REMODEL automated report
A Community Poll on Reproducibility in Computational Geosciences

Robert REINECKE

August 7, 2021

Report last updated:	May 27, 2021
Compilation date:	August 7, 2021
N total:	346
N completed:	263

This document summarizes a full analysis for detailed results. **It is not a full publication but rather an automated representation of the data presented in this repository.**
It summarizes the extensive results and builds the foundation for the publication of a journal paper. It will also be distributed along with the data in the course of a journal and data publication. The following analysis only takes into account the fully completed poll from 263 participants.

Contents

List of Figures

1 Aim of this survey

Software development has become an integral part of the geosciences[1] as models and data processing get more sophisticated. Paradoxically, it poses a threat to scientific progress as the pillar of science, reproducibility, is seldomly reached[2]. Software code tends to be either poorly written and documented or not shared at all; proper software licenses are rarely attributed. This is especially worrisome as scientific results have potential controversial implications for stakeholders and policymakers and may influence the public opinion for a long time[3].

In recent years, progress towards open science has led to more publishers demanding access to data and source code alongside peer-reviewed manuscripts4,5. Still, recent studies find that results can rarely be reproduced [6,7].

In this project, we conduct a poll among the geoscience community which is advertised via scientific blogs (AGU, EGU), research networks (researchgate.net and mailing lists), and social media. Therein, we strive to investigate the causes for that lack of reproducibility. We take a peek behind the curtain and unveil how the community develops and maintains complex code and what that entails for reproducibility[8] . Our survey includes background knowledge, community opinion, and behaviour practices regarding reproducible software development. We postulate that this lack of reproducibility[9] might be rooted in insufficient reward within the scientific community, insecurity regarding proper licencing of software and other parts of the research compendium as well as scientists' unawareness about how to make software available in a way that allows for proper attribution of their work. We question putative causes such as unclear guidelines of research institutions or that software has been developed over decades[10] , by researchers' cohorts without a proper software engineering process[1] and transparent licensing.

To this end, we also summarize solutions like the adaption of modern project management methods from the computer engineering community[11] that will eventually reduce costs while increasing the reproducibility of scientific research[8] .

1 A comment to "Most Computational Hydrology is not Reproducible, so is it Really Science?" R.W. Hut, N.C. van de Giesen, N. Drost, Water Resources Research, 2017

2 Hutton, C., Wagener, T., Freer, J., Han, D., Du_y, C., and Arheimer, B., Most computational hydrology is not reproducible, so is it really science? Water Resources Research, 2016

3 Munafò, M., Nosek, B., Bishop, D. et al., A manifesto for reproducible science. Nat Hum Behav, 2017

4 Executive editors, G. Editorial: The publication of geoscientifc model developments v1.2. Geoscientifc Model Development, 2019

5 Katz, D. S., Niemeyer, K. E., and Smith, A. M., Publish your software: Introducing the journal of open source software (joss), Computing in Science Engineering, 2018

6 Stagge, J. H., Rosenberg, D. E., Abdallah, A. M., Akbar, H., Attallah, N. A., and James, R., Assessing data availability and research reproducibility in

hydrology and water resources. Scientific data, 2019

7 Añel, J. A., García-Rodríguez, M., and Rodeiro, J.: Current status on the need for improved accessibility to climate models code, Geosci. Model Dev., 2021

8 Stodden, V., The reproducible research standard: Reducing legal barriers to scientific knowledge and innovation. IEEE Computing in Science & Engineering, 2009

9 https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

10 Muller, C., Schaphoff, S., von Bloh, W., Thonicke, K., and Gerten, D., Going open-source with a model dinosaur and establishing model evaluation standards. EGU, 2018

11 https://software.rajivprab.com/2019/11/25/the-birth-of-legacy-software-how-change-aversion-feeds-on-itself

Our larger questions:

- Is reproducibility an issue in the geosciences? Are bad code and documentation the root cause of that issue?

- Is model software too complex? Does that hinder reproducibility?

- Are researchers missing the tools and know-how (methods, licenses etc.) to build good model code?

- Is missing funding and time preventing researchers from making their models more accessible?

We define reproducibility as:
"Reproducibility in the context of modeling in the geosciences means that results obtained by a modeling experiment should be achieved again with a high degree of agreement when the study is replicated with the same model design, inputs, and general methodology by different researchers.
We explicitly exclude the retracing of results by means of using a different modeling environment (including variations in model concept, algorithms, input data or methodology)."

2 Data processing

We designed the survey according to standards from psychology research. We apply descriptive statistics to analyse demographic background and basic analysis. Further, we apply inferential statistical methods to test the underlying hypotheses.
The raw data of the survey is stored in the folder **LiveData**. **The raw data has not been modified or cleaned in any way.** To run some basic cleanup run the follwoing script:

```
1  import process_data.py as p
2  p.process()
```

3 Results

All data processing and plotting (including building this document) can be executed by running `python run.py`. Plotting details and addtional processing can be found in th script `plot_all.py`.

Our main hypothesis for this analysis where the following:

- **H1** Young scientists develop software more actively than established researchers.

- **H2** Young scientists are more familiar with software licenses than established researchers.

- **H3** Young scientists are more familiar with modern development methods than established researchers.

- **H4** Senior researcher perceive reproducability as a less severe problem than early career researchers.

- **H5** Software complexity is the main reason for a lack of reproducability.

- **H6** Researchers code frequently but without knowledge about proper engineering methods, licences and tools.

- **H7** The most frequently used language is still C/Fortran. Younger scientists tend to use Python and R; this is consistent throughout fields.

- **H8** Most researchers are autodidacts when it comes to coding.

- **H9** Most researchers have never reproduced code with the original model. Only with their own model. This differs between fields.

- **H10** Practitioners and researchers perceive the issue of reproducibility differently. Scientists are more aware (?).

- **H11** There are more researchers that apply software than they are ones that develop it. This differs between activities and fields.

- **H12** Most researcher do not know if their software belongs to them.

- **H13** Models that are available are hard to use. Causes?: Bad code, no documentation, no input data

- **H14** Senior researchers are convinced their work is reproducible. (much more at least than young scientists)

- **H15** The smaller the scale the more reproducible and accessible.

- **H16** Researchers think that their software is bug free and always correspond to their intended implementation.

- **H17** Most senior scientists think investment in FOSS doesn't pay off.

- **H18** Senior researchers think that their code/project is easier to understand but that conflicts with reality. And younger researchers have the opposite understanding.

- **H19** We need more funding to enable reproducible computational science.

3.1 Sample characteristics (demographics)

Who were our participants? Here we present characteristics of the participants in our poll, i.e. their current career stage, their years of research experience, their geo-scientific field and scale as well as their current focus of work. This is purely descriptive statistics. As we welcomed everyone to our poll, we did not form any assumptions regarding sample characteristics. Also, we tried, but might not have reached a representative sample of the population of geo-scientists.
Here, we report basic sample characteristics. Also, we can check for and report any salient sample properties.
Corresponding survey questions:

- DM01 - What career stage are you in?

- DM02 - For how long have you been working in your research field?

- DM06 - To which field within the geosciences does your research mainly belong?

- DM05 - What geographic scale are you working on?
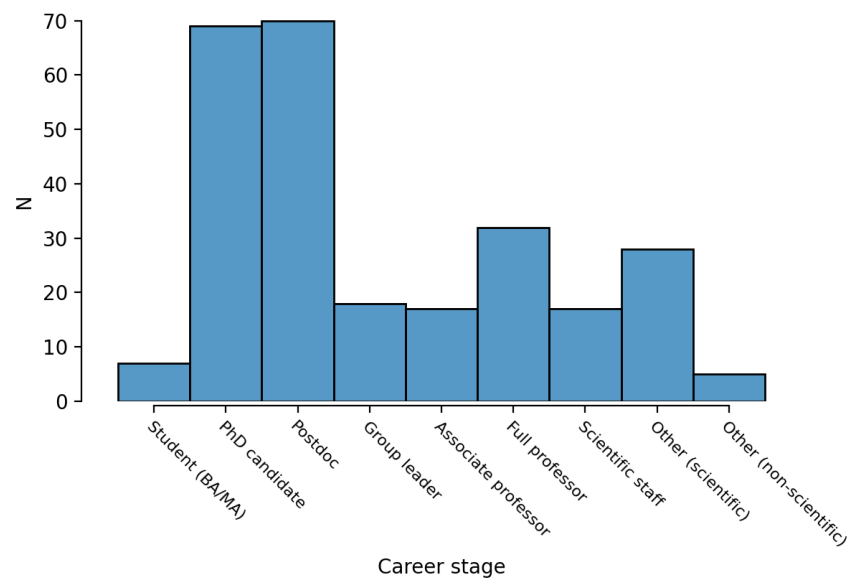
- DM07 - What is the focus of your work?

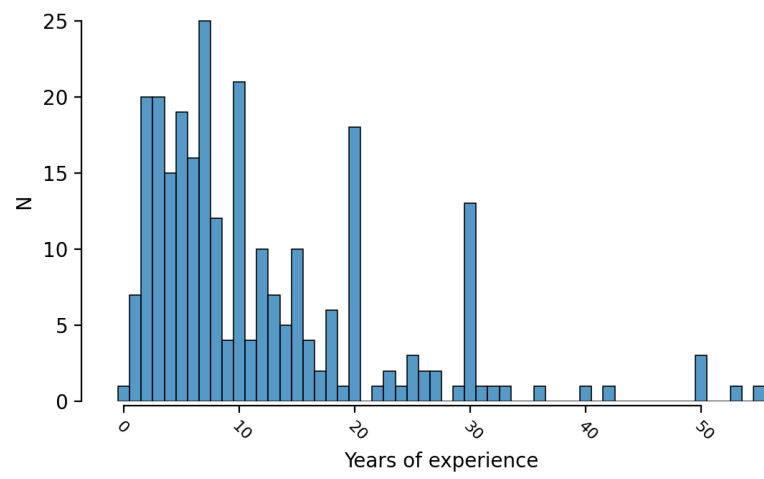Figure 1: DM01 - Career stage of participants

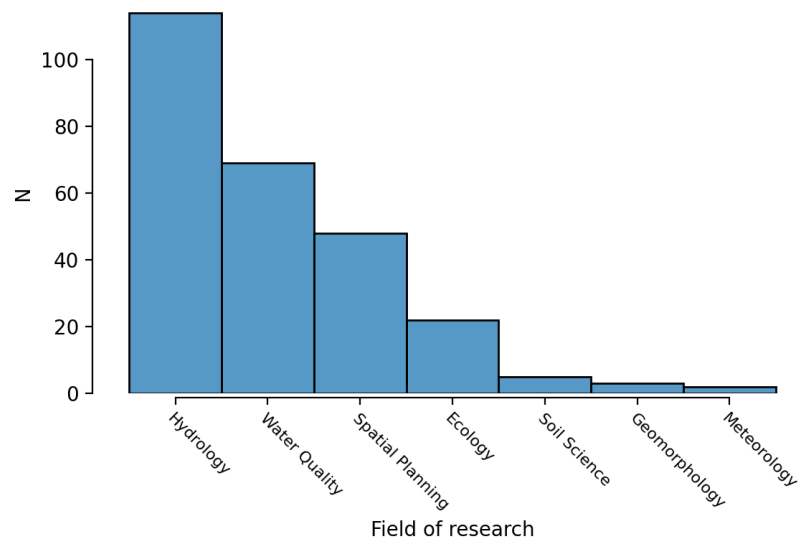Figure 2: DM02 - For how long have you been working in your field?

