Topic for the class-Multiple subplots

Unit _2 : Title-Digital data – an Imprint

Date & Time : 22.8.24 10.00 AM – 10.50 AM

Dr. Bhramaramba Ravi

Professor

Department of Computer Science and Engineering

GITAM School of Technology (GST)

Visakhapatnam – 530045

Email: **bravi@gitam.edu**

Unit2-syllabus

- **UNIT 2 Digital Data-An Imprint 9 hours, P - 2 hours** Type of data analytics (Descriptive, diagnostic, perspective, predictive, Prescriptive.) Exploratory Data Analysis (EDA), EDA-Quantitative Technique, EDA - Graphical Technique. Data Types for Plotting, Data Types and Plotting, Simple Line Plots, Simple Scatter Plots, Visualizing Errors, Density and Contour Plots, Histograms, Binnings, and Density, Customizing Plot Legends, Customizing Color bars, Multiple Subplots, Text and Annotation, Customizing Ticks.
- https://www.coursera.org/learn/data-visualization-r

Multiple subplots

- Sometimes it is helpful to compare different views of data side by side. To this end, Matplotlib has the concept of *subplots*: groups of smaller axes that can exist together within a single figure.
- These subplots might be insets, grids of plots, or other more complicated layouts.
- In this section, we'll explore four routines for creating subplots
- in Matplotlib. 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**
- plt.style.use('seaborn-white')
- **import numpy as np**

Multiple subplots

- **plt.axes: Subplots by Hand**
- The most basic method of creating an axes is to use the plt.axes function. As we've
- seen previously, by default this creates a standard axes object that fills the entire figure.
- plt.axes also takes an optional argument that is a list of four numbers in the
- figure coordinate system. These numbers represent [*bottom*, *left*, *width*,
- *height*] in the figure coordinate system, which ranges from 0 at the bottom left of the
- figure to 1 at the top right of the figure.
- For example, we might create an inset axes at the top-right corner of another axes by
- setting the *x* and *y* position to 0.65 (that is, starting at 65% of the width and 65% of
- the height of the figure) and the *x* and *y* extents to 0.2 (that is, the size of the axes is
- 20% of the width and 20% of the height of the figure). Figure 4-59 shows the result of
- this code:
- In[2]: ax1 = plt.axes() *# standard axes*
- ax2 = plt.axes([0.65, 0.65, 0.2, 0.2])

