Introduction to Python's Matplotlib

Margaret Gratian

NIH NATIONAL CANCER INSTITUTE

2/16/2023

Data Visualization in Python

- Matplotlib is one of the most common libraries for visualization data in Python
- It underpins many other plotting libraries and allows for a lot of customization
- Its interface is a bit confusing, and code associated with a plot can look pretty messy!

Some Matplotlib Based Plotting Libraries

- Seaborn: https://seaborn.pydata.org/
  - Library for statistical data visualization
- GeoPandas: https://geopandas.org/en/stable/
  - Library for working with geospatial data in Python, provides functionality for plotting data on maps
- Pandas: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html
  - Not specifically a visualization library, has a plot() method that can be used on DataFrames
- Networkx: https://networkx.guide/visualization/basics/
  - Library for graphs (networks), not specifically a visualization library but has drawing functionality for visualizing graphs (networks)

Some Non-Matplotlib Based Plotting Libraries

- Bokeh:
  - https://bokeh.org/
  - Library for interactive visualizations, well-suited for web applications
- Plotly:
  - https://plotly.com/python/
  - Library for interactive visualizations

Matplotlib – Two Ways of Using the Library

- **Implicit Pyplot Interface** – creation of figure and axes done for the user
  - Convenient for quick visualizations and data exploration/interactive work
- **Explicit Axes Interface** – visualizations are built step-by-step on a figure and axes enabling customization and control
  - Object oriented implementation – figure and axes are objects you manipulate
  - Everything you add to the plot is explicitly referenced
  - Things are customized after they are created, but before they are shown
- "The difference between these interfaces can be a bit confusing, particularly given snippets on the web that use one or the other, or sometimes multiple interfaces in the same example."

Source: https://matplotlib.org/stable/users/explain/api_interfaces.html

Implicit Pyplot Interface – A Quick Look

matplotlib

Plot types | Examples | Tutorials | Reference | User guide | Develop | Release notes

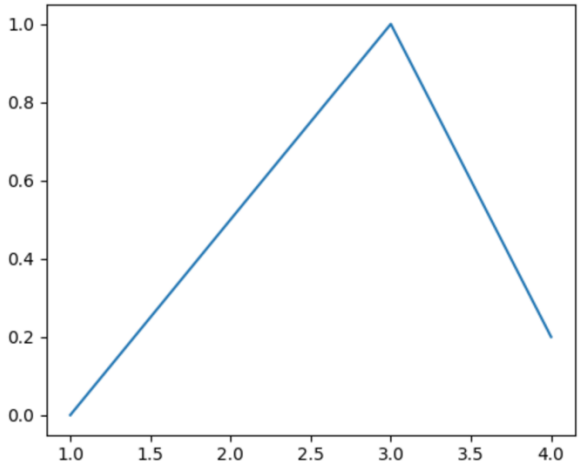
Section Navigation

- Matplotlib Application Interfaces (APIs)
- Backends
- Interactive figures
- Fonts in Matplotlib
- Event handling and picking
- Performance
- Interactive figures and asynchronous programming

```python
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

(Source code, png)




```python
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

Explicit Axes Interface – A Quick Look

matplotlib

Plot types | Examples | Tutorials | Reference | User guide | Develop | Release notes

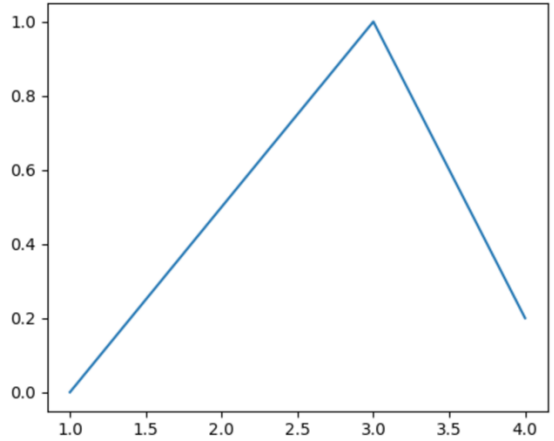
Section Navigation

- Matplotlib Application Interfaces (APIs)
- Backends
- Interactive figures
- Fonts in Matplotlib
- Event handling and picking
- Performance
- Interactive figures and asynchronous programming

```python
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.subplots()
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

(Source code, png)




```python
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.subplots()
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

When To Use Explicit vs. Implicit?

