Goldschmidt 2023 Abstract
https://doi.org/10.7185/gold2023.16190

Building ecosystems of research software: lessons learned from geochronology

PROF. PIETER VERMEESCH

University College London

Presenting Author: p.vermeesch@ucl.ac.uk

Openness and transparency are the key ingredients of reproducible science. Unfortunately, in the Earth Sciences, transparency is impeded by the prevalence of proprietary and closed research software. For example, in isotope geochemistry, mass spectrometer data is generally processed with commercial point-and-click software that does not preserve a paper trail of the underlying calculations. Fortunately, a growing community of computing-minded earth scientists addresses this issue, using collaborative software development platforms such as GitHub. The author's contributions in this community include IsoplotR, Provenance, Radial/DensityPlotter and Simplex. These tools were designed to:

1. Adhere to the "Unix philosophy", which posits that software should be simple, compact, clear, modular, and extensible. Instead of building one large program that aims to do everything, it is better to write several small programs that work together.
2. Avoid unnecessary dependencies. In scientific programming languages such as Python or R, it is common for packages to call other packages, which depend on yet another set of packages and so on. When some of these dependencies violate the Unix philosophy, it is possible that a small package depends on a large one. Thus it is not uncommon for packages to install gigabytes of code to do something inherently simple. This is known as "dependency hell". Dependency hell produces fragile software that is not future proof. To avoid this situation, it is sometimes advisable to reinvent the wheel and re-implement a function from scratch instead of loading a package.
3. Develop an Application Programming Interface (API). Even though most users of geoscience software may prefer to use a Graphical User Interface (GUI), it is important to also cater to "power users" who prefer to use the command line. The latter group may only account for 5% of the users, but they push the field forward, so it is important to be friendly to them.
4. Work with open and human-readable data exchange formats such as JSON or XML. This way, it is possible to build a software 'ecosystem', in which lots of developers can work together whilst using different programming languages.