Python para Ciência de Dados

Matplotlib Básico

Luiz Alberto

Ciência da Computação

May 14, 2019

NumPy

Definição:

Biblioteca para computação científica. Implementa arrays muldimensionais e permite a fácil execução de operações matemáticas (`www.numpy.org`).

- Numeric Python
- Alternativa à Lista em Python: NumPy Array
- Cálculos sobre matrizes inteiras (broacasting)
- Fácil e rápido

NumPy

```
1  import numpy as np
2  alturas = [ 1.73, 1.68, 1.71, 1.89, 1.79 ]
3  pesos = [ 65.4, 59.2, 63.6, 88.4, 68.7 ]
4
5  np_alturas = np.array(alturas)
6  np_pesos = np.array(pesos)
7
8  imcs = np_pesos / np_alturas ** 2
9  print(imcs)
10 # output:
11 # [21.85171573 20.97505669 21.75028214 24.7473475 21.44127836]
```

Comparação com listas

```
1 import numpy as np
2 alturas = [ 1.73, 1.68, 1.71, 1.89, 1.79 ]
3 pesos = [ 65.4, 59.2, 63.6, 88.4, 68.7 ]
4
5 print(alturas/pesos)
6 # output:
7 # TypeError: unsupported operand type(s) for /: 'list' and '
    list'
```

Subsetting

```
1 import numpy as np
2 imcs
3 # array([21.85171573, 20.97505669, 21.75028214, 24.7473475,
      21.44127836])
4 imcs[1]
5 # 21.85171573
6 imcs > 23
7 # array([False, False, False, True, False])
8 imcs[imcs > 23]
9 # array([24.7473475])
```

Arrays multidimensionais

```
1
2 import numpy as np
3 medidas = np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
4                    [65.4, 59.2, 63.6, 88.4, 68.7]])
5
6 print(medidas.shape) # retorna as dimensoes do array
7 # (2, 5)
```

Fatiamento (1)

```
1
2  import numpy as np
3  matriz = np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
4                     [65.4, 59.2, 63.6, 88.4, 68.7]])
5
6  matriz[0]
7  # array([1.73, 1.68, 1.71, 1.89, 1.79])
8
9  matriz[0][2]
10 # 1.71
11
12 matriz[0, 2]
13 # 1.71
14
15 matriz[:, 1:3]
16 # array([[ 1.68,  1.71],
17 #        [59.2 , 63.6]])
18
19 matriz[1,: ]
```

Fatiamento (2)

```
20 # [65.4, 59.2, 63.6, 88.4, 68.7]
```

Estatística básica

```
1
2  import numpy as np
3  array = np.array([[0.173, 0.168, 0.171, 0.189, 0.179],
4                    [0.154, 0.259, 0.163, 0.388, 0.287]])
5
6  print(np.mean(array[1,: ]))
7  # 0.2502
8
9  print(np.median(array[1,: ]))
10 # 0.259
11
12 print(np.corrcoef(array[0, :] , array[1, :]))
13 # [[1.  0.79684]
14 #  [0.79684 1.]]
15
16 print(np.std(array[:, 0]))
17 # 0.0094
```

Geração de dados

```
1
2  import numpy as np
3  alturas = np.round(np.random.normal(1.75, 0.20, 5000), 2)
4  pesos = np.round(np.random.normal(60.32, 15, 5000), 2)
5
6  medidas = np.column_stack((alturas, pesos))
7  print(medidas)
8  # output:
9  #    [[1.73   65.9]
10 #     [2.03   69.91]
11 #     [1.48   63.18]
12 # ...
13 #     [2.     52.73]
14 #     [1.97   27.79]
15 #     [1.89   44.29]]
```

Matplotlib

Definição:

Biblioteca python para plotagem de gráficos 2D (incluindo 3D) (`www.matplotlib.org`).

