

Exercise: Getting started with Matplotlib

Aim: Introduce the Matplotlib interactive plotting tool

Issues covered:

- Importing Matplotlib
- Using the interactive plotting tool
- Generating some simple line graphs
- Saving a figure

If you get bored of having to close the interactive plotting window use:

```
plt.pause(0.1)
```

instead of:

```
plt.show()
```

1. Let's import Matplotlib and create our first plot.

- Type the "import matplotlib.pyplot as plt" in the Python prompt.
- Plot the line defined by `range(10)`.
- Display the plot using `plt.show()`.
- Click the zoom button and then highlight a rectangle in the centre of the plot.
- Click the pan button and then move around the plot (whilst zoomed in).
- Click the back and forward buttons to move through a history of the plots you have generated.
- Click the save button and save your plot as a PNG file.
- Finally, close the plot using the "X" button in the top right corner.

2. Let's create a pretty plot of save chemistry data.

- Our data set is:
Time (decade): 0, 1, 2, 3, 4, 5, 6.
CO₂ concentration (ppm): 250, 265, 272, 260, 300, 320, 389
- Create a line graph of CO₂ versus time. View the plot.
- Re-draw the graph with a blue dashed line.
- Add a title and axis titles to the plot.

3. Let's add a second line to our example.

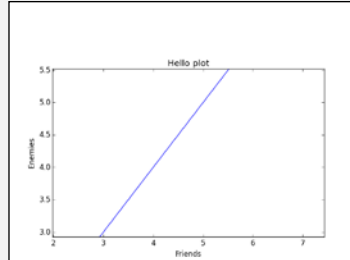
- Continuing with the above data plot, add some additional data:
Temp (°C): 14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2
- Save the output (using Python code) to a PDF file.

Solution: Getting started with Matplotlib

1.

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(range(10))
>>> plt.show()
```

E.g.



2.

```
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> plt.plot(times, co2)
>>> plt.plot(times, co2, 'b--')
>>> plt.title("Concentration of CO2 versus time")
>>> plt.ylabel("[CO2]")
>>> plt.xlabel("Time (decade)")
>>> plt.show()
```

3.

```
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> temp = [14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2]
>>> plt.plot(times, co2, 'b--', times, temp, 'r*-')
>>> plt.show()
>>> plt.savefig("co2_temp.pdf")
```

Exercise: Multiple axes and multiple graphs

Aim: Introduce plotting with multiple axes and multiple graphs on the page

Issues covered:

- Plotting lines with different axes
- Using the subplot function to create multiple graphs on a single page

If you get bored of having to close the interactive plotting window use:

```
plt.pause(0.1)
```

instead of:

```
plt.show()
```

1. Let's re-use our previous example with different axes.

- Import pyplot as "plt" (as before).
- Run the line: `fig, ax1 = plt.subplots()`
- You can now create your first plot using "ax1" instead of "plt".
- Our data set is:
Time (decade): 0, 1, 2, 3, 4, 5, 6.
CO2 concentration (ppm): 250, 265, 272, 260, 300, 320, 389
- Create a line graph of CO₂ versus time. Do not view the plot yet.
- Set the y-axis label to "[CO2]" using the "ax1.set_ylabel" method.
- Get a second axis object using: `ax2 = ax1.twinx()`
- Plot the following temperature values to this second axis:
Temp (°C): 14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2
- Set the second y-axis label to "Temp (degC)" using the "ax2.set_ylabel" method.
- Display the plot using `"plt.show()"`.

2. Let's draw three graphs side by side on a single page.

- Use the "subplot" function to select the first of three plots (side-by-side).
- Plot a line of values: `range(0, 10, 1)`.
- Select the second plot with "subplot".
- Plot a line of values: `range(10, 0, -1)`.
- Select the third plot with "subplot".
- Plot a line of values: `[4] * 10`
- Display the plot using `"plt.show()"`.

Solution: Multiple axes and multiple graphs

1.

