



Python for Astronomers

matplotlib (1)

matplotlib

- library for working with 2D plots
- project was started to emulate matlab
- Three main parts:
 - the **pylab** interface
 - the **matplotlib API** (matplotlib frontend)
 - the **backends** (GTK, GTKAgg, PostScript, PDF, png, ...)

ipython -pylab

- matplotlib can be used interactively and non-interactively
- start an interactive session for matplotlib with
ipython -pylab

simple plotting

```
>>> plot([1,2,4])
```

```
>>> ylabel("just numbers")
```

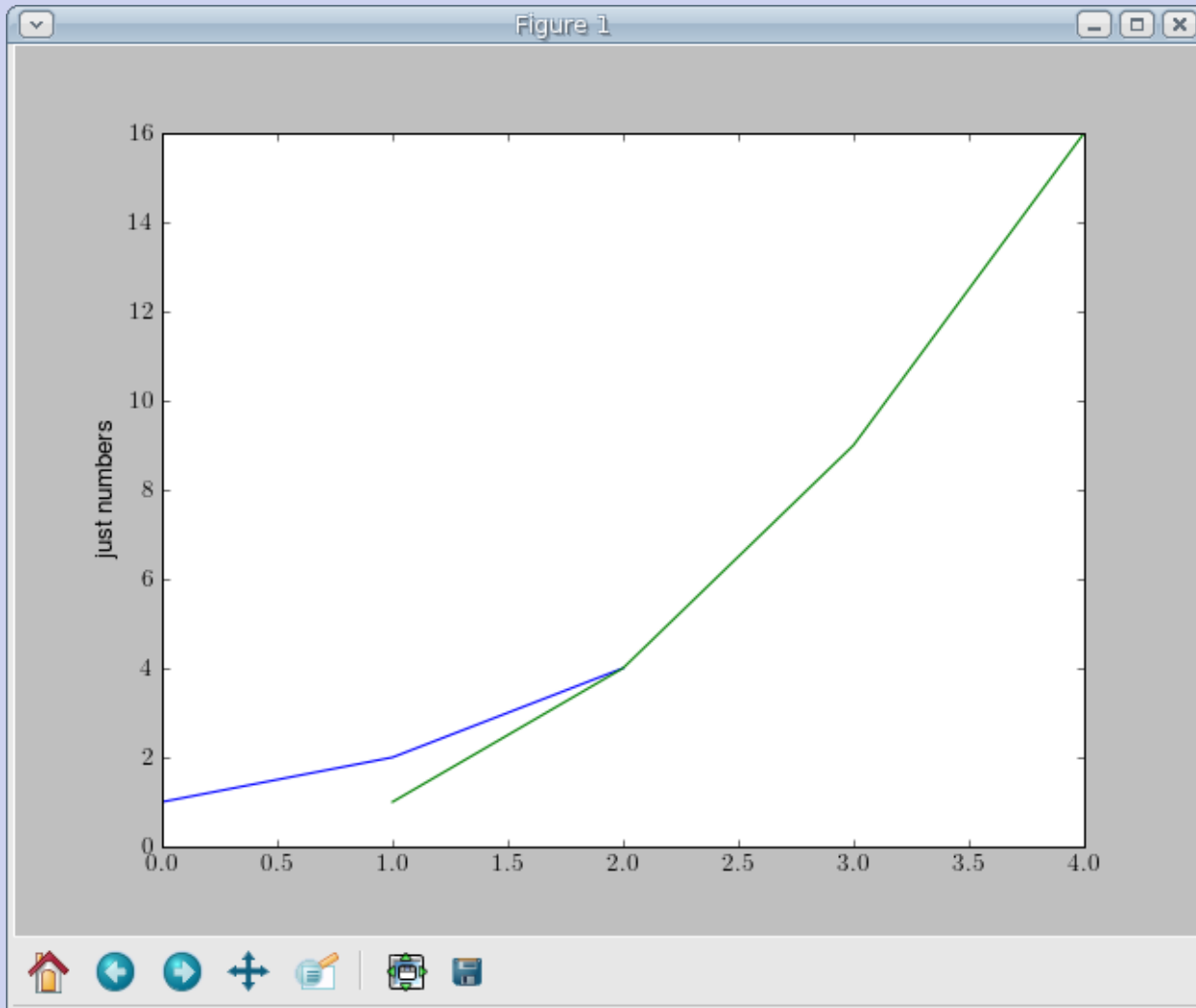
```
>>>
```








```
>>> x = [1,2,3,4]
```

```
>>> y = [1,4,9,16]
```

```
>>> plot(x,y)
```

