Matplotlib

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Alejandro F. Bujan

Bernstein Center Freiburg

Scientific Python programming course, 12. November 2013

Outline

UNI FREIBURG

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Outline

UNI FREIBURG

Introduction

- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell

- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib

- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

What is matplotlib?

UNI FREIBURG

- matplotlib is a plotting library for the Python programming language and its NumPy numerical mathematics extension.

Outline

UNI FREIBURG

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Matplotlib, pylab, and pyplot: how are they related?

UNI FREIBURG

- `matplotlib` is the whole package
- `pylab` is a matplotlib module that gets installed alongside matplotlib
- `pyplot` is a matplotlib module

Pylab (the MATLAB-style)

UNI FREIBURG

Pylab combines the `pyplot` functionality (for plotting) with the numpy functionality (for mathematics and for working with arrays) in a single namespace, making that namespace (or environment) even more MATLAB-like.

```
from pylab import *
x = arange(0, 10, 0.2)
y = sin(x)
plot(x, y)
show()
```

Pyplot

UNI FREIBURG

Pyplot provides the state-machine interface to the underlying plotting library in matplotlib. This means that figures and axes are implicitly and automatically created to achieve the desired plot.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 10, 0.2)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

Object-oriented programming with Pyplot

UNI FREIBURG

For full control of your plots and more advanced usage, use the pyplot interface for creating figures, and then use the object methods for the rest.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 10, 0.2)
y = np.sin(x)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x, y)
plt.show()
```

Observation

Recommended style!

Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Ipython to the rescue

UNI FREIBURG

Fortunately, ipython, an enhanced interactive python shell, becomes matplotlib aware when you start ipython in the pylab mode.

```
bujan@laptop:~$ ipython -pylab
Python 2.7.3 (default, Apr 10 2013, 06:20:15)
IPython 0.12.1 -- An enhanced Interactive Python.
In [1]: x = randn(10000)
In [2]: hist(x, 100)
```

Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Controlling interactive updating

UNI FREIBURG

The interactive property of the pyplot interface controls whether a figure canvas is drawn on every pyplot command. If interactive is False, then the figure state is updated on every plot command, but will only be drawn on explicit calls to `draw()`. When interactive is True, then every pyplot command triggers a draw.

- `isinteractive()` returns the interactive setting True|False
- `ion()` turns interactive mode on
- `ioff()` turns interactive mode off
- `draw()` forces a figure redraw

Controlling interactive updating

UNI FREIBURG

Interactive example

```
import matplotlib.pyplot as plt
plt.ion()
plt.plot([1.6, 2.7])
plt.title("interactive test")
plt.xlabel("index")
ax = plt.gca()
ax.plot([3.1, 2.2])
plt.draw()
```

Controlling interactive updating

UNI FREIBURG

Non-interactive example

```
import matplotlib.pyplot as plt
plt.ioff()
plt.plot([1.6, 2.7])
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
plt.ioff()
for i in range(3):
    plt.plot(np.random.rand(10))
plt.show()
```

Use `show()`

UNI FREIBURG

When you want to view your plots on your display, the user interface backend will need to start the GUI mainloop. This is what `show()` does.

Observation

Because it is expensive to draw, you typically will not want matplotlib to redraw a figure many times in a script.

Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Matplotlib levels

UNI FREIBURG

There are three layers to the matplotlib API:

- FigureCanvas is the area onto which the figure is drawn.
- Renderer is the object which knows how to draw on the FigureCanvas.
- Artist is the object that knows how to use a renderer to paint onto the canvas.
  - primitives: the standard graphical objects we want to paint onto our canvas (Line2D, Rectangle, Text, ...).
  - containers: the places to put them (Axis, Axes and Figure).

Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

How do I plot?

UNI FREIBURG

The standard use is to create a Figure instance,