Multiple subplots

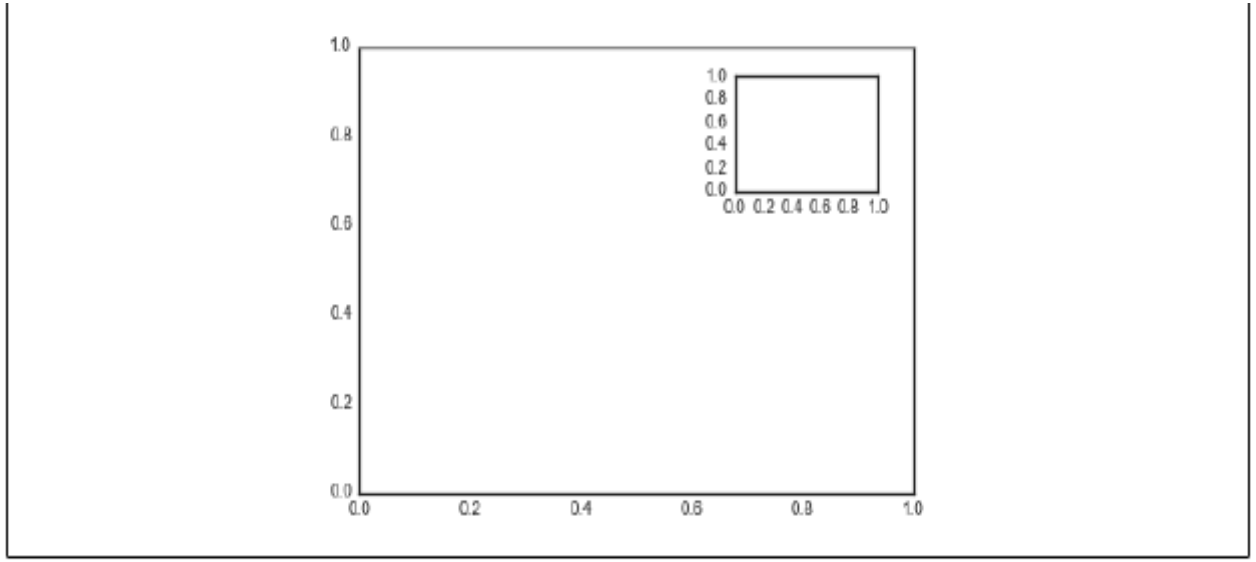

Figure 4-59. Example of an inset axes

Multiple subplots

- The equivalent of this command within the object-oriented interface is fig.add_axes(). Let's use this to create two vertically stacked axes (Figure 4-60):
- In[3]: fig = plt.figure()
- ax1 = fig.add_axes([0.1, 0.5, 0.8, 0.4],
- xticklabels=[], ylim=(-1.2, 1.2))
- ax2 = fig.add_axes([0.1, 0.1, 0.8, 0.4],
- ylim=(-1.2, 1.2))
- x = np.linspace(0, 10)
- ax1.plot(np.sin(x))
- ax2.plot(np.cos(x));

Multiple subplots

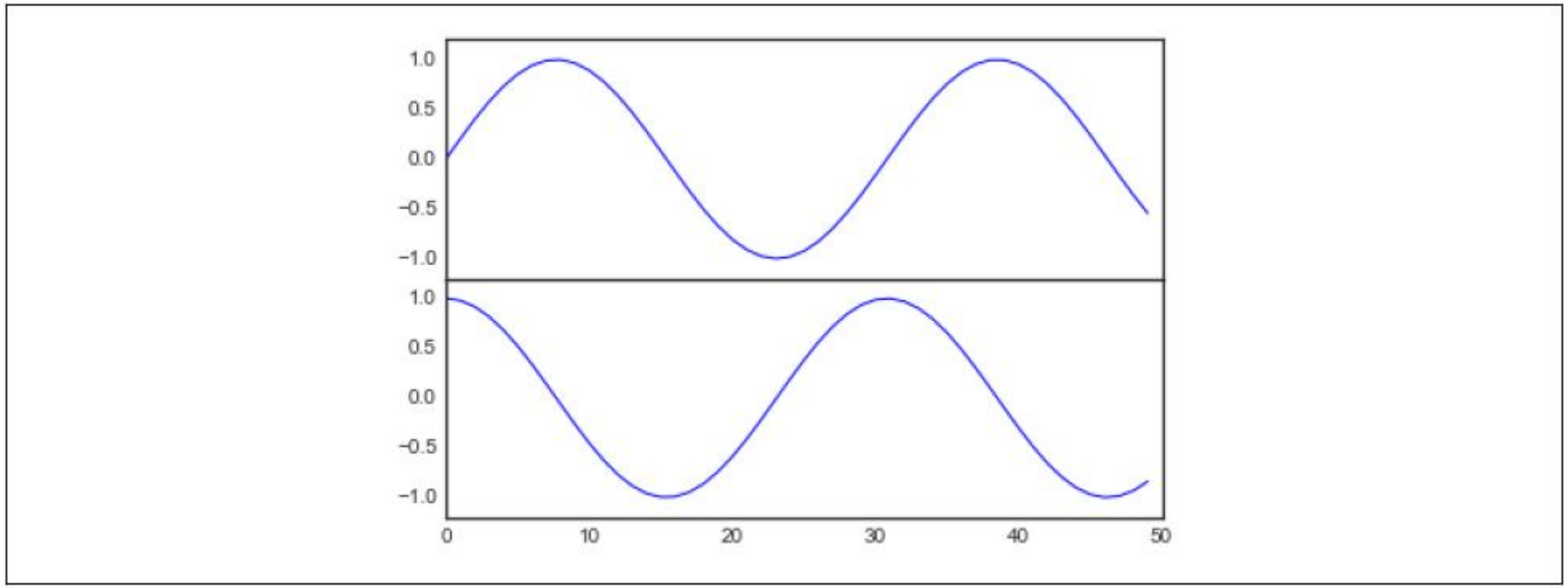

Figure 4-60. Vertically stacked axes example

Multiple subplots

- **plt.subplot: Simple Grids of Subplots**
- Aligned columns or rows of subplots are a common enough need that Matplotlib has several convenience routines that make them easy to create.
- The lowest level of these is plt.subplot(), which creates a single subplot within a grid.
- As you can see, this command takes three integer arguments—the number of rows, the number of columns,
- and the index of the plot to be created in this scheme, which runs from the upper left to the bottom right (Figure 4-61):
- In[4]: **for** i **in** range(1, 7):
- plt.subplot(2, 3, i)
- plt.text(0.5, 0.5, str((2, 3, i)),
- fontsize=18, ha='center')

