Review of “Software Sustainability of global impact models”

Facundo Fabián Sapienza
Monday 26th of August 2024

The manuscript contributes to the scientific literature by quantifying the level of software sustainability of different climate change impact models. This is done by first defining a series of nine indicators that evaluate factors such as automation, documentation, testing, and good coding practices present in the model.

I recommend the publication of the manuscript in GMD after the revisions and comments here presented are addressed. I believe this work brings light into the robustness, reproducibility and overall health of the software that we scientist develop. I personally enjoyed the reading of the manuscript, and I would be glad of seeing this work published. However, before publication I strongly suggest the authors to review and address the following major and minor points. If well overall the manuscript is easy to read and follow, some parts are a bit vague and required more explanations and/or more bibliographical references.

All comments, both major and minors, are aimed to improve the quality of the paper or to dilucidated some of my doubts or questions as I was reading the manuscript.

Major Comments

- It will help to the discussion the early introduction of one or more examples of software that follows the guidelines of “sustainable research software”. I suggest using standard examples for this, for example a major library in Python (numpy, pandas, sklearn) with a link to their respective source code would help the reader to have an idea of how good software looks like.

- [146] **Version control.** My understanding is that in the manuscript the authors assess whether the models use or use not git or a similar version control system. However, I think it is also important to evaluate how version control has been used during the development of the different versions of the model. Did the contributors follow good version control practices (modular commits, push requests, discussions, versioning, etc), or simply put the final version of the software in GitHub or similar? I understand this is a finer analysis which I don’t think is necessary to perform in the manuscript, but I would suggest mentioning this in the Version Control section since it is an important point.

- [154] **Use of open source license**. I think this section requires more discussion or at least a few references supporting the statement *“Open-source licenses foster collaboration and transparency by enabling community contributions and ensuring that software remains freely accessible”*. I partially agree with this statement, but I would like to see the why of this. Furthermore, I think is also important to mention the difference between different types of major licenses (copyleft or permissive), since I think this also has an impact in the meaning of the sentence. This point pops up every time the topic of licenses is addressed in the paper, so I strongly suggest the authors to address what do they mean by open-source licenses and how they relate to copyleft/permissive licenses.

- [159] **Number of active developers.** I think the number of active developers is a good proxy for evaluating how robust is the development of the software, but I don’t think is the only one. I think with the same philosophy one can evaluate the number of commits, push requests, open and closed issues, etc. I suggest mentioning this in the manuscript, and maybe changing the “number of active developments” tag to something more generic that includes these other proxies or a more general concept (e.g., “software robustness”). Furthermore, it may be important to emphasize the distinction between developers and contributors: one can contribute to a project without writing

source code, but for example opening issues, managing a project, writing the documentation, etc. In this sense, contributors help to improve the robustness of the software, even when they don't directly write any line of code.

- [166] **Containerization.** I will suggest here making also a reference to cloud supported containers, such as Binder or GoogleColab, that allow the re-execution of the software. Under the hood, this also work as a container, but I think the deeper concept here is the capacity of re-executing an model with the computational environment requirements. This further resonates with the concept of analysis-ready data, cloud-optimized formats (see "Cloud-Native Repositories for Big Scientific Data" by Abernathey et. al, 2021) in the case of datasets. I think the same ideas apply here for models.

- [178] **Public availability of an (automated) testing suite.** I really like this point. I this is a good idea to look for automation of the tests. However, I would like to point out here that the concept of "automation" is in principle independent and complementary to the existence of the testing suite. Furthermore, the concept of automation applies also to some other indicators. For example, one can automate the creation of documentation using GitHub Pages, using GitHub actions to ensure the containerization of the software (even create all the ingredients for a docker file), and even ensure the compliance with coding standards of the software (see for example the Julia code formatter action: https://github.com/julia-actions/julia-format and I imagine there must be a way to automate the use of Pylint with Python).

- [269] How well documented are the models that just have a README file? In my experience, README documentations tend to be very plain and difficult to navigate. I think it is important to mention something about the quality of the documentation, at least as an observation.

- [295] Following my previous comment, I think it is important to state why licensing is important and the difference between licenses. References or further support here is needed. Copyleft and permissive licenses are very different. I would also suggest pointing which one of the licenses in Figure 2 are copyleft of permissive.

- [378] I don't understand the purpose of this section. The effort here is calculated using equation 1, which (besides being based in many assumptions that the authors do take care of) already suggested that more lines of code mean more effort, which is an expected concussion. With this point in mind, I don't fully understand what new conclusion are made in Section 3.3. There is a chance that here I am missing an important point, and in that case, I would like the authors to clarify. I think here it would be interesting to see how the effort correlated with the number of satisfied indicators. Without further analysis, I would suggest removing or perform a different analysis in this section.

- [410] I think this is a very important question and should be first raised in lines 197-200, and maybe postponing the discussion until later. It is not clear for me that 30-60% is the desired number, since most of the time other heuristics are used for determining the number of comments. In a nutshell, better code requires minimal commenting and clarity in the code. (See book "Beautiful Code" for a nice collection of essays around this precise point.) Furthermore, I think is important to mention the entanglement between writing good documentation and commenting, since many software documentations are generated automatically based on the comments in the code (e.g., based on the docstrings in Python and Julia).

