Jupyter Notebook Demo

May 8, 2017

1 Library Bioinformatics Service

1.1 Jupyter Notebook Tutorial

This tutorial was built in a Jupyter notebook! Various formats of this tutorial can be accessed at https://github.com/oxpeter/library_bioinformatics_service/tree/master/Jupyter

`Created by Peter Oxley for the library bioinformatics service, May 2017`

Installation of Jupyter notebooks is recommended via Anaconda

```
In [1]: # this is a code cell with no output
        a=120
```

```
In [2]: # this is a code cell with output
        # all output to stdout / stderr will be displayed below the cell.
        print(a)
```

```
120
```

2 This is a text cell

It is formatted using **markdown syntax**.

(*to edit a markdown cell, just double click on the text. Don't forget to 'execute' the cell afterwards to implement the formatting*)

Markdown cells within a notebook have a number of advantages: 1. Easy to type 2. Easy to read 3. Great for discussion of code: * Choice of analysis * Choice of parameters * Implications of results * Introduction/methods/conclusions/references...

Cells are switched between code and markdown by using the menu

`Cell > Cell Type > Markdown`

Or by using the dropdown box in the icon bar.

You can even create links!

```
In [3]: import numpy as np
        import pandas as pd
        # notice that this cell doesn't execute when you press enter.
        # Only by pressing shift-enter or alt-enter, or clicking on the 'run' icon.
```

```
In [4]: # this cell does not generate any output to stdout or stderr,
        # so nothing is shown after executing the cell.
        s1 = np.random.normal(0,1,1000)  # generate a random sample with normal distribution (me
        s2 = np.random.normal(2,4,1000)
        df = pd.DataFrame({"s1":s1, "s2":s2})
```

```
In [5]: # this cell outputs to stdout,
        # which is printed immediately following the cell:
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
s1    1000 non-null float64
s2    1000 non-null float64
dtypes: float64(2)
memory usage: 15.7 KB
```

```
In [6]: # table output is formatted to make it easy to view:
        df.T.head()
```

```
Out[6]:          0         1         2         3         4         5         6   \
        s1  0.255287  2.137748  2.840305  0.435784 -0.507710  1.798250  0.376911
        s2  0.054413 -0.065666  3.303618  6.183067  7.700526  9.164256 -1.332347

                 7         8         9      ...          990       991       992  \
        s1 -0.350825  1.154573 -0.483646    ...    -1.166325 -0.677468  0.868495
        s2  2.294891 -3.227784  4.790150    ...     2.076782  4.240831  8.810881

                 993       994       995       996       997       998       999
        s1 -1.701220  0.512454  0.274224  1.088739  1.546190 -0.732312 -0.944138
        s2  0.198935  6.596999  4.334405 -0.724852 -0.378433 -2.635763  7.751945

        [2 rows x 1000 columns]
```

2.0.1 Running code in different languages

The code in this notebook is executed by the designated "kernel" loaded at creation. In this case, the IPython kernel was loaded. All code entered will therefore be interpreted by this kernel and run as python code. However, when the kernel is IPython, you have access to "cell magic" (using the % syntax), where it is possible to have cells run by a different interpreter.

- Using a single % will run the magic on that line only.
- Starting a cell with %% will run the magic on the entire cell.

```
In [7]: %%bash
        # this cell is run in a bash shell created specially for the following code.
        echo "Hello, world"
```

```
Hello, world
```

````
In [8]: # it is also possible to invoke bash commands using the ```!``` syntax:
        !ls -al | head -n 8 | tail -n 2
````

```
-rw-r--r--   1 poxley  staff   12968 May  8 11:14 Jupyter demo handout.ipynb
-rw-r--r--   1 poxley  staff   21257 May  8 12:44 Live Demo!.ipynb
```

```
In [9]: %%html
        <body>
        <h2>This is an html interpreted header</h2>
        <a href="library.med.cornell.edu">This is an html link</a>
        </body>
```

```
<IPython.core.display.HTML object>
```

