

# Python para Análise de Dados

Pacotes Python para  
Visualização de Dados

Eronides Neto  
[efsn@cesar.school](mailto:efsn@cesar.school)



# Eronides Neto

Mestre em Ciência da  
Computação, Eng. Eletrônico  
@UFPE

Especialização em Análise e  
Engenharia de Dados  
@CESAR School

Engenheiro da Computação  
@ CESAR

Professor  
@ CESAR School

LinkedIn: **Eronides da Silva Neto**  
[github.com/eron93br](https://github.com/eron93br)



# Análise e Visualização de Dados

---

Como funcionará a disciplina?



## Parte I - Pacotes para Visualização de Dados

- Revisão Matplotlib/Seaborn\*
- Gráficos interativos com Plotly



## Parte II - Análise de Dados

- Tipos de Análise de Dados
- Storytelling com Dados



## Parte III - Dashboards

- Design e Desenvolvimento
- Dash
- Streamlit



**\*Importante:** O conteúdo/detalhamento desta revisão será baseado na escolha da turma. O material assíncrono estará disponível no Google Classroom. Os códigos fontes utilizados na disciplina estarão disponíveis no GitHub.

# Avaliação

---

- **40% - Atividades** em Sala (Hands-on)
  - Hands On 1 - Pacotes de visualização
  - Hands On 2 - Construção de um Storyboard
  - Hands On 3 - Construção de um Dashboard
- **60% - Apresentação** Final (em grupo\*)



---

**\*Importante:** O detalhamento das regras do projeto serão disponibilizadas assim que o mesmo estiver disponível.

# Ferramentas

---

- PDFs das Aulas no Google Drive/Classroom
- GitHub com códigos, datasets e referências extras
- JupyterBook de roteiro para conteúdo da disciplina



---

\*Importante: Muitas das referências listadas nas aulas e GitHub são de acesso público e open-source.

# Data Viz

## Python para Análise de Dados

Aula 01 - Pacotes Python para Análise de Dados

Eronides da Silva Neto

[efsn@cesar.school](mailto:efsn@cesar.school)



# POR QUE VISUALIZAÇÃO DE DADOS ?



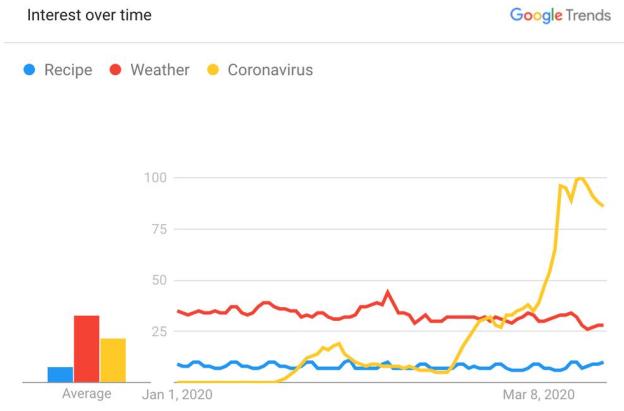
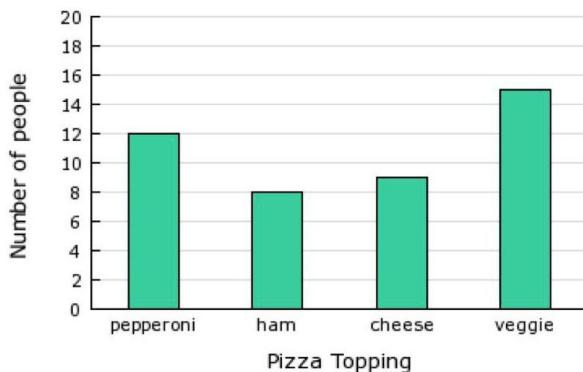
## // visualização·de·dados

1. Apresentação de informações em formato imagético ou gráfico.
2. Permite tomadas de decisão baseada em análises visuais e que conceitos/padrões difíceis possam ser visualizados



# O que é visualização de dados?

**Apresentação** de informações em formato de imagens/gráficos. Tomadas de decisão podem ser feitas a partir destes **resultados visuais**. Além disso, pode permitir a identificação de padrões.



Worldwide. 1/1/20 - 3/20/20. Web Search.

# A importância da visualização de dados

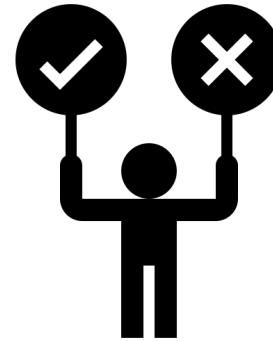
Apresentação visual das informações com o objetivo de comunicá-los da forma mais direta possível para os leitores. É uma maneira de comunicação visual, por isso se torna muito importante.



Dados



Gráficos



Tomada de Decisão

# Exemplos de utilização

JOHNS HOPKINS UNIVERSITY & MEDICINE CORONAVIRUS RESOURCE CENTER

Home Topics ▾ By Region Events & News About Search Data by Region...

Tracking Home Data Visualizations Global Map U.S. Map Data in Motion Tracking FAQ

COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)

