

Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.datacamp.com



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



1 Prepare The Data

Also see Lists & NumPy

1D Data

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

2D Data or Images

```
>>> 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('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

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

Figure

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

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

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

3 Plotting Routines

1D Data

<pre>>>> lines = ax.plot(x,y) >>> ax.scatter(x,y) >>> axes[0,0].bar([1,2,3],[3,4,5]) >>> axes[1,0].barh([0.5,1,2.5],[0,1,2]) >>> axes[1,1].axhline(0.45) >>> axes[0,1].axvline(0.65) >>> ax.fill(x,y,color='blue') >>> ax.fill_between(x,y,color='yellow')</pre>	<p>Draw points with lines or markers connecting them</p> <p>Draw unconnected points, scaled or colored</p> <p>Plot vertical rectangles (constant width)</p> <p>Plot horizontal rectangles (constant height)</p> <p>Draw a horizontal line across axes</p> <p>Draw a vertical line across axes</p> <p>Draw filled polygons</p> <p>Fill between y-values and 0</p>
--	--

2D Data or Images

<pre>>>> fig, ax = plt.subplots() >>> im = ax.imshow(img, cmap='gist_earth', interpolation='nearest', vmin=-2, vmax=2)</pre>	Colormapped or RGB arrays
--	---------------------------

Vector Fields

<pre>>>> axes[0,1].arrow(0,0,0.5,0.5) >>> axes[1,1].quiver(y,z) >>> axes[0,1].streamplot(X,Y,U,V)</pre>	Add an arrow to the axes
--	--------------------------

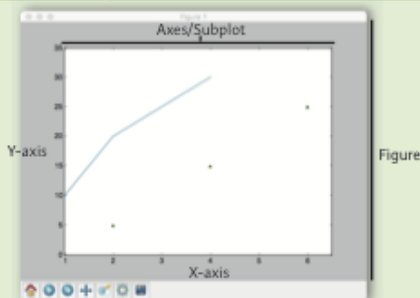
Data Distributions

<pre>>>> ax1.hist(y) >>> ax3.boxplot(y) >>> ax3.violinplot(z)</pre>	Plot a histogram
	Make a box and whisker plot
	Make a violin plot

<pre>>>> axes2[0].pcolor(data2) >>> axes2[0].pcolormesh(data) >>> CS = plt.contour(Y,X,U) >>> axes2[2].contourf(data1) >>> axes2[2] = ax.clabel(CS)</pre>	Pseudocolor plot of 2D array
	Pseudocolor plot of 2D array
	Plot contours
	Plot filled contours
	Label a contour plot

Plot Anatomy & Workflow

Plot Anatomy



Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

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

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> 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')
```

Markers

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

Linestyles

```
>>> plt.plot(x, y, linewidth=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', linewidth=4.0)
```

Text & Annotations

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

Mathtext

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

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0, y=0.1)
>>> ax.axis('equal')
>>> ax.set(xlim=[0,10.5], ylim=[-1.5,1.5])
>>> ax.set_xlim(0, 10.5)
```

Legends

```
>>> ax.set(title='An Example Axes',
          ylabel='Y-Axis',
          xlabel='X-Axis')
>>> ax.legend(loc='best')
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
               ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',
                  direction='inout',
                  length=10)
```

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,
                       hspace=0.3,
                       left=0.125,
                       right=0.9,
                       top=0.9,
                       bottom=0.1)
```

```
>>> fig.tight_layout()
```

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)
>>> ax1.spines['bottom'].set_position(('outward', 10))
```

Add padding to a plot

Set the aspect ratio of the plot to 1

Set limits for x-axis and y-axis

Set limits for x-axis

Set a title and x-axis and y-axis labels

No overlapping plot elements

Manually set x-ticks

Make y-ticks longer and go in and out

Adjust the spacing between subplots

Fit subplot(s) in to the figure area

Make the top axis line for a plot invisible

Move the bottom axis line outward

5 Save Plot

Save figures

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

Save transparent figures

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

6 Show Plot

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

Close & Clear

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```

Clear an axis

Clear the entire figure

Close a window

DataCamp

Learn Python for Data Science Interactively