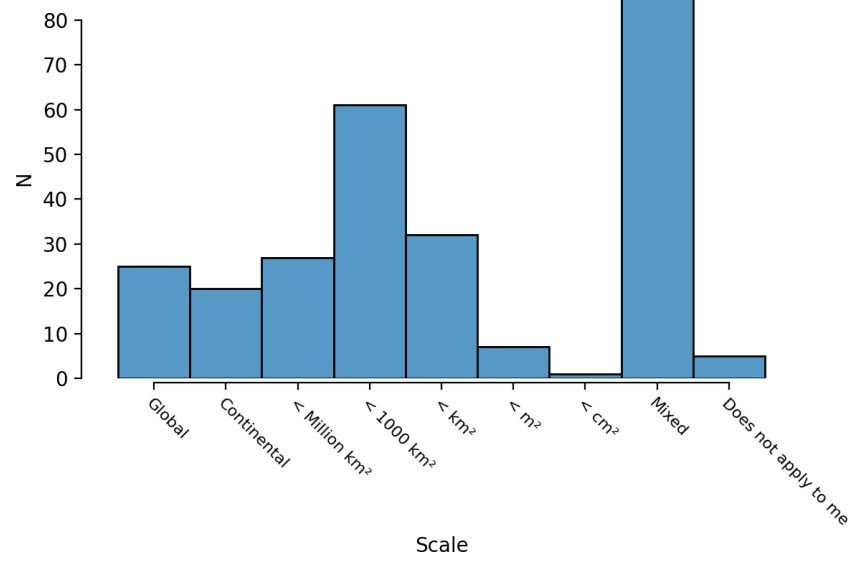Figure 3: DM06 - Which field do you belong to?

Figure 4: DM05 - What scale are you working on?
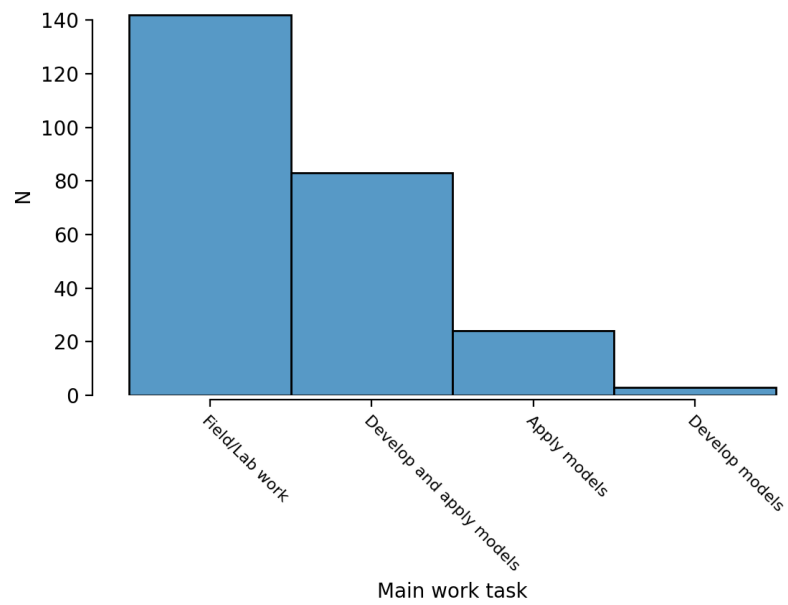
Figure 5: DM07 What is the focus of your work?

3.2 Community opinion on reproducibility

This part covers people's view and opinion. We assume that our defintion of reproducibility was used (annotated at several points in the poll).

This part focuses on how reproducibility is perceived by different researchers and what the causes are for limited reproducibility.

We expected that early career scientists are more aware of the problem than senior scientists. We further expected that a main cause for a lack of reproducibility is the lack of accessible code and input data.

Corresponding survey questions:

- O101 - How strongly do you agree with the following statements?

- O103 - What are the reasons for a lack of reproducibility?

- S113 - How long do you think does it take for an average PhD student to efficiently work with your research software?

Corresponding hypotheses

- H4 Senior researcher perceive reproducability as a less severe problem than early career researchers.

- H5 Software complexity is the main reason for a lack of reproducability.

- H10 Practitioners and researchers perceive the issue of reproducibility differently. Scientists are more aware (?).

- H13 Models that are available are hard to use. Causes?: Bad code, no documentation, no input data.

- H14 Senior researchers are convinced their work is reproducible. (much more at least than young scientists).

- H16 Researchers think that their software is bug free and always correspond to their intended implementation.

O101: Opinion on Reproducibility in Geo-Sciences: How strongly did participants generally agree with statements? Do they consider it a problem at all? Do they think that their work is reproducible?

Figure 6: O101 How strongly do you agree with the following questions?
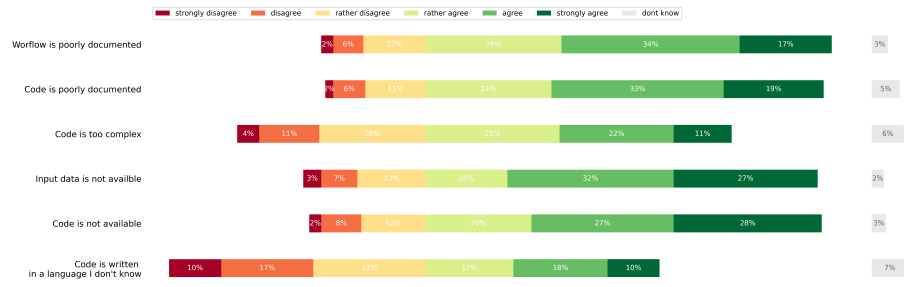
Figure 7: O103 What are the reasons for poor reproducibility?
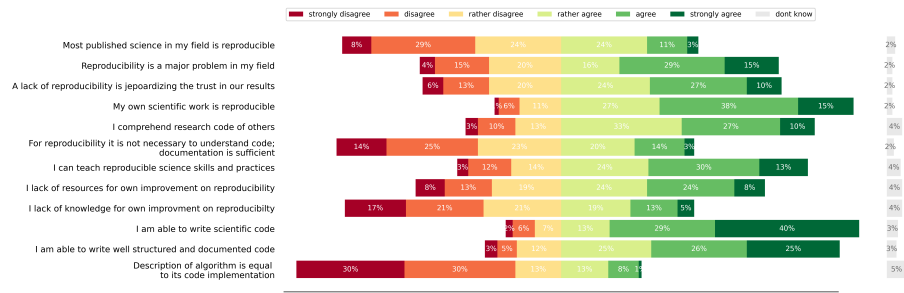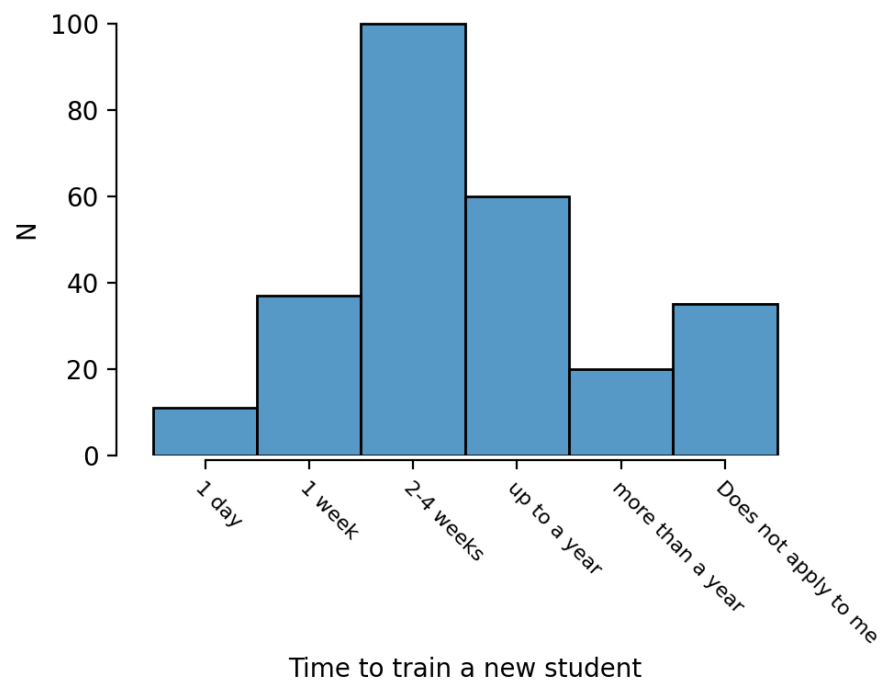
Figure 8: S113 How long does it take to train a new student in your software?

3.2.1   H4 Differences in opinion on reproducibility

Number of established (full Prof.) researchers: 32
Number of young (Students, PhD candidates, Postdocs, Junior Prof., Associate Prof.) researchers: 181
Mean agreement to reproducibility: 3.105058
Mean agreement among young sc.: 3.005650
Mean agreement among established sc.: 3.612903
Median agreement to reproducibility: 3.000000
Median agreement among young sc.: 3.000000
Median agreement among established sc.: 4.000000 Career has no normal distribution
No statistical test possible, samples of to low to do a proper Wilcoxon or even t-test

3.2.2   H5 Reasons for lack of reproducibility

All answers (without don't know) - Mean workflow: 4.400000, code documentation : 4.449799, code complexity: 3.874494, input data not available: 4.505837, code availability: 4.503906, written language: 3.481633
Thus, the main reason for a lack of reproducibility is: Input data, then 2: code availability, 3: documentation, 4: workflow, 5: code complexity, 6: language
Mean agreement code complexity 3.874494
Mean agreement among young sc.: 3.771930
Mean agreement among established sc.: 4.033333
Median agreement: 4.000000
Median agreement among young sc.: 4.000000
Median agreement established sc: 4.000000

3.2.3   H10 Practitioners are less awarte of the issue

No Data to investiagte this question. Only scientists answered the poll.

3.2.4   H13 Reasons for reproducibility

See H5. No addtional data to explore this further.

