

Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

1 Initialize

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

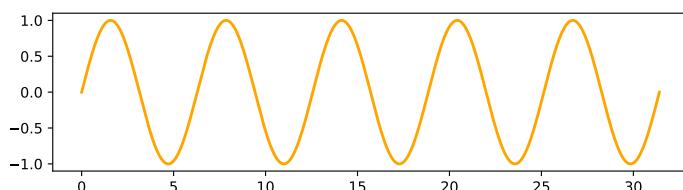
2 Prepare

```
X = np.linspace(0, 4*np.pi, 1000)  
Y = np.sin(X)
```

3 Render

```
fig, ax = plt.subplots()  
ax.plot(X, Y)  
fig.show()
```

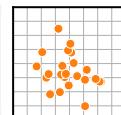
4 Observe



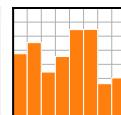
Choose

Matplotlib offers several kind of plots (see Gallery):

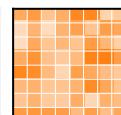
```
X = np.random.uniform(0, 1, 100)  
Y = np.random.uniform(0, 1, 100)  
ax.scatter(X, Y)
```



```
X = np.arange(10)  
Y = np.random.uniform(1, 10, 10)  
ax.bar(X, Y)
```



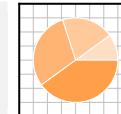
```
Z = np.random.uniform(0, 1, (8,8))  
ax.imshow(Z)
```



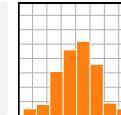
```
Z = np.random.uniform(0, 1, (8,8))  
ax.contourf(Z)
```



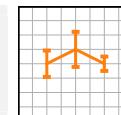
```
Z = np.random.uniform(0, 1, 4)  
ax.pie(Z)
```



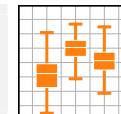
```
Z = np.random.normal(0, 1, 100)  
ax.hist(Z)
```



```
X = np.arange(5)  
Y = np.random.uniform(0,1,5)  
ax.errorbar(X, Y, Y/4)
```



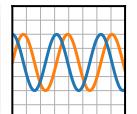
```
Z = np.random.normal(0,1,(100,3))  
ax.boxplot(Z)
```



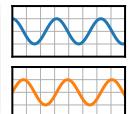
Organize

You can plot several data on the same figure but you can also split a figure in several subplots (named Axes):

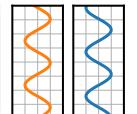
```
X = np.linspace(0,10,100)  
Y1, Y2 = np.sin(X), np.cos(X)  
ax.plot(X, Y1, color="C1")  
ax.plot(X, Y2, color="C0")
```



```
fig, (ax1, ax2) = plt.subplots((2,1))  
ax1.plot(X, Y1, color="C1")  
ax2.plot(X, Y2, color="C0")
```

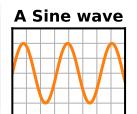


```
fig, (ax1, ax2) = plt.subplots((1,2))  
ax1.plot(Y1, X, color="C1")  
ax2.plot(Y2, X, color="C0")
```



Label (everything)

```
ax.plot(X, Y)  
fig.suptitle(None)  
ax.set_title("A Sine wave")
```



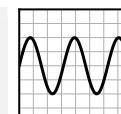
```
ax.plot(X, Y)  
ax.set_ylabel(None)  
ax.set_xlabel("Time")
```



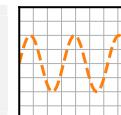
Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

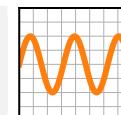
```
X = np.linspace(0,10,100)  
Y = np.sin(X)  
ax.plot(X, Y, color="black")
```



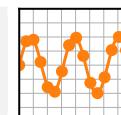
```
X = np.linspace(0,10,100)  
Y = np.sin(X)  
ax.plot(X, Y, linestyle="--")
```



```
X = np.linspace(0,10,100)  
Y = np.sin(X)  
ax.plot(X, Y, linewidth=5)
```



```
X = np.linspace(0,10,100)  
Y = np.sin(X)  
ax.plot(X, Y, marker="o")
```



