*Visualizações em Python (Parte I)*

A visualização dos dados é um aspecto fundamental em qualquer projeto de Data Science, tanto como ferramenta de análise quanto para a comunicação efetiva dos resultados. Para esta tarefa, existem várias bibliotecas de visualização em Python, entre as quais podemos citar: *Matplotlib, Seaborn, Plotly* e *Bokeh*. Nesta aula, focaremos especialmente na **Matplotlib** por sua versatilidade, simplicidade e facilidade de uso. Para representar graficamente as distribuições de variáveis ​​categóricas, faremos uso da **Seaborn**, uma vez que fornece um conjunto de comandos que facilitam esta tarefa.

Introdução a Matplotlib

Matplotlib é a biblioteca de visualização mais usada no ambiente Python. Além da facilidade de uso, permite um alto nível de personalização dos gráficos. Como as outras bibliotecas com as quais estamos trabalhando, a Matplotlib é um código aberto. Dada a sua grande flexibilidade de uso, é também a base sobre a qual outras bibliotecas foram construídas, como a Seaborn, que veremos mais adiante.

Primeiro, importamos as bibliotecas que usaremos durante a aula

| import numpy as np import pandas as pd  # --- Específicas para os gráficos import matplotlib as mpl import matplotlib.pyplot as plt import seaborn as sns # -------------------------------- |
| --- |

Se acontece um erro ao executar as importações, é provável que as bibliotecas necessárias não estejam instaladas. Nesse caso, elas podem ser instaladas executando em uma célula de código

| !conda install matplotlib !conda install seaborn |
| --- |

ou então do Terminal executando

| conda install matplotlib conda install seaborn |
| --- |

A paleta de cores que a matplotlib traz por padrão não é muito estética, mas podemos facilmente mudar o estilo com a seguinte linha:

| mpl.style.use('bmh') # bmh é o nome do estilo que usaremos |
| --- |

Uma lista de estilos disponíveis pode ser consultada [neste link](https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html).

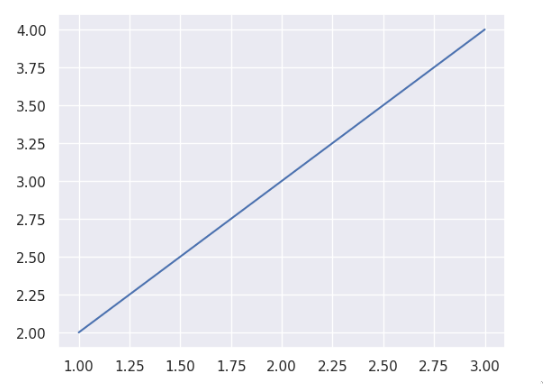
Interfaces

Existem duas maneiras de gerar gráficos com a Matplotlib: usando a *interface baseada em estado* ou usando *a interface* orientada a objetos. Embora a primeira opção possa parecer mais simples no início, a segunda opção dá mais controle sobre os gráficos que geramos, o que no final é vantajoso à medida que nosso código cresce e precisamos gerar um número maior de gráficos.

A interface baseada em estado foi criada com inspiração no MATLAB, um ambiente de desenvolvimento voltado para a ciência e linguagem de programação. Essa interface fornece compatibilidade com programadores acostumados a esse ambiente de programação. Para quem trabalha com Python, ou se dedica à programação orientada a objetos em geral, a interface orientada a objetos oferece maior controle e flexibilidade no uso de gráficos.

Com a interface baseada em estado, uma linha conectando os pontos (1, 2) e (3, 4) pode ser representada graficamente da seguinte forma:

| plt.plot([1, 3], [2, 4]) |
| --- |



Considerando que gerar o mesmo gráfico com a interface orientada a objetos:

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

Como pode ser visto, a função plt.subplots () retorna dois objetos, um objeto do tipo Figure e um objeto do tipo Axes, que atribuímos às variáveis ​​fig e ax respectivamente. Posteriormente manipularemos o gráfico referente ao referido objeto Axes através de ax.

Passamos duas listas como parâmetros para a função plot (). O primeiro contém as coordenadas do eixo x dos pontos que queremos unir e o segundo contém as coordenadas do eixo y.

Misturar o uso dessas interfaces pode causar confusão ao começar a programar em Python. Portanto, nos limitaremos a usar apenas a segunda opção, que também recomendamos por ser a mais utilizada.

Comandos básicos

Vamos continuar com o mais importante: contextualizar o gráfico. Ao fazer um gráfico, é sempre uma boa ideia incluir informações sobre o que está sendo exibido. Para isso, recomenda-se adicionar um título, rotular os eixos e inserir uma legenda que informe o que representa cada elemento traçado. Ao rotular os eixos, adicione a unidade de medida, se possível.

