Effective open-science practices for organizing a scientific software repository

Teixeira JMC, Bonvin AMJJ
Bijvoet Centre for Biomolecular Research, Faculty of Science - Chemistry, Utrecht University,
Padualaan 8, 3584 CH Utrecht, the Netherlands

The way we develop scientific software has drastically evolved, especially in recent years, from “scripts in a folder” to open source projects deposited in worldwide distributed community platforms. This paradigm change occurred because we, developers of scientific software, felt an increasing need to unite efforts and resources among the community. The surge of new online platforms, such as GitHub or GitLab, made this leap also possible (*or was it the other way around?*). Software developers and users can now interact in ways never seen before, boosting the development of projects and sparking discussions to the “commit” resolution. However, with *great power comes greater responsibility*. Having our code open to the wild facilitates usability and promotes collaboration and progress. But, despite most of the research projects being now open source, there is still a huge leap between a source/project that is open and a source/project that is usable and that others can build upon. In other words, we need our house clean for guests to feel comfortable and make the magic happen. Users and other developers (and our future selves) expect our project to be readable, understandable, operable, modifiable, testable. Therefore, embracing an open-source community means adopting best organizing practices if we wish our repository to shine, our community to grow, and our project to thrive. But, what are “best practices”? We refer to “best practices” as any behavior we do today that naturally solves or avoids problems in the future. Repository organizing practices are agnostic to the project scope, hence can be adopted by anyone, and englobe source organization, documentation, versioning, contributing guidelines, traceability (issues and pull requests), testing, and CI/CD. As we will discuss: *source organization* communicates where implementations reside and where new ones should be placed; *documentation* tells you everything and should be versioned as well; *traceability* is crucial to maintain a cohesive community and register history; *testing* makes you sleep well at night (aim beyond 100% coverage); CI/CD “saves” you countless time and from *forgot-the-keys* situations; *versioning* defines API expectations and grant full-reproducibility.

Maintaining such practices takes effort (at the beginning way beyond coding); yet, we shouldn't see them as a *pain in the neck* but as a relief, for us to sleep well at night, knowing everything is perfect. We should rewire our brains to dislike chaotic practices naturally. To share our experience in effective good practices, we selected two of our repositories HADDOCK3 and pdb-tools projects, because we believe learning by example is an excellent practice in the field of open science. Utrecht University recently awarded our pdb-tools package the `AWESOME SOFTWARE` badge, meaning it considered it among top packages regarding openness, reusability, and transparency, following FAIR principles and Open Science spirit. You can find HADDOCK3 and pdb-tools in the links below:

https://github.com/haddocking/haddock3 ● https://github.com/haddocking/pdb-tools

Keywords: Python, Software, Open-Source, Best-Practices, CI/CD, Teamwork, GitHub