Explore

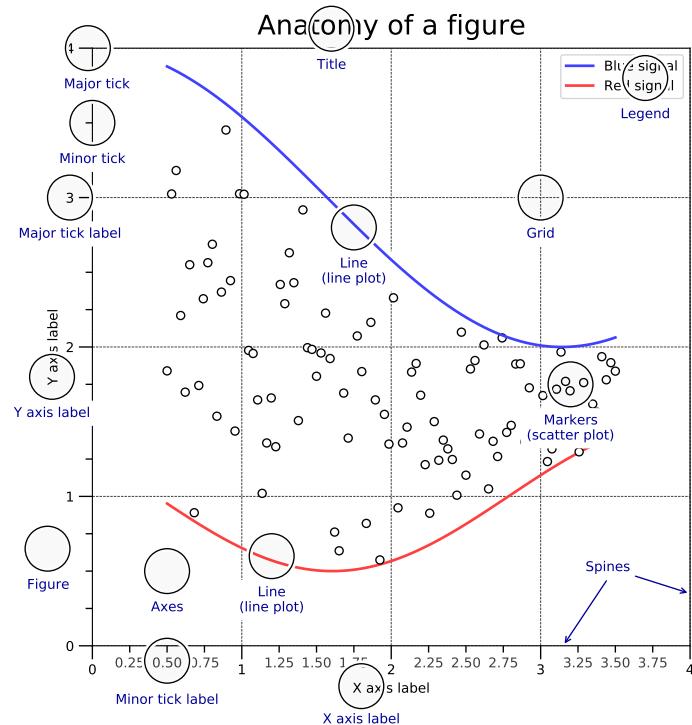
Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)  
fig.savefig("my-first-figure.pdf")
```

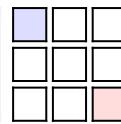
Matplotlib for intermediate users

A matplotlib figure is composed of a hierarchy of elements that forms the actual figure. Each element can be modified.

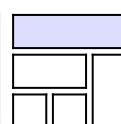


Figure, axes & spines

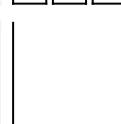
```
fig, axs = plt.subplots((3,3))
axs[0,0].set_facecolor("#dddddff")
axs[2,2].set_facecolor("#fffffd")
```



```
gs = fig.add_gridspec(3, 3)
ax = fig.add_subplot(gs[0, :])
ax.set_facecolor("#dddddff")
```

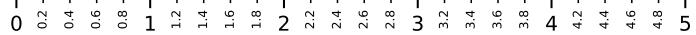


```
fig, ax = plt.subplots()
ax.spines["top"].set_color("None")
ax.spines["right"].set_color("None")
```



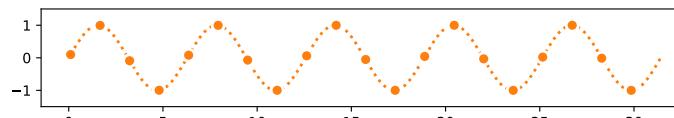
Ticks & labels

```
from mpl.ticker import MultipleLocator as ML
from mpl.ticker import ScalarFormatter as SF
ax.xaxis.set_minor_locator(ML(0.2))
ax.xaxis.set_minor_formatter(SF())
ax.tick_params(axis='x', which='minor', rotation=90)
```



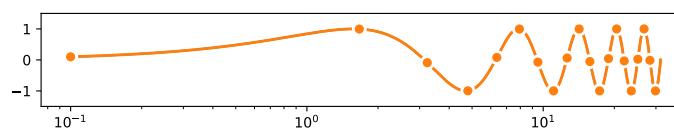
Lines & markers

