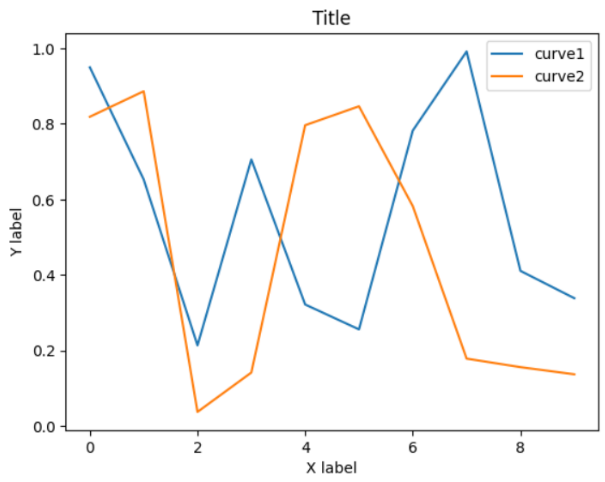
Graphics in Python : Matplotlib

Monika Baloda

Matplotlib is the main library for graphics in python. Large part of data visualization is done here. It can produce from simple summary plots like boxplot to complex correlation maps between manu variables. We import this library as *import matplotlib.pyplot as plt.* Now we can use any function of this library by just writing *plt.function()* where function() is a name of the desired function we want to use.

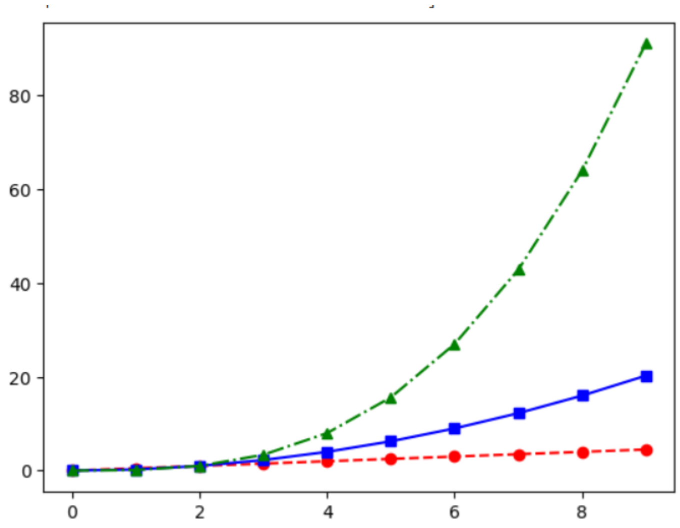
1. Making our first graph/plot in Python
   Our first code will import the library as described above. In order to make our life easy and facilitate the dealings with numbers we all import *numpy* library. We write the following code:
   ```
   import matplotlib.pyplot as plt
   import numpy as np
   ```
   Now to make our first plot, we need to generate some data. The following codes generate two variables *x* and *y* :
   ```
   x = np.linspace(-5,5,100)
   y = 5*x
   ```
   To make our first plot we can just write the following command :
   ```
   plt.plot(x,y)
   ```
2. Adding information to the plot:
   We would ideally like to label the axis, write a title on the plot. The following code does it for us:
   ```
   plt.plot(np.random.rand(10))  #creates first curve using random numbers
   plt.plot(np.random.rand(10)) #creates second curve using random numbers
   plt.legend(["curve1","curve2"]) #puts a legend by naming which curve is which
   plt.xlabel("X label")           #labels x axis
   plt.ylabel("Y label")          #labels y axis
   plt.title("Title")            #puts a title
   ```

The plot looks like the following:



3. Formatting the style of our plot

   The Python library gives us greater control over the color, marker and linestyle of our plot. While color is easily understandable, by ‘marker’, we mean how our datapoint should look like, a hollow-circle, solid triangle, a pentagon. The linestyle is about how our lines should look : whether solid, dotted, dashed, dashdots or some other formats.
   To illustrate this, we plot three lines, each being differently formatted:



The code used for this plot is:

t = np.arange(0, 5, 0.5) #this creates and stores data in t
print(t)   #this prints data
*plt.plot(t, 'ro--', t**2, 'bs-', t**3, 'g^-.') #this is the main code which generates the plot*

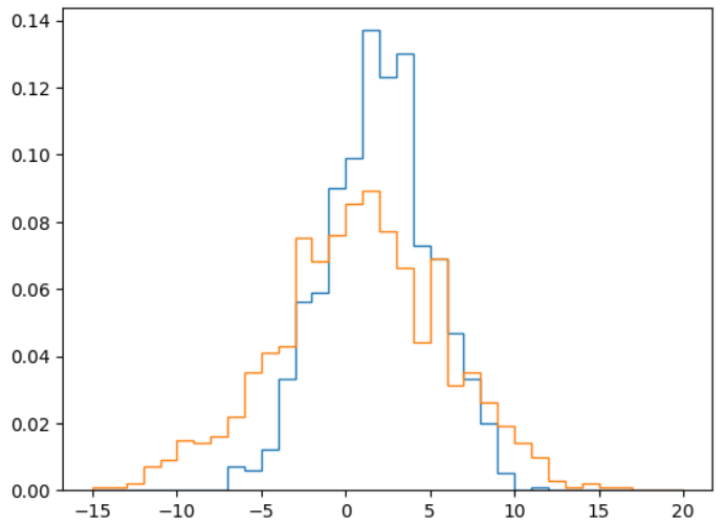
Here ‘bs-’ means blue color, squared marker and solid line while ‘ro–’ means red color, circle marker and dotted lines. We can similarly make many different types of lines. The detailed discussion on color is here, marker is here and linestyle is here.

4. <u>Plotting Numerical Variables : Histograms</u>
   Histogram is the plot which shows us the discrete frequency distribution of a given variable. To put our learning above in the action, we use it on histograms. This is particularly useful for continuous variables. To make a histogram, we first generate the data using numpy functions and then create a histogram with additional arguments. The code is:
   #create data
   np.random.seed(206)
   x1 = np.random.normal(2,3,1000)
   x2 = np.random.normal(1,5,1000)

   #plotting simple histogram
   plt.hist(x1)

   #plot two histogram on same graph with some additional style
   plt.hist(x1, bins=np.arange(-15, 21), histtype='step', density=True)
   plt.hist(x2, bins=np.arange(-15, 21), histtype='step', density=True)

   To save some space, we are just giving the last plot. It looks like the following:



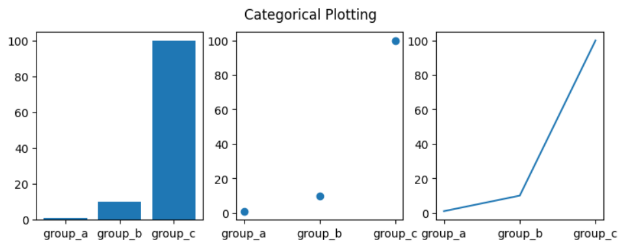
   For detailed discussion on histograms in python, one go to the link here: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

5. <u>Plotting Categorical Variables: Bargraphs</u>
   Categorical variables are different from numeric variables because they may not be comparable or ordering may not be possible. Similar to what we did above, we generate the variables and then different from above, we generate three side by side graphs:
   The detailed discussion is available here. The code is:
   names = ['group_a', 'group_b', 'group_c']
   values = [1, 10, 100]

```
plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```

The plot is :


Categorical Plotting
group_a
group_b
group_c