- When quickly trying to understand your data or work out ideas, the implicit interface can save time and allow you to focus on the data instead of the details (such as font sizes, colors, etc.)
- Once you start wanting to customized your plot for some final output (e.g., a paper or presentation), it may be time to shift to managing things with the explicit interface
- Matplotlib's documentation recommends choosing explicit plotting over implicit
- Most likely, you'll end up using a mixture of the two ☺

Pyplot Tutorial – A Matplotlib Tutorial Adapted to Public NIH Award Data

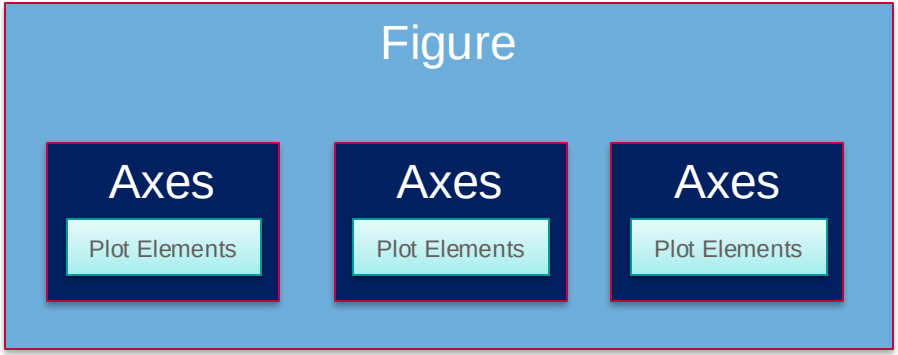
- Original tutorial: https://matplotlib.org/stable/tutorials/introductory/pyplot.html
- Pyplot documentation: https://matplotlib.org/stable/api/pyplot_summary.html
- Key points:
  - All adjustments to the plot are done via **plt.method_name()**, where the method name may be things like “plot”, “title”, “legend”
  - At its simplest, you can call **plt.plot(*x*, *y*, data=*pandas_df_name*)**
  - If you want to adjust the size of your plot, this must be the first thing you do, via **plt.figure(figsize=(*width*, *height*))**
  - After your initial plotting action (where you specify the data and plot style, e.g., bar, scatter, etc.) you can gradually add features and adjustments to the plot

The Lifecycle of a Plot – A Matplotlib Tutorial Adapted to Public NIH Award Data

- Original tutorial: https://matplotlib.org/stable/tutorials/introductory/lifecycle.html#sphx-glr-tutorials-introductory-lifecycle-py
- Terminology:
  - **Figure** – the final image (think of it like a canvas), containing one or more axes
  - **Axes** – lives in a figure, and contains the elements of your visualization such as the coordinate system, line (or bar, or scatter, etc.), tick marks, etc.). Plotting is done from the axes
  - **Axis** – the x/y axis on a plot (distinct from axes!)
- Key points:
  - Most adjustments made on the **axes** of the plot (e.g**., ax.set_title("title")**)
  - **Axes** refers to the Matplotlib *class* defining the plotting area where most things are placed. When you write **fig, ax = plt.subplots()**, you've created an *instance* of the Axes class called **ax** (you've also created an *instance* of the Figure *class* called fig!)

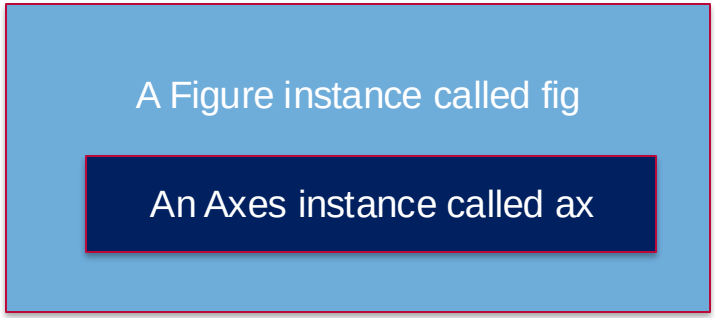
Figures and Axes

The Figure and Axes classes



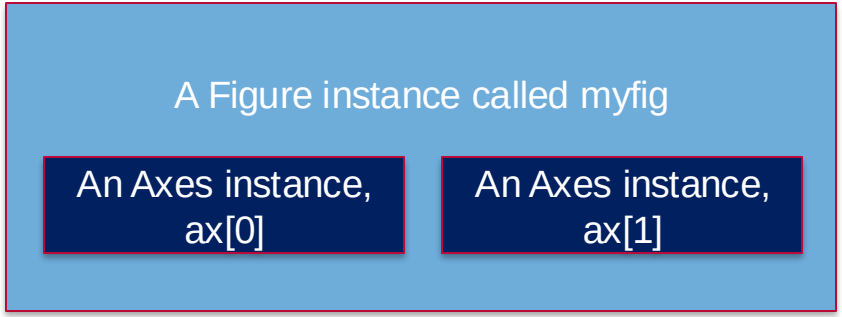
A Figure is a type of canvas that can hold one or multiple Axes. Plotting occurs within an Axes.

Ex 1: fig, ax = plt.subplots()



Ex 2: myfig, axs = plt.subplots(2)

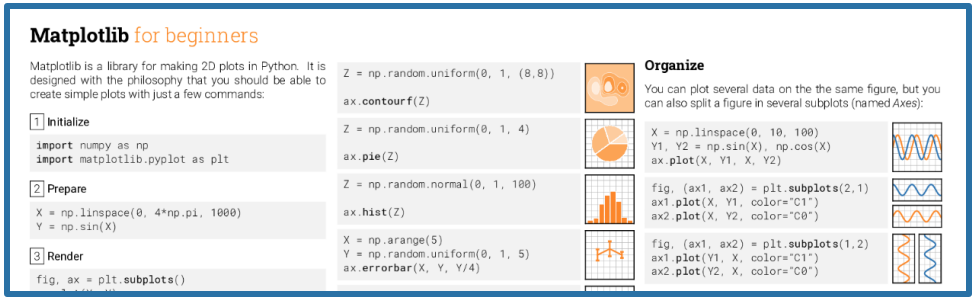
By specifying 2 subplots, we've created an array of 2 axes now that we can access using list notation.

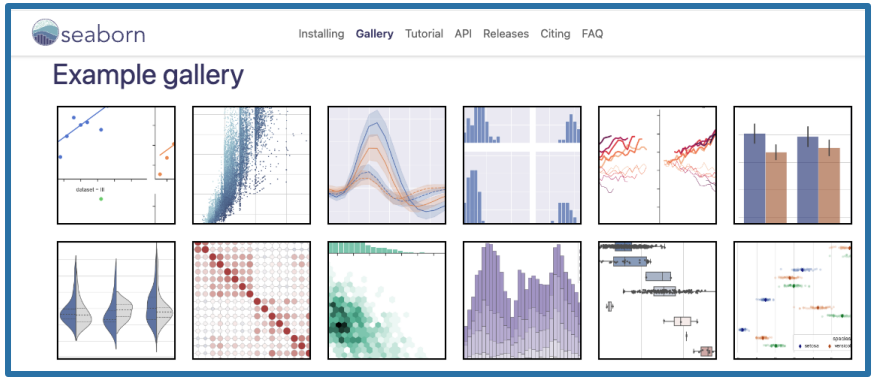


See more examples of multiple axes here:
https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html

Additional Resources

- Matplotlib's "cheatsheets": https://matplotlib.org/cheatsheets/
  - Quick references and handouts for plotting
- Seaborn's Gallery: https://seaborn.pydata.org/examples/index.html
  - A lot of different plot examples with associated code





NIH NATIONAL CANCER INSTITUTE

www.cancer.gov www.cancer.gov/espanol