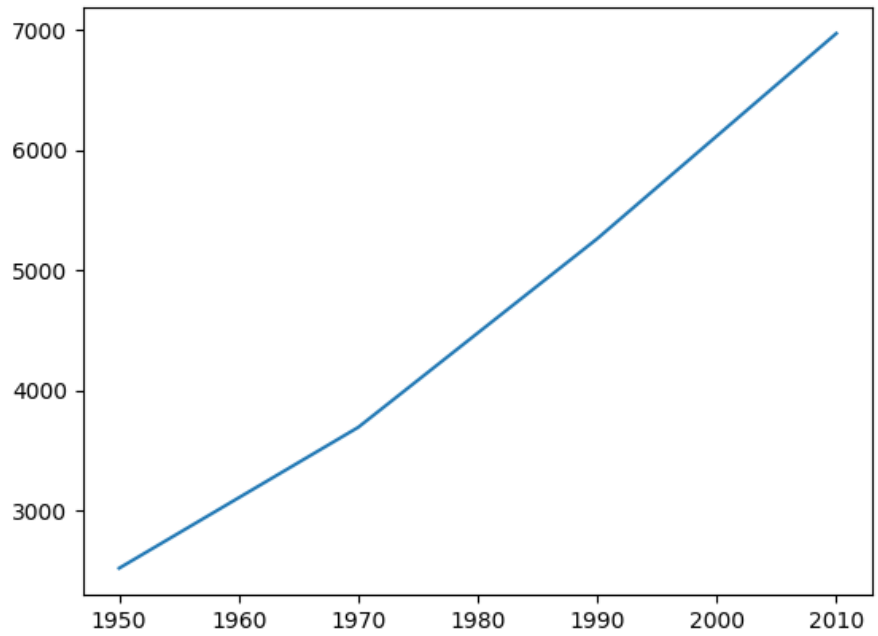
- Simplicidade de utilização
- Desenvolvimento gradual e interativo
- Grande controle sobre os elementos gráficos
- Exportação em formatos PNG, PDF, SVG e EPS

Vizualização de Dados

- Muito importante na visualização de dados
  - Explorar os dados
  - Apresentar "insights"

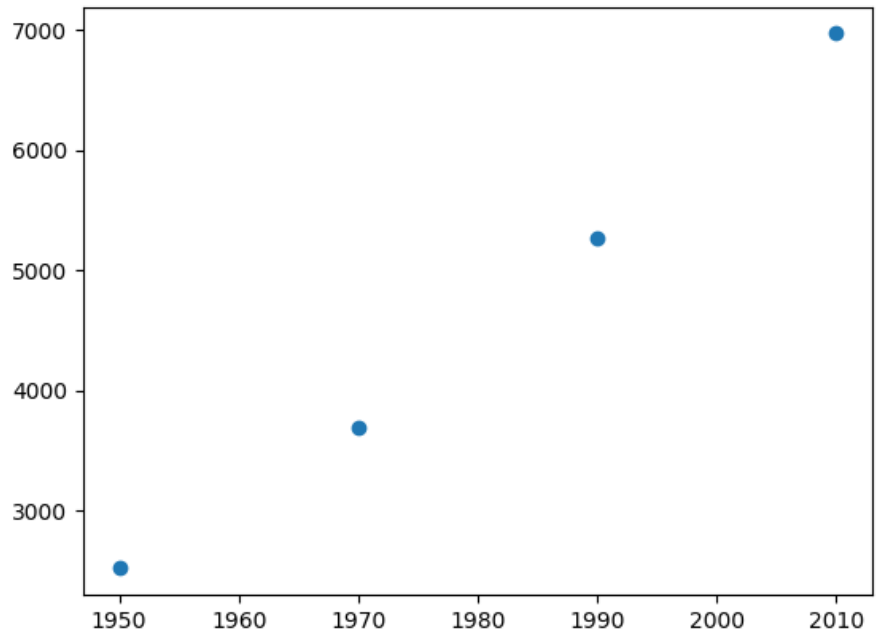
Line plot

```
1 import matplotlib.pyplot as plt
2 anos = [1950, 1970, 1990, 2010]
3 pops = [2519, 3692, 5263
4        , 6972]
5
6 plt.plot(anos, pops)
7 #          X         Y
8
9 plt.show()
```



Scatter plot

```
1 import matplotlib.pyplot as plt
2 anos = [1950, 1970, 1990, 2010]
3 pops = [2519, 3692, 5263
4       , 6972]
5
6 plt.scatter(anos, pops)
7 #           X         Y
8
9 plt.show()
```