3.2.5   H14 Senior researchers are convinced their work is reproducible. (much more at least than young scientists).

Mean agreement on: my own research is reproducible: 4.428571
Mean agreement among young sc.: 4.413408
Mean agreement among estbalished sc: 4.562500
Median agreement: 5.000000
Median agreement among young sc: 5.000000
Median agreement among established sc.: 5.000000

3.2.6 H16 Researchers think that their software is bug free and always correspond to their intended implementation.

We did not ask a question that perfectly relates. Parts are answered in Fig. 6. Main relation to question: Implementing an algorithm based on a description from a publication yourself is the same as using the exact software package/original code that was used in that very publication.
Mean agreement - description is same as implementation: 2.389558

3.3 Reproducibility Practices and Skills

This part covers "actual" behaviour. This is only a self-report assessment.
Summarized, we wanted to know if researchers actively participate in reproducing results. Furthermore, we were interested in finding out what kind of tools and licences they use to publish their own results.
We expected to see that that less than 50% are actively reproducing results and that general knowledge about tools and licenses is low.
Corresponding hypotheses

- H6 Researchers code frequently but without knowledge about engeneering methods licences and tools.

- H9 Most researchers have never reproduced code with the original model. Only with their own model. This differs between fields.

- H12 Most researchers don't know if their software belongs to them.

- H2 Young scientists are more familiar with licencing issues.

- H7 The most frequently used language is still C/Fortran. Younger scientists tend to use Python and R

- H13 Models that are available are hard to use. Causes?: Bad code, no documentation, no input data

- H3 Young scientists are more familiar with modern development methods than established researchers.

Corresponding survey questions:

- O102 Did you actively reproduce scientific results in the past?

- S103 How often do you use research software?

- S110 How often do you develop research software?

- S202 Do you own your software?

- S112 Which licences are familiar?

- S101 What kind of programming languages are used?

- S111 Which licences do you use?

- S104 Do you practice any of the following methods?

- S105 Do you these tools?

- S106 Gow did you learn to program?
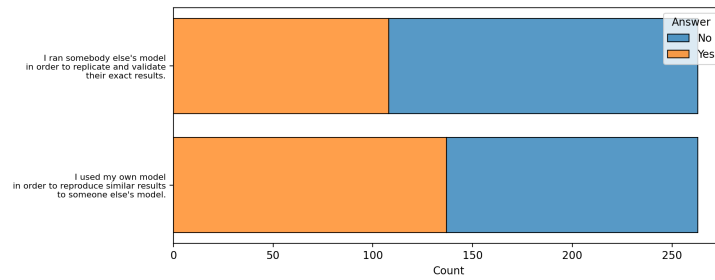
Figure 9: O102 Did you reproduce results?
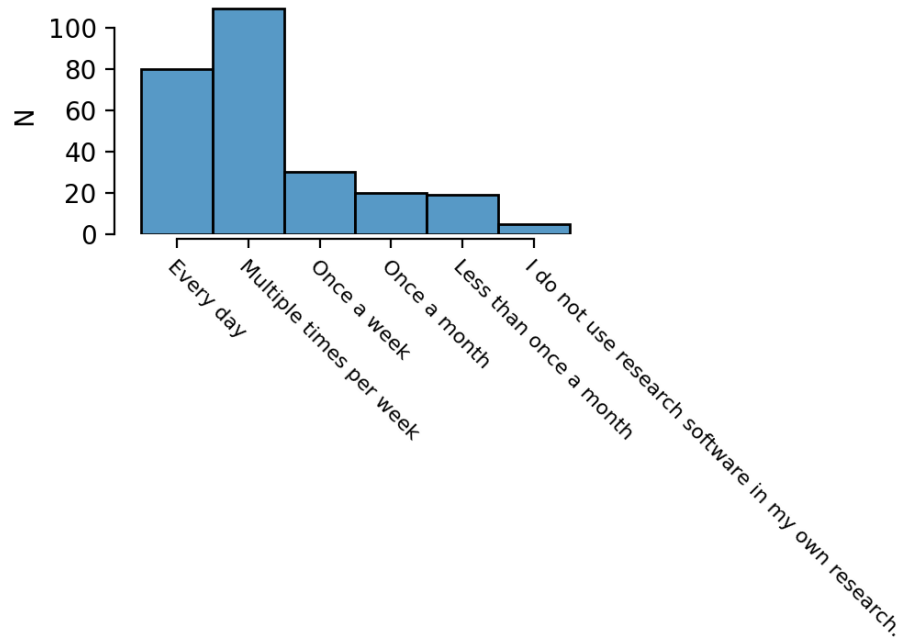
Usage of research software

Figure 10: S103 Frequence of using software.
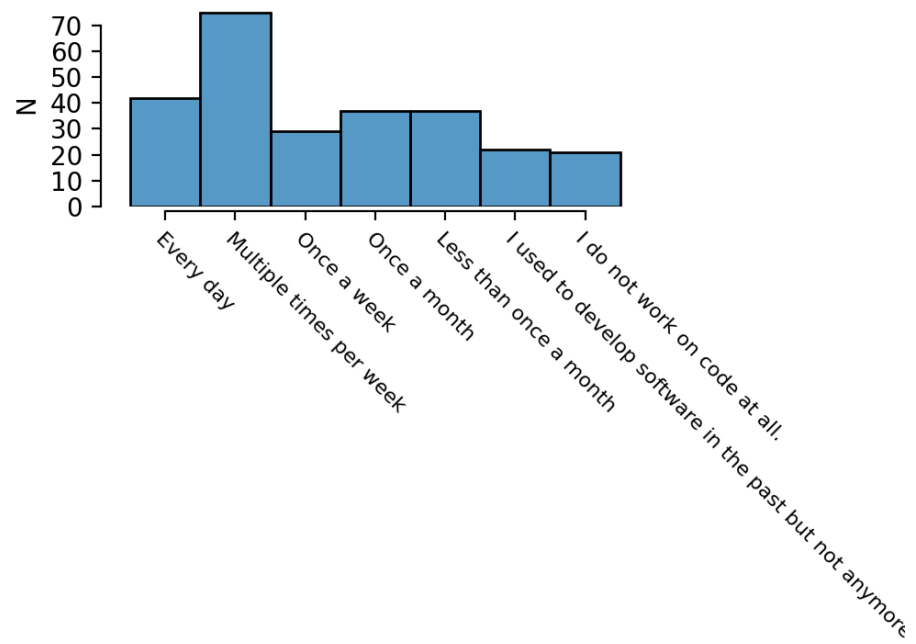
Frequence of research code development

Figure 11: S110 Frequence of developing software

Figure 12: S202 Do you own your software?

3.3.1 H6 Knowledge

Figure 13: S111 What software license are you using?

Figure 14: S112 Which licenses are you familiar with?

Figure 15: S101 What languages are you using?
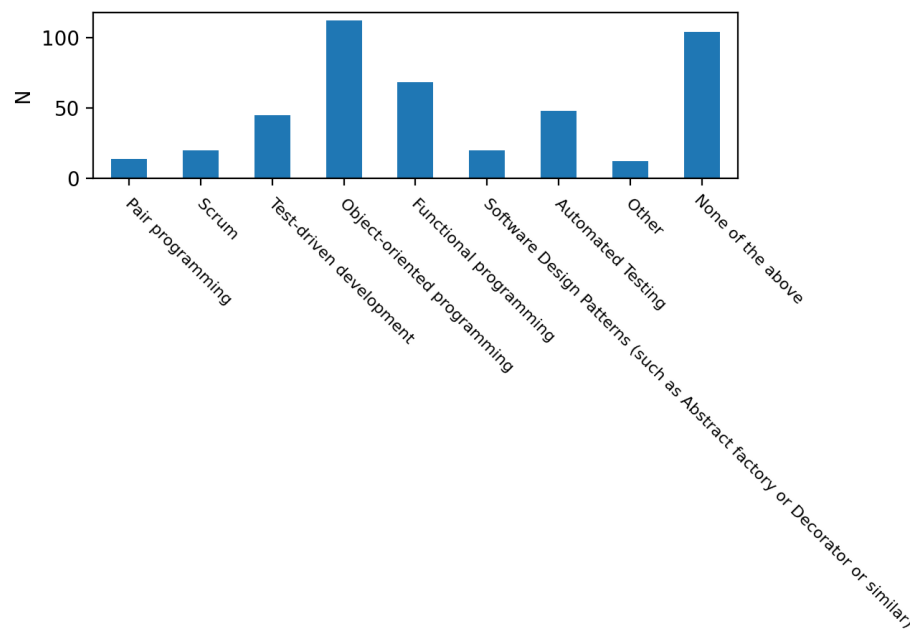
S104

Figure 16: S104 What methods are you applying?
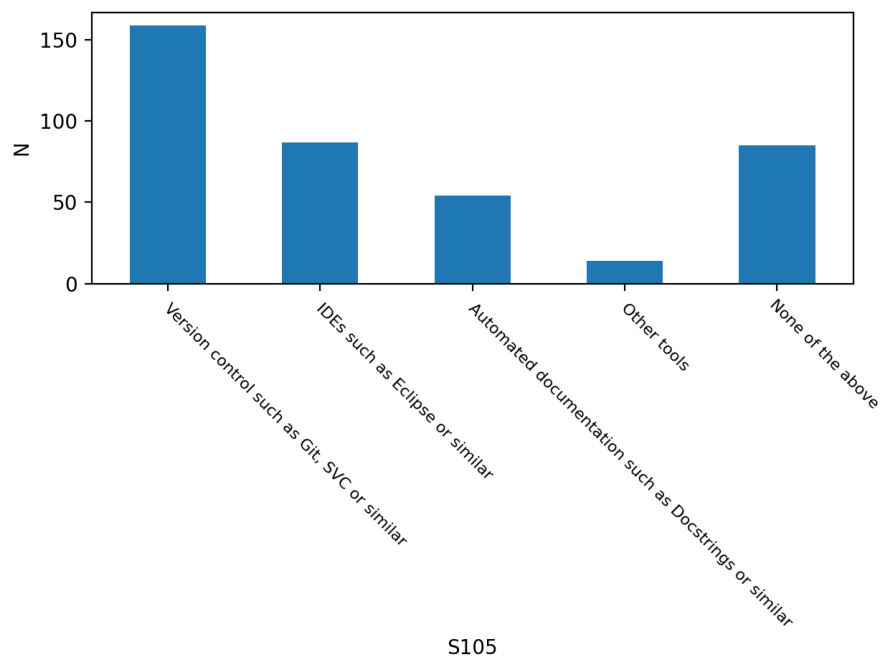
Figure 17: S105 What tools are you using?
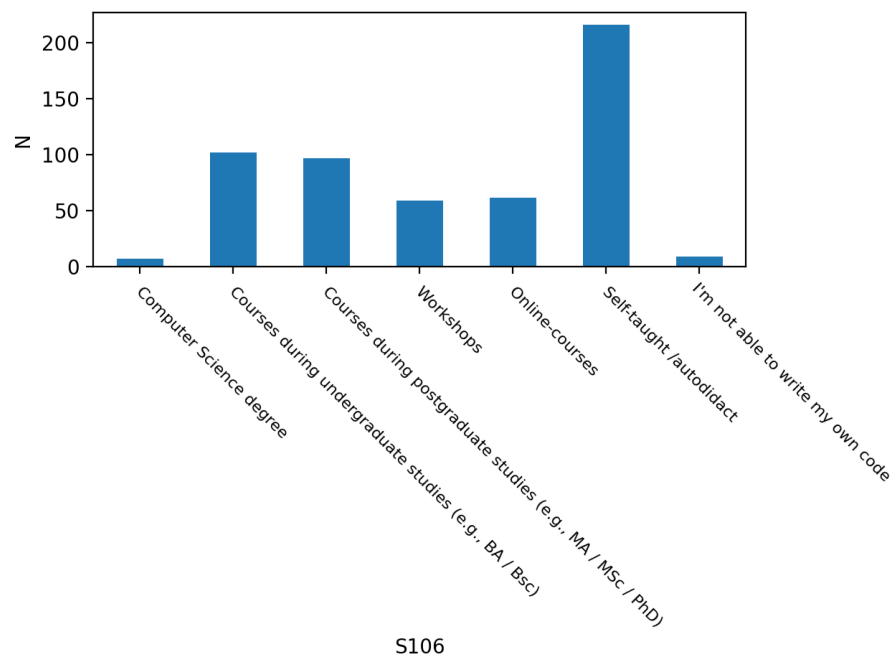
Figure 18: S106 What training do you have?

