IPythonNotebookIntroduction

February 21, 2015

1 Getting started with Python and the IPython notebook

The IPython notebook is an interactive, web-based environment that allows one to combine code, text and graphics into one unified document. All of the lectures in this course have been developed using this tool. In this lecture, we will introduce the notebook interface and demonstrate some of its features.

2 Cells

The IPython notebook has two types of cells:

```
* Markdown
* Code
```

Markdown is for text, and even allows some typesetting of mathematics, while the code cells allow for coding in Python and access to many other packages, compilers, etc.

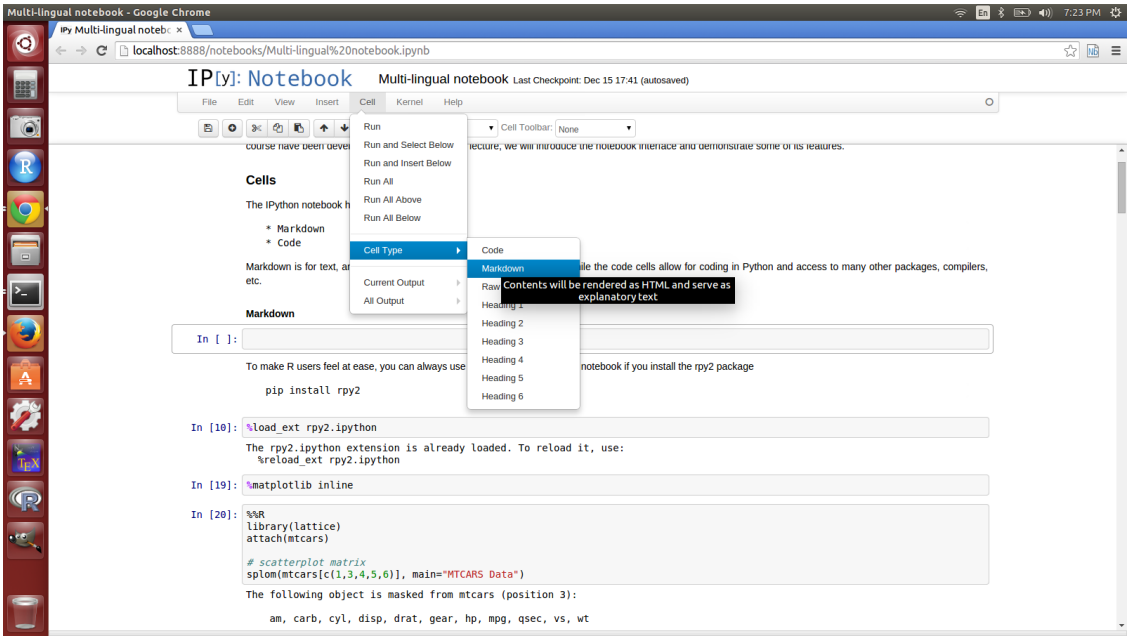
2.1 Markdown

To enter a markdown cell, just choose the cell tab and set the type to 'Markdown'.

```
In [2]: from IPython.display import Image
```

```
In [3]: Image(filename='screenshot.png')
```

Out[3]:



The current cell is now in Markdown mode, and whatever is entered is assumed to be markdown code. For example, text can be put into *italics* or **bold**. A bulleted list can be entered as follows:

Bulleted List * Item 1 * Item 2

Markdown has many features, and a good reference is located at:

http://daringfireball.net/projects/markdown/syntax

2.2 Code Cells

Code cells take Python syntax as input. We will see a lot of those shortly, when we begin our introduction to Python. For the moment, we will highlight additional uses for code cells.