```
>>> import matplotlib.pyplot as plt
>>> fig, ax1 = plt.subplots()
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> ax1.plot(times, co2, "b--")
>>> ax1.set_ylabel("[CO2]")
>>> ax2 = ax1.twinx()
>>> temp = [14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2]
>>> ax2.plot(times, temp, "r*-")
>>> ax2.set_ylabel("Temp (degC)")
>>> plt.show()
```

2.

```
>>> plt.subplot(1, 3, 1)
>>> x = range(0, 10, 1)
>>> plt.plot(x)
>>> plt.subplot(1, 3, 2)
>>> y = range(10, 0, -1)
>>> plt.plot(y)
>>> plt.subplot(1, 3, 3)
>>> z = [4] * 10
>>> plt.plot(z)
>>> plt.show()
```

Exercise: Plotting gridded data on a map

Aim: Introduce plotting gridded data using Basemap

Issues covered:

- Importing Basemap
- Using Basemap for geospatial plotting
- Integration with Matplotlib

1. Let's grab some data from a NetCDF file and quickly plot it.

- The file "example_data/tas.nc" contains surface air temperature differences. We can extract the data and prepare it by importing the "example_code/map_data.py" module.
- Import everything to the local scope: `from example_code.map_data import *`
- The following variables now exist in the local scope: `tas` (temperature), `lons` (longitudes for all grid boxes), `lats` (latitudes for all grid boxes).
- Import Basemap and Pyplot with:

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
```
- Create a new figure: `fig = plt.figure()`
- Set up a Basemap instance with a regular lat/lon coordinate reference system:

```
m = Basemap(projection='cyl', llcrnrlat=-90, urcrnrlat=90,
            llcrnrlon=-180, urcrnrlon=180, resolution='c')
```
- Add coastlines: `m.drawcoastlines()`
- Create a "Jet" colour map to plot the data:

```
im1 = m.pcolormesh(lons, lats, tas, shading='flat',
                  cmap=plt.cm.jet, latlon=True)
```
- Save the plot as "tas1.png".
- Display the plot.

2. Let's jazz up the plot by adding some features.

- Follow the instructions above but this time we'll add in some features.
- Add a title: "Change in Surface Air Temperature from MOHC HadGEM2-ES"
- Add some vertical and horizontal grid lines using:

```
m.drawparallels(np.arange(-90.,99.,30.))
m.drawmeridians(np.arange(-180.,180.,60.))
```
- Add a colour bar after generating the colour map "im1", with:

```
cb = m.colorbar(im1, "bottom", size="5%", pad="2%")
```
- Save the plot as "tas2.png". Compare the plot with that produced above.
- Display the plot.

Solution: Plotting gridded data on a map

1.

```
>>> from example_code.map_data import *
>>> from mpl_toolkits.basemap import Basemap
>>> import matplotlib.pyplot as plt

>>> fig = plt.figure()
>>> m = Basemap(projection='cyl', llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180,
urcrnrlon=180, resolution='c')

>>> m.drawcoastlines()
>>> im1 = m.pcolormesh(lons, lats, tas, shading='flat', cmap=plt.cm.jet,
latlon=True)

>>> plt.savefig("tas1.png")
>>> plt.show()
```

2.

```
>>> from example_code.map_data import *
>>> from mpl_toolkits.basemap import Basemap
>>> import matplotlib.pyplot as plt

>>> fig = plt.figure()
>>> plt.title('Change in Surface Air Temperature from MOHC HadGEM2-ES')
>>> m = Basemap(projection='cyl', llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180,
urcrnrlon=180, resolution='c')

>>> m.drawcoastlines()
>>> m.drawparallels(np.arange(-90.,99.,30.))
>>> m.drawmeridians(np.arange(-180.,180.,60.))

>>> im1 = m.pcolormesh(lons, lats, tas, shading='flat', cmap=plt.cm.jet,
latlon=True)
>>> cb = m.colorbar(im1, "bottom", size="5%", pad="2%")

>>> plt.savefig("tas2.png")
>>> plt.show()
```

The plots should look like this:

