Basic Plotting

Adam Richards

Galvanize, Inc

Last updated: April 7, 2017

1 matplotlib

2 plotting

3 customizing

4 higher-level

5 references

MPL Objectives

- Understand how figure, subplot, axis work together
- Plot inside and outside of Jupyter notebooks
- Understand MPL fundamentals well enough to learn effectively

---

- Get a feel for customization
- Explore the Seaborn graphics library

matplotlib

The most frequently used plotting package in Python, matplotlib, is written in pure Python and is heavily dependent on NumPy.

- Plots should look great i.e. publication quality
- Text should look great (antialiased, etc.)
- Postscript output for inclusion with TEX documents
- Embeddable in a GUI for application development
- Code should be easy to understand and extend
- Making plots should be easy

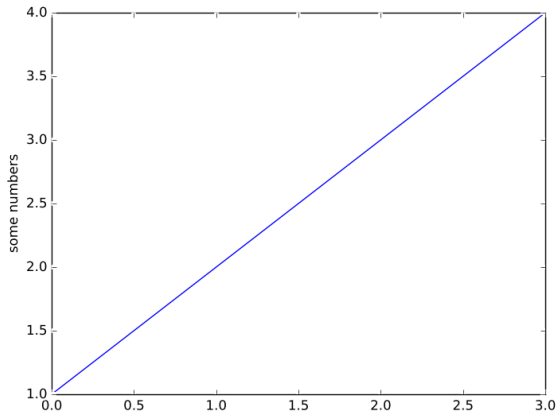
When using in publications or white papers → [1]

Matplotlib is conceptually divided into three parts:

- pylab interface (similar to MATLAB) - pylab tutorial
- Matplotlib frontend or API - artist tutorial
- backends - drawing devices or renderers