Last Updated at (M/D/YYYY)  
1/14/2023, 8:20 AM

Cases | Deaths by Country/Region/Sovereignty

Japan  
28-Day: 4,354,428 | 8,938  
Totals: 31,330,494 | 62,272

US  
28-Day: 1,723,841 | 12,108  
Totals: 101,641,427 | 1,099,823

Korea, South  
28-Day: 1,644,890 | 1,559  
Totals: 29,774,321 | 32,912

Brazil  
28-Day: 750,845 | 3,507  
Totals: 36,620,371 | 695,317

Germany  
28-Day: 658,561 | 4,038  
Totals: 37,605,135 | 163,775

France  
28-Day: 642,184 | 2,988  
Totals: 39,646,833 | 164,388

China  
28-Day: 629,410 | 1,724  
Totals: 4,852,776 | 18,126

Taiwan\*  
28-Day: 120,496 | 1935

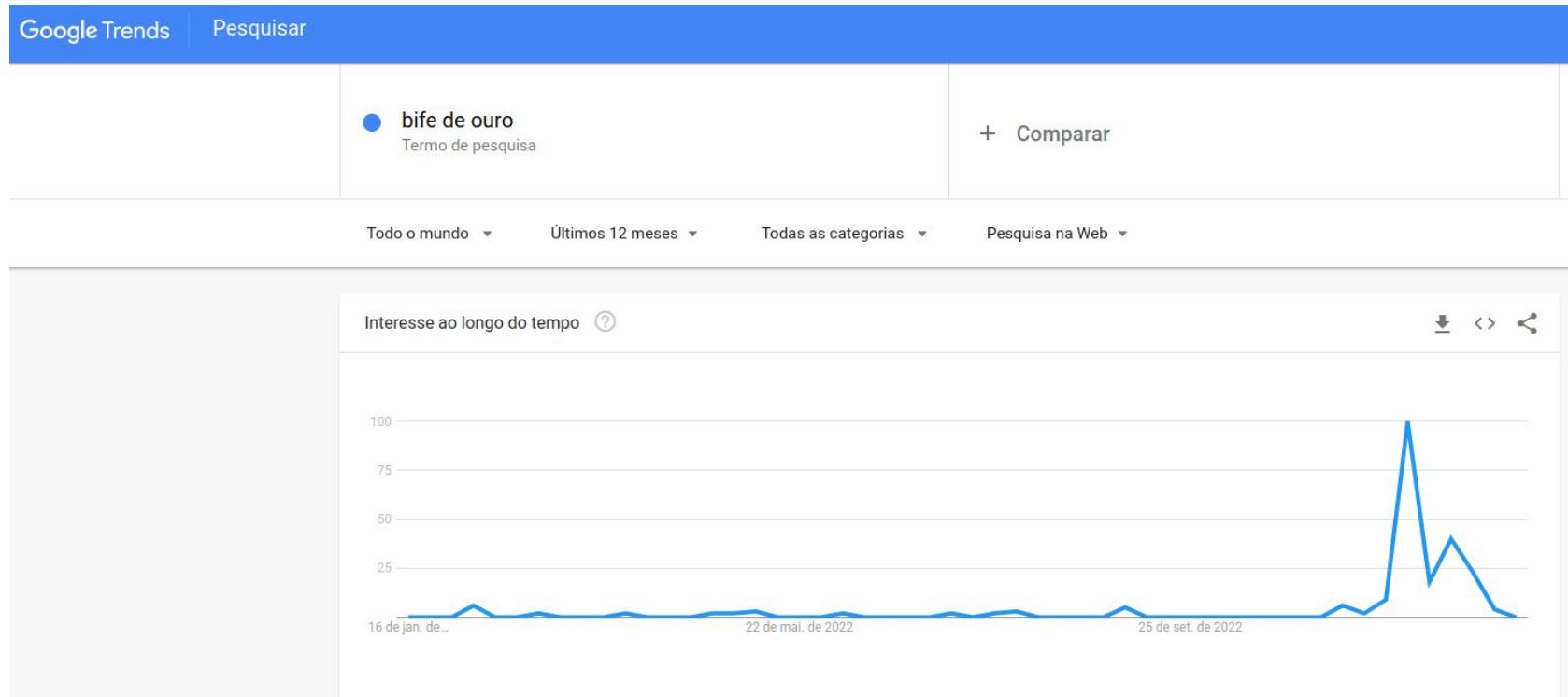
Total Cases: 666,733,170 | Total Deaths: 6,722,505 | Total Vaccine Doses Administered: 13,215,350,706

28-Day Cases: 14,006,756 | 28-Day Deaths: 56,580 | 28-Day Vaccine Doses Administered: 136,547,263

Get HERE | GADM | FAO | NOAA | USGS | Powered by Esri

# Exemplos de utilização

Google Trends



# Exemplos de utilização



# O passo a passo da visualização de dados

Entenda os dados que você está tentando visualizar e que tipo de informação você irá transmitir.



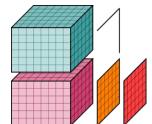
# Fazendo a mágica acontecer!

Quais são os conhecimentos de Análise de Dados até o momento?

Ambiente interativo para execução do software



Bibliotecas de Manipulação de Dados



xarray



Bibliotecas de Visualização de Dados



# Matplotlib

# Matplotlib: visualização com Python

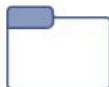
Talvez a mais famosa [biblioteca](#) para criação de gráficos estáticos e visualizações interativas com Python.

Passo a passo para uso:

1. Importação
2. Definir o gráfico desejado
3. Incorpore as **boas práticas** e faça a mágica acontecer!



Getting Started



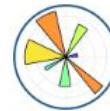
Examples



Reference



Cheat Sheets



Documentation

# Visualização com Python: Matplotlib

---

Alguns métodos (**disponíveis para Series e DataFrames**) interessantes são:

- Gráficos de histograma: hist()
- Gráficos de área: area()
- Gráficos de pizza: pie()
- Gráficos de barra: bar()
- Gráficos de dispersão: scatter()

E vários outros exemplos na [referência](#) da biblioteca! :)

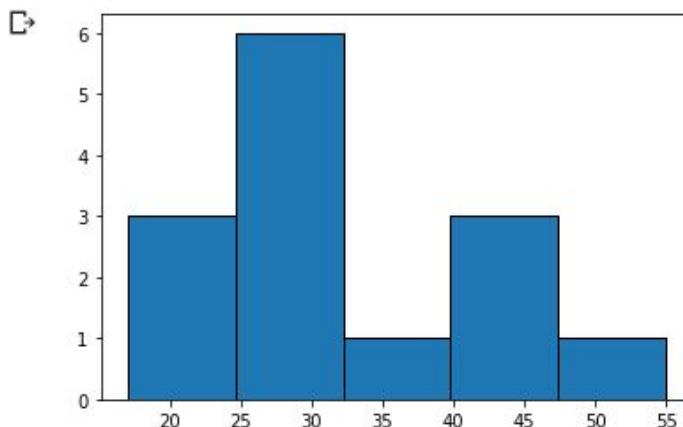
# Gráfico de Histograma

Representação apro

```
# lista para realizar o grafico
horas = [17, 20, 22, 25, 26, 27, 30, 31, 32, 38, 40, 40, 45, 55]

# inicializacao
fig, ax = plt.subplots(figsize = (6, 4))

# plot
ax.hist(horas, bins=5, edgecolor="black");
```



# Gráfico de Barra

Excelente alternativa para diferenciar categorias em seus valores.

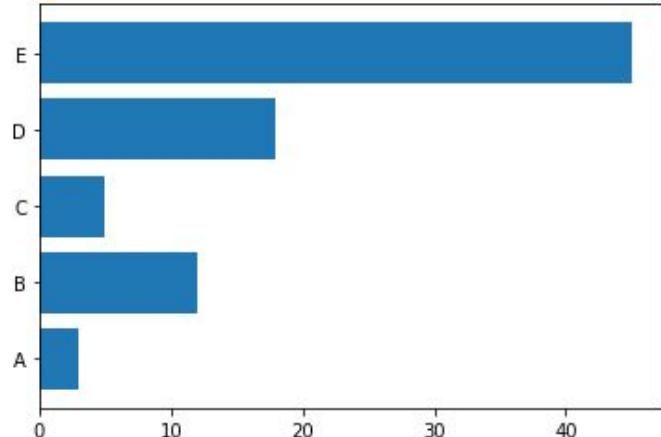
Em muitos dos casos, torna-se excelente alternativa para gráficos de pizza.

```
# lista com valores
valores = [3, 12, 5, 18, 45]
barras = ('A', 'B', 'C', 'D', 'E')
y_pos = np.arange(len(barras))

# Cria barras horizontais
plt.barh(y_pos, valores)

# Cria o nome da legenda no eixo x
plt.yticks(y_pos, barras)

# visualiza o grafico
plt.show()
```



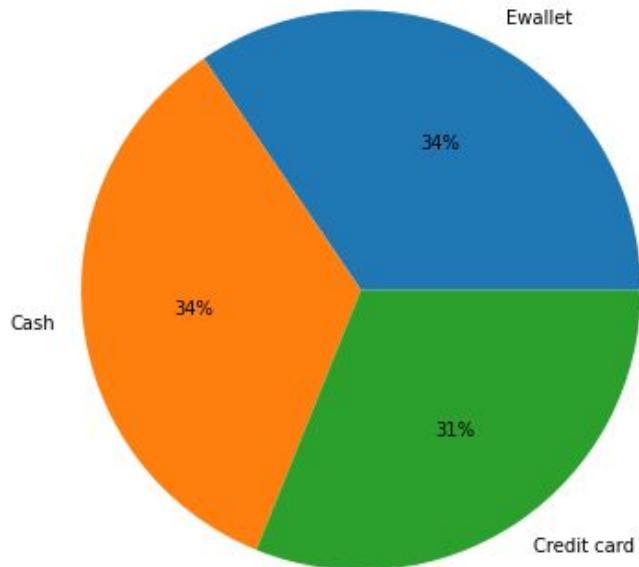
# Gráfico de Pizza

Os gráficos de pizza são usados para representar porcentagens.

