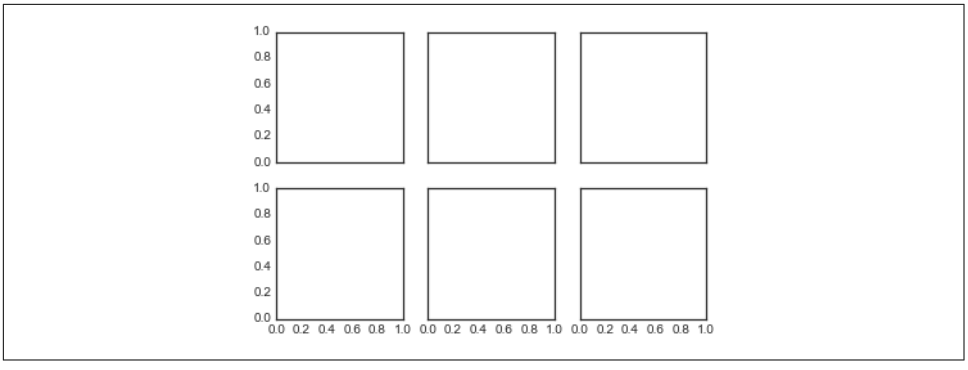

Figure 4-63. Shared x and y axis in plt.subplots()

Note that by specifying `sharex` and `sharey`, we've automatically removed inner labels on the grid to make the plot cleaner. The resulting grid of axes instances is returned within a NumPy array, allowing for convenient specification of the desired axes using standard array indexing notation (Figure 4-64):

```
In[7]: # axes are in a two-dimensional array, indexed by [row, col]
       for i in range(2):
           for j in range(3):
               ax[i, j].text(0.5, 0.5, str((i, j)),
                             fontsize=18, ha='center')
       fig
```

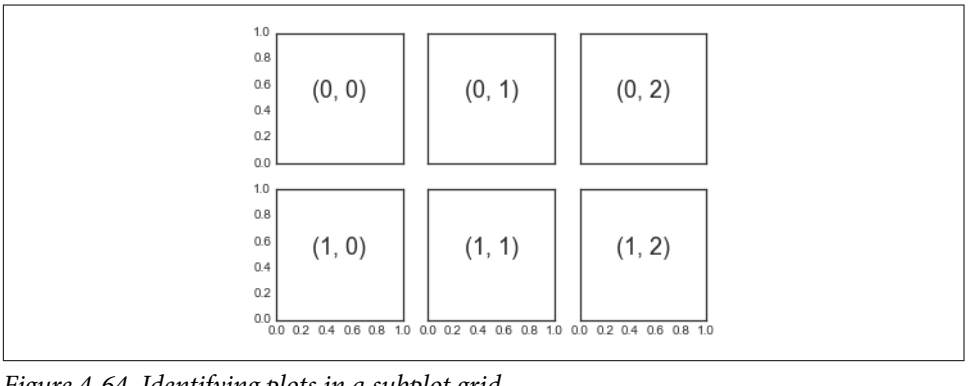

Figure 4-64. Identifying plots in a subplot grid

In comparison to `plt.subplot()`, `plt.subplots()` is more consistent with Python's conventional 0-based indexing.

plt.GridSpec: More Complicated Arrangements

To go beyond a regular grid to subplots that span multiple rows and columns, `plt.GridSpec()` is the best tool. The `plt.GridSpec()` object does not create a plot by

itself; it is simply a convenient interface that is recognized by the `plt.subplot()` command. For example, a gridspec for a grid of two rows and three columns with some specified width and height space looks like this:

```
In[8]: grid = plt.GridSpec(2, 3, wspace=0.4, hspace=0.3)
```

From this we can specify subplot locations and extents using the familiar Python slicing syntax (Figure 4-65):

```
In[9]: plt.subplot(grid[0, 0])
       plt.subplot(grid[0, 1:])
       plt.subplot(grid[1, :2])
       plt.subplot(grid[1, 2]);
```