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.set_ylabel('some numbers')
ax.plot([1,2,3,4])
plt.show()
```



Note that the x-axis was automatically generated

On Jupyter and data visualization...

Who is in your audience?

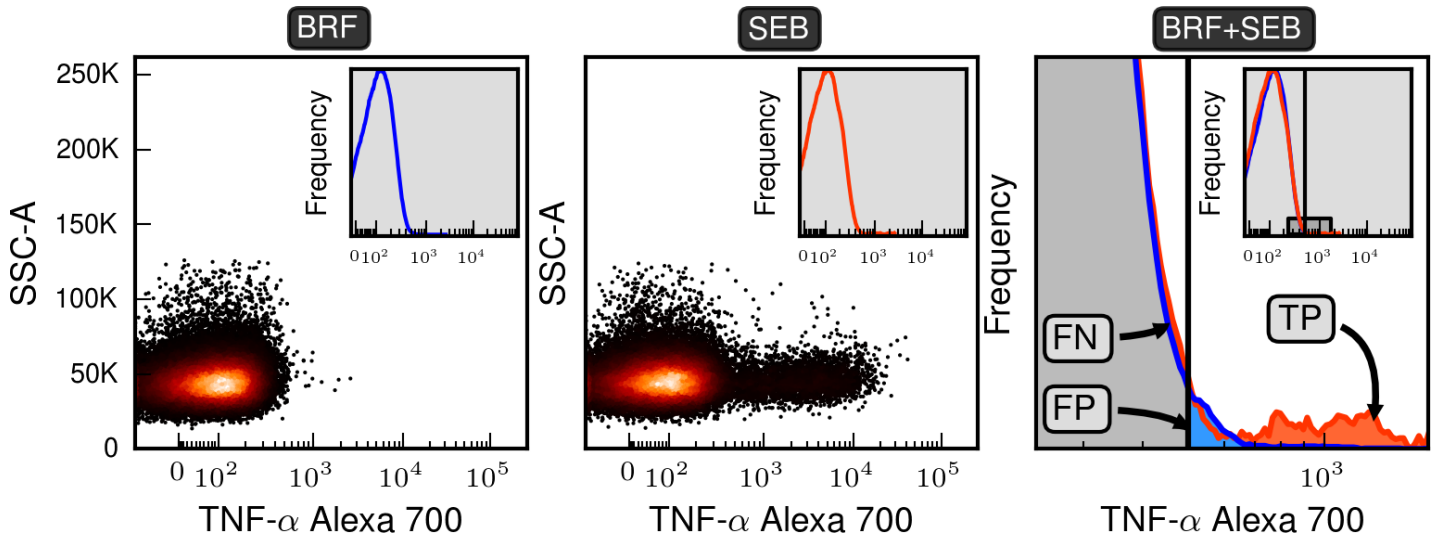
- How much customization do I need?
- How fast does it need to be done?
- Is it complicated?
- Version control, reproducibility

So many choices...

1. Environment - IPython, Jupyter, scripts, Sphinx, reportlab
2. Plotting tool - Pandas, Seaborn, Plotly, Bokeh, MPL-pylab, MPL-artist
3. Deliverable - webpage, report, presentation, white-paper, publication, dashboard

What is it that is I expect to see?

It can get complicated...

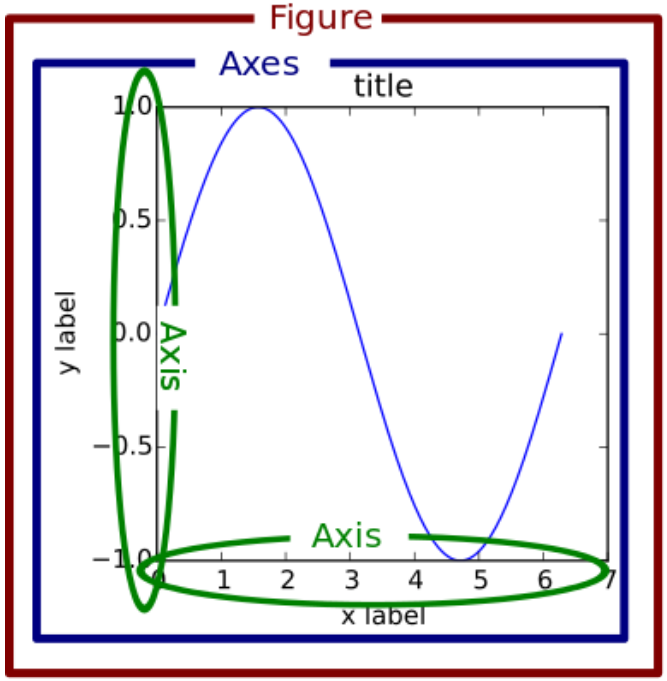


http://www.sciencedirect.com/science/article/pii/S0022175914001185

mpl basics

The shell...

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
ax = plt.add_subplot(1,1,1)
...
ax.set_title('foo')
ax.set_ylabel('y')
ax.set_xlabel('x')
plt.savefig('foo.png',dpi=400)
```



If you are in Jupyter then use `%matplotlib inline`

Most backends support png, pdf, ps, eps and svg.

The supported file formats depend on the selected backend...

Let go through some examples

Backends

```
import matplotlib as mpl
mpl.use('PS')
```

Backend	Description
GTKAgg	Agg rendering to a GTK 2.x canvas (requires PyGTK and pycairo or cairocffi; Python2 only)
GTK3Agg	Agg rendering to a GTK 3.x canvas (requires PyGObject and pycairo or cairocffi)
GTK	GDK rendering to a GTK 2.x canvas (not recommended) (requires PyGTK and pycairo or cairocffi; Python2 only)
GTKCairo	Cairo rendering to a GTK 2.x canvas (requires PyGTK and pycairo or cairocffi; Python2 only)
GTK3Cairo	Cairo rendering to a GTK 3.x canvas (requires PyGObject and pycairo or cairocffi)
WXAgg	Agg rendering to to a wxWidgets canvas (requires wxPython)
WX	Native wxWidgets drawing to a wxWidgets Canvas (not recommended) (requires wxPython)
TkAgg	Agg rendering to a Tk canvas (requires TkInter)
Qt4Agg	Agg rendering to a Qt4 canvas (requires PyQt4 or `pyside`)
Qt5Agg	Agg rendering in a Qt5 canvas (requires PyQt5)
macosx	Cocoa rendering in OSX windows (presently lacks blocking show() behavior when matplotlib is in non-interactive mode)