Pense sempre duas vezes antes de usar um gráfico de pizza, talvez um de barra (barplot) seja suficiente.

```
# figure(figsize=(8, 6))
plot1 = plt.figure(figsize=(12,7))
gender = df.Payment.value_counts()
labels = df.Payment.unique()
plt.pie(gender,labels=labels,autopct='%.0f%%')
plt.title("Formas de pagamento utilizadas")
plt.show()
```

Formas de pagamento utilizadas



# Tabelão de Comandos



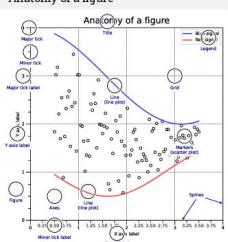
Cheat sheet Version 3.5.0

**Quick start**

```
X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')
fig.savefig('figure.pdf')
fig.show()
```

## Anatomy of a figure



## Subplots layout

```
subplots(s=(rows,cols,...)
fig, axes = plt.subplots(3, 3)
G = gridspec(rows,cols,...)
ax = G[0,1]
ax.inset_axes(extent)
d=make_axes_locatable(ax)
ax = d.new_horizontal('10%')
```

## Getting help

- matplotlib.org
- [github.com/matplotlib/matplotlib/issues](https://github.com/matplotlib/matplotlib/issues)
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- gitter.im/matplotlib
- [@Matplotlib users mailing list](https://twitter.com/matplotlib)

**Basic plots**

- plot([X],Y,[fmt],...)
- X,Y,fmt,color,marker,linestyle
- scatter(X,Y,...)
- X,Y,[sizes,[c]olors,marker,cmap]
- bar([x],height,...)
- x,height,width,bottom,align,color
- imshow(Z,...)
- Z,cmap,interpolation,extent,origin
- contour(f)(([X],[Y],Z,...)
- X,Y,Z,levels,colors,extent,origin
- pcolormesh([X],[Y],Z,...)
- X,Y,Z,vmin,vmax,cmap
- quiver([X],[Y],U,V,...)
- X,Y,U,V,C,units,angles
- pie(X,...)
- X,explode,labels,colors,radius
- text(x,y,text,...)
- x,y{text,va,ha,size,weight,transform}
- fill\_between(x)(...)
- X1,T1,T2,color,where

## Advanced plots

- step(X,Y,[fmt],...)
- X,Y,fmt,color,marker,where
- boxplot(X,...)
- X,notch,sym,bootstrap,widths
- errorbar(X,Y,xerr,yerr,...)
- X,Y,xerr,yerr,fmt
- hist(X,bins,...)
- X,bins,range,density,weights
- violinplot(...)
- D,positions,whisker,vert
- barbs([X],[Y],U,V,...)
- X,Y,U,V,C,length,pivot,sizes
- eventplot(positions,...)
- positions,orientation,lineoffsets
- hexbin(X,Y,C,...)
- X,Y,C,gribsizes,bins

**Scales**

- ax.set\_[xy]scale(scale,...)
- z any values
- linear
- log
- logit
- any values

**Projections**

- subplot(...,projection=p)
- p=polar'
- p=3d'
- p=orthographic()
- from cartopy.crs import Cartographic

**Lines**

- linestyle or ls
- capstyle or dash.capstyle
- "butt"
- "round"
- "projecting"

**Markers**

- marker
- markeredgecolor
- markerfacecolor
- markeredgewidth
- markersize
- markerstyle

**Colors**

- color
- color\_hex
- color\_name
- color\_hex\_alpha
- color\_hex\_alpha\_alpha

**Colormaps**

- plt.get\_cmap(name)
- Uniform
- Sequential
- Diverging
- Qualitative
- Cyclic

**Tick locators**

```
from matplotlib import ticker
ax.set_xscale('linear', major_locator=ticker.MaxNLocator())
ticker.NullLocator()
ticker.MultipleLocator(5)
ticker.FixedLocator([0, 5])
ticker.LinearLocator(base=10, n_ticks=3)
ticker.IndicatorLocator(base=10, offset=0.25)
ticker.MaxNLocator(n=4)
ticker.LogLocator(base=10, numticks=15)
```

**Tick formatters**

```
from matplotlib import ticker
ax.xaxis.set_major_formatter(formatter)
ticker.NullFormatter()
ticker.FixedFormatter(['zero','one','two','...'])
ticker.FuncFormatter(lambda x, pos: "%0.2f" % x)
ticker.FormatStrFormatter("%d")
ticker.ScalarFormatter()
ticker.StrMethodFormatter("%s")
```

**Ornaments**

- ax.legend(...)
- handles,labels,loc,title,frameon

**Legend**

**Event handling**

```
fig, ax = plt.subplots()
def on_click(event):
    print(event)
fig.canvas.mpl_connect('button_press_event', on_click)
```

**Text**

**Animation**

```
import matplotlib.animation as mpl
T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpl.FuncAnimation(
    plt.gcf(),
    animate,
    interval=5
)
plt.show()
```

**Styles**

```
plt.style.use(style)
```

**Quick reminder**

```
ax.grid()
ax.set_xlim(vmin, vmax)
ax.set_xlabel(label)
ax.set_xticks(ticks, [labels])
ax.set_xticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, ...)
ax.set_axis_on/off())
fig.subtitle(title)
fig.patch.set_alpha(alpha)
plt.gcf().patch.set_alpha(alpha)
mpl.rcParams['axes.linewidth']=1, ...
[fig].patch.set_alpha(alpha)
text=r'$\frac{e^{-i\pi f}}{(2^n)^2}$'
```

**Keyboard shortcuts**

ctrl + S	Save	ctrl + W	Close plot
F	Reset view	F	Fullscreen 1/1
V	View forward	B	View back
P	Pan view	O	Zoom to rect
X	pan/zoom	Y	pan/zoom
G	Minor grid 0/1	G	Major grid 0/1
L	X axis log/linear	L	Y axis log/linear

**Ten simple rules**

1. Know Your Audience
2. Identify Your Message
3. Adapt the Figure
4. Captions Are Not Optional
5. Be True to the Defaults
6. Use Cool Effects Wisely
7. Do Not Mislead the Reader
8. Avoid "Chartjunk"
9. Message Trumps Beauty
10. Get the Right Tool

# Importação e Uso

- Recomendo sempre definir o tamanho da figura com o **figsize**
- Coloque **legendas e título**, isso facilita o reconhecimento posteriormente

```
import matplotlib.pyplot as plt          #importa a lib

plot1 = plt.figure(figsize=(12,7))
gender = df.Payment.value_counts()
labels = df.Payment.unique()
plt.pie(gender,labels=labels,autopct='%.0f%%')
plt.title("Formas de pagamento utilizadas")
plt.legend(title='Gender')
plt.show()
```