Multiple subplots

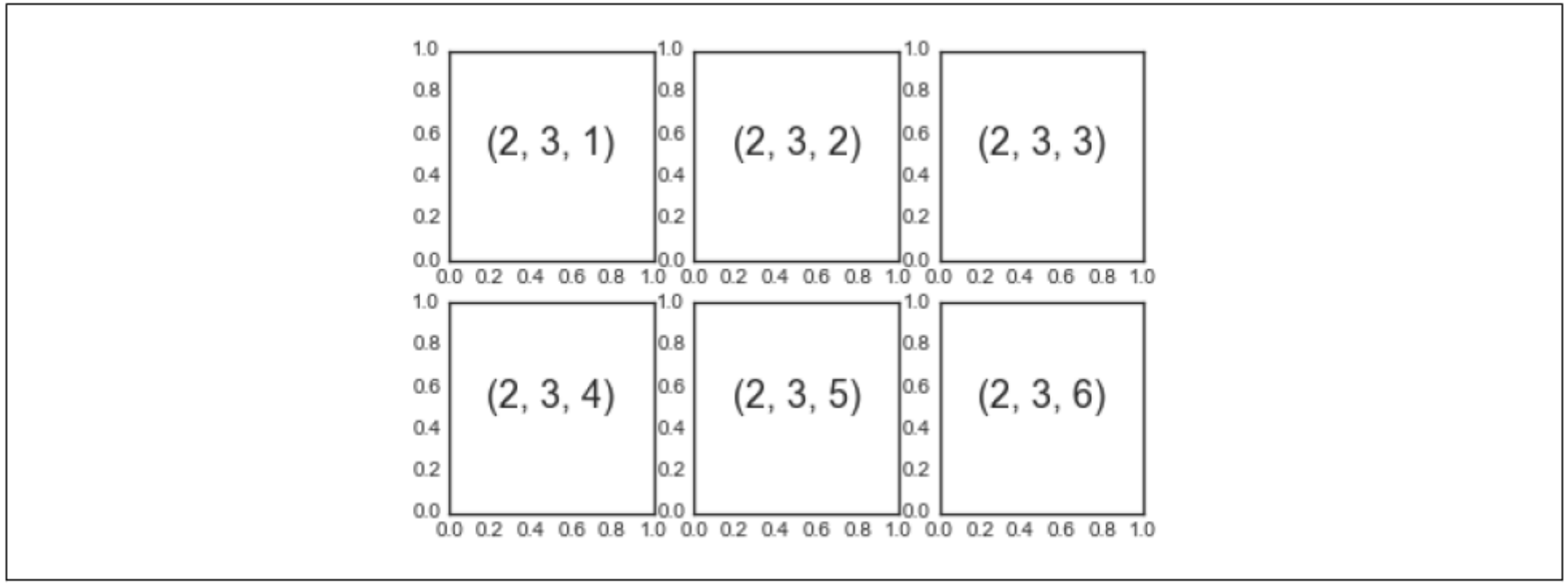

Figure 4-61. A plt.subplot() example

Multiple subplots

- **plt.subplots: The Whole Grid in One Go**
- It can become quite tedious when you're creating a large grid of subplots, especially if you'd like to hide the x- and y-axis labels on the inner plots.
- For this purpose, plt.subplots() is the easier tool to use (note the s at the end of subplots).
- Rather than creating a single subplot, this function creates a full grid of subplots in a single line, returning them in a NumPy array.
- The arguments are the number of rows and number of columns, along with optional keywords sharex and
- sharey, which allow you to specify the relationships between different axes.
- Here we'll create a 2×3 grid of subplots, where all axes in the same row share their y-axis scale, and all axes in the same column share their x-axis scale (Figure 4-63):
- In[6]: fig, ax = plt.subplots(2, 3, sharex='col', sharey='row')

