distribuição, gráficos de categorias, gráficos de regressão e gráficos de correlação. Ele fornece funções específicas para criar esses tipos de gráficos, como distplot(), barplot(), regplot() e heatmap();

- Facet Grids: o Seaborn permite criar Facet Grids, que são grades de subplots que dividem os dados em diferentes subconjuntos com base em variáveis categóricas. Isso é útil para comparar relações entre diferentes categorias em um único gráfico, facilitando a análise exploratória de dados em várias dimensões;

- Visualização de regressão: o Seaborn fornece recursos avançados para plotagem de modelos de regressão. Ele oferece funções como lmplot(), regplot() e jointplot() para visualizar relações lineares, ajustar modelos de regressão e representar gráficos de dispersão com distribuições marginais;

- Estética e temas: o Seaborn permite personalizar a estética dos gráficos com várias opções de formatação, incluindo títulos, rótulos dos eixos, tamanhos de fonte e configurações de grade. Além disso, ele oferece diferentes temas visuais, como darkgrid, whitegrid, dark, white e ticks, que podem ser definidos usando a função set_theme().

Esses são alguns conceitos e algumas teorias fundamentais relacionados à biblioteca Seaborn. Ele simplifica a criação de gráficos estatísticos atraentes e fornece recursos adicionais para aprimorar a visualização de dados. Ao explorar mais a biblioteca, você descobrirá uma ampla gama de opções de personalização e recursos avançados para criar visualizações informativas e impactantes.

> > 3.1. Gráfico de linha Seaborn

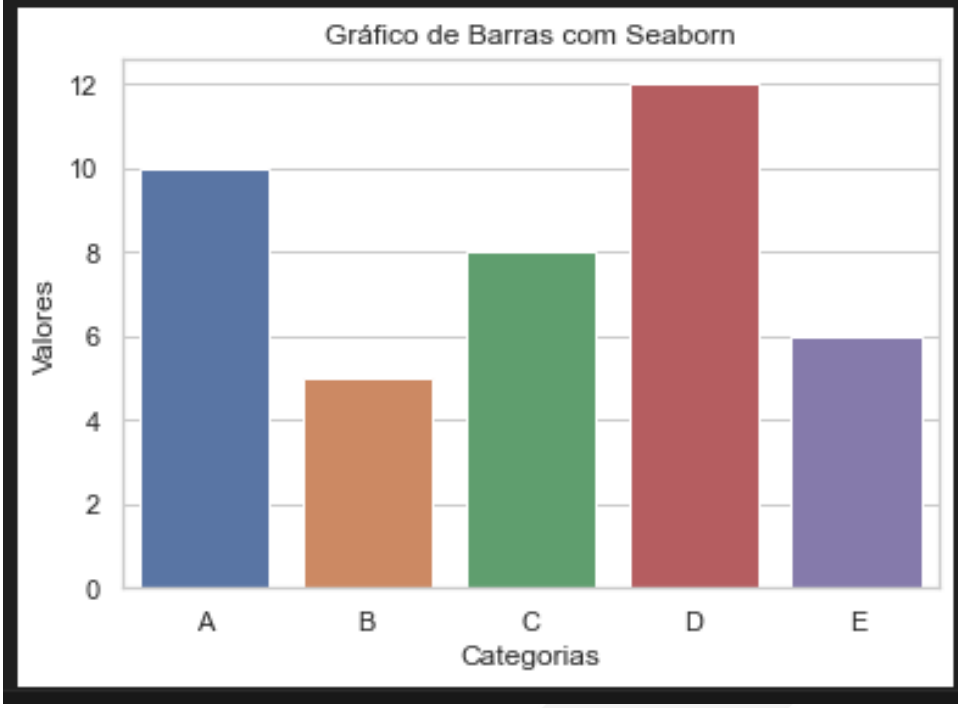
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Dados
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Configurações do estilo Seaborn
sns.set(style="darkgrid")

# Criando o gráfico de linha
sns.lineplot(x=x, y=y)

# Adicionando rótulos e título
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linha com Seaborn')

# Exibindo o gráfico
plt.show()
```

✓ 0.2s

> > 3.2. Gráfico de Barras com Seaborn

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Dados
x = ['A', 'B', 'C', 'D', 'E']
y = [10, 5, 8, 12, 6]

# Configurações do estilo Seaborn
sns.set(style="whitegrid")

# Criando o gráfico de barras
sns.barplot(x=x, y=y)

# Adicionando rótulos e título
plt.xlabel('Categorias')
plt.ylabel('Valores')
plt.title('Gráfico de Barras com Seaborn')

# Exibindo o gráfico
plt.show()
```

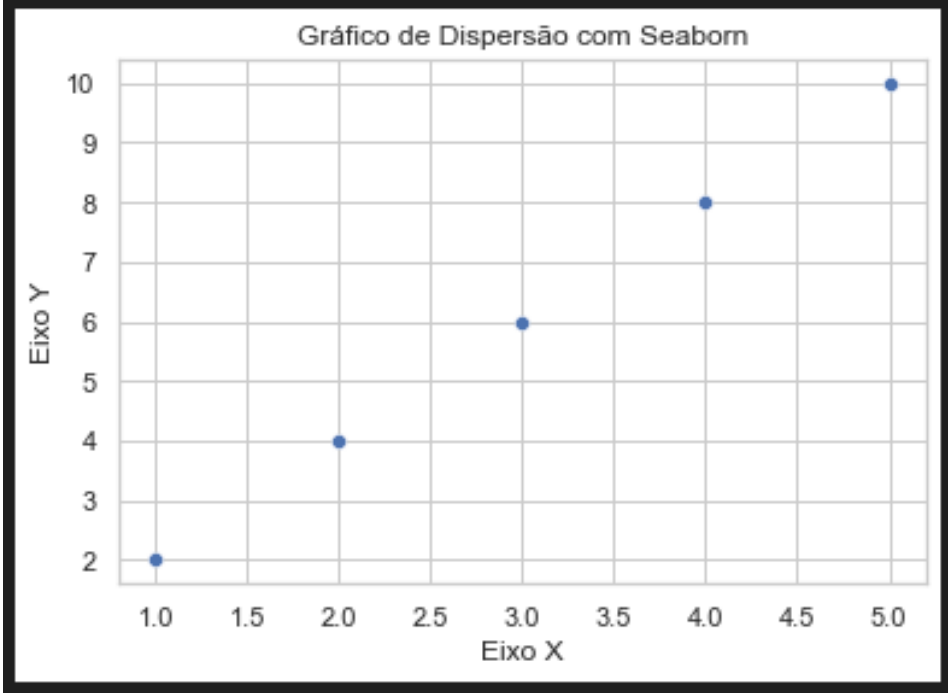
✓ 0.1s



> > 3.3. Gráfico de Dispersão com Seaborn

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Dados
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Configurações do estilo Seaborn
sns.set(style="whitegrid")

