

Computational Environments and Toolchains

Topic 02 — Scientific Computing using Python

Lecture 01 — Matplotlib

Kieran Murphy and David Power

Department of Computing and Mathematics,
SETU (Waterford).
(kieran.murphy@setu.ie, david.power@setu.ie)

Autumn Semester, 2024/25

Outline

- matplotlib (2D and 3D plotting library)
- numpy (high performance matrix library)
- scipy (scientific computation library)

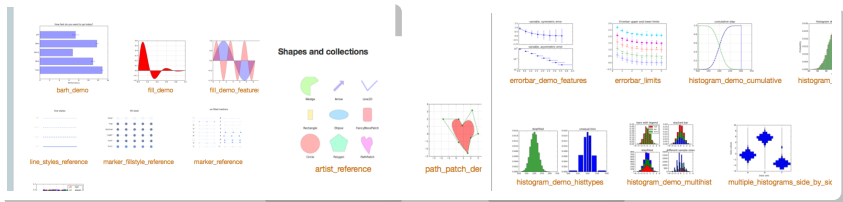
Outline

Mathplotlib — 2D and 3D plotting library

1.1	Introduction	3
1.2	Examples	6

A 20 Second Intro to Matplotlib

- Matplotlib is the standard plotting library for scientific python
 - Design objectives
 - Plots should look great – publication quality.
 - Supports formats suitable for inclusion with \TeX documents.
 - Embeddable in a graphical user interface for application development.
 - Code should be easy enough to understand and extendable.
 - Mostly it is for 2D data (including surface plots of $f(x, y)$, etc.)
 - Active development with lots of new features
 - Best way to figure out how to do something: look at the gallery
- <http://matplotlib.org/gallery.html>



Importing Matplotlib

- There are several interfaces to matplotlib that provide varying amounts of access to its underlying functionality

http://matplotlib.org/faq/usage_faq.html

- `matplotlib` is the entire package.
- `pyplot` is a module within matplotlib that provides easy access to the core plotting routines.
- `pylab` combines `pyplot` and `numpy` into a single namespace to give a MatLab like interface
 - This is recommended for interactive work but I tend to just import both `pylab` and `numpy` separately.
- A number of toolkits extend the functionality
 - `basemap` and `cartopy`: mapping (e.g. projecting onto a globe, geographical boundaries)
 - `mplot3d`: basic 3-d plotting.
 - `AxesGrid`: high-level methods for arranging multiple plots together in a figure

Mathplotlib Backends

`matplotlib` can output to a number of different devices—the backends provide this functionality:

- Interactive backends (allow for plotting to the screen, and updates (animations) with each command (if desired)):
 - `pygtk`, `wxpython`, `tkinter`, `qt`, `macosx`
- Hardcopy backends (saving to external image files):
 - PNG, SVG, PDF, PS

To select a backend:

`backend.py`

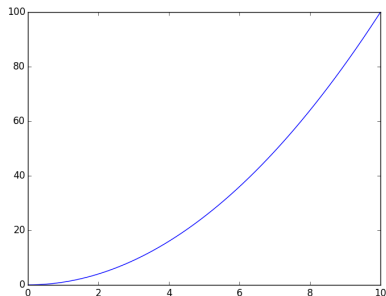
```
1 import matplotlib
2 matplotlib.use("PS")
3 import matplotlib.pyplot as plt
```

In `iPython`, use the `%matplotlib inline` to get images embedded in the notebooks.

Line Plot (Basic)

