

Ouick start

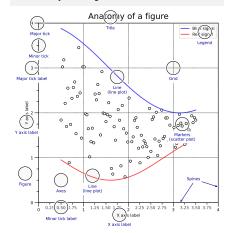
import numpy as np import matplotlib as mpl import matplotlib.pyplot as plt

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

subplot[s](rows,cols,...) fig, axs = plt.subplots(3, 3)G = gridspec(rows,cols,...) API ax = G[0,:]ax.inset_axes(extent) d=make axes locatable(ax) API ax = d.new_horizontal('10%')

Getting help

matplotlib.org

github.com/matplotlib/matplotlib/issues

discourse.matplotlib.org

stackoverflow.com/questions/tagged/matplotlib | gitter.im/matplotlib

¥ twitter.com/matplotlib

✓ Matplotlib users mailing list



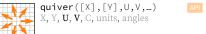
scatter(X,Y,...) X, Y, [s]izes, [c]olors, marker, cmap











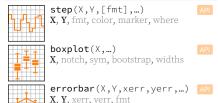






Advanced plots

API



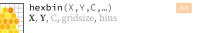






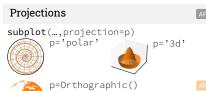






ax.set_[xy]scale(scale,...) MMMMMM linear log any values values > 0 symlog logit any values 0 < values < 1

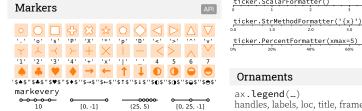
Scales





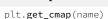






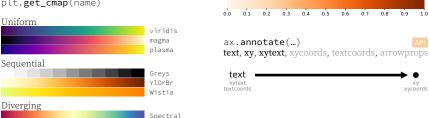






Colormaps

Cyclic



coolwarm

Event handling

Tick locators

ticker.NullLocator()

ticker.AutoLocator()

ticker.MaxNLocator(n=4)

Tick formatters

ticker.NullFormatter()

ticker.ScalarFormatter()

Ornaments

ax.legend(...)

Legend ←

ax.colorbar(...)

from matplotlib import ticker

ticker.FormatStrFormatter('>%d<')

ticker.PercentFormatter(xmax=5)

handles, labels, loc, title, frameon

Label 1

Label 2

mappable, ax, cax, orientation

Label 3

Label 4

from matplotlib import ticker

ticker.MultipleLocator(0.5)

ticker.FixedLocator([0, 1, 5])

ticker.LinearLocator(numticks=3)

ax.[xy]axis.set [minor|major] locator(locator)

0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

ticker.IndexLocator(base=0.5, offset=0.25)

ticker.LogLocator(base=10, numticks=15)

ax.[xy]axis.set_[minor|major]_formatter(formatter)

ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)

ticker.FixedFormatter(['', '0', '1', ...])

0.25 0.50 | 0.75 | 0.25 | 0.50 0.75 | 3 0.25 0.50 0.75 | 4

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

Animation

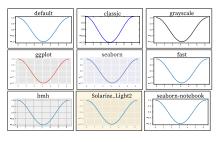
import matplotlib.animation as mpla

```
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 = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Styles

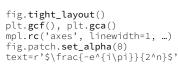
API

plt.style.use(style)



Quick reminder

