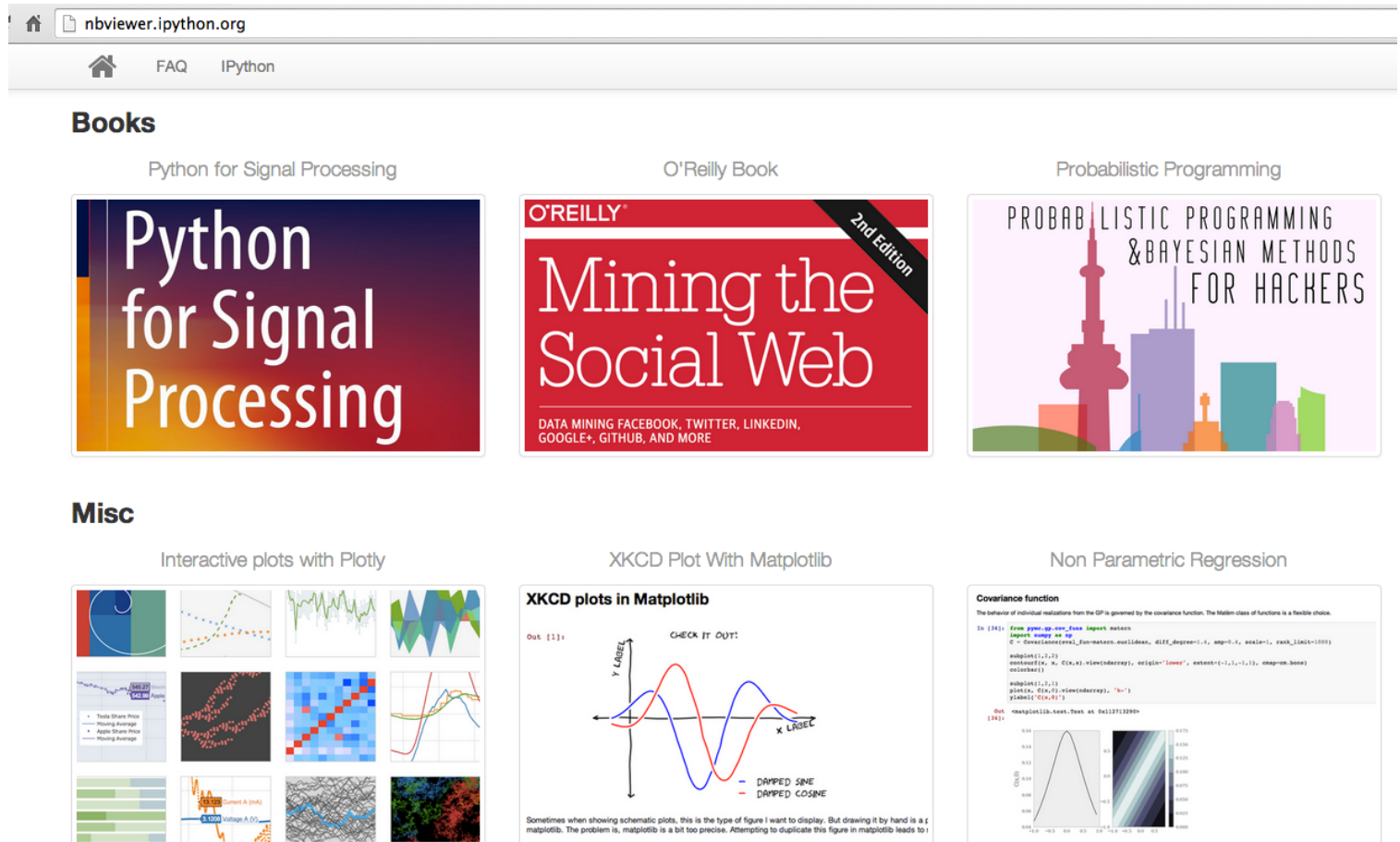


Running Code in the IPython Notebook

First and foremost, the IPython Notebook is an interactive environment for writing and running Python code.

Motivation



The screenshot shows the nbviewer.ipython.org website. The header includes a home icon, the URL, and links to FAQ and IPython. The main content is divided into two sections: 'Books' and 'Misc'.

