Data visualization in python

Day 2
A variety of packages and philosophies

• (today) matplotlib: http://matplotlib.org/
  – Gallery: http://matplotlib.org/gallery.html
  – Frequently used commands: http://matplotlib.org/api/pyplot_summary.html

• Seaborn: http://stanford.edu/~mwaskom/software/software/seaborn/

• ggplot:
  – R version: http://docs.ggplot2.org/current/
  – Python port: http://ggplot.yhathq.com/

• Bokeh (live plots in your browser)
Matplotlib

- Gallery: http://matplotlib.org/gallery.html
- Top commands: http://matplotlib.org/api/pyplot_summary.html
- Provides "pylab" API, a mimic of matlab
- Many different graph types and options, some obscure
Matplotlib

• Resulting plots represented by python objects, from entire figure down to individual points/lines.

• Large API allows any aspect to be tweaked

• Lengthy coding sometimes required to make a plot "just so"
Seaborn

- [https://stanford.edu/~mwaskom/software/seaborn/](https://stanford.edu/~mwaskom/software/seaborn/)
- Implements more complex plot types
  - Joint points, clustergrams, fitted linear models
- Uses matplotlib "under the hood"
Others

• ggplot:
  – (Original) R version: http://docs.ggplot2.org/current/
  – A recent python port: http://ggplot.yhathq.com/
  – We'll discuss this on the R side tomorrow, both the basics of both work similarly.

• Bokeh (live plots in your browser)

• Plotting functionality built-in to pandas
Using matplotlib

• This 'magic' command tells ipython:
  – Load matplotlib (import as the alias "mpl")
  – Load the pyplot interface (as "plt"), which approximates the plotting functionality and syntax of MATLAB. Put the output inline with notebook results (rather than saving to file, opening a new window, etc)

In[1]: %pylab inline

• What if we're not using ipython notebook?

  import matplotlib as mpl
  import pyplot as plt
  import numpy as np

All the magic commands:
Generate some data to plot

- Draw 100 samples into x from N(0, 10)
- Draw 100 samples into y from N(20, 2)
- Set z = 3 times y plus x plus N(0, 1)

- Inspect sample mean and standard deviation using numpy functions mean, std:

```python
>>> print 'x mean: ' , np.mean(x)
>>> print 'x std: ' , np.std(x)
x mean: 0.0820478565308
x std: 9.9856477737
```
Scatterplots

- `plt.scatter`
- `plt.title`
- `plt.xlabel`
- `plt.ylabel`

http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.scatter
Overlay multiple series on a single plot

- Simply issue more than one plotting command in a row

- Just a few of the parameters you can customize:
  - marker
  - color (for other plot types, edgecolor, fillcolor)
  - label
  - Size

- `plt.legend()` adds a legend
Adjacent plots

```python
>>> my_figure, my_axes = plt.subplots(1, 2, sharey=True, sharex=True)
>>> my_axes[0].scatter(...)
# ...

http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.subplots
```
Boxplots

- `plt.boxplot(...)`

[Diagram showing boxplots with labels x, y, z]

[Link to documentation: http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.boxplot]
Histograms

- `plt.hist(...)`

---

[Online documentation link]

http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.hist
Why are these binned differently?

```
In [35]: plt.hist(meansland2[:, 0], color='red')
plt.hist(meansland2[:, 1], color='blue')

Out[35]: (array([ 3., 16., 55., 141., 253., 238., 174., 95., 18., 7.]),
array([[ 0.13066485, 0.20263161, 0.27459836, 0.34656511, 0.41853187,
        0.49049862, 0.56246537, 0.63443213, 0.70639888, 0.77836563,
        0.85033238]),
        <a list of 10 Patch objects>))
```

What's all this?
Check the manual...

Plot a histogram.

Compute and draw the histogram of \( x \). The return value is a tuple \((n, bins, patches)\) or \([([n_0, n_1, ...], bins, [patches_0, patches_1, ...])\) if the input contains multiple data.

Multiple data can be provided via \( x \) as a list of datasets of potentially different length \(([x_0, x_1, ...])\), or as a 2-D ndarray in which each column is a dataset. Note that the ndarray form is transposed relative to the list form.

Masked arrays are not supported at present.

**Parameters:**

- \( x : (n,) \) array or sequence of \((n,)\) arrays
  - Input values, this takes either a single array or a seqency of arrays which are not required to be of the same length
  - **bins** : integer or array_like, optional
    - If an integer is given, \( bins + 1 \) bin edges are returned, consistently with numpy.histogram() for numpy version \( \geq 1.3 \).
    - Unequally spaced bins are supported if bins is a sequence.
    - default is 10

- **range** : tuple or None, optional

**Returns:**

- \( n : \) array or list of arrays
  - The values of the histogram bins. See normed and weights for a description of the possible semantics. If input \( x \) is an array, then this is an array of length \( n_{bins} \). If input \( x \) is a sequence arrays \([data_1, data_2, ...]\), then this is a list of arrays with the values of the histograms for each of the arrays in the same order.
  - **bins** : array
    - The edges of the bins. Length \( nbins + 1 \) (\( nbins \) left edges and right edge of last bin). Always a single array even when multiple data sets are passed in.
  - **patches** : list or list of lists
    - Silent list of individual patches used to create the histogram or list of such list if multiple input datasets.

http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.hist
Get bin boundaries from 1st hist, use in 2nd

_ = something(...) here,

means call function something (or interpret some expression), get the result, and then toss it (don't put in a variable)

In [47]: _, da_bins, _ = plt.hist(meansland2[:, 0], bins=20, color='red')
_ = plt.hist(meansland2[:, 1], bins=da_bins, color='blue')
No fill color – can see through overlapping bins

```python
In [44]: da_bins, _ = plt.hist(meansland2[:, 0], bins=20, ec='red', fc='none')
    _ = plt.hist(meansland2[:, 1], bins=da_bins, ec='blue', fc='none')
```