Python_Matplotlib_Cheat_Sheet.pdf

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.

matplotlib

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))
```

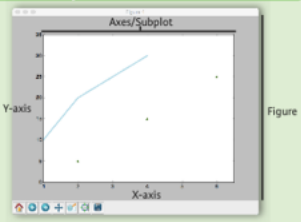
Axes

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)
```

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]                          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')
>>> plt.show()                             Step 6
```

3 Plotting Routines

1D Data

`>>> 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 horiontal 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 o

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 2D vector fields

Data Distributions

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

2D Data or Images

`>>> 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

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)`	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	

`>>> 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	
`>>> 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	

`>>> 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	
`>>> 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

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

DataCamp
Learn Python for Data Science Interactively

① Prepare the Data :→

1-D data –

```
import numpy as np
x = np.linspace(0,10,100)
```

→ is a tuple of 3 values that is passed as arguments to numpy function 'np.linspace()'

→ 0 (Start) and 10 (end) of interval
100 → No. of equally spaced value between start to end.

→ np.linspace used to create 1-dimensional array of evenly spaced numbers over a specified interval.

→ In this case it generate 100 evenly spaced values between 0 to 10 inclusive.

```
y = np.cos(x)
z = np.sin(x)
```

⇒ It creates an array 'y' containing the cosine of each value in 'x' using the 'cos' function from numpy.

→ create array 'z' containing sine of each value in 'x' using 'sine' function from Numpy.

2-D Data or Images –

```
data = 2 * np.random.random((10,10))
```

→ data is 10x10 numpy array of random numbers between 0 and 2.

```
data2 := 3 * np.random.random((10,10))
```

→ data 2 is a 10x10 numpy array of random numbers between 0 and 3.

```
Y, X = np.mgrid[-3:3:100j, -3:3:100j]
```

→ two 100x100 numpy arrays that together create a grid of points ranging from -3 to 3.

```
U = -1 - x**2 + y
V = 1 + x - y**2
```

→ calculate value of U and V at each point on grid.

```
from matplotlib.cbook import, get_sample_data
img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

Create the Plot -

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

Figure -

```
fig = plt.figure()
```
→ creates figure object with default size and configurations.

```
fig2 = plt.figure(figsize = plt.figaspect(2.0))
```
→ creates a figure object with a specific aspect ratio of 2:1, which means width of figure will be twice the height.

This is achieved by passing the 'figaspect' parameter with a value of 2.0 to the 'figsize' argument.

Axes -

```
fig.add_axes()
```
→ creates a new set of axes within the figure.

```
ax1 = fig.add_subplot(221)
```
→ creates a subplot in a 2x2 grid layout, placing it in the first position (top left)

```
ax3 = fig.add_subplot(212)
```
→ creates another subplot in 2x1 grid layout, placing it in second position (bottom)

```
fig3, axes = plt.subplots(nrows = 2, ncols = 2)
fig4, axes2 = plt.subplots(ncols = 3)
```

→ plt.subplots() is a function that returns a figure object and an array of axes object. The first argument passed to the function specifies the number of rows of subplots, and second argument specifies the no. of columns of subplots.

In this case, 'nrows = 2' and 'ncols = 2', so function creates a grid of 2 rows & 2 columns, resulting in 4 subplots in total.

- The function returns two object : 'fig' and 'axes'. 'fig' is a reference to the figure object, and 'axes' is an array of axes object, with dimensions '(nrows, ncols)'.

→ This code creates a figure with three subplots

arranged horizontally in one row. The first subplot is assigned to variable 'axes2' and other two subplots are created but not assigned to any variables.

To modify the code to create only one subplot:-

```
fig, axes = plt.subplots(ncols=1)
```

3 Plotting Routines :-

Plotting routine is a tool used to create visual representations of data that can aid in the analysis and interpretation of the data.

1D Data -

```
lines = ax.plot(x,y)
```

→ used to visualize the trend of data over a period of time.

```
ax.scatter(x,y)
```

→ visualize relationship between two variables.

```
axes[0,0].bar([1,2,3],[3,4,5])
```

→ specifies the x-coordinates or positions where the bars will be plotted along the horizontal axis.

→ specifies the heights of bars to be plotted. The first bar height 3, second's height 4, 3rd height 5.

horizontal bar chart

```
axes[1,0].barh([0.5,1,2.5],[0,1,2])
```

→ [0.5, 1, 2.5] specifies y-coordinates of the bars. [0,1,2] specifies length of bars, 1st bar length 0, 2nd bar length 1, 3rd bar length 2.

```
axes[1,1].axhline(0.45)
```

axhline is a method of 'Axes' class in matplotlib that creates horizontal line. 0.45 specifies y-position in horizontal line.

```
axes[0,1].axvline(0.65)
```

0.65 specifies x-position of vertical line.

```
ax.fill(x,y,color='blue')
ax.fill_between(x,y,color='yellow')
```

ax.( ) is a method of 'Axes' class.

x is array of x-coordinates that define the shape to be filled.

y is array of y-coordinate, color specifies fill color.

2-D Data or Images.