A New Funding Strategy for Scientific Software and an Experiment

Breck Baldwin, Safety 3rd, breckbaldwin@gmail.com

Challenge

The depth, breadth, and impact of open source scientific software exceeded any expectations that one might have had at the start of the millennium. We enjoy high-quality, largely free, state-of-the-art technology all of which came to be due to a nearly magical confluence of cultural changes in software developers, backed by huge technology companies and the development of open source behavioral conventions that coordinate disparate developers towards a common goal. Even Microsoft supports and releases open source software nowadays, inconceivable in 1999 but it started in 2004 (https://en.wikipedia.org/wiki/Microsoft_and_open_source)

Nobody planned this, it just came to be and we are very fortunate to have it. However the heady days of software creation are meeting the realities of middle age which are far less 'fun' and the work has become more like actual work involving maintenance, support, and shoring up all the fast fun work with re-engineered, re-conceptualized versions of what already exists. This is work-work that is best handled by people paid, not volunteers.

A very welcome development is that scientific funding agencies are recognizing that all this software could use some resources to help maintain it, e.g., NASA's recent call for infrastructure funding with the TOPS program. I have been executive director of a major open source software organization, Stan, and if you have not heard of Stan then you hopefully have heard of NumFOCUS of which Stan is a member along with NumPy, SciPy, Julia, and others. This position paper comes from what I thought was a more ideal way to allocate limited resources for scientific software.

So I will argue that funding processes need to evolve as scientific software has evolved. Currently, broadly speaking, reviewers expect novelty in proposals, lean hard on the reputations of the principle investigator/team and then, if awarded, awards less funds than requested. This has problems. Addressing each in turn:

Innovation: Funders/reviewers, even for infrastructure proposals, want something new. In my experience, 'infrastructure' in an RFP (request for proposals) means ‘building new infrastructure’, not 'stay the course'. Bigger, better, faster even if it is not a new idea. This is not what maturing open source projects need. As time advances software needs to be redesigned, perhaps made smaller, documentation needs to be rewritten--all of this is frankly boring but vital non-innovative work.

Reputation: The creators/maintainers of much open source scientific software are relatively junior, not famous, and often don't have strong publication records. Teams of them work from soft money jobs with tremendous responsibility and impact but that is not reflected in easily assessed citations in Nature or Science.

Volatile award amounts: The traditional 10-30% haircut upon award hurts small organizations and any efforts at diversity expressed in the funding application. A 90% paid position is not a position anymore if that is the only source of funding, a diversity initiative will be the first thing cut. There is no 'other source' of funding like teaching fellowships at a University.

Opportunity

The opportunity here revolves around fundamentally changing how funding decisions get made. The 'Intellectual Merit' and corresponding translation into whatever DOE/DARPA/NASA/NIH etc... calls it has to go. In place of innovation, I suggest focusing on anti-innovation by evaluating past and current metrics with the future being "more of the same". In more detail, I suggest metrics like:

1. How much is the software used as quantified by citation count, downloads, forum traffic or breathless letters of support from Nobel laureates?

2. Evidence of an active development community--pull request processing, governance, forums.
3. Evidence that the project knows how to spend money in support of project goals.

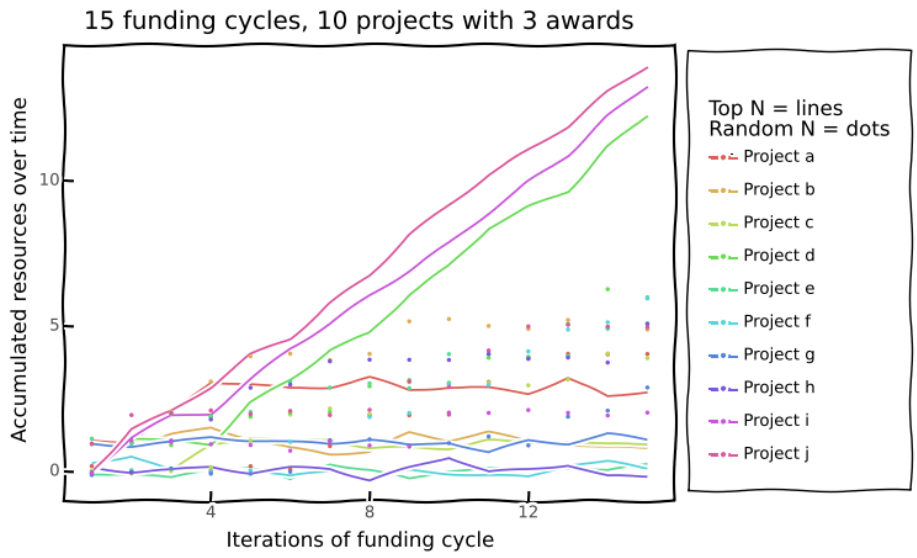
Also, define the merit of the proposal to be well defined/easy to compute hurdles that an organization can aspire to achieve in checklist like fashion-- "Pull request aging less than 10 days on average, check, the average response to forum post 36 hours, check, diversity program in place, check..." Defining such hurdles will be a positive influence on scientific software development if done well, but they must be backed by actual funding decisions.

Stop funding 'Top N' selection: Algorithmic policies impose numerical scores whose goal is to establish the relative rank of all proposals with some wiggle room for program manager judgment. If reviewers/program managers were infallible this would be a great strategy but I question the oracular performance of this arrangement. Any total order selection process reinforces biases, easy quid-pro-quo understandings, and pure reputation-based funded-for-being-funded feedback loops.

Start funding 'Random N' above a threshold: We could send all reviewers off to Baldwin's re-education camp for unbiased evaluation, or... just have reviewers eliminate proposals that seem very unlikely to succeed and fund those that remain. I suggest the awardees be chosen by random draw if there are not sufficient funds. Also, no funding reductions to 'stretch' the budget.

Maturity

Why now? Obviously, federal funders have figured out that infrastructure funding for all this free software might be a good idea. The graph below is a very simple simulation demonstrating the consequences of selection of the top N proposals based on reviewer scores compared to randomly selecting N proposals that are above a threshold. The simulation can be altered and experimented with at: https://blooming-lake-98194.herokuapp.com/.



Resources are reported on the y axis with 1 unit per award, an award increases the reputation of a project by .1 and the score is reputation + constant merit (.2) + random noise. This applies to either selection strategy. The 'Top N' strategy concentrates resources on past awardees given their reputation increment from past funding creating a funding-for-being_funded loop. 'Random N'' strategy, dots, awardees are selected that are above a threshold of .3. There is no advantage being way above .3 or barely above .3. The latter spreads resources much more evenly.

An Experiment

These position papers are supposed to be about the science of scientific software, so I'll suggest an experiment. DOE's next call for proposals should award 50% based on the standard funding process and 50% with a random draw from a pool of competent submissions--scientific software or not. Establish metrics criteria ahead of time and, as much as it pains me as a Bayesian to say, we have a perfect null hypothesis test of the oracle of current practices against the power of randomness.