Guia de acesso rápido – Python para Data Science – Matplotlib

adminvooo

Matplotlib é uma biblioteca Python de plotagem gráfica 2D. Matplotlib permite produzir gráficos de qualidade em diversos formatos e em ambientes interativos em várias plataformas.

Essa guia traz exemplos e os principais comandos utilizados no Matplotlib, de modo que serve de referência rápida para plotar gráficos e visualizar dados.

Anatomia e fluxo geral para plotar gráficos

Fluxo

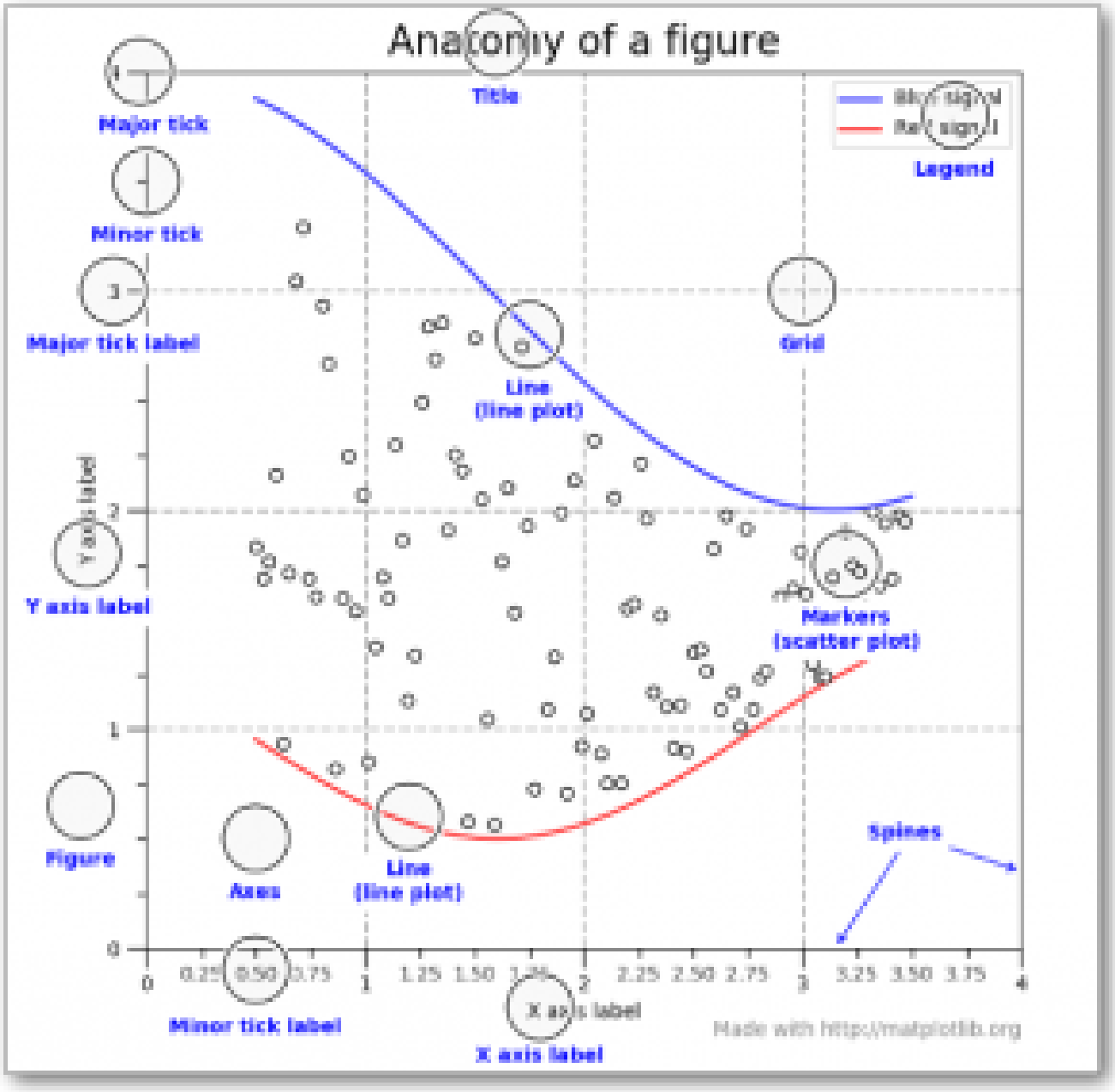
O procedimento básico para plotar gráficos com Matplotlib segue os seguintes passos:

1. Preparando os dados
2. Criando os gráficos
3. Rotinas de plotagem
4. Customizando
5. Salvando
6. Exibindo

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] #Passo 1
>>> y = [10,20,25,30]
>>> fig = plt.figure() #Passo 2
>>> ax = fig.addsubplot(111) #Passo 3
>>> ax.plot(x, y, color='lightblue', linewidth=3) #Passo 4
>>> ax.scatter([2,4,6], [5,15,25], color='darkgreen', marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png') #Passo 5
>>> plt.show() #Passo 6
```

Anatomia de uma figura

A imagem abaixo mostra os componentes de uma figura utilizando-se Matplotlib.



---

1. Preparando os dados

Dados 1D

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

Imagens ou dados 2D

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
```

```
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('aexs_grid/bivarate_normal.py'))
```

---

2. Criando o gráfico

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

Figura

```
>>> fig = plt.figure()
>>> fig2 =plt.figure(figsize=plt.figaspect(2.0))
```