```
X = np.linspace(0.1, 10*np.pi, 1000)
Y = np.sin(X)
ax.plot(X, Y, "C1o:", markevery=25, mec="1.0")
```



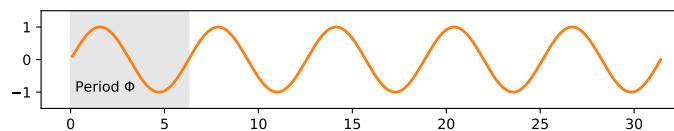
Scales & Projections

```
fig, ax = plt.subplots()
ax.set_xscale("log")
ax.plot(X, Y, "C1o-", markevery=25, mec="1.0")
```



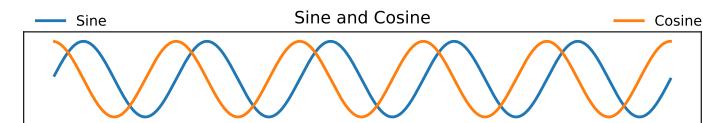
Text & Ornaments

```
ax.fill_betweenx([-1,1],[0],[2*np.pi])
ax.text(0, -1, r"Period $\Phi$")
```



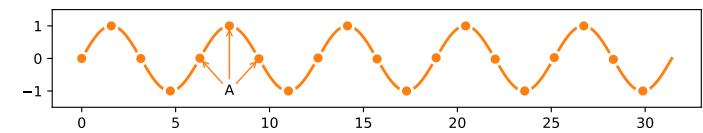
Legend

```
ax.plot(X, np.sin(X), "C0", label="Sine")
ax.plot(X, np.cos(X), "C1", label="Cosine")
ax.legend(bbox_to_anchor=(0,1,1,.1), ncol=2, mode="expand", loc="lower left")
```



Annotation

```
ax.annotate("A", (X[250],Y[250]), (X[250],-1),
ha="center", va="center", arrowprops =
{"arrowstyle": "->", "color": "C1"})
```



Colors

Any color can be used but Matplotlib offers sets of colors:

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

Size & DPI

Consider a square figure to be included in a two-columns A4 paper with 2cm margins on each side and a column separation of 1cm. The width of a figure is $(21 - 2*2 - 1)/2 = 8\text{cm}$. One inch being 2.54cm, figure size should be 3.15×3.15 in.

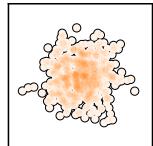
```
fig = plt.figure(figsize=(3.15,3.15), dpi=50)
plt.savefig("figure.pdf", dpi=600)
```

Matplotlib tips & tricks

Transparency

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

```
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)
```



Rasterization

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

```
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)
```

Offline rendering

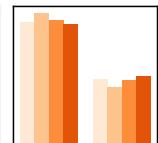
Use the Agg backend to render a figure directly in an array.

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

Range of continuous colors

You can use colormap to pick a range of continuous colors.

```
X = np.random.randn(1000, 4)
cmap = plt.get_cmap("Blues")
colors = [cmap(i) for i in [.2, .4, .6, .8]]
ax.hist(X, 2, histtype='bar', color=colors)
```



Text outline

Use text outline to make text more visible.

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



Multiline plot

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

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



Dotted lines

To have rounded dotted lines, use a custom linestyle and modify dash_capstyle.

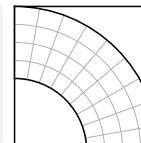
```
ax.plot([0, 1], [0, 0], "C1",
        linestyle=(0, (0.01, 1)), dash_capstyle="round")
ax.plot([0, 1], [1, 1], "C1",
        linestyle=(0, (0.01, 2)), dash_capstyle="round")
```



Combining axes

You can use overlaid axes with different projections.

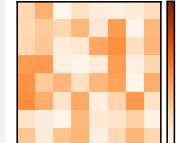
```
ax1 = fig.add_axes([0, 0, 1, 1],
                   label="cartesian")
ax2 = fig.add_axes([0, 0, 1, 1],
                   label="polar",
                   projection="polar")
```



Colorbar adjustment

You can adjust colorbar aspect when adding it.

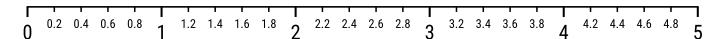
```
im = ax.imshow(Z)
cb = plt.colorbar(im,
                  fraction=0.046, pad=0.04)
cb.set_ticks([])
```



Taking advantage of typography

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