2.0.2 Sharing variables between languages

It is even possible to capture the variables from each cell/interpreter/language, and pass them into others:

```
In [10]: # capturing the output of the bash ls command:
         directory_contents = !ls -la
         directory_contents
```

```
Out[10]: ['total 848',
          'drwxr-xr-x  16 poxley  staff     544 May  8 21:08 .',
          'drwxr-xr-x   6 poxley  staff     204 May  5 13:10 ..',
          'drwxr-xr-x  12 poxley  staff     408 May  8 12:18 .ipynb_checkpoints',
          '-rw-r--r--   1 poxley  staff   40217 May  7 20:41 First jupyterhub notebook!.ipynb',
          '-rw-r--r--   1 poxley  staff  121899 May  8 21:08 Jupyter Notebook Demo.ipynb',
          '-rw-r--r--   1 poxley  staff   12968 May  8 11:14 Jupyter demo handout.ipynb',
          '-rw-r--r--   1 poxley  staff   21257 May  8 12:44 Live Demo!.ipynb',
          '-rw-r--r--   1 poxley  staff    2074 May  7 23:11 Untitled.ipynb',
          '-rw-r--r--   1 poxley  staff      72 May  7 23:11 Untitled1.ipynb',
          '-rw-r--r--   1 poxley  staff   14768 May  7 23:22 Untitled2.ipynb',
          '-rw-r--r--   1 poxley  staff   11359 May  7 23:43 Untitled3.ipynb',
          '-rw-r--r--   1 poxley  staff      72 May  8 12:13 Untitled4.ipynb',
          '-rw-r--r--   1 poxley  staff   53248 May  5 15:06 jupyterhub.sqlite',
          '-rw-r--r--   1 poxley  staff    1530 May  5 15:04 jupyterhub_config.py',
          '-rw-------   1 poxley  staff    2733 May  5 14:05 jupyterhub_cookie_secret',
          '-rw-r--r--   1 poxley  staff  126788 May  8 20:27 rpy2_setup demo.ipynb']
```

```
In [11]: %%bash -s "$a"
         # The above line puts the variable a into the bash shell as a positional parameter.
         # Be aware of any characters (eg. quotation marks) in the python variable -
         # these will need to be escaped before being passed to the bash cell.
         echo $1
```

```
120
```

```
In [12]: # an alternative to send variables into bash:
         !echo {a * 2}
```

```
240
```

```
In [13]: # R requires a few extra steps to access
         # rpy2 provides access to R from within Python
         # (you can read more here: http://rpy2.readthedocs.io)
         # after installing rpy2 - we load the extension into the kernel:
         %load_ext rpy2.ipython

         # now we can access the installed version of R
         iris_dataset = %R iris
```

```
In [14]: iris_dataset.describe()
```