# Como criar um gráfico

---



```
fig, ax = plt.subplots()  
ax.plot(x, y)  
plt.show()
```

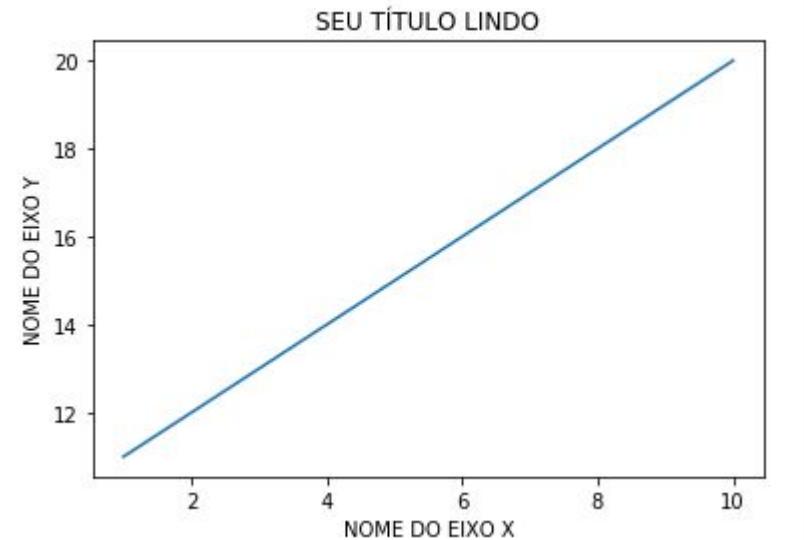


```
plt.plot(x, y)
```

# Título e nome dos eixos



```
plt.plot(x, y)
plt.title('SEU TÍTULO LINDO')
plt.xlabel('NOME DO EIXO X')
plt.ylabel('NOME DO EIXO Y')
plt.show()
```



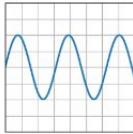
# Referências: Tutorial Matplotlib

[W3 Schools](#)

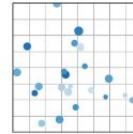
[Tipos de Gráficos](#): referência da Matplotlib

## Basic

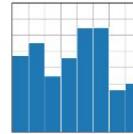
Basic plot types, usually y versus x.



plot(x, y)



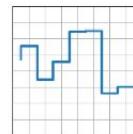
scatter(x, y)



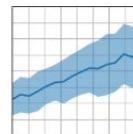
bar(x, height)



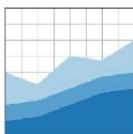
stem(x, y)



step(x, y)



fill\_between(x, y1, y2)



stackplot(x, y)

# Matplotlib Documentação

☰ README.rst

## Matplotlib tutorial

Nicolas P. Rougier

DOI [10.5281/zenodo.28747](https://doi.org/10.5281/zenodo.28747)

Table of Contents

- [Introduction](#)
- [Simple plot](#)
- [Figures, Subplots, Axes and Ticks](#)
- [Animation](#)
- [Other Types of Plots](#)
- [Beyond this tutorial](#)
- [Quick references](#)

Sources are available from [github](#)



[Plot types](#) [Examples](#) [Tutorials](#) [Reference](#) [User guide](#) [Develop](#) [Releases](#)

## Users guide

### General

- Getting started
  - [Installation quick-start](#)
  - [Draw a first plot](#)
  - [Where to go next](#)
- Installation
  - [Installing an official release](#)
  - [Third-party distributions](#)
  - [Installing a nightly build](#)
  - [Installing from source](#)
  - [Installing for development](#)
  - [Frequently asked questions](#)



# Seaborn

# Seaborn

---

Biblioteca de visualização de dados baseada no Matplotlib que fornece interface de alto nível para desenhar gráficos estatísticos de forma atraente.

## Diferença **Matplotlib x Seaborn**

- Matplotlib: realize plots básicos.
- Seaborn: variedade de padrões, com menos complexidade e sintaxe (API alto nível).



# Importação e Uso

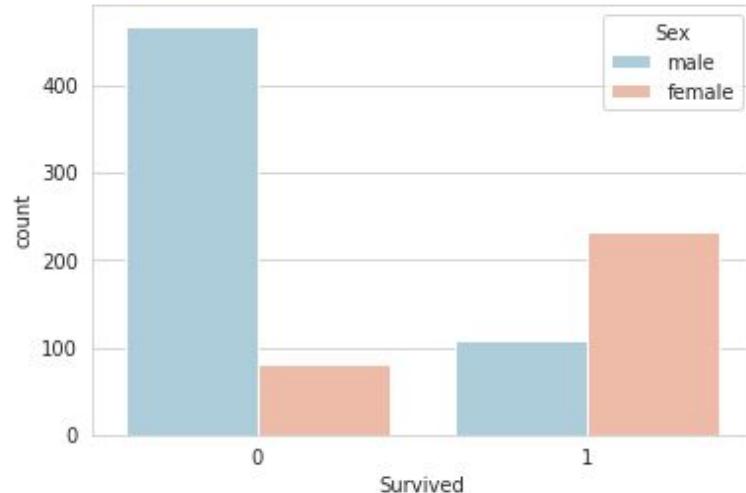
- Recomendo sempre definir o tamanho da figura com o figsize
- Coloque legendas e título, isso facilita o reconhecimento posteriormente

```
import seaborn as sns

plt.figure(figsize=(10,6))    # defina o tamanho de uma figura
plot2 = sns.heatmap(titanic.isnull(), yticklabels=False, cmap="YlOrBr")
plot2.set_xlabel("legenda eixo x", fontsize=16)
plot2.set_ylabel("legenda eixo y", fontsize=16)
```

# Exploração dos Dados

- Contagem



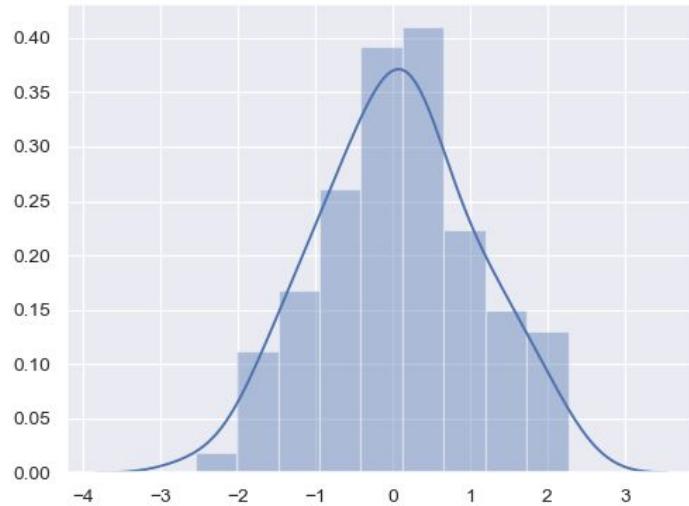
```
# Exibe um gráfico de contagem, configurando antes para exibir o grid
```

```
sns.set_style('whitegrid')
```

```
sns.countplot(x='Survived', hue='Sex', data=df, palette='RdBu_r')
```

# Histograma

- Representação aproximada da distribuição de dados numéricos ou categóricos



```
# Show a default plot with a kernel density estimate and  
# histogram with bin size determined automatically  
  
import seaborn as sns, numpy as np  
sns.histplot(x)
```

# Gráficos importantes para AED\*

- Descobrir a distribuição dos dados
- Countplots
- Boxplots
- Swarmplots
- Correlação

