Topic for the class-Customizing ticks

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

Customizing ticks

- Matplotlib's default tick locators and formatters are designed to be generally sufficient in many common situations, but are in no way optimal for every plot.
- This section will give several examples of adjusting the tick locations and formatting for the particular plot type you're interested in.
- Before we go into examples, it will be best for us to understand further the object hierarchy of Matplotlib plots.
- Matplotlib aims to have a Python object representing everything that appears on the plot: for example, recall that the figure is the bounding box within which plot elements appear.
- Each Matplotlib object can also act as a container of sub-objects; for example, each figure can contain one or more axes objects, each of which in turn contain other objects representing plot contents.
- The tick marks are no exception.
- Each axes has attributes xaxis and yaxis, which in turn have attributes that contain all the properties of the lines, ticks, and labels that make up the axes.

Customizing ticks

- Within each axis, there is the concept of a *major* tick mark and a *minor* tick mark.
- As the names would imply, major ticks are usually bigger or more pronounced, while minor ticks are usually smaller.
- By default, Matplotlib rarely makes use of minor ticks, but one place you can see them is within logarithmic plots (Figure 4-73):
- In[1]: %matplotlib inline
- **import matplotlib.pyplot as plt**
- plt.style.use('seaborn-whitegrid')
- **import numpy as np**
- In[2]: ax = plt.axes(xscale='log', yscale='log')