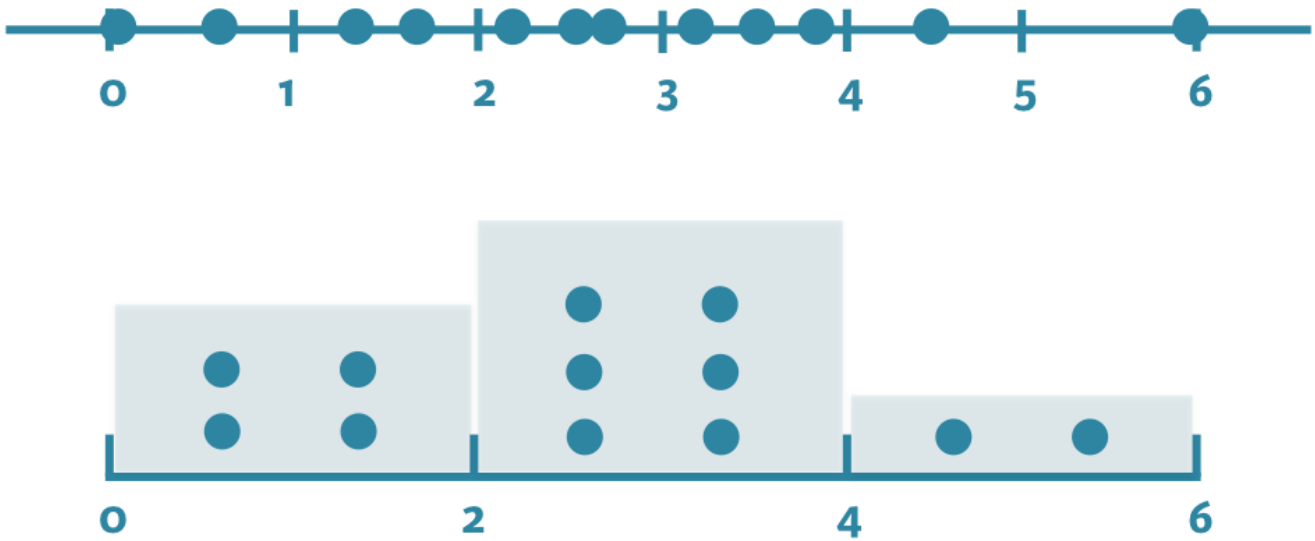


Hora de colocar a mão na massa (1)

Salvar cada um dos exercícios a seguir em um arquivo separado.