# Criando o gráfico de dispersão
sns.scatterplot(x=x, y=y)

# Adicionando rótulos e título
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Dispersão com Seaborn')

# Exibindo o gráfico
plt.show()
```

✓ 0.2s

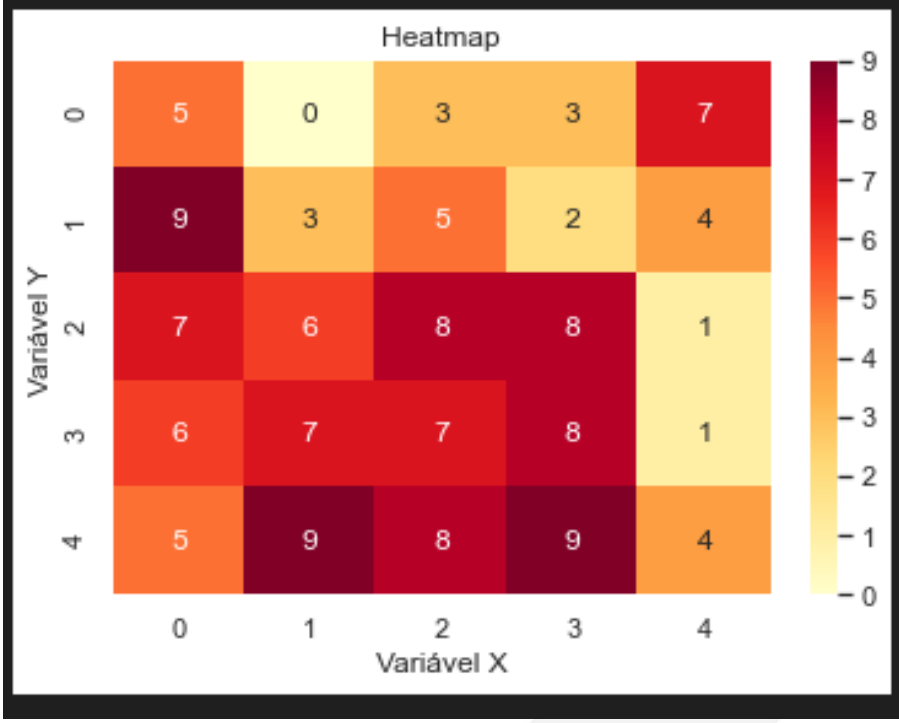


> > 3.4. Gráfico de Heatmaps em Seaborn

Os heatmaps, ou mapas de calor, são uma forma eficaz de visualizar a relação entre duas variáveis categóricas em forma de matriz. O Seaborn oferece a função heatmap() para criar esse tipo de gráfico. Aqui está um exemplo de como usar o heatmap no Seaborn:

```python
import seaborn as sns
import numpy as np

# Dados simulados
np.random.seed(0)
data = np.random.randint(0, 10, (5, 5))

# Criando o heatmap
sns.heatmap(data, annot=True, cmap='YlOrRd')

# Adicionando rótulos e título
plt.xlabel('Variável X')
plt.ylabel('Variável Y')
plt.title('Heatmap')

# Exibindo o gráfico
plt.show()
```

✓ 0.3s

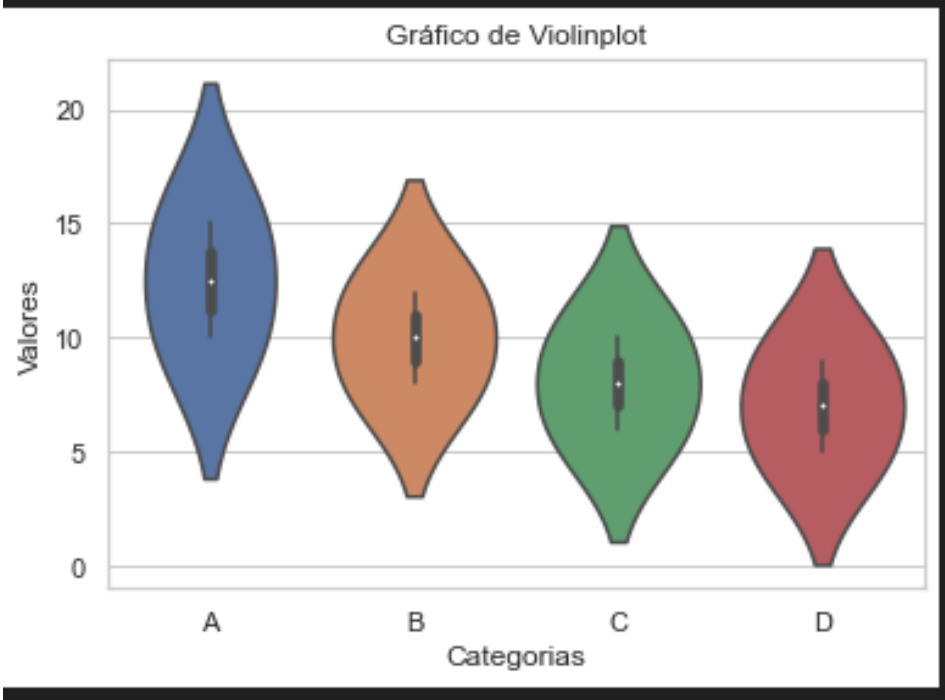


> > 3.5. Gráfico de Violinplot em Seaborn

O gráfico de violinplot é uma visualização estatística que combina um boxplot com um gráfico de densidade, permitindo visualizar a distribuição de uma variável numérica em relação a uma variável categórica. O Seaborn oferece a função violinplot() para criar esse tipo de gráfico. Aqui está um exemplo de como usar o violinplot no Seaborn:

```
import seaborn as sns

# Dados
x = ['A', 'A', 'B', 'B', 'C', 'C', 'D', 'D']
y = [10, 15, 8, 12, 6, 10, 5, 9]

# Criando o gráfico de violinplot
sns.violinplot(x=x, y=y)

# Adicionando rótulos e título
plt.xlabel('Categorias')
plt.ylabel('Valores')
plt.title('Gráfico de Violinplot')

# Exibindo o gráfico
plt.show()
```

✓ 0.2s



4. Biblioteca Ploty

A biblioteca Plotly é uma poderosa ferramenta de visualização interativa em Python, que permite criar gráficos interativos e personalizados. Ela oferece uma ampla variedade de tipos de gráficos, desde gráficos básicos, como barras e dispersões, até gráficos mais complexos, como gráficos de superfície 3D e gráficos de densidade.

A principal característica do Plotly é a sua capacidade de criar gráficos interativos que podem ser explorados e manipulados pelo usuário. Esses gráficos podem ser salvos como arquivos HTML ou incorporados em aplicativos da web, facilitando a criação de visualizações dinâmicas e interativas.
Além disso, o Plotly possui uma sintaxe simples e intuitiva, tornando a criação e a personalização de gráficos mais acessíveis. Ele também é compatível com várias bibliotecas e frameworks populares, como o Jupyter Notebook, o Pandas e o Dash.

Aqui estão alguns exemplos dos tipos de gráficos que você pode criar com o Plotly:

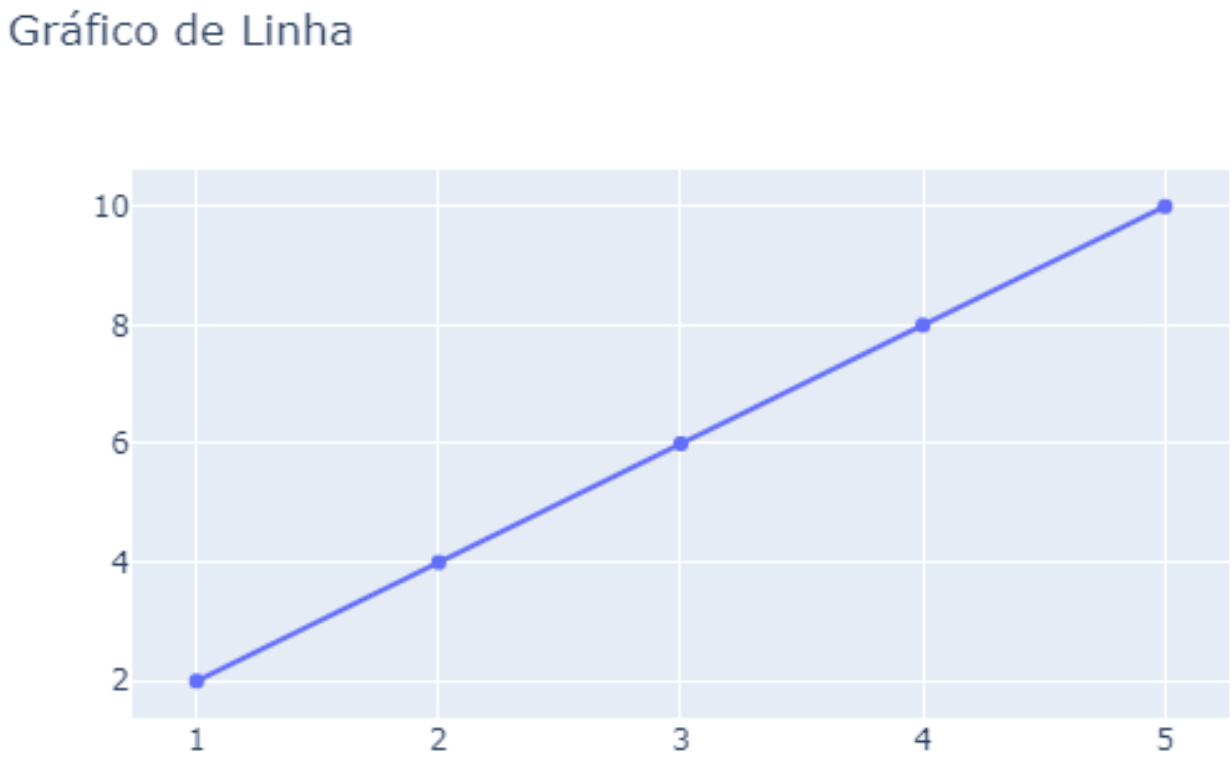
- Gráficos de linha;
- Gráficos de barras;
- Gráficos de dispersão;
- Gráficos de pizza;
- Gráficos de área;
- Gráficos de superfície 3D;
- Gráficos de caixa;
- Gráficos de densidade.

A biblioteca Plotly também oferece recursos avançados, como animações, subplots, seleção e filtro de dados interativos, entre outros.
Para começar a usar o Plotly, você precisará instalar a biblioteca usando o pip. Em seguida, você pode importar a biblioteca e começar a criar seus gráficos. Consulte a documentação oficial do Plotly para obter mais informações sobre como instalar e usar a biblioteca, além de exemplos detalhados de diferentes tipos de gráficos e suas opções de personalização.

> > 4.1. Gráfico de linha em Plotly

```
import plotly.graph_objects as go

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

fig = go.Figure(data=go.Scatter(x=x, y=y))
fig.update_layout(
    title='Gráfico de Linha',
    width=600,    # Define a largura do gráfico como 600 pixels
    height=400    # Define a altura do gráfico como 400 pixels
)
fig.show()
```

✓ 0.0s



> > 4.2. Gráfico de barras em Plotly

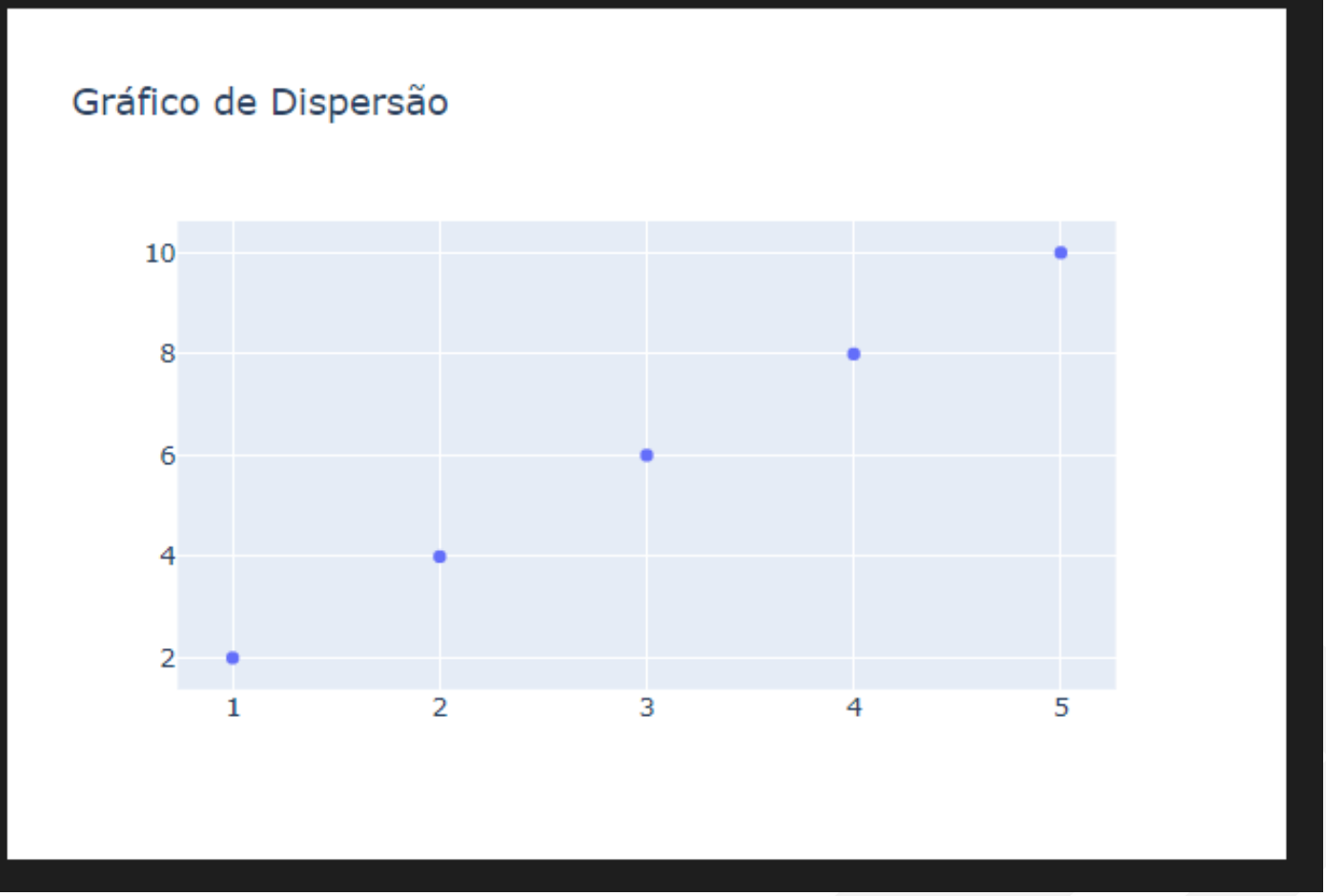
```python
import plotly.graph_objects as go