Customizing ticks

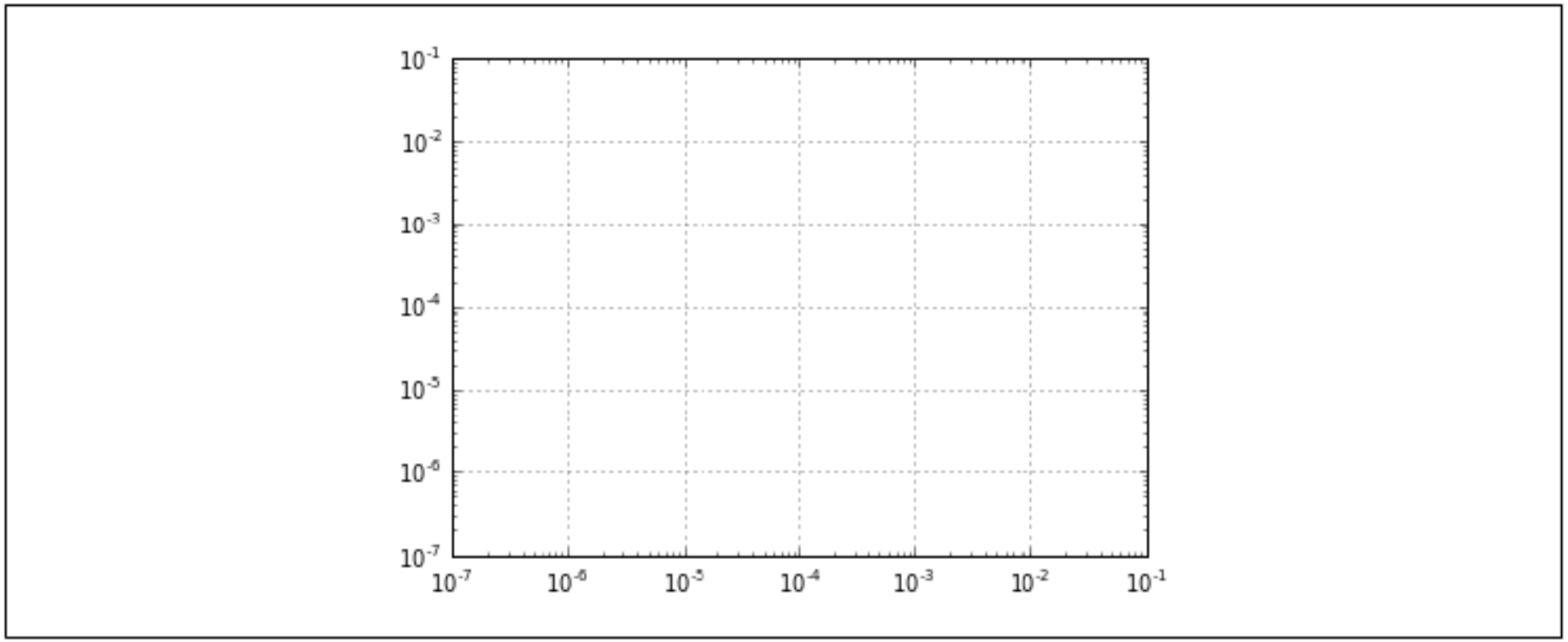

Figure 4-73. Example of logarithmic scales and labels

Customizing ticks

- We see here that each major tick shows a large tick mark and a label, while each minor tick shows a smaller tick mark with no label.
- We can customize these tick properties—that is, locations and labels—by setting the formatter and locator objects of each axis.
- Let's examine these for the x axis of the plot just shown:
- In[3]: **print**(ax.xaxis.get_major_locator())
- **print**(ax.xaxis.get_minor_locator())
- <matplotlib.ticker.LogLocator object at 0x107530cc0>
- <matplotlib.ticker.LogLocator object at 0x107530198>
- In[4]: **print**(ax.xaxis.get_major_formatter())
- **print**(ax.xaxis.get_minor_formatter())
- <matplotlib.ticker.LogFormatterMathtext object at 0x107512780>
- <matplotlib.ticker.NullFormatter object at 0x10752dc18>
- We see that both major and minor tick labels have their locations specified by a LogLocator (which makes sense for a logarithmic plot).
- Minor ticks, though, have their labels formatted by a NullFormatter; this says that no labels will be shown.

Customizing ticks

- **Hiding Ticks or Labels**
- Perhaps the most common tick/label formatting operation is the act of hiding ticks or labels.
- We can do this using plt.NullLocator() and plt.NullFormatter(), as shown here (Figure 4-74):
- In[5]: ax = plt.axes()
- ax.plot(np.random.rand(50))
- ax.yaxis.set_major_locator(plt.NullLocator())
- ax.xaxis.set_major_formatter(plt.NullFormatter())

Customizing ticks

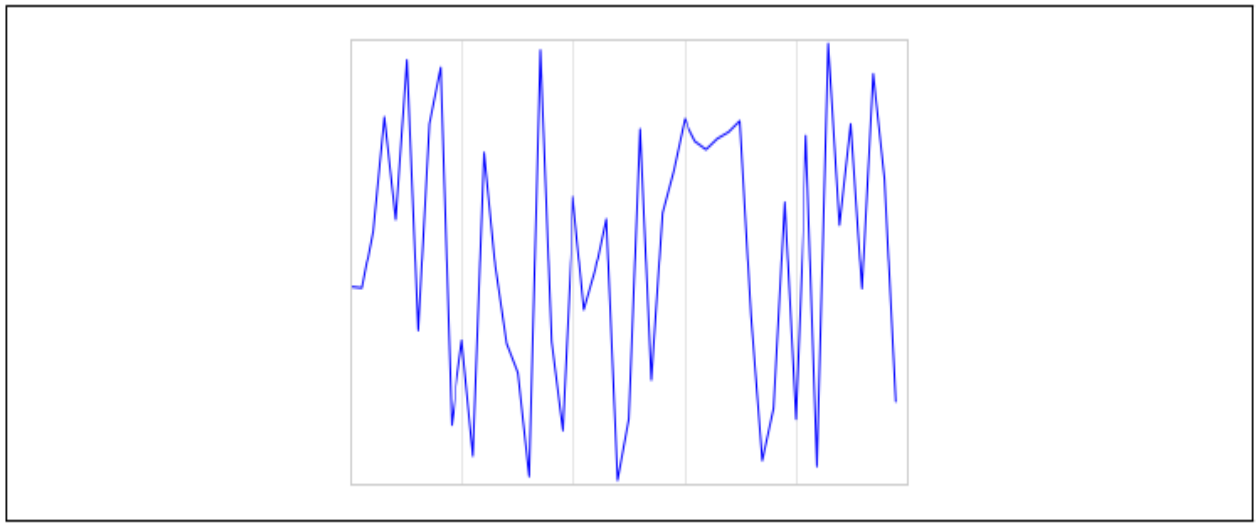

Figure 4-74. Plot with hidden tick labels (x-axis) and hidden ticks (y-axis)