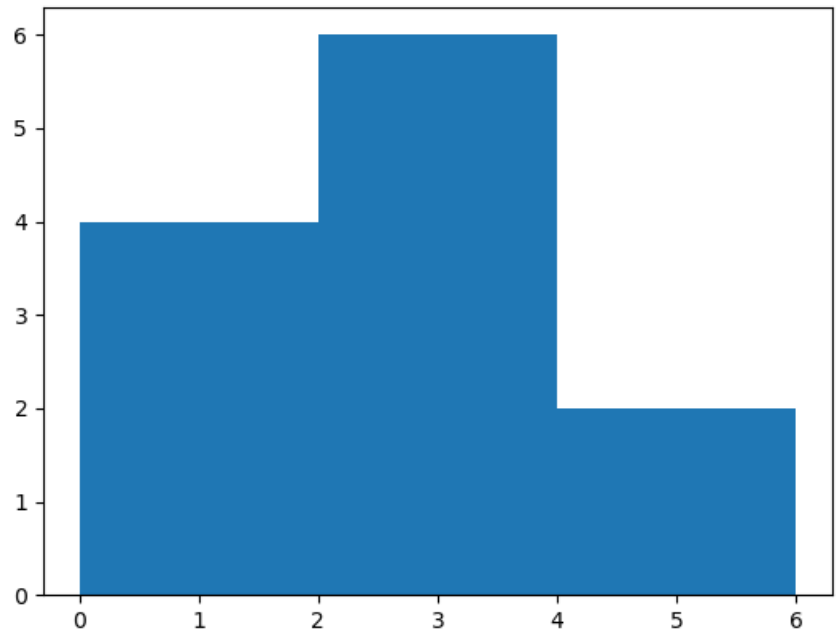
Histogram

- Utilizado para explorar dados
- Fornece uma idea da distribuição dos dados



Histogram

```
1 import matplotlib.pyplot as plt
2 valores = [
3       0,   0.6, 1.4, 1.6
4     , 2.2, 2.5, 2.6, 3.2
5     , 3.5, 3.9, 4.2, 6
6 ]
7
8 plt.hist(valores, bins=3)
9 plt.show()
```

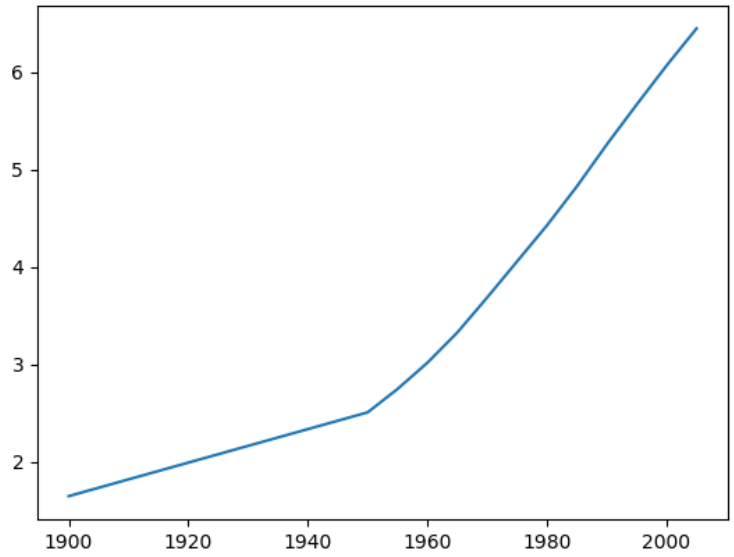


Customização

- Existem muitas opções
  - Diferentes tipos de gráficos
  - Diversas customizações
- A escolha depende
  - Dados
  - Estória a ser contada

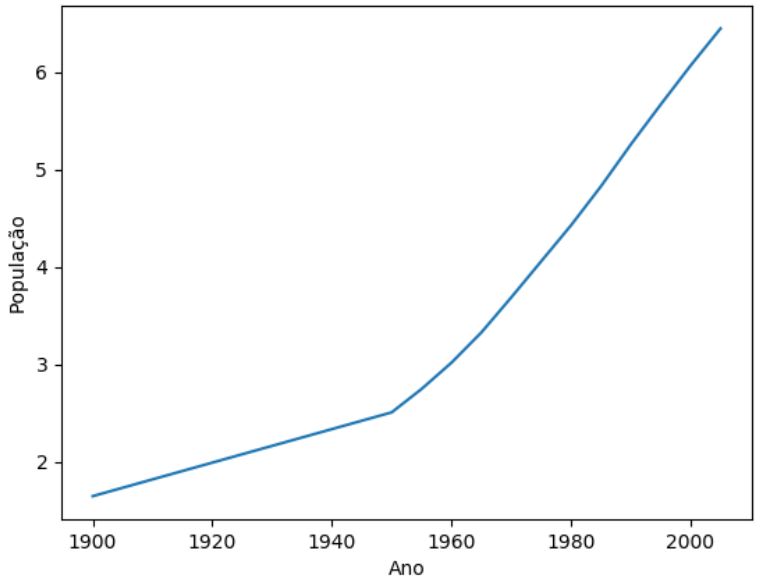
Customização

```
1  import matplotlib.pyplot as plt
2  anos = [
3    1900, 1950, 1955, 1960, 1965,
4    1970, 1975, 1980, 1985, 1990,
5    1995, 2000, 2005
6   ]
7  pops = [
8    1.65, 2.51, 2.75, 3.02, 3.33,
9    3.69, 4.06, 4.43, 4.83, 5.26,
10   5.67, 6.07, 6.45
11 ]
12
13 plt.plot(anos, pops)
14 #           X         Y
15
16 plt.show()
```



