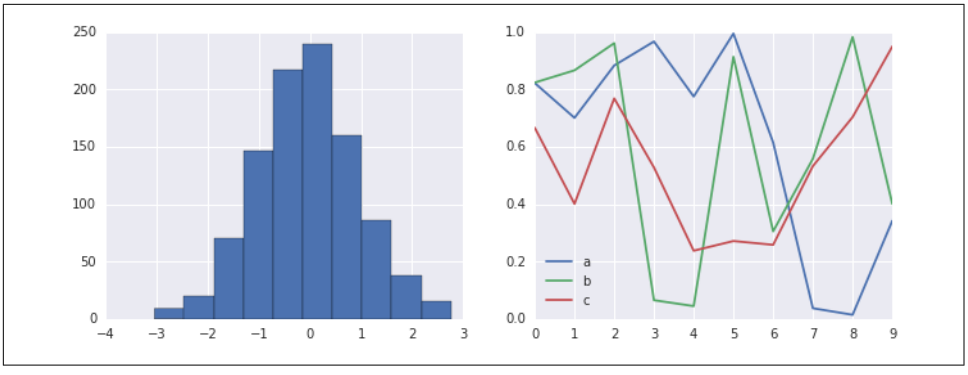

Figure 4-91. Seaborn's plotting style

With all of these built-in options for various plot styles, Matplotlib becomes much more useful for both interactive visualization and creation of figures for publication. Throughout this book, I will generally use one or more of these style conventions when creating plots.

Three-Dimensional Plotting in Matplotlib

Matplotlib was initially designed with only two-dimensional plotting in mind. Around the time of the 1.0 release, some three-dimensional plotting utilities were built on top of Matplotlib's two-dimensional display, and the result is a convenient (if somewhat limited) set of tools for three-dimensional data visualization. We enable three-dimensional plots by importing the `mplot3d` toolkit, included with the main Matplotlib installation (Figure 4-92):

```
In[1]: from mpl_toolkits import mplot3d
```

Once this submodule is imported, we can create a three-dimensional axes by passing the keyword `projection='3d'` to any of the normal axes creation routines:

```
In[2]: %matplotlib inline
       import numpy as np
       import matplotlib.pyplot as plt

In[3]: fig = plt.figure()
       ax = plt.axes(projection='3d')
```

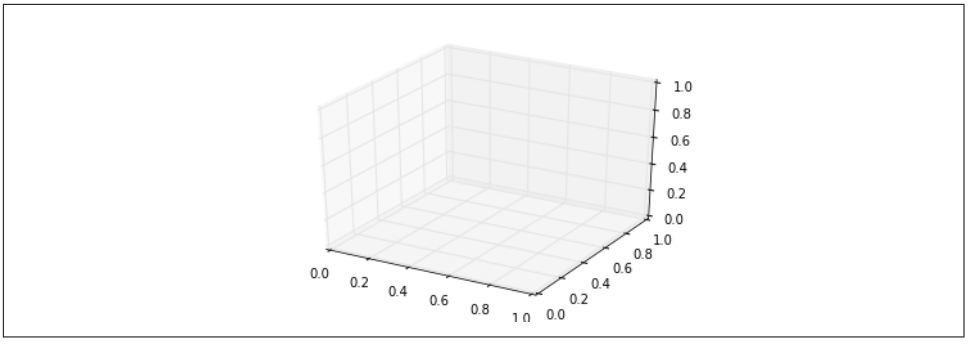

Figure 4-92. An empty three-dimensional axes

With this 3D axes enabled, we can now plot a variety of three-dimensional plot types. Three-dimensional plotting is one of the functionalities that benefits immensely from viewing figures interactively rather than statically in the notebook; recall that to use interactive figures, you can use `%matplotlib notebook` rather than `%matplotlib inline` when running this code.

Three-Dimensional Points and Lines

The most basic three-dimensional plot is a line or scatter plot created from sets of (x, y, z) triples. In analogy with the more common two-dimensional plots discussed earlier, we can create these using the `ax.plot3D` and `ax.scatter3D` functions. The call signature for these is nearly identical to that of their two-dimensional counterparts, so you can refer to "Simple Line Plots" on page 224 and "Simple Scatter Plots" on page 233 for more information on controlling the output. Here we'll plot a trigonometric spiral, along with some points drawn randomly near the line (Figure 4-93):

```
In[4]: ax = plt.axes(projection='3d')

       # Data for a three-dimensional line
       zline = np.linspace(0, 15, 1000)
       xline = np.sin(zline)
       yline = np.cos(zline)
       ax.plot3D(xline, yline, zline, 'gray')

       # Data for three-dimensional scattered points
       zdata = 15 * np.random.random(100)
       xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
       ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
       ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```