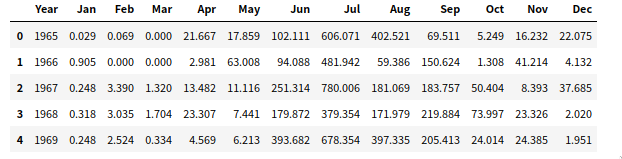
* **Para etiquetar os eixos**: os métodos *set\_xlabel* () e *set\_ylabel* () do objeto ax recebem como parâmetro o rótulo dos eixos x e y respectivamente. A forma de usá-lo é a seguinte: ax.set\_xlabel ('etiqueta do eixo X').
* **Para adicionar um título:** use o método *set\_title* () do objeto ax. (ax.set\_title())
* **Para adicionar uma legenda:** use o método *legend()* do objeto ax. (ax.legend())

Exemplo de uso

Como dados de teste, serão usados ​​os dados de precipitação da aula anterior. Este conjunto de dados pode ser baixado [neste link](https://www.kaggle.com/abhishekmamidi/precipitation-data-of-pune-from-1965-to-2002).

df\_chuvas = pd.read\_csv("<caminho>/pune\_1965\_to\_2002.csv")

df\_chuvas.head()



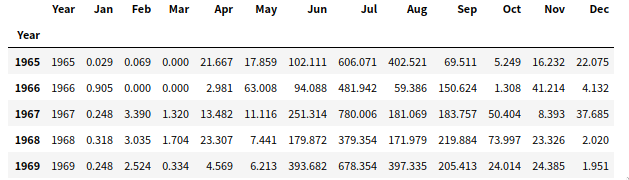
O método .head() retorna as primeiras 5 fileiras do data frame, que é útil para visualizar os dados.

Por conveniência, é preferível que o índice do data frame seja a coluna Year. Isso pode ser feito da seguinte forma:

Primeiro, a coluna Year é selecionada com o método df\_chuvas ["Year"]. Isso retorna uma *Série de Pandas*, como vimos antes. Lembre-se de que um objeto Séries nada mais é do que uma estrutura de dados análoga a um *Numpy Array*, mas com a adição de um índice, que coincide com o índice do data frame.

Podemos atribuir a saída deste método diretamente à propriedade de índice do quadro de dados da seguinte forma

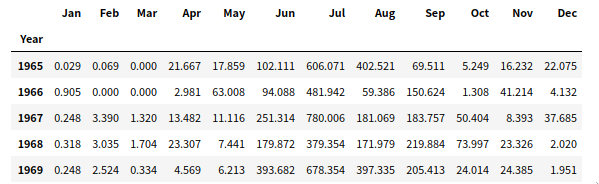
| df\_chuvas.index = df\_chuvas["Year"] df\_chuvas.head() |
| --- |



Embora o índice tenha sido atualizado, a coluna Year em excesso deve ser removida. Observe que agora a coluna é um dado e também um índice. O método .drop () do Pandas permite que você elimine linhas ou colunas especificadas e retorna um novo quadro de dados sem essas linhas ou colunas. Para atualizar o data frame, é então necessário atribuí-lo à variável df\_*chuvias*.

df\_chuvas = df\_chuvas.drop("Year", axis="columns")

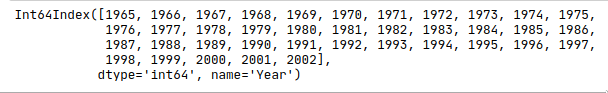
df\_chuvas.head()



Com isso, o data frame está pronto para funcionar.

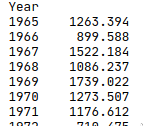
Conforme mencionado acima, o método ax.plot () requer como parâmetros as coordenadas dos pontos que a linha deve unir. O ano será traçado no eixo horizontal e a precipitação acumulada em cada ano no eixo vertical. Portanto, definimos as variáveis

| x = df\_chuvas.index print(x) |
| --- |



e calculamos as precipitações acumuladas em cada ano com o método .sum()

| y = df\_chuvas.sum(axis="columns") print(y) |
| --- |



Passando x e y para o método ax.plot() e adicionando o título e etiquetas:

| fig, ax = plt.subplots(figsize=(12, 4))  ax.plot(x, y, label='Precipitações acumuladas') ax.set\_xlabel('Ano') # Etiqueta eixo x ax.set\_ylabel('Precipitação acumulada (mm.)') # Etiqueta eixo y ax.set\_title('Precipitações acumuladas segundo o ano') # Título ax.legend() # Inserir a legenda |
| --- |