x = ['A', 'B', 'C', 'D']
y = [10, 5, 8, 12]

fig = go.Figure(data=go.Bar(x=x, y=y))
fig.update_layout(
    title='Gráfico de Barras',
    width=600,    # Define a largura do gráfico como 600 pixels
    height=400    # Define a altura do gráfico como 400 pixels
)
fig.show()
```

✓ 0.0s

> > 4.3. Gráfico de dispersão em Plotly

```
import plotly.graph_objects as go

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

fig = go.Figure(data=go.Scatter(x=x, y=y, mode='markers'))
fig.update_layout(
    title='Gráfico de Linha',
    width=600,   # Define a largura do gráfico como 600 pixels
    height=400   # Define a altura do gráfico como 400 pixels
)
fig.update_layout(title='Gráfico de Dispersão')
fig.show()
```

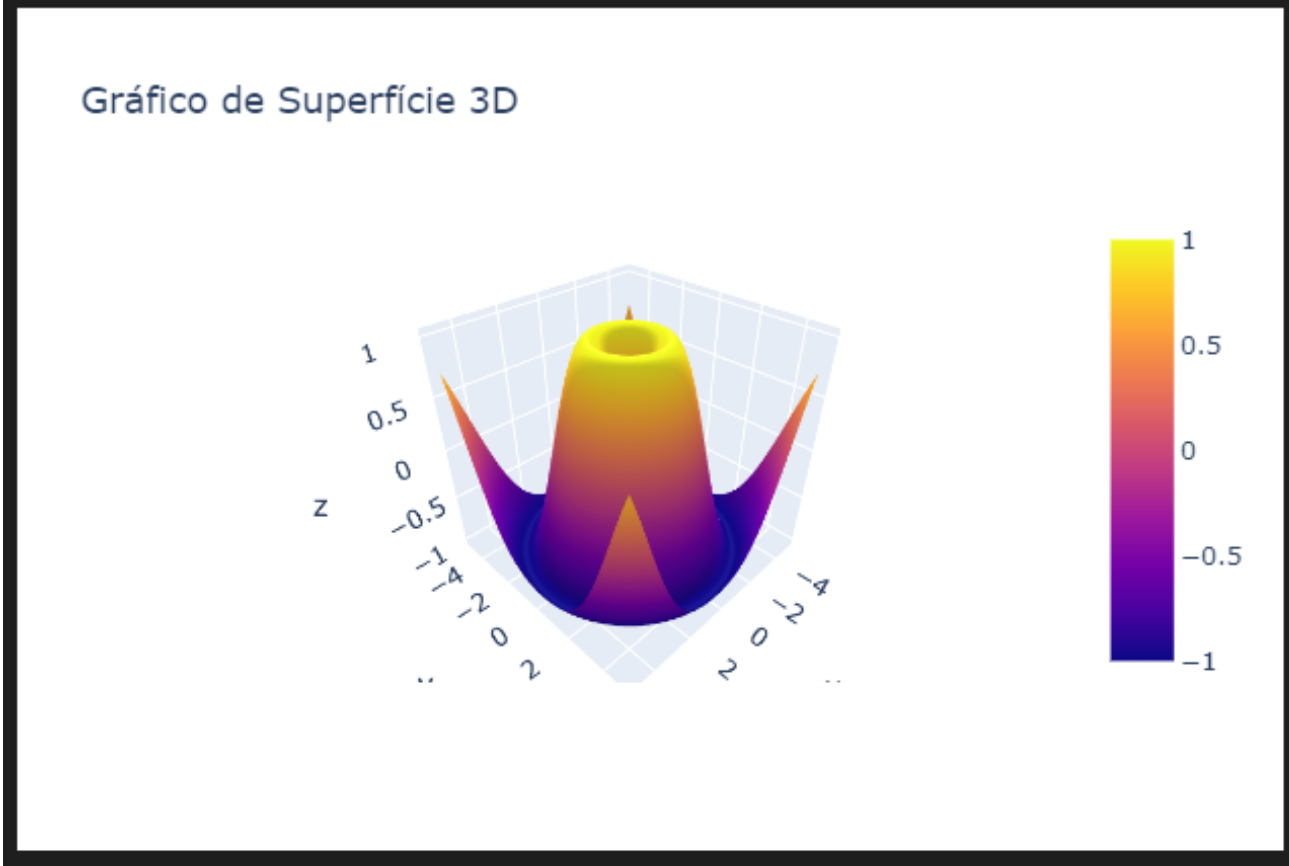
✓ 0.0s



> > 4.4. Gráficos de superfície 3D em Plotly

```python
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.sin(np.sqrt(X**2 + Y**2))