Eixos

Todo gráfico é montado em relação a um Eixo. Em geral, um subgráfico é suficiente. Um subgráfico é um eixo num sistema de grades.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # linha-coluna-número
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows = 2, ncols = 2)
>>> fig4, axes2 = plt.subplots(ncols = 3)
```

---

3. Rotinas de plotagem

Dados 1D

```
>>> fig, ax = plt.subplots()
# Plota pontos conectados por linhas e marcadores
>>> lines = ax.plot(x, y)
# Plota pontos desconectados, em escala ou coloridos
>>> ax.scatter(x, y)
# Plota retângulos verticais (comprimento constante)
>>> axes[0,0].bar([1,2,3], [3,4,5])
# Plota retângulos horizontais (altura constante)
>>> axes[1,0].barh([0.5,1,2.5], [0,1,2])
# Plota uma linha horizontal através dos eixos
>>> axes[1,1].axhline(0.45)
# Plota uma linha vertical através dos eixos
>>> axes[0,1].axvline(0.65)
# Plota polígonos preenchidos
>>> ax.fill(x, y, color='blue')
# Preenche entre valores-y e 0
>>> ax.fill_between(x, y, color='yellow')
```

Dados ou imagens 2D

```
>>> fig, ax = plt.subplots()
# Matrizes RGB mapa-coloriradas
>>> im = ax.imshow(img, cmp='gist earth', interpolation='nearest', vmin=-2,
vmax=2)
# Plotagem pseudo-colorida de matriz 2D
>>> axes2[0].pcolor(data2)
>>>axes2[0].pcolormesh(data)
# Contornos da plotgem
>>> CS - plt.contour(Y, X, U)
```

```
# Contornos da plotgem preenchida
>>> axes2[2].countourf(data1)
# Rotula um contorno de plotagem
>>>axes2[2] = ax.clabel(CS)
```

Campos vetoriais

```
# Adiciona uma flecha ao eixo
>>> axes[0,1].arrow(0,0,0.5,0.5
# Plota um campo 2D de flechas
>>> axes[0,1].quiver(y,z)
>>> axes[0,1].streamplot(X, Y, U, V)
```

Distribuições

```
# Plota um histograma
>>> ax1.hist(y)
# Cria uma caixa e uma plotagem whisker
>>> ax3.boxplot(y)
# Cria uma plotagem violino
>>> ax3.violinplot(z)
```

---

4. Customizando

Cores, barras coloridas e mapas coloridos

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha=0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img, cmap='seismic')
```

Marcadores

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x, y, marker=".")
>>> ax.plotx, y, marker="o")
```

Estilos de linhas

```
>>> plt.plot(x, y, linewidht=4.0)
>>> plt.plot(x, y, ls='solid')
>>> plt.plot(x, y, ls='--')
>>> plt.plot(x, y, '--', x**2, y**2, '-.')
>>> plt.setp(lines, color='r', linewidht=4.0)
```

Texto e anotações

```
>>> ax.text(1, -2.1, 'Gráfico de exemplo', style='italic')
>>> ax.annotate("Sine", xy=(8, 0), xycoords='data', xytext=(10.5, 0),
textcoords='data', arrowprops = dict(arrowstyle="-.", connectionstyle="arc3"),)
```

Texto matemático

```
 >>> plt.title(r'$sigma_i = 15$', fontsize=20)
```

Limites, legendas e leioutes

Limites e auto-escala

```
# Adiciona preenchimento a uma plotagem
>>> ax.margins(x=0.0, y=0.1)
# Define o aspecto da razão da plotagem como 1
>>> ax.axis('equal')
# Define os limites dos eixos x e y
>>> ax.(xlim=[0,10.5], ylim=[-1.5,1.5])
# Define os limites do eixo x
>>> ax.set_xlim(0, 10.5)
```

Legendas

```
# Define o título e os rótulos dos eixos x e y
>>> ax.set(title='Eixo de exemplo', ylabel='Eixo-Y', xlabel='Eixo-X')
# Não permite sobreposição entre os elementos da plotagem
>>> ax.legend(loc='best')
```

Tiques

```
# Define manualmente os tiques de x
>>> ax.xaxies.set(ticks=range(1,5), ticklabels=[3,100,-12,'foo'])
# Faz os tiques de y mais longos e saindo e entrando
>>> ax.tick_params(axis='y', direction='inout', length=10)
```

Espaçamento do sub-gráfico

```
# Ajusta o espaço entre subgráficos
>>> fig3.subplots_adjust(wspace=0.5, hspace=0.3, left=0.125, right=0.9, top=0.9,
bottom=0.1)
# Encaixa o sugráfico na área da figura
>>> fig.tight_layout()
```

Coluna dos eixos (spines)

```
# Torna invisível a linha superior do eixo
>>> ax1.spines['top'].set_visible(False)
# Move para fora a linha inferior do eixo
>>> ax1.spines['bottom'].set_position(('outward',10))
```

---

5. Salvando

Salvando figuras

```
>>> plt.savefig('foo.png')
```

Salvando figuras transparentes

```
>>> plt.savefig('foo.png', transparent=True)
```

---

6. Exibindo

```
>>> plt.show()
```

Fechando e limpando

```
#Limpa um eixo
>>> plt.cla()
#Limpa a figura inteira
>>> plt.clf()
#Fecha uma janela
>>> plt.close()
```