Multiple subplots

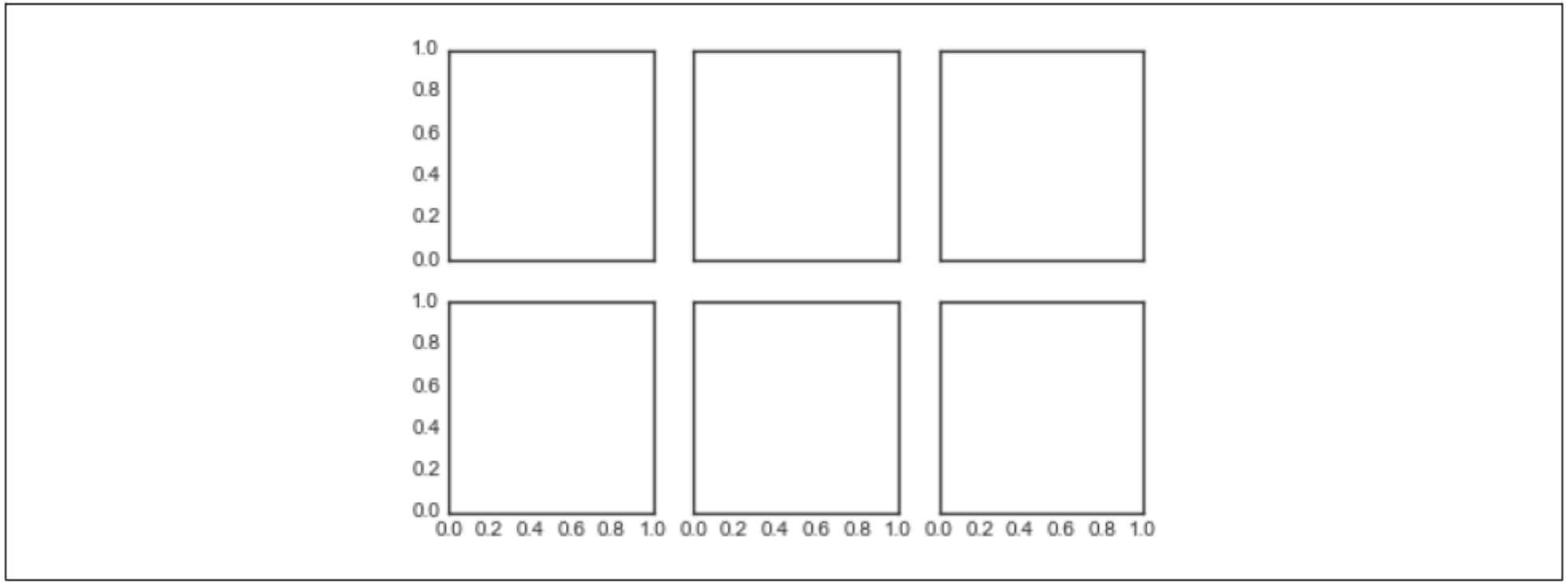

Figure 4-63. Shared x and y axis in plt.subplots()

Multiple 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')