2.2.1 Magic Commands

Magic commands work a lot like OS command line calls - and in fact, some are just that. To get a list of available magics:

```
In [4]: %lsmagic

Out[4]: Available line magics:
        %alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors  %

        Available cell magics:
        %%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript  %%latex  %%perl  %

        Automagic is ON, % prefix IS NOT needed for line magics.
```

Notice there are line and cell magics. Line magics take the entire line as argument, while cell magics take the cell. As 'automagic' is on, we can omit the % when making calls to line magics.

```
In [5]: ls

IntroductionToPython.ipynb         IPythonNotebookPolyglot.ipynb
IPythonNotebookIntroduction.ipynb  screenshot.png

In [22]: cp IntroductionToPython.ipynb IP2.ipynb

In [23]: ls

IntroductionToPython.ipynb  IPythonNotebookIntroduction.ipynb  screenshot.png
IP2.ipynb                   IPythonNotebookPolyglot.ipynb

In [24]: rm IP2.ipynb

In [25]: ls

IntroductionToPython.ipynb         IPythonNotebookPolyglot.ipynb
IPythonNotebookIntroduction.ipynb  screenshot.png
```

We can make all the above system calls in one cell, by using the cell magic, %%system

```
In [6]: %%system
        cp IntroductionToPython.ipynb  IP2.ipynb
        ls
        rm IP2.ipynb
        ls
```

```
Out[6]: ['IntroductionToPython.ipynb',
         'IP2.ipynb',
         'IPythonNotebookIntroduction.ipynb',
         'IPythonNotebookPolyglot.ipynb',
         'screenshot.png',
         'IntroductionToPython.ipynb',
         'IPythonNotebookIntroduction.ipynb',
         'IPythonNotebookPolyglot.ipynb',
         'screenshot.png']
```

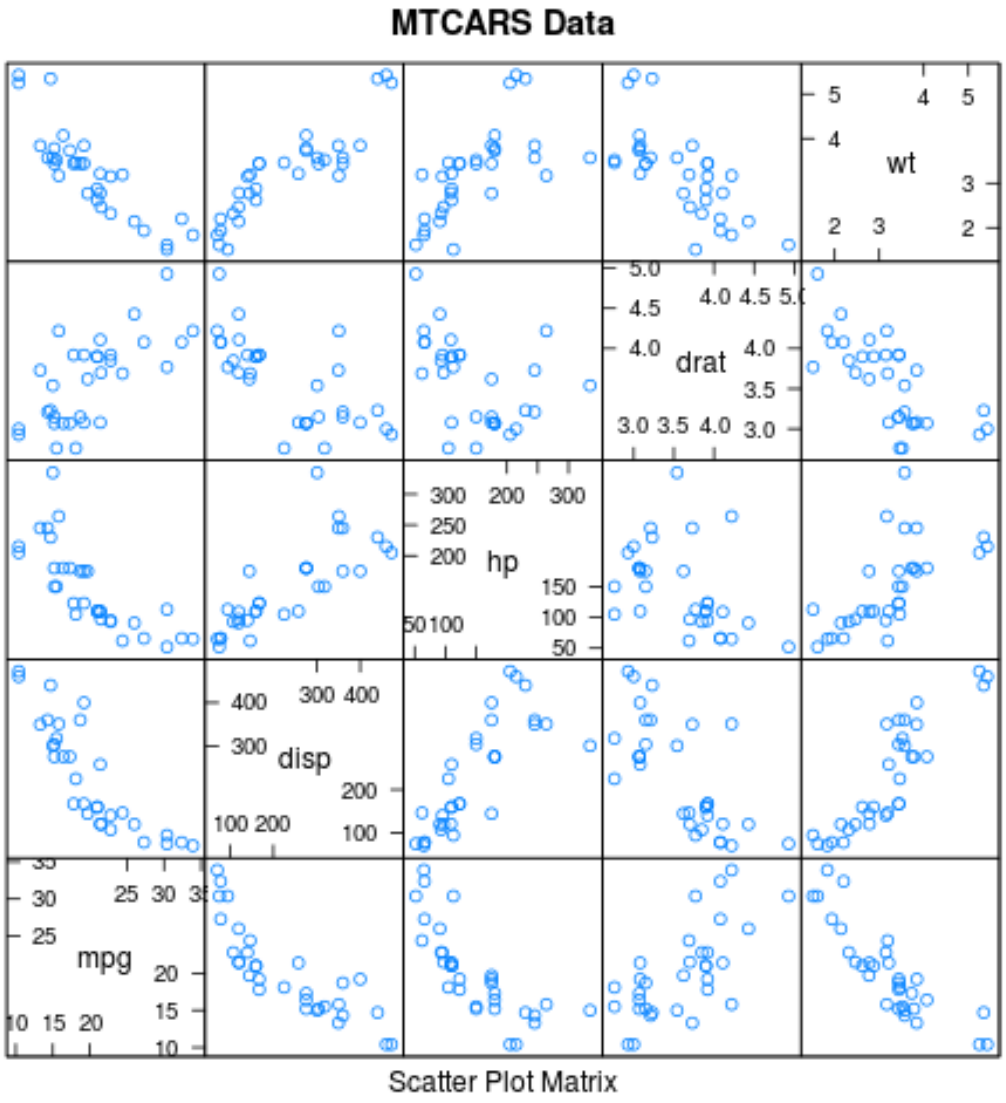
But magics are much more than system calls! We can even use R from within the IPython notebook if you install the rpy2 package

```
pip install rpy2
```

```
In [7]: %load_ext rpy2.ipython
```

```
In [8]: %matplotlib inline
```

```
In [9]: %%R
        library(lattice)
        attach(mtcars)

        # scatterplot matrix
        splom(mtcars[c(1,3,4,5,6)], main="MTCARS Data")
```