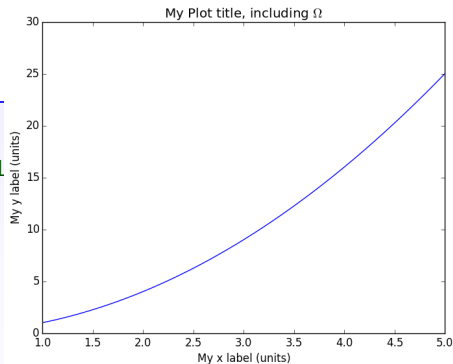
line_plot.py

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 x = np.linspace(0, 10, 1000) # evenly spaced points
9 y = np.power(x, 2)           # element-wise squaring
10
11 plt.plot(x, y)
12
13 plt.savefig("line_plot.png")
```

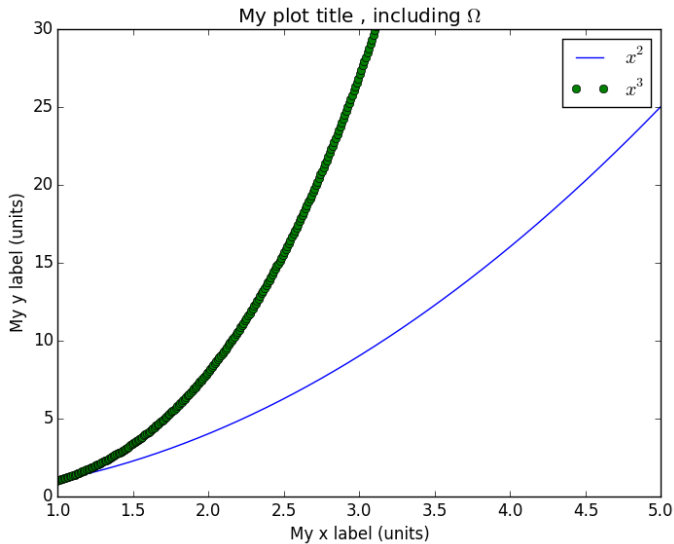


Line Plot (Proper)

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 x = np.linspace(0, 10, 1000)
9 y = np.power(x, 2)
10
11 plt.plot(x, y)
12 plt.xlim((1, 5))
13 plt.ylim((0, 30))
14 plt.xlabel("My_x_label_(units)")
15 plt.ylabel("My_y_label_(units)")
16 plt.title("My_Plot_title,_including_\\Omega$")
17
18 plt.savefig("line_plot_proper.png", bbox_inches="tight")
```



Line Plot (Fancy)

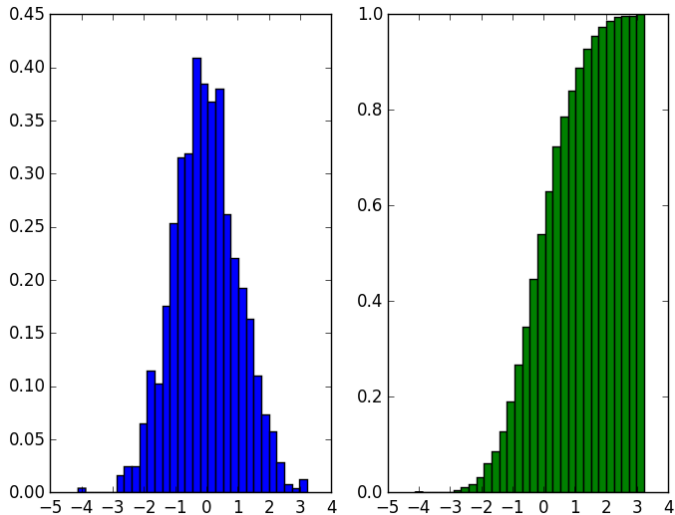


Line Plot (Fancy) — Code

line_plot_fancy .py

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 x = np.linspace(0, 10, 1000)
9 y1 = np.power(x, 2)
10 y2 = np.power(x, 3)
11
12 plt.plot(x, y1, "b-", x, y2, "go")           # look at me !
13 plt.xlim((1, 5))
14 plt.ylim((0, 30))
15 plt.xlabel("My_x_label_(units)")
16 plt.ylabel("My_y_label_(units)")
17 plt.title("My_plot_title_,_including_\Omega$")
18 plt.legend(("x^2$", "x^3$"))                 # and me !
19
20 plt.savefig("line_plot_fancy.png", bbox_inches="tight")
```

Histogram

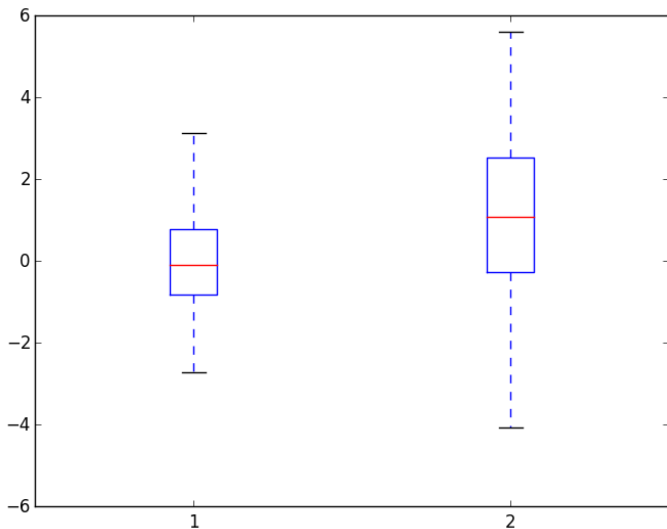


Histogram — Code

`histogram.py`

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 data = np.random.randn(1000)
9
10 # histogram (pdf)
11 plt.subplot(1, 2, 1)
12 plt.hist(data, bins=30, normed=True, facecolor="b")
13
14 # empirical cdf
15 plt.subplot(1, 2, 2)
16 plt.hist(data, bins=30, normed=True, color="g", cumulative=True)
17
18 plt.savefig("histogram.png", bbox_inches="tight")
```

