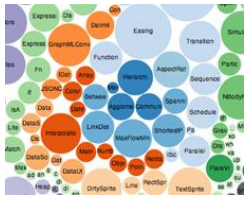
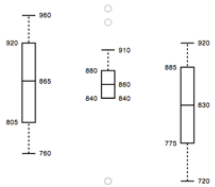
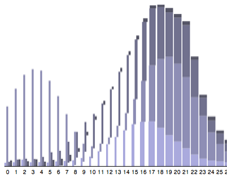
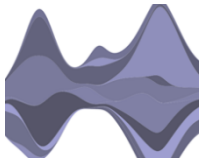
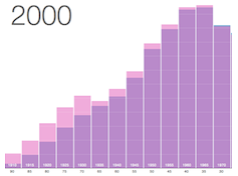
Representação gráfica com matplotlib

Sumário

- Introdução
- Gráficos de linhas
- Gráficos de barra
- Gráficos de tarte
- Acesso de ficheiros CSV
- Stackplot
- Histogramas
- Scatter Plots
- Séries Temporais
- Dados em tempo real

Introdução

- Matplotlib (https://matplotlib.org/) é uma biblioteca que permite criar graficos com dados.
- Muito utilizado em data science.
- Existem vários tipos de gráficos que se conseguem criar.
- Skill muito valorizado pelas empresas, analisar dados e representar de forma gráfica interessante.

Iniciação

- Deve-se instalar a biblioteca com
  `pip install matplotlib`

- Como exemplo de demonstração, são usados dados do stackoverflow, sobre survey de 2019
  do salário médio de developers por idade
  https://insights.stackoverflow.com/survey/2019

```
dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]
```

Gráfico com uma linha

- Basta definir as listas dos valores para x e y (`dev_x, dev_y`)
- `plt.plot(dev_x, dev_y)` cria uma linha com os dados
- `plt.show()` mostra o gráfico, com uma linha neste caso

```
from matplotlib import pyplot as plt

# idades de developer
dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

# Salarios médios anuais de developer por idade
dev_y = [38496, 42000, 46752, 49320, 53200,
56000, 62316, 64928, 67317, 68748, 73752]

plt.plot(dev_x, dev_y)

plt.show()
```

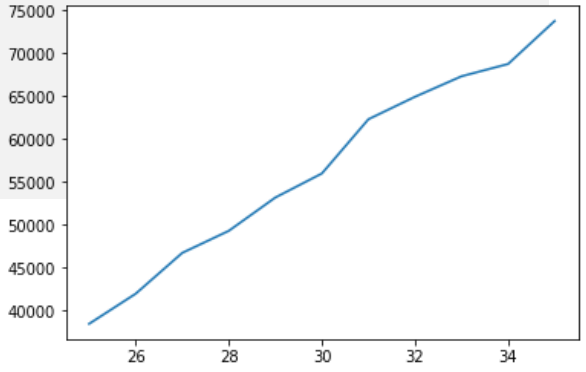
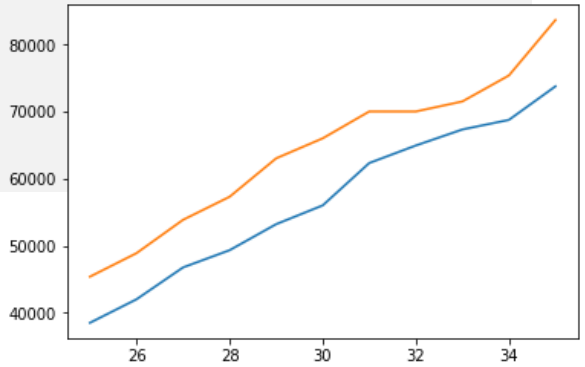


Gráfico com duas linhas

- Podemos adicionar uma segunda linha (salários de devs. Python), com o mesmo plt.plot(idade_x, py_dev_y)

```
from matplotlib import pyplot as plt

idade_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]

plt.plot(idade_x, dev_y)

py_dev_y = [45372, 48876, 53850, 57287, 63016,
            65998, 70003, 70000, 71496, 75370, 83640]

plt.plot(idade_x, Python_dev_y)

plt.show()
```



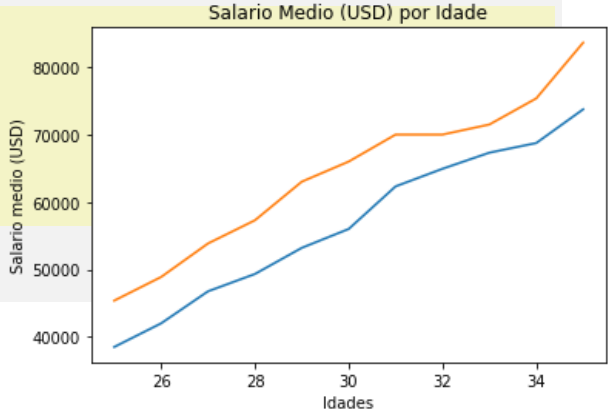
Gráficos com títulos

Pode-se adicionar títulos aos eixos e gráfico:

- `plt.xlabel('Idade’)`: título do eixo dos x
- `plt.ylabel('Salario')`: título do eixo dos y
- `plt.title('Titulo')` : título do gráfico

Gráficos com títulos

```
from matplotlib import pyplot as plt

idade_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]

plt.plot(idade_x, dev_y)

py_dev_y = [45372, 48876, 53850, 57287, 63016,
            65998, 70003, 70000, 71496, 75370, 83640]

plt.plot(idade_x, Python_dev_y)

plt.xlabel('Idades')
plt.ylabel('Salario medio (USD)')
plt.title('Salario Medio (USD) por Idade')

plt.show()
```



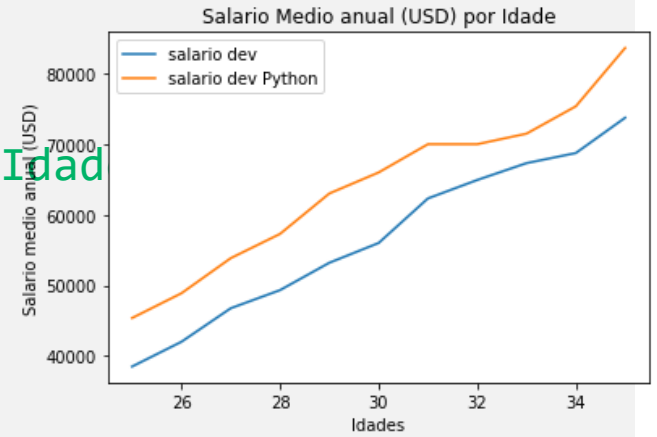
Linhas com títulos

Pode-se adicionar títulos às linhas:

- `plt.legend()`: gera legendas associadas às labels de cada linha
- `plt.ylabel('Salario')`: título do eixo dos y
- `plt.title('Titulo')` : título do gráfico

Linhas com títulos

```
from matplotlib import pyplot as plt

idade_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]

plt.plot(idade_x, dev_y, label = 'salario dev')

py_dev_y = [45372, 48876, 53850, 57287, 63016,
            65998, 70003, 70000, 71496, 75370, 83640]

plt.plot(idade_x, Python_dev_y, label = 'salario dev Python')

plt.xlabel('Idades')
plt.ylabel('Salario medio anual (USD)')
plt.title('Salario Medio anual (USD) por Idad

plt.legend()

plt.show()
```



Formatação das linhas

O metodo `plt.plot()` aceita argumentos para formatar as linhas [1]:

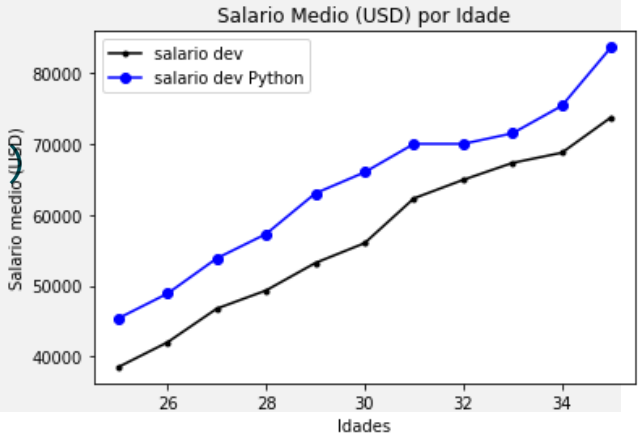
- `marker='.'`
- `linestyle='–'`
- `linewidth=3`
- `color='b'` (blue, k=black, etc... aceita tb codigo hex)

Pode-se também adicionar uma grelha, com o método:

- `plt.grid(True)`

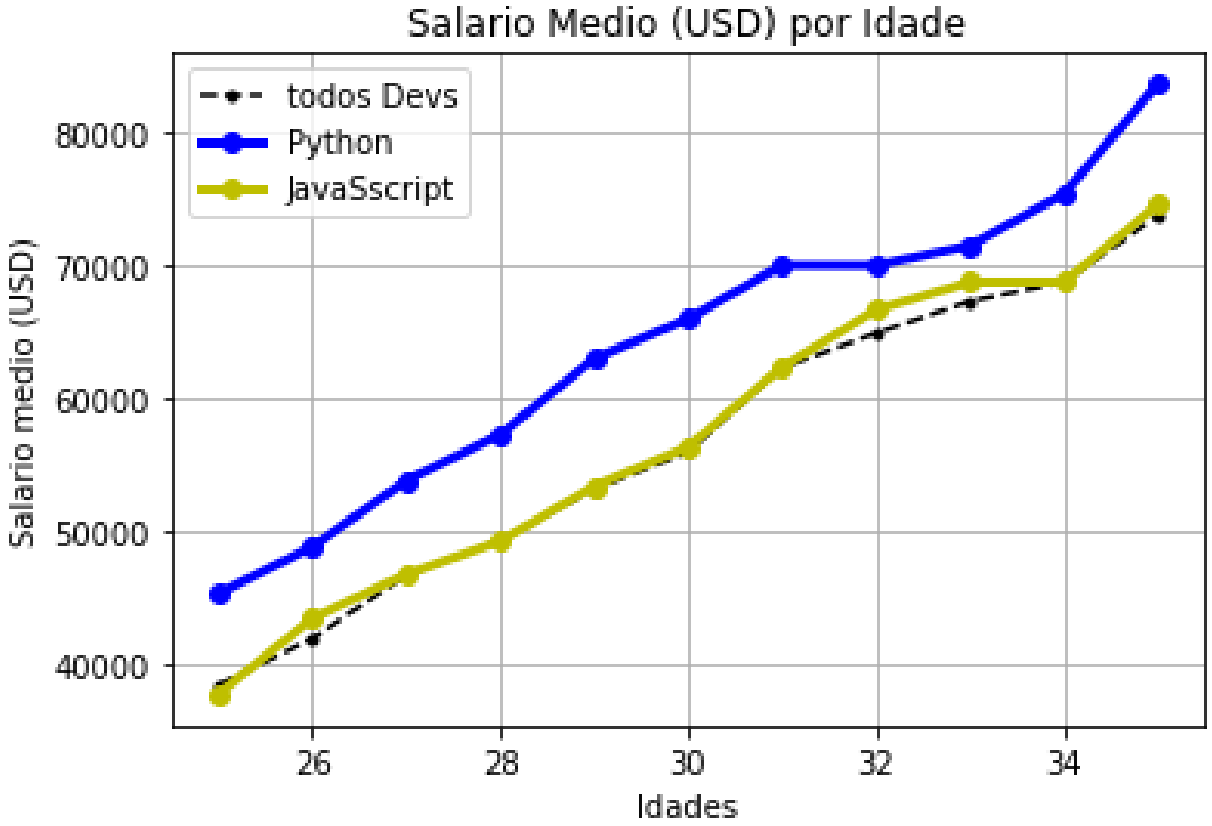
Formatação das Linhas

```
from matplotlib import pyplot as plt

idade_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]

plt.plot(idade_x, dev_y, color='black', marker='.',
         label = 'salários dev')

py_dev_y = [45372, 48876, 53850, 57287, 63016,
            65998, 70003, 70000, 71496, 75370, 83640]

plt.plot(idade_x, Python_dev_y, color='blue', marker='o',
         label = 'salario dev Python')

plt.xlabel('Idades')
plt.ylabel('Salario medio anual (USD)')
plt.title('Salario Medio anual (USD) por Idade')

plt.legend()

plt.show()
```



Adicionando uma nova linha

- Podemos adicionar uma nova linha com dados de salários de devs JavaScript, e formatá-la.



Estilos

Existem também vários estilos disponíveis, que podem ser listados com o comando:

- `print(plt.style.available)`

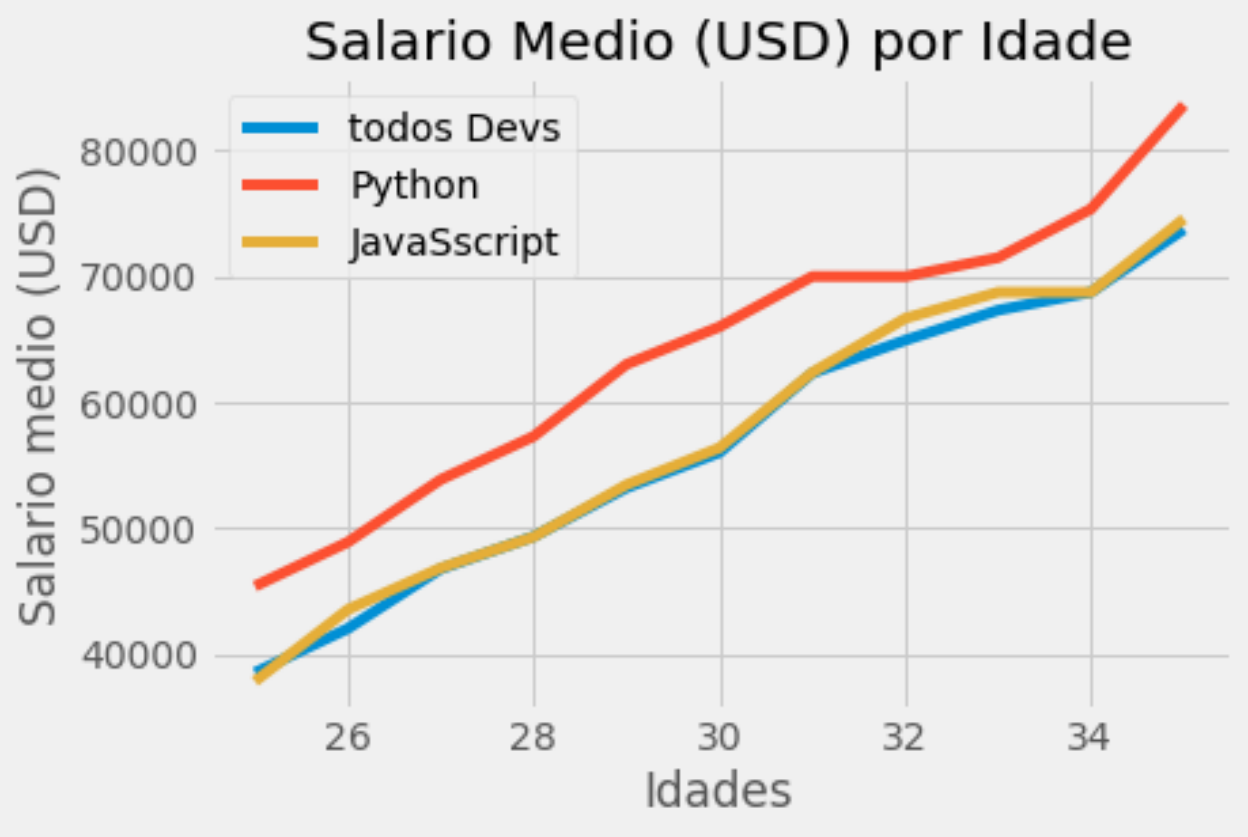
```
['bmh', 'classic', 'dark_background', 'fast',
'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-
bright', 'seaborn-colorblind', 'seaborn-dark-
palette', 'seaborn-dark', 'seaborn-darkgrid',
'seaborn-deep', 'seaborn-muted', 'seaborn-notebook',
'seaborn-paper', 'seaborn-pastel', 'seaborn-poster',
'seaborn-talk', 'seaborn-ticks', 'seaborn-white',
'seaborn-whitegrid', 'seaborn', 'Solarize_Light2',
'tableau-colorblind10', '_classic_test’]
```