S111

What software licences do you use? Results: [None 'none' 'hfvjhgjh' 'GPL' 'MIT' 'Creative Commons' 'CECILL-C Licence (a French equivalent to the L-GPL licence)' 'Creative Commons licenses' 'None' 'MIT, LGPL' 'MIT / GPL3' 'GPLv3' 'Creative Common' 'GNU' 'Apache License v2' 'GPL MIT' 'GNU GPL v3' 'MIT license' 'CC BY-SA' 'Apache License' 'BSD' 'mit' 'GPL, PD, WTFPL' 'MATLAB' 'GNU General Public License (GPL)' 'Gpl v3' 'GPL, MIT' 'CC' 'mit, ah hoc' '???' 'GPL, MIT, Apache' 'GNU General Public License (GPL), BSD 3-Clause' 'Apache 2.0' 'US Government work - Public Domain' 'NONE' 'BSD 3-Clause' 'GPL3' 'Apache' 'FreeBSD' 'SWMM' 'BSD, GNU' 'GPL or CC-4.0 non-commercial' 'BSD 3-Clause, MIT' 'GNU 3, MIT' 'MIT, GPL' 'GNU GPL' 'educational' 'MIT, GNUv3, Creative Commons' 'GNU General Public License' 'MIT, GPL, or commercial (depending on use)' 'MatLab' 'BSD3, GPL' 'GPL v3' 'NOne' 'BSD-2-Clause License' 'GNU LGPL-2.1 License' 'MIT, BSD']

S112

Which of the licences are you familar with?

Other licences mentioned: [None 'CC0, Unlicense, WTFPL' 'LGPL' 'PD, WTFPL' 'None' 'Creative Commons' 'ArcGIS, Django' "ESRI's ArcGIS" 'MatLab' 'MIT']

S101

What kind of programming languages do you mainly use?

Other languages mentioned: ['Matlab', 'MATLAB', 'NCL', 'GAMS', 'matlab', 'ECMAScript', 'bash', 'Java', 'MatLab', 'Linux bash', 'shell', 'MPI', 'Stella', 'OpenFOAM', 'Delphi', 'PERL, MatLab, Pascal', 'Bash', 'Legacy (meta) languages in MS Office and ESRI: VBcode, AML (1981-1999), model builder. TBX ', 'Matlab and if possible I try to provided also compiled versione for complete reproducibility', 'whatever the students want to learn and /or makes sense for a particular project', 'Basic', 'Matlab, Julia', 'Octave/Matlab', 'Julia', 'Matlab, IDL'] **S104**

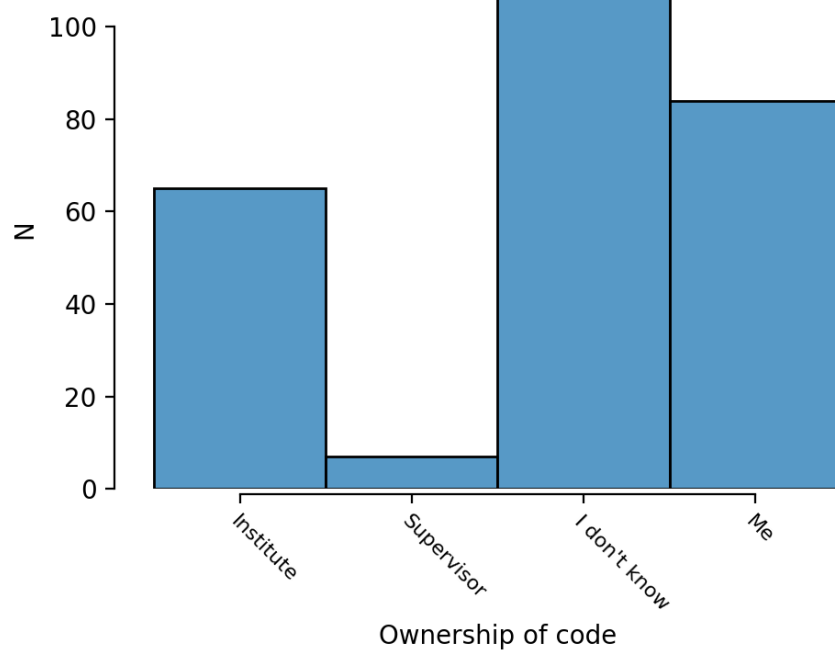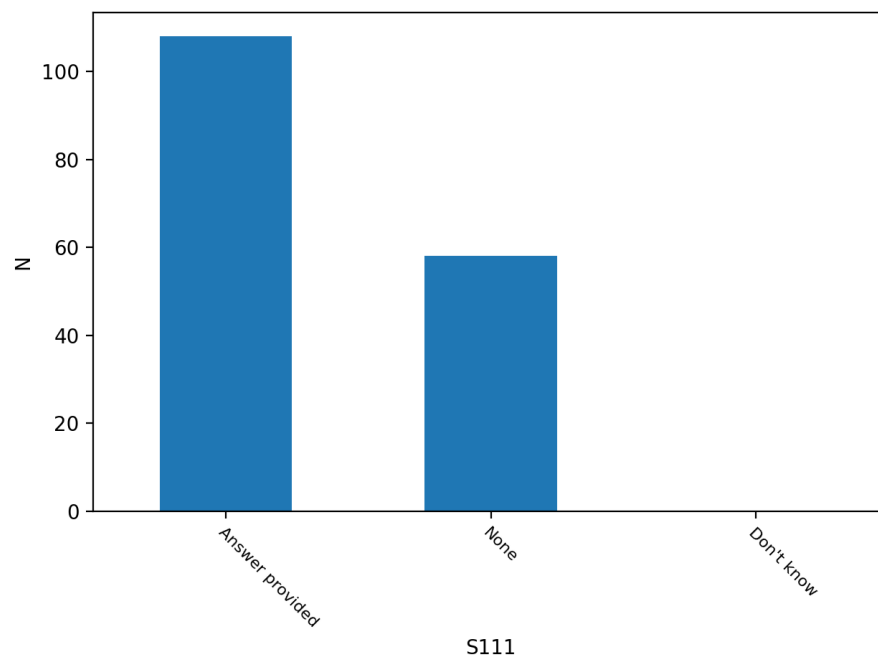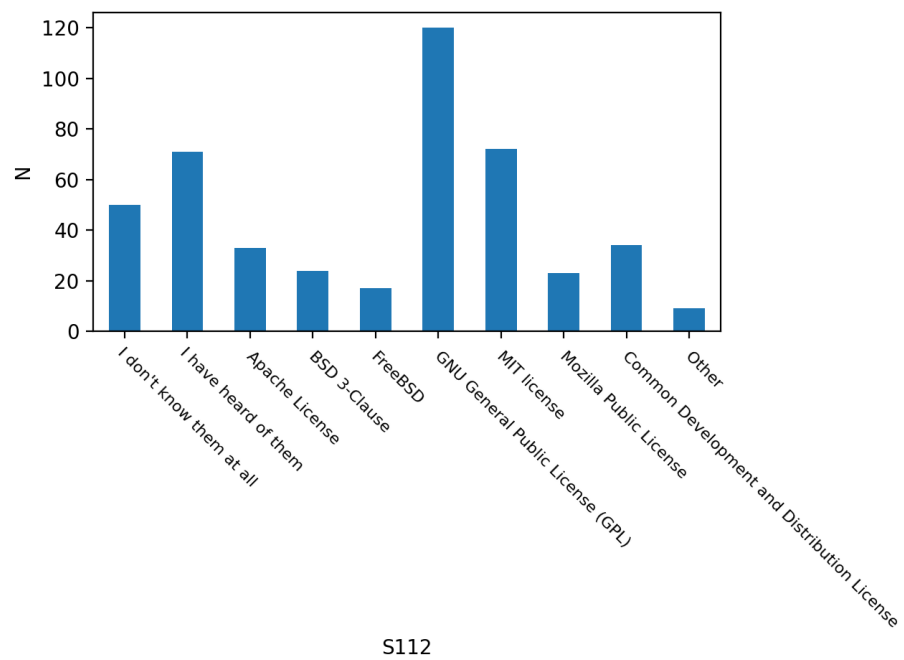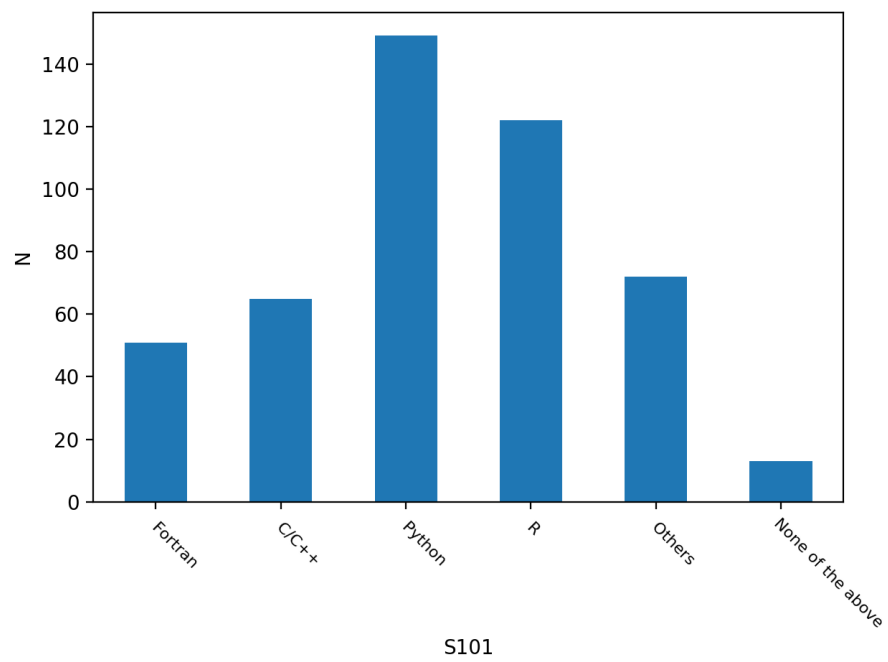What methods are you applying?

