**Matplotlib Library**

Pyplot is a collection of functions that make matplotlib work like MATLAB.

Pyplot provides a state machine interface to the matplotlib library. We mostly use NumPy arrays to store the data we like to plot.

In common practise we do the following:

*import matplotlib.pyplot as plt*

the basic very useful function is plt.plot() which can be used to plot lines and markers.

Example:

*plt.plot([0,1,4,9,16])* 🡨 The plot is done by taking X-axis values as the index an Y -axis values

as the data at that index

*plt.show()* #can be used in Python to actually show generated plots

We can also generate graphs using this syntax:

*plt.plot(x,y,”bo-”,<keyword\_arguments>)* where x and y are vectors or lists of data, b=blue,o=circles,-=solid line, thus we get blue circled , markers

3rd argument examples:

“bo-” , ”gs-” , “rd-”

Keyword arguments:

linewidth=<size\_int>

markersize-<size\_int>

Customizing Plots:

* Add legend*: legend(loc=”<location\_string>”)*
* Adjust axes: *axis([xmin, xmax, ymin, ymax])*
* Set axis labels: *xlabel(‘<label\_string>’), ylabel(‘<name\_string>’)*
* Save figure: *savefig(“name.filetype”)*

Plt also knows LaTeX

Example script:

*import numpy as np*

Chart, line chart

Description automatically generated*import matplotlib.pyplot as plt*

*x=np.linspace(0,10,20)*

*y1=x\*\*2*

*y2=x\*\*1.5*

*plt.plot(x,y1,"rd-",linewidth=1,markersize=4,label="First")*

*plt.plot(x,y2,"gd-",linewidth=1,markersize=4,label="Second")*

*plt.xlabel("$X$")*

*plt.ylabel("$Y$")*

*plt.axis([-0.5, 10.5, -5, 105])*

*plt.legend(loc="upper left")*

*#plt.savefig("myplot.pdf")*

Using Logarithmic Axes:

*semilogx()*

*semiology()*

*loglog()*

We can use these functions just like plt.plot(). But these will plot the log values. To linearly space the values on the loglog plot, we change the dataset x to logspace with the right parameters.

Example script:

*import numpy as np*

*import matplotlib.pyplot as plt*

Chart, line chart

Description automatically generated*x=np.logspace(-1,1,20)*

*y1=x\*\*2*

*y2=x\*\*1.5*

*plt.loglog(x,y1,"rd-",linewidth=1,markersize=4,label="First")*

*plt.plot(x,y2,"gd-",linewidth=1,markersize=4,label="Second")*

*plt.xlabel("$X$")*

*plt.ylabel("$Y$")*

*plt.axis([-0.5, 10.5, -5, 105])*

*plt.legend(loc="upper left")*

*#plt.savefig("myplot.pdf")*

Histograms

plt.hist(x) # x is the array of data, by default hist uses 10 evenly spaced bins

Ex:

import numpy as np

x = np.random.normal(size=1000)

plt.hist(x)

Many optional arguments for the hist() function. Some of them are:

normed=True (density=True) 🡨 y-axis will have proportions of observation rather than number of observations, i.e. the histogram will be normalized. NOTE: better to use density due to exceptions in certain cases (result same as normed)

bins=<np.vector> 🡨 x axis data group of equal width; for n groups, n+1 data items needed in vector

cumulative=True 🡨 A cumulative figure will be plotted

histtype=’step’ 🡨 It was become a step histrogram

Gama distribution

np.random.gamma(<shape\_k>,<scale\_Ø>, <num of data> ) returns an array of gramma distributed data

subplot() enables us to get several subplots within each figure.

Syntax

subplot(<no. rows> , <no. columns> , <plot\_number>) #without comma also works

Example Script:

*x=np.random.gamma(5,10,100000)*

*plt.figure()*

*plt.subplot(2,2,1)*

*Chart, histogram

Description automatically generatedplt.hist(x,bins=30)*

*plt.subplot(2,2,2)*

*plt.hist(x,bins=30,density=True)*

*plt.subplot(2,2,3)*

*plt.hist(x,bins=30,cumulative=True)*

*plt.subplot(2,2,4)*

*plt.hist(x,bins=30,density=True,cumulative=True)*