TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

The University of Texas at Austin

Matplotlib 101

S. Charlie Dey, Director of Training and Professional Development

Science in the Cloud, 2019

Matplotlib, What is it?

It's a graphing library for Python. It has a nice collection of tools that you can use to create anything from simple graphs, to scatter plots, to 3D graphs. It is used heavily in the scientific Python community for data visualisation.

Matplotlib, First Steps

Let's plot a simple sin wave from 0 to 2 pi.

First let's, get our code started by importing the necessary modules.

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

Matplotlib, First Steps

Let's add the following lines, we're setting up x as an array of 50 elements going from 0 to 2*pi

```
x = np.linspace(0, 2 * np.pi, 50)
plt.plot(x, np.sin(x))
plt.show() # Show the graph.
```

Let's run our cell!

Matplotlib, a bit more interesting

Let's plot another curve on the axis

```
plt.plot(x, np.sin(x),
         x, np.sin(2 * x))
plt.show()
```

Let's run our cell!

TACC

Matplotlib, a bit more interesting

Let's see if we can make the plots easier to read

```
plt.plot(x, np.sin(x), 'r-o',
         x, np.cos(x), 'g--')
plt.show()
```

Let's run this cell!

Matplotlib, a bit more interesting

```
Colors:
Blue – ‘b’
Green – ‘g’
Red – ‘r’
Cyan – ‘c’
Magenta – ‘m’
Yellow – ‘y’
Black – ‘k’ (‘b’ is taken by blue so the last letter is used)
White  – ‘w’
```

TACC

Matplotlib, a bit more interesting

```
Lines:
    Solid Line – ‘-‘
    Dashed – ‘–‘
    Dotted – ‘.’
    Dash-dotted – ‘-:’
Often Used Markers:
    Point – ‘.’
    Pixel – ‘,’
    Circle – ‘o’
    Square – ‘s’
    Triangle – ‘^’
```

TACC

Matplotlib, Subplots

Let's split the plots up into subplots

```
plt.subplot(2, 1, 1) # (row, column, active area)
plt.plot(x, np.sin(x), 'r')
plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x), 'g')
plt.show()
```

using the subplot() function, we can plot two graphs at the same time within the same "canvas". Think of the subplots as "tables", each subplot is set with the number of rows, the number of columns, and the active area, the active areas are numbered left to right, then up to down.

Matplotlib, Scatter Plots

Let's take our sin curve, and make it a scatter plot

```
y = np.sin(x)
plt.scatter(x,y)
plt.show()
```

call the scatter() function and pass it two arrays of x and y coordinates.

Matplotlib, add a touch of **color**

Let's mix things up, using random numbers and add a colormap to a scatter plot

```
x = np.random.rand(1000)
y = np.random.rand(1000)
size = np.random.rand(1000) * 50
color = np.random.rand(1000)
plt.scatter(x, y, size, color)
plt.colorbar()
plt.show()
```

TACC

Matplotlib, add a touch of **color**

Let's see what we added, and where that takes us

```
...
plt.scatter(x, y, size, color)
plt.colorbar()
...
```

We brought in two new parameters, size and color, which will vary the diameter and the color of our points. Then adding the colorbar() gives us a nice color legend to the side.

Matplotlib, Histograms

A histogram is one of the simplest types of graphs to plot in Matplotlib. All you need to do is pass the hist() function an array of data. The second argument specifies the amount of bins to use. Bins are intervals of values that our data will fall into. The more bins, the more bars.

```
plt.hist(x, 50)
plt.show()
```

Matplotlib, Adding Labels and Legends

Let's go back to our sin/cos curve example, and add a bit of clarification to our plots

```
x = np.linspace(0, 2 * np.pi, 50)
plt.plot(x, np.sin(x), 'r-x', label='Sin(x)')
plt.plot(x, np.cos(x), 'g-^', label='Cos(x)')
plt.legend() # Display the legend.
plt.xlabel('Rads') # Add a label to the x-axis.
plt.ylabel('Amplitude') # Add a label to the y-axis.
plt.title('Sin and Cos Waves') # Add a graph title.
plt.show()
```

Matplotlib, Plotting a DataFrame

Let's look at a generic dataframe

```
ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000',
periods=1000))
ts = ts.cumsum()  ## cumulative sum

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,columns=['A',
'B', 'C', 'D'])
df = df.cumsum()


df.plot(kind='scatter',x='A',y='B',color=blue)
plt.show()
```

TACC

Matplotlib, Plotting a DataFrame

Let's look at a generic dataframe

```
df.plot(kind='scatter',x='A',y='B',color=blue)
df.plot(kind='bar',x='A',y='B')
df.plot(kind='line',x='A',y='B')
```

TACC

Matplotlib, Plotting a DataFrame

Let's look at a generic dataframe

```
ax = plt.gca()

df.plot(kind='line',x='A',y='B',ax=ax)
df.plot(kind='line',x='A',y='C', color='red', ax=ax)

plt.show()
```

Matplotlib, Using a Mesh

Let's go back to our dataframe, and graph out x*x+y*y as a mesh

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import
Axes3D
```

Matplotlib, Using a Mesh

Let's create our function and our empty array and look at the contour

```
z1 = np.empty([2001,2001])
im = plt.imshow(z1, cmap='hot')
fig = plt.colorbar(im,
orientation='horizontal')
plt.show(fig)
```

TACC

Matplotlib, Using a Mesh

Filling out an array manually...

```
i = 0
j = 0
t = 1
z = np.empty([2001,2001])
for x in np.arange(-10,10,.01):
    i = i + 1
    j = 0
    for y in np.arange(-10,10,.01):
        j = j + 1
        z[i][j] = x*x + y*y
```

TACC

Matplotlib, Using a Mesh

Here we can simplify it!

```
def f1(x,y):
    return (x*x+y*y)

x = np.linspace(-10, 10, 2000)
y = np.linspace(-10, 10, 2000)

X, Y = np.meshgrid(x, y)
Z = f1(X, Y)
```

TACC

Matplotlib, Using a Mesh

And now plot it out

```
fig1 = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='hot')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z');
ax.view_init(45, 0)
```

TACC

Questions? Comments?