Usam-se com o comando:

- `plt.style.use('fivethirtyeight')`

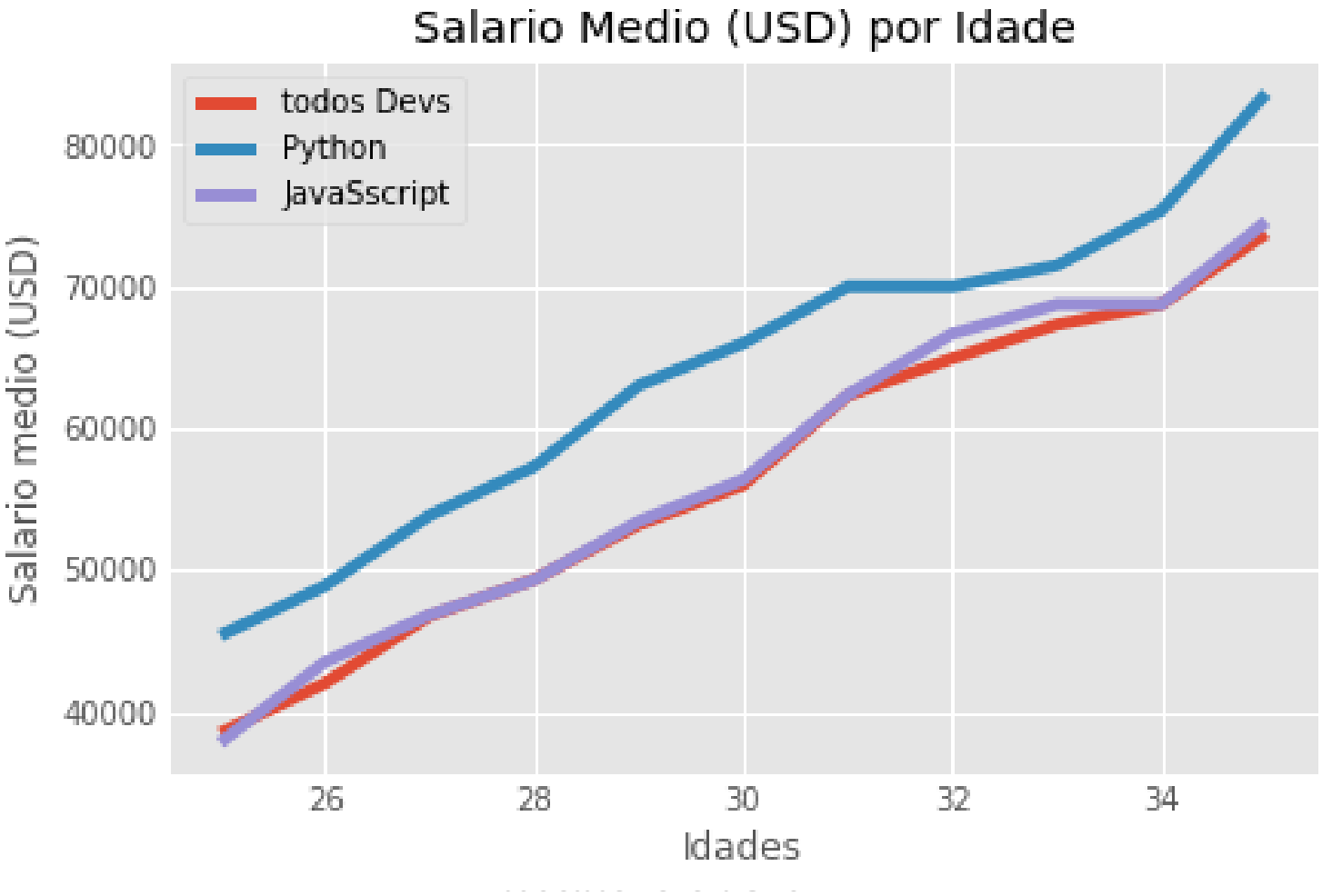
Estilos

Formato `plt.style.use('fivethirtyeight')`



Estilos

Formato `plt.style.use('ggplot')`

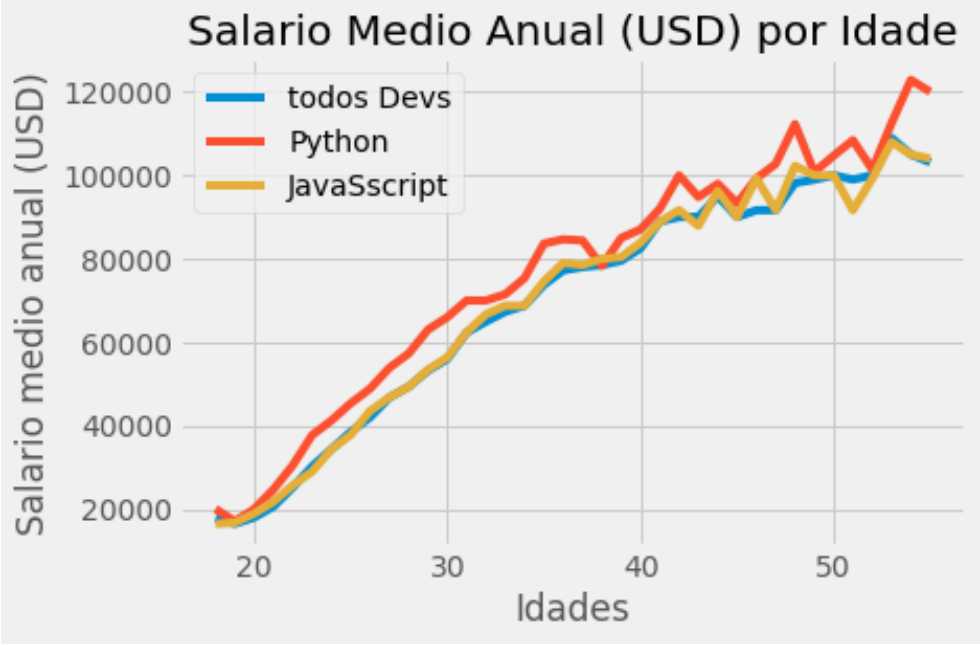


Layout

- `plt.tight_layout()` imprime de forma mais compacta

Gravar figura

- `plt.savefig('plot.png ', bbox_inches='tight')`: permite gravar a imagem com um nome.
  - bbox_inches=`'tight'` garante que a imagem seja gravada sem cortes
- É possível gravar em formato svg [1]

