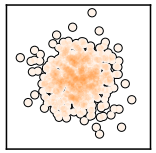
Matplotlib dicas & truques

Transparência

Gráficos de dispersão podem ser melhorados ao usar a transparência (alpha) para mostrar uma área com maior densidade. Múltiplos gráficos de dispersão podem ser usados para delinear uma fronteira.

```
X = np.random.normal(-1, 1, 500)
Y = np.random.normal(-1, 1, 500)
ax.scatter(X, Y, 50, "0.0", lw=2) # optional
ax.scatter(X, Y, 50, "1.0", lw=0) # optional
ax.scatter(X, Y, 40, "C1",  lw=0, alpha=0.1)
```



Rasterização

Se a sua figura tem muitos elementos gráficos, como um gráfico de dispersão gigante, você pode rasterizá-los para salvar memória e manter outros elementos em formato vetorial.

```
X = np.random.normal(-1, 1, 10_000)
Y = np.random.normal(-1, 1, 10_000)
ax.scatter(X, Y, rasterized=True)
fig.savefig("rasterized-figure.pdf", dpi=600)
```

Renderização offline

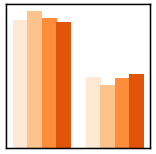
Use o backend Agg para renderizar a figura diretamente em um array.

```
from matplotlib.backends.backend_agg import FigureCanvas
canvas = FigureCanvas(Figure())
... # draw some stuff
canvas.draw()
Z = np.array(canvas.renderer.buffer_rgba())
```

Faixa contínua de cores

Você pode usar um mapa de cor (colormap) para escolher de uma faixa de cores contínuas.

```
X = np.random.randn(1000, 4)
cmap = plt.get_cmap("Oranges")
colors = cmap([0.2, 0.4, 0.6, 0.8])

ax.hist(X, 2, histtype='bar', color=colors)
```



Contorno de texto

Use o contorno de texto para deixar o texto mais visível.

```
import matplotlib.patheffects as fx
text = ax.text(0.5, 0.1, "Label")
text.set_path_effects([
  fx.Stroke(linewidth=3, foreground='1.0'),
  fx.Normal()])
```

Plot multilinhas

Você pode plotar várias linhas ao mesmo tempo usando None como separador.

```
X,Y = [], []
for x in np.linspace(0, 10*np.pi, 100):
  X.extend([x, x, None]), Y.extend([0, sin(x), None])
ax.plot(X, Y, "black")
```

Linhas pontilhadas

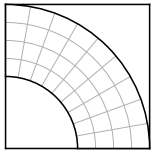
Para obter linhas pontilhadas arredondadas, use o `linestyle` customizado e modifique o `dash_capstyle`.

```
ax.plot([0,1], [0,0], "C1",
        linestyle = (0, (0.01, 1)), dash_capstyle="round")
ax.plot([0,1], [1,1], "C1",
        linestyle = (0, (0.01, 2)), dash_capstyle="round")
```

Eixos combinados

Você pode usar eixos sobrepostos com diferentes projeções.

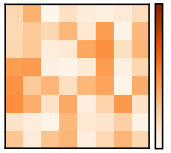
```
ax1 = fig.add_axes([0,0,1,1],
                   label="cartesian")
ax2 = fig.add_axes([0,0,1,1],
                   label="polar",
                   projection="polar")
```



Ajuste da barra de cores

Você consegue ajustar o tamanho de uma barra de cores (colorbar) quando a adiciona.

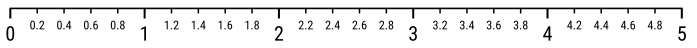
```
im = ax.imshow(Z)

cb = plt.colorbar(im,
        fraction=0.046, pad=0.04)
cb.set_ticks([])
```



Tirando vantagem da tipografia

Você pode usar uma fonte condensada como a Roboto Condensed para salvar espaço nos rótulos dos ticks.

```
for tick in ax.get_xticklabels(which='both'):
    tick.set_fontname("Roboto Condensed")
```



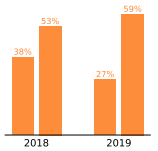
Se livrando das margens

Uma vez que sua figura estiver pronta, você pode chamar `tight_layout()` para remover as margens brancas. Se houver alguma margem adicional, você pode usar a utilidade do `pdfcrop` (vem com o TeX live).

Hachuras

Você pode alcançar um efeito visual legal com padrões de hachuras.

```
cmap = plt.get_cmap("Oranges")
plt.rcParams['hatch.color'] = cmap(0.2)
plt.rcParams['hatch.linewidth'] = 8
ax.bar(X, Y, color=cmap(0.6), hatch="/" )
```



Leia a documentação

Matplotlib tem uma documentação extensiva explicando os detalhes de cada comando e é geralmente acompanhado por exemplos. Junto com uma galeria online gigante, essa documentação é uma mina de ouro.