Books

- Python for Signal Processing**: A book cover with a blue and orange gradient.
- O'Reilly Book**: A book cover for 'Mining the Social Web' by Michael H. Kohn, 2nd Edition, with a red background.
- Probabilistic Programming**: A book cover for 'Probabilistic Programming & Bayesian Methods for Hackers' by John D. Cook, with a pink background and a city skyline.

Misc

- Interactive plots with Plotly**: A grid of various interactive plots including line charts, bar charts, and heatmaps.
- XKCD Plot With Matplotlib**: A plot titled 'CHECK IT OUT!' showing a damped sine wave and a damped cosine wave. The text below the plot reads: 'Sometimes when showing schematic plots, this is the type of figure I want to display. But drawing it by hand is a s*** matplotlib. The problem is, matplotlib is a bit too precise. Attempting to duplicate this figure in matplotlib leads to f***'.
- Non Parametric Regression**: A plot titled 'Covariance function' showing a Gaussian curve and a heatmap of a covariance matrix.

First, let's take a look at the IPython shell that the Notebook sits on top of

Setup a directory for this work

```
cd <YOURFORKEDREPO>
git pull origin master
cd lab_submissions/lab03
mkdir <FLASTNAME>
cd <FLASTNAME>
```

IPython -- An enhanced Interactive Python

IPython offers a combination of convenient shell features, special commands and a history mechanism for both input (command history) and output (results caching, similar to Mathematica). It is intended to be a fully compatible replacement for the standard Python interpreter, while offering vastly improved functionality and flexibility.

Magic Commands

```
%magic
```

TAB completions

Dynamic Object Information

```
?word
```

Input and Output caching

To run:

```
ipython
```

```
?
```

review the help later if interested

IPython Notebook

- Full manual: <http://ipython.org/notebook.html>

Starting the notebook server Start running a notebook server from the command line using the following command:

```
ipython notebook
```

This will print some information about the notebook server in your console, and open a web browser to the URL of the web application (by default, <http://127.0.0.1:8888>).

Code cells allow you to enter and run Python code

Run a code cell using **Shift-Enter** or pressing the "Play" button in the toolbar above:

```
In []: a = 10
```

```
In []: print(a)
```

Managing the IPython Kernel

Code is run in a separate process called the IPython Kernel. The Kernel can be interrupted or restarted. Try running the following cell and then hit the "Stop" button in the toolbar above.

```
In [4]: import time
        time.sleep(10)
        print('OK, I\'m up!')
```

OK, I'm up!

If the Kernel dies you will be prompted to restart it. Here we call the low-level system `libc.time` routine with the wrong argument via `ctypes` to segfault the Python interpreter:

```
In [*]: import sys
        from ctypes import CDLL
        # This will crash a Linux or Mac system; equivalent calls can be made on Wind
        ows
        dll = 'dylib' if sys.platform == 'darwin' else 'so.6'
        libc = CDLL("libc.%s" % dll)
        libc.time(-1) # BOOM!!
```

All of the goodness of IPython works

Here are two system aliases:

```
In []: pwd
```

```
In []: ls
```

Any command line program can be run using `!` with string interpolation from Python variables:

```
In []: message = 'The IPython notebook is great!'
        # note: the echo command does not run on Windows, it's a unix command.
        !echo $message
```

Tab completion works:

```
In []: import numpy
        numpy.random.
```

Tab completion after `(` brings up a tooltip with the docstring:

```
In []: numpy.random.rand(
```

```
numpy.random.rand()
```

Adding `?` opens the docstring in the pager below:

```
In []: magic?
```

Working with external code

There are a number of ways of getting external code into code cells.

Pasting code with `>>>` prompts works as expected:

```
In []: >>> the_world_is_flat = 1
>>> if the_world_is_flat:
...     print("Be careful not to fall off!")
```

The `%load` magic lets you load code from URLs or local files:

```
In []: %matplotlib inline
```

```
In [15]: %load http://matplotlib.sourceforge.net/mpl_examples/pylab_examples/integral_
demo.py
```