

centro
universitário



Programação Científica

Prof. Dr. Danilo H. Perico

Visualização de Dados

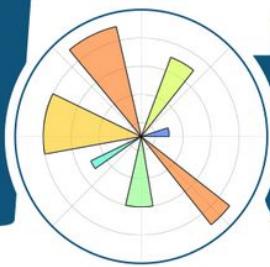
Visualização de Dados

- A visualização de dados consiste na apresentação de informações através de elementos visuais, como gráficos, diagramas, mapas, tabelas, entre outros
 - Por meio dela, fica muito mais fácil analisar os resultados, auxiliando o processo de identificar padrões e tendências e tomar decisões
 - Boas visualizações de dados ajudam a descomplicar as informações, transmitindo uma mensagem clara e objetiva

Benefícios da Visualização de Dados

- Com o enorme volume de informações disponíveis, as ferramentas de visualização de dados são cada vez mais fundamentais para as empresas
- Benefícios:
 - Apresentação mais envolvente / Absorção rápida das informações
 - Facilita a tomada de decisão
 - Auxilia a encontrar conexões importantes para solucionar os problemas da sua empresa

matplotlib





- Matplotlib é uma biblioteca de software para criação de gráficos e visualizações de dados em geral, feita para a linguagem de programação Python e sua extensão de matemática NumPy
- Criada pelo biólogo e neurocientista **John D. Hunter**, a biblioteca possui uma comunidade ativa atuando em seu desenvolvimento



Artigo sobre o Matplotlib:

<https://ieeexplore.ieee.org/document/4160265>

Publicado em 18/Junho/2007

A screenshot of the IEEE Xplore digital library interface. At the top, it shows the IEEE Xplore logo, a sign-in menu with 'Access provided by: Centro Universitário Fai', and a 'Sign Out' button. Below the header is a search bar with the word 'All' and a magnifying glass icon. To the right of the search bar is an 'ADVANCED SEARCH' link. The main content area displays the article 'Matplotlib: A 2D Graphics Environment' by John D. Hunter. The article title is in large bold black font. Below the title are buttons for 'Publisher: IEEE', 'Cite This', and a red 'PDF' button. Further down, there are three boxes showing citation metrics: '13243 Paper Citations', '12 Patent Citations', and '17682 Full Text Views'. The page also includes a navigation breadcrumb: 'Journals & Magazines > Computing in Science & Engine... > Volume: 9 Issue: 3'. On the right side of the page, there are several small icons for sharing and managing the document.

Matplotlib: A 2D Graphics Environment

Publisher: IEEE

Cite This

PDF

John D. Hunter All Authors

13243

Paper
Citations

12

Patent
Citations

17682

Full
Text Views



SCIENTIFIC PROGRAMMING

Editors: George K. Thiruvathukal, gkt@cs.luc.edu
Konstantin Läufer, laufer@cs.luc.edu



MATPLOTLIB: A 2D GRAPHICS ENVIRONMENT

By John D. Hunter

Matplotlib is a 2D graphics package used for Python for application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems.

I began learning Python in 2001, mainly as a way to procrastinate during the final stages of preparing my dissertation. I was hooked in short order. My numerical and statistical workflow at the time was a mix of Fortran, C++, and Matlab; I used the file system to communicate be-

in a GUI for application development, support different platforms, offer extremely high-quality raster and vector (primarily PostScript) hardcopy output for publication, provide support for mathematical expressions, and work interactively from the shell.



- Oferece uma interface de programação orientada a objetos para incluir gráficos em aplicações usando toolkits de interface gráfica
- Permite que você trace gráficos diferentes, como gráficos de barras, dispersão, histogramas, espectros e muito mais
- Muitas vezes requer apenas algumas linhas de código

matplotlib pyplot

- É instalado junto com o Anaconda
- Para usar:

```
import matplotlib.pyplot as plt
```

- Pyplot:
 - Permite a criação de figuras, uma área para exibir o gráfico na figura, desenhar linhas na área do gráfico, decorar o gráfico com rótulos, etc.

matplotlib pyplot

- No pyplot, vários estados são preservados após a chamada de uma função dentro de um contexto, simplificando assim o seu trabalho sobre a figura ou área de desenho atuais
- Assim, você pode ter uma figura complexa formada por várias áreas de gráficos distintas, e o pyplot mantém para você informações sobre para cada área de desenho

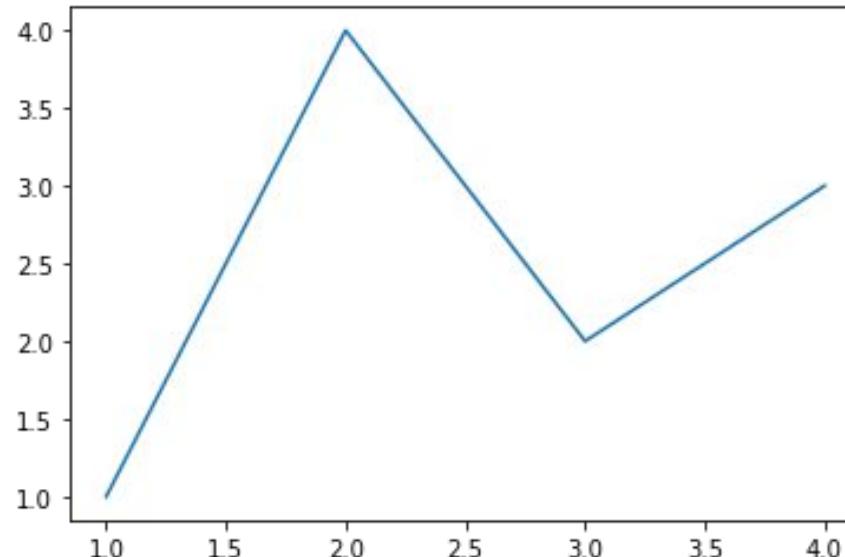
matplotlib Criando um Gráfico

- Podemos criar um **objeto gráfico** usando a função `subplots()`:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.show()
```

matplotlib Criando um Gráfico



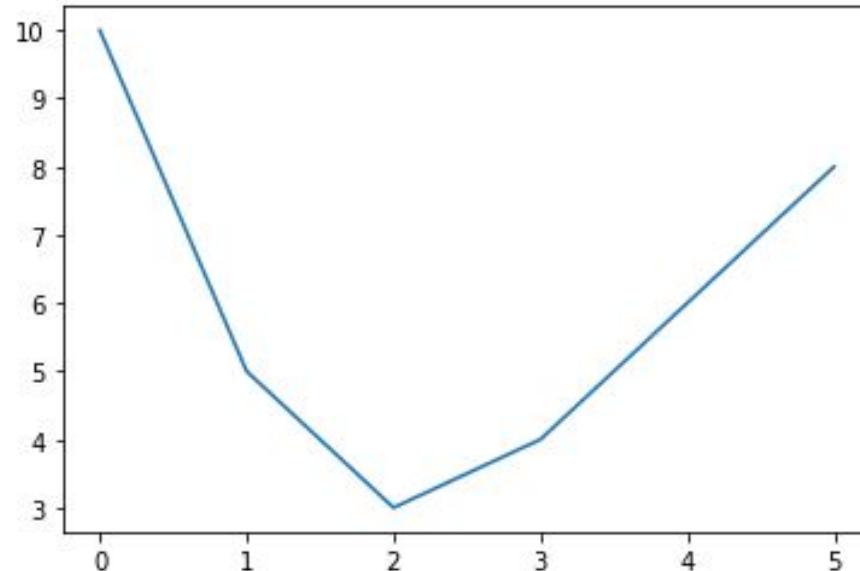
matplotlib Criando um Gráfico

- Podemos criar um objeto gráfico usando a função `plot()`:

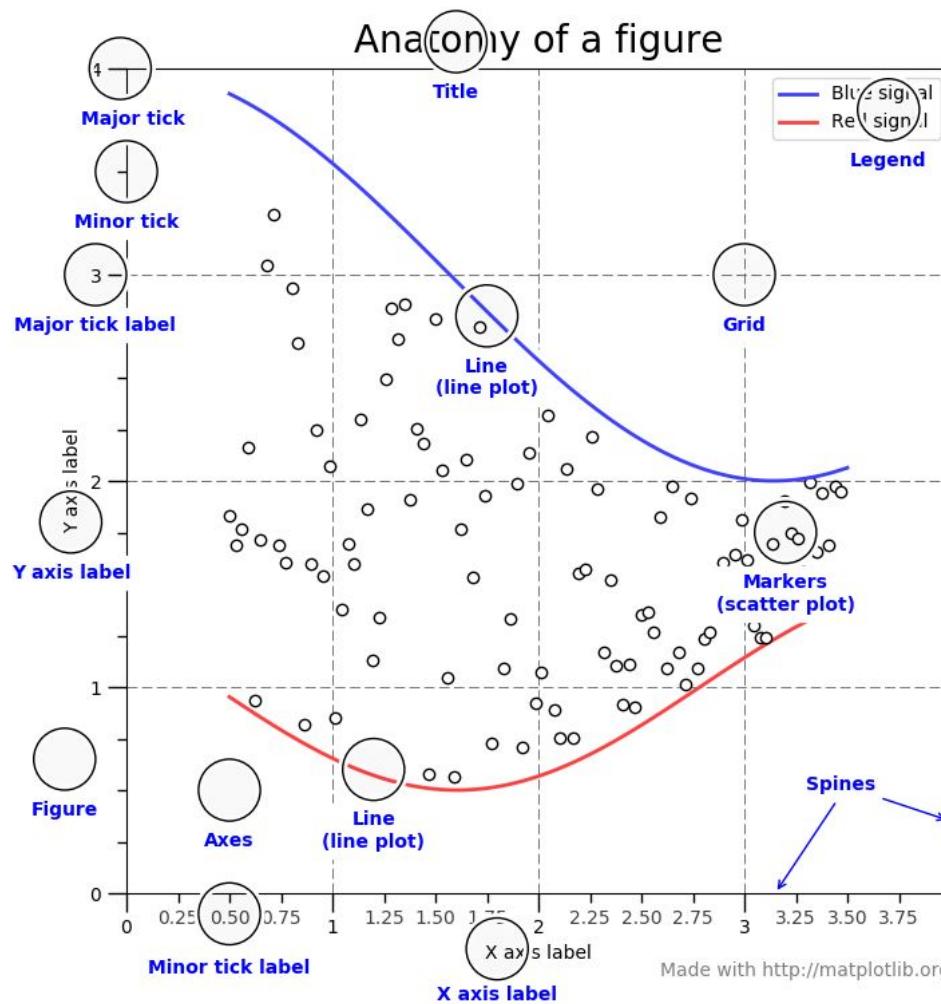
```
import matplotlib.pyplot as plt  
plt.plot( [10,5,3,4,6,8] )  
plt.show()
```

- O método `plot()` executa o desenho nos gráficos
- Ele cria o gráfico se ele ainda não existir

matplotlib Criando um Gráfico



Anatomia de um Gráfico



matplotlib Criando um Gráfico

- Conforme acabamos de ver, o matplotlib tem 2 sintaxes possíveis: uma baseada em **MATLAB** e outra em **Orientação a Objetos**

```
import matplotlib.pyplot as plt  
plt.plot( [10,5,3,4,6,8] )  
plt.show()
```

Padrão com POO

Padrão do MATLAB

```
import matplotlib.pyplot as plt  
  
fig, ax = plt.subplots()  
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  
plt.show()
```

matplotlib Criando um Gráfico

- O padrão baseado em **MATLAB** parece ser mais fácil, mas tem menos recursos

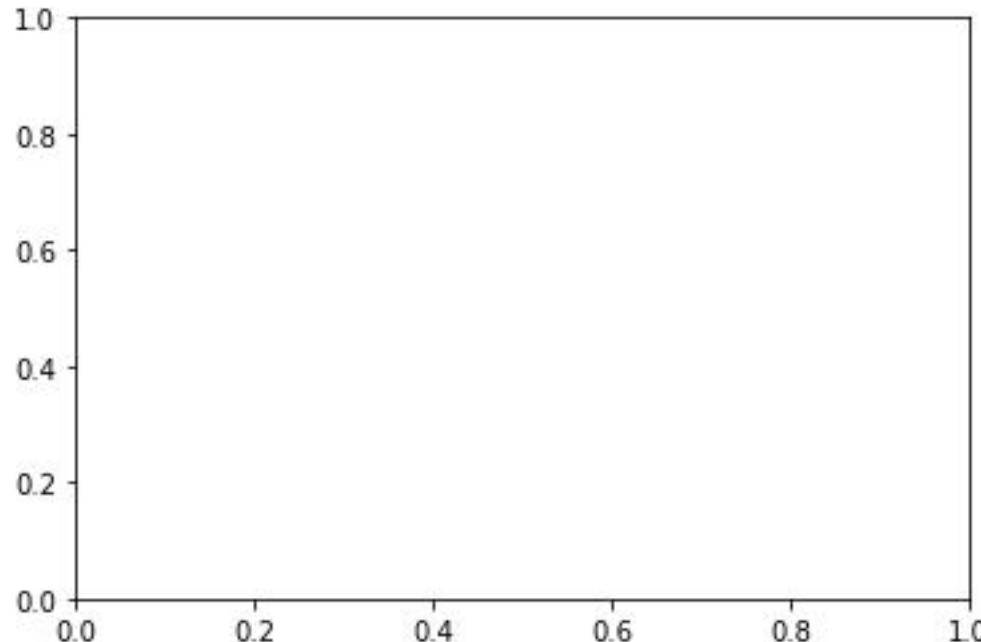
matplotlib Criando uma Figura

- A maneira mais fácil de se criar uma figura é usando a função `figure()`:

```
import matplotlib.pyplot as plt

fig = plt.figure() # an empty figure
fig, ax = plt.subplots() # single Axes
fig, axs = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes
```

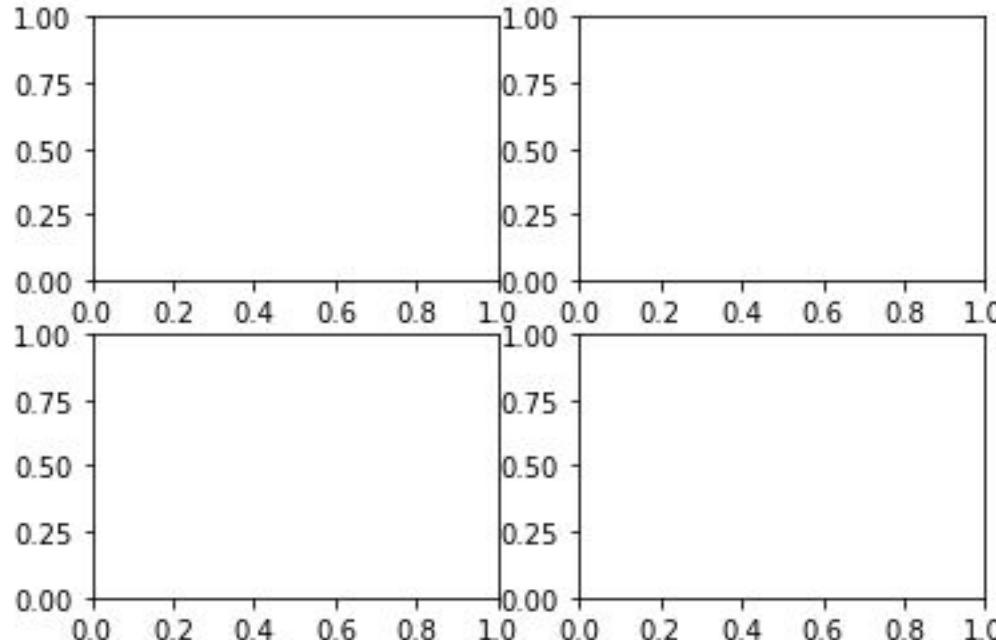
matplotlib Criando uma Figura com 1 Gráfico



```
fig, ax = plt.subplots()
```



Criando uma Figura com 2x2 Gráficos



```
fig, axs = plt.subplots(2, 2)
```

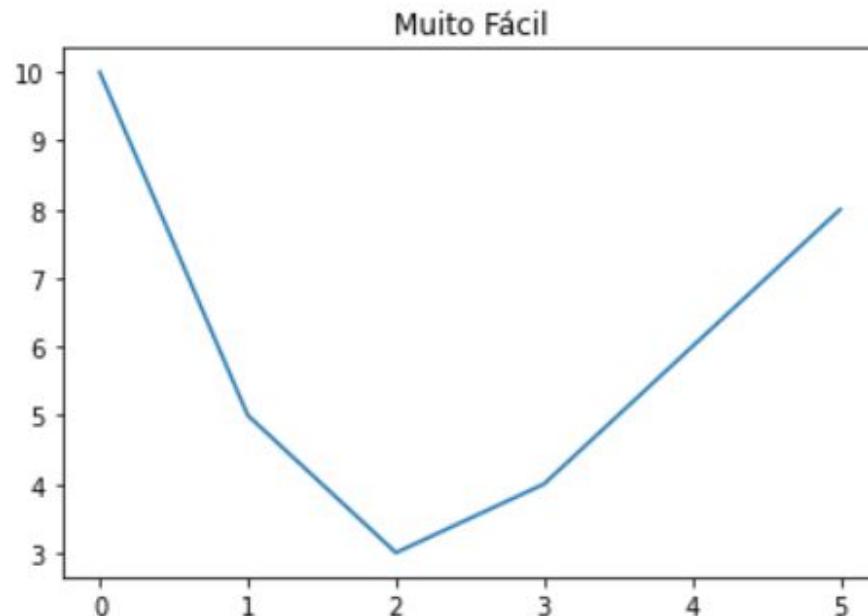
matplotlib Gráfico com Título

- Podemos criar um título para a figura usando a função `title()`:

```
import matplotlib.pyplot as plt  
  
plt.plot( [10,5,3,4,6,8] )  
plt.title("Muito Fácil")
```

No padrão baseado em MATLAB

matplotlib Gráfico com Título



matplotlib Títulos e Rótulos

- Podemos definir os títulos e rótulos dos eixos usando os métodos (padrão com POO):
 - `set_title()` - um Título é definido
 - `set_xlabel()` - uma etiqueta (ou rótulo) para o eixo X é definida
 - `set_ylabel()` - uma etiqueta (ou rótulo) para o eixo Y é definida

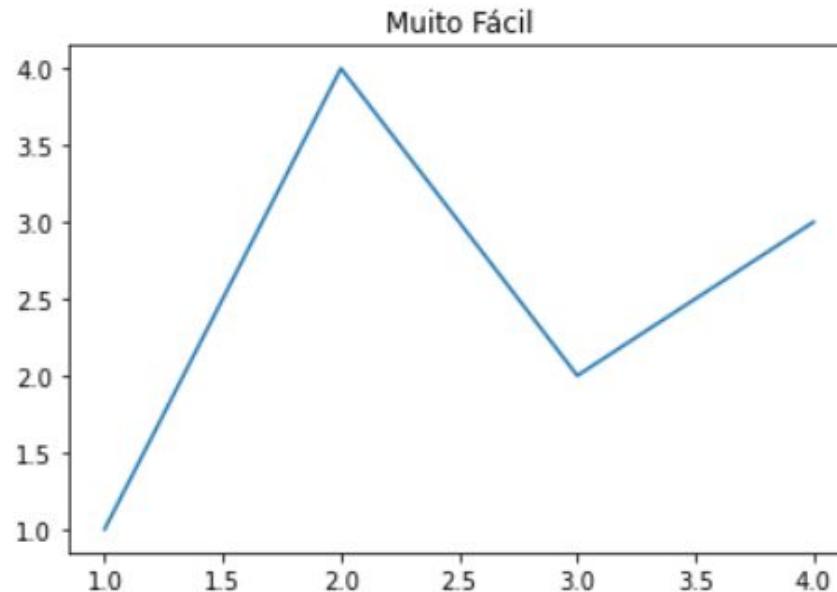
matplotlib Títulos e Rótulos

- Podemos criar um objeto gráfico usando a função `set_title()`:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
```

matplotlib Títulos e Rótulos



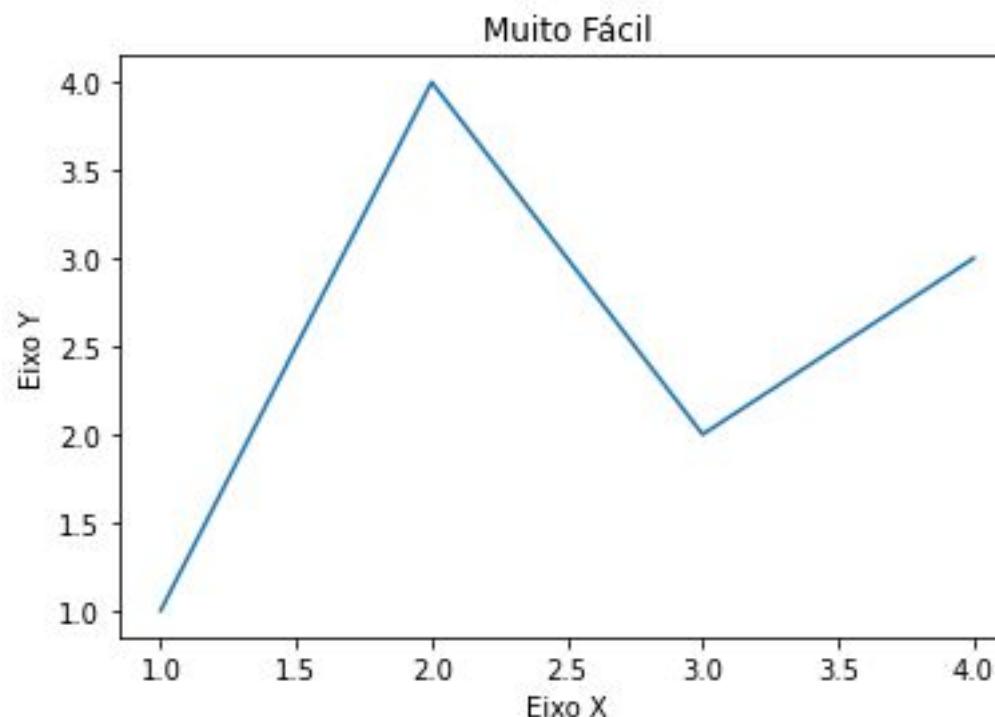
matplotlib Títulos e Rótulos

- Podemos criar um objeto gráfico usando a função `set_title()`, `set_xlabel()` e `set_ylabel()`:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```

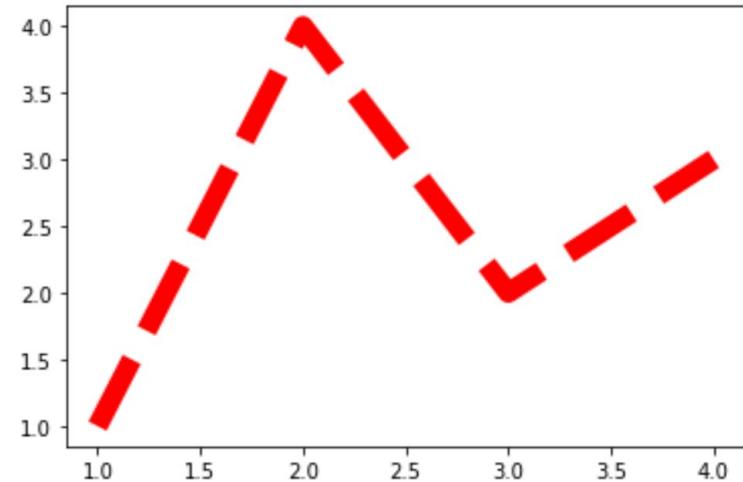
matplotlib Títulos e Rótulos



matplotlib Estilo da linha

- Podemos mudar o tipo da linha de dados passando parâmetros para a função `plot()`