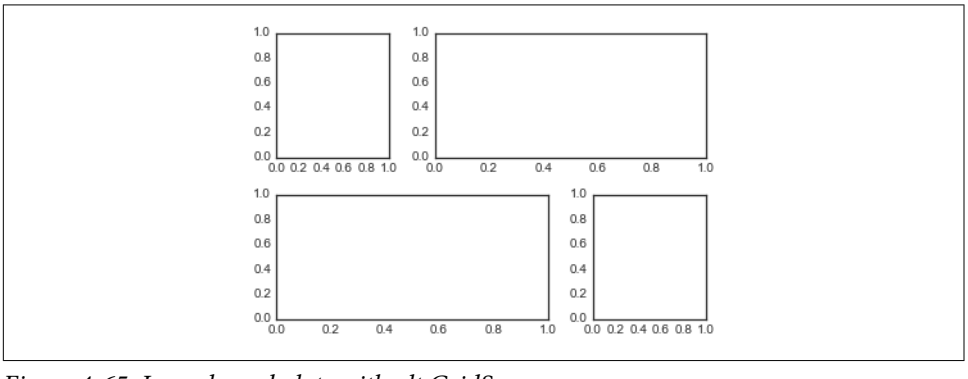

Figure 4-65. Irregular subplots with plt.GridSpec

This type of flexible grid alignment has a wide range of uses. I most often use it when creating multi-axes histogram plots like the one shown here (Figure 4-66):

```
In[10]: # Create some normally distributed data
        mean = [0, 0]
        cov = [[1, 1], [1, 2]]
        x, y = np.random.multivariate_normal(mean, cov, 3000).T

        # Set up the axes with gridspec
        fig = plt.figure(figsize=(6, 6))
        grid = plt.GridSpec(4, 4, hspace=0.2, wspace=0.2)
        main_ax = fig.add_subplot(grid[:-1, 1:])
        y_hist = fig.add_subplot(grid[:-1, 0], xticklabels=[], sharey=main_ax)
        x_hist = fig.add_subplot(grid[-1, 1:], yticklabels=[], sharex=main_ax)

        # scatter points on the main axes
        main_ax.plot(x, y, 'ok', markersize=3, alpha=0.2)

        # histogram on the attached axes
        x_hist.hist(x, 40, histtype='stepfilled',
                    orientation='vertical', color='gray')
        x_hist.invert_yaxis()
```

```
y_hist.hist(y, 40, histtype='stepfilled',
            orientation='horizontal', color='gray')
y_hist.invert_xaxis()
```

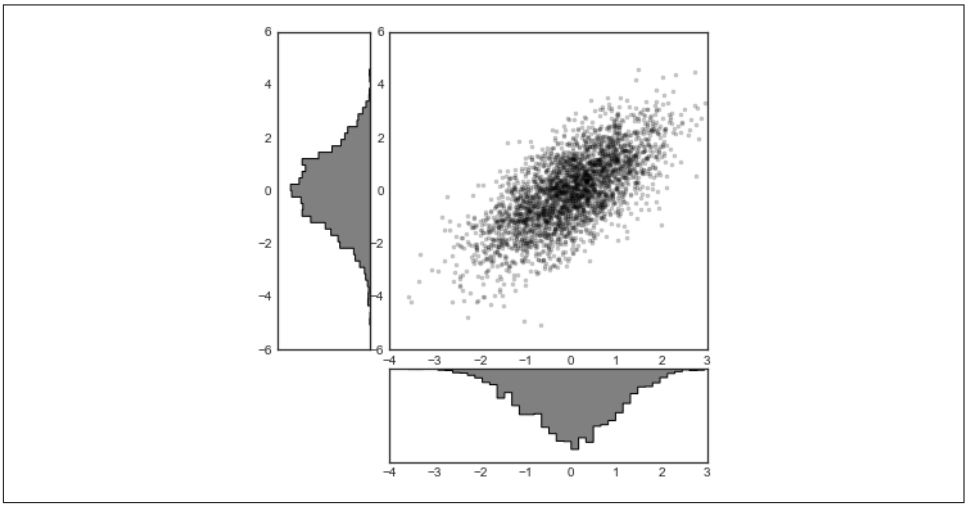

Figure 4-66. Visualizing multidimensional distributions with plt.GridSpec

This type of distribution plotted alongside its margins is common enough that it has its own plotting API in the Seaborn package; see "Visualization with Seaborn" on page 311 for more details.

Text and Annotation

Creating a good visualization involves guiding the reader so that the figure tells a story. In some cases, this story can be told in an entirely visual manner, without the need for added text, but in others, small textual cues and labels are necessary. Perhaps the most basic types of annotations you will use are axes labels and titles, but the options go beyond this. Let's take a look at some data and how we might visualize and annotate it to help convey interesting information. We'll start by setting up the notebook for plotting and importing the functions we will use:

```
In[1]: %matplotlib inline
       import matplotlib.pyplot as plt
       import matplotlib as mpl
       plt.style.use('seaborn-whitegrid')
       import numpy as np
       import pandas as pd
```