```
ax.grid()
ax.patch.set_alpha(0)
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(list)
ax.set_[xy]ticklabels(list)
ax.set_[sup]title(title)
ax.tick_params(width=10, ...)
ax.set_axis_[on|off]()
```



Keyboard shortcuts



b View back

O Zoom to rect

y Y pan/zoom

- f View forward
- p Pan view
- x X pan/zoom g Minor grid 0/1
 - G Major grid 0/1
- X axis log/linear L Y axis log/linear

Ten simple rules

1. Know Your Audience

2. Identify Your Message

3. Adapt the Figure

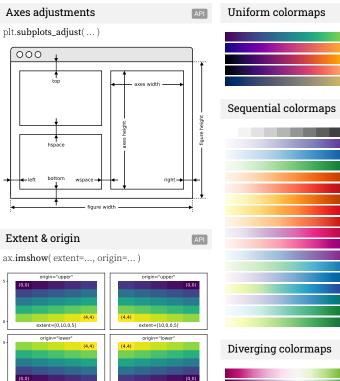
4. Captions Are Not Optional

5. Do Not Trust the Defaults 6. Use Color Effectively

7. Do Not Mislead the Reader

8. Avoid "Chartiunk"

9. Message Trumps Beauty 10. Get the Right Tool



extent=[0.10.0.5] extent=[10.0.0.5] Text alignments API ax.text(..., ha=... , va=..., ...)

1	1	(1,1)
\mathbb{I}	thlat	top
TIVIC	ιτρησι	baseline
(0,0) left	center	right

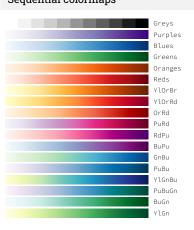
Text parameters
ax.text(, family=, size=, weight=)

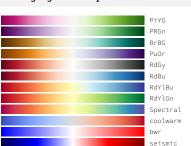
xx-large	(1.73)
x-large	(1.44)
large	(1.20)
medium	(1.00)
small	(0.83)
x-small	(0.69)
xx-small	(0.58)
	x-large large medium small x-small

)
)
)
)
)

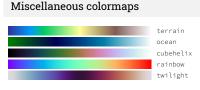
The quick brown fox jumps over the lazy dog	monospace
The quick brown fox jumps over the lazy dog	serif
The quick brown fox jumps over the lazy dog	sans
The quick brown fox jumps over the lazy dog	cursive
The quick brown fox jumps over the lazy dog	italic
The quick brown fox jumps over the lazy dog	normal
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG	small-caps
The quick brown fox jumps over the lazy dog	normal

viridis plasma inferno magma cividis Greys Purples

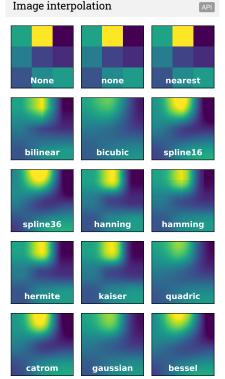








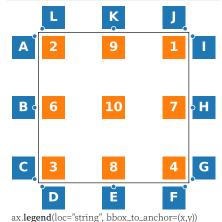




sinc

lanczos

mitchell

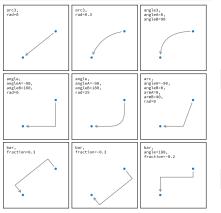


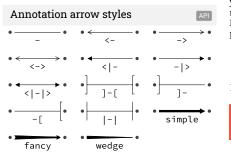
Legend placement

2: upper left 9: upper center 1: upper right 10: center 6: center left 7: center right 3: lower left 8: lower center 4: lower right

A: upper right / (-0.1,0.9) B: center right / (-0.1,0.5) C: lower right / (-0.1,0.1) D: upper left / (0.1,-0.1) E: upper center / (0.5,-0.1) F: upper right / (0.9, -0.1) G: lower left / (1.1,0.1) H: center left / (1.1.0.5) I: upper left / (1.1,0.9) J: lower right / (0.9,1.1) K: lower center / (0.5,1.1) L: lower left / (0.1,1.1)

Annotation connection styles





How do I resize a figure? \rightarrow fig.set_size_inches(w, h) ... save a figure? → fig.savefig("figure.pdf") ... save a transparent figure? → fig.savefig("figure.pdf", transparent=True) ... clear a figure/an axes? \rightarrow fig.clear() \rightarrow ax.clear() ... close all figures? → plt.close("all") ... remove ticks? \rightarrow ax.set_[xy]ticks([]) ... remove tick labels? → ax.set_[xv]ticklabels([])

→ ax.spines['top'].set_visible(False) ... hide legend border? → ax.legend(frameon=False) ... show error as shaded region?

 \rightarrow ax.set_[xv]ticks(rotation=90)

→ ax.fill_between(X, Y+error, Y-error) ... draw a rectangle?

 \rightarrow ax.add_patch(plt.Rectangle((0, 0), 1, 1)

... draw a vertical line? \rightarrow ax.axvline(x=0.5) ... draw outside frame?

... rotate tick labels?

... hide top spine?

 \rightarrow ax.plot(..., clip_on=False)

... use transparency? \rightarrow ax.plot(..., alpha=0.25)

... convert an RGB image into a gray image?

 \rightarrow grav = 0.2989*R + 0.5870*G + 0.1140*B ... set figure background color?

→ fig.patch.set_facecolor("grey") ... get a reversed colormap?

→ plt.get_cmap("viridis_r") ... get a discrete colormap?

 \rightarrow plt.get_cmap("viridis", 10) ... show a figure for one second?

 \rightarrow fig.show(block=False), time.sleep(1)

Performance tips



Beyond Matplotlib

Seaborn: Statistical Data Visualization Cartopy: Geospatial Data Processing yt: Volumetric data Visualization mpld3: Bringing Matplotlib to the browser Datashader: Large data processing pipeline plotnine: A Grammar of Graphics for Python

Matplotlib Cheatsheets Copyright (c) 2021 Matplotlib Development Team Released under a CC-BY 4.0 International License