# Criando o gráfico de superfície 3D
fig = go.Figure(data=[go.Surface(x=x, y=y, z=Z)])
fig.update_layout(
    title='Gráfico de Linha',
    width=600,    # Define a largura do gráfico como 600 pixels
    height=400    # Define a altura do gráfico como 400 pixels
)
fig.update_layout(title='Gráfico de Superfície 3D')
fig.show()
```

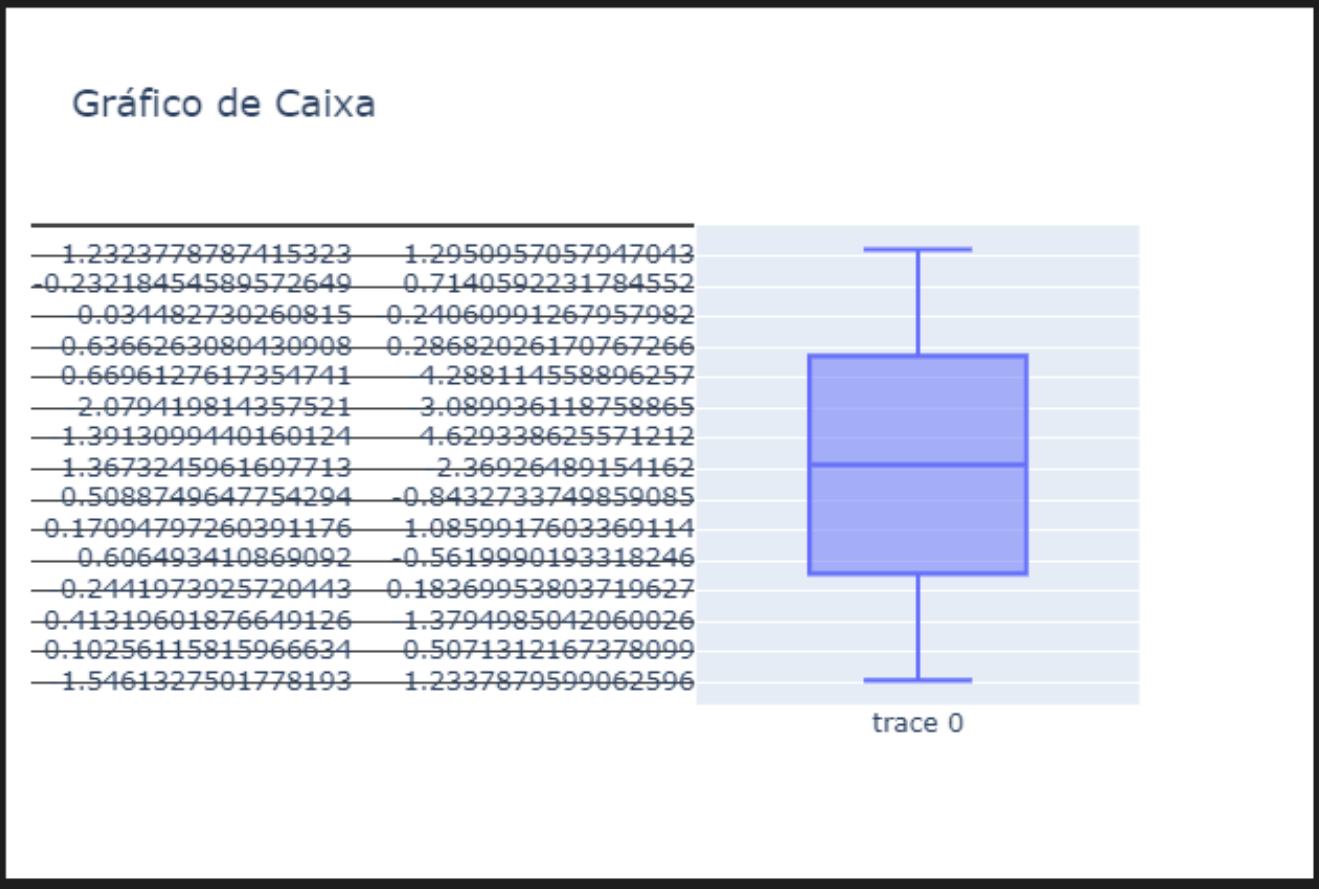
✓ 6.5s



> > 4.5. Gráficos de caixa em Plotly

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

# Criando os dados
data = [np.random.normal(0, std, 100) for std in range(1, 4)]

# Criando o gráfico de caixa
fig = go.Figure(data=go.Box(y=data))
fig.update_layout(
    title='Gráfico de Linha',
    width=600,    # Define a largura do gráfico como 600 pixels
    height=400    # Define a altura do gráfico como 400 pixels
)
fig.update_layout(title='Gráfico de Caixa')
fig.show()
```

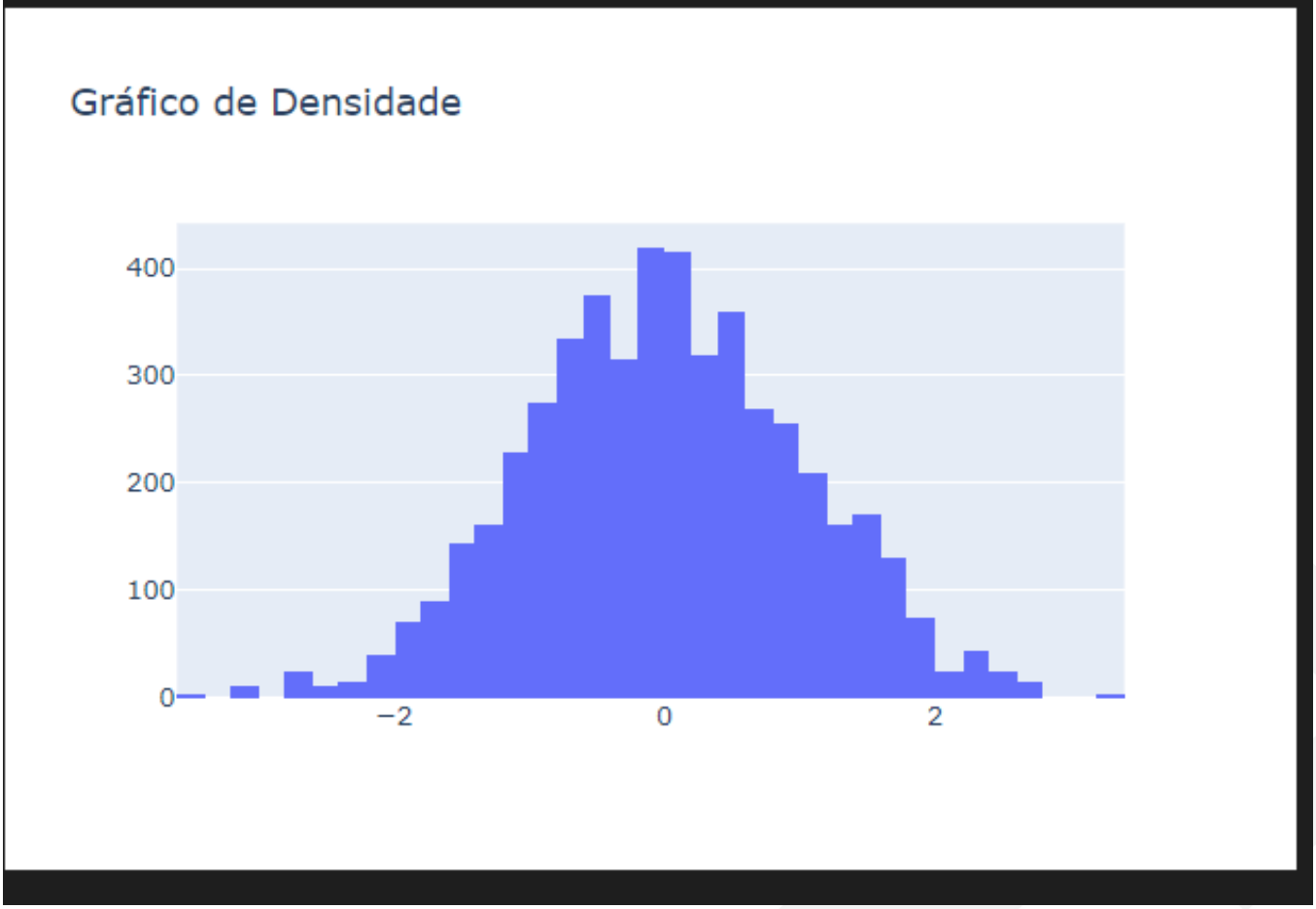
✓ 0.0s



> > 4.6. Gráficos de Densidade

```python
import plotly.graph_objects as go
import numpy as np

# Criando os dados
data = np.random.randn(1000)

# Criando o gráfico de densidade
fig = go.Figure(data=go.Histogram(x=data, histnorm='density'))
fig.update_layout(
    title='Gráfico de Linha',
    width=600,   # Define a largura do gráfico como 600 pixels
    height=400   # Define a altura do gráfico como 400 pixels
)
fig.update_layout(title='Gráfico de Densidade')
fig.show()
```