- [490] Here it is mentioned a very important point: design principles, or design patterns. More than writing code that follows certain standards, it is important to think about the overall software architecture of the model. E.g., if I am working in Python, what are going to be my classes? How do they interact with each other? How data will be processed? Etc. None of the indicators really

address this aspect of software development (which I think is fine for the scope of the manuscript), but I think the authors should emphasize this point in the recommendation section.

- [496] As it was already mentioned, I think is important to remark that what here is referred as internal and external documentation sometimes are the same. Documentation can be created from comments in the code, specially docstrings, and this is actually a great documentation practice, to make the internal and external documentation to be the same so there is no repetition nor contradiction between the two.

- Since automation is mentioned as playing an important role, I would suggest including automation in a more general sense in the recommendation section. This includes using GitHub Actions to run the test suite, automate the creation of the documentation (static or dynamic), check for code style, check packages dependencies, etc. Another interesting tool the authors may want to consider mentioning is Makefile.

- Since the paper addresses the important aspect or sustainable software, I would strongly suggest that the code used to generate the analysis, and the figures of the manuscript are presented in the same standard that the nine indicators dictate. I think this will really improve the quality of the work, and it can serve as an example in the manuscript itself of "how things should be done". I think readers would like to see that, and I think it is part of the philosophy of the manuscript to promote these good practices. Furthermore, I would make the point that the same tools that had been used in this manuscript can be used for analyzing other source code models, so the code used in the manuscript can be re-usable by other users.

Minor Comments

- [30] I am a bit confused by what Earth system modelling entitles. When presenting the models, we are talking about models in agriculture, biomes, fire, etc. In line 33, it says that "While so-called Earth System Models always include the simulation of atmospheric processes and thus compute climate variables and how they change due to greenhouse gas emissions [...]". However, this excludes many "Earth system models", including all those not based in atmospheric processes. I think the phrasing of Earth system model should be narrowed to what the models are for. For example, all the models in the ISIMIP are about the impact of climate change.

- [66-70] Repeats reference Anzt et. al. (2021)

- [83-99] I suggest splitting this paragraph into two, with the second stating what is done specifically in this work (maybe just break before "In this study, we assess...").

- [105] Data in this line means "model"? I will suggest not to use "data" to refer to the models in the ISIMIP database, since it is a bit confusing. If this is referring to another type of data, maybe explain.

- [110-111] This sentence requires rephasing, since it is ambiguous what it means by "in the described way" (meaning GitHub/GitLab or also including source code in reference papers).

- [189] Is PEP8 the de-facto coding style in Python? I think there used to be some alternatives and may be important to mention something like this. Furthermore, in other programming languages there are more than one coding style that are accepted by the community (e.g., in Julia), so it may be important to mention that the important aspect of this is that the developers of one model stick with one style, rather that sticking with one single style.

- Table 1. Following the logic in the text, I will suggest making the division in the table between best practice in software engineering and source code quality (e.g., adding this information in the table, dividing them with a horizontal line). I think this will make the conceptual difference clearer than using the footnote. Check punctuation at the end of the descriptions. Some of the items end with dot, not dot, or comma.

- [245] These are the 32 models mentioned in line 110, right? I would mention this again for clarity.

- Figure 1. I will suggest sorting the bars in increasing/decreasing order. This is a comment I had with all the rest of the tables of the paper, where I would order the bars for clarity.

- [255] I will suggest not starting the sentence with a numeral since this is uncommon and non-recommended in formal English.

- [289] There is some repetition between what is said here and the paragraph in line 255 and Figure 1. I think mentioning git and the corresponding platform (GitHub, …) should be made once in the same section for clarity in the text.

- Consider introducing Table 2 before Figure 1, since it contains the overall information of the model.

- It would be great to add an extra column to Table 2 with the year of the model (last version) and sort by this. I think it would be interesting to see if the availability of documentation, version control, etc, had improved over the years. Also interesting to see programming language used and how this changed over the years.

- [291] I don't agree with the statement that "Developers' preference for Git highlights its user-friendly nature and effectiveness in supporting collaborative efforts". I think there may be other reasons for this, since there were and are other version control systems that are as user-friendly as git but didn't became that popular. One reason for this is that GitHub naturally uses git, and that many developers use VSCode with also supports git and GitHub. If the authors want to keep this sentence as it is, the statement should be supported by references.

- [306] It is important to start saying what is a good number of contributors, or what is expected to see here. It is unclear to me that ~10 contributors is robust enough.

- [320] Mention which container platform these 5 models used. Did all used Docker? If so, how do they share it?

- [326] I am curious: do the models with test suite use a preferred programming language? Does the programming language plays a role in how easy is to implement the test suite? Maybe the authors want to answer to this question in the manuscript, I think it will make an interesting point.

- [249] I will suggest not starting the sentence with a numeral.

- [429] I really enjoyed this section, and I think it will improve to the communication of the paper to put this section earlier in the manuscript, maybe between the introduction of the indicators and the analysis and the results.

- [478] Same point about permissive and copyleft licenses. What do the authors mean by open-source licenses? Do you mean permissive?

- [482] I am not sure that version control ensures software reproducibility, not with other important tools (environment or containerization, testing suite, etc).

General comments

- I strongly suggest sorting all the tables in decreasing or increasing order for readability.

- I suggest the authors to do a English style revision of the manuscript since there are different styles in the manuscript. This includes the starting of the sentences with numerals and the use or not use of contractions.

- In the same style that Figure 9, I think a figure at the beginning of the manuscript summarizing the indicators and what are good software practices will improve the manuscript.