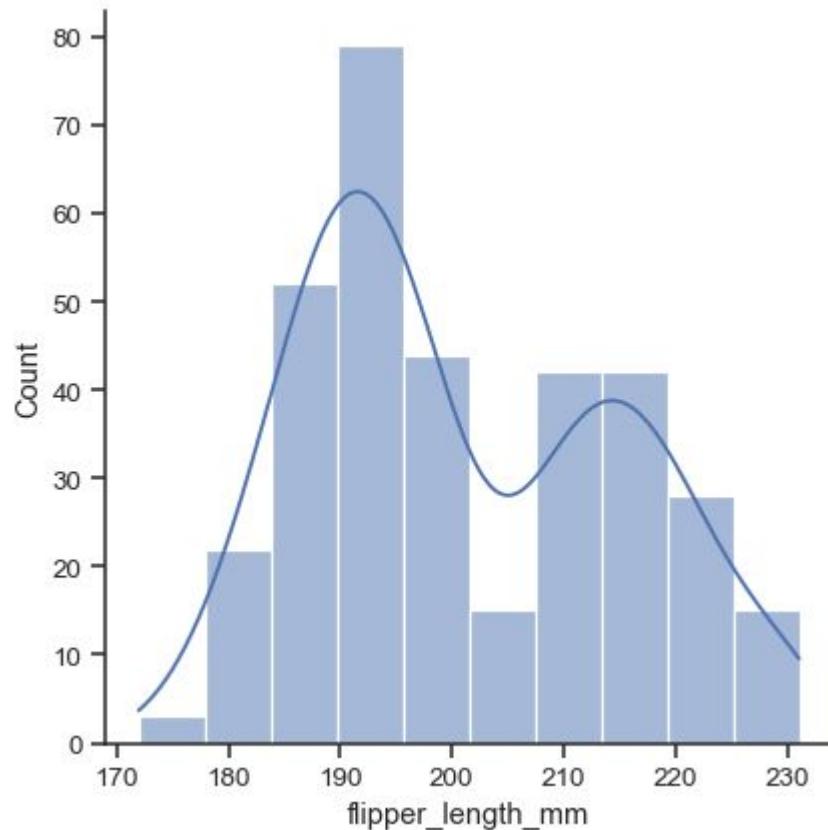


# Displot

```
sns.displot(data=penguins, x="flipper_length_mm", kde=True)
```

O que o gráfico pode nos passar:

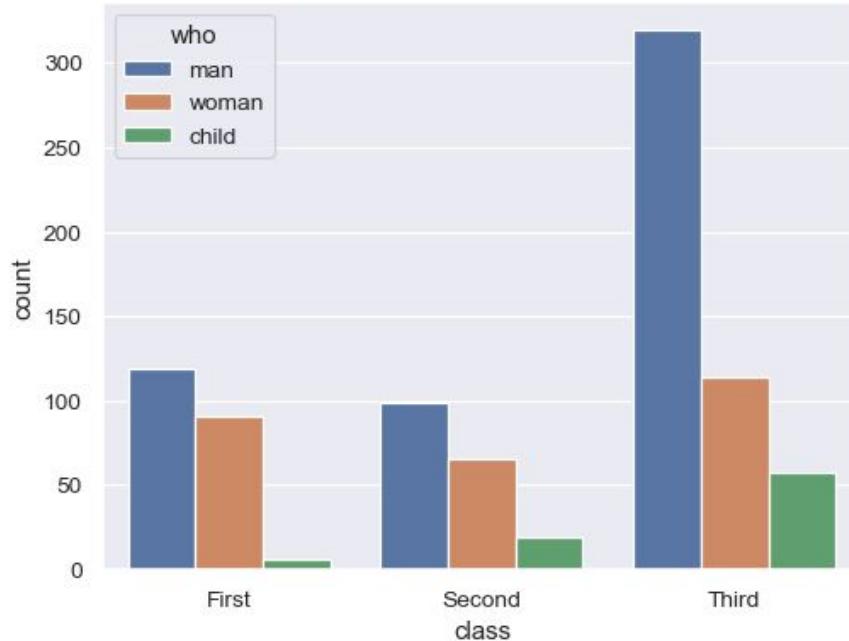
- Distribuição normal ou não
- Concentração de dados em regiões



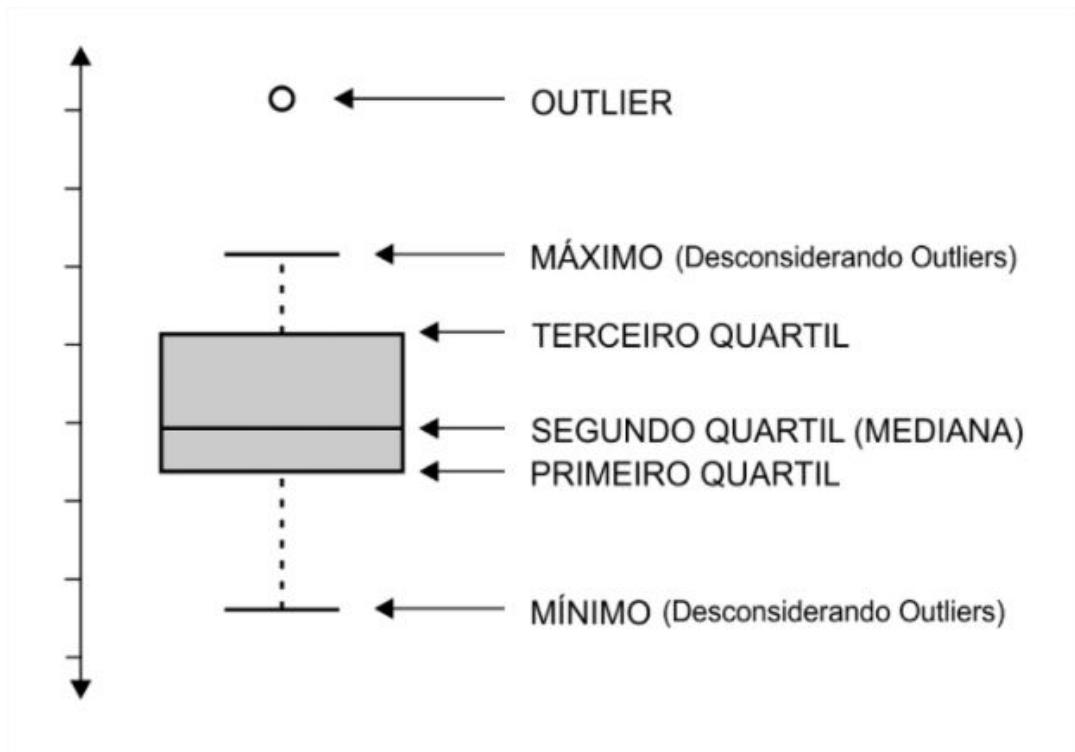
# Countplot

```
ax = sns.countplot(x="class", hue="who", data=titanic)
```

Outro gráfico simples mas muito útil que pode ser utilizado é o countplot.



