HPC-BP Webinar Q&A

Research Software Sharing, Publication, & Distribution Checklists

Date: 20 May 2026
Presented by: Dr. Richard J. Acton (Babraham Institute)

(The slides are available via the link in the page's sidebar.)

Q: Regarding versioning: if a project started, commit messages might be of lesser quality. How should one improve the quality to reach “gold” level or will old “bad” commits prevent improved quality assessment?

A: I don’t think that an early git history that is not up to ‘gold standard’ should necessarily be an impediment to reaching it in the longer term. There is inevitably an element of subjective judgement when applying these criteria. Many projects don’t anticipate the scale that they will achieve and adopt the practices that work better for large scale projects from the beginning - these practices can add friction initially so may not make sense from the beginning. For example If your project adopted conventional commits at some point so that you can parse your commit messages to extract information about your project, having your tooling aware of when this practice started and commit hooks to enforce this correct formatting in place. I see no reason why this would not allow a project with an initially messy commit history to get a gold or platinum rating.

Q: What are your thoughts on “learning” or “developing” a specific protocol e.g., CUDA for GPU porting as opposed to learning the paradigms of the same on an open source product. Especially in this age of hyper contextual/specialized and optimized paradigms.

A: This is very much dependent on why you are learning or developing the protocol. If your institution is heavily invested in the proprietary tooling it may just be impractical to go against the grain on this. You may need performance now with the tools available to you. If your project is one you can run in 30 years and performance is not as critical, maybe your calculus is different. In the long run the more open platforms tend to win out as competitors to incumbents tend to use openness as part of their strategy for coalition

building to displace them. Using vulkan / opengl / opencl tend to be more platform agnostic but especially in the case of the latter two afford less granular control, (as I understand it - GPU programming is not my area), so may or may not be able to fulfill your performance requirements. Ultimately it is a question of evaluating the trade-offs for your specific problem.

Q: Under documentation, documentation of theory and assumptions is not explicitly mentioned. Is this an intentional omission? Documentation of theory and assumptions is more about replicability than reproducibility, so maybe this is outside of the scope of the current work?

A: Insofar as the theory and assumptions underpinning the work are relevant to design choices made in your software they should be documented. In general my view is If it helps someone understand what the code is doing and why it adds value to the codebase to document it. This is especially the case if it augments any information that may be available in an accompanying manuscript, if there is not one this becomes more important to document alongside the code. A quick outline of theory and assumptions with reference to an accompanying manuscript and the addition of details not present, or where the mapping from prose to code is unclear, in a separate description of your methods to the documentation or comments is worth adding.
(I'd welcome an issue of PR with your thoughts on how best to adjust my current wording to reflect this.)

Q: What are best practices for sharing a code and specifying credit giving for your work on a code repository not shared by the original author but you have only improved it and not solely written?

A:It seems to me there are a few considerations here, authorship and credit can be tricky to navigate especially when works can emerge from complex project histories without a plan for how people are going to be credited from the beginning. Also norms can vary quite a bit from one sub-discipline to another. When thinking about this I always start with the general goal of accurately representing contributions to & responsibilities for the project.

I would guess that your permission to share the work stems from the previous authors' having worked for the same employer as you / on the same project. It would be important to ensure that you do have the legal rights to publish the work.

Your scenario leaves out a few specifics in particular if you are still in contact with the original author. I'm inclined to think that you are not in contact or you might then be

posing this question to them. They may readily be able to answer the question of how they would, or would not, like to be credited if you are in touch. I would try to get in touch and communicate your intentions if you are not. If one wants to avoid misrepresenting original authors' contributions - what if there were other authors who contributed but this is not well documented? - it can be tricky to do so without their involvement. When you are not in contact with an original author it can be tricky to give them appropriate credit without the implication of ongoing responsibility for the work or endorsement of it.

The norm with most open source projects which are forked seems to be simply to point to the original project and say this project forked off of this one and to largely leave it at that. The version control history affords anyone who cares to look insight into how much the new fork has changed from the original, which is different from most other works. As you said that the original author never shared the project the analogy to a project fork does not quite stand. When taking on the maintainership of an open source project even if you did not write much of the original code-base you are taking responsibility for it now so it is you who has to field communication about the project and vouch for its correctness / quality not the original author.