```
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure(1,figsize=(8,4))
# figure number, size of the figure in inches
```

use the Figure to create one or more Axes or Subplot instances,

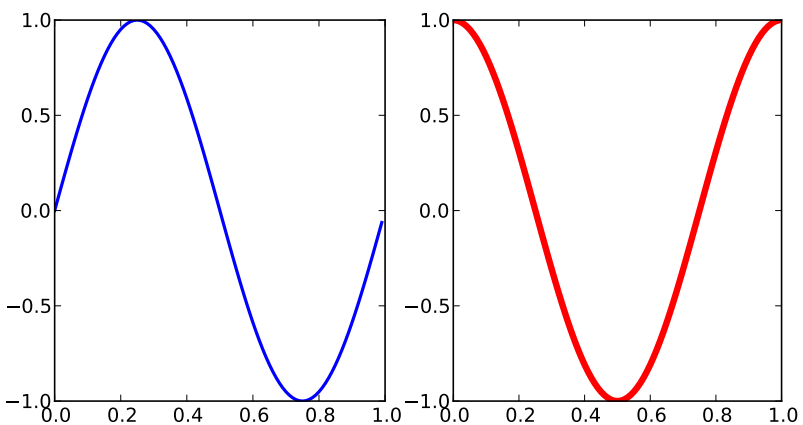
```
ax = fig.add_subplot(1,2,1)
# one row, two columns, first plot
ax2 = fig.add_axes([0.5, 0.1, 0.5, 0.9])
# list of [left, bottom, width, height]
# values in 0-1 relative figure coordinates
```

How do I plot?

UNI FREIBURG

and use the Axes instance helper methods to create the primitives (2D lines in this case).

```
t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2*np.pi*t)
c = np.cos(2*np.pi*t)
line,  = ax.plot(t, s, color='blue', linewidth=2)
line2, = ax2.plot(t, c, color='red', linewidth=4)
plt.show()
```



Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Customizing your objects

UNI FREIBURG

Every element in the figure is represented by a matplotlib Artist, and each has an extensive list of properties to configure its appearance. Each of the properties is accessed with an old-fashioned setter or getter.

```
lw = line.get_linewidth()
line.set_linewidth(2*lw)
```

If you want to set a number of properties at once, you can also use the set method with keyword arguments.

```
line.set(color='green',linewidth=2)
```

Customizing your objects

UNI FREIBURG

If you are working interactively at the python shell, a handy way to inspect the Artist properties is to use the `matplotlib.artist.getp()` function (simply `getp()` in pylab), which lists the properties and their values.

Use the `matplotlib.artist.setp()` command to set multiple properties on a list of lines.

```
lines = plt.plot(x1, y1, x2, y2)
# use keyword args
plt.setp(lines, color='r', linewidth=2.0)
# or MATLAB style string value pairs
plt.setp(lines, 'color', 'r', 'linewidth', 2.0)
```

Outline

Introduction
- What is matplotlib?
- Matplotlib modules: pylab and pyplot

Using matplotlib in a python shell
- Ipython to the rescue
- Controlling interactive updating

How to plot with Matplotlib
- Intro
- But, how do I plot?
- Customizing objects
- Object containers

How to

Object containers

UNI FREIBURG

Now that we know how to inspect and set the properties of a given object we want to configure, we need to now how to get at that object.
As you add subplots and axes to the figure these will be appended to the `Figure.axes`. Similarly when you call `ax.plot`, it creates a Line2D instance and adds it to the `Axes.lines` list.

```
for axes in fig.axes:
    for lines in axes.lines:
        lines.set_color('magenta')
```

Object container

UNI FREIBURG

Figure and Axes contain a Patch (which is a Rectangle instance for Cartesian coordinates and a Circle for polar coordinates). This patch determines the shape, background and border of the plotting region

```
ax = fig.add_subplot(111)
rect = ax.patch # a Rectangle instance
rect.set_facecolor('green')
```

Object container

UNI FREIBURG

Axes contains two important Artist containers: the XAxis and YAxis, which handle the drawing of the ticks and labels. These are stored as instance variables xaxis and yaxis.