matplotlib



-    home, forward, back
-  pan & zoom
-  zoom to rect
-  configure subplot
-  save to file

simple plotting

```
>>> clf() # clear current figure
```

```
>>> plot(x, y, 'ro-')
```

↓

Colors: *blue*, *green*, *red*, *cyan*, *magenta*,
yellow, *black*, *white*

↓

Marker symbols: **o** - circle, **^** - triangle up,
< - triangle left, **s** - square, **+** - plus symbol,
. - points, **,** - pixels, rest see docs

↓

Linestyle: - solid, -- dashed, -. dash-dotted,
: dotted

simple plotting

```
>>> t = linspace(0, 2*pi, 10)
>>> s = linspace(-pi, pi, 10)
>>>
>>> plot(t, sin(t), s, cos(s))
>>> plot(t, 0.2+cos(t), 'g^-',
...      s, 0.2+sin(s), 'rs:')
```

line properties

```
>>> t = linspace(0,2*pi,1000)
>>> st = sin(t)
>>>
>>> plot(t, st, color='green')
>>> plot(t, st**2, color='#C000C0')
>>> plot(t, st+1, color=(0.5,0,0,1))
>>> # RGB as hex #C000C0
>>> # RGB(A) as tuple (0.5,0,0,0.5)
>>> # alpha: 0.0 transparent, 1 opaque
```

line properties

```
>>> x = arange(10)
>>> y = x**2
>>> clf()
>>> plot(x, y, color='red', linewidth=2,
...      linestyle=':')
>>>
>>> plot(x, y+1, c='green',
...      lw=3, ls='--')
```

marker

```
>>> plot(x,y, color='r',  
...      marker='s', markerfacecolor='m',  
...      markeredgecolor='g')  
>>>  
>>> # !!! to get unfilled markers:  
>>> plot(x,y+1, color='red',  
...      marker='v',  
...      markerfacecolor='None')
```

lines

```
>>> clf()
>>> lines = plot(x,y, x,y+10)
>>> print lines
>>> lines[0].set_color('red') # mpl API
>>> draw()
>>> l0 = lines[0]
>>> l0.set<Tab>
>>>
>>> setp(lines, linestyle='--')
>>> setp(lines, marker='D')
```

Hipparcos CMD

```
>>> import numpy
>>> filename = '/home/mmetz/hipcmd.dat'
>>> cmd = numpy.loadtxt(filename)
>>>
>>> # recall: 2nd column BT-VT
>>> #           3rd column VT
>>> clf()
>>> plot(cmd[:,1], cmd[:,2], 'k,')
```

Hipparcos CMD

```
>>> xlim(-1, 3)
>>> ylim(26, 8)
>>>
>>> xlabel('BT-VT')
>>> ylabel('VT')
```

Hipparcos CMD

```
>>> # numpy arrays
>>> color = cmd[:,1]
>>> red = (color>0.8)
>>> print red.dtype
>>> blue = ~red
>>>
>>> plot(cmd[red,1], cmd[red,2], 'r,')
>>> plot(cmd[blue,1], cmd[blue,2], 'b,')
```

numpy bool arrays

```
>>> # testing numpy boolean arrays !
>>> lst = []
>>> if lst: print 'ok'
>>>
>>> ba = numpy.array([True, False])
>>> if ba.any(): print 'any'
>>> if ba.all(): print 'all'
```

plotting

```
>>> clf()
```

```
>>>
```

```
>>> t = linspace(0, 2*pi, 1000)
```