**Observações:**

* O parâmetro *figsize* na chamada a *plt*.*subplots* () permite alterar o tamanho do gráfico. Nesse caso, é estabelecido um tamanho de 12 unidades no eixo x e 4 unidades no eixo y. Essa proporção é mais adequada para séries temporais.
* Observe o parâmetro *label* no método ax.plot(). Este é o texto que será inserido na legenda
* O método ax.legend() insere automaticamente a legenda em algum canto que não cobre a série temporal. Você pode modificar este comportamento para que seja inserido em outro canto usando o *parâmetro loc*. Para ver as opções para este e outros parâmetros, você pode abrir uma nova célula e executar 'ax.legend?' Isto irá abrir uma pequena aba com a documentação oficial da referida função, que contém uma breve descrição, os parâmetros de entrada, os valores permitidos para cada um deles e os objetos que ela retorna. Se a qualquer momento você não conseguir lembrar quais parâmetros uma determinada função recebe, adicione um ponto de interrogação no final dela e execute a célula.

No gráfico anterior observa-se que existe uma margem vazia no início e no final da série temporal, para recortá-la pode ser adicionada a seguinte linha:

| ax.set\_xlim(df\_chuvas.index[0], df\_chuvas.index[-1]) |
| --- |

Vamos relembrar abaixo como acessamos os elementos de uma estrutura em Python:

*df\_chuvas.index[0]* devolve o primeiro elemento do índice, isto é, 1965.

*df\_chuvas.index[-1]* devolve o último elemento do índice, isto é, 2002.

Por outro lado, o método *ax.set\_xlim()* permite que você defina os limites do gráfico no eixo x. Desta forma, forçamos matplotlib a representar graficamente apenas de 1965 a 2002.

O gráfico resultante é então:



Por outro lado, também é possível definir limites no eixo y com o método *ax.set\_ylim().*

Exportando os gráficos

A Matplotlib permite que você salve as figuras geradas com uma única linha de código:

| fig.savefig("precipitaciones\_año.pdf") |
| --- |

Este método recebe como parâmetro o nome do arquivo junto com sua extensão, que determina o formato em que o arquivo será exportado. Os formatos mais usados ​​que a matplotlib suporta são jpeg, jpg, png, pdf, svg.

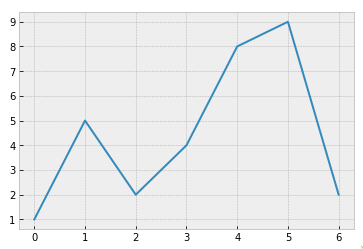
Uma vez executado este código, o gráfico será salvo com uma extensão PDF na mesma pasta onde o notebook está armazenado.

Gráficos comuns

Gráficos de linhas

Os gráficos de linha são adequados para exibir dados com sequenciamento de tempo, como séries temporais. Como no exemplo anterior, para gerar um gráfico de linha, basta passar como parâmetro as coordenadas dos pontos que a linha deve unir. Essas podem ser listas de Python ou Numpy arrays alternadamente. Se uma única matriz for fornecida para a função plot (), Matplotlib assume que ela se refere às coordenadas no eixo *y*. *Por exemplo, no gráfico*:

| fig, ax = plt.subplots() ax.plot([1, 5, 2, 4, 8, 9, 2]) |
| --- |

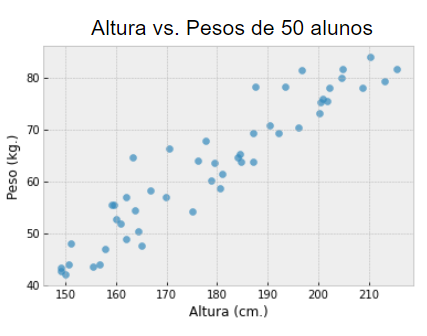
**

A Matplotlib assume implicitamente como coordenadas no eixo x na matriz [0, 1, 2, 3, 4, 5, 6].

