Mitch Negus
9/14/2017

IPython: A System for Interactive Scientific Computing

Pérez, F. and Granger, B.

Python: An Ecosystem for Scientific Computing

Pérez, F., Granger, B., and Hunter, J.

Jupyter Notebooks–a publishing format for reproducible computational workflows

The Jupyter Development Team, *et al.*

Summary

This week's readings focus on three evolutions of Python and their merits as tools for scientific computation. First, the Python language itself is presented as an invaluable tool for scientific computing. With its tremendous versatility, it can be optimized for a variety of applications (fast calculations, symbolic manipulations, data visualization), while also serving as an intuitive scripting language. Moreover, it is open source, which allow it to fit the scientific ideals of transparency and reproducibility. Another article discusses the IPython project and its features to further improve Python's flexibility; it provides a more robust interactive working environment. These interactive characteristics,, such as access to previous state histories, command line interfacing, and parallelization capabilities, are tremendously useful for scientific workflows. Finally, the third article covers Jupyter notebooks, a more recent development built off the IPython project. These notebooks extend Python's use in the scientific computing community even further. Jupyter notebooks allow prose to be interspersed with code, giving authors better tools to reach users or anyone reviewing the published code. At the same time, Jupyter notebooks are designed on JSON formats and can be displayed in HTML, allowing them to be run both locally and remotely through a web browser.

Exploration

I had several reactions while reading this week's articles. First, I was interested to read a more accessible history and broad overview of the Python language. The presentation in the 2011 article was a distinctly different perspective than I have encountered as a Python user reading documentation. While reading the article on IPython, I also became curious about how many new features IPython has that keep it more useful for interactivity than Python by itself. Some of the features mentioned, such as tab completion and access to previously saved states, now seem to be incorporated into the prepackaged Python interpreter. I imagine that magic commands and parallelization capabilities may still be unique to IPython, but I am curious to know if more distinguishing features have been added in the decade since the IPython article's publishing.

Notes

——Python——
-can function standalone
-its value becomes apparent as a wrapper for other codes
-allows easy interface with web/databases
-emerged as historical trend of excess labor compared to compute resources shifted
-python is ideal for scientific collaboration as it also adheres to free, OS ideals

STRUCTURE
-python is a general purpose language
-rich variety of types
-clear and concise syntax
-diverse third-party libraries afford versatility to optimize
-open source means that though portability may suffer, it is not insurmountable (python is free, just run mult. versions)

——IPython——
-interactivity*
-tab completion*
-access to previously saved states*
-magic commands
-parallelization
*denotes that this feature now also seems to be incorporated into standard python console

——Jupyter——
-notebooks allow code to be interspersed with prose (more sophisticated documentation; key for scientific reproducibility) -jupyter notebooks accessed through web browser; can be hosted either locally or remotely -nbconvert allows static versions to be created -nbviewer allows web hosting -Binder enables live sharing, virtual environment -prominent users (LIGO, biologists, geologists, comp. scientists)