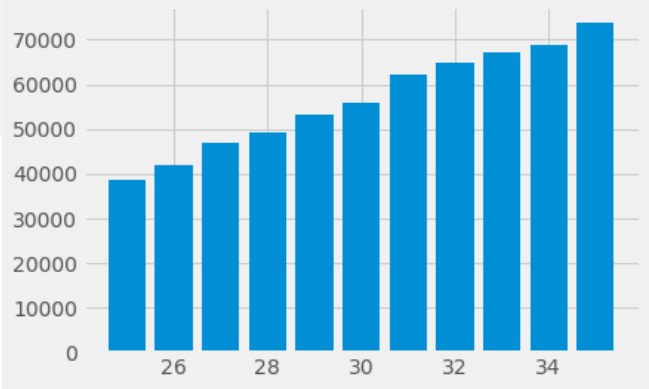


Gráficos de barras

Gráficos de barras

- Basta definir as listas dos valores para x e y (`dev_x, dev_y`)
- `plt.bar(dev_x, dev_y)` cria um grafico de barras

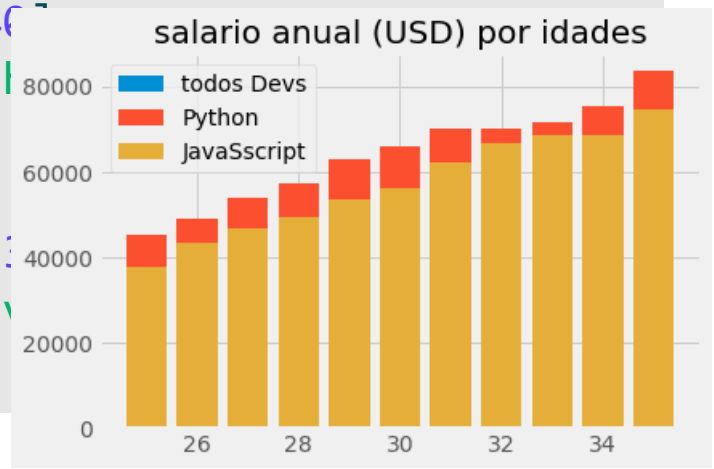
```
from matplotlib import pyplot as plt

plt.style.use('fivethirtyeight’)

dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

dev_y = [38496, 42000, 46752, 49320, 53200,
         56000, 62316, 64928, 67317, 68748, 73752]
plt.bar(dev_x, dev_y)

plt.show()
```



Multiplos gráficos de barras

- No entanto, múltiplos sobrepoem-se não permitindo a sua leitura.

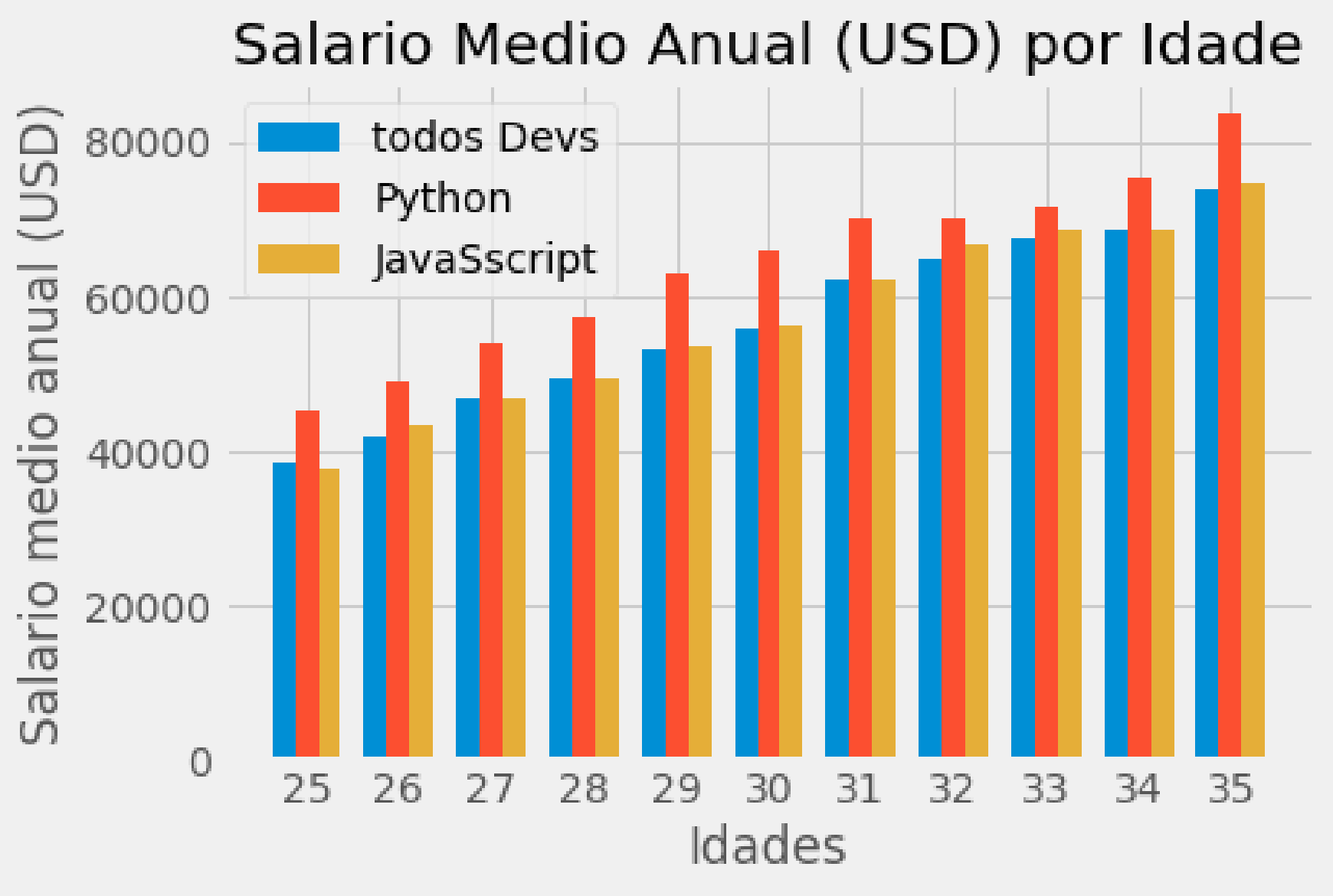
```
from matplotlib import pyplot as plt

idades_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
dev_y = [38496, 42000, 46752, 49320, 53200,
56000, 62316, 64928, 67317, 68748, 73752]
plt.bar(idades_x, dev_y, label = 'todos Devs')

py_dev_y = [45372, 48876, 53850, 57287, 63016,
65998, 70003, 70000, 71496, 75370, 83640]
plt.bar(idades_x, py_dev_y,label = 'Pyth

js_dev_y = [37810, 43515, 46823, 49293,
56373, 62375, 66674, 68745, 68746, 74581]
plt.bar(idades_x, js_dev_y, label = 'Jav
plt.show()
```