Others mentioned: [None 'Procedural programming' 'UML' 'Continuous Integration' 'Validation by comparison of numercial results with analytical solution or code-to-code intercomparison' 'None' 'Yelling at machine' 'git reviews with other researchers on the same project' 'continuous-integration' 'Maybe?' 'team code review' 'Open-source development (issues / pull-requests on GitHub, GitLab, etc.)'] **S105**

What tools are you using?

Others mentioned: [None 'Pycharm' 'Automatic formatting; issue tracker' 'ReadThe-Docs/TravisCi/PyCharm/GitHub/' 'Doxygen ' 'Code quality Checkers' 'Markdown' 'Continuous Integration tools: TravisCI, CircleCI' 'Cloud drive storage such as Box to maintain an archive of previous versions' 'No' 'jupyter hubs to make sure all members of the team work on the same distribution' 'Autoformatters, Pre-commit hooks, Test Coverage, ' 'Jupyter notebook' 'autoformatters, linters, test frameworks, continuous integration tools' 'Continuous integration']**S203**

What keeps you from publishing as Open Source?

Other reasons mentioned: [None "I don't write code" "I publish my code in out

institutions library network, which can be freely accessed, but I'm not sure if this qualifies per se open source." 'The model I am developing is not mine and the owners do not want to make it open-source. ' 'Code is unique for a certain data set. ' 'code not ready - still in development phase' 'I try to publish most, but often time is lacking to bring it into a shape that it can be used by anyone (all bugs checked, properly documented)' 'if possible in terms of time, codes have been published' 'Too shy, worried about code quality' "Haven't published any yet" 'I don?t know if it would be useful' "Haven't come to completion of PhD projects yet, but I plant to make everything open source" 'not good enough to publish' 'Supervisor opposition ' 'The exceutable files are available on the web as a package that is regularly updated' 'most of software has been developped during non-paid time and/or for personal consulting purposes. Some concern on giving away "for free" all of this immense invested non-paid time. Further, my code is relatively simple.' 'LPJ-GUESS model that we frequenctly use is not openly licensed by the main development group in Lund.' 'sensitive data/privacy and legal issues' 'I have not spent a lot of time thinking about the subject, but I do publish my code on github' 'Priority. I publish open data... I publish maps... I include scripts/code in some datasets... For others, it is so legacy format, that I archive it privately. ' 'not sure how and what are the standards' 'No specific reason']

3.3.2 H9 Activity of reproduction

See also Fig. 9
We mainly have answers from hydrology. Splitting and plotting the results proably wont get much additonal information that is valuable. Core for plotting this is implemented in the plotting script.

3.3.3 H12 Software ownership

See Fig. 12.

3.3.4 H2 Licensing

Younger = less than 20 years of experience.
Number of younger researchers which do not know about licences: 104
Number of established researchers which do not know about licences: 17

3.3.5 H7 The most frequently used language

See Fig. 15

3.3.6 H13 Models that are available are hard to use

Not possible to clearly answer this question.

3.3.7 H3 Young scientists are more familiar with modern development methods

Younger = less than 20 years of experience.
Number of younger researchers which do not know about methods: 86
Number of established researchers which do not know about methods: 18

3.4 Hurdles against and Solutions towards Reproducibility

In this part we are assessing what keeps scientists from publishing as Open Source and what we can do do increase reproducibility. We expected to see lack of funding as mean reason and a main solution to the reproducibility problem. Corresponding hypotheses

- H19 We need more funding to enable reproducible computational science.

Corresponding survey questions:

- S203 - What keeps you from publishing as open source?

- S201 - What would help to increase reproducibility?

- S204 - Open text - what can the community do?

3.4.1 H19 Funding necessary?

See Fig. 20
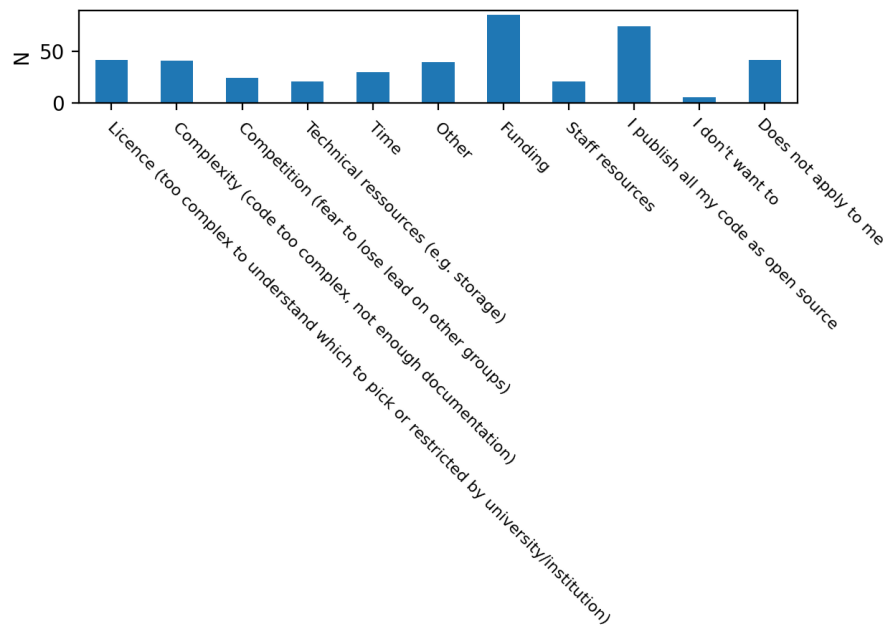
Figure 19: S203 What keeps you from publishing as Open Source?
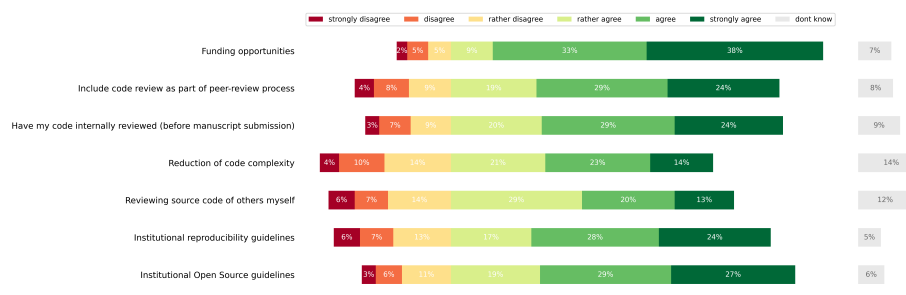
Figure 20: S201 What would help to increase reproducibility?, First question mean is at 4.95

3.5 All fulltext answers

This is a manually cleaned collection of the full text answers. If you would like to access the uncleaned data use the S204.csv file.

The following organizes the answers into overarching ideas that were suggested. If an answer suggested multiple ideas it was split. But no content was removed!
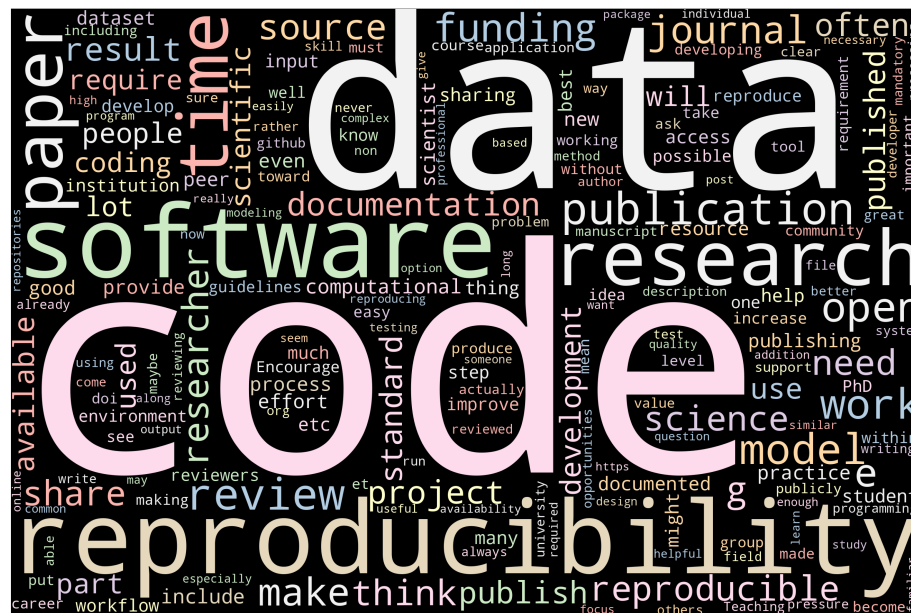
Figure 21: Full text answers as word cloud.

It is complicated

- In the climate community this is not very easily addressed. Reproducing the output of global climate models would be absurdly time and resource consuming. Reproducing the paper figures based on the author's code and data used is doable, but is it complete reproducibility?

- The emphasis of reproducibility and novelty is killing sciences in the developing world, especially in over-exploited complex GW systems.

- I am overwhelmed with the endless variety of licenses, programing languages, sharing repositories, the constant migration towards newer approaches that make older ones obsolete. I don't know how to, but if this could be streamlined it would be most useful to me. The community would need to agree to some standard sharing norms that we all should follow.

- Thanks for taking up the idea. Really important issue. However for me reproducibility in modelling in hydrology would not mean just coding

issues, the questionnaire is a bit biased toward formal criteria concerning coding. Reproducibility of research in mathematical modelling (including statistical models) in hydrology is a more complex issue. I doubt, that even being offered the codes, all raw data and a detailed description of the work done, it would be easy to reproduce someones results nowadays. These include subjective interpretations of partial results and decisions taken based on these. Hydrology is not a lab science.

- I am not so sure that in our field reproducibility MUST be a condition for publication. It is not only a matter of considering the code. It works with data, which, depending on several circumstances, might have been acquired to a very high cost in financial and physical terms. Sharing this data might have certain non-convenient related aspects. For instance, multiple analyses might be associated to a unique data set. Writing a manuscript, depending on job-related conditions, may take a while. In the meanwhile, between a published work instant, some other research group might be working with ""your"" data and publish one of your ideas, before you, who actually collected the data; all of this, just because for publishing your past research you were obliged to share all of your data and code just because of ""reproducibility"". Further, it is my oppinion that ""doubting"" on our research results just because we do not share everything, including software and data, is not that correct, nor that logical. Shall we doubt on what has been already published?... on the work of Beven, Refsgaard, Abbot, etc (hydrology)? Beven have shared TOPMODEL, most probably becasuse he was always interested in so, but al;so because its original development was financed explicitly with that purpose in mind... The Danish effort of Abbot and Refsgaard, MIKE SHE, is commercial... what will happen when I do research with a commercial software and cannot share it... even if I share the data, the research won't be 100% replicable.... does it mean that the research is bad and as such non-publicable?