It may be helpful to think about the differences between Authorship and Contributorship: Authorship in the context of scientific publishing can connote taking responsibility for the correctness and quality of the work especially by primary and senior authors. I'm not aware of particularly well formulated norms around how to allocate credit to someone who has left an academic project and is no longer in contact with the team that is continuing their work, or using a method that they developed as academic publishing largely follows an authorship not a contributorship model. Meaning that if someone has left and is not there to vouch for their work as an author their contributions may go unnoted.

It can be helpful to have a structured language for describing different parties' contributions to a project. Contributor Roles Ontologies & Taxonomies (CROTs) like CrediT, ScoRo, CRO, or TaDiRAH. Provide such frameworks and could make bibliometric analysis of individual contributions more conducive to larger scale study if more widely adopted in a machine readable fashion. See https://doi.org/10.1080/08989621.2022.2161049 for an interesting review pertaining to the ethics of their design and use.

Q: Is everything that we post to GitHub automatically licensed?

A: *Disclaimer: I am not a lawyer this is not legal advise*

No* You or your organisation if your contract with them grants them rights over your work product retain the rights to your work by default. To allow others to use your work without exposing themselves to liability for copyright infringement or having to individually negotiate a license with you / your employer, you would need to provide a suitable license under which others can use your work. The licensing section of the checklists and many other resources online provide some guidance on license choice. All you usually need to do is include the licence file in your repository with the copyright holder's name and the year.

If your employer owns the rights to your work product you should check with them about their policies / contract terms about your contributions to open source projects. Many academic institutions have exemptions allowing their staff leeway to openly license their academic outputs but the wording is often specific to things like academic papers and may need updating to reflect the role that producing and contributing to open software now plays in the academic ecosystem. Talk to your institutions about this and make sure that policy lines up with practice.

A note on public domain: As this seminar was organised in the context of a US based HPC network there may be US government employees in the audience who have to release their work via a commitment to the public domain. I just wanted to note that this does not exist in all jurisdictions so it is preferable to use an off-the-self license that includes a public domain commitment and does its best to approximate one in other jurisdictions. I've encountered public domain software from US government agencies which only make a public domain commitment which is excluded from build caches by packagers as distributing the binary may expose the cache operator to liability in jurisdictions where public domain commitment is not a thing. Please save some Europeans the compile time.

* When you accept the github terms of service you are agreeing to let others fork your code on github. From the github TOS (2026-05-21): https://docs.github.com/en/site-policy/github-terms/github-terms-of-service#5-license-grant-to-other-users

> *5. License Grant to Other Users*
>
> *Your Content that you post publicly, including issues, comments, and contributions to other Users' repositories, may be viewed by others. By setting your repositories to be viewed publicly, you agree to allow others to view and*

> *"fork" your repositories (this means that others may make their own copies within the Service in repositories they control).*
>
> *By making a repository public, you grant other Users a nonexclusive, worldwide license to use, display, perform and reproduce (by forking) Your Content through the Service as permitted by GitHub's functionality. You may grant additional rights by adopting a license. If you post Content you did not create or own, you are responsible for ensuring it is licensed under terms that permit these uses.*

Q: What about the simulations that require large amounts of resources that might not be available to reviewers to re-run a case? Could it be maybe useful to provide a minimum reproducible example as you do with bug reports?

A: This can be a challenging problem especially if your simulations are numerically intensive enough to require hardware specific optimisations that limit their portability without significant additional work.

Your suggestion of providing a minimal reproducible example is an excellent one and the approach that I would think is most practical in most cases. It is probable that you already developed such examples whilst developing your simulation code to informally test your simulation code. Taking the extra step to formalise such tests using a testing framework that checks that your simulation outputs are within expected bounds and/or reproducible with a given random seed if applicable is a best practice anyway to ensure that any changes you make to it don't end up producing outputs that violate your assumptions. I would expect that re-using the functions that you write to make these tests in a minimal example would work well.

One can imagine cases where the decisions riding on the outcome of a simulation are consequential enough that even if it is compute intensive you might want to check its robustness to being re-run on a different compute infrastructure even with the high compute costs. Economists (being economists) have done market based modeling of the economics of computational reproducibility (https://doi.org/10.2139/ssrn.4811513) and they indicate that there are economies of scale* to be had from journals outsourcing reproducibility checking if they institute it. It would be interesting to see journals partner with HPC providers to perform reproducibility checks - not always necessarily at full scale - of computationally intensive simulation workflows as part of their manuscript acceptance criteria.

*(this specific case is skewed a little by the high cost and prevalence of paid data

sources in finance/economics but I suspect their conclusion would still hold just with a smaller cost saving in other domains)