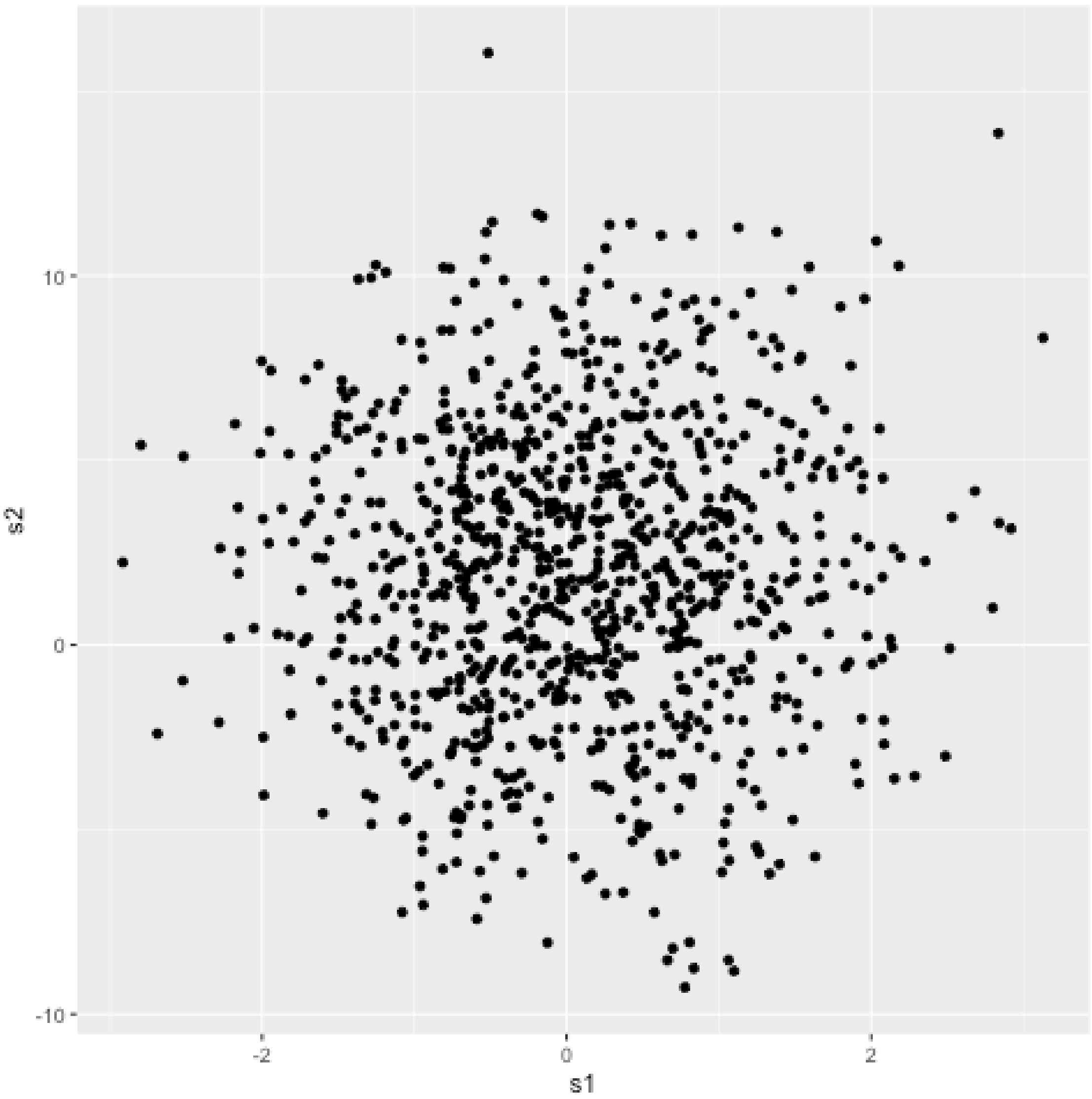
```
Out[14]:        Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
         count    150.000000   150.000000    150.000000   150.000000
         mean       5.843333     3.057333      3.758000     1.199333
         std        0.828066     0.435866      1.765298     0.762238
         min        4.300000     2.000000      1.000000     0.100000
         25%        5.100000     2.800000      1.600000     0.300000
         50%        5.800000     3.000000      4.350000     1.300000
         75%        6.400000     3.300000      5.100000     1.800000
         max        7.900000     4.400000      6.900000     2.500000
```

```
In [15]: %%R -i df
         # the above line sets R as the interpreter for this cell,
         # and imports the variable df (it will be referenced in this cell using the same name)

         # Now we can manipulate and graph the dataframe using R functions:
         require(ggplot2)
         ggplot(data=df) + geom_point(aes(x=s1, y=s2))
```

```
/Users/poxley/anaconda/envs/rpy2_setup/lib/python3.5/site-packages/rpy2/rinterface/__init__.py:1

  warnings.warn(x, RRuntimeWarning)
```



2.0.3 Other useful IPython cell magic

IPython magics don’t only let you use other language interpreters.

```
In [16]: # change the current working directory
         %cd jupyterhub/
```

```
[Errno 2] No such file or directory: 'jupyterhub/'
/Users/poxley/Documents/7. Technology/Bioinformatics/workshops/jupyterhub
```

```
In [17]: # list the variables currently available to the kernel
         %who
```

```
a          df          directory_contents         iris_dataset         np          pd          s1
```

```
In [18]: # list the variables and their string representation
         %whos
```

```
Variable             Type         Data/Info
-------------------------------------------
a                    int          120
df                   DataFrame               s1         s2\<...>\n[1000 rows x 2 columns]
directory_contents   SList        ['total 848', 'drwxr-xr-x<...>7 rpy2_setup demo.ipynb']
iris_dataset         DataFrame         Sepal.Length  Sepal.<...>n\n[150 rows x 5 columns]
np                   module       <module 'numpy' from '/Us<...>kages/numpy/__init__.py'>
pd                   module       <module 'pandas' from '/U<...>ages/pandas/__init__.py'>
s1                   ndarray      1000: 1000 elems, type `float64`, 8000 bytes
s2                   ndarray      1000: 1000 elems, type `float64`, 8000 bytes
```

```
In [19]: %%time
         for i in range(10):
             !sleep 1
```

```
CPU times: user 232 ms, sys: 147 ms, total: 379 ms
Wall time: 11.2 s
```