Multiple subplots

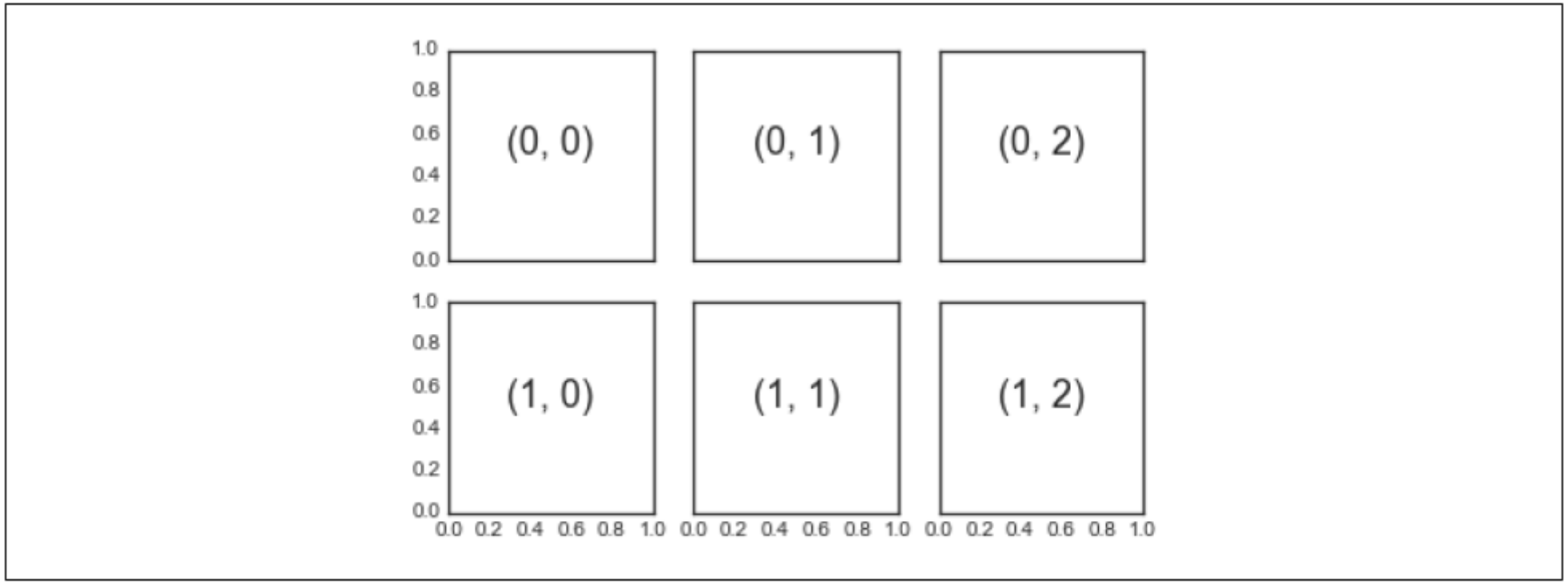

Figure 4-64. Identifying plots in a subplot grid

Multiple subplots

- 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]);