Gráficos de dispersão

Os gráficos de dispersão são especialmente úteis quando você tem uma grande quantidade de dados numéricos emparelhados. Eles permitem ver rapidamente a existência (ou não) de uma relação entre as variáveis ​​que são representadas graficamente e, também, mostram a presença de subgrupos dentro do conjunto de dados que é representado graficamente. Por exemplo, considere medir a altura e o peso de 50 alunos(as). Intuitivamente, pode-se pensar que, em geral, pessoas mais altas são mais pesadas do que pessoas mais baixas, o que implicaria em uma correlação positiva entre as variáveis. Se essa hipótese for verdadeira, essa relação positiva deve ser refletida no gráfico como uma nuvem de pontos concentrada na diagonal ascendente do gráfico.

| pesos = [42.8, 43.3, 42. , 44. , 44.1, 43.5, 48.1, 48.9, 47.7,46.9,50.4,  52.7, 51.8, 54.5, 54.2, 56.9, 55.4, 55.5, 57.1, 58.3, 63.7, 58.8,  64.6, 60.2, 64. , 63.8, 61.4, 66.3, 64.7, 63.9, 69.3, 67.9, 65.2,  70.8, 70.5, 69.3, 75.3, 75.5, 78.2, 78. , 73.2, 78. , 80.1, 78.2,  76. , 81.5, 79.4, 81.8, 81.8, 84.1] alturas = [149. , 149. , 149.9, 156.8, 150.6, 155.4, 151. , 162. , 165.,  157.8, 164.4, 160.1, 160.8, 163.8, 175.2, 162. , 159.5, 159.2,  169.8, 166.7, 179.4, 180.6, 163.3, 178.8, 176.3, 184.8, 181. ,  170.5, 184.1, 187.1, 187.1, 177.7, 184.5, 190.3, 196. , 192.1,  200.4, 201.8, 187.5, 202.1, 200.3, 208.8, 204.6, 193.5, 200.9,  196.8, 213.1, 204.8, 215.5, 210.2]  fig, ax = plt.subplots() ax.scatter(alturas, pesos, alpha=0.7) ax.set\_title('Altura vs. Peso de 50 alunos') ax.set\_xlabel('Altura (cm.)') ax.set\_ylabel('Peso (kg.)') |
| --- |
|  |

****

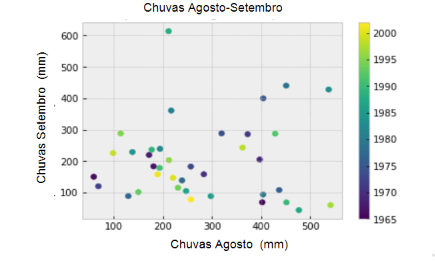
**Observação:**

* O conjunto de dados não precisa ser classificado em ordem crescente para fazer um gráfico de pontos. O que importa é que os elementos tenham a mesma ordem em relação ao que representam. Por exemplo, se o primeiro item na lista de peso for o peso de John, então o primeiro item na lista de altura deve ser a altura de John e não de outra pessoa. Da mesma forma, os valores de peso e altura devem ser pareados para os 49 alunos(as) restantes.
* O parâmetro *alfa* representa a transparência dos pontos e deve ser um valor maior que 0 e menor ou igual a 1. Quanto menor o alfa, maior a transparência. Um ponto completamente sólido possui alfa = 1.

Se tivermos um grande número de medidas, é aconselhável estabelecer um certo grau de transparência nesses tipos de gráficos, pois os pontos tendem a se cobrir. Além disso, ao estabelecer certa transparência é mais claro identificar as regiões com maior concentração de pontos.

Agora, as medições de precipitação serão comparadas em dois meses consecutivos ao longo dos anos. As variáveis ​​correspondentes podem ser chuva em agosto e chuva em setembro. Em seguida, procedemos à seleção das referidas colunas de interesse com os métodos *df\_chuvas ['Aug'] e df\_chuvas ['Sep']*, que podem ser passados ​​diretamente para o método *ax.scatter()*.

| fig, ax = plt.subplots()  mapeamento\_cores = ax.scatter(df\_chuvas['Aug'], df\_chuvas['Sep'], c=df\_chuvas.index) fig.colorbar(mapeamento\_cores) # Adiciona a barra de cores  ax.set\_title('Precipitações Agosto-Setembro') ax.set\_xlabel('Precipitações em Agosto (mm.)') ax.set\_ylabel('Precipitações em Setembro (mm.)') |
| --- |



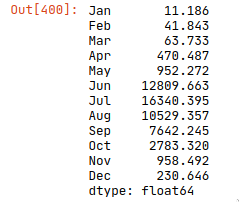
Observa-se então uma nuvem de pontos muito mais dispersa do que no caso peso-altura anterior, o que significa que a relação entre essas variáveis ​​é mais fraca.

**Observação:** o parâmetro 'c' na chamada a ax.scatter(), permite associar uma cor a cada ponto. Neste caso, as medições mais recentes foram marcadas com cores claras e as mais antigas com cores escuras. O método ax.scatter() também recebe um parâmetro 's' que permite associar um determinado tamanho a cada ponto para que seja possível adicionar mais uma dimensão de análise ao gráfico.

