Matplotlib tips & tricks

Transparency

Scatter plots can be enhanced by using transparency (alpha) in order to show area with higher density. Multiple scatter plots can be used to delineate a frontier.

```
500)
X = np.random.normal(-1, 1,
Y = np.random.normal(-1, 1,
```



Multiline plot

Rasterization

If your figure has many graphical elements, such as a huge scatter, you can rasterize them to save memory and keep other elements in vector format.

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

Offline rendering

Use the Agg backend to render a figure directly in an array.

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

Range of continuous colors

You can use colormap to pick from a range of continuous

```
ax.hist(X, 2, histtype='bar', color=colors)
                                                                        colors = cmap([0.2, 0.4, 0.6, 0.8])
                                     cmap = plt.get_cmap("Oranges")
X = np.random.randn(1000, 4)
```

Text outline

Use text outline to make text more visible.

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



pad=0.04)

fraction=0.046,

You can adjust a colorbar's size when adding it.

Colorbar adjustment

Taking advantage of typography

You can use a condensed font such as Roboto Condensed to save space on tick labels.

You can plot several lines at once using None as separator.

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")

X, Y = [], []

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

0 02 0.4 0.6 0.8 7 1.2 1.4 1.6 1.8 2 2.2 2.4 2.6 2.8 3 3.2 3.4 3.6 3.8 4 4.2 4.4 4.6 4.8

to remove white margins. If there are remaining margins, Once your figure is finished, you can call tight_layout() you can use the pdfcrop utility (comes with TeX live)

Hatching

To have rounded dotted lines, use a custom linestyle and

modify dash_capstyle.

Dotted lines

You can achieve a nice visual effect with thick hatch pat-

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

linestyle = (0, (0.01, 2)), dash_capstyle="round")

ax.plot([0,1], [1,1], "C1", ax.plot([0,1], [0,0], "C1"

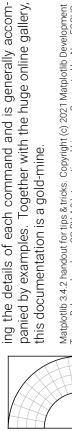
linestyle = (0, (0.01, 1)), dash_capstyle="round")



Combining axes

You can use overlaid axes with different projections.

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



panied by examples. Together with the huge online gallery, this documentation is a gold-mine.

Matplotlib comes with an extensive documentation explain-

Read the documentation

Matplotlib 3.4.2 handout for tips & tricks. Copyright (c) 2021 Matplotlib Development Team. Released under a CC-BY 4.0 International License. Supported by NumFOCUS.