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='r', linestyle='--', linewidth=10)
```

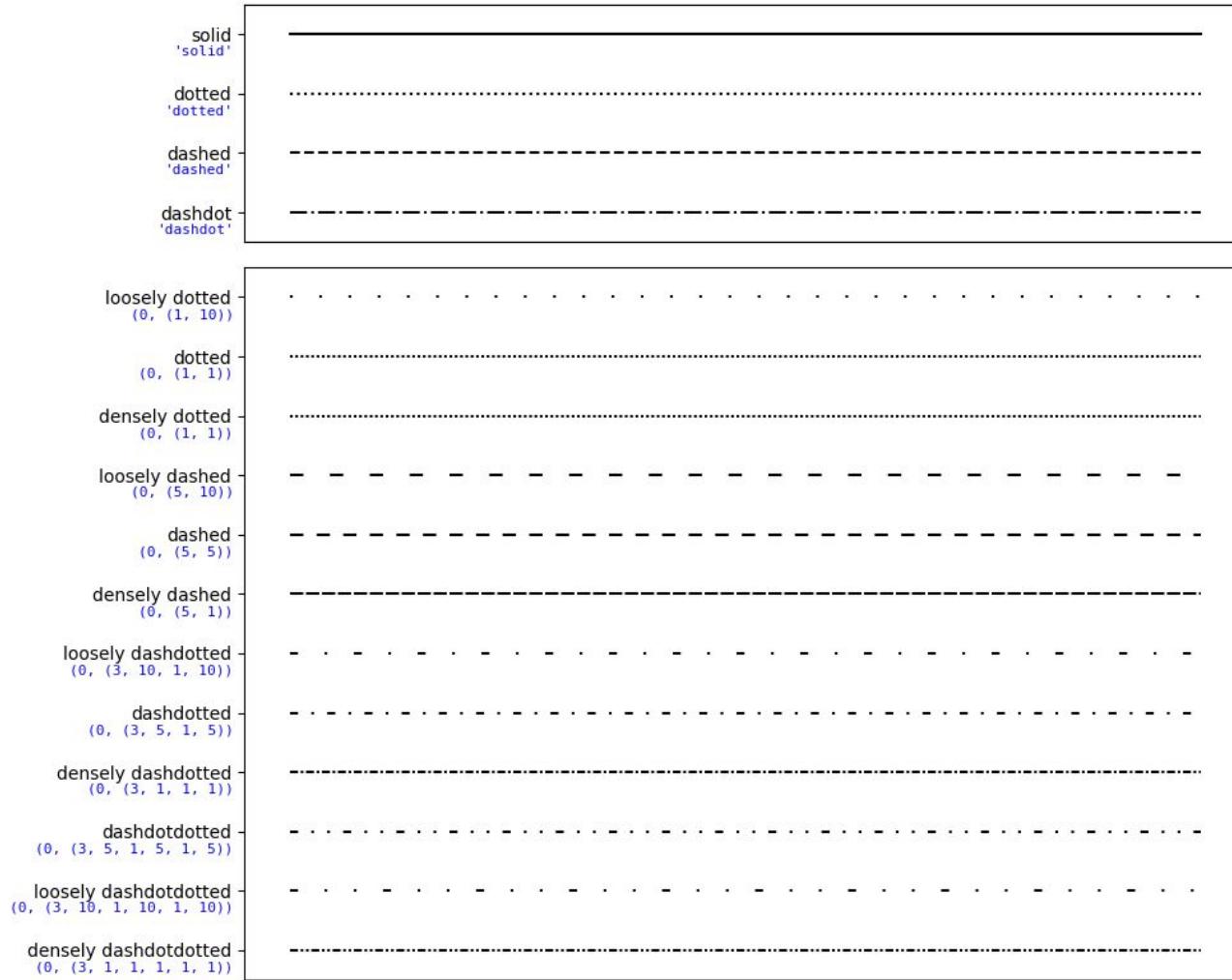


matplotlib Tipos de Linha

line styles



Tipos de Linha



matplotlib Cores

- Cores Básicas:

Base Colors



- Paleta de cores

Tableau Palette



Cores

CSS Colors

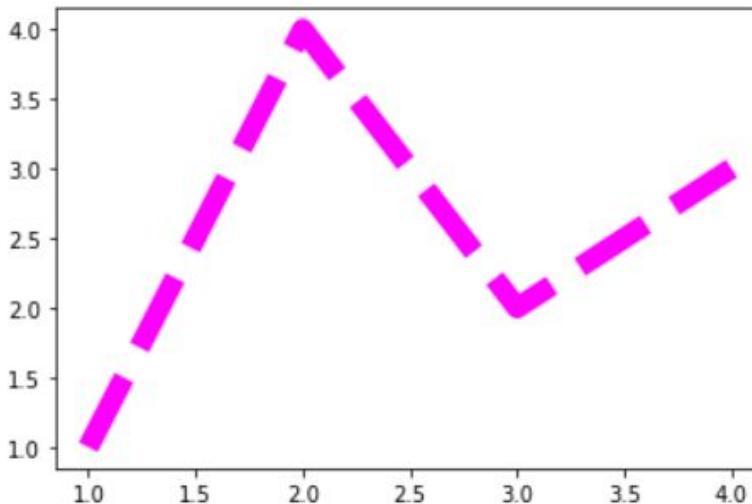
black	bisque	forestgreen	slategrey
dimgray	darkorange	limegreen	lightsteelblue
dimgrey	burlywood	darkgreen	cornflowerblue
gray	antiquewhite	green	royalblue
grey	tan	lime	ghostwhite
darkgray	navajowhite	seagreen	lavender
darkgrey	blanchedalmond	mediumseagreen	midnightblue
silver	papayawhip	springgreen	navy
lightgray	moccasin	mintcream	darkblue
lightgrey	orange	mediumspringgreen	mediumblue
gainsboro	wheat	mediumaquamarine	blue
whitesmoke	oldlace	aquamarine	slateblue
white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink



Cores - mudando a linha para ‘fuchsia’

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='fuchsia', linestyle='--', linewidth=10)
```

```
[<matplotlib.lines.Line2D at 0x17d96542730>]
```

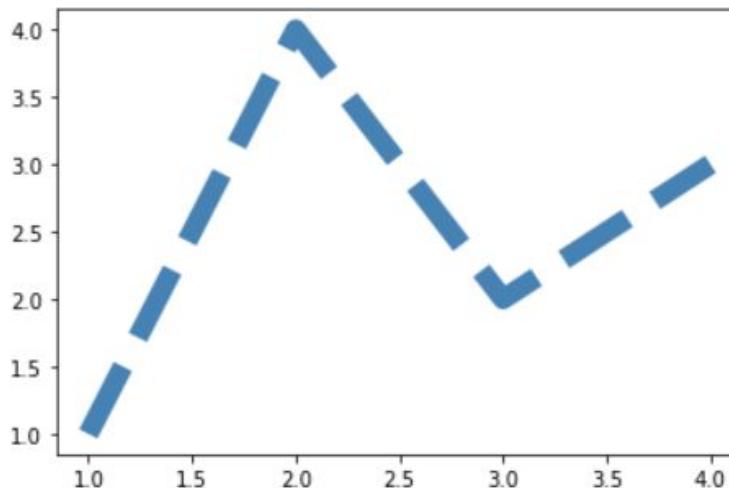




Cores - mudando a linha para 'steelblue'

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='steelblue', linestyle='--', linewidth=10)
```

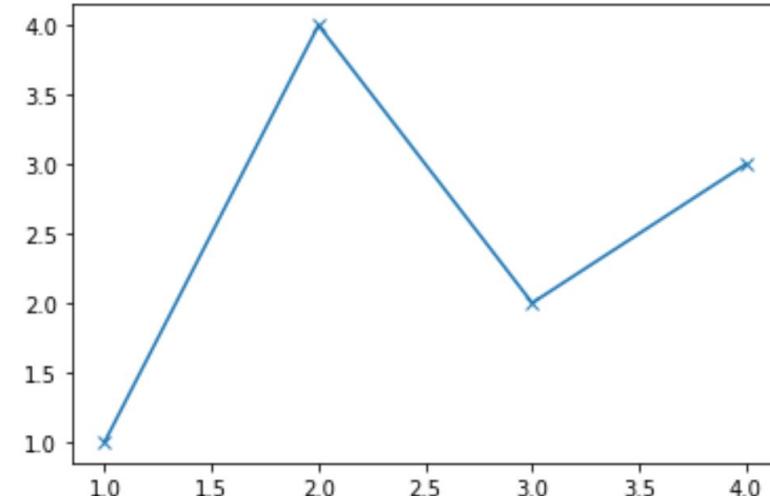
```
[<matplotlib.lines.Line2D at 0x17d9659a820>]
```



matplotlib Marcador na linha

- Podemos usar um marcador na linha de dados passando parâmetros para a função `plot()`

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
```

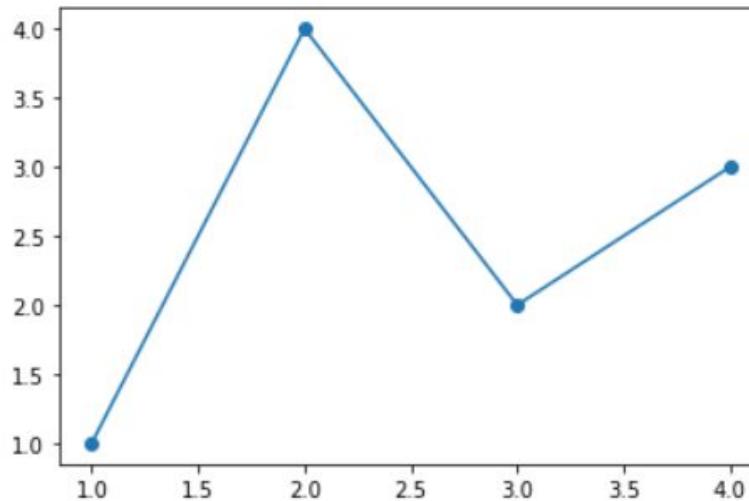


matplotlib

Marcador na linha - *marker='o'*

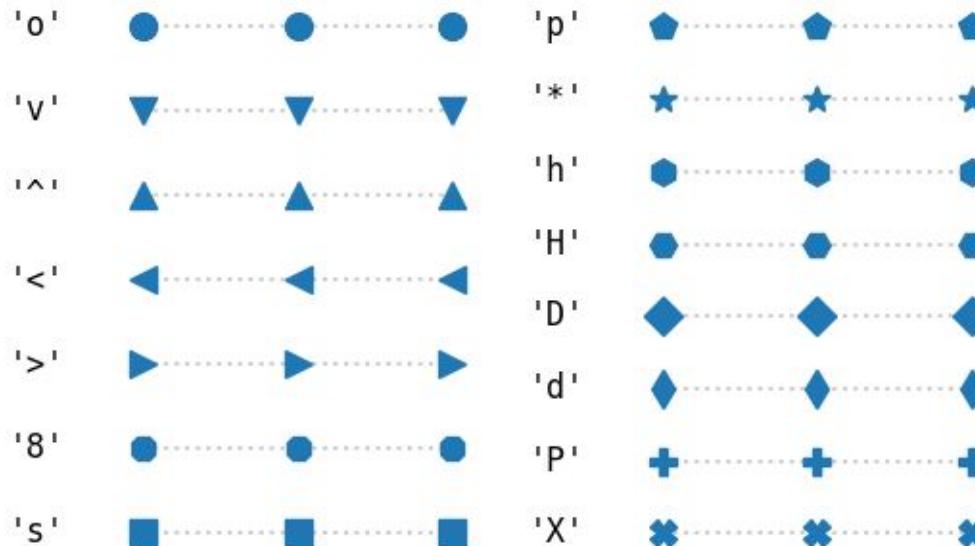
```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'o')
```

```
[<matplotlib.lines.Line2D at 0x17d966488b0>]
```



matplotlib Marcadores

Filled markers



matplotlib Marcadores

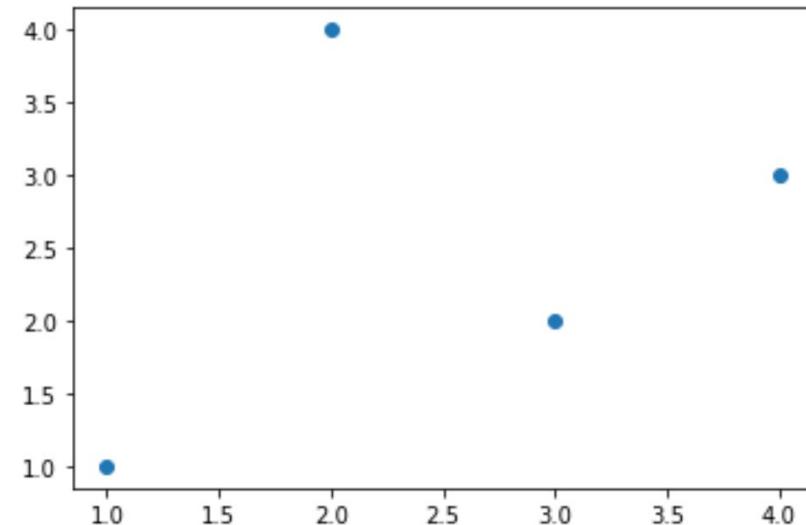
Un-filled markers

'.'	•	•	•	1	-	-	-
','	.	.	.	2			
'1'	Y	Y	Y	3	[]]
'2'	L	L	L	4	<	<	<
'3'	L	L	L	5	▶	▶	▶
'4'	Y	Y	Y	6	▲	▲	▲
'+'	+	+	+	7	▼	▼	▼
'x'	X	X	X	8	<	<	<
' '				9	▶	▶	▶
'—'	—	—	—	10	▲	▲	▲
'0'	—	—	—	11	▼	▼	▼

matplotlib Removendo a linha

- Podemos retirar a linha de dados passando parâmetros para a função `plot()`.