Multiplos gráficos de barras

- Para representar corretamente:
- `x_indexes = np.arange(len(idades_x))`: cria-se uma lista de índices com a biblioteca numpy
- `width = 0.25`: Especifica-se uma largura
- `plt.bar(x_indexes - width, ...`: Desloca-se cada indice
- `width=width`: especifica-se a largura das barras
- `plt.xticks(ticks=x_indexes, labels=idades_x)`: epecifica-se os titulos das ticks como sendo as idades

Multiplos gráficos de barras



Utilizando dados de um ficheiro

- Considere um ficheiro .csv onde cada linha identifica, para uma pessoa, as linguagens em que trabalha:

```
Responder_id,LanguagesWorkedWith
1,HTML/CSS;Java;JavaScript;Python
2,C++;HTML/CSS;Python
3,HTML/CSS
4,C;C++;C#;Python;SQL
5,C++;HTML/CSS;Java;JavaScript;Python;SQL;VBA
6,Java;R;SQL
7,HTML/CSS;JavaScript
...
```

- Existem dois campos, separados por ”,” :
  - Responder_id
  - LanguagesWorkedWith

Biblioteca csv

- Importando a biblioteca csv, `import csv`
- `csv.DictReader(csv_file)` guarda cada linha num dicionario com chaves:
  - `Responder_id`
  - `LanguagesWorkedWith`

Responder_id	LanguagesWorkedWith
1	HTML/CSS;Java;JavaScript;Python
2	C++;HTML/CSS;Python
3	HTML/CSS
4	C;C++;C#;Python;SQL
5	C++;HTML/CSS;Java;JavaScript;Python;SQL;VBA
6	Java;R;SQL
7	HTML/CSS;JavaScript
...	...

Biblioteca csv

- Queremos identificar as linguagens mais populares.

```
import csv
from matplotlib import pyplot as plt

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    row = next(csv_reader)
    print(row)
```

```
OrderedDict([
   ('Responder_id', '1’),
   ('LanguagesWorkedWith',
'HTML/CSS;Java;JavaScript;Python’)
])
```

Biblioteca csv

- Queremos identificar as linguagens mais populares.

```
import csv
from matplotlib import pyplot as plt

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    row = next(csv_reader)
    print(row['LanguagesWorkedWith'].split(';'))
```

```
['HTML/CSS', 'Java', 'JavaScript', 'Python']
```

Classe Counter

- Classe da biblioteca collections
- Counter faz contagem de ocorrencia de chaves em listas:

```
from collections import Counter
c = Counter({'Python':1, 'JavaScript': 1})
c
```

Counter({'Python’: 1, 'JavaScript': 1})

```
c.update(['C++', 'Python'])
c.update(['C++', 'Python'])
c
```

Counter({'Python': 3, 'JavaScript': 1, 'C++': 2})

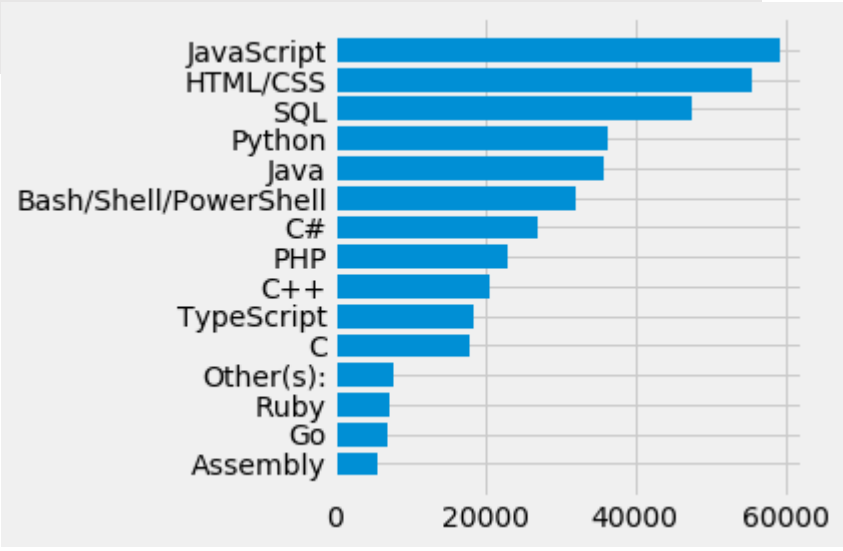
Contador de linguagens

```
import csv
from collections import Counter
from matplotlib import pyplot as plt

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    language_counter = Counter()
    for row in csv_reader:
        lista = row['LanguagesWorkedWith'].split(';’)
        language_counter.update(lista)
print(language_counter)
```

Counter({'JavaScript': 59219, 'HTML/CSS': 55466, 'SQL': 47544, 'Python': 36443, 'Java': 35917, 'Bash/Shell/PowerShell': 31991, 'C#': 27097, 'PHP': 23030, 'C++': 20524, 'TypeScript': 18523, 'C': 18017, 'Other(s):': 7920, 'Ruby': 7331, 'Go': 7201, 'Assembly': 5833, 'Swift': 5744, 'Kotlin': 5620, 'R': 5048, 'VBA': 4781, 'Objective-C': 4191, 'Scala': 3309, 'Rust': 2794, 'Dart': 1683, 'Elixir': 1260, 'Clojure': 1254, 'WebAssembly': 1015, 'F#': 973, 'Erlang': 777})

Gráfico de popularidade de linguagens

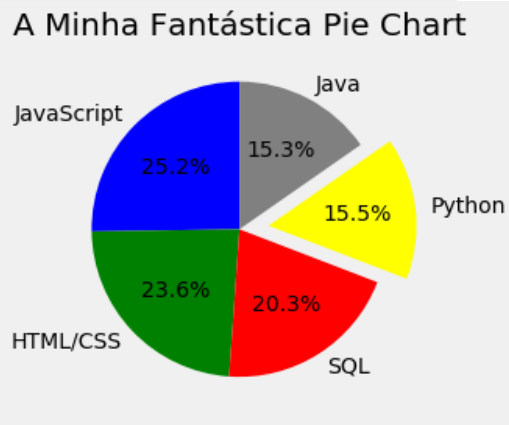
```
...
languages_x = []
popularity_y = []

for language,popularity in language_counter.most_common(15):
    languages_x.append(language)
    popularity_y.append(popularity)

languages_x.reverse()
popularity_y.reverse()
plt.barh(languages_x, popularity_y)
plt.show()
```



Pie Chart

- `plt.pie()`: cria uma pie chart para uma lista

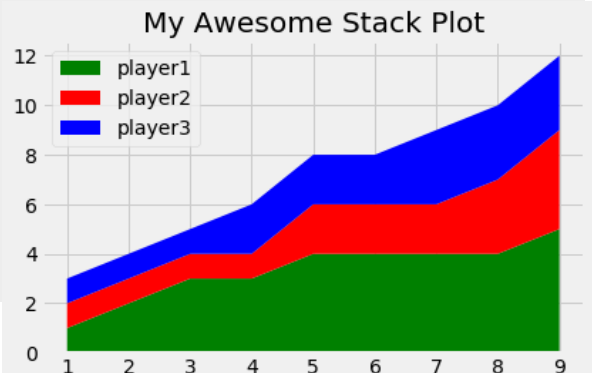
```
from matplotlib import pyplot as plt
plt.style.use("fivethirtyeight")

fatias = [59219, 55466, 47544, 36443, 35917]
etiquetas = ['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java']
cores = ['blue', 'green', 'red', 'yellow', 'grey']
explode = [0, 0, 0, 0.2, 0]

plt.pie(fatias, labels=etiquetas, colors=cores,
        explode=explode, startangle=90, autopct='%1.1f%%')

plt.title("A Minha Fantástica Pie Chart")
plt.tight_layout()
plt.show()
```