Customizing ticks

- Notice that we've removed the labels (but kept the ticks/gridlines) from the x axis, and removed the ticks (and thus the labels as well) from the y axis.
- Having no ticks at all can be useful in many situations—for example, when you want to show a grid of
- images.
- For instance, consider Figure 4-75, which includes images of different faces, an example often used in supervised machine learning problems.

Customizing ticks

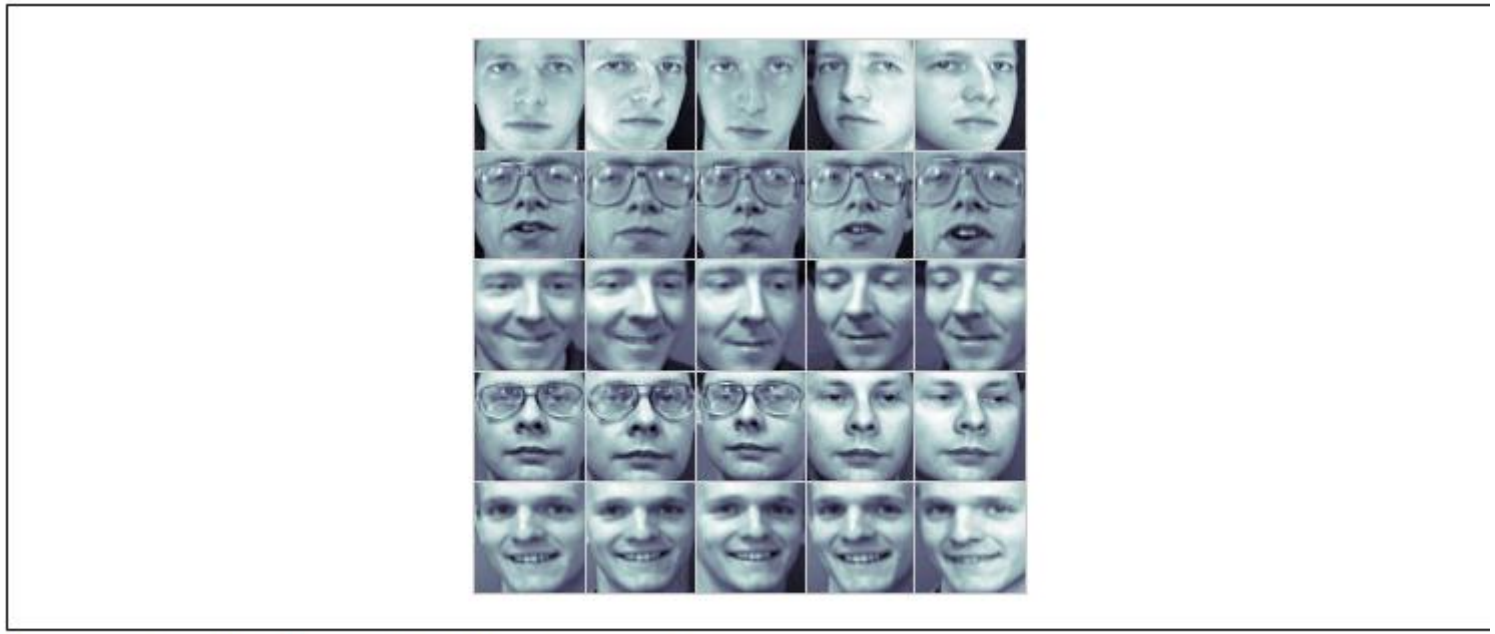

Figure 4-75. Hiding ticks within image plots

Customizing ticks

- Each image has its own axes, and we've set the locators to null because the tick values (pixel number in this case) do not convey relevant information for this particular visualization.