Start simple and build

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))
ax = fig.add_subplot(111)

ax.set_ylabel('something')
ax.set_title('something')

t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2*np.pi*t)
line, = ax.plot(t, s, color='blue', lw=2)
plt.show()
```

BREAKOUT

Useful plotting functions

command	description
`plot`	plot lines and/or markers
`bar`	bar plot
`error bar`	error bar plot
`boxplot`	boxplot
`histogram`	histogram
`pie`	pie charts
`imshow`	heatmaps/images
`scatter`	scatter plots

The gallery will be your new friend

BREAKOUT

Getting comfortable with axes

Plot 1 (there is an error in there)

```
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(222)
```

Plot 2 (What does this do?)

```
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(212)
```

Plot 3
Hint: `add_axes(left,bottom,width,height)`

```
ax1 = fig.add_subplot(211)
ax2 = fig.add_axes([0.25,0.1,0.5,0.3])
```

BREAKOUT

Objectives

- ✓ Understand how figures, subplots, axes work together in matplotlib
- ✓ Plot inside and outside of Jupyter notebooks
- ✓ Understand MPL fundamentals well enough to learn effectively

---

- Get a feel for MPL customization
- Explore the Seaborn graphics library

Useful customization functions

command	description
`text`	add text to an axis
`table`	embed a table in the axes
`suptitle`	figure title
`ylim/xlim`	get/set the limits of x and y
`imshow`	heatmaps/images
`xticks/yticks`	get/set limits of tick locations
`tight_layout`	tries to make whitespace look right

The axis whitespace buffer does not look right

```
def fix_whitespace(ax,buff=0.01):
    """use x and y to add well spaced margins"""
    xmin,xmax = ax.get_xlim()
    ymin,ymax = ax.get_ylim()
    xbuff = buff * (xmax - xmin)
    ybuff = buff * (ymax - ymin)
    ax.set_xlim(xmin-xbuff,xmax+xbuff)
    ax.set_ylim(ymin-ybuff,ymax+ybuff)
```

Do this after you have plotted the data

More FAQs

How do I change the number of ticks on my axis?

```
from matplotlib.ticker import MaxNLocator

ax.xaxis.set_major_locator(MaxNLocator(5))
ax.yaxis.set_major_locator(MaxNLocator(3))
```

How do I force the aspect ratio to be square?

```
for ax in [ax1,ax2,ax3]:
    ax.set_aspect(1./ax.get_data_ratio())
```

Can I remove the axis and ticks to plot a legend or table?

```
ax.set_yticks([])
ax.set_xticks([])
ax.set_frame_on(False)
```

More FAQs

How do I change the font size and font type for my tick labels?

```
font_size = 11
font_name = 'sans-serif'
for t in ax.get_xticklabels():
    t.set_fontsize(font_size-1)
    t.set_fontname(font_name)
for t in ax.get_yticklabels():
    t.set_fontsize(font_size-1)
    t.set_fontname(font_name)
```

What about legends?

```
leg = ax.legend(handles,lables)
ltext = leg.get_texts()
for i in range(len(ltext)):
    plt.setp(ltext[i],fontsize=fontSize)
```

style_sheets

```
with plt.style.context('fivethirtyeight'):
    plt.plot(x, np.sin(x) + x + np.random.randn(50))
    plt.plot(x, np.sin(x) + 0.5 * x + np.random.randn(50))
    plt.plot(x, np.sin(x) + 2 * x + np.random.randn(50))
```

or

```
plt.style.use('dark_background')
```

But be careful with the latter in Jupyter!

Higher level interfaces

- seaborn
- holoviews
- ggplot

Where do we go from here?

- 3D plots in MPL
- Interactive widgets in MPL
- Embedded plots
- Image analysis - PIL
- Mayavi - 3D data viz

Useful links

- list of plotting commands
- matplotlib howtos
- pylab tutorial
- artist tutorial
- FAQs

References I

J.D. Hunter, *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering **9(3)** (2007), 90–95.