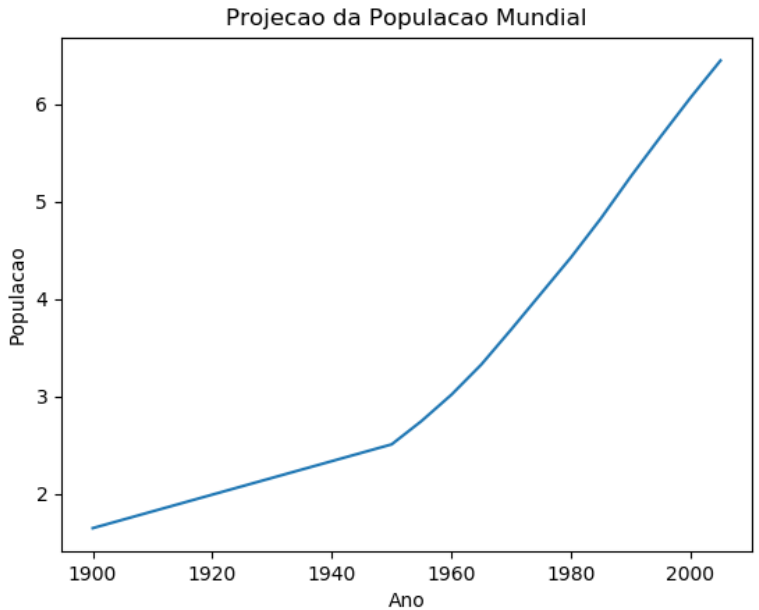
Títulos dos eixos X e Y

```
1  import matplotlib.pyplot as plt
2  anos = [
3    1900, 1950, 1955, 1960, 1965,
4    1970, 1975, 1980, 1985, 1990,
5    1995, 2000, 2005
6   ]
7  pops = [
8    1.65, 2.51, 2.75, 3.02, 3.33,
9    3.69, 4.06, 4.43, 4.83, 5.26,
10   5.67, 6.07, 6.45
11 ]
12
13 plt.plot(anos, pops)
14 plt.xlabel('Ano')
15 plt.ylabel('Populacao')
16 plt.show()
```



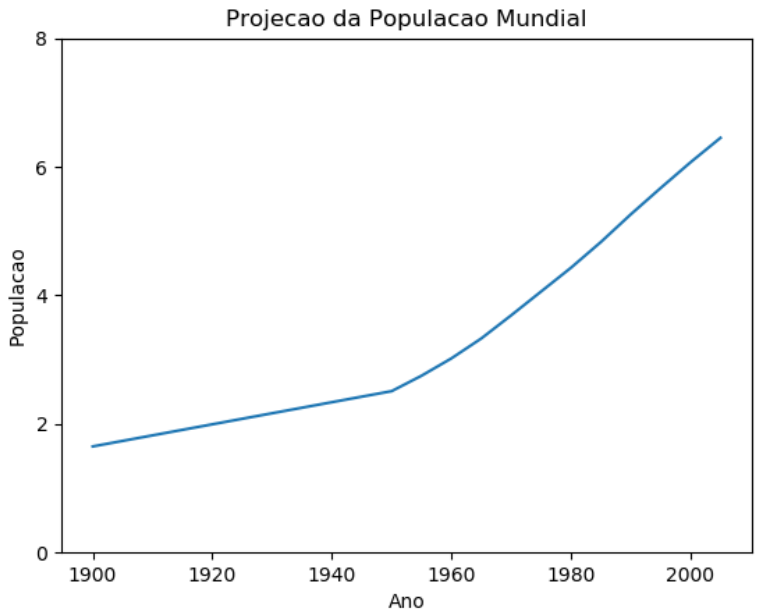
Título Principal

```
1 plt.plot(anos, pops)
2 plt.xlabel('Ano')
3 plt.ylabel('Populacao')
4 plt.title('Projecao da Populacao
      Mundial')
5 plt.show()
```