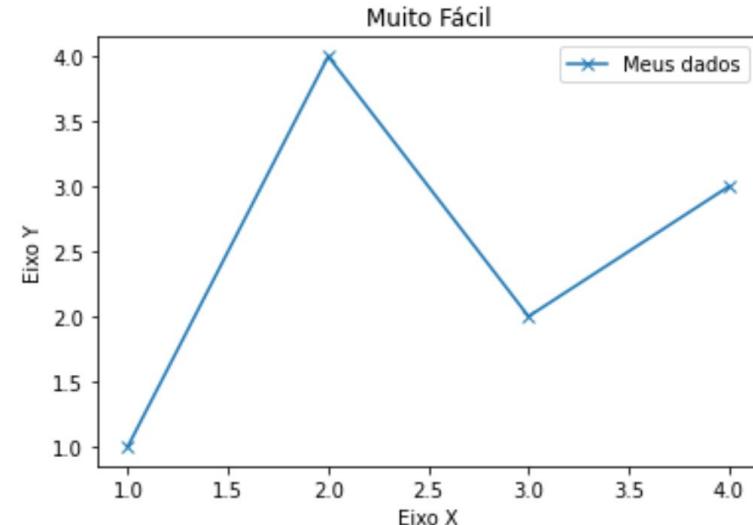
```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'o', linestyle = 'None')
```



matplotlib Legenda

- Podemos apresentar a legenda usando a função `legend()` :

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.legend(['Meus dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```



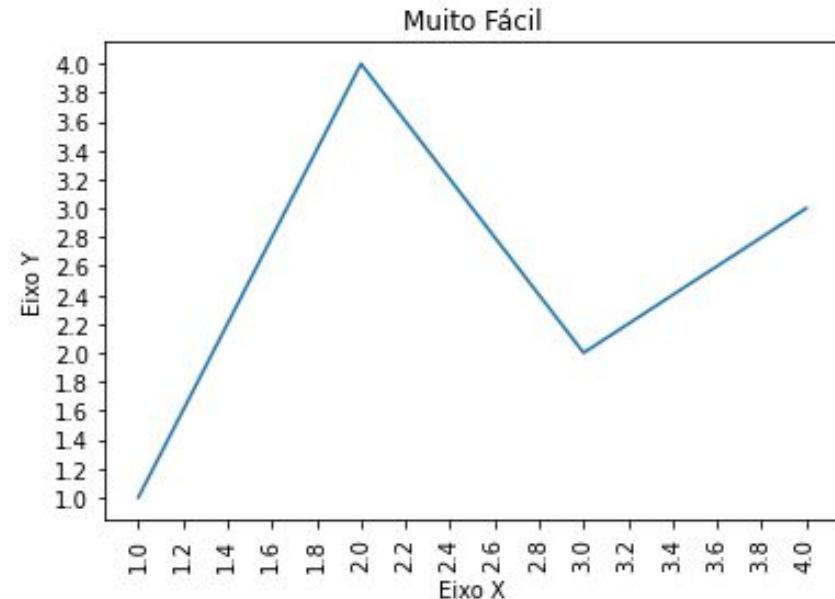
matplotlib Ticks

- Podemos mudar o espaçamento dos dados nos eixos usando a função `xticks()` e `yticks()`:

```
plt.xticks(np.arange(min(x), max(x)+0.1, 0.2), rotation=90)  
plt.yticks(np.arange(min(y), max(y)+0.1, 0.2))
```

matplotlib Ticks

```
import numpy as np
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.legend(['Meus dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
plt.xticks(np.arange(min(x), max(x)+0.1, 0.2), rotation=90)
plt.yticks(np.arange(min(y), max(y)+0.1, 0.2))
```



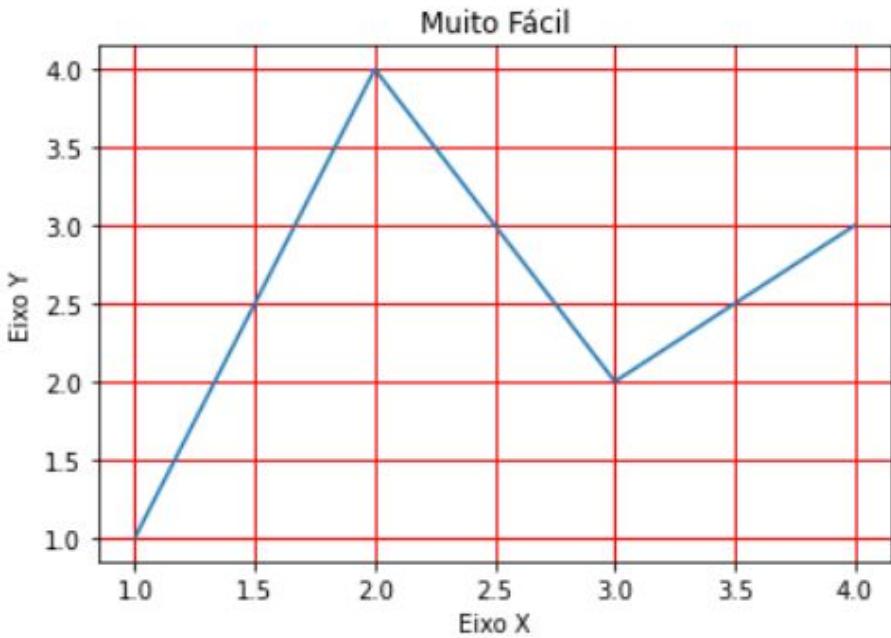
matplotlib Grids

- Podemos criar uma grade para a figura com usando a função `grid()`:

```
ax.grid(color='r', linestyle='-', linewidth=1)  
ax.grid(color='g', linestyle='--', linewidth=1)  
ax.grid(color='b', linestyle='-.', linewidth=1)  
ax.grid(color='y', linestyle=':', linewidth=1)
```

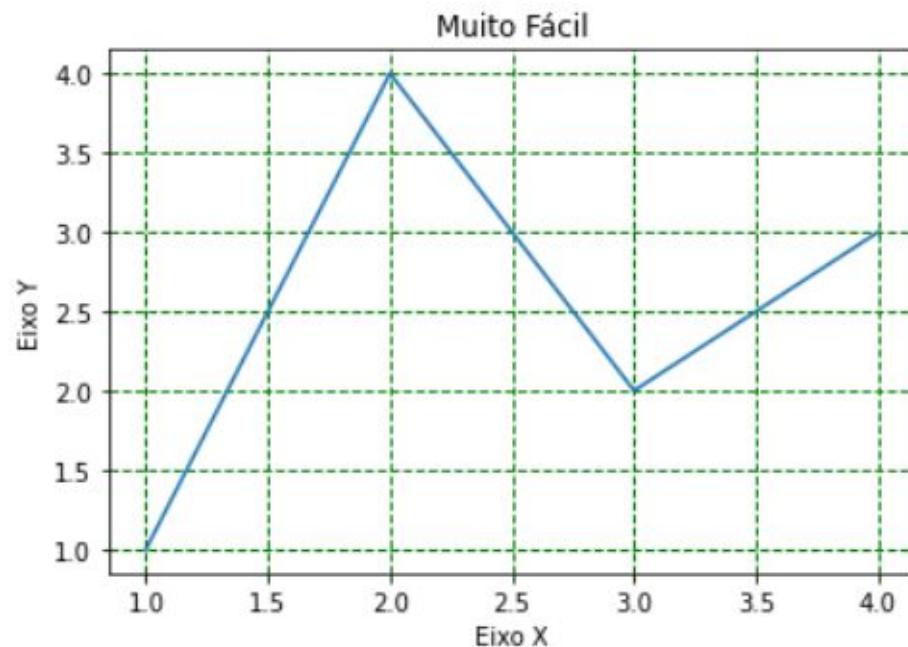
matplotlib Grids

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='r', linestyle='--', linewidth=1)
```



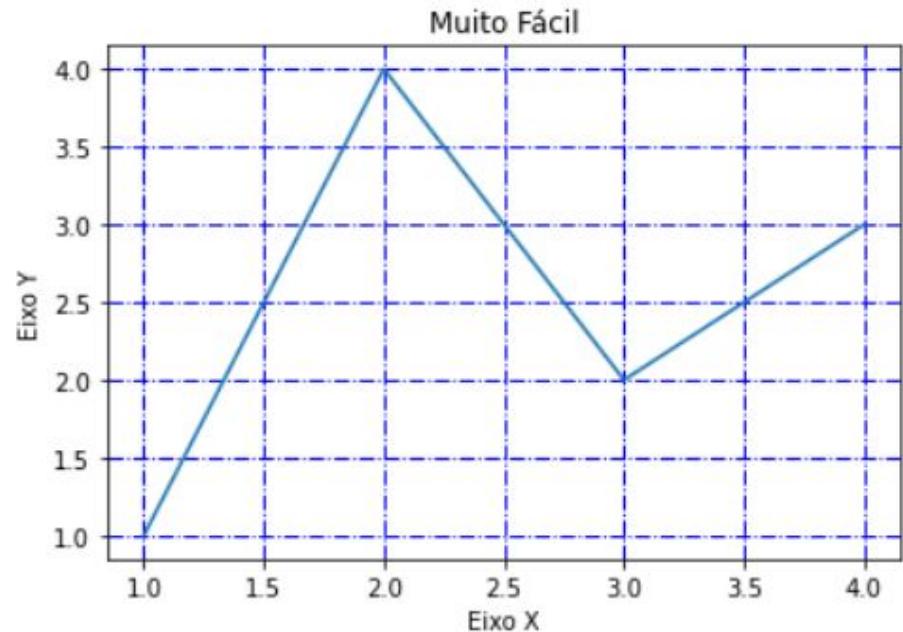
matplotlib Grids

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='g', linestyle='--', linewidth=1)
```



matplotlib Grids

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='b', linestyle='-.', linewidth=1)
```

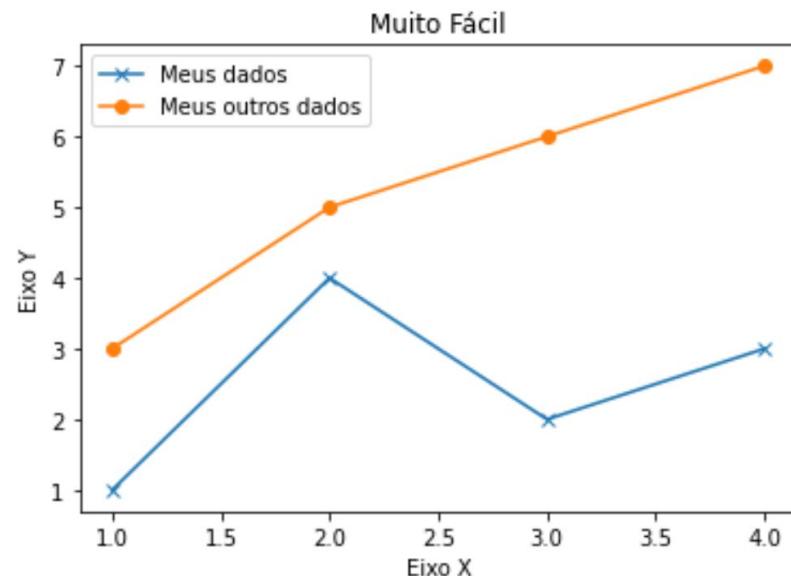




Mais de um conjunto de dados

- É possível imprimir mais de um conjunto de dados, várias linhas, no mesmo gráfico

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
z = [3, 5, 6, 7]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.plot(x, z, marker = 'o')
ax.legend(['Meus dados', 'Meus outros dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```



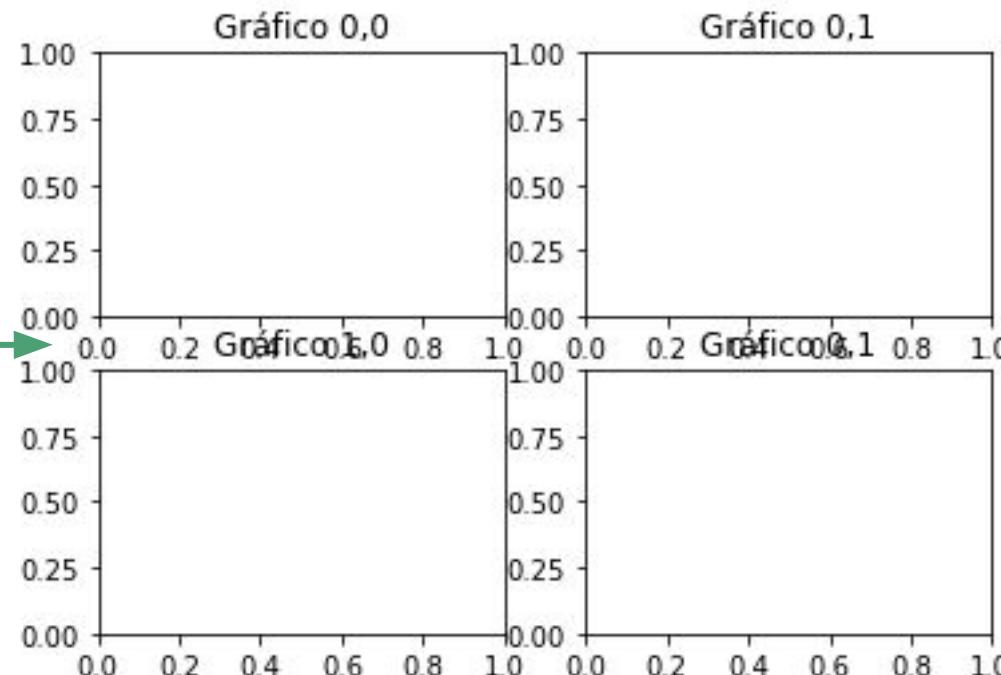
matplotlib Sub-gráficos

- Podemos criar um título para a figura com subplots usando a função `set_title()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 1,1")
```

