Matplotlib

Neelofer Banglawala nbanglaw@epcc.ed.ac.uk
Kevin Stratford kevin@epcc.ed.ac.uk

Original course authors:
Andy Turner
Arno Proeme

Reusing this material

www.archer.ac.uk

support@archer.ac.uk

[Matplotlib] What is matplotlib? I

- Matplotlib is a plotting library for Python
  - Philosophy : “make the easy things easy and the hard things possible”.
- Capable of:
  - interactive and non-interactive plotting
  - Producing publication-quality figures
- Large amount of functionality:
  - Scientific and statistical plots
  - Heatmaps, contours, surfaces
  - Geographical and map-based plotting
- Closely integrated with numpy
  - Use numpy functions for reading data
  - As data is in numpy, matplotlib can plot it easily
- Documentation:
  - http://matplotlib.org/

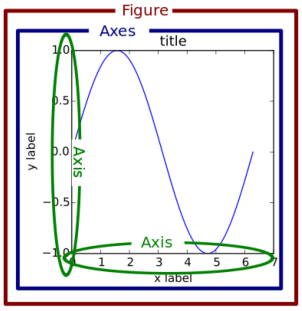
[Matplotlib] What is matplolib? II

- People often want to have a quick look at data in a plain text file
- Gnuplot/Excel often used for this but matplotlib can provide a simple, feature-rich replacement
- Manipulate data interactively and replot
- Can save the session to keep record of what you did if required

Creating high-quality plots is easy in matplotlib

[Matplotlib] Basic concepts

- Everything is assembled by Python commands
- Create a figure with an axes area (this is the plotting area)
- Can create multiple plots in one figure
- Only one figure (or axes) is active at a given time (i.e. current figure, current axes)
- In an IPython shell you can plot to the screen (interactive mode) or you can save to image (non-interactive mode)
- Can use the *show()* command in, for example, a Python script to display the plot



matplotlib.pyplot contains the high-level functions we need to do all the above and more

[Matplotlib] Basic plotting

Launch an IPython shell, import *pyplot* and *numpy*

In [ ]:
```
# add 'inline' option if using a notebook
%matplotlib inline
import matplotlib.pyplot as plt; import numpy as np
```

In [ ]:
```
xmin=0; xmax=10; pts = 50;
x = np.linspace(xmin, xmax, pts);
y = np.cos(x);
```

In [ ]:
```
# line, markers, 2 plots, fig, title then plot
plt.plot(x,y,'ro'); # , x, y, 'g-'# #
```

[Matplotlib] Saving images to file

- Saving to image file is simple using *savefig*
- File format is determined from the extension you supply
- Resolution set using dpi option
- Commonly supports: png, jpg, pdf, ps

In [ ]:
```
# save image to file in different formats
plt.savefig("cos_plot.pdf");
plt.savefig("cos_plot.png", dpi=300);  # higher resolution (dpi)
```

Time to create some plots. Please complete **Basic Plotting** (pages 1 - 7) of the Matplotlib exercise.

[Matplotlib] What is a backend? (++)

Matplotlib consists of two parts, a frontend and a backend:

- Frontend : the user facing code i.e the plotting code
- Backend : does all the hard work behind-the-scenes to make the figure

By offering different backends, Matplotlib can support a wide range of different use cases and output formats. There are two types of backend:

- User interface, or "**interactive**", backends
- Hardcopy, or "**non-interactive**", backends to make image files
  - e.g. Agg (png), Cairo (svg), PDF (pdf), PS (eps, ps)
  - These are known as rendering engines and determine how your image is drawn
- Check which backend is being used with: *matplotlib.get_backend()*
  - Default backend on ARCHER is Qt4Agg
- Switch to a different backend with *matplotlib.use(...)*
  - Must issue command **before** importing matplotlib.pylot (or *%matplotlib*)

[Matplotlib] What is interactive mode? (++)

Here we mean that a figure displays to screen as soon as you call either *plt.figure()* or *plt.plot()*.

- Furthermore, the displayed figure does not prevent you from issuing commands in the IPython shell. This means you can update the figure and see the resulting changes immediately.

In contrast, in **"non-interactive"** mode, the figure will not display to screen, unless you call *show()*. This is what happens when you create figures in scripts.

- If you show the figure, it "blocks" any further commands being issued in the shell until you have to closed the figure.

To confuse things, an "interactive" backend does not guarantee your figures will automatically display to screen. Matplotlib has a Boolean variable in its configuration file (the *matplotlibrc* files, more of that later) that sets the interactivity.

- You can query this with: *matplotlib.is_interactive()*

In most cases you don't need to worry about this. The easiest way to ensure interactivity is to launch an IPython shell with the *--matplotlib* command or to issue *%matplotlib* within the IPython shell before issuing any other command.

