

Saving Figures to File

One nice feature of Matplotlib is the ability to save figures in a wide variety of formats. You can save a figure using the `savefig()` command. For example, to save the previous figure as a PNG file, you can run this:

```
In[5]: fig.savefig('my_figure.png')
```

We now have a file called `my_figure.png` in the current working directory:

```
In[6]: !ls -lh my_figure.png  
-rw-r--r--  1 jakevdp  staff   16K Aug 11 10:59 my_figure.png
```

To confirm that it contains what we think it contains, let's use the IPython Image object to display the contents of this file (Figure 4-2):

```
In[7]: from IPython.display import Image  
Image('my_figure.png')
```

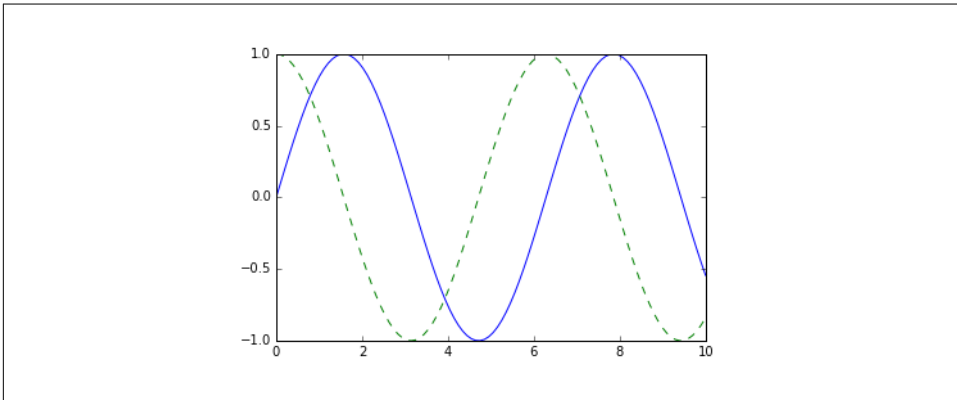


Figure 4-2. PNG rendering of the basic plot

In `savefig()`, the file format is inferred from the extension of the given filename. Depending on what backends you have installed, many different file formats are available. You can find the list of supported file types for your system by using the following method of the figure canvas object:

```
In[8]: fig.canvas.get_supported_filetypes()  
Out[8]: {'eps': 'Encapsulated Postscript',  
         'jpeg': 'Joint Photographic Experts Group',  
         'jpg': 'Joint Photographic Experts Group',  
         'pdf': 'Portable Document Format',  
         'pgf': 'PGF code for LaTeX',  
         'png': 'Portable Network Graphics',  
         'ps': 'Postscript',  
         'raw': 'Raw RGBA bitmap',  
         'rgba': 'Raw RGBA bitmap',
```

```
'svg': 'Scalable Vector Graphics',
'svgz': 'Scalable Vector Graphics',
'tif': 'Tagged Image File Format',
'tiff': 'Tagged Image File Format'}
```

Note that when saving your figure, it's not necessary to use `plt.show()` or related commands discussed earlier.

Two Interfaces for the Price of One

A potentially confusing feature of Matplotlib is its dual interfaces: a convenient MATLAB-style state-based interface, and a more powerful object-oriented interface. We'll quickly highlight the differences between the two here.

MATLAB-style interface

Matplotlib was originally written as a Python alternative for MATLAB users, and much of its syntax reflects that fact. The MATLAB-style tools are contained in the pyplot (`plt`) interface. For example, the following code will probably look quite familiar to MATLAB users (Figure 4-3):

```
In[9]: plt.figure() # create a plot figure

# create the first of two panels and set current axis
plt.subplot(2, 1, 1) # (rows, columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and set current axis
plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x));
```

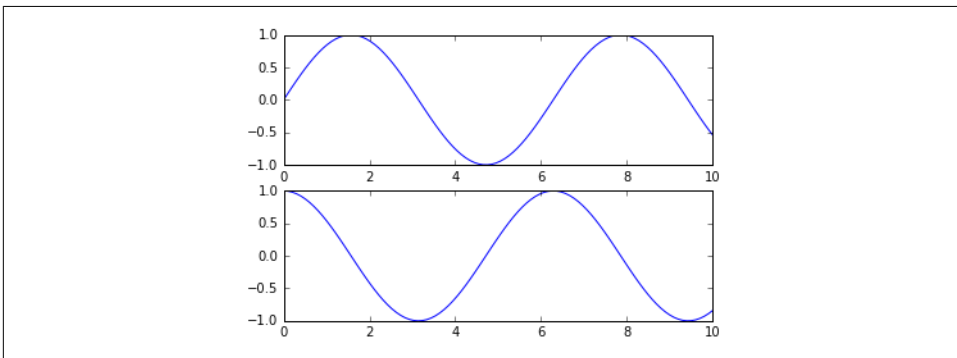


Figure 4-3. Subplots using the MATLAB-style interface

It's important to note that this interface is *stateful*: it keeps track of the “current” figure and axes, which are where all `plt` commands are applied. You can get a reference to