Matlab works too:

```
pip install pymatbridge
```

```
In [15]: !pip install --upgrade pymatbridge
```

```
Requirement already up-to-date: pymatbridge in /home/bitnami/anaconda/lib/python2.7/site-packages
Cleaning up...
```

```
In [16]: import pymatbridge as pymat
         ip = get_ipython()
         pymat.load_ipython_extension(ip)
```

```
In [17]: %%matlab
         xgv = -1.5:0.1:1.5;
         ygv = -3:0.1:3;
         [X,Y] = ndgrid(xgv,ygv);
```

```
V = exp(-(X.^2 + Y.^2));
surf(X,Y,V)
title('Gridded Data Set', 'fontweight','b');
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)

<ipython-input-17-8ef3de53fe4f> in <module>()
----> 1 get_ipython().run_cell_magic(u'matlab', u'', u"\nxgv = -1.5:0.1:1.5;\nygv = -3:0.1:3;\n[X,Y]


/home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/interactiveshell.pyc in run_cell
   2160             magic_arg_s = self.var_expand(line, stack_depth)
   2161             with self.builtin_trap:
-> 2162                 result = fn(magic_arg_s, cell)
   2163             return result
   2164


/home/bitnami/anaconda/lib/python2.7/site-packages/pymatbridge/matlab_magic.pyc in matlab(self,


/home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/magic.pyc in <lambda>(f, *a, **
    191     # but it's overkill for just that one bit of state.
    192     def magic_deco(arg):
--> 193         call = lambda f, *a, **k: f(*a, **k)
    194
    195         if callable(arg):


/home/bitnami/anaconda/lib/python2.7/site-packages/pymatbridge/matlab_magic.pyc in matlab(self,
    215             e_s += "\n----------------------"
    216             e_s += "\nAre you sure Matlab is started?"
--> 217             raise RuntimeError(e_s)
    218
    219


RuntimeError: There was an error running the code:

xgv = -1.5:0.1:1.5;
ygv = -3:0.1:3;
[X,Y] = ndgrid(xgv,ygv);
V = exp(-(X.^2 + Y.^2));
surf(X,Y,V)
title('Gridded Data Set', 'fontweight','b');
----------------------
Are you sure Matlab is started?
```

And it is also OK if you prefer Octave. Just type

```
pip install oct2py
```

```
In [18]: %load_ext octavemagic
```

```
In [23]: %%octave
         A = reshape(1:4,2,2);
         b = [36; 88];
         A\b
         [L,U,P] = lu(A)
         [Q,R] = qr(A)
         [V,D] = eig(A)
```

