How to be a good member of a scientific software community [Article v0.1]

Alan Grossfield[1*]

[1]University of Rochester Medical Center, Department of Biochemistry and Biophysics

This LiveCoMS document is maintained online on GitHub at https://github.com/GrossfieldLab/software_community; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated June 14, 2021

Abstract

Software is ubiquitous in modern science — almost any project, in almost any discipline, requires some code to work. However, many (or even most) scientists are not programmers, and must rely on programs written and maintained by others. As a result, a crucial but often neglected part of a scientist's training is learning how to use new tools, and how to exist as part of a community of users. This article will discuss key behaviors that can make the experience quicker, more efficient, and more pleasant for the user and developer alike.

***For correspondence:**
alan_grossfield@urmc.rochester.edu (AG)

1 Introduction

Most practicing scientists (even the ones who write code as part of their research) spend more time as consumers as opposed to producers of software. Moreover, we are constantly learning new skills and solving new problems, which often means learning to use new programs or new aspects of large packages. This, combined with the complex nature of scientific software (and the often low quality of the associated documentation) means we have to ask for help. Sometimes it's because we don't know how to accomplish a specific task. Others, it's because there's a feature we need that isn't implemented. Inevitably, there are bugs.

In all of these cases, it is necessary to interact with the people who develop and support the code. How you go about it has an enormous impact on your likelihood of success and how you are viewed, and asking complete strangers for help is often intimidating, especially the first time you do it. However, the best way to do so is rarely taught explicitly — at best, it's something one picks up by example, from fellow lab members or the PI. *T*he goal of this paper is to reveal the hidden curriculum – what's expected of a software consumer, how to ask for help, how to contribute productively to a software community (regardless of whether you can write code).

This article is written from a perspective of academic open-source scientific software. Much of what we suggest is universal, but some points — like the reminder that the people providing the support are likely volunteers — are not. Regardless, we view the interaction between developers and users as an informal social contract, where each has obligations and expectations for the other. Obviously, every software community is unique, and many have specific conventions for interaction, but we hope that the recommendations we make are at the very least a good starting point for new users of scientific software.

connect to open science: FAIR principles say developers should want code to be reusable. Publishing the code isn't enough, you need to make it usable as well. That means support.

2 Target Audience

This paper is primarily aimed at junior scientists who are new to performing research and inexperienced at asking for help. However, we hope it will be valuable to anyone who uses software to do their research and struggles to ask for help.

3 Good practices in asking for help

4 Obligations of software developers

This paper has focused on what software consumers should do when asking for help. However, this does not mean that the developers of scientific software are without responsibilities. If we want users to behave in a certain way, it behooves us to tell them that. Users approaching us in good faith deserve to be treated fairly and courteously.

5 Checklists

GOOD COMMUNITY MEMBER

- ☐ Tries to solve problem themselves first
- ☐ Asks for help in the right place
- ☐ Writes informative bug reports
- ☐ Cites and acknowledges software appropriately
- ☐ Contributes to the community
- ☐ Treats fellow members and developers with courtesy and respect

POOR COMMUNITY MEMBER

- ☐ Doesn't read the manual or search the internet before asking for help
- ☐ Doesn't use the correct venue to ask for help
- ☐ Writes vague or unhelpful bug reports, or doesn't respond to questions
- ☐ Is rude or demanding when requesting support
- ☐ Treats fellow community members disrespectfully

A GOOD COMMUNITY

- ☐ Helps users solve their problems
- ☐ Is friendly and supportive when responding to questions
- ☐ Is receptive to suggestions and critiques, regardless of the source
- ☐ Encourages participation from users of all experience levels
- ☐ Encourages respectful treatment of all community members, and calls out disrespectful behavior

6 Author Contributions

The initial version of this paper was written by Alan Grossfield.

For a more detailed description of author contributions, see the GitHub issue tracking and changelog at https://github.com/GrossfieldLab/software_community.

7 Other Contributions

For a more detailed description of contributions from the community and others, see the GitHub issue tracking and changelog at https://github.com/GrossfieldLab/software_community.

8 Potentially Conflicting Interests

Alan Grossfield serves as a consultant to two companies, Moderna Therapeutics and Atelerix Life Sciences.

9 Funding Information

This work supported in part by NIH R21GM138970 to AG.

Author Information

ORCID:

Alan Grossfield: 0000-0002-5877-2789