- Funding dedicated to this would be great. Stricter requirements (e.g., code review with publication) could be useful, but I worry about unintended consequences and the practicality - e.g., might become more difficult to find reviewers, more difficult for early career researchers in computational fields to publish relative to competitiors in non-computational fields.

- recognize that just sharing the code does not gaurantee reprodicibility. Results are very dependent on libraries used, often even operating systems used (famous example where MRI results were treated differently on mac vs windows machines in genetics research because the file sorting algorithm was different for those operating systems) It is therefore - important to consider the entire compute environment used when working towards fully reproducible results - try to keep this compute environment as lightweight, portable and machine agnostic as possible. Currently containers are a great way of achieving this.

- The COMPLEXITY of reproducibility is often underestimated. This issue has been a major topic within the social sciences, and there now exist a nice collection of resources that necessarily reflect the broad range of what is considered ""reproducible"" and there are insights and resources portable to the geosciences, see e.g. https://www.bitss.org/

- Blind peer reviews seem to be the most essential step for any science publication. I have experienced too many ""names"" getting their work published without merit, and it seems that would be all the worse in publishing of models and code.

- Several of my colleagues have sepnt their career to develop certain codes for their own research. They consider these as their livelihood, and would never surrender them to others.

- I have not any idea to propose, but it is a serious problem

- I think a code review as part of peer-review process would be very helpful but not possible for, e.g., large hydrological models.

Share more and share data and model together

- Foster putting codes publicly available (e;g. through github)

- Share not only model, but when using application: (1) Share the input and settings you have used to run the model. Share (if possible) the dataset that you have used to compare the simulation results with (when comparing simulations with real data).

- Publish data alongside research papers. With the documentation contained within the paper, the results should be reproducible.

- Publication of code and (exemplary, applied) data with the manuscript as a standard ? but also acknowledging the extra efforts required for this.

- A more detailed description in the paper and a share option for code "share code and data with others, stronger collaborative efforts rather than competitive mindest, reduce the amount of code developing by individual working groups and rely more on available codes and community efforts for code development for similar questions and applications.

- develop and maintain open access databases

- Agree on common tests for code, models.

- Encourage open source code and compiled tools for reproducibility and wider application

- Make data and code publicly available and thoroughly documented

- SHARE! Share your data, share your code!

- Share datasets used

- First the data should be published along with the publication, then comes the code. Code without the source data does not really help.

- Share everything

- Share all model code with sufficient documentation. This documentation should also include an overview over the structure of the model (which files are used for what) because otherwise it is very hard to understand. It often takes me a long time to find the model code online and if I find it, it is often very poorly documented, which means you would have to reach out to the authors for an explanation (e.g. the very often used PCR-GLOBWB. I think there should be more funding and credit allocated to documenting code.

- Hmm... I believe that code is often rewritten by many researchers. As a result, many resources are wasted on developing new code. While the creation of new code is important, especially for early-career scientists, I think great resources should be put towards developing generalizable and robust code that can be applied by many

- Provide open access codes and documentations. Publish codes and tutorials

- I think in general we shall be more confident about the codes we use and develop to share them also more openly. More education and workshops about coding would help.

- Well documented applications that also include e.g how to set it up on eg a windows computer. Only this way this becomes accessible to users rather than staying in academic institutions.

- Improve sharing culture of open access data - data is often the problem

- Models needs data and produce data. data should go with the FAIR priciples

- Providing step-by-step supporting information besides the results. Listing all the assumptions.

- Encourage publishing of code online in repostries like github.

- One challenge that I see with reproducibility is the need for openly accessible data. Individuals who fund field campaigns that generate data feel somewhat ""used"" by modellers who do not obtain funding to generate data and in some cases expect that all data should be freely available. Essentially, if one researcher goes through the trouble of collecting and funding data why should the rewards of their efforts go to others before they have had the opportunity of fully exploiting that data. Additionally,

if you put a temporal embargo on data availability so that data is private for some duration this may prevent students or PIs from publishing findings involving that data within that time period because of journal data sharing requirements. Perhaps using synthetic data is an option for demonstrating analysis and validity without exploiting data generators.

- It's not only about codes but also about data which should be published along with the code. I think the past decade has seen some progress in making codes publicly available.

- Show your work in appendix, attachment, online, etc

- - We need to have an honest discussion about why people don't share their code. My impression is that the answers are similar to why anonymous reviews should be maintained. - Researchers are rarely required to produce good code. I mean when was the last time someone took your code apart and tested it? And producing good code takes practice, but also the need to want to write well-structured and well-tested code. I think the problem can only be improved if we accept that all of us are part of it and that all of us probably produce research output that is not reproducible.

- Share codes or softwares but overall share the input files. Do not publish papers that relying on code and data do not provide them. Stop using proprietary close-source software. Ever.

- The models that we use need to be open-source. Also, sharing the scripts (e.g. via GitHub) that we use for data post-processing would be useful and maybe even necessary.

Check during submission and journal responsibilities

- Explore a badging scheme for published papers that include reproducible code or data.

- Most data are already provided within the most publications. So why not using a software tool during the submission to check reproducibility of results based on the code and the data

- Journals should enforce FAIR not just get everyone away with it

- Submitting for publications and working on open source codes could be made the norms by the editors.

- code repository connected to scientific journals where the modelling study is published

- Publishers, Editors, and Reviewers should require all source code to be made available in open repositories (if necessary with a restricted access like Zenodo)

- Increase mentions of code sharing and research reproducibility in research publications, include code review in peer-review process.

- not accept papers anymore that obviously have something to hide or are completely unwilling to share.

- more critical peer review on methodology/code (when supplied as supplementary material, no review is required/possible)

- When reviewing a manuscript (mine or other), I ask myself, can I make the same model from the information in the code? Are inputs explained? what assumptions were made? Making these things very clear can help.

- I would love to discuss how to standardize reporting model setups. I think this should probably rather come from journals.

- Include all codes during publication of work, should be mandatory by all journals.

- Move beyond the static paper and encourage to have the code as supplement, or do things like jupyter notebooks in addition or instead of a paper.

- Encourage / demand data and software to be published along with research papers. Strengthen the role of peer-review (e.g. stronger acknowledge Publons) and post-publication review (see Pubpeer).

- A big help would be to have code available as frequently as possible and it being part of the review process. Datasets should be easily made available. When those cannot be shared, present synthetic series and results to be replicated by the community.

- only reproducible Research for publication

- Include more code in papers and describe it Better/required documentation

- Journals should mandate reproducibility and include it in the review process.

- We are moving towards the right direction. Code should be attached to publications along with ALL input data and the reviewers should be able just to run it to see if it works, if it generates the same outputs presented in the paper and eventually to check if all steps are correctly implemented.

- Just share the code with good documentation and comments. So many high impact journals have no code documentation at all (even recent ones, despite apparent journal requirements). There is a huge burden of time for researchers to do this, though.

- Impose to publish with open source model codes

- Every paper should provide code and data for running and plotting every figure in it.

- A reproducibility-test should be part of the review-process.

- Encourage code releases or links to source-code repositories as part of the manuscript acceptance/publication process, similar to how data is now commonly required to be released. There also needs to be an acknowledgement of the importance of releasing code in funding opportunities, and specific funds allocated for that.

- Move towards requiring all source code and data (that does not have legal restrictions on its use) to be published or deposited in a publicly available respository for all publications.

- Journals should require code/data for publication where possible. Code review would be great, but it would probably be too time consuming for volunteer reviewers. Maybe journals could hire some data scientists to just check the code for correctness as a separate part of the review?

- Journals should move towards requiring experimental reproducibility as a part of criteria for publication. This means abandoning the publish or perish model as a basis for professional advancement. To enable this, collaborations to assess reproducibility should be professionally rewarded.

- Rolf Hut (yes, I know I broke annonimity...) ": "Journals should push for open and reproducible code and open data. If researchers do not want to share their code/data, they should be able to give a good reason. While clear guidelines are helpful, it should also be made relatively easy. There's already a lot we have to do as researchers, so there's limited capacity to also become a ""professional software developer"". I think being too demanding (e.g. as a journal) might put people off, so we should balance the requirements with the effort they involve. That's also true for reviewing. Code reviewing is generally a great idea, but it will make reviewing more time consuming. Then I think we should make a distinction depending on the type of paper/code. If it's a paper that presents a new model, for example, then I would require a more thorough code review compared to a paper that just uses that model to produce results. Generally, we should distinguish between research software, i.e. where the focus is on the software itself, and research that uses software. Well written and documented code is always helpful, but quite often specific code used to setup a model for a specific case study simply isn't reused that often. So I think it is potentially unnecessary to expect the same standards (e.g. documentation) in that case."

- enforce transparency policies with journals.

- ask for providing codes during article submission

- Require code to be published/available if it is used/referenced in a manuscript.

- Pay people who conduct peer review, and require reproducability be taken into account in the peer review process

- Put reproducibility as ""mandatory"" in the publication of our scientific results

- Make code publication mandatory, including working examples (at least small sample dataset). As witht he EGU copernicus journals, require data publication on some repository. If the latter is not possible due to data restrictions, require data repository of processed dataset (e.g. alternated values - same data disstribution and trend ...) and the entire post-processing workflow.

- Demand that the code is peer reviewed by qualified peer reviewers

- Have our journals require access to all code just like access to all data.

- I think we should put more emphasis on open source code sharing during publication. Now almost every journal asks to give a statement on data availability, they should do the same with code availability. Then it would be much clearer to the readers where to find the code or that the authors give the posibility to ask for the code.

- All computation-related publications should require codes to be available (including all relevant inputs) at the time of publication, with the option to embargo the ability to download those codes for a short duration to protect from other research getting scooped. Introducing software methods in papers without including the actual code leaves the work of reproducing those codes up to less skilled and uninformed graduate students, who are much more likely to apply the methods erroneously. Nobody doing active computational research in our fields should be publishing computational research for which the codes are not available, and the culture needs to reflect that.

- Make it a part of reviewing process for publications

- Recognize that papers, code, and workflows might all need individual review and that each of these reviews take different skillsets. An example might be paired publications between a traditional journal and a journal focused on the software such as the Journal of Open Source Software).

- Be part of the publication process. Availability of easy to use tools to facilitate reproducibility (e.g. cloud environment).

- Mandatory data and input files submission during submission to a journal.

- More options for publishing software: a DOI is not enough for 1+ more years of research. ""Classic"" journals are generally not interested in publishing software papers.

- Force authors to provide the data and their code on publication.

- And have th code reviewed by dedicated code peer-review.

- Force journals to allow the publication of reproducibility studies, which, in sedimentology/basin analysis, is basically impossible.

- asking for code / data avilability declarations (as some journals do by now) is good but code review should also be enforced (in my case it was never done even when asked by the journal, e.g. GMD)

- code review would make review process even more time demanding -¿ publishers could pay reviewers (I often wonder what APCs are actually good for)

Need for specialized personel

- We need people who can help to improve reproducibility, as many researchers with a focus on analytics and conceptualization are not familiar with coding and licenses. Still those researchers have great ideas that should not be ranked lower just because those people are not familiar with this new, necessary movement

- Invite experienced software engineers as consultants for model software development.