Customizing ticks

- **Reducing or Increasing the Number of Ticks**
- One common problem with the default settings is that smaller subplots can end up with crowded labels. We can see this in the plot grid shown in Figure 4-76:
- In[7]: fig, ax = plt.subplots(4, 4, sharex=True, sharey=True)

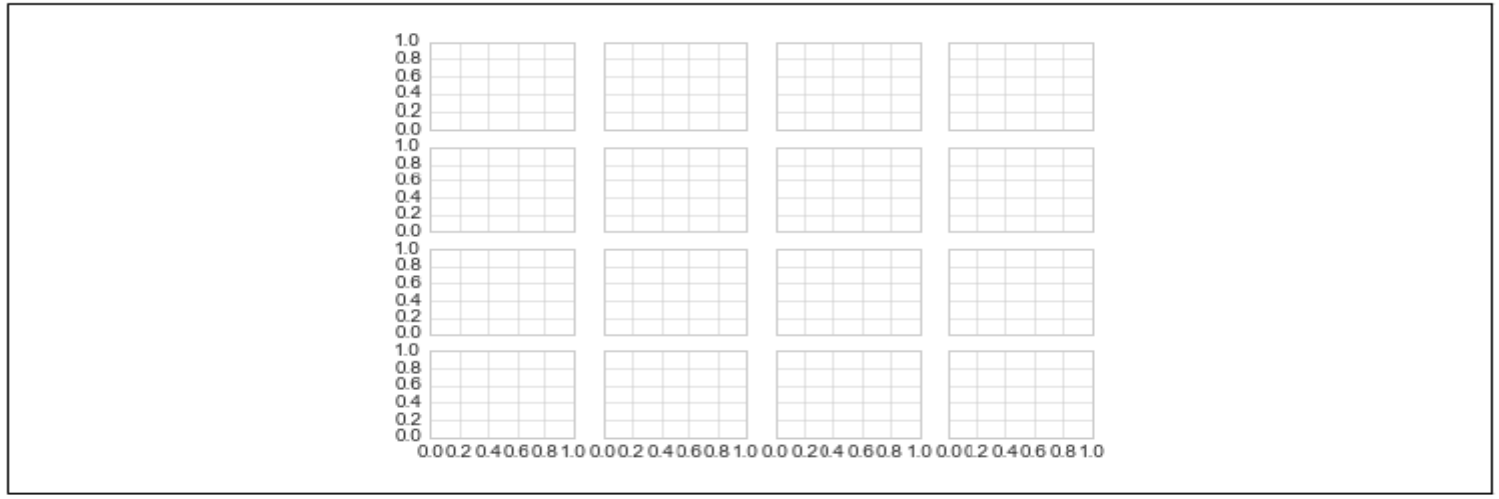

Figure 4-76. A default plot with crowded ticks

Customizing ticks

- Particularly for the x ticks, the numbers nearly overlap, making them quite difficult to decipher.
- We can fix this with the plt.MaxNLocator(), which allows us to specify the maximum number of ticks that will be displayed.
- Given this maximum number, Matplotlib will use internal logic to choose the particular tick locations (Figure 4-77):
- In[8]: *# For every axis, set the x and y major locator*
- **for** axi **in** ax.flat:
- axi.xaxis.set_major_locator(plt.MaxNLocator(3))
- axi.yaxis.set_major_locator(plt.MaxNLocator(3))
- fig