```
for tick in ax.get_xmajorticks():
    tick.set_fontname("Roboto Condensed")
```



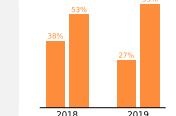
Getting rid of margins

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

Hatching

You can achieve nice visual effect with thick hatch patterns.

```
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="/")
```



Read the documentation

Matplotlib comes with an extensive documentation explaining every details of each command and is generally accompanied by examples with. Together with the huge online gallery, this documentation is a gold-mine.

matplotlib

Cheat sheet Version 3.2 API

Quick 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='C1')

fig.savefig("figure.pdf")
fig.show()
```

Anatomy of a figure

Basic plots

<code>plot([X], Y, [fmt], ...)</code>	<code>API</code>
X, Y, fmt, color, marker, linestyle	

<code>scatter(X, Y, ...)</code>	<code>API</code>
X, Y, [s]izes, [c]olors, marker, cmap	

<code>bar[h](x, height, ...)</code>	<code>API</code>
x, height, width, bottom, align, color	

<code>imshow(Z, [cmap], ...)</code>	<code>API</code>
Z, cmap, interpolation, extent, origin	

<code>contour(f)([X], [Y], Z, ...)</code>	<code>API</code>
X, Y, Z, levels, colors, extent, origin	

<code>quiver([X], [Y], U, V, ...)</code>	<code>API</code>
X, Y, U, V, C, units, angles	

<code>pie(X, [explode], ...)</code>	<code>API</code>
Z, explode, labels, colors, radius	

<code>text(x, y, text, ...)</code>	<code>API</code>
x, y, text, va, ha, size, weight, transform	

<code>fill_between(x)(...)</code>	<code>API</code>
X, Y1, Y2, color, where	

Advanced plots

<code>step(X, Y, [fmt], ...)</code>	<code>API</code>
X, Y, fmt, color, marker, where	

<code>boxplot(X, ...)</code>	<code>API</code>
X, notch, sym, bootstrap, widths	

<code>errorbar(X, Y, xerr, yerr, ...)</code>	<code>API</code>
X, Y, xerr, yerr, fmt	

<code>hist(X, bins, ...)</code>	<code>API</code>
X, bins, range, density, weights	

<code>violinplot(D, ...)</code>	<code>API</code>
D, positions, widths, vert	

<code>barbs([X], [Y], U, V, ...)</code>	<code>API</code>
X, Y, U, V, C, length, pivot, sizes	

<code>eventplot(positions, ...)</code>	<code>API</code>
positions, orientation, lineoffsets	

<code>hexbin(X, Y, C, ...)</code>	<code>API</code>
X, Y, C, gridsize, bins	

<code>xcorr(X, Y, ...)</code>	<code>API</code>
X, Y, normed, detrend	

Basic plots

<code>plot([X], Y, [fmt], ...)</code>	<code>API</code>
X, Y, fmt, color, marker, linestyle	

<code>scatter(X, Y, ...)</code>	<code>API</code>
X, Y, [s]izes, [c]olors, marker, cmap	

<code>bar[h](x, height, ...)</code>	<code>API</code>
x, height, width, bottom, align, color	

<code>imshow(Z, [cmap], ...)</code>	<code>API</code>
Z, cmap, interpolation, extent, origin	

<code>contour(f)([X], [Y], Z, ...)</code>	<code>API</code>
X, Y, Z, levels, colors, extent, origin	

