

# PYTHON FOR OCEAN SCIENCES

INSTALLATION, SET UP & PACKAGE MANAGEMENT  
USING ANACONDA



Nia Jones  
[niajones@bangor.ac.uk](mailto:niajones@bangor.ac.uk)

# Why?

- Appetite for learning Python at SOS
- Create a Python community to facilitate learning, support and using the language for research.
- Building on previous meetings in SOS





# WHY PYTHON?



- Proprietary, **closed source** software (\$\$\$)
- Interpreted language
- Popular in **academic, scientific and engineering** circles
- Strong at **signal and image processing**



- Free and **open-source**
- Large **community support**
- Interpreted language
- Uses libraries
- General purpose
- Strong for **Big Data**
- Simple syntax



- Free and **open-source**
- Uses libraries/packages
- Slower Runtime
- Strong in **statistical analysis**
- Strong in **visualisation**
- Complex syntax



Paul Romer

## *Jupyter, Mathematica, and the Future of the Research Paper*

*Fri, Apr 13, 2018*

t Newsletters

*The Atlantic*

SCIENCE

## THE SCIENTIFIC PAPER IS OBSOLETE

Here's what's next.

By James Somers

APRIL 5, 2018

SHARE ▼

“It’s incalculable, literally ... how much is **lost**, and how much time is **wasted**, and how many results are **misinterpreted** or are **misrepresented**.”

**Theodore Gray**, co-founder of Wolfram Research, on the cons of a traditional research paper compared to a ‘*computational essay*’.

# (AN) ARCHITECTURE FOR DATA SCIENCE USING PYTHON

- Python is the programming language



- Anaconda is a 'distribution' for data science specific tools.



This is all you need  
to download onto  
your computer.

- Conda is the package and environment manager within Anaconda.





## Environments

Use **CONDA** to  
create environments

Environment\_1

e.g PhD Chapter 1

Environment\_2

e.g Paper X

Environment\_3

e.g Paper Y

## Packages

Use **CONDA** to  
install packages

Python 2.0

numpy

scipy

Python 3.0

pandas

scipy

Python 3.0

numpy



# DOWNLOADING ANACONDA

Download the graphical installer here: <https://www.anaconda.com/products/distribution>

Windows Documentation: <https://docs.anaconda.com/anaconda/install/windows/>

MacOS Documentation: <https://docs.anaconda.com/anaconda/install/mac-os/>

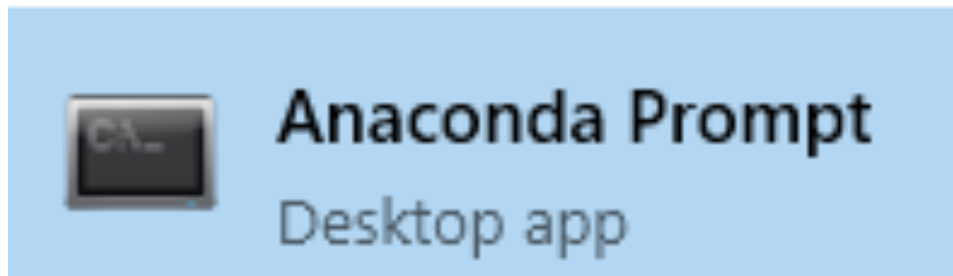
\* You need **minimum 5 GB disk space** to download and install Anaconda. A smaller distribution, '**Miniconda**', is available but you'll have to do a lot more of the set up yourself.



# CONDA – Access through the command line

## Windows

On windows, 'Anaconda Prompt' should have been installed within your Anaconda Distribution. This is how you access 'conda'.



## Mac

On a mac, navigate to the 'Terminal App'. This is how you access 'conda'.



# CONDA – Access through the command line

To verify your conda is working type the following into the command line:

```
conda --version    # Conda displays the number of the version installed.
```

Now let's make our own 'environment' called 'python\_tutorials'

```
conda create --name python_tutorials    # Create new environment
```

Get a list of all your python environments printed to the command window

```
conda info --envs    # List the environments
```

# CONDA – Access through the command line

To start working within your environment you first have to 'activate' it.

```
conda activate python_tutorials # Activate chosen environment
```

In Python, there are different packages which expand Python's basic capabilities. Think of it like your data science toolbox. To list these packages type in the following:

```
conda list # list all of the packages within the activated environment
```

Oh no! There are no tools in the toolbox. We'll have to download and install some packages. But first, a few words on python packages and why conda is so valuable...