Customizing ticks

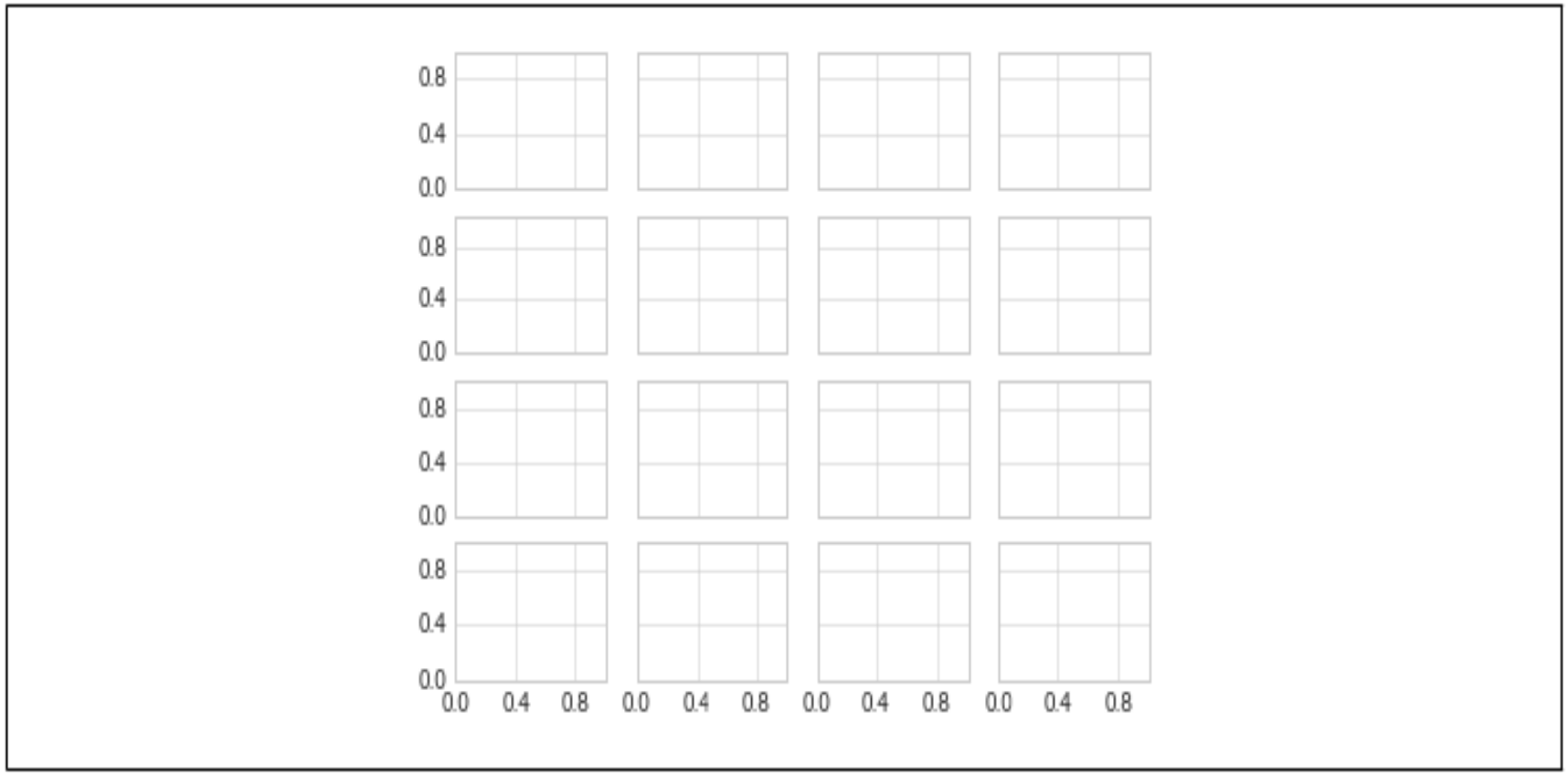

Figure 4-77. Customizing the number of ticks

Customizing ticks

- **Fancy Tick Formats**
- Matplotlib's default tick formatting can leave a lot to be desired; it works well as a broad default, but sometimes you'd like to do something more.
- Consider the plot shown in Figure 4-78, a sine and a cosine:
- In[9]: *# Plot a sine and cosine curve*
- fig, ax = plt.subplots()
- x = np.linspace(0, 3 * np.pi, 1000)
- ax.plot(x, np.sin(x), lw=3, label='Sine')
- ax.plot(x, np.cos(x), lw=3, label='Cosine')
- *# Set up grid, legend, and limits*
- ax.grid(True)
- ax.legend(frameon=False)
- ax.axis('equal')
- ax.set_xlim(0, 3 * np.pi);