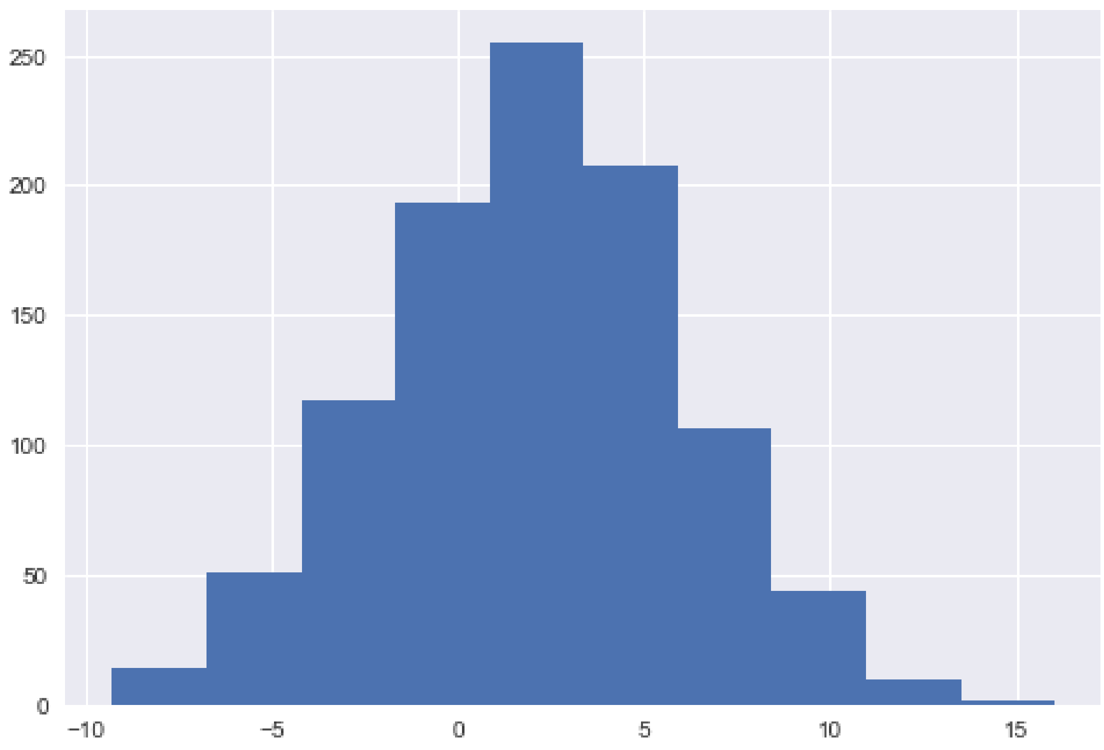
```
In [20]: %%timeit
         np.random.normal(0,1,1000).sum()
```

```
37.6 ţs ś 336 ns per loop (mean ś std. dev. of 7 runs, 10000 loops each)
```

```
In [21]: # to capture plot output and display it inline:
         %matplotlib inline

         import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [22]: df['s2'].hist();
         plt.show()
```



```
In [23]: # using the question mark will bring up any help documentation
         ?pd.DataFrame
```

2.1 Other functions of Jupyter notebooks

Tab completion Works for variables, modules, functions, function parameters, and cell magics.

Notebook extensions

- **nbpresent** from Anaconda will help you convert the notebook into interactive powerpoint-style presentations
- **nbextensions** provides access to a host of different extensions. Instructions for installing this extension can be found here

MathJax and Latex support The markdown box is MathJax aware, so you can do cool things such as:

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

Latex can also be leveraged to export notebooks to pdf

Export notebook to other files Including .pdf (using Latex), .html, .py, .rst, and .md. Use `File > Download as > ...`

when all else fails...

```
In [ ]:
```