Installing Python - Jupyter Notebook

November 20, 2017

0.1 What is Jupyter?

Jupyter allows you to create and share documents that contain code, results, graphs, explanatory text and equations. The project started out at as IPython Notebook (you will find a lot of useful online resources under this name), but has grown to support not only Python, but over 40 other programming languages. The name "Jupyter" is a combination of Julia, Python, and R, and a full list of available kernels can be found on the Jupyter documentation page.

Working with Jupyter notebooks is extremely easy: you create a new notebook and edit it in your browser. A Python (or R, Julia, etc.) kernel evaluates the code you type in notebook "cells" and displays the result below the cell. The resulting file can be exported as HTML, Markdown, LaTeX and PDF. The notebook itself is stored in JSON format. Notebooks also render nicely on GitHub.

0.2 Installing Jupyter

You need to have Python installed (version 3.3 or greater, or 2.7) to be able to install Jupyter. #### A. If you don't have Python installed: For new Python users, it is highly recommended that you use Anaconda, which installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

1. Download Anaconda. Choose your operating system. You can choose whether to install Python 3.5 or 2.7. Python 3 is the future of Python, but it is not backwords compatible. Still, most useful pachages have now been updated to work with Python 3, so choose this version. You can always go back to conda and install another environment for Python 2 if you need it.

2. Install the version of Anaconda which you downloaded, following the instructions on the download page:

Important instructions: 1. On windows: Do NOT install into paths that include spaces. If possible, install directly in C: When asked, choose to NOT add to path. 2. On Mac: When asked, DO choose to add to path.

While the installer runs, listen to me speaking about important things.

3. You now have Python and Jupyter notebook installed. You can start the notebook server from Anaconda, or from the command line (using Terminal on Mac/Linux, Command Prompt on Windows) by running: jupyter notebook
4. To update packages and install Python packages which were not included in Anaconda, you can use *conda*. To update `conda`: conda update conda To see what packages are installed: conda list To update packages: conda update --all To install a package: conda install rpy2

B. If you have used Python before:

1. You can install Jupyter using *pip*: pip3 install --upgrade pip pip3 install jupyter
2. You now Jupyter Notebook installed. You can start the notebook server from the command line (using Terminal on Mac/Linux, Command Prompt on Windows) by running: jupyter notebook
3. You can also update Jypyter using `pip` (and install other packages): To update Jupyter: pip install --update jupyter

0.3 Running the notebook

You can start the notebook from the command line:

```
jupyter notebook
```

The notebook will open in your browser. You can also specify which browser to use:

```
jupyter notebook --browser=chrome
```

By default, the notebook server starts on port 8888, or, if this port is not available, to the next available one, such as 9999. You may also specify a port manually:

```
jupyter notebook --port 9999
```

When you are done, you can kill Jupyter by hitting "Ctrl + C" in the terminal window.

0.4 Notebook structure

You can launch Jupyter from any location in your computer. By default, it will launch in your current working directory:

You can navigate through the files in your computer to find the one you want to edit, and click the "Upload" button in the upper right corner. You can also click the "New" button to create a new file. You will be asked to select the type of file you want to open from the dropdown menu. For example, to open a new notebook using a Python 2 kernet choose "Python 2" in the notebooks section. This will open a new notebook in a new window (or tab) in your browser.

A Jupyter notebook consists of cells. You can specify which type of cell you want to use in the drop-down menu on top of the page, or using keyboard shortcuts. The two main types of cells that we're using are code cells and markdown cells. You can turn a cell into a text cell by hitting "Esc, y" and into a markdown cell by hitting "Esc, m".

You have to press Enter or double-click on a cell to enter editing mode. You can add new cells by hitting he **+** button top of the notebook, and you can move cells up and down to change their order using the arrows. You can also use shortcuts: "Esc, b" to insert a cell below and "Esc, a" to insert a cell above. Here is a list of useful Jupyter shortcuts.

0.5 Code cells

Code cells contain the code you want to run. To execute the code in a cell, hit "Shift+Enter". This executes the current cell and moves on to the next one. You can also use the menu button for this, by pressing the "play" button on the toolbar or select "Run" from the "Cell" menu above. Another way to run the current cell if by hitting "Ctrl+Enter", which runs thew current cell but does not select the next one.

Here is an example of some simple code in a cell:

```
In [2]: a=0
        for i in range(5):
            a+=i
            print("i=", i, "a=", a)
```

```
i= 0 a= 0
i= 1 a= 1
i= 2 a= 3
i= 3 a= 6
i= 4 a= 10
```

You can edit the code and run it again until you get the desired result.

Results generated by running the code in a cell are available in all other cells, including the ones above, if you execute them again. You can tell the order in which cells were executed by looking at the counter in front of them, "In[]".

```
In [3]: print(a)
```

```
10
```

Graphs can be displayed **inline**, directly in the notebook. You can specify this option by using the `%matplotlib inline` or `%pylab inline` magic functions.

```
In [4]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

This will make all subsequently ran cells display graphs inline.

```
In [5]: import pylab as plb
        import numpy as np
        from scipy.stats import norm
        x = np.linspace(-5,5,1000)
        y = norm.pdf(x)     # for example
        plb.plot(x,y)
        plb.title('Normal distribution')
        plb.show()
```