<code>quiver([X], [Y], U, V, ...)</code>	<code>API</code>
X, Y, U, V, C, units, angles	

<code>pie(X, [explode], ...)</code>	<code>API</code>
Z, explode, labels, colors, radius	

<code>text(x, y, text, ...)</code>	<code>API</code>
x, y, text, va, ha, size, weight, transform	

<code>fill_between(x)(...)</code>	<code>API</code>
X, Y1, Y2, color, where	

Scales

<code>ax.set_xy scale(scale, ...)</code>	<code>API</code>
linear any values	

<code>log values > 0</code>	
any values	

<code>symlog 0 < values < 1</code>	
any values	

<code>logit 0 < values < 1</code>	
any values	

Projections

<code>subplot(..., projection=p)</code>	<code>API</code>
p='polar'	

<code>p='3d'</code>	
<code>from cartopy.crs import Cartographic</code>	

Lines

<code>linestyle or ls</code>	<code>API</code>
"--", ":", "-.", "-.", "(0, (0.01, 2))	

<code>capstyle or dash_capstyle</code>	<code>API</code>
"butt", "round", "projecting"	

Markers

<code>o, O, s, S, +, P, X, *, x, p, d, D, <, >, ^, v, 1, 2, 3, 4, 5, 6, 7, \$, %, \$S, %S, \$*\$, \$*\$, \$-\$, \$-\$, \$+, \$-, \$0, \$0S, \$0S, \$0\$</code>	<code>API</code>
markerkey	

10	(0, -1)	(25, 5)	(0, 25, -1)	
----	---------	---------	-------------	--

Colors

C0, C1, C2, C3, C4, C5, C6, C7, C8, C9	'Cn'
b, g, r, c, m, y, k, w	'x', 'name', '(R, G, B, [A])', '#RRGGBB[AA]', 'x.y'

DarkRed, Firebrick, Crimson, IndianRed, Salmon	
(1,0,0), (1,0,0,0.75), (1,0,0,0.5), (1,0,0,0.25)	
#FF0000, #FF0000BB, #FF000088, #FF000044	

Colormaps

<code>plt.get_cmap(name)</code>	<code>API</code>
---------------------------------	------------------

Uniform	viridis magma plasma
---------	----------------------------

Sequential	Greys YlOrBr Wistia
------------	---------------------------

Diverging	Spectral coolwarm RdGy
-----------	------------------------------

Qualitative	tab10 tab20
-------------	----------------

Cyclic	twilight
--------	----------

Tick locators

```
from matplotlib import ticker
ax.[x|y]axis.set_[minor|major]_locator(locator)
```

<code>ticker.NullLocator()</code>	<code>API</code>
-----------------------------------	------------------

<code>ticker.MultipleLocator(0.5)</code>	<code>API</code>
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0	

<code>ticker.FixedLocator([0, 1, 5])</code>	<code>API</code>
0.0 0.25 0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0 3.25 3.5 3.75 4.0 4.25 4.5 4.75	

<code>ticker.IndexLocator(base=0.5, offset=0.25)</code>	<code>API</code>
0.0 0.25 0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0 3.25 3.5 3.75 4.0 4.25 4.5 4.75	

<code>ticker.AutoLocator()</code>	<code>API</code>
0.0 0.25 0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0 3.25 3.5 3.75 4.0 4.25 4.5 4.75	

<code>ticker.MaxNLocator(n=4)</code>	<code>API</code>
0.0 0.25 0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0 3.25 3.5 3.75 4.0 4.25 4.5 4.75	

<code>ticker.LogLocator(base=10, numticks=15)</code>	<code>API</code>
10 ¹ 10 ² 10 ³ 10 ⁴ 10 ⁵ 10 ⁶ 10 ⁷ 10 ⁸ 10 ⁹	

Animation

```
import matplotlib.animation as mpl_a
```

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

Styles

<code>plt.style.use(style)</code>	<code>API</code>
-----------------------------------	------------------