```
for label in axes.xaxis.get_ticklabels():
    label.set_color('orange')
```

Note that the Axes contains many helper methods which forward calls on to the Axis instances so you often do not need to work with them directly unless you want to.

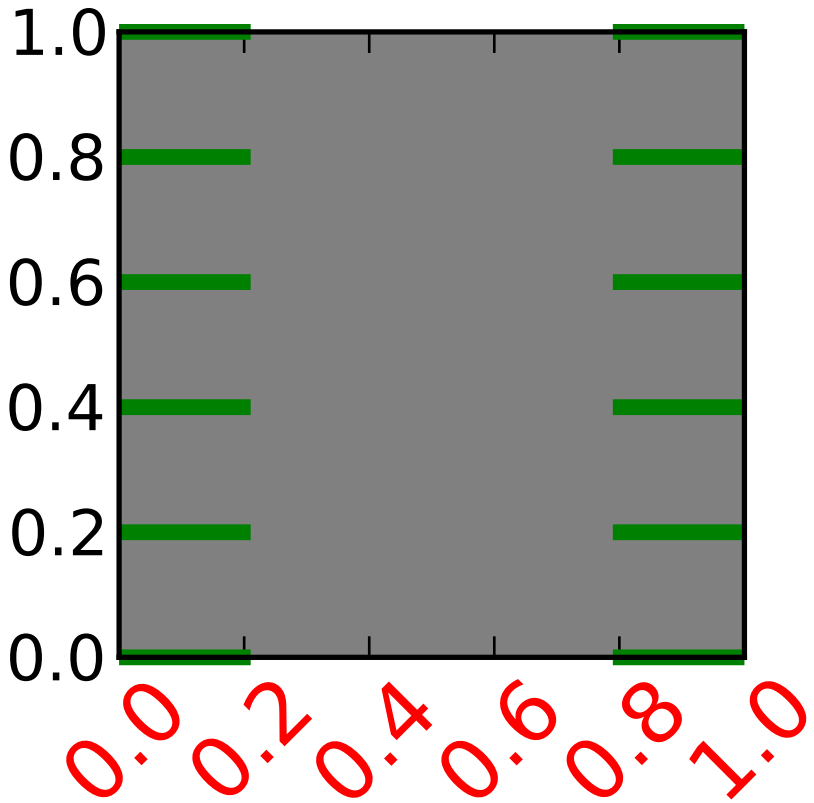
```
for label in axes.get_xticklabels():
    label.set_color('orange')
```

Example

UNI FREIBURG

```
fig = plt.figure(2,figsize=(4,4))
fig.clf()
rect = fig.patch # a rectangle instance
rect.set_facecolor('yellow')
ax1 = fig.add_axes([0.1, 0.5, 0.4, 0.4])
rect = ax1.patch
rect.set_facecolor('gray')
for label in ax1.xaxis.get_ticklabels():
    # label is a Text instance
    label.set_color('red')
    label.set_rotation(45)
    label.set_fontsize(16)
for line in ax1.yaxis.get_ticklines():
    # line is a Line2D instance
    line.set_color('green')
    line.set_markersize(25)
    line.set_markeredgewidth(3)
```

Example



How to cite Matplotlib

UNI FREIBURG

```
@article{Hunter2007,
Author = {Hunter, John D.},
Journal = {Computing In Science \& Engineering},
Month = {May-Jun},
Number = {3},
Pages = {90--95},
Publisher = {IEEE COMPUTER SOC},
Times-Cited = {21},
Title = {Matplotlib: A 2D graphics environment},
Type = {Editorial Material},
Volume = {9},
Year = {2007}}
```

That's all folks!

UNI FREIBURG

Don't forget

Visit the website `http://matplotlib.org/index.html` for more examples!

Bibliography

Hunter, John D. *Matplotlib: A 2D graphics environment*. Computing In Science & Engineering, IEEE COMPUTER SOC, 2007