Box Plot



Box Plot — Code

box_plot.py

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 samp1 = np.random.normal(loc=0., scale=1., size=100)
9 samp2 = np.random.normal(loc=1., scale=2., size=100)
10 plt.boxplot((samp1, samp2))
11
12 plt.savefig("box_plot.png", bbox_inches="tight")
```

In descriptive statistics, a box plot is a convenient way of graphically depicting groups of numerical data through their quartiles — min, first quartile, median (second quartile), third quartile, and max.

https://en.wikipedia.org/wiki/Box_plot

Image Plot

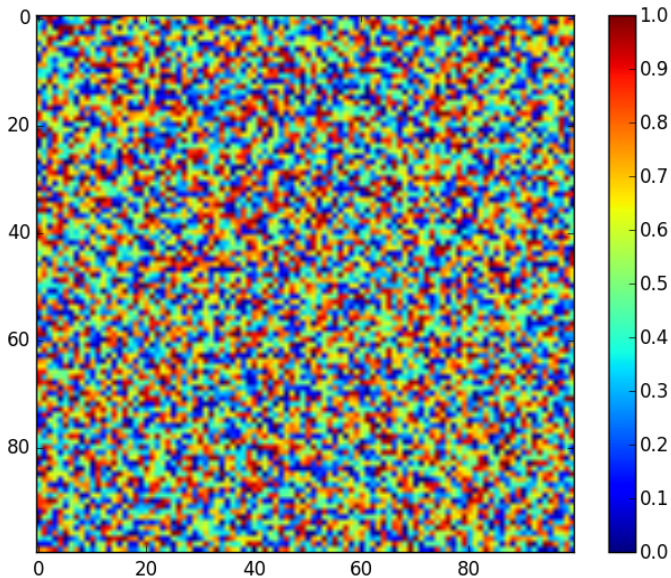
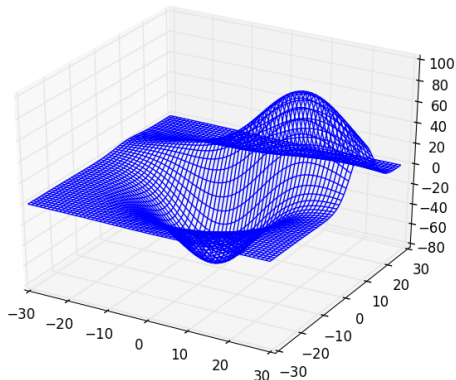


Image Plot — Code

image_plot.py

```
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 A = np.random.random((100, 100))
9 plt.imshow(A)           # Display an image on the axes
10 plt.hot()               # set colormap to hot
11 plt.colorbar()          # Add a colorbar to a plot
12
13 plt.savefig("image_plot.png", bbox_inches="tight")
```

Wire Plot



```
5 from mpl_toolkits.mplot3d import axes3d
6 import matplotlib.pyplot as plt
7
8 ax = plt.subplot(111, projection="3d")
9
10 X, Y, Z = axes3d.get_test_data(0.1)
11 ax.plot_wireframe(X, Y, Z)
12
13 plt.savefig("wire_plot.png", bbox_inches="tight")
```