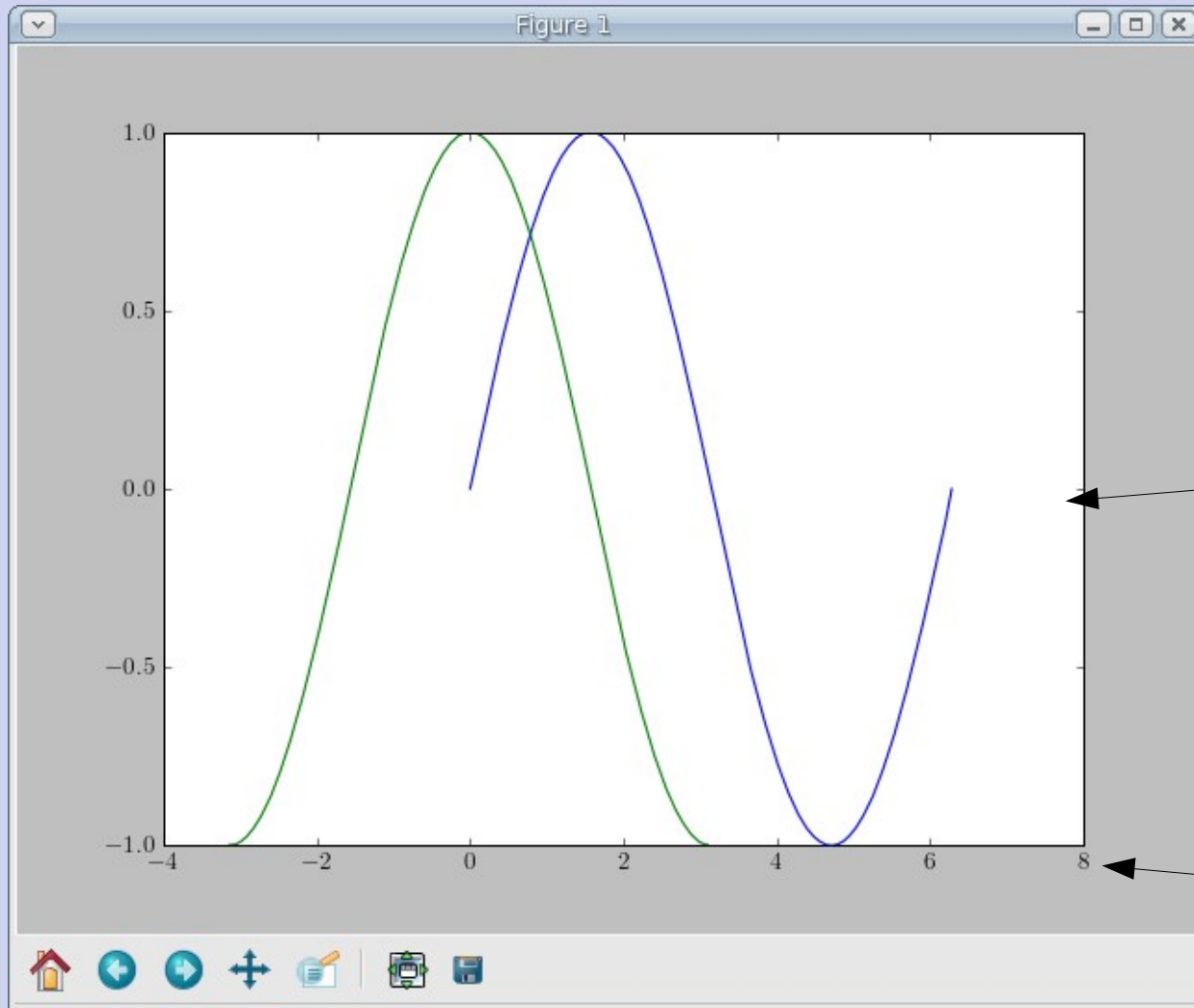
```
>>> s = linspace(-pi, pi, 1000)
```

```
>>>
```

```
>>> plot(t, sin(t))
```

```
>>> plot(s, cos(s))
```