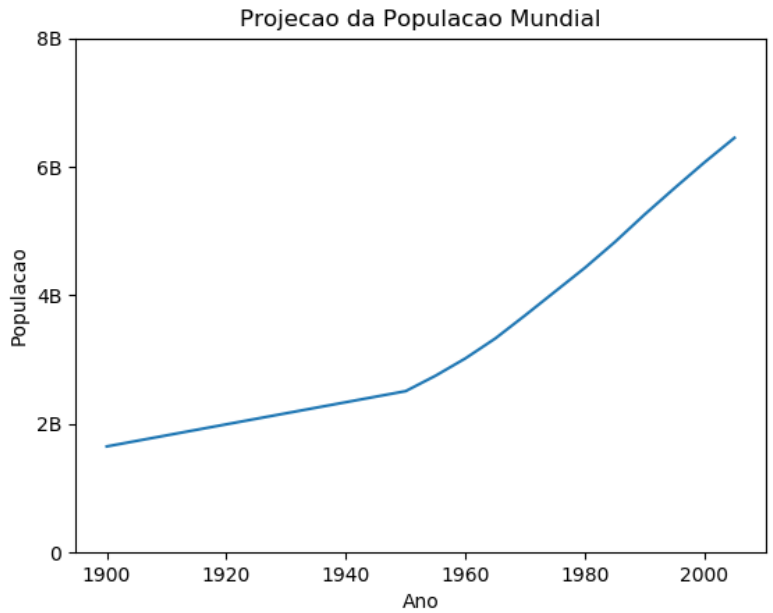
Ticks (1)

```
1 plt.plot(anos, pops)
2 plt.xlabel('Ano')
3 plt.ylabel('Populacao')
4 plt.title('Projecao da Populacao
      Mundial')
5 plt.yticks([0, 2, 4, 6, 8])
6 plt.show()
```



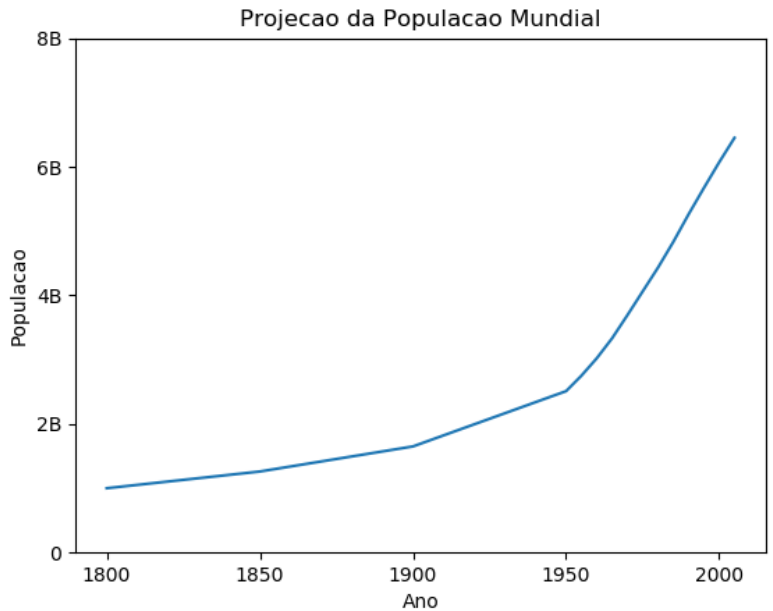
Ticks (2)

```
1 plt.plot(anos, pops)
2 plt.xlabel('Ano')
3 plt.ylabel('Populacao')
4 plt.title('Projecao da Populacao
      Mundial')
5 plt.yticks([0, 2, 4, 6, 8],
6      ['0', '2B', '4B', '6B', '8B'])
7 plt.show()
```



Adicionando Dados Históricos

```
1 anos = [1800, 1850] + anos
2 pops = [1.0, 1.26] + pops
3 plt.plot(anos, pops)
4 plt.xlabel('Ano')
5 plt.ylabel('Populacao')
6 plt.title('Projecao da Populacao
      Mundial')
7 plt.yticks([0, 2, 4, 6, 8],
8      ['0', '2B', '4B', '6B', '8B'])
9 plt.show()
```



Antes x Depois