- Hire more research software engineers that oversee sustainable software development practices and help with standardising, testing, and publishing research software.

- support more software engineering support to researchers

- Speaking for myself, I have never had formal training in code development, so I assume my code is usually inefficient. But it works for the task at hand. If I knew that there would be a large group of people wanting to use or reproduce my model, it might make sense to work with a programmer/developer to clean, optimize, and document the code, and deliver it with sample datasets. But my perception is that there are few people who would bother to try to reproduce my work. I do make an effort (when public data is part of the model inputs) to provide the code in a reproducible format like an R Notebook. But the majority of my code is theoretical and has no external inputs.

- Funding opportunities are great, but are yet another thing to apply for and maybe or maybe not get. My biggest struggle is trying to reproduce a result by following the model description in a paper that does not also include the code. Often I am not able to because the description is incomplete. But it is a difficult thing to catch in review because it is so time-intensive.

If the code is released alongside the paper and is well-documented, it can be tested much more efficiently, and discrepancies between the code and the description can be caught. But even that is above the level of time investment of a typical journal review. Requiring code be made avaiable to reviewers (even if not released with the publication) would help.

- Have more software engineers (scientific software developers) active in research teams (i.e., part of the section staff, from other departments or eScience centers, external collaborators, etc.). Software development is a very useful skill to develop for a researcher, but more and more an additional requirement that adds a lot of burden on researchers alongside other skiils like admin, management, teaching, writing, etc. Software becomes more and more complex (design patterns, handling big datasets, running code on cloud/HPC, CPU/GPU, managing dependencies, legal aspects of publish code, rapidly evolving software ecosystems, interactiing with open-source communities, etc.) and it is too much to rest on the back of researchers / scientists only, in my opinion.

- Involve software developers early in the research workflow (i.e., as soon as writing a new projects, not for cleaning the code just before submitting or publishing a paper) so that they can work jointly with researchers on projets, focusing on software design, offering personalized support and giving advices on the technologies that are best suited to a given problem. This is, in my opinion and from my experience, very effective (much less work burden, mutual motivation and suggestions) and highly rewarding for both software developers and researchers.

- Offer more career development opportunities for scientific software developers in the acamedia.

- increase best practices workshops, provide a programer for each research group. Sometimes, the time is not efficiently used because researchers with poor programmer skills use more time than an actual programmer.

Teaching

- A lot of today's scientific endeavor is performed through a process akin to software development yet there is a blatant gap between the standards to which scientific code and industry code are held to. For the sake of reproducibility and as a service to many graduates that will pursue a career in the private sector, industry standards should be taught and enforced. Publishing code and data is not enough if the authors are the only persons able to make sense of it. Versioning, packaging, long-form documentation, testing, continuous integration and deployment should become widely accepted standards in scientific software development as they have become in the private sector.

- Teaching reproducibility (with respect to software, data and measurements) in environmental sciences.

- provide better material to learn about reproducibility and it's benefits, standardize the teaching of coding skills and their ""best practices"" in MSc or at the latest PhD programms, allocate TIME (and thus money) for making work reproducible

- Often clear guidelines and courses for beginners on how to ensure reproducibility from the very beginning are missing. For a beginner the different steps and tasks can seem overwhelming. Teaching from the very beginning and setting out the rules for a minimum standard could help improve the reproducibility.

- We need a lot more training.

- Dedicate a few months at the beginning of a PhD or Postdoc position for a crashcourse in professional programming.

- Mandatory crashcourses for senior researchers and professors in how to *evaluate* code quality. If journal reviewers and supervisors have no idea what good software looks like, it will never be rewarded.

- Universities should teach how to increase computational reproducibility.

- Provide resources to help with the open science

- build on scientific code of others, and develop it differentially (in the sense of changesets).

- Give researchers the resources they need to learn to produce readable, well documented code and have consistent standards across journals regarding code and data availability.

- More classes at university level on coding and/or workshops for more advanced researchers

- Continous training is very important for awareness creation.

- Include these concepts in courses taught to undergrads and graduate students

- There should be more courses on ""code documentation"" and ""publishing your code"" e.g. as public packages - not only at the university/institution level but internationally as well. From my experience, most people code for their own purposes, but they are willing (or even happy) to share their code, but they feel embarassed of its structure and functionality. To overcome this, they do not only need to learn the best practices, but also how to publish code; from small packages for simple tasks to complex packages for complex tasks. For the latter, often time to maintain these packages is lacking ? this should be accounted for somehow.

- We should develop curriculum for reproducibility that grad students should be encouraged to take. We should have seminars at conferences devoted to reproducibility. We should encourage the creation of reproducibility working groups at institutions that can serve as a resource to their colleagues and offer occasional workshops

- Need to include more programming / coding classes at the undergraduate level. Tailored to earth and environmental sciences. Don't learn coding when you are a postdoc!

- More courses on e.g. GIT.

- Train students early on proper computer science techniques, methods, and documentation. The biggest problem is that few of us really know what we're doing–we write code that is confusing and poorly documented. I also think that if there were more pressure to publish open-source, it would encourage researchers to provide better documentation.

- Design curriculum for a complete course(s) to be taken as upper-level, but definitely by grad school. Have sessions at AGU centered around doing science for scientists and science - reproducibility in programming is just one arm; it also needs to be discussed for collecting data, field methods - students are not taught even how to think in terms of designing and communicating workflows, so we have a long way to go.

- More support and opportunities to work with someone good at coding to improve my own coding and to edit it to be more shareable. Also requiring all undergraduates and graduate students to take a class in coding and documentation. I'm such a hack because I really was just picking stuff up ad hoc. Without knowing how to code, you are a little illiterate in this day and age. So, requiring coding like we require writing courses makes a lot of sense.

- Better inform researchers about open-source code publishing (lots of researchers are still afraid of publicly sharing their own code and/or that someone else will steal their code).

- Also, I think improving the testing of models should be a focus point. I have the feeling many people do not know how to write tests that will automatically be run for every model update. This makes code very prone to errors that may remain undetected for a long time. I think there should be more workshops given on this topic and software testing should be compulsory for models that are widely used.

- higher emphasis of code development and reproducibility at university (already at BSc level)

- offer teaching on useful tools for reproducibility of computational research in study programs (e.g. version control system, automated tests, ?)

44

Practices

- Shadow or Pair computing could be made the norm.

- At my institution, we are lately developing a strong focus on reproducibility. Especially for young scientists who are starting their projects, e.g., PhD students (in particular those that come from a non-programmer background) this topic seems to be overwhelming. They are struggling to get familiar with the models, with programming, with their research projects, and they are now also asked to strongly focus on reproducibility, ideally to a level where they think about Continuous Integration, Conainerizing of models and scripts, and extensive documentation. This is a lot to ask of a beginner. What we often hear from them is that they are already severely time-constrained and would like to dedicate more time to reproducibility, but don't see how they can do it within their allotted time for their project/thesis. Reproducibility should become more of a standard procedure in modeling-associated research, but to actually make it happen, it seems that it needs to be planned into the design of the research projects right from the start, to ensure that there is enough time available that can be dedicated to make research reproducible, and to teach (young) scientists who are not yet familiar with the concept and the means to implement it.

- I would recommend reproducing the result by a colleague not part of the concerned project/paper before submission. I believe this will increase the confidence and reproducibility of the research. Communicate the workflow

- Require all code open source.

- Sharing workflows for modeling, with *all* assumptions. Often, scientific modeling workflows have un-documented steps (e.g. manual data cleanup) that prevent reproducibility - and these are places where errors easily creep in.

- We should also normalize code review *well* before the peer review stage. These are standard industry practices with lots of evidence to show that they reduce bugs and improve the development process.

- Publish code as publication.

- Advocate for more transparency and documentation of code, making it readily and easily available as a extra resource in peer-reviewed publications.

- Present your code to your lab mates, receive critiques

- Formalize the conceptual modelling steps prior to program coding

- Code more readable and simple and use online computation environments to share their own simulation environment, e.g. MyBinder

- Code review within my research group has been an incredible process for improving the readability, structure, and best practices of my code. Code review events or conferences might be a good way to improve reproducibility.

- Encourage standards / best practices for developing, documenting, and archiving code, including scripts for pre-/post-processing inputs/output data.

- Emphasize documentation, read-me?s. Transfer of information after someone finishes a PhD is so lacking because people are normally scrambling to finish, and don?t think to put their models in a shared drive

- Include the code as annnex to the manuscript with a clear description of the steps used to get to the presented results (i.e. imputs, paramenters type and values, followed procedure, exceptions, approximations...)

- All the inputs used for the computation must be rigorously described, and also the methodology for make sure that the produced results are correct (e.g.: convergence studies in CFD)

- The computational cost of replicating a global, continental, large-sample hydrological studies is too high. A sollution would be to create a demonstration dataset, which is a subset of the whole study, for testing reproducibility.

- Develop community standards for transparency and open-source softwater development (especially in the research community)

- Mostly an attitude values thing. - Value publication of workflows and data similar to scientific findings - Encourage the citing of dataset/code DOIs (in addition to any papers describing said data/code) - Move away from the ""I needed time to create this code and I don't want others to benefit from my work"" attitude (by doing e.g. the two things above)

- Have more allocated time (i.e. and that relates closely to funding) to prepare codes that are relatively easy to read by others in order to be able to share them.

- Careful consideration of what artifacts must be archived for reproduciblility (e.g., input files, pre-processing steps, etc) and appropriate locations for such artifacts. The EarthCube ModelData RCN has made some nice progress here. https://modeldatarcn.github.io/rubrics-worksheets/rubrics.html

- A potentially controversial comment, which comes from the perspective of a research scientist who often develops software for new applications rather than using software: I think all computational code, computational environment, and worksflows, should be documented such that the author can reproduce their work, and such that an independent individual willing

to put in the effort can reproduce the work. When the investigator is doing science, creating software, and undertaking the computation, it is too much to ask that all work is documented to a high level such that it is EASY for an independent individual to reproduce the work.

- Publish papers as open access. - Always publish code and data with papers under an open access license. - Open source climate models.

Institutional support and change

- The science community and funding institutions should put as high value on a good sceintific code - reviewed, documented and published - as on scientific papers. This actually also applies to research data that are along with software less valued than a paper. This requires a shift in a mindset which has started, but will take some time (years-decades) till we are there. You can do a PhD with three published papers, but not with a paper + a documented open-source code and a documented doi-Dataset! This is a point where to start change things. Of course, this may open door to easy-way PhDs consisting of a few subroutines, an Excel-Table and a low-profile paper, but this is what the supervisor and finally the PhD commission should look at.

- Universities should provide service entities which guide, support and review scientific software development to avoid the autodidact diversity of styles and mistakes.

- Universities should recognise Research Software (and related papers in journals such as JOSS) as publications and part of PhD theses.