Figure & axes



Figure

Axes

Axis

axes

```
>>> # pylab interface:
```

```
>>> xlim(-pi, 2*pi)
```

```
>>> xlabel('T')
```

```
>>>
```

```
>>> # matplotlib API:
```

```
>>> ax = gca()          # get current axes
```

```
>>> ax.set_xlim(-pi, pi) # lims of xaxis
```

```
>>> ax.set_xlabel('t')
```

```
>>> draw()
```

Figure

```
>>> fig = gcf()      # get current Figure
>>>
>>> print fig.get_figwidth()
>>>     # figure width in inch
>>>     # resize the figure and try again
>>>
>>> fig.clf()
>>> draw()
```

Figures

```
>>> fig1 = figure()
>>> fig2 = figure( figsize=(8,8) )
>>> draw()
>>> nax = fig2.gca()
>>> draw()
>>>
>>> ax = gca()           # pylab
>>> print nax is ax
```

subplots

```
>>> fig2.clf()
>>> draw()
>>>
>>> ax1 = subplot(211) # pylab
>>> # equiv: subplot(2,1,1)
>>> # (nrows, ncols, nplot)
>>>
>>> ax2 = fig2.add_subplot(212)
>>> draw()
```

subplots

```
>>> # pylab
>>> plot(t, sin(t)) # plot to current
>>>                 # axes in current
>>>                 # figure, ie. ax2
>>>
>>> # matplotlib API
>>> ax1.plot(s, sin(s))
>>> draw()
```

subplots

```
>>> # what if I forgot to assign the
>>> # return value of subplot to a var?
>>> fig = gcf()
>>> print fig.axes
>>> ax1 = fig.axes[0]
>>>
>>> # for figures:
>>> f1 = figure(1)
>>> f2 = figure(num=2)
```


Exercise subplots

- create 2x2 subplots and plot x , x^{**2} , x^{**3} and $\text{sqrt}(x)$ for $x=\text{linspace}(0,1,100)$ in the 4 subplots
- experiment with setting x/ylabels for the subplots and axis limits

working with text

```
>>> clf()
>>> plot(t, sin(t))
>>>
>>> xlabel('t')
>>> xlabel('$t$')
>>> ylabel('$\sin(t)$')
>>> title('plot of $\sin(t)$')
>>> subtitle('The figure title')
```

Exercise

- open the configure subplot dialog  and change the values “left” and “top”
- reset view to home

working with text

```
>>> # add text to the plot
>>> text(4, 0.8, 'text in plot')
>>>
>>> v = 3*pi/4
>>> text(v, sin(v), '$s = \sin(t)$')
```

working with text

```
>>> text(pi,0, 'More Text',
...       horizontalalignment='right')
>>> text(pi,0, 'Last Text',
...       verticalalignment='top')
>>>
>>> # also short forms: ha & va; can be
>>> # ha: left, center, right
>>> # va: top, center, bottom, baseline
>>> # defaults are: left, bottom
```

working with text

```
>>> # text(x,y, 'text') places the text
>>> # at x,y in data coords
>>> ax = gca()
>>> text(0.1, 0.9, 'in axes coords',
...      transform=ax.transAxes)
>>> figtext(0.5, 0.5, 'in fig coords')
>>>
>>> # fig/axes coords x,y:
>>> #   0,0 left, bottom
>>> #   1,1 right, top
```

working with text

```
>>> text(1, -0.5, 'Some Text',
...      fontsize=14, color='red',
...      fontweight='bold',
...      style='italic')
>>>
>>> # matplotlib API
>>> txt = text(0.5, -0.5, 'More Text')
>>> txt.set_color('red')
>>> draw()
```

working with text

```
>>> text(1.5, 0.8, 'Cats & Dogs',
...      name='Verdana')
>>>
>>> text(1.5, 0.2, 'Dragons',
...      family='serif')
>>>
>>> # available font families:
>>> #   serif, sans-serif, cursive,
>>> #   fantasy, monospace
```

working with text

```
>>> text(0.5, 0.5, 'Inclinde text',  
...      rotation=30)
```

```
>>>
```

```
>>> text(0.5, 0.5, 'Hurt\nmy eyes',  
...      color = 'red',  
...      backgroundcolor='green')
```

working with text

```
>>> boxprops = {'facecolor':'red',  
...           'alpha':0.8,  
...           'pad':5}  
>>>  
>>> text(2,0.5, 'Text in a box',  
...     bbox=boxprops)
```