matplotlib Sub-gráficos

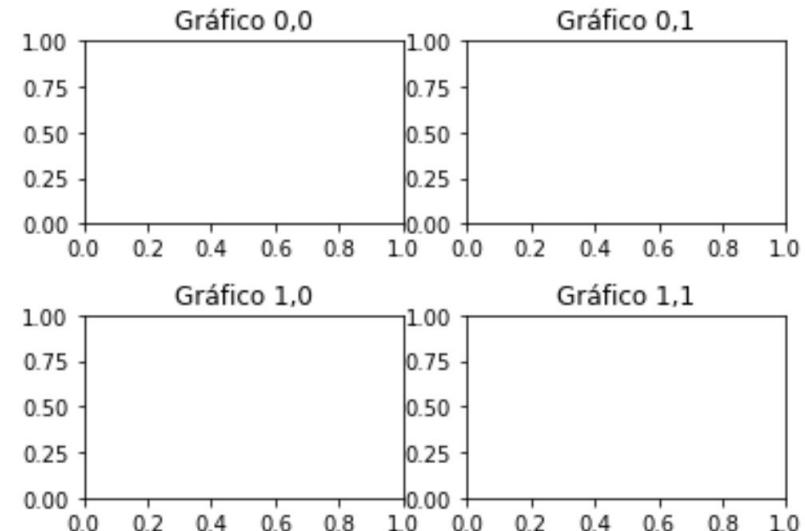
problema



matplotlib Sub-gráficos

- Para ajustar o espaço podemos usar a função `subplot_adjust()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust( hspace=0.5 )
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 1,1")
```



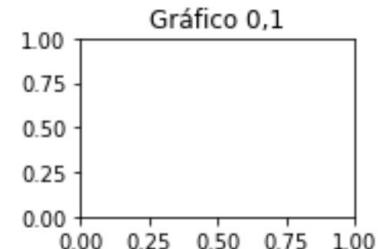
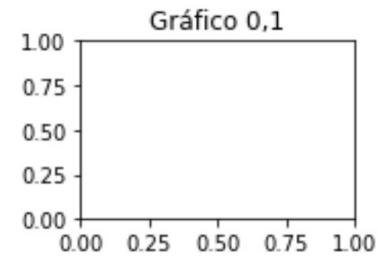
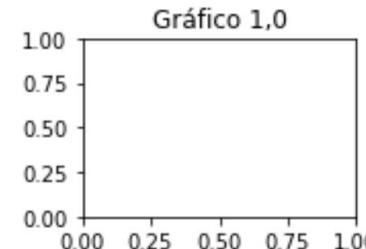
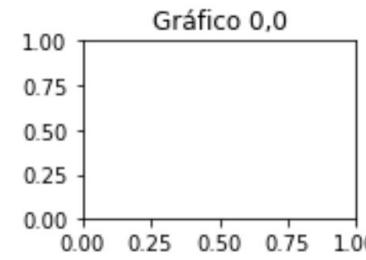
matplotlib Sub-gráficos

- `subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)`
 - *left = the left side of the subplots of the figure*
 - *right = the right side of the subplots of the figure*
 - *bottom = the bottom of the subplots of the figure*
 - *top = the top of the subplots of the figure*
 - *wspace = the amount of width for blank space between subplots*
 - *hspace = the amount of height for white space between subplots*

matplotlib Sub-gráficos

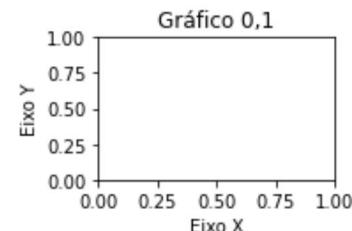
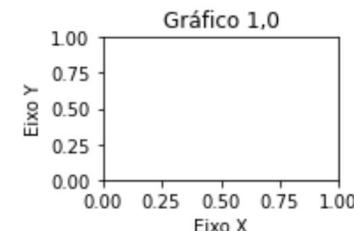
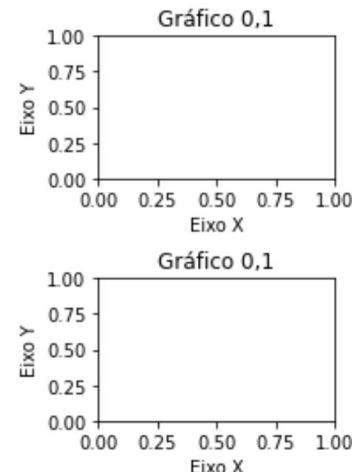
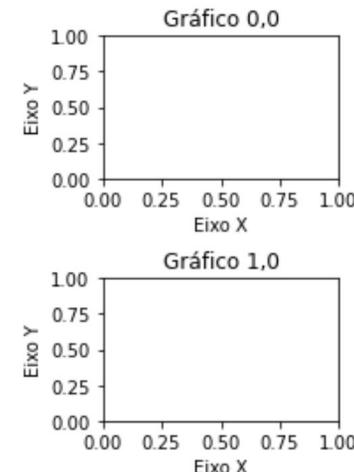
- Para ajustar o espaço podemos usar a função `subplot_adjust()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust(hspace=0.5,wspace=0.5)
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 0,1")
```



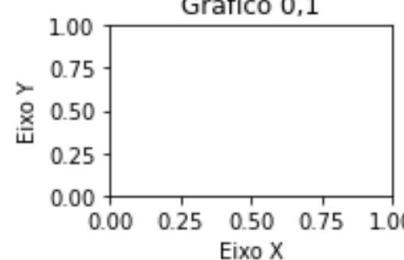
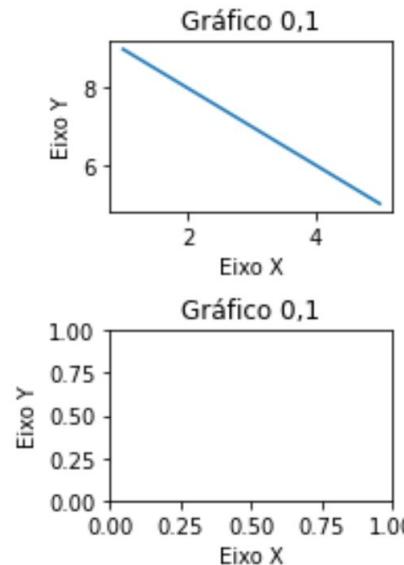
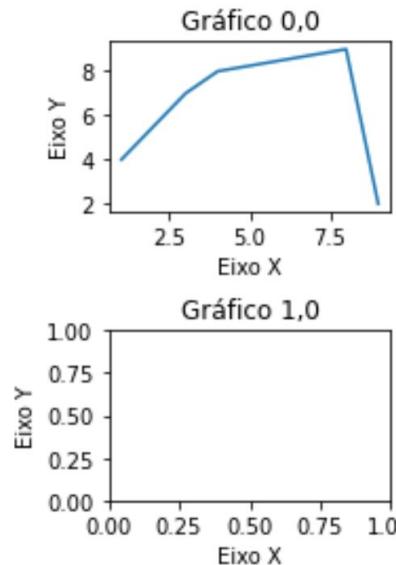
matplotlib Sub-gráficos

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust(hspace=0.7, wspace = 0.5)
axs[0,0].set_title("Gráfico 0,0")
axs[0,0].set_xlabel ("Eixo X")
axs[0,0].set_ylabel ("Eixo Y")
axs[0,1].set_title("Gráfico 0,1")
axs[0,1].set_xlabel ("Eixo X")
axs[0,1].set_ylabel ("Eixo Y")
axs[1,0].set_title("Gráfico 1,0")
axs[1,0].set_xlabel ("Eixo X")
axs[1,0].set_ylabel ("Eixo Y")
axs[1,1].set_title("Gráfico 0,1")
axs[1,1].set_xlabel ("Eixo X")
axs[1,1].set_ylabel ("Eixo Y")
```

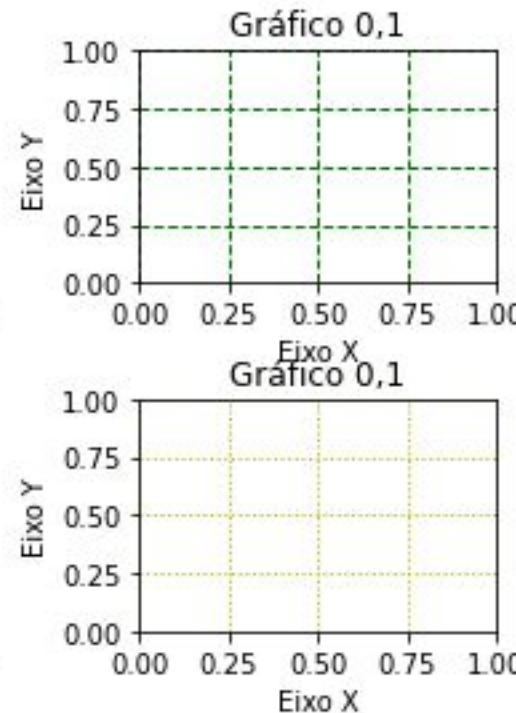
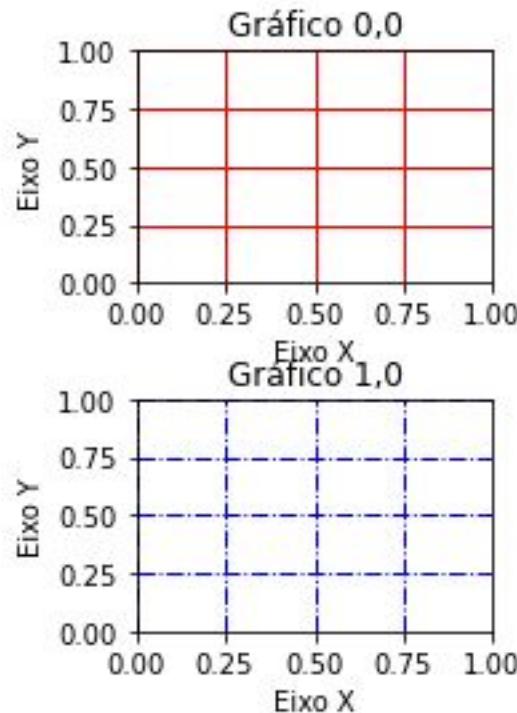


matplotlib Sub-gráficos

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust( hspace=0.7 , wspace = 0.5)
axs[0,0].set_title("Gráfico 0,0")
axs[0,0].set_xlabel ("Eixo X")
axs[0,0].set_ylabel ("Eixo Y")
axs[0,0].plot([1,3,4,8,9],[4,7,8,9,2])
axs[0,1].set_title("Gráfico 0,1")
axs[0,1].set_xlabel ("Eixo X")
axs[0,1].set_ylabel ("Eixo Y")
axs[0,1].plot([1,2,3,4,5],[9,8,7,6,5])
axs[1,0].set_title("Gráfico 1,0")
axs[1,0].set_xlabel ("Eixo X")
axs[1,0].set_ylabel ("Eixo Y")
axs[1,1].set_title("Gráfico 0,1")
axs[1,1].set_xlabel ("Eixo X")
axs[1,1].set_ylabel ("Eixo Y")
```



matplotlib Sub-gráficos



matplotlib Mais funções

- Cada função tem muitos parâmetros
- Por exemplo, ver:
 - https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.grid.html
 - https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D.set_linestyle

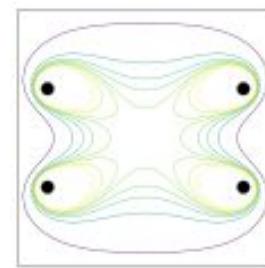
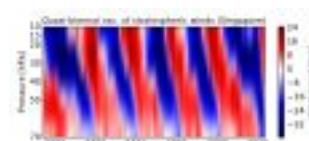
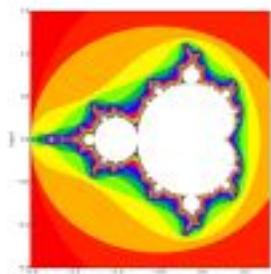
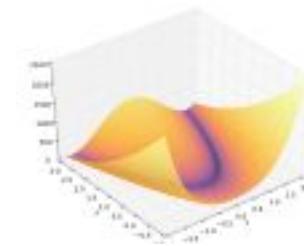
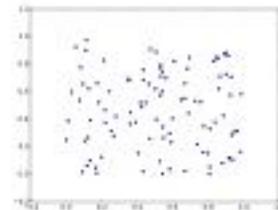
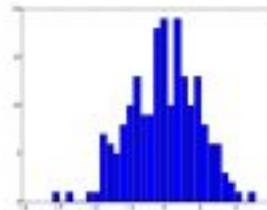
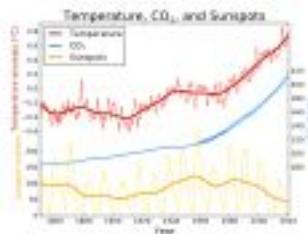
Tipos de Gráficos

matplotlib Tipos de gráficos

- A biblioteca matplotlib possui muitos tipos diferentes de gráficos:
 - Linha
 - Barra
 - Pizza
 - Histograma
 - Scatter plot
 - 3D plot
 - Image plot
 - Contour plot
 - Polar plot

matplotlib Tipos de gráficos

- A biblioteca matplotlib possui muitos tipos diferentes de gráficos:



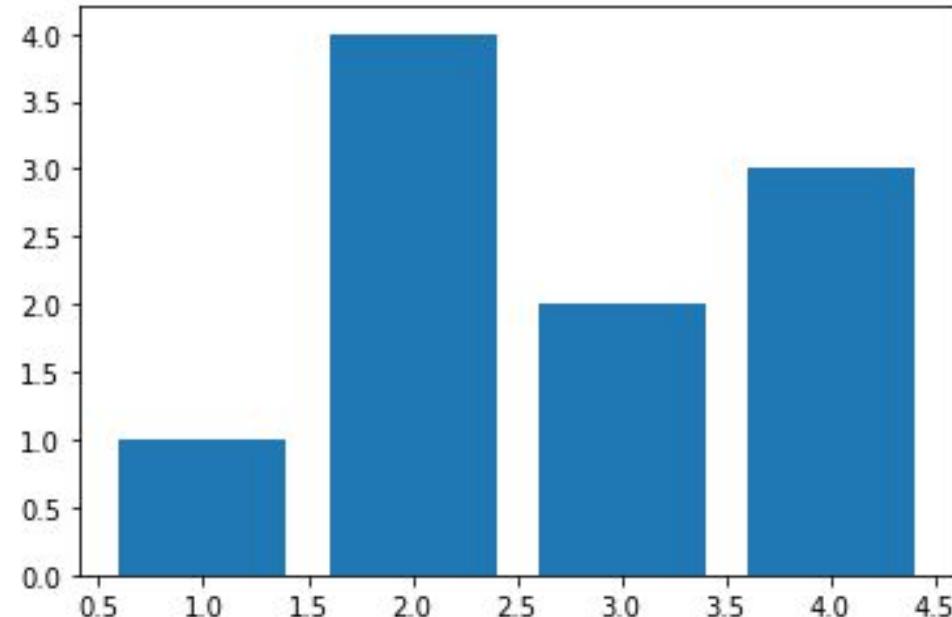
matplotlib Gráfico de barras

- **matplotlib.pyplot.bar**

- O gráfico de barras é um gráfico com barras retangulares e comprimento proporcional aos valores que ele apresenta.
- As barras são posicionadas em x com o alinhamento fornecido. Suas dimensões são dadas por largura e altura.

matplotlib Gráfico de barras

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.bar(x,y)
plt.show()
```



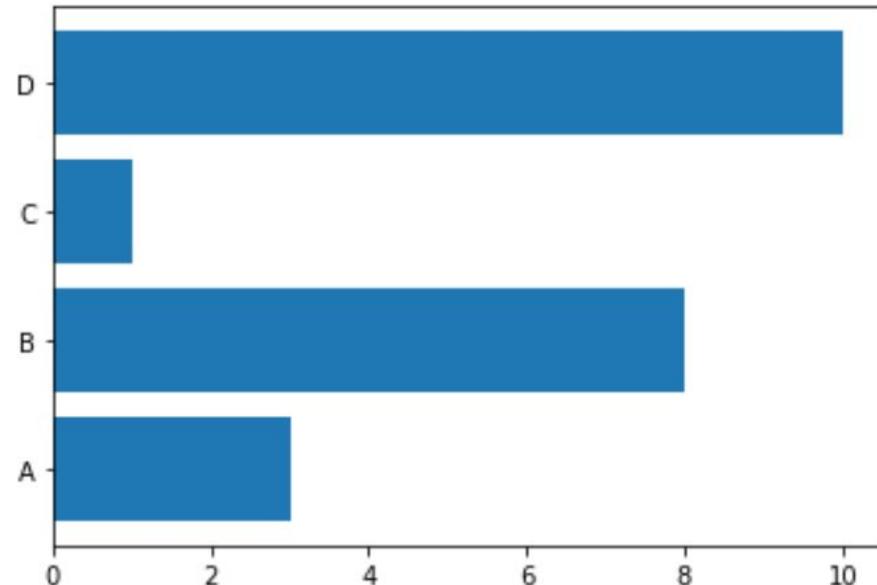
matplotlib

Gráfico de barras horizontais

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y)
plt.show()
```



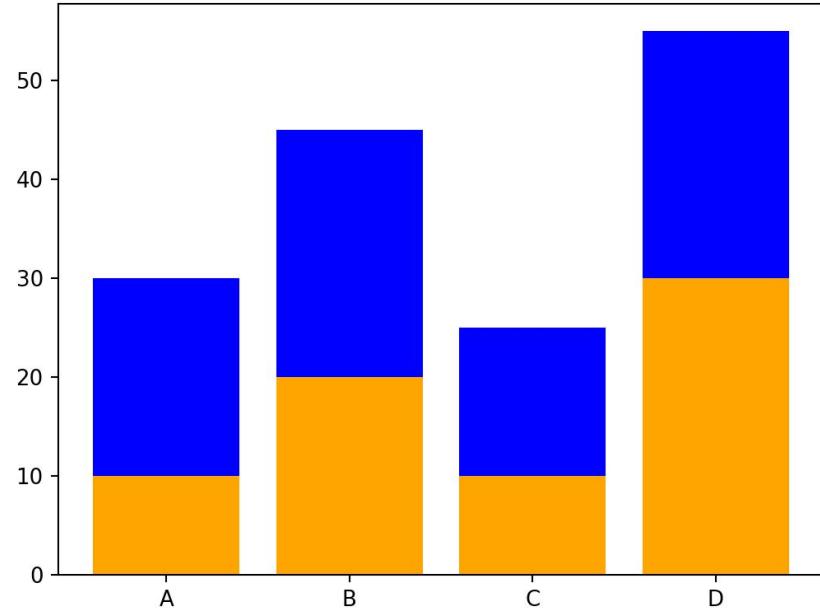
matplotlib Gráfico Stacked Bar

gráficos interativos embutidos no notebook

```
%matplotlib notebook
import matplotlib.pyplot as plt

x = ['A', 'B', 'C', 'D']
y1 = [10, 20, 10, 30]
y2 = [20, 25, 15, 25]

plt.bar(x, y1, color='orange')
plt.bar(x, y2, bottom=y1, color='blue')
plt.show()
```



matplotlib Gráfico de setores - pizza

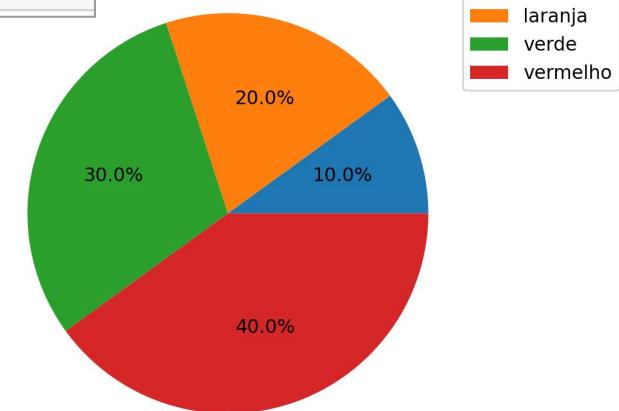
- *matplotlib.pyplot.pie*
 - Faz um gráfico de pizza de um array x
 - A área fracionada de cada fatia é dada por x/soma(x)
 - As fatias são plotadas no sentido anti-horário, por padrão começando no eixo x



Gráfico de setores - pizza

```
import numpy as np

x = np.array([1, 2, 3, 4])
fig, ax = plt.subplots()
ax.pie(x, autopct= '%.1f%%')
ax.legend(["azul", "laranja", "verde", "vermelho"], loc='upper right',
           bbox_to_anchor=(1.3, 1.))
plt.show()
```

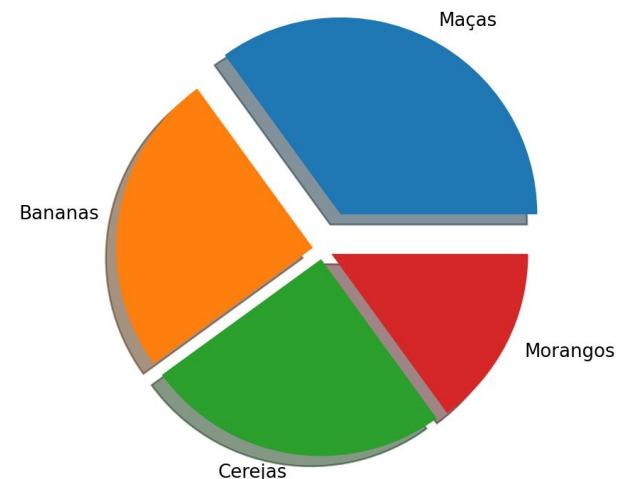


matplotlib Gráfico de setores - pizza

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Maças", "Bananas", "Cerejas", "Morangos"]
myexplode = [0.2, 0.05, 0.05, 0.05]

plt.pie(y, labels = mylabels, explode = myexplode, shadow = True)
plt.show()
```

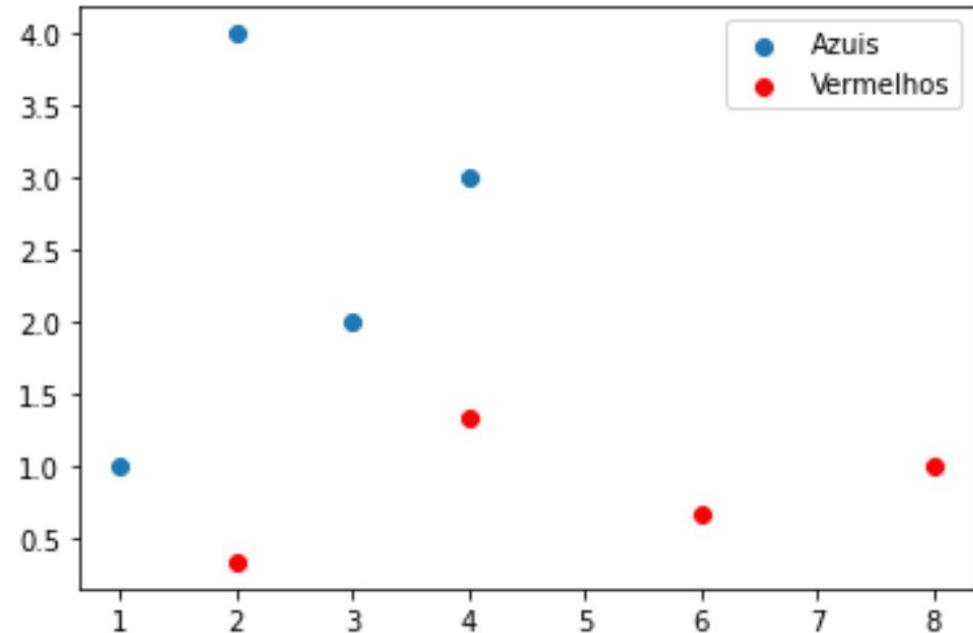


matplotlib Gráfico de dispersão

- *matplotlib.pyplot.scatter*
 - É um gráfico de dispersão de y vs. x com marcador de tamanho e/ou cor variável.

matplotlib Gráfico de dispersão

