Data Visualization in Python

Data Visualization

> *Data graphics visually display measured quantities by means of the combined use of points, lines, a coordinate system, numbers, words, shading, and color.*

– Edward Tufte, *The Visual Display of Quantitative Information*

Data Visualization in Python

- Matplotlib
  - Matlab-like plotting interface
  - The granddaddy of all scientific plotting in Python
  - Powerful, low-level
  - Built on NumPy arrays
- Seaborn
  - Higher-level API on top of Matplotlib
  - Integrates with Pandas DataFrames
- Bokeh or Plotly/Dash
  - Interactive visualizations like D3

Matplotlib

Standard import:

```
import matplotlib.pyplot as plt
```

Three contexts:

- Python script: (example)

  ```
  xs = np.linspace(0, 10, 100)
  plt.plot(xs, np.sin(xs))
  plt.plot(xs, np.cos(xs))
  plt.show()
  ```

  After constructing plot, call `plt.show()` only once, typically at end of script. Calling `plt.show()` multiple times is undefined and may cause problems.

- iPython:

  ```
  In [4]: %matplotlib
  Using matplotlib backend: MacOSX
  ```

  Now any plot command will open a figure window. Can force redraw with `plt.draw()`.

- Jupyter Notebook:

  ```
  %matplotlib notebook
  ```

  Embed interactive plots in notebook.

  ```
  %matplotlib inline
  ```

  Embed static plot images in notebook. We'll usually use this option

Figures and Axes

Every plot resides in a figure, which can have a number of subplots.

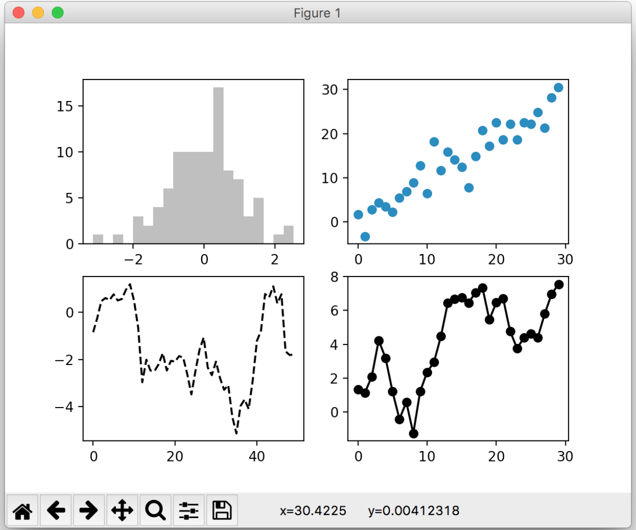
```
fig = plt.figure()
```

Here we make four subplots in a 2x2 layout and put a different kind of plot in each one. Notice 1-based indexing third argument – top left to bottom right.

```
In [6]: ax1 = fig.add_subplot(2, 2, 1)

In [7]: ax2 = fig.add_subplot(2, 2, 2)

In [9]: ax3 = fig.add_subplot(2, 2, 3)

In [10]: ax4 = fig.add_subplot(2, 2, 4)

In [13]: ax1.hist(np.random.randn(100), bins=20, color=’k’, alpha=0.3)
Out[13]: ... elided for brevity
 <a list of 20 Patch objects>)

In [14]: ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
Out[14]: <matplotlib.collections.PathCollection at 0x11477c1d0>

In [15]: ax3.plot(np.random.randn(50).cumsum(), ’k--’)
Out[15]: [<matplotlib.lines.Line2D at 0x114411fd0>]

In [18]: ax4.plot(np.random.randn(30).cumsum(), ’ko-’)
Out[18]: [<matplotlib.lines.Line2D at 0x1146ce0b8>]
```

Figures

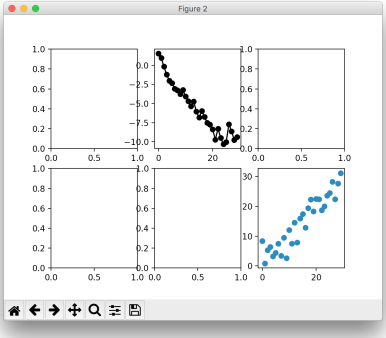
The commands on the previous slide would produce this:



plt.subplots

Matplotlib includes a convenience method for making subplots.

```
In [20]: fig, axes = plt.subplots(2, 3)

In [22]: axes[0,1].plot(np.random.randn(30).cumsum(), 'ko-')
Out[22]: [<matplotlib.lines.Line2D at 0x1204e4470>]

In [23]: axes[1,2].scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
Out[23]: <matplotlib.collections.PathCollection at 0x1204f8940>
```



Note the 0-based indexing for axes.

Colors, Markers, and Line Styles

Notice the 'ko-' in `plot(np.random.randn(30).cumsum(), 'ko-')`

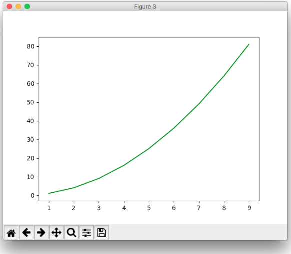
- 'k' is a color for the marker and line used in the plot. A few examples:
    - 'b' - blue
    - 'g' - green
    - 'r' - red
    - 'k' - black
    - 'w' - white
- 'o' is a marker. A few examples:
    - '.' - point marker
    - ',' - pixel marker
    - 'o' - circle marker
    - 'v' - triangle$_{\text{down}}$ marker
    - '$^{'}$ - triangle$_{\text{up}}$ marker
    - '<' - triangle$_{\text{left}}$ marker
    - '>' - triangle$_{\text{right}}$ marker
- '-' is a line style. A few examples:
    - '-' - solid line style
    - '–' - dashed line style
    - '-. - dash-dot line style
    - ':' - dotted line style

For complete details see documentation for `plt.plot`

Subplots Shortcut

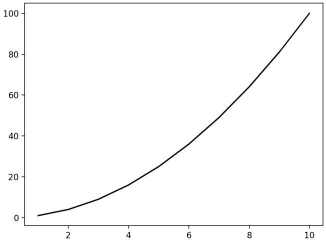
```
In [35]: xs,ys = list(range(1, 10)), [x**2 for x in range(1, 10)]

In [37]: fig, axis = plt.subplots(1, 1)

In [38]: axis.plot(xs, ys, linestyle='-', color='g')
Out[38]: [<matplotlib.lines.Line2D at 0x120c60518>]
```

- Notice that if you create a figure with one subplot `plt.subplots` returns a single axis instead of an array of axes.
- Notice also the explicit linestyle and color.



What’s wrong with this picture?

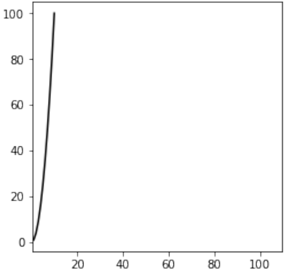
```
In [7]: xs,ys = list(range(1, 11)), [x**2 for x in range(1, 11)]

In [8]: plt.plot(xs, ys, ’k-’)
Out[8]: [<matplotlib.lines.Line2D at 0x1145205f8>]
```



Square Plot

Square plot makes x and y axes equal:

```
xs,ys = list(range(1, 11)), [x**2 for x in range(1, 11)]
plt.plot(xs, ys, 'k-')
plt.axis('square')
```



See docs for `xlim` and `ylim`.