Stack Plot

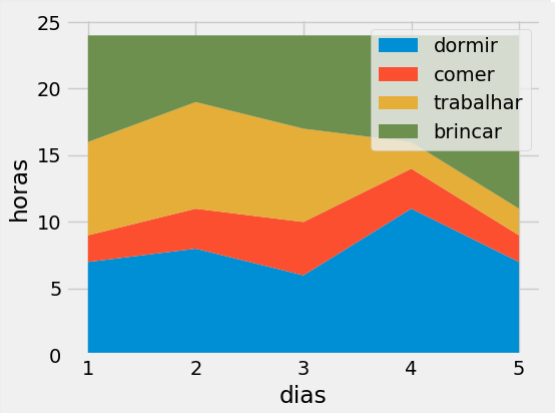
- `plt.stackplot` cria um grafico em pilha

```
from matplotlib import pyplot as plt
plt.style.use("fivethirtyeight")

minutes = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# pontos marcados ao longo do tempo
player1 = [1, 2, 3, 3, 4, 4, 4, 4, 5]
player2 = [1, 1, 1, 1, 2, 2, 2, 3, 4]
player3 = [1, 1, 1, 2, 2, 2, 3, 3, 3]

labels = ['player1', 'player2', 'player3']
colors = ['green', 'red', 'blue']
plt.stackplot(minutes, player1, player2, player3,
              labels=labels, colors=colors)
plt.legend(loc='upper left')
plt.title("My Awesome Stack Plot")
plt.tight_layout()
plt.show()
```



Stack Plot

```
import matplotlib.pyplot as plt
plt.style.use("fivethirtyeight")

dias = [ 1, 2, 3, 4, 5]

dormir =     [ 7, 8, 6,11, 7]
comer =      [ 2, 3, 4, 3, 2]
trabalhar = [ 7, 8, 7, 2, 2]
brincar =    [ 8, 5, 7, 8,13]

labels = ['dormir', 'comer', 'trabalhar', 'brincar']

plt.stackplot(dias, dormir, comer, trabalhar, brincar,
labels=labels)
plt.legend()
plt.title("Ocupação do dia")
plt.tight_layout()
plt.show()
```

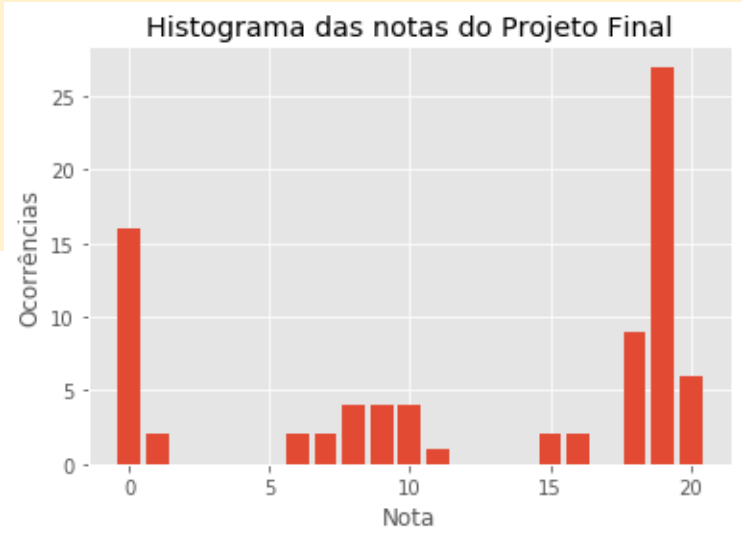


Stack Plot

- Muito usado pelo *youtube analytics*:
  - Uso de tráfego
  - Total de visualizações de tráfego

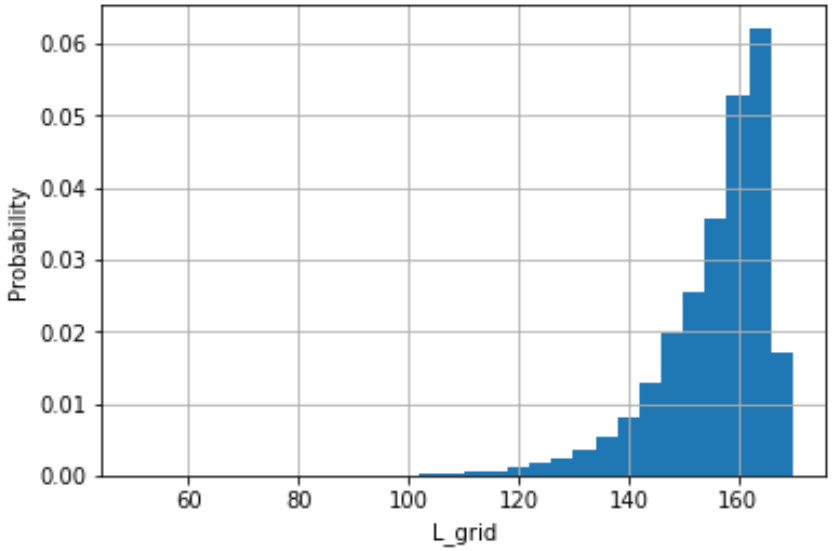
Histogramas

```
from matplotlib import pyplot as plt
from statistics import stdev, mean
plt.style.use('ggplot')

with open("Contest__18__LP12020PF_3.csv") as f:
    lista = f.readlines()

    notas = [int(linha.split(";")[-2]) if (linha.split(";")[-2]).isdigit() else 0
             for linha in lista[1:]] # campos vazio são 0
    media = sum(notas)/len(notas)
    stedv = stdev(notas)
    print(f"Prjcts = {len(notas)}, media = {media:.1f}, stedv: {stdev(notas):.1f}")
    hist = [notas.count(nota) for nota in range(0,21)]

    plt.bar(list(range(0,21)), hist)
    plt.xlabel("Nota")
    plt.ylabel("Ocorrências")
    plt.title("Histograma das notas do Projeto")

    plt.show()
```