```
x = np.array([1, 2, 3, 4])
y = np.array([1, 4, 2, 3])
fig, ax = plt.subplots()
ax.scatter(x,y)
ax.scatter(x*2,y/3, color="r")
ax.legend(["Azuis", "Vermelhos"])
plt.show()
```



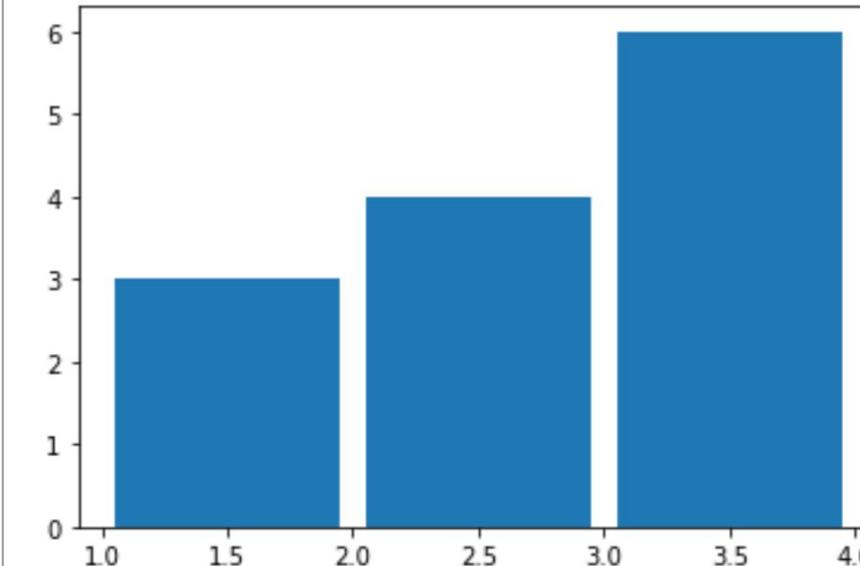
matplotlib Histograma

- *matplotlib.pyplot.hist*

- O histograma é a representação gráfica em colunas ou em barras de um conjunto de dados previamente tabulado e dividido em classes uniformes ou não uniformes
- É um gráfico que mostra a distribuição de acontecimentos registrados em todo o espectro, sendo muito importante para análise estatística.

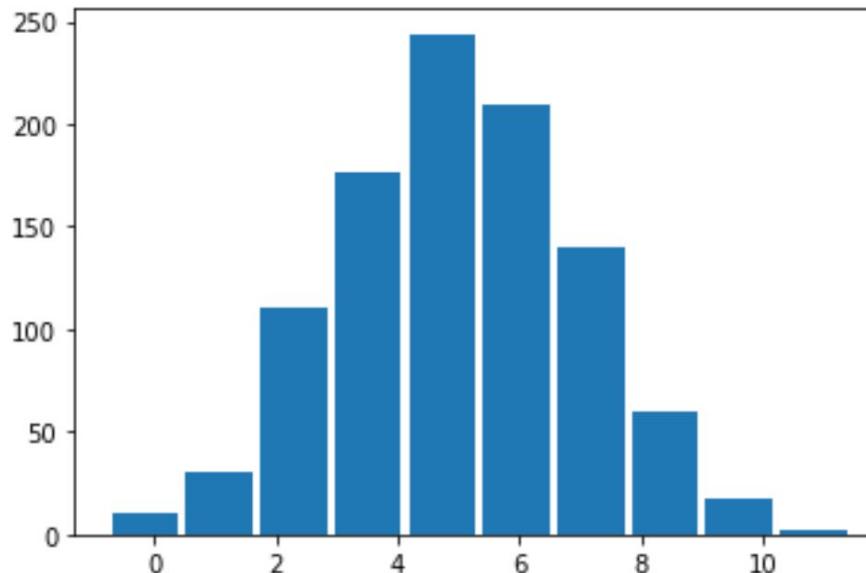
matplotlib Histograma

```
x = [1, 2, 3, 4, 3, 2, 3, 2, 1, 3, 3, 2, 1]
fig, ax = plt.subplots()
ax.hist(x, bins=3, rwidth=0.9)
plt.show()
```



matplotlib Histograma

```
x = np.random.normal(loc=5, scale=2, size=1000)
fig, ax = plt.subplots()
ax.hist(x, bins=10, rwidth=0.9)
plt.show()
```



matplotlib Animação

Vamos imprimir o gráfico de uma gaussiana que se desloca no eixo x

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},$$

Para fazer animações simples, podemos usar o método
`plt.pause(0.1)`

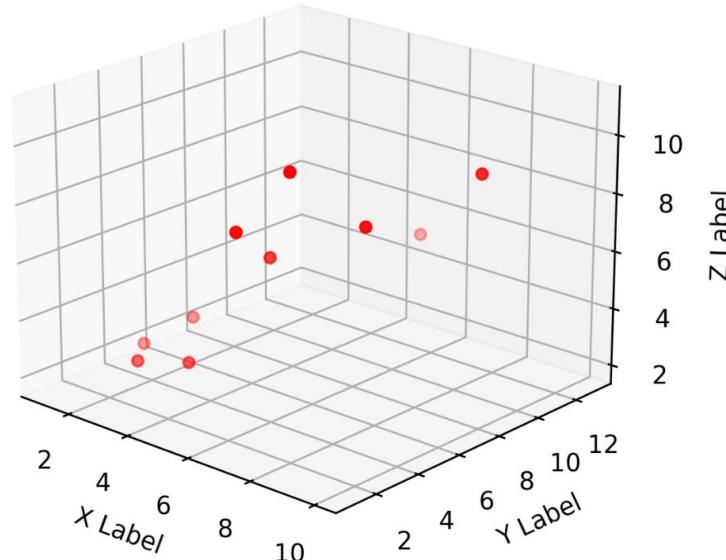
matplotlib 3D

- *mpl_toolkits.mplot3d*
 - Toolkit para desenhar gráficos em 3D
 - `add_subplot(projection='3d')`

matplotlib 3D

```
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x =[1,2,3,4,5,6,7,8,9,10]
y =[5,6,2,3,13,4,1,2,4,8]
z =[2,3,3,3,5,7,9,11,9,10]
ax.scatter(x, y, z, c='r', marker='o')

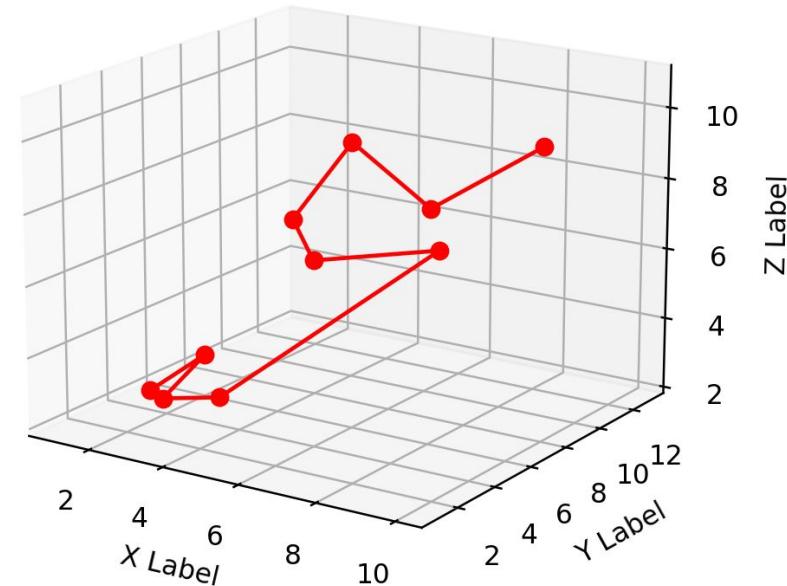
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
```



matplotlib 3D

```
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x =[1,2,3,4,5,6,7,8,9,10]
y =[5,6,2,3,13,4,1,2,4,8]
z =[2,3,3,3,5,7,9,11,9,10]
ax.plot(x, y, z, c='r', marker='o')

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
```



matplotlib 3D

```
import matplotlib.pyplot as plt
import numpy as np

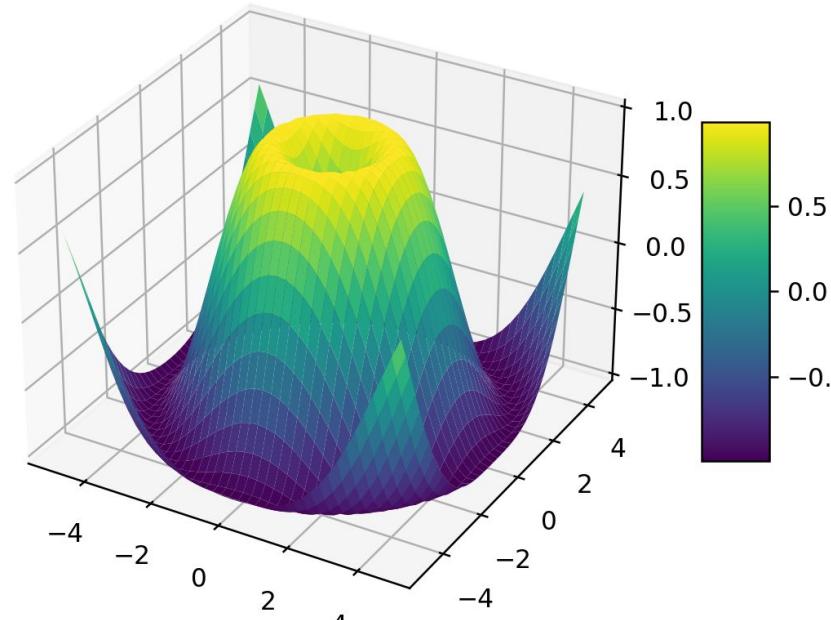
fig, ax = plt.subplots()
ax = fig.add_subplot(projection='3d')

X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

surf = ax.plot_surface(X, Y, Z, cmap='viridis',
                      linewidth=0)

ax.set_zlim(-1.01, 1.01)
fig.colorbar(surf, shrink=0.5, aspect=5)

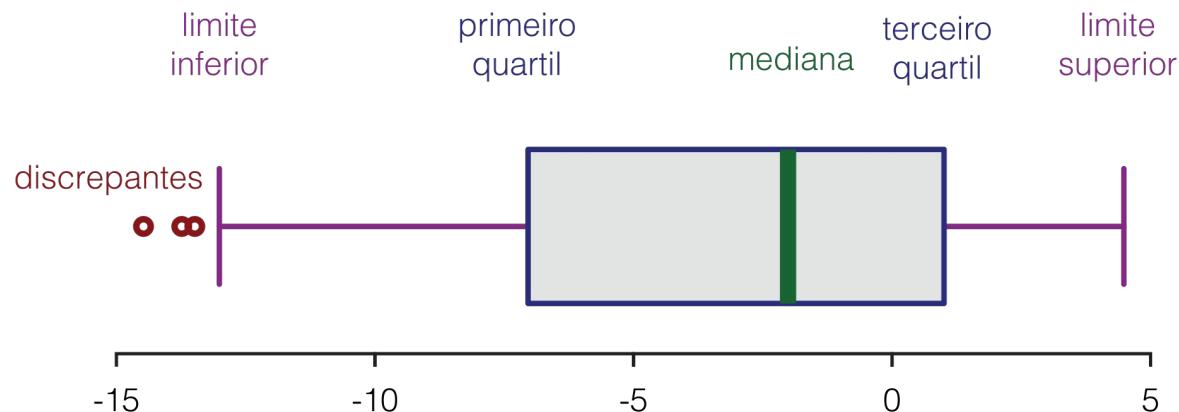
plt.show()
```



matplotlib Boxplot

- `matplotlib.pyplot.boxplot`

- É um gráfico para representar a variação de dados observados de uma variável numérica por meio de *quartis*.



matplotlib Boxplot

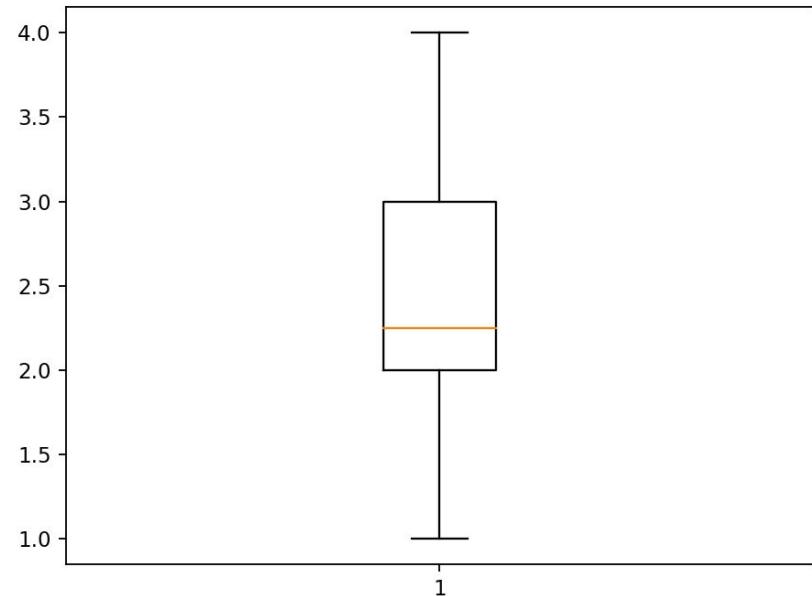
- A mediana divide o conjunto em 2 partes iguais
- Os quartis dividem em 4 partes iguais
- Os decis em 10 partes iguais e
- Os centis em 100 partes iguais

O objetivo das separatrizes é proporcionar uma melhor idéia da dispersão do conjunto, principalmente da simetria ou assimetria da distribuição

matplotlib Boxplot