<code>default</code>	<code>classic</code>	<code>grayscale</code>
----------------------	----------------------	------------------------

<code>ggplot</code>	<code>seaborn</code>	<code>fast</code>
---------------------	----------------------	-------------------

<code>bmh</code>	<code>Solarize_Light2</code>	<code>seaborn-notebook</code>
------------------	------------------------------	-------------------------------

Tick formatters

<code>from matplotlib import ticker</code>	
<code>ax.[x y]axis.set_[minor major]_formatter(formatter)</code>	

<code>ticker.NullFormatter()</code>	<code>API</code>
-------------------------------------	------------------

<code>ticker.FixedFormatter(['', '0', '1', '2', '3', '4', '5'])</code>	<code>API</code>
0.0 0.25 0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0 3.25 3.5 3.75 4.0 4.25 4.5 4.75	

<code>ticker.FuncFormatter(lambda x, pos: "%.%f" % x)</code>	<code>API</code>
[0.001] [1.000] [2.00] [3.00] [4.00] [5.00]	

<code>ticker.FormatStrFormatter('%.2d')</code>	<code>API</code>
>1 >2 >3 >4 >5	

<code>ticker.ScalarFormatter()</code>	<code>API</code>
0 1 2 3 4 5	

<code>ticker.StrMethodFormatter('{x}')</code>	<code>API</code>
0.0 1.0 2.0 3.0 4.0 5.0	

<code>ticker.PercentFormatter(xmax=5)</code>	<code>API</code>
0% 20% 40% 60% 80% 100%	

Ornaments

<code>ax.legend(...)</code>	<code>API</code>
handles, labels, loc, title, frameon	

<code>Legend</code>	<code>title</code>
handlelength	label
labelspacing	handle
borderpad	markerfacecolor (mfc)
borderspacing	markeredgecolor (mec)
borderaxespad	numpoints = scatterpoints

Quick reminder

<code>ax.grid()</code>	
------------------------	--

<code>ax.patch.set_alpha(0)</code>	
------------------------------------	--

<code>ax.set_xy lim(vmin, vmax)</code>	
--	--

<code>ax.set_xy label(label)</code>	
-------------------------------------	--

<code>ax.set_xy ticks(list)</code>	
------------------------------------	--

<code>ax.set_xy ticklabels(list)</code>	
---	--

<code>ax.set_sup title(title)</code>	
--------------------------------------	--

<code>ax.tick_params(width=10, ...)</code>	
--	--

<code>ax.set_axis_on off()</code>	
-----------------------------------	--

<code>ax.tight_layout()</code>	
--------------------------------	--

<code>plt.gcf(), plt.gca()</code>	
-----------------------------------	--

<code>mpl.rc('axes', linewidth=1, ...)</code>	
---	--

<code>fig.patch.set_alpha(0)</code>	
-------------------------------------	--

<code>text=r'\$\frac{-1}{\pi}\int_{-\pi}^{\pi} f(\phi) e^{i\phi} d\phi\$'</code>	
--	--

Keyboard shortcuts

<code>ctrl+s</code> Save	<code>ctrl+w</code> Close plot
--------------------------	--------------------------------

<code>r</code> Reset view	<code>f</code> Fullscreen 0/1
---------------------------	-------------------------------

<code>f</code> View forward	<code>b</code> View back
-----------------------------	--------------------------

<code>p</code> Pan view	<code>o</code> Zoom to rect
-------------------------	-----------------------------

<code>x</code> X pan/zoom	<code>y</code> Y pan/zoom
---------------------------	---------------------------

<code>g</code> Minor grid 0/1	<code>G</code> Major grid 0/1
-------------------------------	-------------------------------

<code>l</code> X axis log/linear	<code>L</code> 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 "Chartjunk"
9. Message Trumps Beauty
10. Get the Right Tool

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 (c) 2020 Nicolas P. Rougier

Released under a CC-BY 4.0 International License