[Matplotlib] Plot customisations

There are many ways to customise a plot. Play with the following properties.

In [ ]:
```
# Ex: set the figure size and add a plot
fig=plt.figure(figsize=(4,4));
plt.plot(x,y,'c-')
```

In [ ]:
```
# Ex: linewidth, and
# linestyles: '-', '.-', ':', '--'
plt.plot(x,y,'k-',linewidth=2.0)
```

In [ ]:
```
# Ex: markers and their properties
# unfilled markers: '.',+','x','1' to '4','|'
plt.plot(x,y,'x',markersize=10)
```

In [ ]:
```
# filled markers: 'o', 's','*','d','>','^','v', 'p', 'h'
plt.plot(x,y,'8',markerfacecolor='None',markeredgecolor='g',
         markersize=10)
```

[Matplotlib] Plot customisations II

Set x-axis and y-axis limits, adjust title font properties

In [ ]:
```
# Ex: x,y, axis limits:
plt.xlim((xmax*0.25,xmax*0.75));
plt.ylim((np.cos(xmin*0.25),np.cos(xmax*0.75)));
plt.plot(x,y,'mo-')
```

In [ ]:
```
# Ex: title placement and font properties
plt.plot(x,y,'x')
plt.suptitle('A Centered Title', fontsize=20)
# loc: center, left, right
# verticalalignment: center, top, bottom, baseline
plt.title('A Placed Title', loc='left', verticalalignment='top' )
```

[Matplotlib] Plot customisations III

Add tickmarks and annotations.

In [ ]:
```
# Ex: tick marks
fig=plt.figure(figsize=(4,3.5)); plt.plot(x,y,'x');
nticks = 5;
tickpos = np.linspace(xmin,xmax,nticks);
labels = np.repeat(['tick'],nticks);
plt.xticks(tickpos, labels, rotation='vertical');
```

In [ ]:
```
# Ex++: arrows and annotations
plt.plot(x,y,'x');
atext='annotate this'; arrowtip=(1.5,0.5); textloc=(3, 0.75);
plt.annotate(atext, xy=arrowtip, xytext=textloc,
             arrowprops=dict(facecolor='black', shrink=0.01),)
```

[Matplotlib] Subplots

- There can be multiple plots, or subplots, within a figure
- Use *subplot(nrows, ncols, plot number)* to place plots on a regular grid
- The most recently created subplot is the current plot

[Matplotlib] Subplots II

- Can move between subplots by creating each subplot with a "handle" for each **axes**

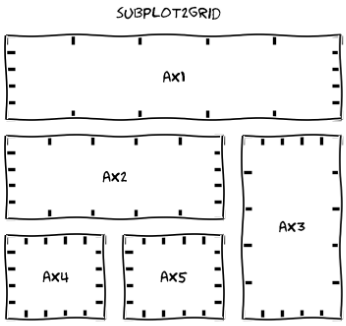
In [ ]:
```
(fig, axes) = plt.subplots(nrows=2, ncols=2);
axes.size

axes[0,0].plot(x,y,'g-');
axes[1,1].plot(x,y,'r-');
```

- Control space between subplots

In [ ]:
```
# subplots_adjust(left=None, bottom=None, right=None, top=None,
# wspace=None, hspace=None)
plt.subplots_adjust(hspace=0.001)
```

[Matplotlib] Advanced : subplot2grid

- For more control over subplot layout, use *subplot2grid*
- Subplots can span more than one row or column



In [ ]:
```
# Ex++: subplot2grid(shape, loc, rowspan=1, colspan=1)
fig = plt.figure()
ax1 = plt.subplot2grid((3, 3), (0, 0)); ax1.plot(x,y,'r-');
ax2 = plt.subplot2grid((3, 3), (0, 1), colspan=2); ax2.plot(x,y,'g-');
ax3 = plt.subplot2grid((3, 3), (1, 0), colspan=2, rowspan=2); ax3.plot(x,y,'b-'
);
ax4 = plt.subplot2grid((3, 3), (1, 2), rowspan=2); ax4.plot(x,y,'c-');
```