- Each institutions/university should make reproducibility a priority, create data bases like OpenABM website to upload open source codes.

- Guidelines for reproducible software generation, not just reproducible research in general, would be helpful, in addition to funding for improving established research software.

- Institutional know-how passing on and being updated. Organize within the institution. Create side projects within the institution where researchers can actually work and publish their code together. Make these side projects mandatory.

- Make reproducibility more understood by management and funding agencies

- Having an AGU or University wide standard to follow in our discipline would be nice as an early career scientist in order to follow best practices.

- Offer more guidance and resources to groups from institutions.

- In my country (UK) there is very little funding for good coding practice and almost no professional incentives for such activities (i.e., reproducibility and production of open source software has never come up in an appraisal or promotion case for me). Maintaining software for the community is a lot of work and my employer would much rather see me write papers with that time. Universities like citations: more standarisation of software citation might change this balance of incentives. There needs to be more funding for research software engineers.

- Best pracctices in universitites/groups.

- enforce clear and transparent guidelines by employer / university

General change, time, funding, recognition

- Provide best practice guidelines, funding, and recognition for reproducible science. The current currency is publications, not well documented, tested, and reproducible research.

- I think that time is a very important factor. Many scientists work in projects that up to now often do not allocate time to the reproducibility process. So the time to invest in reproducibility competes with time to produce project results. I think it could help to actually allocate time (and thus funding) to making science reproducible in project proposals.

- And we need to acknowledge that this is a considerable time investment for PhD students and post-docs who are the main scientists developing code. Current projects have very tight timelines that are often not feasible because of poor planning and poor time management - this is without investing time in reproducibility. Time devoted to reproducibility needs to be factored into project proposals and projects need to be guided with a Scrum framework that allows scientists to quickly see what is working and what is not, and to promote working in teams (which would facilitate code review). A lot of coders and professors I know work as 'lone wolves' - but making code open source and reproducible will require us to help each other, work in teams, and promote a spirit of openness. Right now, no one in my project would have the skills to review my code. I would have to ask other lab members not paid by my project to do the work which is asking for a lot of time, especially since this person would not be familiar with the scientific goals of my model and the questions it is supposed to answer. Reproducibility is incredibly important and I think most people want to improve, but we need to be given the environment where we can make such a time investment.

- Reduce publication pressure! Quality must come before quantity.

- Code and data publications should be treated as valuable as/equally to papers (including funding opportunities, best practices, peer-review obligation, importance for applications for scholarships/positions).

- funding for staff / coders in addition to funding for scientists.

- increase funding

- Common standards have to be checked regularly. Many manuscripts are published too fast... My supervisor (habilitated years ago) often told me to do / code / calculate things I know to be absolutely wrong. I refused to do what he asked, because he did not want to make research reproducible, but to make his papers be published very soon. For him its much more relevant that his studies are published fast, not that their content makes sence and is reproducible. I will leave science as soon as possible - I cannot longer support this.

- Require reproducibility by funding agencies.

- appreciating formulas within papers more - necessary to increase reproducibility

- Include software and data reproducibility requirements into funding guidelines of major funding agencies.

- eward more the work that helps achieving reproducibility! for istance at the moment a paper presenting a new, clearer implementation of existing models/methods would have a hard time being published (reviewers would say ""it's not new!"") though it could be very impactful. Whereas a paper presenting a ""new"" model/method, which maybe is only a small incremental addition to an existing one, will be much easier to publish, and even more so if it is coded in a ""new"" package, which however only add to the complexity and fragmentation of the landscape!

- Value more in CVs as peer revirewed publications code publications, code repositories, documentation writing Provide more founding for this in applied science research poposals. Need to have permanent staff in research institutions to support researcher for code reviewing documenting sharing.

- Run collective projects trying to reproduce selected studies

- Develop funder requirements that scientific code and data be publicly available. i.e., treat software similarly to data and technology, with a degree of public accessibility for publicly-funded research, but the opportunity to protect and commercialize resulting IP

- maybe give more credit to work/effort of developing model/code.

- more funding sources to improve reproducibility

- Include/require resources in project/grant proposals for making research reprodicable.

- Funding programs offering more funding to develop software for science. Often there are no good funding options to follow up on code developed as part of science research projects. If funding for software for science were more broadly available could have better maintained, accessible, and usable software.

- There is must be a plateform globally where students from all over the world may interact each other about there similar research projects and this plateform must be supported from professional institutions like in form of scholarships/fellowships etc.

- The most effective measures would be to change the incentives and provide learning opportunities. Currently, the incentive is to publish a lot and not worry about quality/reproducibility. As long as we are measured on how many papers are published in top journals, reproducibility will be at best a low priority. If funding/hiring/promotion is based on quality and reproducibility, then we'll invest in that. Teaching skills like https://software-carpentry.org fills in the gaps to get people to this standard.

- I think that a stronger recognition of computational research outputs (script, software, testcases, dataset, etc etc) in terms of career visibility/indexing/scoring is needed. Otherwise researchers have no time and resources to both publish ""standard"" scientific contributions and also reproducible comp. research. This has to pay back in terms of career development and cv, so it needs full recognitions by Insitutions, Journals and academic performance metrics.

- We should all have more time at hand and less pressure to keep publishing.

- Make reproducing other work a minimum standard for tenure review or graduate student review. The pressure to produce novel results to the exclusion of replication will never permit strong motivation. Funding agencies also have to act in concert.

- Funding for this opportunity.

- Value the time it takes to document code and computational workflows - and these efforts must count towards promotion in the same way as journal articles.

- Motivation in the form of credit or publications. The pressure for publication is insane these days. If you don't have a permanent position, putting effort into maintaining software for external users, many of whom lack relevant skills, means losing time to publish papers and keep your job.

- Wrapping up data, code and results in a reproducible manner requires lots of effort. Lack of funding/time for that might be the main bottleneck for me

- And convince grant people that it could be beneficial to pay for reproducibility study, because it is impossible to get funding for ""redoing something that was already done before""."

- make it compulsory, provide time and money to make computational reserach reprodubile, reward the work

Guidelines

- My use of code pertains to processing data, modeling (using a mixture of existing modeling programs + my own analyses), and plotting. So, my response here does NOT pertain only to software development, but the full workflow of computational research. That said, what would be helpful to me in making my code more reproducible would be clear and simple guidelines and templates – for the specific code environment (e.g., R, Python) that I use – things that REDUCE time and effort on doing reproducible/open research, not things that increase time and effort for me and students.

- Best practice guidelines specific to the use of particular kinds of codes (e.g. computational fluid dynamics). But such guidelines need to be sensitive to the kinds of questions asked of models in the geosciences and not the kinds of highly idealised problems that are characteristic of some applications (w.g. where boundary conditions are perfectly known).

- We need a ""standards and ecosystem"" approach. Scientists need to be cognizant of how their software will work with software and techniques from other scientists. This means established formats for common data, and tools for completing common data-processing steps.

Other

- Hybrid research with computational and data collection/creation and data product production goals... Besides writing papers.... They suffer from time pressure to fulfill ideal standards of documentation and have everything reviewed. Time wise this is not possible and it is a perverse thing to ask to young researchers unless also older PhD thesis requirements are dropped in number of chapters etc. I also think that open data should be 'reviewed by usage', after its publication, and be allowed to be corrected/updated/version managed without having to make a new doi or whatever.. 'living' data or 'living code' (services) is/are more important than published static data and code (archives)

- Publish program in open source code

- I would love people to point out positive examples of reproducibility.

- develop slowly an obligation to improve reproducibility

- nothing

- no suggestion

- Not sure

- no ideas

- no ideas

- not idea

- none

- None

- I don't know!

- I don't know.

- Open Science

- not sure

- N/A

- Make a common aggreement

- Make sure it works as described

- I am not really sure why you think this is necessary

- organize respective hackathons

- Conduct more seminars, webinars, conferences.

- Open Repositories; Foster Open Source as good practice

- Be honest

- use open science

- Write better code and better documentation

- Raise awareness about the importance of reproducibility in science.

- I have found that the codes from the research domains of computer science, electrical engineering, mathematics are easily available on github and hence reproducible. They have simple datasets, unlike earth science papers in which there are lots of variations in the datasets depending upon the geography etc. The earth science papers can be divided into two parts, one includes pure alorithm development and the other as an applications of the developed algorithm.

- I don't know. I'm not sure that reproducability is a problem as the code we write isn't close to capturing the complexity of the phenomena that we're observing.

- I find that the codes that I produce for my research are easy enough for anyone interested to reproduce based on the math that is presented in my papers. Besides, those who could not reproduce the code, would certainly not be able to run the ocde either, even if i gave it to them.

- Promote it as tools and make in applicable for non-IT-experts

- Take the time to think about how we can keep our work reproducible.

- have continuos, trackable workflow, from data gathering to publications

- Key is to assume very little to no experience when documenting your code. In general, good documentation is key, it is surprising how little it happens.

- We should increase collaboration across all climate/environment/water disciplines and the biggest challenge around me is to learn languages like R, Linux, C++ etc."

- My departement does not have enouhg funding to get enough licenses for non open access software/programming languages such as MATLAB. I believe, however, that producing code or software with open access tools would improve reporducibility as more people would be in a position to actually use, review (and perhaps improve) new and existing tools.

- Quality of code on github is too uneven, some of it first-rate, some of it awful. MATLAB File Exchange is a more reliable venue to look for code about something I work on. For reproducibility, I've used Code Ocean and plan to use it again. It establishes not just a place for the code, but also for an environment to run it. Therefore, reviewers can test the code without downloading and installing on their own system. (I ran into a problem trying to review some Jupyter notebooks. I found them difficult to install on my system. I'm not a Python expert, so it would have been nicer to have it on Code Ocean.)

- A lot. It is hard to summarize ideas succinctly in a comment like this, but see https://doi.org/10.1016/j.envsoft.2020.104888 http://doi.org/10.1016/j.envsoft.2019.03.020 http://dx.doi.org/10.1016/j.envsoft.2020.104753 https://doi.org/10.1061/(ASCE)WR.1943-5452.0001215

- Open access code, git versioning, data repository (even included in git LFS), collaborated in a structured agile development workspace

- Publish matehenatical models.

- better document and provide online supportive courses

- Increase human resources dedicated to it publish the code with easy to use licenses.

- Design and manage research projects as real teamwork (not just the loose ?collaboration? where everybody just does their own thing). Teamwork like in Scrum!" Most important part in my area is that input data have to become open source!

- From what I understood, for a researcher to be successful, he/she must publish as much new data and answer as much new questions as possible what let no space to test for reproducibility of previous works.

- I have moved into programming in a way that tries to always start from the barebones of the data and maybe hiding sections later but always keeping it in a direct line of order so people can see the transformation of the source data directly before being applied into a model. Rather than transforming and cleaning the data up in one step and then presenting this data because this can cause issues with reproducibility as there will be a raw and cleaned data set but not always a clear distinction on which should be submitted.