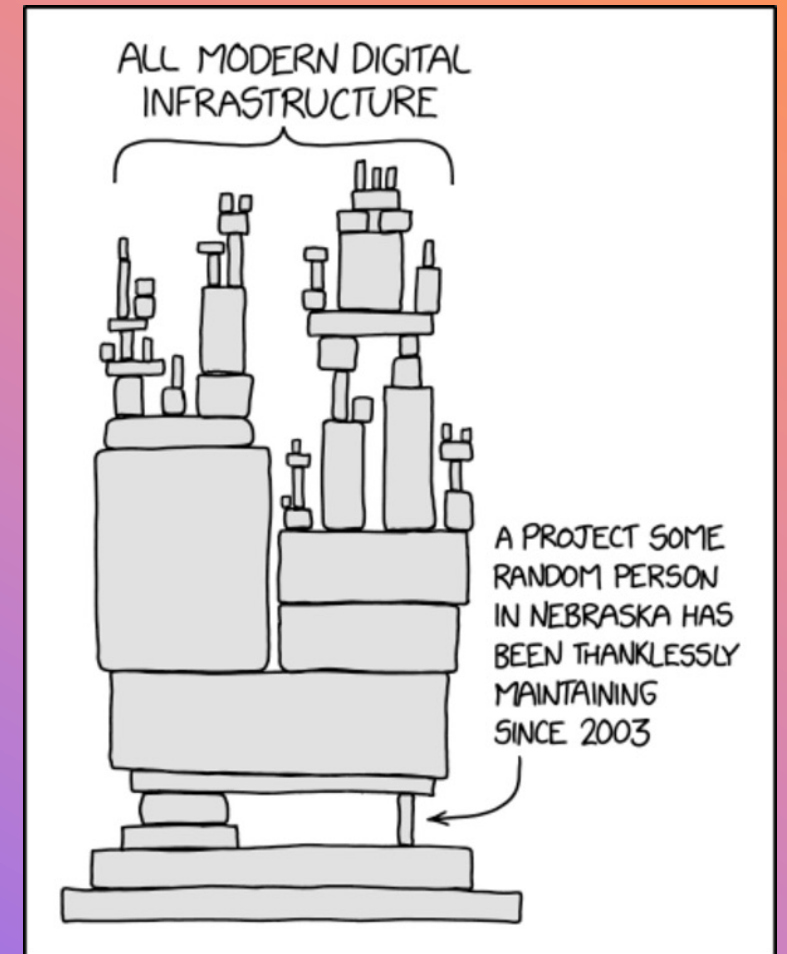
# PYTHON PACKAGES AND THE NINE CIRCLES OF DEPENDENCY HELL.

## Dependency hell

*a colloquial term for the frustration of some software users who have installed software packages which have dependencies on specific versions of other software packages.*

Python is largely 'package' driven. Packages depend on other packages which depend on other packages *ad infinitum*. If one of those packages changes, or isn't accounted for in an update, the whole system could break.

To overcome this, conda has a dependency resolver which identifies additional packages which need installing/updating. So **congratulations**, you may have (mostly) avoided dependency hell.





# PYTHON PACKAGES AND THE NINE CIRCLES OF DEPENDENCY HELL.

Some popular python packages for data science...

<b>NumPy</b>	Array based data
<b>Pandas</b>	Labelled and heterogeneous data
<b>SciPy</b>	Scientific computing
<b>Matplotlib</b>	Visualisations
<b>Scikit Learn</b>	Machine Learning

# CONDA – Access through the command line

Install some of these packages to your activated environment:

```
conda install numpy pandas scipy matplotlib # install 4 different packages to  
the activated environment.
```

Now when you list your packages you should see a list of the ones you've just installed.

```
conda list # list all of the packages within the activated environment
```

# ANACONDA NAVIGATOR – Access through a GUI

Manage packages  
and environments

The screenshot displays the Anaconda Navigator application window. The left sidebar contains navigation links: Home, Environments (highlighted with a purple box), Learning, and Community. The main panel shows a grid of application tiles. At the top of this panel, there are filters: 'All applications' (dropdown), 'on' (dropdown), 'base (root)' (dropdown, highlighted with a pink box), and 'Channels' (button). The application tiles include:

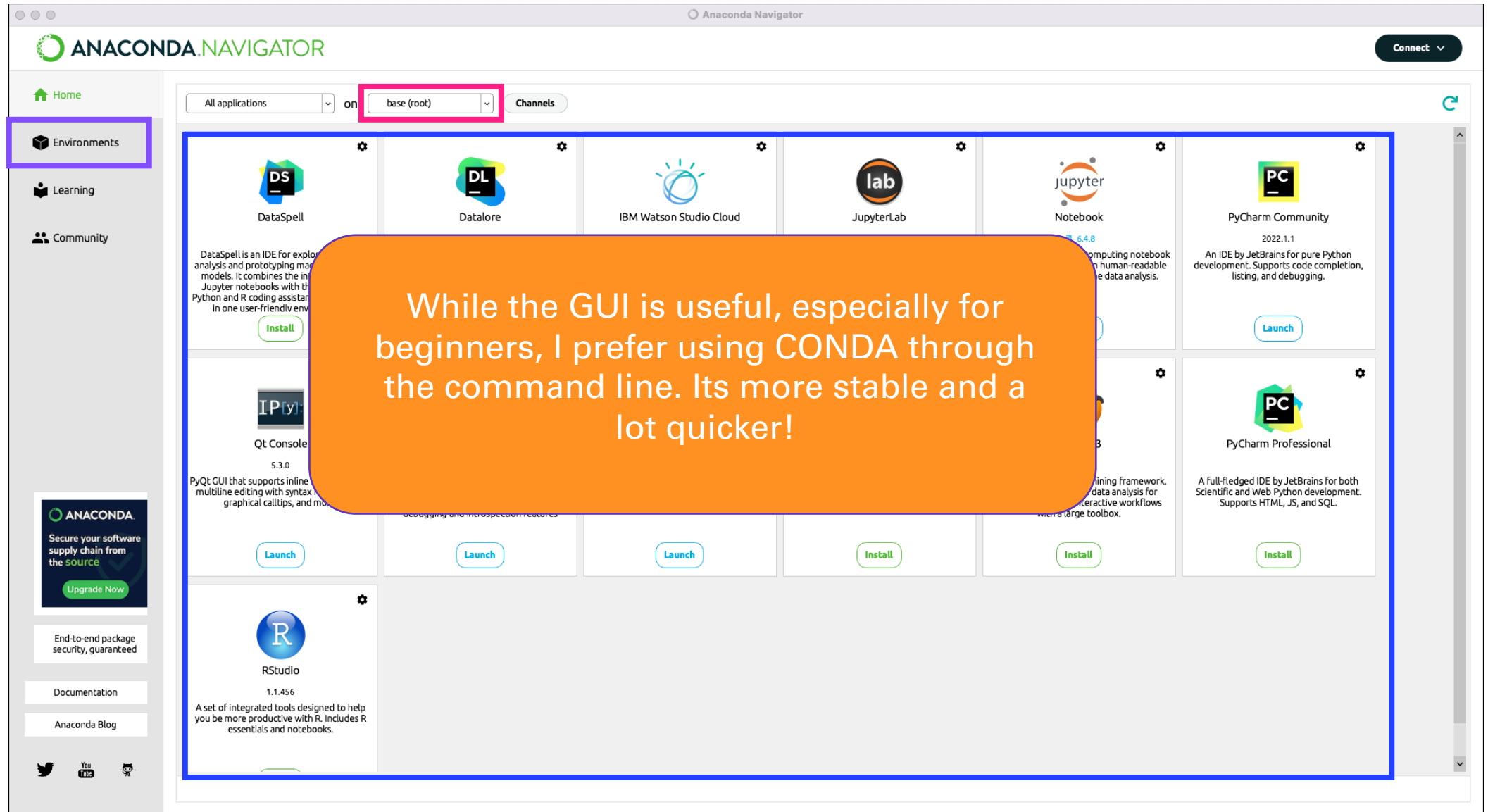
- DataSpell: IDE for exploratory data analysis and prototyping machine learning models. [Install]
- Datalore: Online Data Analysis Tool with smart coding assistance by JetBrains. [Launch]
- IBM Watson Studio Cloud: Tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. [Launch]
- JupyterLab 3.3.2: An extensible environment for interactive and reproducible computing. [Launch]
- Notebook 6.4.8: Web-based, interactive computing notebook environment. [Launch]
- PyCharm Community 2022.1.1: An IDE by JetBrains for pure Python development. [Launch]
- Qt Console 5.3.0: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Launch]
- Spyder 5.1.5: Scientific Python Development Environment. [Launch]
- VS Code 1.68.1: Streamlined code editor with support for development operations like debugging, task running and version control. [Launch]
- Glueviz 1.0.0: Multidimensional data visualization across files. [Install]
- Orange 3 3.26.0: Component based data mining framework. [Install]
- PyCharm Professional: A full-fledged IDE by JetBrains for both Scientific and Web Python development. [Install]
- RStudio 1.1.456: A set of integrated tools designed to help you be more productive with R. [Launch]

At the bottom left, there is an 'ANACONDA' banner with the text 'Secure your software supply chain from the source' and an 'Upgrade Now' button. Below this are links for 'End-to-end package security, guaranteed', 'Documentation', and 'Anaconda Blog'. Social media icons for Twitter, YouTube, and GitHub are at the bottom.