Gráficos de barras

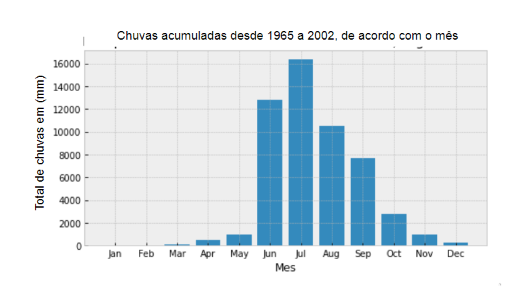
Uma maneira de comparar variáveis ​​categóricas é usando um gráfico de barras. Por exemplo, se você deseja visualizar quais são os meses com menor ou maior volume de chuva, pode acumular as medições de acordo com o mês em que ocorreram. Em primeiro lugar, é calculada a precipitação acumulada nos diferentes meses:

| precipitações\_acumuladas = df\_chuvas.sum() print(precipitações\_acumuladas) |
| --- |



Os meses estão localizados no eixo horizontal e as chuvas acumuladas no eixo vertical.

| fig, ax = plt.subplots(figsize=(8,4)) precipitações\_acumuladas = df\_chuvas.sum() ax.bar(df\_chuvas.columns, precipitações\_acumuladas) ax.set\_title('Precipitações acumuladas desde 1965 a 2002, segundo o mês') ax.set\_ylabel('Precipitação total (mm.)') ax.set\_xlabel('Mês') |
| --- |



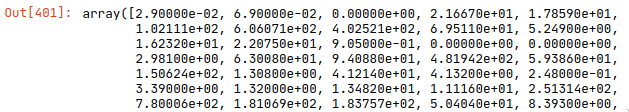
Histograma

Um histograma é uma representação gráfica de uma variável em forma de barras, onde a área de cada barra é proporcional à frequência dos valores representados. Dessa forma, permitem obter uma visão geral da distribuição do conjunto de dados.

Para preparar o histograma, primeiro o conjunto de dados deve ser categorizado em diferentes *intervalos de valores* que não se sobrepõem, são chamados de intervalos de classe. Posteriormente, a frequência absoluta de ocorrência de cada um dos intervalos é calculada a partir das observações. No caso de chuvas, o mínimo observado durante todo o período de registro é de 0 mm e o máximo é de 780 mm. Portanto, devemos construir intervalos de classe que vão de 0 a 780. Se usarmos 10 intervalos de classe, então cada um terá uma largura de (780 - 0) / 10 = 78 unidades.

O método para construir o histograma é ax.hist (), que usa uma matriz unidimensional (um conjunto de observações) como parâmetro. No entanto, nosso quadro de dados é uma estrutura bidimensional (por mês e por ano). Uma maneira conveniente de transformar todo o conteúdo do quadro de dados em uma única matriz é usar o seguinte método

| df\_chuvas.values.flatten() |
| --- |



Para corroborar as dimensões da matriz você pode usar o método .shape():

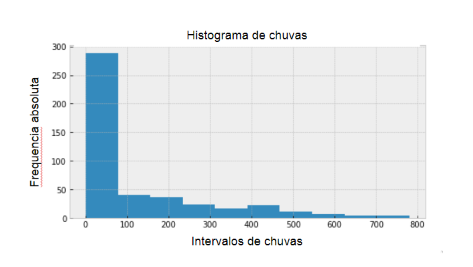
| df\_chuvas.values.flatten().shape() |
| --- |



Isto é, ao aplicar o método flatten() sobre os valores do data frame obtém-se uma matriz "achatado", com 456 elementos.

O gráfico:

| fig, ax = plt.subplots(figsize=(8, 4)) ax.hist(df\_chuvas.values.flatten(), bins=10) ax.set\_title('Histograma de precipitações') ax.set\_xlabel('Intervalos de precipitações (mm.)') ax.set\_ylabel('Frequência absoluta') |
| --- |



O número de intervalos de classe é especificado como o parâmetro *bins* na chamada do método ax.hist(). A forma do histograma muda ligeiramente dependendo de quantos intervalos de aula são considerados. Como pode ser visto na figura, o intervalo [0, 78) é o que ocorre com mais frequência, ou seja, a maioria das precipitações têm valores nessa faixa.

**Observação**: embora no histograma anterior representemos a *frequência absoluta* de cada um dos intervalos, também é válido representar a *frequência relativa* ou a *percentagem em relação ao total*.