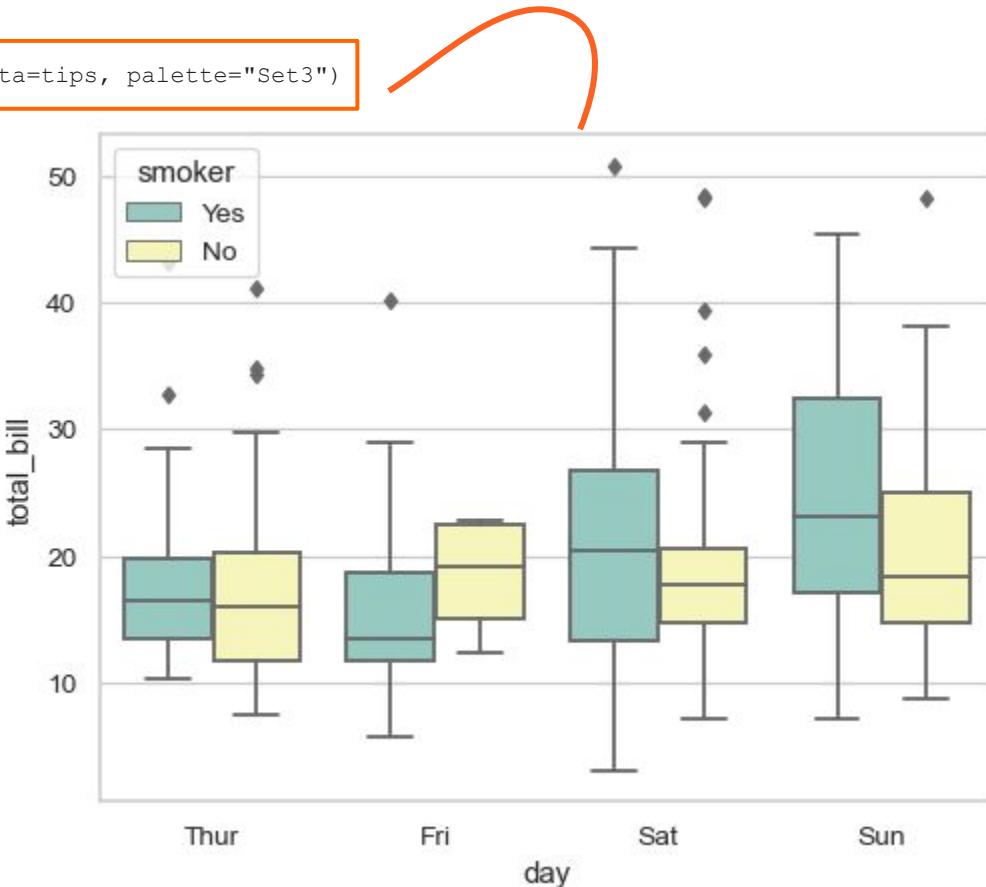
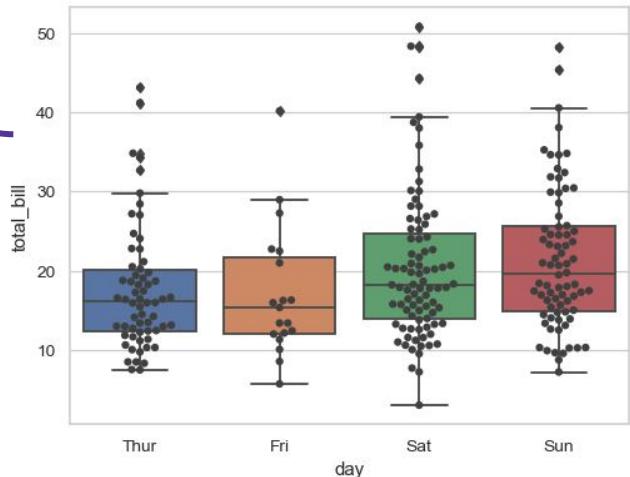
# Boxplot



# Boxplot no Seaborn

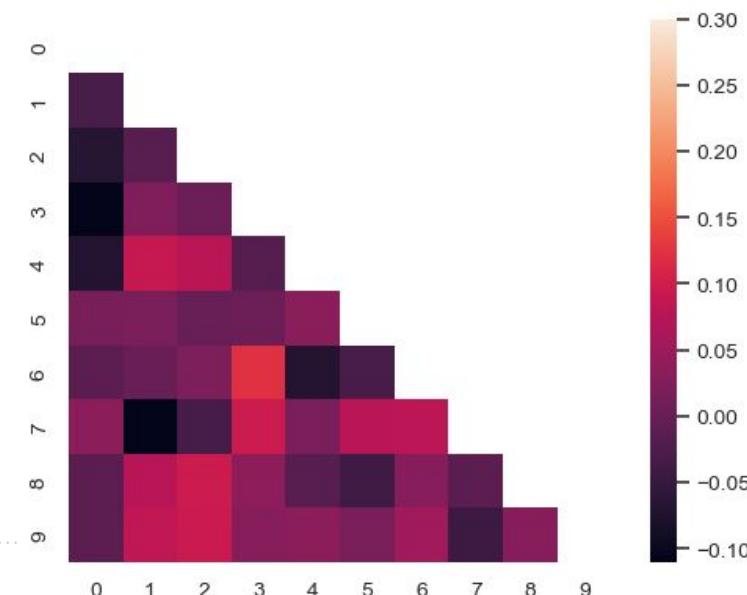
```
ax = sns.boxplot(x="day", y="total_bill", hue="smoker", data=tips, palette="Set3")
```

```
ax = sns.boxplot(x="day", y="total_bill", data=tips)
ax = sns.swarmplot(x="day", y="total_bill", data=tips, color=".25")
```

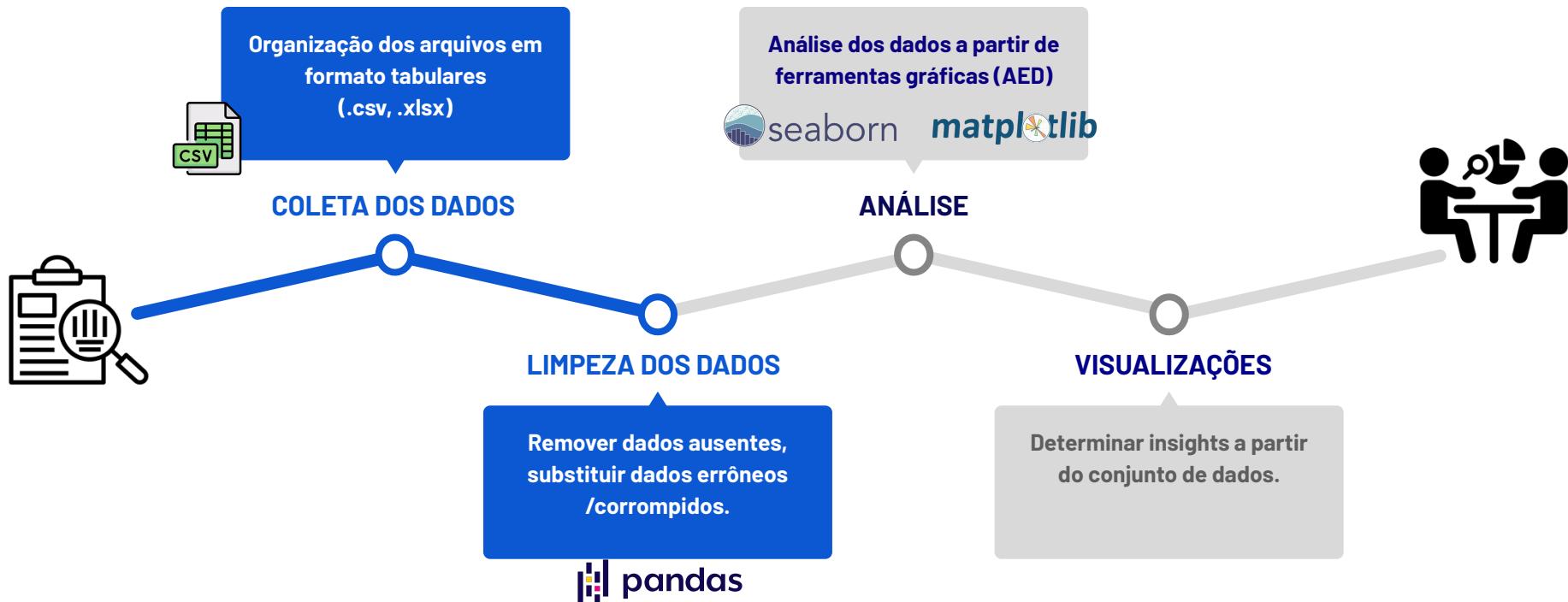


# Correlação

```
corr = np.corrcoef(np.random.randn(10, 200))
>>> mask = np.zeros_like(corr)
>>> mask[np.triu_indices_from(mask)] = True
>>> with sns.axes_style("white"):
...     f, ax = plt.subplots(figsize=(7, 5))
...     ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True)
```