```
x = [1, 2, 3, 4, 3, 2, 2.5, 3, 2, 1, 3, 3, 2, 1]
x.sort()
print(x)
fig, ax = plt.subplots()
ax.boxplot(x)
plt.show()
```

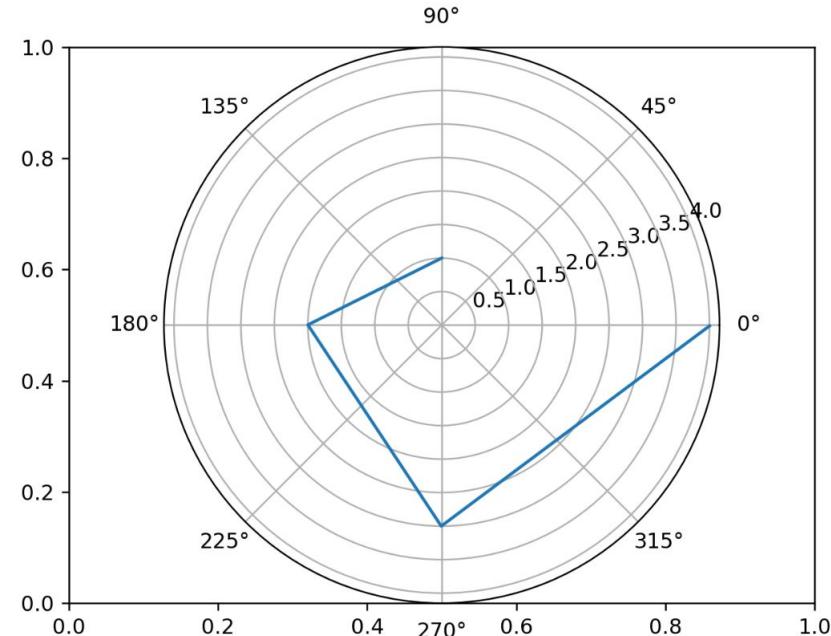
```
[1, 1, 1, 2, 2, 2, 2.5, 3, 3, 3, 3, 3, 4]
```



matplotlib Polar

- *matplotlib.pyplot.polar*

```
r = [1, 2, 3, 4]
t = [1.57, 3.14, 4.71, 6.28]
fig, ax = plt.subplots()
plt.polar(t,r)
plt.show()
```



Exercício

Faço o gráfico da função abaixo para x no intervalo [0.5, 6.2]

$$f(x) = -2x^5 + 20x^4 - 52x^3 - 50$$



seaborn



seaborn

- Seaborn é uma biblioteca de visualização de dados Python baseada em Matplotlib
- Ele fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos
- Funciona muito bem com o **Pandas (próxima aula)**



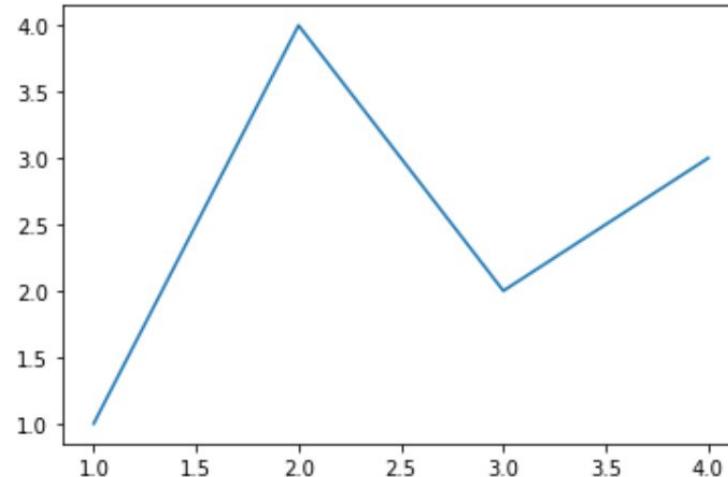
seaborn Linha

gráficos estáticos embutidos no notebook

```
%matplotlib inline
import seaborn as sns

x_data = [1, 2, 3, 4]
y_data = [1, 4, 2, 3]

sns.lineplot(x=x_data, y=y_data)
plt.show()
```

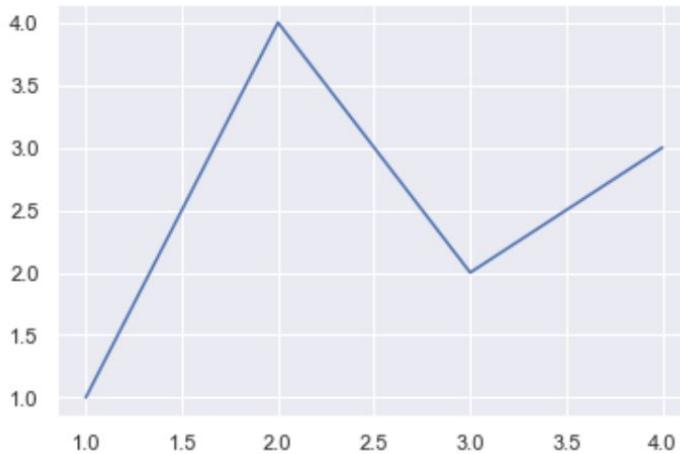




seaborn Linha

gráficos estáticos embutidos no notebook

```
%matplotlib inline  
import seaborn as sns  
  
sns.set()  
x_data = [1, 2, 3, 4]  
y_data = [1, 4, 2, 3]  
  
sns.lineplot(x=x_data, y=y_data)  
plt.show()
```

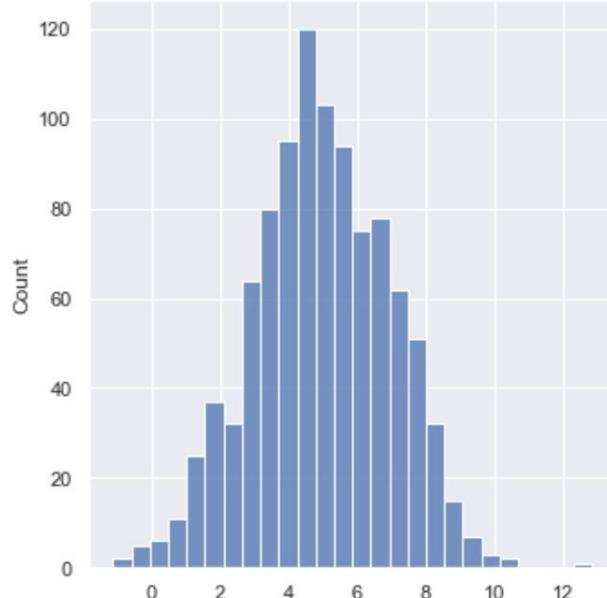


Podemos sobreescrver os parâmetros do matplotlib para produzir uma saída visualmente superior



seaborn Histograma

```
x = np.random.normal(loc=5, scale=2, size=1000)  
  
sns.displot(x)  
  
<seaborn.axisgrid.FacetGrid at 0x1e5ad06d6a0>
```



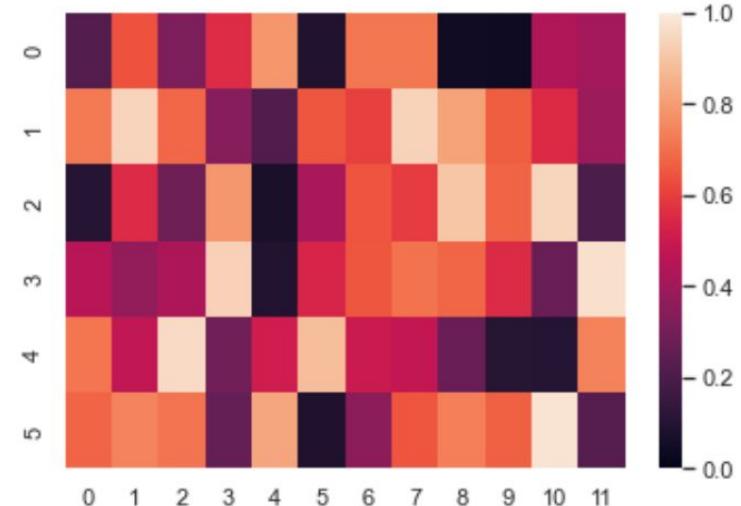


seaborn HeatMap

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()
uniform_data = np.random.rand(6, 12)
print(uniform_data)
ax = sns.heatmap(uniform_data, vmin=0, vmax=1)
plt.show()
```

```
[[0.21509936 0.63692204 0.31949472 0.55651947 0.78250354 0.08852977
 0.71871369 0.71526784 0.04310999 0.0384657 0.43308995 0.40446546]
[0.71886999 0.93999975 0.68240243 0.34011463 0.21022424 0.64632981
 0.60123802 0.93515354 0.81474037 0.66357884 0.54797006 0.38945483]
[0.09948031 0.55151239 0.2779765 0.78876252 0.06692622 0.41769458
 0.64134011 0.59144708 0.89926969 0.67222694 0.94509359 0.19838676]
[0.45684052 0.36742959 0.42720872 0.93160699 0.09280038 0.5382347
 0.64834425 0.70489772 0.67951986 0.55132222 0.27026409 0.97192904]
[0.71179309 0.47401588 0.95523273 0.28599871 0.50816308 0.88005888
 0.4983702 0.47899952 0.27232627 0.10354146 0.0988185 0.73934867]
[0.67505499 0.74252355 0.70807812 0.26150803 0.8227762 0.08568798
 0.35119713 0.6472129 0.73419945 0.66437395 0.97931446 0.22629551]]
```





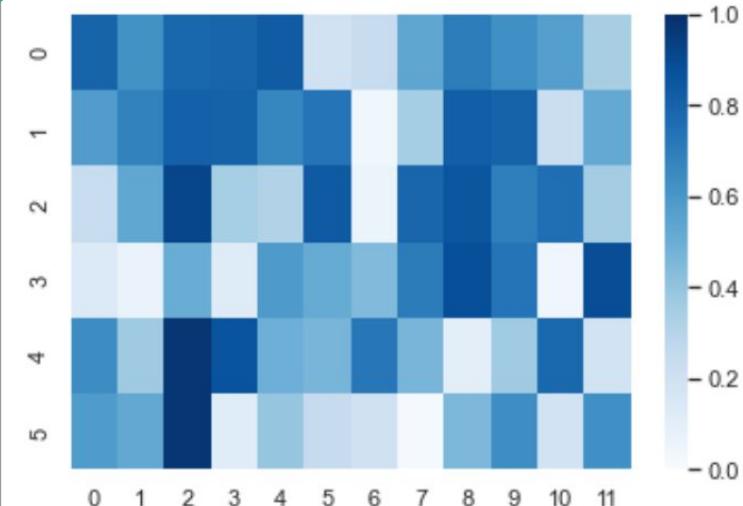
seaborn HeatMap

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()
uniform_data = np.random.rand(6, 12)
print(uniform_data)
ax = sns.heatmap(uniform_data, vmin=0, vmax=1, cmap='Blues')
plt.show()
```

```
[[0.80023948 0.62440821 0.78776169 0.79317107 0.83484808 0.19237352
 0.24199976 0.53747919 0.70430773 0.63032004 0.56926131 0.34393001]
[0.57941323 0.68721016 0.81071854 0.80853406 0.67103914 0.73506462
 0.04136299 0.35374826 0.82012017 0.80396793 0.22284567 0.52084647]
[0.23514158 0.53370351 0.9160807 0.34768838 0.31629655 0.83626097
 0.05756296 0.78951507 0.8550148 0.69652576 0.75645849 0.3587652 ]
[0.1405345 0.06572181 0.50637929 0.11750004 0.58619647 0.51277367
 0.44762787 0.70835383 0.8816872 0.73764601 0.03903143 0.88875385]
[0.64305056 0.37246098 0.97747785 0.86647103 0.49370042 0.46549621
 0.72269851 0.46726261 0.10469507 0.36742802 0.78311787 0.18883762]
[0.58361196 0.52517752 0.97525232 0.11345287 0.39767389 0.2475025
 0.20199952 0.01946687 0.45646195 0.6392743 0.19007077 0.63034253]]
```

colormap: altera o padrão de cor





seaborn HeatMap

- Exemplo: gerar o HeatMap para demonstrar qual é o país que mais contribui com emissões de CO₂
- Dataset: ***global_warming.csv***
 - *O dataset tem 15 linhas (países) e 19 colunas (anos)*
 - *Dataset bruto: não está pronto para criar mapa de calor, pois o mapa de calor precisa de dados numéricos 2D*
 - *Vamos ter que tratar o dataset*





- Plotly é uma biblioteca de visualização de dados para Python
- São vários produtos: criação de dashboards até clientes SQL.
- Plotly permite que você utilize seus gráficos em aplicações e, claro, Jupyter Notebooks.



- Dataset **IRIS**
- O conjunto de dados de Flores de Íris (Iris Flower), ou Fisher's Iris, é um conjunto de dados multivariado, introduzido por Sir Ronald Aylmer Fisher (1936) como um exemplo de análise discriminante
- Conjunto finito de espécies de Iris = Virginica, Setosa e Versicolor
- Para identificar a diferença entre as espécies, são consideradas as seguintes informações: Comprimento da sépala e da pétala / Largura da sépala e da pétala



Iris Versicolor



Iris Setosa



Iris Virginica



- Fazer gráfico 3D com as seguintes informações:
 - sepal.length
 - sepal.width
 - petal.length

Entrega

Earthquake dataset

- Tratar e visualizar informações relevantes provenientes do dataset earthquake (terremotos registrados de 1965 a 2016)

<https://www.kaggle.com/datasets/usgs/earthquake-database>

- Você deve apresentar **pelo menos 6 gráficos** da sua preferência e as respectivas análises em texto
- 1 gráfico é obrigatório: exibir a **localização de cada terremoto no mapa mundi** com a sua respectiva magnitude

<https://plotly.com/python/map-configuration/>

<https://plotly.com/python/bubble-maps/>