Activate different environments

Install and launch different programmes to begin coding in Python

# ANACONDA NAVIGATOR – Access through a GUI





## \*Actually\* accessing and using Python.

Now you hopefully grasp the basics of what Anaconda is and have it installed on your computer, you now need something to be able to **write Python code in a meaningful way**.

There are many different software packages, or **Integrate Development Environments (IDEs)**, that allow you to use Python. I'll be concentrating on two of them in these sessions: **Spyder** and **Jupyter Notebooks**.



To be able to use these, type the following two lines of code into your activated environment in your Anaconda Prompt/Terminal. You can also do this through Anaconda Navigator if you want to. (I recommend installing both for now)

```
conda install jupyter notebook  
conda install spyder
```

\*Actually\* accessing and using Python.



- More of a traditional coding environment
- **Familiar** layout
- .py files
- Strong for **exploratory data analysis** and **debugging**.



- **Notebook** environment
- Streamlined experience
- .ipynb files
- **Sharing** friendly
- Useful for **visualisation**

ianitagomez/Local/Dev-Spyder/spyder/spyder/plugins/plots

/Users/juanitagomez/Local/Dev-Spyder/spyder/spyder/plugins/plots/plugin.py

plugin.py - plots | chart\_plot\_example.py | plugin.py - ipythonconsole

```

1  # -*- coding: utf-8 -*-
2  #
3  # Copyright © Spyder Project Contributors
4  # Licensed under the terms of the MIT License
5  # (see spyder/_init_.py for details)
6
7  """
8  Plots Plugin.
9  """
10
11 # Third party imports
12 from qtpy.QtCore import Signal
13
14 # Local imports
15 from spyder.api.plugins import Plugins, SpyderDockablePlugin
16 from spyder.api.translations import get_translation
17 from spyder.plugins.plots.widgets.main_widget import PlotsWidget
18
19 # Localization
20 _ = get_translation('spyder')
21
22 class Plots(SpyderDockablePlugin):
23     """
24     Plots plugin.
25     """
26     NAME = 'plots'
27     REQUIRES = [Plugins.IPythonConsole]
28     TABIFY = [Plugins.VariableExplorer, Plugins.Help]
29     WIDGET_CLASS = PlotsWidget
30     CONF_SECTION = NAME
31     CONF_FILE = False
32     DISABLE_ACTIONS_WHEN_HIDDEN = False
33
34     # --- SpyderDockablePlugin API
35
36     def get_name(self):
37         return _('Plots')
38
39     def get_description(self):
40         return _('Display, explore and save console generated plots.')
41
42     def get_icon(self):
43         return self.create_icon('hist')
44
45     def register(self):
46         # Plugins
47         ipyconsole = self.get_plugin(Plugins.IPythonConsole)
48
49         # Signals
50         ipyconsole.sig_shellwidget_changed.connect(self.set_shellwidget)
51         ipyconsole.sig_shellwidget_process_started.connect(
52             self.add_shellwidget)
53         ipyconsole.sig_shellwidget_process_finished.connect(
54             self.remove_shellwidget)

```

Name	Type	Size	Value
bool	bool	1	True
data	Array of str128	(3, 3)	ndarray object of numpy module
datetime_object	datetime	1	2021-04-14 17:35:14.687085
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	53	/Users/Documents/spyder/spyder/tests, test_dont_use.py
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
myset	set	3	{'2', '1', '3'}
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylst	list	2	[123, 'efgh']
x	float64	1	1.1235123099439

Help | Variable Explorer | Files | Code Analysis

0 %

Plots | IPython console | History

conda: spyder\_dev (Python 3.8.5) | LSP Python: ready | master | Line 10, Col 1 | UTF-8 | LF | RW | Mem 64%

## PyCon 2018: Using pandas for Better (and Worse) Data Science

GitHub: <https://github.com/justmarkham/pycon-2018-tutorial>

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
pd.__version__
```

Out[1]: '0.24.1'

### Dataset: Stanford Open Policing Project ([video](#))

```
In [2]: # ri stands for Rhode Island
ri = pd.read_csv('police.csv')
```

```
In [3]: # what does each row represent?
ri.head()
```

Out[3]:

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	violation_raw	violation	search_
0	2005-01-02	01:55	NaN	M	1985.0	20.0	White	Speeding	Speeding	
1	2005-01-18	08:15	NaN	M	1965.0	40.0	White	Speeding	Speeding	
2	2005-01-23	23:15	NaN	M	1972.0	33.0	White	Speeding	Speeding	
3	2005-02-20	17:15	NaN	M	1986.0	19.0	White	Call for Service	Other	
4	2005-02-20	17:15	NaN	M	1986.0	19.0	White	Call for Service	Other	



+



○



# NEXT SEMINAR:

## JANUARY 23<sup>RD</sup>

### THE BASICS AND IMPORTING DATA