# Análise de Dados: o resumo



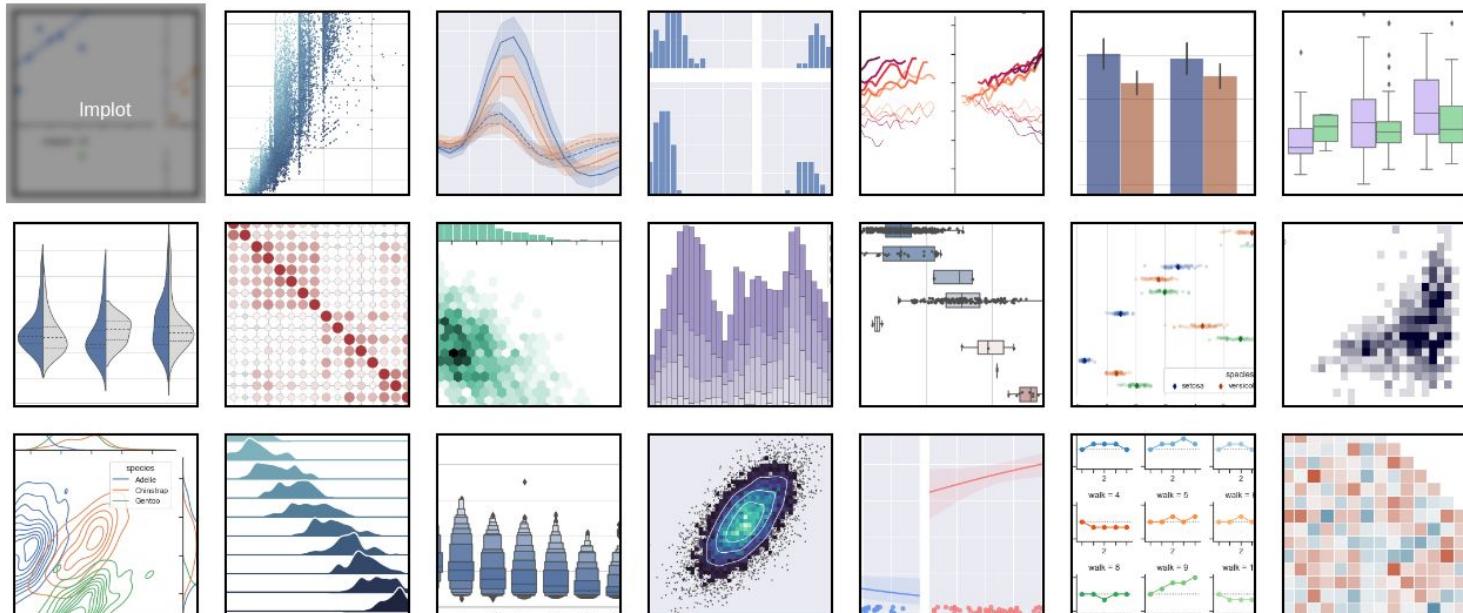
# Seaborn Example Gallery



Installing [Gallery](#) Tutorial API Releases Citing FAQ



## Example gallery



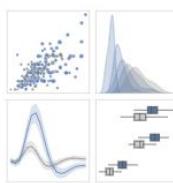
# Seaborn Documentação



Installing Gallery **Tutorial** API Releases Citing FAQ

An introduction to seaborn  
Overview of seaborn plotting functions  
Data structures accepted by seaborn  
The `seaborn.objects` interface  
Properties of Mark objects  
Visualizing statistical relationships  
Visualizing distributions of data  
Visualizing categorical data  
Statistical estimation and error bars  
Estimating regression fits  
Building structured multi-plot grids  
Controlling figure aesthetics  
Choosing color palettes

## User guide and tutorial



An introduction to seaborn  
A high-level API for statistical graphics  
Multivariate views on complex datasets  
Opinionated defaults and flexible customization

## API Overview #



Overview of seaborn plotting functions  
Similar functions for similar tasks  
Figure-level vs. axes-level functions  
Combining multiple views on the data

# Plotly

# Plotly: a biblioteca gráfica interativa

---

- Biblioteca de visualização de dados para **Python**, Javascript e R.
- Suporte para **criação de dashboards** baseado em dados SQL
- Permite uso em aplicações e alta capacidade de **customização**

Acesse em [plotly.com/python](https://plotly.com/python)



# Uso básico do Plotly

---

Alguns métodos (**disponíveis para Series e DataFrames**) interessantes são:

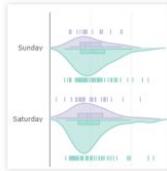
- Gráficos de histograma: **hist()**
- Gráficos de área: **area()**
- Gráficos de pizza: **pie()**
- Gráficos de barra: **bar()**
- Gráficos de dispersão: **scatter()**

E vários outros exemplos na [referência](#) da biblioteca! :)

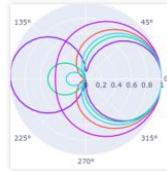


## Fundamentals

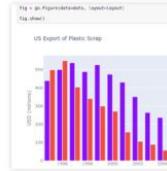
[More Fundamentals »](#)



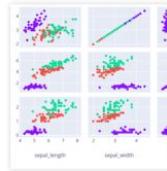
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



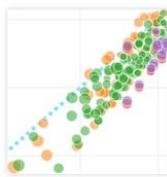
Plotly Express



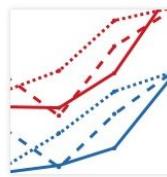
Analytical Apps with Dash

## Basic Charts

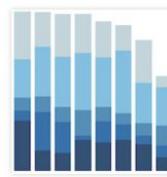
[More Basic Charts »](#)



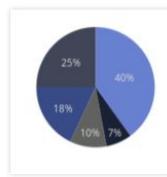
Scatter Plots



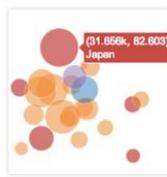
Line Charts



Bar Charts



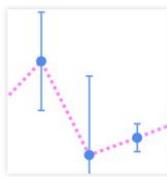
Pie Charts



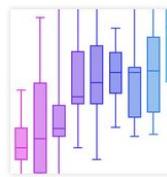
Bubble Charts

## Statistical Charts

[More Statistical Charts »](#)



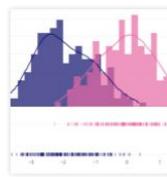
Error Bars



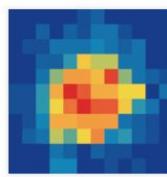
Box Plots



Histograms



Distplots



2D Histograms



# Criando um gráfico no Plotly em três etapas



```
# Criando gráfico
trace = go.Scatter(x = df['coluna_x'],
                    y = df['coluna_y'],
                    mode = 'markers')

# Armazenando gráfico em uma lista
data = [trace]

# Criando Layout
layout = go.Layout(title='Titulo do Grafico',
                    yaxis={'titulo do eixo y'},
                    xaxis={'titulo do eixo x'})

# Criando figura que será exibida
fig = go.Figure(data=data, layout=layout)

# Exibindo figura/gráfico
py.iplot(fig)
```