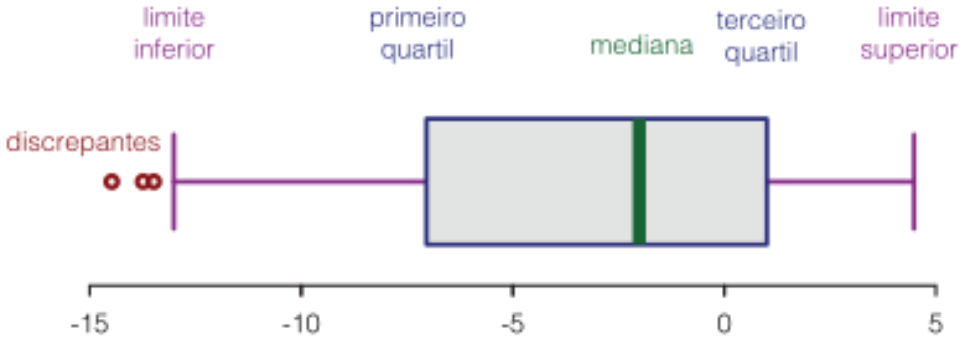
Histogramas

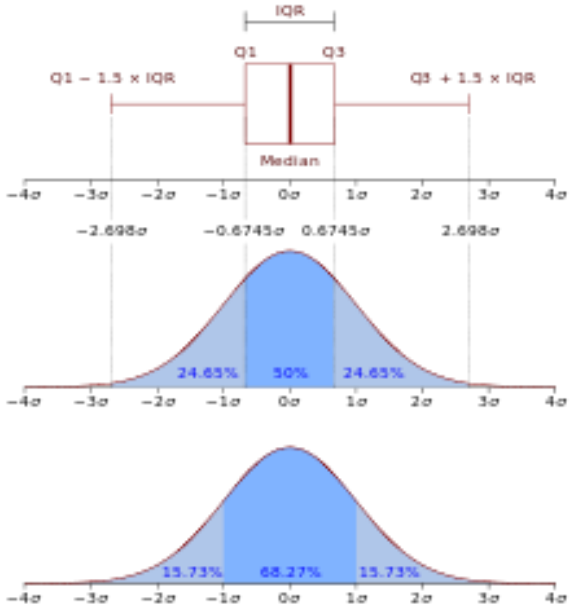
- def plothistogram(data):
  - plt.hist(data, density=True, bins=30) # density
  - plt.ylabel('Probability')
  - plt.xlabel('L_grid')
  - plt.grid()
  - plothistogram(L_grid)



Box Plot

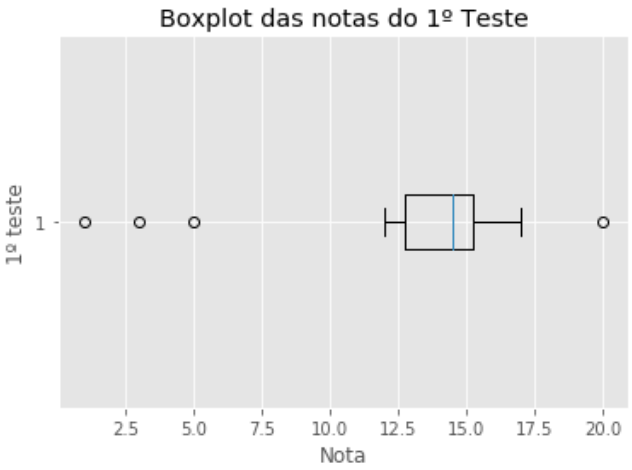
- Em estatística descritiva, um box plot é um diagrama de quartis.
- Ferramenta gráfica para representar a variação de dados observados de uma variável numérica por meio de quartis.
- Os valores atípicos ou outliers (valores discrepantes) podem ser plotados como pontos individuais





Box Plot

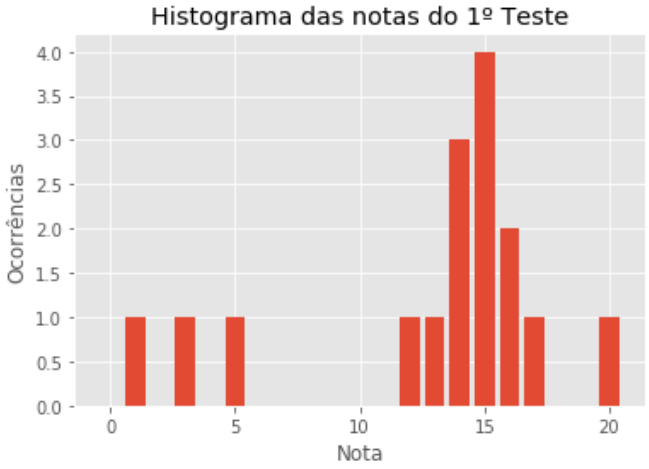
```
teste1 = [15,13,12,15,16,14,15,17,
          14,16,14,15,3,5,1,20]

plt.boxplot(teste1, vert=False)
```



Histograma

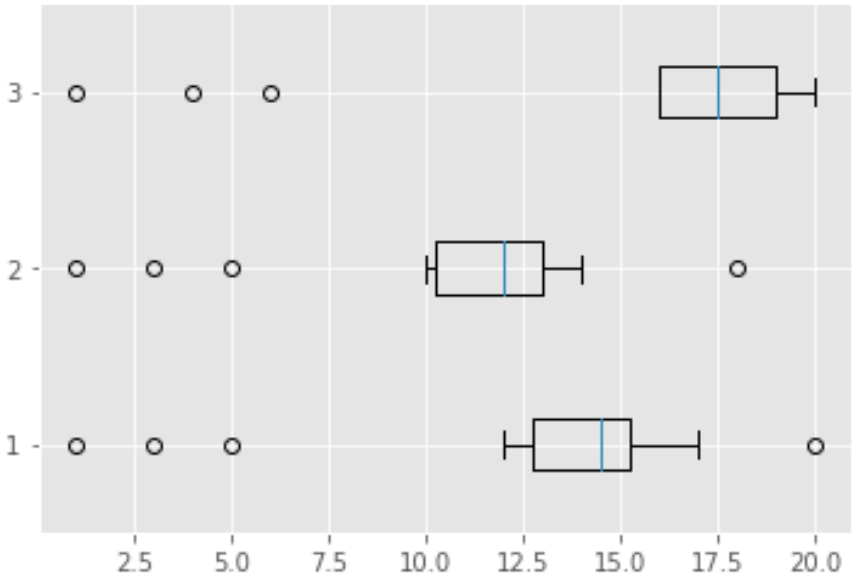
```
hist = [teste1.count(nota)
        for nota in range(0,21)]

plt.bar(list(range(0,21)), hist)

plt.show()
```

*O **boxplot** é uma representação **mais sintética** da informação estatística dos dados do que um **histograma***



Múltiplos Box Plot

```
teste1 = [15,13,12,15,16,14,15,17,14,16,14,15,3,5,1,20]
teste2 = [12,13,11,14,12,11,10,12,14,13,18,1,3,5]
teste3 = [18,16,19,17,19,20,18,17,19,20,18,17,16,1,4,6]

plt.boxplot([teste1,teste2,teste3], vert=False)
```

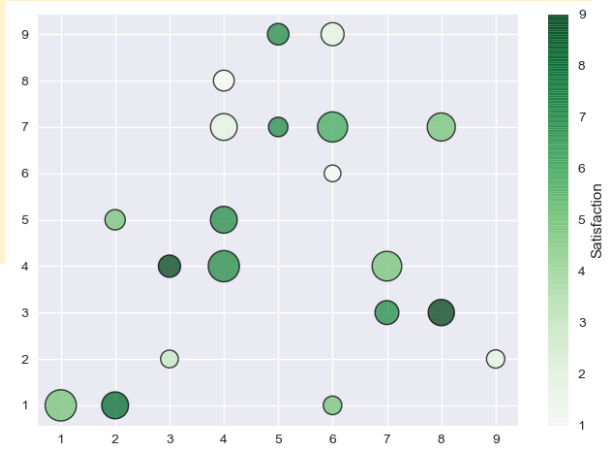


- Mais detalhes sobre box plot [1,2]