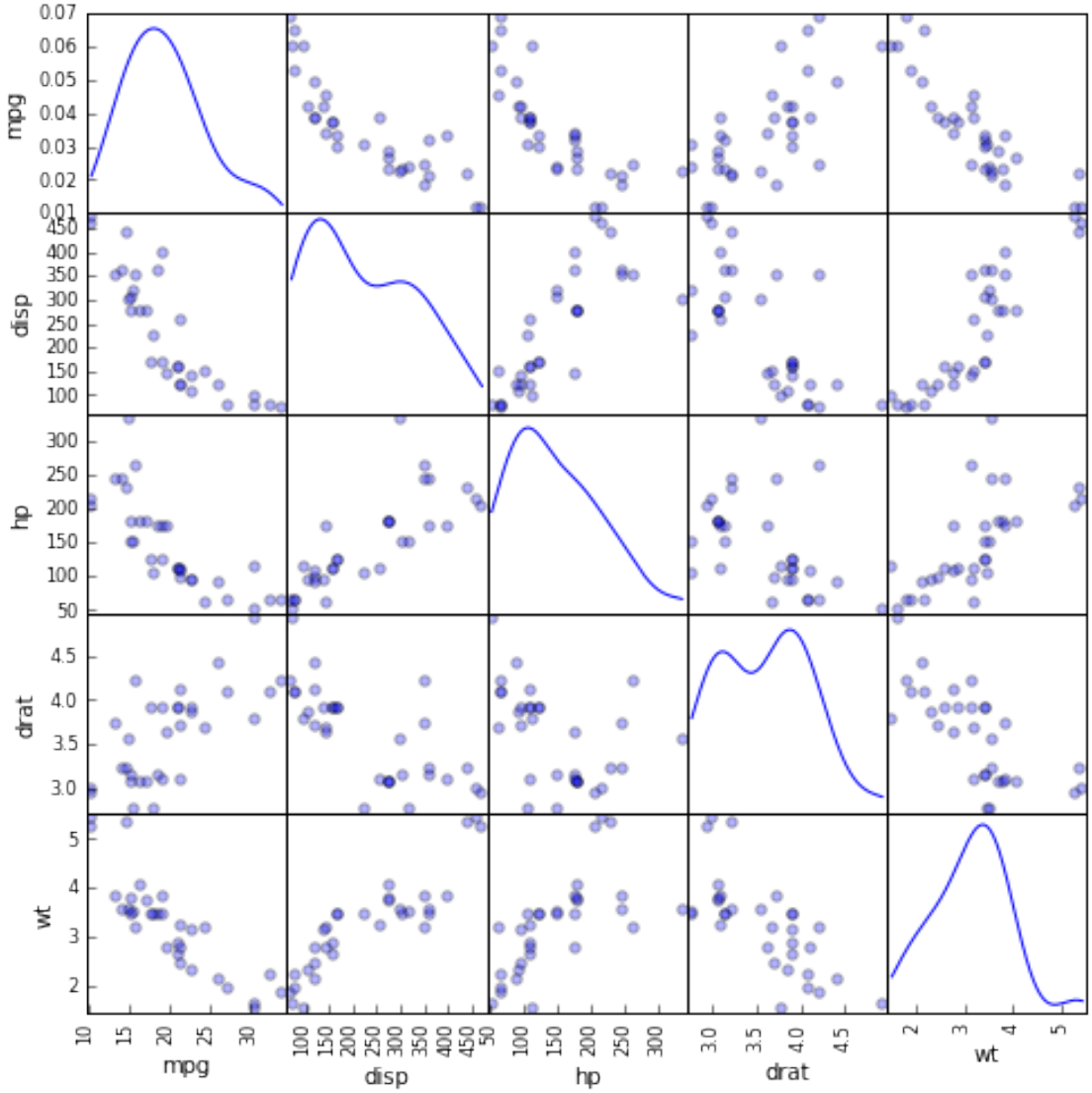
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)