✓ 0.0s



5. Biblioteca Bokeh

A biblioteca gráfica Bokeh é uma biblioteca de visualização interativa em Python que permite criar gráficos elegantes e interativos para análise de dados. Ela é especialmente útil para a criação de gráficos interativos em aplicativos web.

Alguns recursos e conceitos importantes do Bokeh incluem:

- Bokeh Models: o Bokeh trabalha com um conjunto de modelos que representam visualizações gráficas, como figuras, eixos, legendas etc. Esses modelos são compostos para criar visualizações completas;

- Bokeh Plotting: o Bokeh possui uma API de alto nível chamada "bokeh.plotting" que fornece uma interface fácil e intuitiva para criar gráficos rápidos e simples. Ele também suporta uma API de nível mais baixo chamada "bokeh.models" para criação de gráficos mais personalizados;

- Integração com o navegador: o Bokeh permite que você gere visualizações em um formato interativo, que pode ser incorporado em aplicativos web e exibido em um navegador. Ele gera gráficos interativos usando JavaScript, permitindo que os usuários interajam com as visualizações;

- Suporte a diferentes tipos de gráficos: o Bokeh oferece suporte a uma ampla variedade de tipos de gráficos, incluindo gráficos de linha, gráficos de barra, gráficos de dispersão, gráficos de área, gráficos de pizza, gráficos de caixa e gráficos de histograma, dentre outros;
- Interatividade: o Bokeh permite que você adicione recursos interativos aos seus gráficos, como ferramentas de zoom, pan, seleção, realce e tooltips. Isso permite que os usuários explorem e analisem os dados de forma interativa;

- Personalização: o Bokeh fornece uma ampla gama de opções de personalização para ajustar a aparência dos gráficos. Você pode personalizar cores, estilos de linha, tamanho de marcadores, rótulos de eixo e títulos, dentre outros.

A biblioteca Bokeh é uma ferramenta poderosa para criar visualizações interativas em Python. Ela combina a facilidade de uso com recursos avançados de personalização e interatividade. Se você deseja criar gráficos interativos em aplicativos web ou explorar seus dados de forma interativa, o Bokeh pode ser uma ótima escolha.

A biblioteca Bokeh oferece uma variedade de tipos de gráficos e visualizações interativas. Aqui estão alguns dos principais tipos de gráficos que você pode criar com o Bokeh:

- Gráficos de Linhas: permitem traçar pontos conectados por linhas para visualizar tendências ou padrões ao longo do tempo ou de uma variável;
- Gráficos de Barras: exibem barras verticais ou horizontais para comparar valores entre diferentes categorias ou grupos;
- Gráficos de Dispersão: mostram pontos individuais em um plano cartesiano, permitindo identificar relacionamentos ou clusters entre duas variáveis;
- Gráficos de Área: exibem áreas preenchidas entre uma linha (ou uma curva) e um eixo, representando a distribuição cumulativa ou o volume acumulado;
- Gráficos de Superfície 3D: representam uma superfície tridimensional com base em coordenadas (x, y, z), criando uma visualização interativa da forma e da altura da superfície;
- Gráficos de Caixa: apresentam a distribuição de um conjunto de dados em termos de quartis, mediana e valores atípicos, ajudando a identificar a dispersão e os valores extremos;
- Gráficos de Densidade: mostram a distribuição de probabilidade de um conjunto de dados, representando a densidade relativa dos valores em um intervalo contínuo;
- Gráficos de Histograma: apresentam a distribuição de frequência de um conjunto de dados em intervalos ou classes, usando barras para representarem a contagem ou a proporção de observações em cada intervalo;
- Gráficos de Mapa: possibilitam visualizar dados geográficos em mapas interativos, permitindo exibir informações sobre regiões ou marcadores em um contexto espacial.

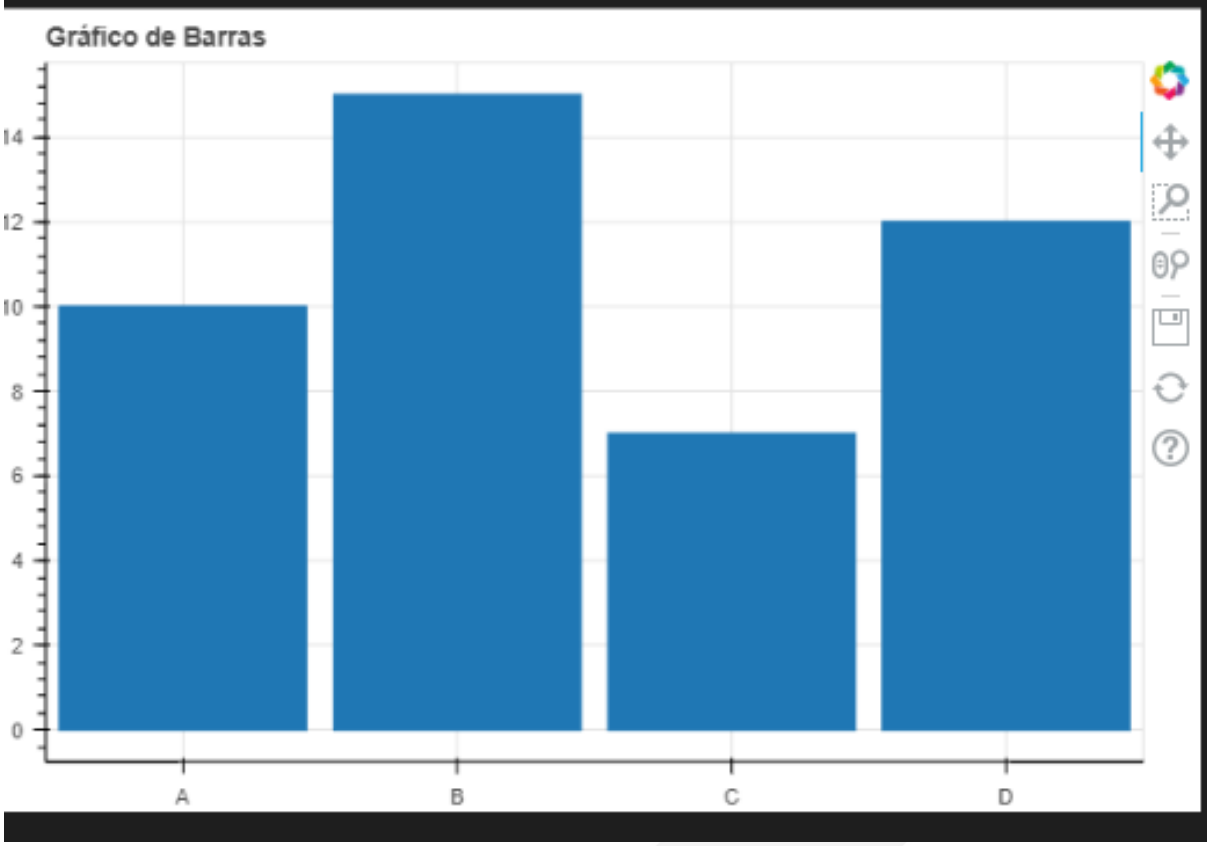
Esses são apenas alguns exemplos dos tipos de gráficos que você pode criar com a biblioteca Bokeh. O Bokeh também oferece uma variedade de opções de personalização e interatividade para ajudar na exploração e na análise dos dados.