Customizing ticks

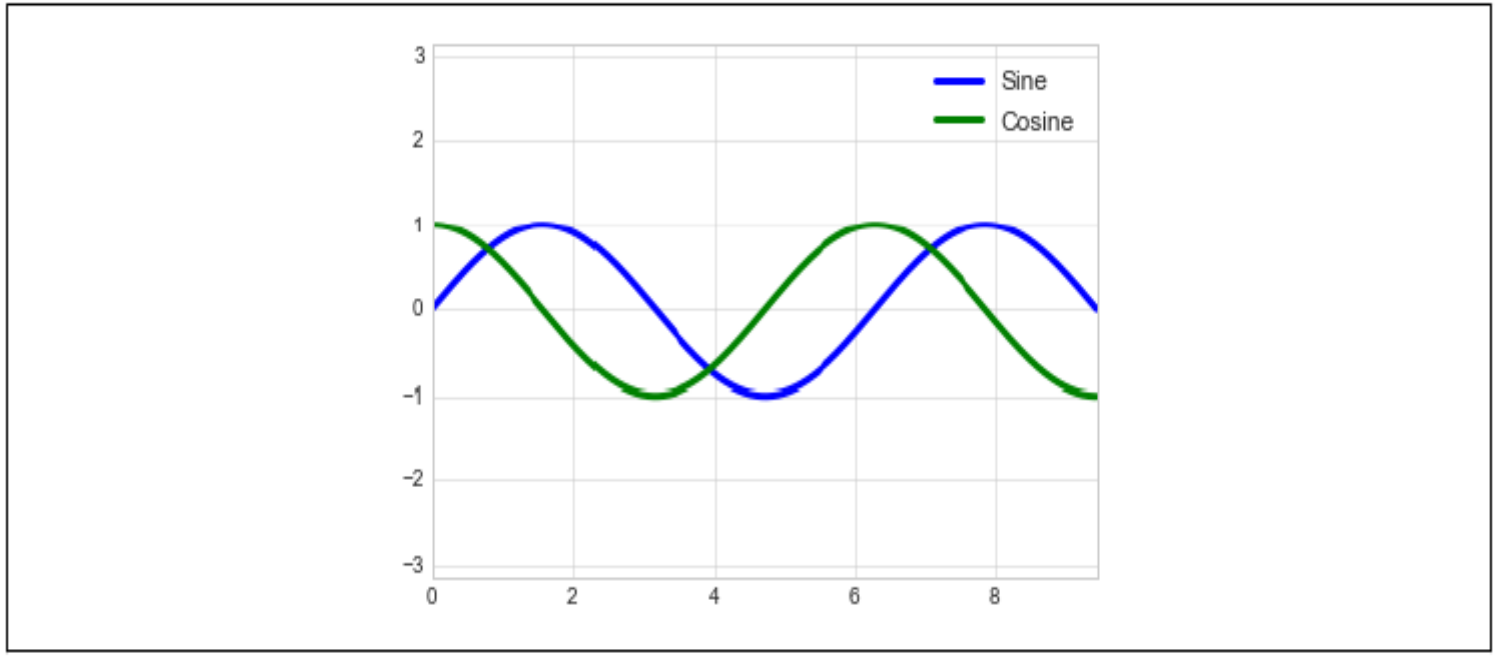

Figure 4-78. A default plot with integer ticks

Customizing ticks

- There are a couple changes we might like to make.
- First, it's more natural for this data to space the ticks and grid lines in multiples of π.
- We can do this by setting a MultipleLocator, which locates ticks at a multiple of the number you provide. For good measure, we'll add both major and minor ticks in multiples of $\pi/4$ (Figure 4-79):
- In[10]: ax.xaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
- ax.xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 4))
- fig