Multiple subplots

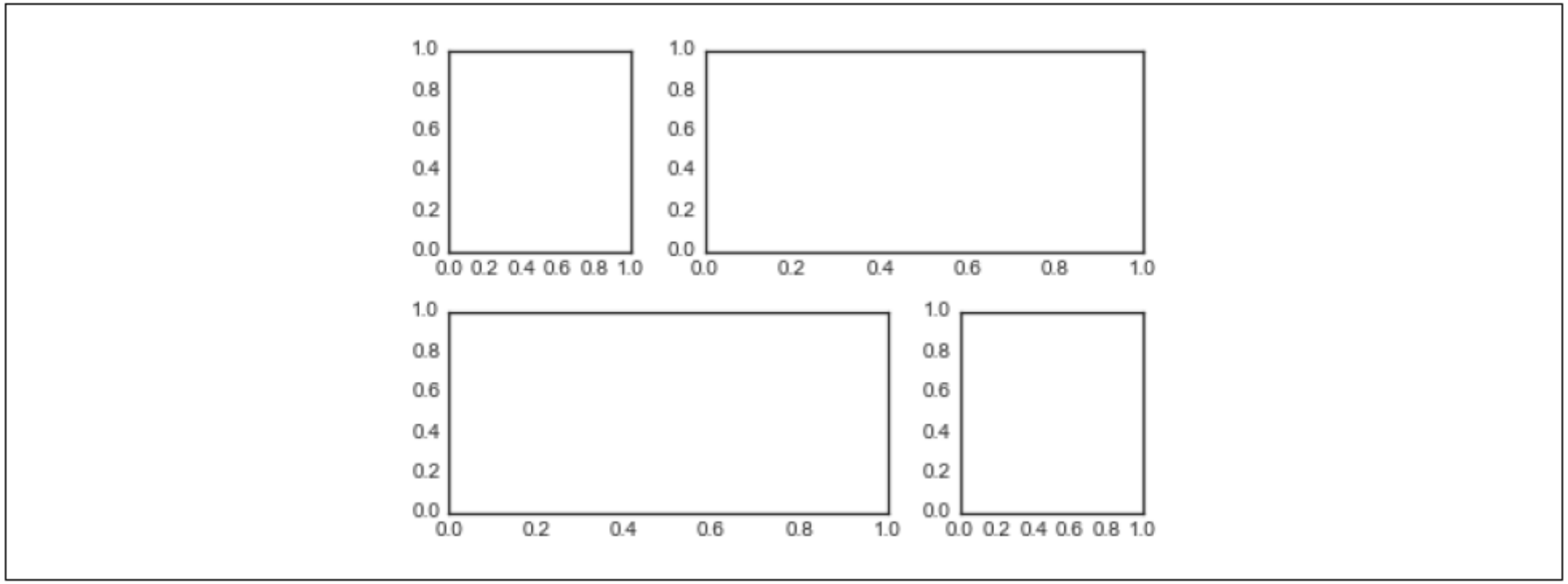

Figure 4-65. Irregular subplots with plt.GridSpec

Multiple subplots

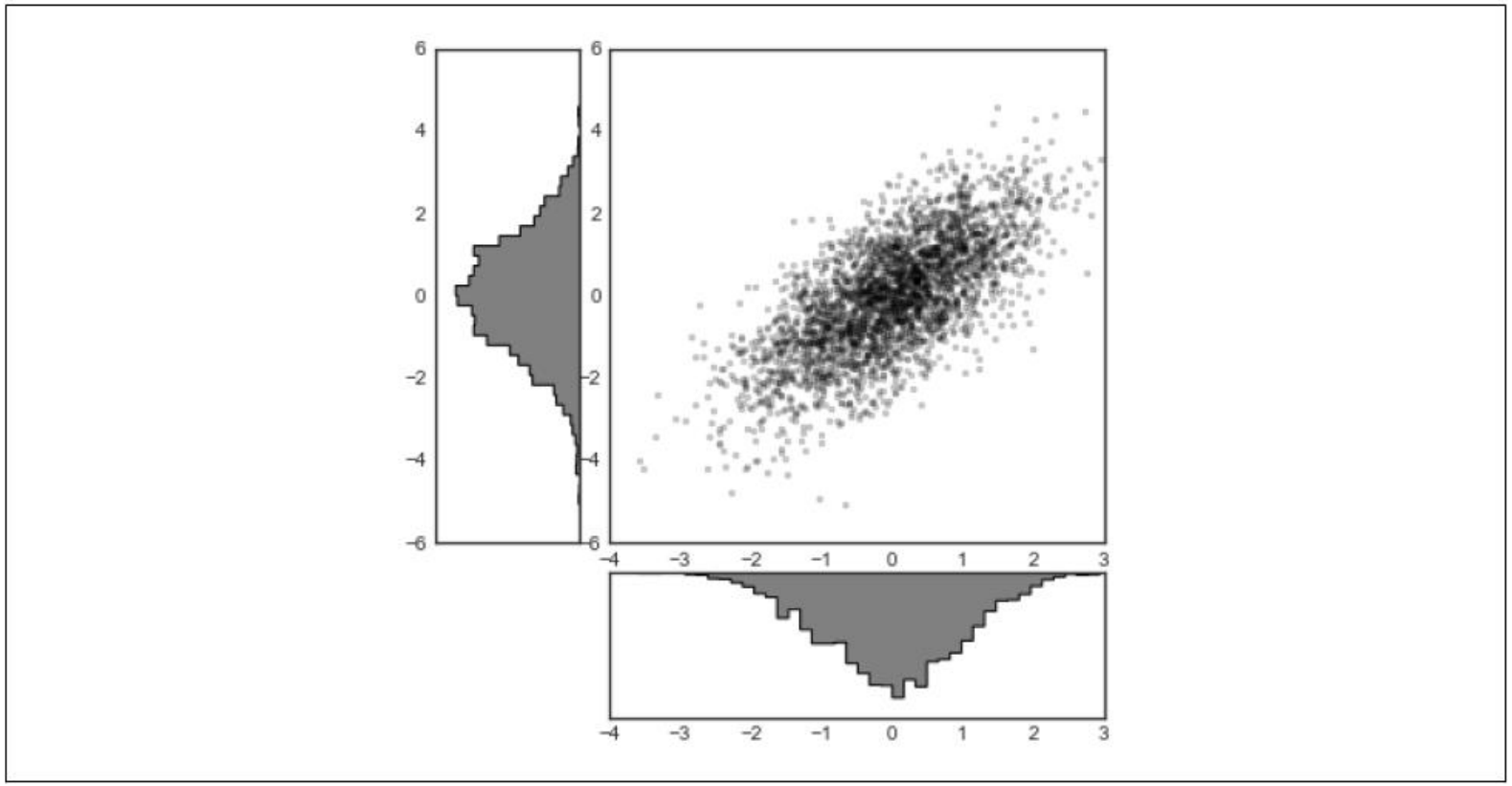

Figure 4-66. Visualizing multidimensional distributions with plt.GridSpec

Multiple subplots

- This type of distribution plotted alongside its margins is common enough that it has its own plotting API in the Seaborn package

THANK YOU