> > 5.1. Gráfico em barras com Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

# Ativar a saída inline dos gráficos no Jupyter Notebook
output_notebook()

# Dados para o gráfico de barras
categorias = ['A', 'B', 'C', 'D']
valores = [10, 15, 7, 12]

# Criar a figura do gráfico de barras
fig = figure(x_range=categorias, plot_height=400, plot_width=600, title='Gr

# Adicionar as barras ao gráfico
fig.vbar(x=categorias, top=valores, width=0.9)

# Exibir o gráfico
show(fig)
```

✓ 18.0s

BokehJS 2.3.2 successfully loaded.

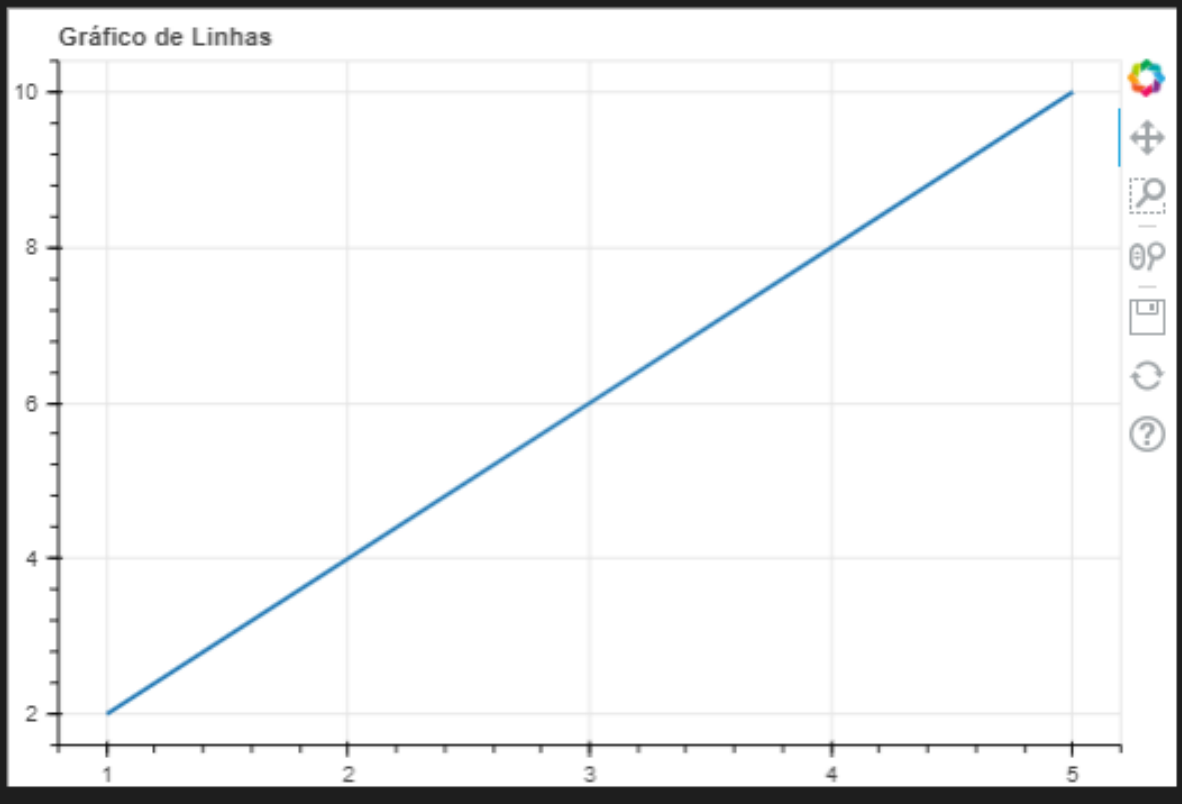


> > 5.2. Gráfico de linhas com Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

# Ativar a saída inline dos gráficos no Jupyter Notebook
output_notebook()

# Dados para o gráfico de linhas
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Criar a figura do gráfico de linhas
fig = figure(plot_height=400, plot_width=600, title='Gráfico de Linhas')

# Adicionar a linha ao gráfico
fig.line(x, y, line_width=2)

# Exibir o gráfico
show(fig)
```

✓ 0.0s

BokehJS 2.3.2 successfully loaded.