# Criando um gráfico no Plotly em três etapas



```
# Criando gráfico
trace = go.Scatter(x = df['coluna_x'],
                    y = df['coluna_y'],
                    mode = 'markers')

# Armazenando gráfico em uma lista
data = [trace]

# Criando Layout
layout = go.Layout(title='Titulo do Grafico',
                    yaxis={'titulo do eixo y'},
                    xaxis={'titulo do eixo x'})

# Criando figura que será exibida
fig = go.Figure(data=data, layout=layout)

# Exibindo figura/gráfico
py.iplot(fig)
```

1

Definição do tipo de Gráfico usando Plotly



# Criando um gráfico no Plotly em três etapas



```
# Criando gráfico
trace = go.Scatter(x = df['coluna_x'],
                    y = df['coluna_y'],
                    mode = 'markers')

# Armazenando gráfico em uma lista
data = [trace]

# Criando Layout
layout = go.Layout(title='Titulo do Grafico',
                    yaxis={'titulo do eixo y'},
                    xaxis={'titulo do eixo x'})

# Criando figura que será exibida
fig = go.Figure(data=data, layout=layout)

# Exibindo figura/gráfico
py.iplot(fig)
```

2

Criação de Layout a partir de GraphicObject

# Criando um gráfico no Plotly em três etapas

```
# Criando gráfico
trace = go.Scatter(x = df['coluna_x'],
                    y = df['coluna_y'],
                    mode = 'markers')

# Armazenando gráfico em uma lista
data = [trace]

# Criando Layout
layout = go.Layout(title='Titulo do Grafico',
                     yaxis={'titulo do eixo y'},
                     xaxis={'titulo do eixo x'})

# Criando figura que será exibida
fig = go.Figure(data=data, layout=layout)

# Exibindo figura/gráfico
py.iplot(fig)
```

3

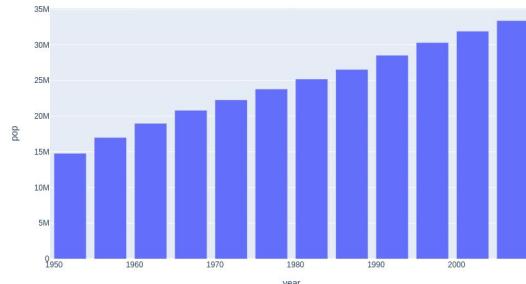
Exibição da Figura



# Plotly Express

O módulo `plotly.express` (importado como `px`) contém funções que podem criar figuras inteiras com único comando. Plotly Express é uma parte interna da biblioteca Plotly e é o ponto de partida recomendado para a criação de figuras mais comuns, também integrando com Dashboards.

```
import plotly.express as px
data_canada = px.data.gapminder().query("country == 'Canada'")
fig = px.bar(data_canada, x='year', y='pop')
fig.show()
```



# Importação e Uso do Plotly

```
import matplotlib.pyplot as plt          #importa a lib

import plotly.graph_objects as go
import plotly.offline as po
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.express as px
import plotly.figure_factory as ff
```





# Hands On!

**Plotly**

# Ydata Profiling



# Ferramenta para geração de AED de forma automática

- Suponha uma ferramenta para fazer uma análise exploratória dos Dados (AED) de forma automática. Uma solução seria tentar o **chatGPT**, certo?



faca a análise exploratoria dos dados do arquivo .csv do titanic



Claro! Aqui está um exemplo de análise exploratória dos dados do arquivo CSV do Titanic:

```
python
import pandas as pd

# Carregar o dataset Titanic
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
df = pd.read_csv(url)
```



```
# Visualizar as primeiras linhas do dataset
print(df.head())

# Verificar as informações gerais do dataset
print(df.info())

# Calcular estatísticas descritivas das colunas numéricas
print(df.describe())

# Verificar a contagem de valores únicos em cada coluna categórica
print(df.nunique())

# Verificar a contagem de valores ausentes em cada coluna
print(df.isnull().sum())

# Fazer uma análise de sobrevivência por gênero
survived_gender = df.groupby('Sex')['Survived'].mean()
print(survived_gender)

# Fazer uma análise de sobrevivência por classe
survived_class = df.groupby('Pclass')['Survived'].mean()
print(survived_class)
```

**Até pode fazer um certo sentido a resposta, mas cade a análise exploratória? Os gráficos...**

# Ydata Profiling (Pandas Profiling)

---

- O **pandas-profiling** é uma biblioteca Python que gera perfis automáticos de dados para análise exploratória de datasets. Ela fornece uma visão abrangente e detalhada das características e propriedades do conjunto de dados.

Alguns gráficos apresentados são:

- Correlacao
- Histogramas
- Tabelas de frequêcia
- Matriz de dispersao
- Análise de variáveis fortemente correlacionadas



→ Além de apresentar uma visão geral dos dados, dos tipos, valores únicos, ausentes e estatísticas descritivas.

Disponível em <https://github.com/ydataai/ydata-profiling>



# Python Graph Gallery

# Python Graph Gallery

## The Python Graph Gallery



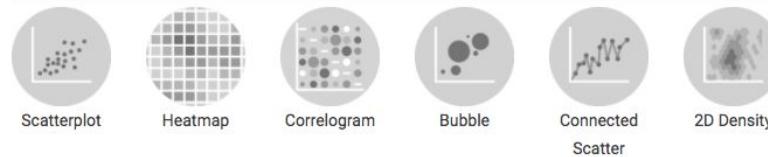
Welcome to the Python Graph Gallery, a collection of hundreds of charts made with [Python](#). Charts are organized in about 40 sections and always come with their associated reproducible code. They are mostly made with [Matplotlib](#) and [Seaborn](#) but other library like [Plotly](#) are sometimes used.

- [Python Graph Gallery](#)

### Distribution



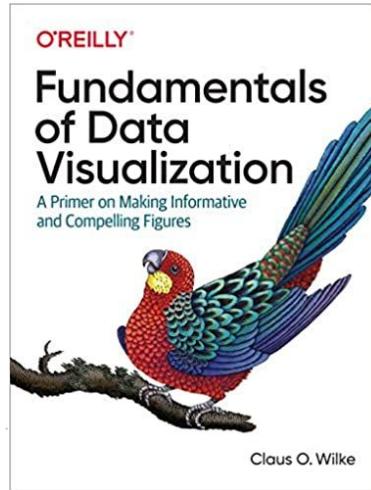
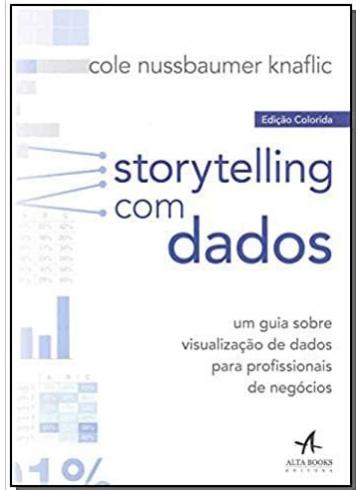
### Correlation



# Referências

---

- Knafllic, Cole Nussbaumer. **Storytelling com dados: um guia sobre visualização de dados para profissionais de negócios.** Alta Books, 2019.
- Wilke, Claus O. **Fundamentals of data visualization: a primer on making informative and compelling figures.** O'Reilly Media, 2019.



# Tutoriais e Links

---

- [Python Graph Gallery](#)
- [Data Analysis with Python: Zero to Pandas - Course Project Guidelines](#)
- Kaggle





c . e . s . a . r  
s c h o l