Nanny tales about Matplotlib

Benjamin Audren

École Polytechnique Fédérale de Lausanne

November 7, 2013

Idea

Motivation

Last week, you saw most of what **Matplotlib** is capable of doing. But when confronted with it alone, one can get lost with its **documentation**, and they way the **examples** are written.

Idea

Motivation

Last week, you saw most of what **Matplotlib** is capable of doing. But when confronted with it alone, one can get lost with its **documentation**, and they way the **examples** are written.

First Goal

To provide you with a **basic understanding** of the main differences between using the **machine state** inside **pyplot** and explicitely accessing the objects through their reference.

Idea

Motivation

Last week, you saw most of what **Matplotlib** is capable of doing. But when confronted with it alone, one can get lost with its **documentation**, and they way the **examples** are written.

First Goal

To provide you with a **basic understanding** of the main differences between using the **machine state** inside **pyplot** and explicitely accessing the objects through their reference.

Second Goal

Ensure that the **vocabulary** is written once and for all, and understood.

Outline

Outline

Foreword on interactive mode

It is not much, really

Simply prevent you from using `plt.show()`

Foreword on module names

Pylab and Pyplot

Matplotlib is the parent module, that contains two submodules.

- **Pyplot** contains simply the plotting tool, with its machine state (later), etc...
- **Pylab** wraps in **Pyplot and Numpy** for convenience.

Launchin ipython with pylab

This does `from matplotlib.pylab import`

Understanding the documentation

`http://matplotlib.org/api/pyplot_summary.html`
You will find here a complete list of all the functions inside pyplot. **There are major differences between them !**, though they are presented at the same level...

On the many ways of using Pyplot

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
```

machine state

```
    plt.plot(np.cos(x))

    plt.show()
```

standard

```
    fig = plt.figure()
    ax = fig.add_subplot()

    ax.plot(np.cos(x))

    plt.show()
```

On the many ways of using Pyplot

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
```

machine state

```
    plt.subplot(121)
    plt.plot(np.cos(x))

    plt.subplot(122)
    plt.plot(np.sin(x))

    plt.show()
```

standard

```
    fig = plt.figure()
    ax1 = fig.add_subplot(121)
    ax2 = fig.add_subplot(122)

    ax1.plot(np.cos(x))
    ax2.plot(np.sin(x))

    plt.show()
```

On the many ways of using Pyplot

machine state

```
    plt.subplot(121)
    plt.plot(np.cos(x))

    plt.subplot(122)
    plt.plot(np.sin(x))

    plt.xlim(0, 10)
    plt.show()
```

standard

```
    fig = plt.figure()
    ax1 = fig.add_subplot(121)
    ax2 = fig.add_subplot(122)

    ax1.plot(np.cos(x))
    ax2.plot(np.sin(x))

    ax2.set_xlim(0, 10)
    plt.show()
```

On the many ways of using Pyplot

machine state

```
    plt.plot(np.cos(x))

    plt.figure()
    plt.plot(np.sin(x))

    plt.show()
```

standard

```
    fig1 = plt.figure()
    ax1 = fig1.add_subplot()

    ax1.plot(np.cos(x))

    fig2 = plt.figure()
    ax2 = fig2.add_subplot()
    ax2.plot(np.sin(x))

    plt.show()
```

How does this state machine work ?

`plt.plot(x)` does different things depending on the context:

- it searches for the current figure (`plt.gcf()`), and create one if absent
- it adds an axes object to this current figure (`plt.gcf().add_subplot(111)`)
- it plots on this object x (`plt.gca().plot(x)`)

General rules

When encountering a script written in a certain way

Go to the documentation, search for the first attribute of this **machine state function** and search inside this object for a function called `set_something()`

General rules

When encountering a script written in a certain way

Go to the documentation, search for the first attribute of this **machine state function** and search inside this object for a function called `set_something()`

Example

`http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.xlim` It tells you about current axes object, go to axes documentation `http://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes.set_xlim` Et voila !

Sometimes, you need better options

Matplotlib module

matplotlibrc file

contains settings used for **all** your figures

Sometimes, you need better options

Matplotlib module

matplotlibrc file

contains settings used for **all** your figures

When changing options in one script

```
import matplotlib as mpl
mpl.rcParams['lines.linewidth'] = 2
mpl.rcParams['lines.color'] = 'r'
mpl.rcParams['text.usetex'] = True
```

Outline

Basic objects

Elementary structures

- figure (called from pyplot `fig = plt.figure()`)
- axes (attached to a figure `ax = fig.add_subplot(111)`
- axis (attached to an axes instance)
- spines (attached to an axes instance)

`http://matplotlib.org/api/spines_api.html`

Artists

What are they ?

Actual guys that go and draw things on the canvas !
There are two types:

- **Primitives**: Line2D, Rectangle, Text
- **Containers**: Axis, Axes, Figure

This is a good read `http://matplotlib.org/users/artists.html`