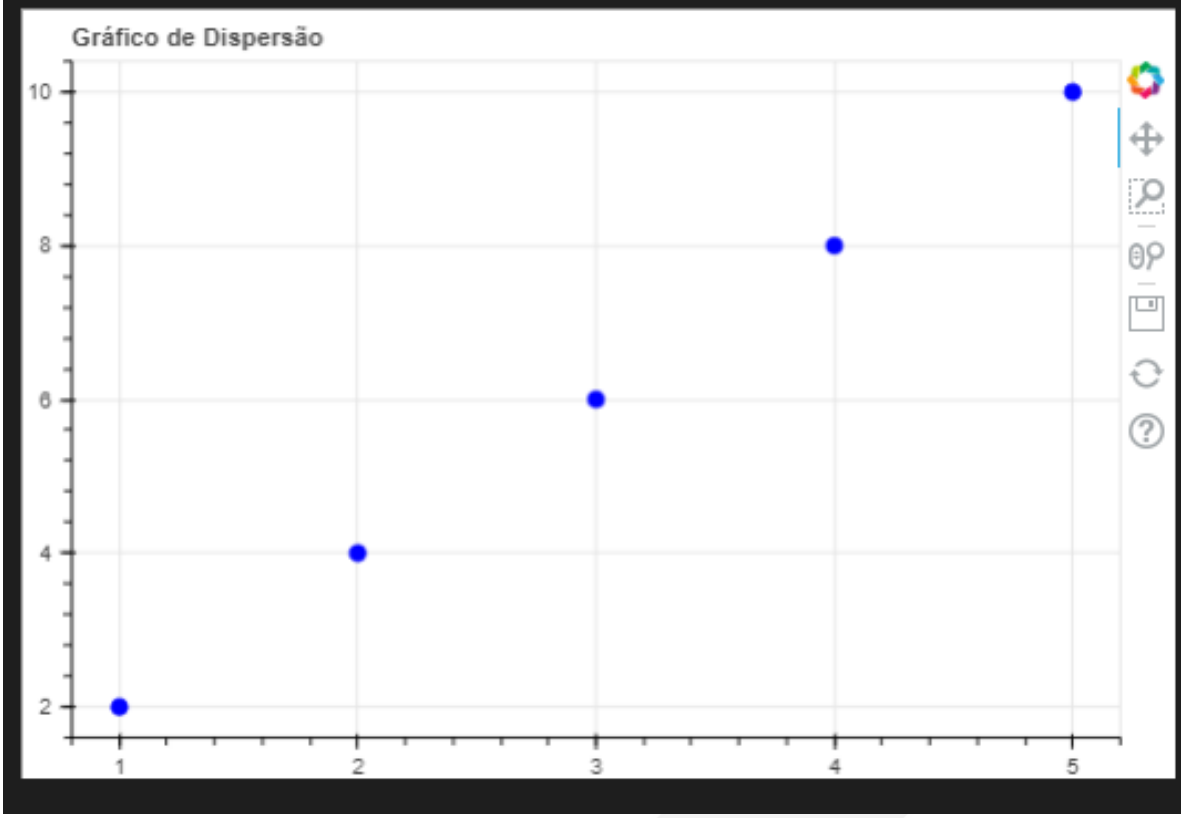


> > 5.3. Gráfico de dispersão com Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

# Ativar a saída inline dos gráficos no Jupyter Notebook
output_notebook()

# Dados para o gráfico de dispersão
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Criar a figura do gráfico de dispersão
fig = figure(plot_height=400, plot_width=600, title='Gráfico de Dispersão')

# Adicionar os pontos de dispersão ao gráfico
fig.scatter(x, y, size=8, color='blue')

# Exibir o gráfico
show(fig)
```

✓ 0.0s

BokehJS 2.3.2 successfully loaded.

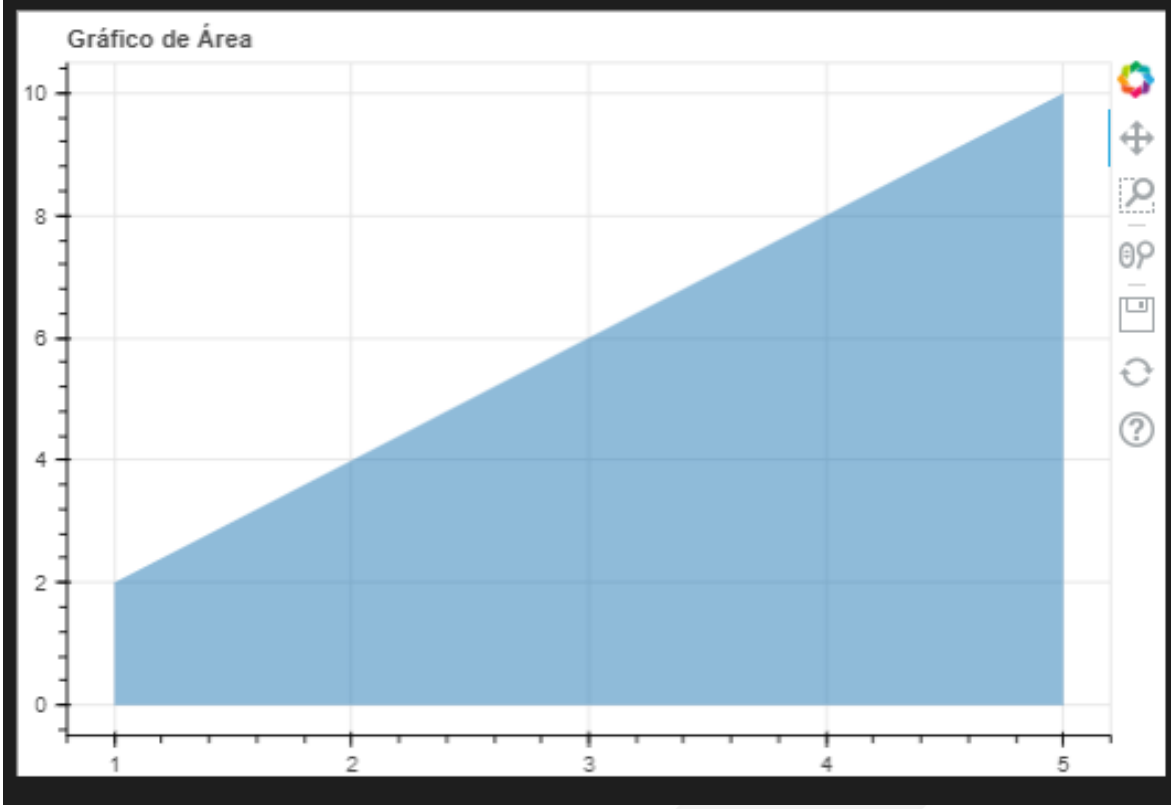


> > 5.4. Gráfico de área com Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

# Ativar a saída inline dos gráficos no Jupyter Notebook
output_notebook()

# Dados para o gráfico de área
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Criar a figura do gráfico de área
fig = figure(plot_height=400, plot_width=600, title='Gráfico de Área')

# Adicionar a área ao gráfico
fig.varea(x=x, y1=y, y2=0, fill_alpha=0.5)

# Exibir o gráfico
show(fig)
```

✓ 0.0s

BokehJS 2.3.2 successfully loaded.



Referências bibliográficas

BANIN, Sérgio Luiz. Python 3 - Conceitos e Aplicações - Uma Abordagem Didática [BV:MB]. São Paulo: Érica, 2018.

CHEN, Daniel. Análise de dados com Python e Pandas. São Paulo: Novatec, 2018.

FORBELLONE, André L. V.; EBERSPACHER, Henri F. Lógica de Programação: a construção de algoritmos e estruturas de dados [BV:PE]. 3. ed. São Paulo: Editora Pearson, 2005.

LIMA, Janssen dos Reis. Consumindo a API do Zabbix com Python [BV:PE]. Rio de Janeiro: Editora Brasport, 2016.

PERKOVIC, Ljubomir. Introdução à Computação Usando Python - Um Foco no Desenvolvimento de Aplicações [BV:MB]. 1ª ed. Rio de Janeiro: LTC, 2016.

SEBESTA,Robert W. Conceitos de Linguagens de Programação. 11. ed. Porto Alegre: Grupo A, 2011.
TUCKER, Allen; NOONAN, Robert. Linguagens de Programação: Princípios e Paradigmas. Porto Alegre: Grupo A.

VLADISHEV, A. Consumindo a API do Zabbix com Python. Rio de Janeiro: Brasport.

Digital College

ENSINO DE HABILIDADES DIGITAIS

digitalcollege.com.br