Scatter Plots

- Cada par (x,y) tem associado uma intensidade de cor (popularidade) e tamanho (população)

```
from matplotlib import pyplot as plt
plt.style.use('seaborn')

x = [5, 7, 8, 5, 6, 7, 9, 2, 3, 4, 4, 4, 2, 6, 3, 6, 8, 6, 4, 1]
y = [7, 4, 3, 9, 1, 3, 2, 5, 2, 4, 8, 7, 1, 6, 4, 9, 7, 7, 5, 1]
colors=[7,5, 9, 7, 5, 7, 2, 5, 3, 7, 1, 2, 8, 1, 9, 2, 5, 6, 7, 5]
sizes = [209, 486, 381, 255, 191, 315, 185, 228, 174,
         538, 239, 394, 399, 153, 273, 293, 436, 501, 397, 539]

plt.scatter(x,y, s=sizes, c=colors, cmap='Greens',
edgecolor='black', linewidth=1, alpha=0.75)

cbar = plt.colorbar()
cbar.set_label('Satisfaction')
plt.show()
```

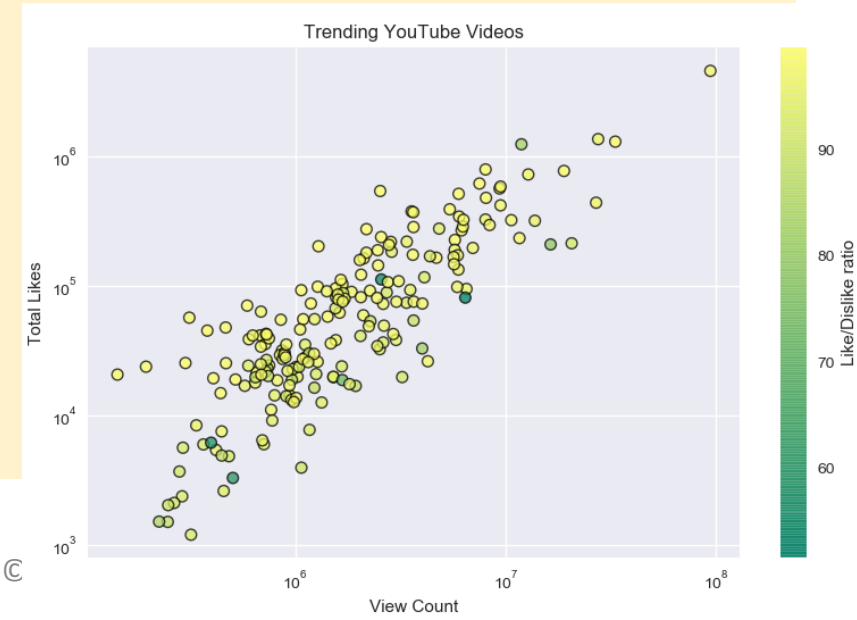


Scatter Plots

- CSV com info de top 200 Youtube videos sobre views, likes e like/dislike ratio

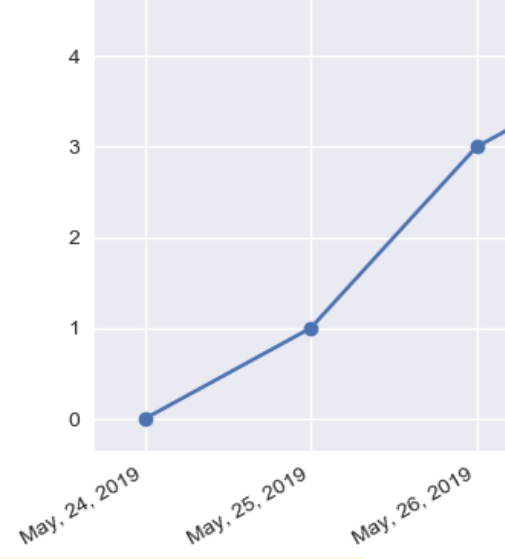
Mostrar legenda de cor, associada a ao cmap por termos especificado cor (c) como raio

```
import pandas as pd
from matplotlib import pyplot as plt
plt.style.use('seaborn')

data = pd.read_csv('2019-05-31-data.csv')
view_count = data['view_count’] ; likes = data['likes’]; ratio = data['ratio']

plt.scatter(view_count, likes, c=ratio, cmap='summer',
edgecolor='black', linewidth=1, alpha=0.75)

cbar = plt.colorbar();
cbar.set_label('Like/Dislike ratio')

plt.xscale('log’); plt.yscale('log')

plt.title('Trending YouTube Videos’)
plt.xlabel('View Count’)
plt.ylabel('Total Likes')
plt.tight_layout()
plt.show()
```



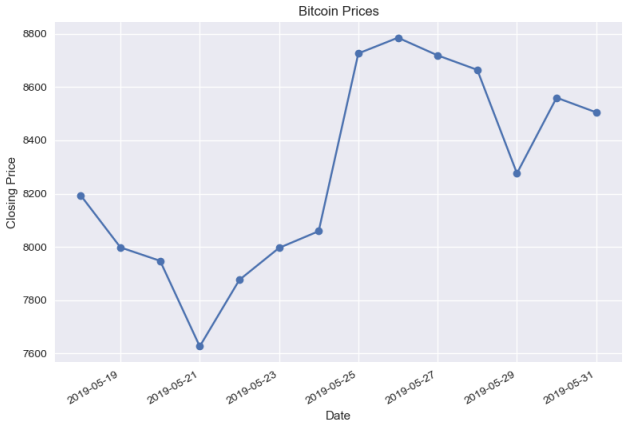
Series Temporais

- Podemos formatar as datas como quisermos, nome do mês, etc. Ver aqui outros exemplos.




```
import pandas as pd
from datetime import datetime, timedelta
from matplotlib import pyplot as plt
from matplotlib import dates as mpl_dates
plt.style.use('seaborn')

dates = [ datetime(2019, 5, 24),  datetime(2019, 5, 25),
datetime(2019, 5, 26),  datetime(2019, 5, 27),
datetime(2019, 5, 28),  datetime(2019, 5, 29),
datetime(2019, 5, 30)  ]

y = [0, 1, 3, 4, 6, 5, 7]
plt.plot_date(dates, y, linestyle='solid')

plt.gcf().autofmt_xdate() #Get Current Figure
date_format = mpl_dates.DateFormatter('%b, %d, %Y')
plt.gca().xaxis.set_major_formatter(date_format)

plt.show()
```

Series Temporais




```
import pandas as pd
from datetime import datetime, timedelta
from matplotlib import pyplot as plt
from matplotlib import dates as mpl_dates
plt.style.use('seaborn')

data = pd.read_csv('data.csv')

data['Date'] = pd.to_datetime(data['Date’]) #converte para data
data.sort_values('Date', inplace=True) #ordena

price_date = data['Date']
price_close = data['Close']

plt.plot_date(price_date,price_close, linestyle='solid')

plt.gcf().autofmt_xdate() # inclina as datas

plt.title('Bitcoin Prices’); plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.show()
```

Dados em Tempo-Real

- `from matplotlib.animation import FuncAnimation`

- `FuncAnimation`: classe que executa periodicamente função e atualiza figura

- `FuncAnimation(plt.gcf(), animate, interval=1000)`
  - `plt.gcf()`: figura que queremos animar (gcf)
  - `animate`: funçao que queremos correr
  - `interval=1000`: intervalo [ms]

Dados em Tempo-Real

```
import random
from itertools import count
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

x_vals = []
y_vals = []
index = count() #contador, retornando o proximo valor
def animate(i): #função animate, que concatena elementos
    x_vals.append(next(index))
    y_vals.append(random.randint(0, 5))

    plt.cla() # apaga figura anterior
    plt.plot(x_vals, y_vals)

# classe para plotar novos dados. Damos figura que queremos
animar (gcf), e função que queremos correr, e intervalo
ani = FuncAnimation(plt.gcf(), animate, interval=1000)
plt.tight_layout()
plt.show()
```