    <ipython-input-23-fd6df88570f6> in <module>()
----> 1 get_ipython().run_cell_magic(u'octave', u'', u'\nA = reshape(1:4,2,2); \nb = [36; 88];\nA\\b'

    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/interactiveshell.pyc in run_cell
   2160             magic_arg_s = self.var_expand(line, stack_depth)
   2161             with self.builtin_trap:
-> 2162                 result = fn(magic_arg_s, cell)
   2163             return result
   2164

    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/extensions/octavemagic.pyc in octave

    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/magic.pyc in <lambda>(f, *a, **
    191     # but it's overkill for just that one bit of state.
    192     def magic_deco(arg):
--> 193         call = lambda f, *a, **k: f(*a, **k)
    194
    195         if callable(arg):

    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/extensions/octavemagic.pyc in octave
    327         except (oct2py.Oct2PyError) as exception:
    328             msg = exception.message
--> 329             msg = msg.split('# ___<end_pre_call>___ #')[1]
    330             msg = msg.split('# ___<start_post_call>___ #')[0]
    331             raise OctaveMagicError('Octave could not complete execution.  '

    IndexError: list index out of range
```

2.2.2 We will redo these examples in Python

```
In [24]: import pandas as pd
         import numpy as np
         import statsmodels.api as sm
         from pandas.tools.plotting import scatter_matrix
```

```
In [25]: # First we will load the mtcars dataset and do a scatterplot matrix
```

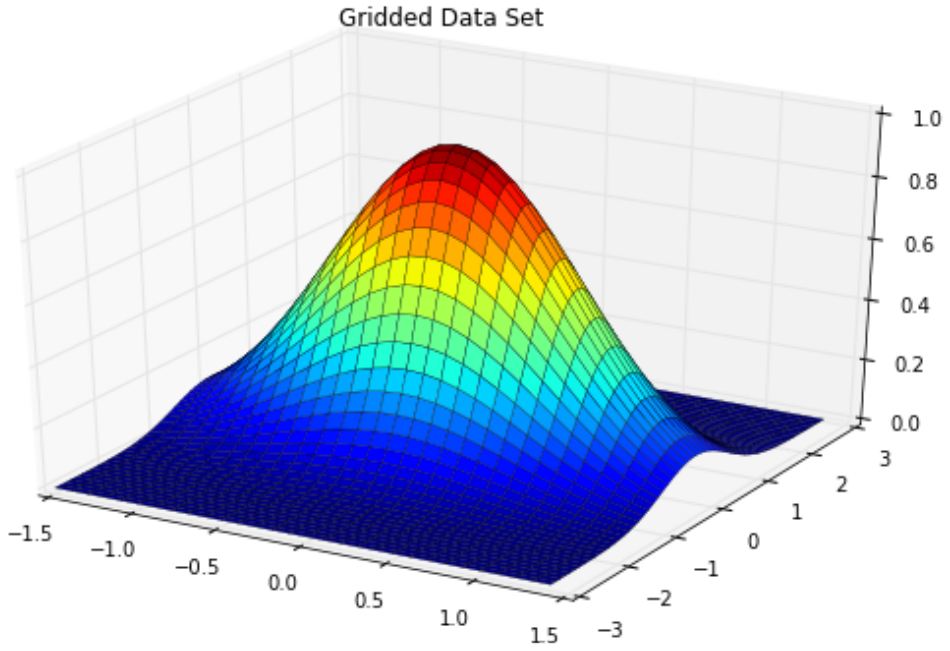
```
mtcars = sm.datasets.get_rdataset('mtcars')
df = pd.DataFrame(mtcars.data)
scatter_matrix(df[[0,2,3,4,5]], alpha=0.3, figsize=(8, 8), diagonal='kde', marker='o');
```



In [26]: *# Next we will do the 3D mesh*

```
xgv = np.arange(-1.5, 1.5, 0.1)
ygv = np.arange(-3, 3, 0.1)
[X,Y] = np.meshgrid(xgv, ygv)
V = np.exp(-(X**2 + Y**2))

