Python for Data Science
Matplotlib
Cheat Sheet

f616 *adapted from datacamp.com*

Introductory Note

This document is an adaption of the original datacamp.org cheat sheet.

- https://www.datacamp.com/resources/cheat-sheets/matplotlib-cheat-sheet-plotting-in-python
- https://github.com/f616/Python-Matplotlib-Cheat-Sheet

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

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))
```

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.

Axes

```
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 Save Plot

```
plt.savefig('foo.png')  #Save figures
plt.savefig('foo.png', transparent=True)  #Save transparent figures
```

4 Show Plot

```
plt.show()
```

5 Plotting Routines

1D Data

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

2D Data

```
fig, ax = plt.subplots()
im = ax.imshow(img, cmap='gist_earth', interpolation='nearest', vmin=-2, vmax=2)  #Colormapped or RGB arrays
axes2[0].pcolor(data2)  #Pseudocolor plot of 2D array
axes2[0].pcolormesh(data)  #Pseudocolor plot of 2D array
CS = plt.contour(Y,X,U)  #Plot contours
axes2[2].contourf(data1)  #Plot filled contours
axes2[2]= ax.clabel(CS)  #Label a contour plot
```

Vector Fields

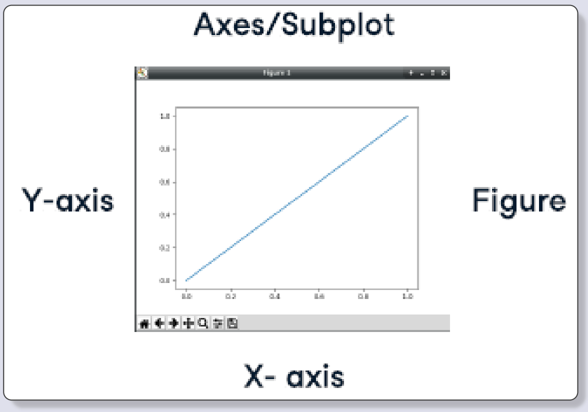
```
axes[0,1].arrow(0,0,0.5,0.5)  #Add an arrow to the axes
axes[1,1].quiver(y,z)  #Plot a 2D field of arrows
axes[0,1].streamplot(X,Y,U,V)  #Plot a 2D field of arrows
```

Data Distributions

```
ax1.hist(y)  #Plot a histogram
ax3.boxplot(y)  #Make a box and whisker plot
ax3.violinplot(z)  #Make a violin plot
```

6 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

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

7 Close and Clear

```python
plt.cla()  #Clear an axis
plt.clf()  #Clear the entire figure
plt.close()  #Close a window
```

8 Plotting Customize Plot

Colors, Color Bars & Color Maps

```python
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')
```

Makers

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

Linestyles

```python
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

```python
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

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

Limits & Autoscaling

```python
ax.margins(x=0.0,y=0.1)  #Add padding to a plot
ax.axis('equal')  #Set the aspect ratio of the plot to 1
ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])  #Set limits for x-and y-axis
ax.set_xlim(0,10.5)  #Set limits for x-axis
```

Legends

```python
ax.set(title='An Example Axes', ylabel='Y-Axis', xlabel='X-Axis')  #Set a title and x-and y-axis labels
ax.legend(loc='best')  #No overlapping plot elements
```

Ticks

```python
ax.xaxis.set(ticks=range(1,5), ticklabels=[3,100,-12,'foo'])  #Manually set x-ticks
ax.tick_params(axis='y', direction='inout', length=10)  #Make y-ticks longer and go in and out
```

Subplot Spacing

```python
fig3.subplots_adjust(wspace=0.5, hspace=0.3, left=0.125, right=0.9, top=0.9,
 bottom=0.1)  #Adjust the spacing between subplots
fig.tight_layout()  #Fit subplot(s) in to the figure area
```

Axis Spines

```python
ax1.spines['top'].set_visible(False)  #Make the top axis line for a plot invisible
ax1.spines['bottom'].set_position(('outward',10))  #Move the bottom axis line outward
```