Legends

```
>>> clf()
```

```
>>> plot(t, sin(t))
```

```
>>> plot(t, cos(t))
```

```
>>> legend( ('sin', 'cos') )
```

Legends

```
>>> clf()
>>> plot(t, sin(t), label='sin')
>>> plot(t, cos(t), label='$\cos$')
>>> legend()
>>>
>>> # matplotlib API
>>> lines = plot(t, sin(t))
>>> lines[0].set_label('$\sin(t)$')
```

Legends

```
>>> # location code:
```

```
>>> legend(loc=2)
```

```
>>>
```

```
>>> # location string:
```

```
>>> legend(loc='lower left')
```

```
>>>
```

```
>>> # axes coords (not plot coords !):
```

```
>>> legend(loc=(0.5, 0.5))
```

Exercise Text

- Add some text in a plot, once using data coordinates and once using axes coords (`... , transform=ax.transAxes`)
- Change the x/y limits or use the Pan/Zoom Tools to see the different behaviour
- add multiline text ('Hallo\nUniverse') to the plot, and check the behaviour of the parameter `multialignment='right'` ('center', 'left')

log-log plots

```
>>> clf()
```

```
>>> semilogy(t, t**2)
```

```
>>>
```

```
>>> clf()
```

```
>>> semilogx(t, sqrt(t))
```

```
>>>
```

```
>>> clf()
```

```
>>> loglog(t, sqrt(t**3))
```

scatter plots

```
>>> import numpy
>>> r1 = numpy.random.random(10)
>>> r2 = numpy.random.random(10)
>>>
>>> # scatter plot using plot
>>> plot(r1, r2, ls='None', marker='o',
...      markersize=10)
```

scatter plots

```
>>> scatter(r2, r1)
>>> scatter(r2, r1, s=80, c='red',
...         marker='s')
>>>
>>> # plot: markersize linear scale
>>> #         default=6
>>> # scatter: s area scale, default=20
>>> #         ie. diam = sqrt(area)
```

scatter plots

```
>>> cols = ['red', 'blue', 'green']
```

```
>>> scatter(r1+0.1, r2, c=cols)
```

```
>>>
```

```
>>> sizes = linspace(20, 80, 10)
```

```
>>> scatter(r1+0.2, r2, s=sizes)
```

histograms

```
>>> clf()
>>> m, s = 40., 3.
>>> X = m + s * numpy.random.randn(888)
>>> hist(X)
>>>
>>> n, b, p = hist(X)
>>> # numbers, binedges, patches
>>> print n, sum(n)
>>> print b
```

histograms

```
>>> clf()
>>> binedges = linspace(30,50,21)
>>> print binedges
>>>
>>> n, b, p = hist(X, bins=binedges)
>>> print b
>>> print len(n)
>>> print len(b)
```

histograms

```
>>> clf()
```

```
>>>
```

```
>>> n, b, p = \
```

```
...     hist(X, bins=20, range=(30, 50))
```

```
>>> print b
```

```
>>> print len(b)
```

histograms

```
>>> clf()
```

```
>>> hist(X, histtype='stepfilled')
```

```
>>>
```

```
>>> hist(X, histtype='step',
```

```
...     cumulative=True)
```

```
>>>
```

```
>>> clf()
```

```
>>> hist(X, normed=True)
```

Final note

- If you want to use pylab in a script !

```
>>> import pylab
```

```
>>> import numpy
```

```
>>> x = numpy.linspace(0,1,100)
```

```
>>> pylab.plot(x, x**2)
```

```
>>> pylab.show()
```

Exercise

- The files HD152200.uvspec and HD152234.uvspec in /home/mmetz contain FUSE spectra (thanks to Ole !) of the star HD* as ASCII data (wavelength, count rate).
 - i) Read the data, normalise the count rates, and plot both spectra into one subplot.
 - ii) Create a subplot below the first one and plot binned spectra (use numpy to bin the data)