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.plot_surface(X, Y, V, rstride=1, cstride=1, cmap=plt.cm.jet, linewidth=0.25)
plt.title('Gridded Data Set');
```



In [27]:
```
# And finally, the matrix manipulations

import scipy

A = np.reshape(np.arange(1, 5), (2,2))
b = np.array([36, 88])
ans = scipy.linalg.solve(A, b)
P, L, U = scipy.linalg.lu(A)
Q, R = scipy.linalg.qr(A)
D, V = scipy.linalg.eig(A)
print 'ans =\n', ans, '\n'
print 'L =\n', L, '\n'
print "U =\n", U, '\n'
print "P = \nPermutation Matrix\n", P, '\n'
print 'Q =\n', Q, '\n'
print "R =\n", R, '\n'
print 'V =\n', V, '\n'
print "D =\nDiagonal matrix\n", np.diag(abs(D)), '\n'
```

```
ans =
[ 16.  10.]

L =
[[ 1.          0.        ]
 [ 0.33333333  1.        ]]
```

```
U =
[[ 3.          4.        ]
 [ 0.          0.66666667]]

P =
Permutation Matrix
[[ 0.  1.]
 [ 1.  0.]]

Q =
[[-0.31622777 -0.9486833 ]
 [-0.9486833   0.31622777]]

R =
[[-3.16227766 -4.42718872]
 [ 0.         -0.63245553]]

V =
[[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]

D =
Diagonal matrix
[[ 0.37228132  0.        ]
 [ 0.          5.37228132]]
```

2.2.3 Using Julia

```
In [30]: %load_ext julia.magic
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)

    <ipython-input-30-5bcfdab8fb0a> in <module>()
----> 1 get_ipython().magic(u'load_ext julia.magic')


    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/interactiveshell.pyc in magic(s
   2203         magic_name, _, magic_arg_s = arg_s.partition(' ')
   2204         magic_name = magic_name.lstrip(prefilter.ESC_MAGIC)
-> 2205         return self.run_line_magic(magic_name, magic_arg_s)
   2206
   2207     #-------------------------------------------------------------------------


    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/interactiveshell.pyc in run_line
   2124                 kwargs['local_ns'] = sys._getframe(stack_depth).f_locals
   2125             with self.builtin_trap:
-> 2126                 result = fn(*args,**kwargs)
   2127             return result
   2128


    /home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/magics/extension.pyc in load_ext
```

```
/home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/magic.pyc in <lambda>(f, *a, **k
    191     # but it's overkill for just that one bit of state.
    192     def magic_deco(arg):
--> 193         call = lambda f, *a, **k: f(*a, **k)
    194
    195         if callable(arg):


/home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/magics/extension.pyc in load_ext
     61         if not module_str:
     62             raise UsageError('Missing module name.')
---> 63         res = self.shell.extension_manager.load_extension(module_str)
     64
     65         if res == 'already loaded':


/home/bitnami/anaconda/lib/python2.7/site-packages/IPython/core/extensions.pyc in load_extensior
     96             if module_str not in sys.modules:
     97                 with prepended_to_syspath(self.ipython_extension_dir):
---> 98                     __import__(module_str)
     99             mod = sys.modules[module_str]
    100             if self._call_load_ipython_extension(mod):


ImportError: No module named julia.magic
```

```
In [29]: %%julia
         1 + sin(3)
```

```
ERROR: Cell magic `%%julia` not found.
```

```
In []: %%julia
       s = 0.0
       for n = 1:2:10000
           s += 1/n - 1/(n+1)
       end
       s # an expression on the last line (if it doesn't end with ";") is printed as "Out"
```

```
In []: %%julia
       f(x) = x + 1
       f([1,1,2,3,5,8])
```

2.2.4 Using Perl

```
In []: %%perl
       use strict;
       use warnings;

       print "Hello World!\n";
```

We hope these give you an idea of the power and flexibility this notebook environment provides!