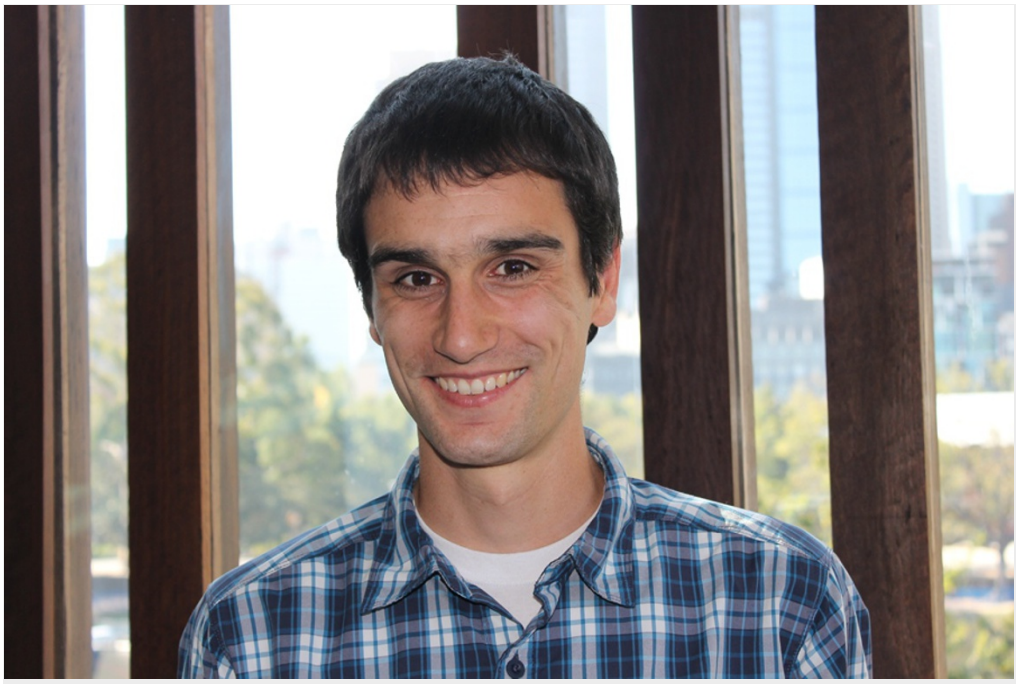
NATURE | TOOLBOX: Q&A

My digital toolbox: Climate scientist Damien Irving on Python libraries

Preset software packages in Python allow for custom scientific analysis.

Richard Van Noorden

30 January 2015



Damien Irving uses a variety of software packages to analyse and report on climate data.

Damien Irving, a weather and climate scientist at the University of Melbourne in Australia, discusses the software and tools that he finds most useful in his research.

How would you describe your research?

As a climate scientist, I'm currently studying the influence of the winds in the upper atmosphere on extreme weather in the Southern Hemisphere. Because I am interested in scientific computing and open science, I am also an instructor with Software Carpentry, write a blog about research skills in the weather and climate sciences, and work part time on an initiative called the Research Bazaar, which aims to help researchers to improve their digital research skills.

I'm currently rushing around in preparation for the Research Bazaar conference in Melbourne (16–18 February). We will be training researchers in digital research tools for data analysis, mapping, computer-aided design and document preparation. There is huge demand.

Which software and online tools do you use regularly, and why?

For simple data analysis, weather and climate researchers have almost universally settled on Network Common Data Form (netCDF) as a default file format to describe data — air temperatures, wind measurements and so on. Because this is so widespread, a collection of programs has been developed that use the command line to manipulate netCDF files. Known as the netCDF Operators and Climate Data Operators, they allow for simple tasks such as editing a file's metadata, and for basic data analysis, such as calculating indices of extreme weather (for example, the duration of a heat wave, or rainfall totals).

For more complex analysis, I use the programming language Python. Researchers at the Program for Climate Model Diagnosis and Intercomparison (PCMDI) at Lawrence Livermore National Laboratory in California have developed a collection of libraries for Python called the Climate Data Analysis Tools (CDAT), which build on Python's generic data-analysis libraries. These software packages come with convenient functions for calculating climatologies (the average weather conditions for a given month or season) and other statistical quantities commonly used in weather and climate science.

I use a tool that the PCMDI (and others) has built on top of CDAT called UV-CDAT (the UV stands for Ultrascale Visualization) for visualizing files quickly, but for creating publication-quality figures I use the Python libraries Iris and Cartopy. These were created by a team at the UK Met Office to add some great functionality to Python's generic plotting libraries. They are specific for weather and climate science, and are good for plotting data on different map projections, for instance.

It used to be difficult to install custom libraries into Python — especially the other pieces of software that each library depends on to function. But Continuum Analytics in Austin, Texas, have released Anaconda, a free package that bundles together around 200 of the most popular Python libraries for science, maths, engineering and data analysis. For installing additional libraries, they have developed a package manager called Conda. I used Conda to install Iris, Cartopy and CDAT on my own machine. If you want to see more about the software and tools that I use on a regular basis, I've listed the full collection in this blogpost.

Visit the Toolbox hub for more articles

Which emerging tools do you have your eyes on?

I've either just started using or would like to use the following tools (listed in order of my level of excitement):

Pandas and xray

There's been a lot of buzz in the past couple of years around a Python library called Pandas. It's used for working with tabular — that is, two dimensional — data. One of the major keys to its popularity is that fact that you can index (that is, reference, locate or specify) values in your data array according to their row and column label, rather than by a numeric index (which is how things traditionally work in programming). Stephan Hoyer and others at the Climate Corporation (a climate technology and insurance company in San Francisco, California that works with weather data) recognized that it would be useful to be able to do this with higher-dimensional data — so one could, for instance, index climate model data according to actual latitude, longitude and time values. They have come up with a library, called xray, which aims to be the Pandas of N-dimensional data. I can't wait to try it.

Authorea

Many people in the weather and climate sciences use LaTeX to prepare documents, because of the ease with which it handles mathematical equations. Online LaTeX editors such as writeLaTeX and ShareLaTeX have been around for quite a while, but the end result when using these editors is a static PDF. Authorea is an online LaTeX editor that is really shaking things up. You can export a PDF (or Word) document if you like, but the primary end product is essentially a blog post written in LaTeX. In other words, your academic paper is a living, breathing web document that people can comment and collaborate on. I'm writing my next journal paper with Authorea, and so far I'm enjoying the experience. (For more on Authorea and other document-writing tools, see 'Scientific writing: the online cooperative').

Julia

Like many scientific Python programmers, I'm keeping a close eye on a new language called Julia. High-level programming languages such as Python, Interactive Data Language and MATLAB are great because they're easy for people to read (which speeds up the code development process), but the trade-off is that they run much slower than languages such as C or Fortran. Julia offers the best of both worlds: it is easy to read and fast. In the long run, the success of this language for scientists probably depends on how important the speed is. Existing languages such as Python have large user and developer communities, which means there are lots of useful add-on libraries out there (including CDAT, Iris and Cartopy). A great deal of effort would be required to move this over to Julia.

Would you recommend any websites, training courses or books for learning about scientific software?

My recommendation would be to attend a Software Carpentry workshop. If there isn't one coming up soon in your region (check this page for upcoming workshops), e-mail Software Carpentry and tee up a workshop up for your home department, institution or conference. In the meantime, the online Software Carpentry lesson notes are great, and I've also put together an orientation guide for weather and climate scientists.

Nature | doi:10.1038/nature.2015.16805