Customizing ticks

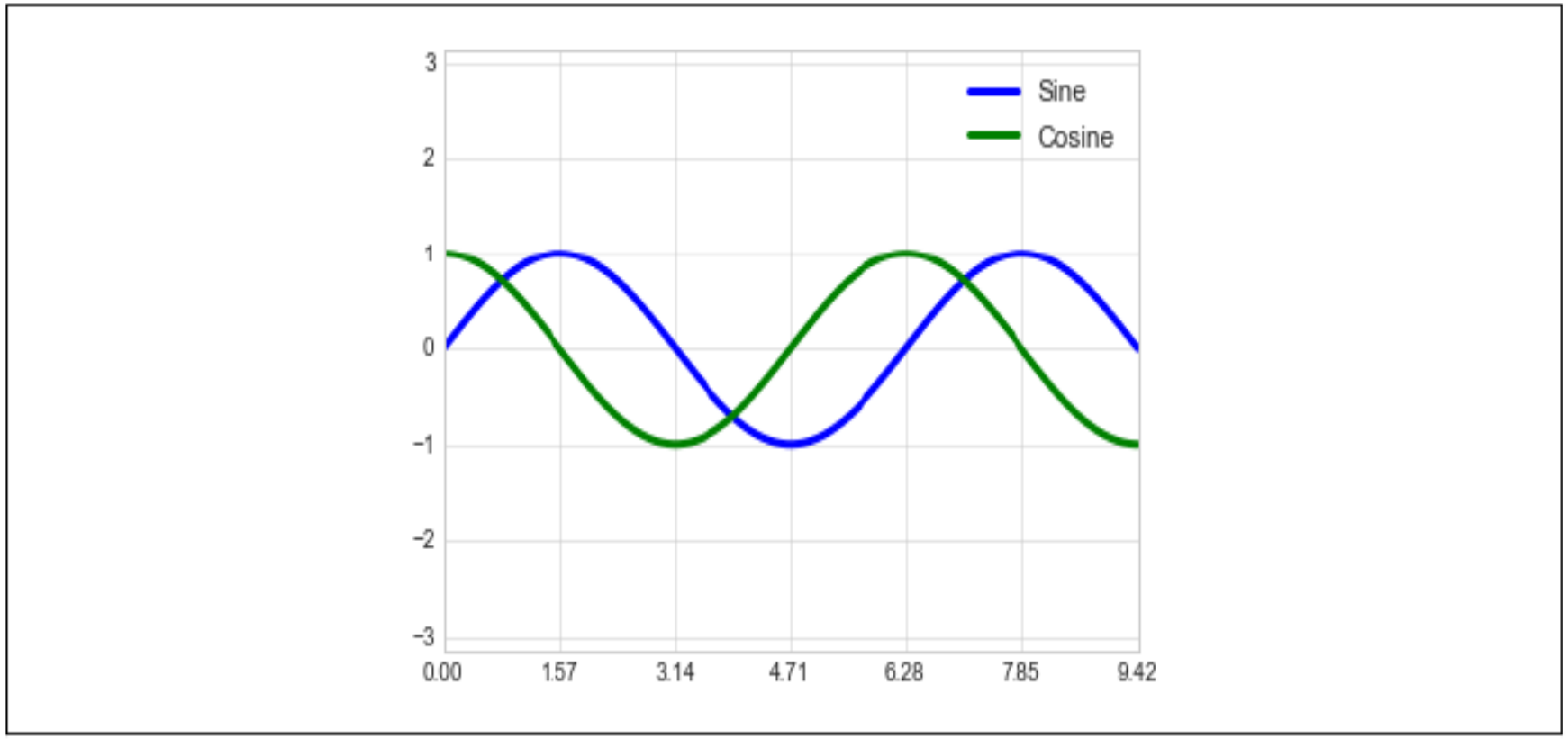

Figure 4-79. Ticks at multiples of pi/2

Customizing ticks

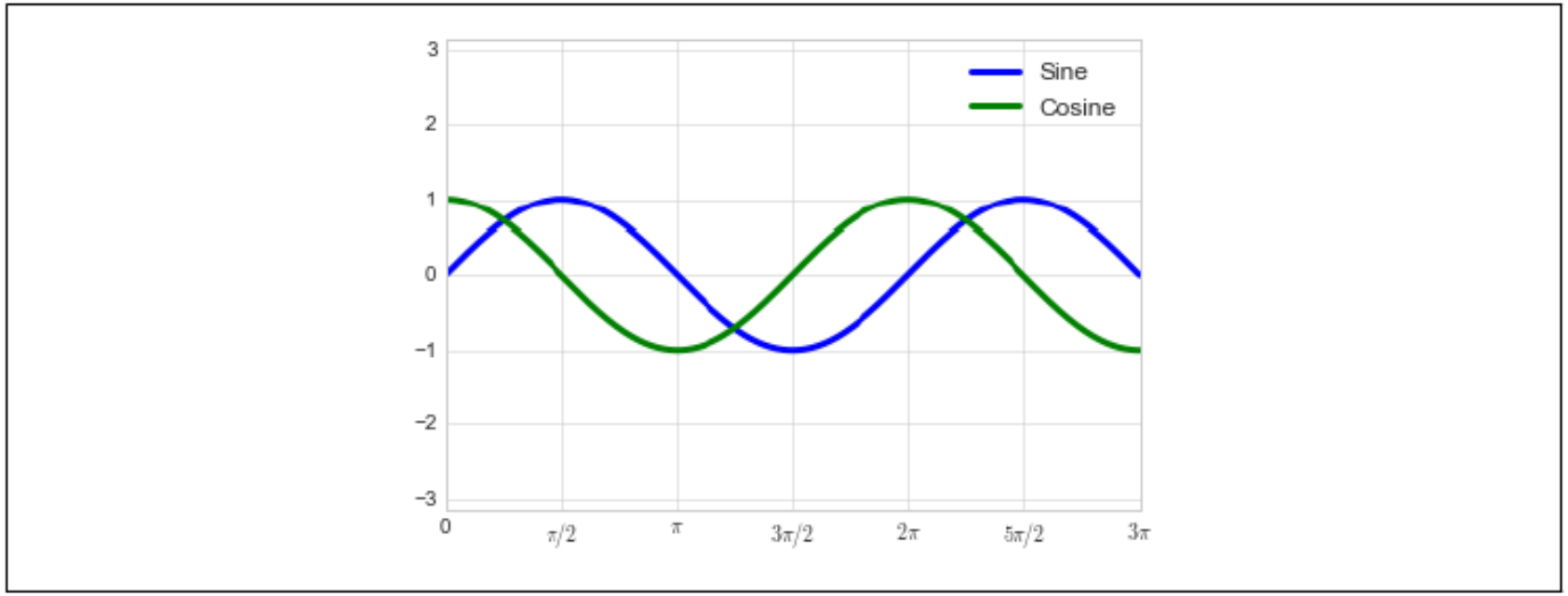

Figure 4-80. Ticks with custom labels

Customizing ticks

- **Summary of Formatters and Locators**

Locator class	Description
`NullLocator`	No ticks
`FixedLocator`	Tick locations are fixed
`IndexLocator`	Locator for index plots (e.g., where `x = range(len(y))`)

Customizing ticks

Locator class	Description
`LinearLocator`	Evenly spaced ticks from `min` to `max`
`LogLocator`	Logarithmically ticks from `min` to `max`
`MultipleLocator`	Ticks and range are a multiple of base
`MaxNLocator`	Finds up to a max number of ticks at nice locations
`AutoLocator`	(Default) `MaxNLocator` with simple defaults
`AutoMinorLocator`	Locator for minor ticks

Formatter class	Description
`NullFormatter`	No labels on the ticks
`IndexFormatter`	Set the strings from a list of labels
`FixedFormatter`	Set the strings manually for the labels
`FuncFormatter`	User-defined function sets the labels
`FormatStrFormatter`	Use a format string for each value
`ScalarFormatter`	(Default) Formatter for scalar values
`LogFormatter`	Default formatter for log axes

THANK YOU