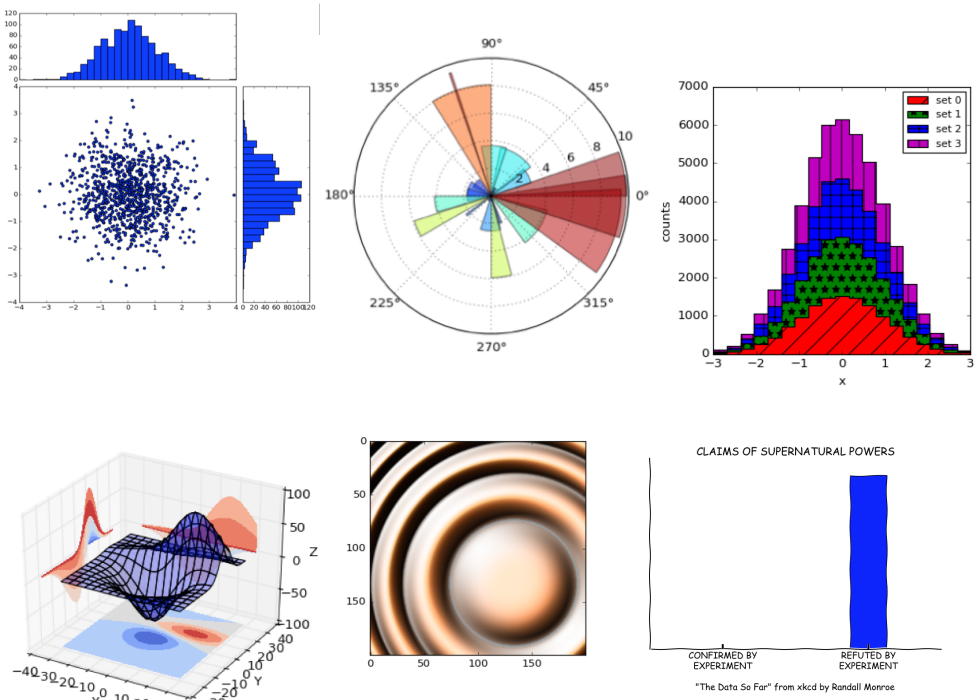
[Matplotlib] Customise some subplots

- Go back to the Matplotlib exercise and **create multiple customised plots** (pages 8 - 11)

[Matplotlib] Other type of plots

Only just skimmed the surface of what is possible

http://matplotlib.org/gallery.html



Notes

- the above slide looks terrible but it looks reasonable in a slideshow... promise...

[Matplotlib] Advanced : animation

Can even create animations (from Nicolas P. Rougier, https://github.com/rougier)

In [ ]:
```
#from matplotlib import use
## animation doesn't work with macosx backend!
#use("nbagg")
#import earthquakes;
```

[Matplotlib] Images for publication

- Matplotlib uses ***matplotlibrc*** configuration files to customize and set defaults for all kinds of properties (*rc settings*, *rc parameters*)
- You will most likely want different settings for each journal
- useful to keep a different *matplotlibrc* file for each journal

From Damon McDougall: http://bit.ly/1jluuU0

[Matplotlib] *matplotlibrc* settings

import a particular settings file with:

```
from matplotlib import rc_file.rc_file('/path/to/my/matplotlibrc')
```

Font sizes and types:

axes.labelsize : 9.0 # fontsize of the x any y labels
xtick.labelsize : 9.0 # fontsize of the tick labels
ytick.labelsize : 9.0 # fontsize of the tick labels
legend.fontsize : 9.0 # fontsize in legend
font.family : serif
font.serif : Computer Modern Roman
Marker size : lines.markersize : 3

Use TeX to format all text : text.usetex : True

This is only available with the Agg, PS, PDF backends

[Matplotlib] Settings for a nice figure ratio

Here are some settings you can use to create a nice figure ratio

```
WIDTH = 500.0 # Figure width in pt (usually from LaTeX)
FACTOR = 0.45 # Fraction of the width you'd like the figure to use
widthpt = WIDTH * FACTOR
inperpt = 1.0 / 72.27
# use the Golden ratio because it looks good
golden_ratio = (np.sqrt(5) - 1.0) / 2.0
widthin = widthpt * inperpt
heightin = widthin * golden_ratio
figdims = [widthin, heightin] # Dimensions as list
fig = plt.figure(figsize=figdims)
```

[Matplotlib] Include images in LaTeX

When you include the figure in the LaTeX source you should specify the scale factor as the width:

```
\begin{figure}
  \includegraphics[width=0.45\textwidth]{figure.pdf}
\end{figure}
```

Complete the Matplotlib exercise and **create a publication standard image** (pages 12 - 13)

[Matplotlib] Summary

- Simple, interactive plotting:
  - NumPy allows you to easily read data
- Plotting syntax is simple and concise
- Complex plotting types also available
  - Can start from code for simple plots
  - Many examples available online
- Producing publication-ready images is relatively simple
  - Easily customised for different scenarios
- The more you use matplotlib, the more you get out of it!
- Other packages
  - Bokeh : interactive visualisation library.