Towards readable code in neuroscience

Juan Luis Riquelme[1,2] *and Julijana Gjorgjieva*[1,2]✉

Code has become central to neuroscience, and the neuroscience community must take steps to ensure its reproducibility and best coding practices. Improving code readability benefits individual researchers and the wider neuroscience community.

Scientific research requires a diverse set of skills, including performing statistical analyses, writing manuscripts and, increasingly, developing software for experiment design, testing theories and processing data sets. A 2014 UK survey reported that 69% of researchers consider software essential for their work and that 56% develop their own software tools[1]. Particularly in neuroscience, the quantity and complexity of data often make automated processing tools essential for analysis. Most data published today have been processed by software, including that used in imaging pipelines, for sorting recorded neuronal spikes and for model simulations. As of January 2021, about 1 in 5 PhD or postdoctoral positions on the online FENS (Federation of European Neuroscience Societies) Job Market explicitly reference 'Python', 'MATLAB' or 'programming'. Therefore, ensuring that neuroscientists follow good coding practices is becoming increasingly important.

Mistakes in scientific software can have profound impacts. In 2006, five manuscripts were retracted after the authors found an issue in their custom code to analyse protein structure[2]. In 2016, a bug was detected in a common fMRI analysis implementation, raising questions about the results of over 3,000 articles[3]. Cases such as these have increased efforts to improve code reproducibility[4] (see Related links). For example, Nature journals require availability statements, and some even make code reproducibility integral to the review process (see Related links). However, code readability, a different yet equally important aspect of good practice, is often neglected. Code is readable if others can easily understand and modify it. With research software often written by people without formal training in software engineering, it is often accepted that writing scientific code need not follow best practices, as long as it works[5]. This tolerance has subtle implications that affect individual researchers, research laboratories and the neuroscience community. Here, we explain the benefits of code readability and what individual researchers and the wider community can do to improve code readability in neuroscience.

The value of readable code

The author understands code best at the time of first writing it, and will update and rewrite it multiple times. As every modification involves reading, the author is also the primary reader of the code and the person to benefit most from its readability[6]. With research projects typically spanning many years, a focus on readability at the time of writing saves time in the future, reducing 'technical debt'.

Readable code also benefits lab colleagues and collaborators. Given the collaborative nature of science, code initially written by one person is often used and further developed by others. Poor readability has a cost: for example, how often have manuscripts been delayed because of hard-to-find bugs? How many emails have been exchanged about data formats? How much time has been spent rewriting an analysis from scratch because the original author left the lab? Thus, lab heads and project leaders should promote readable code from the start. Readable code is more likely to be readily corrected and to remain usable for longer.

Finally, readable code benefits the entire research community. Software that originates as a tool for a specific project sometimes expands and becomes fundamental for new scientific discoveries. NEST, a simulator of networks of neurons initially developed for a thesis, is now credited in nearly 500 publications and has attracted contributions from nearly 100 authors. Although not all scientific software will become a community project, if it does, its success relies on the ease with which it can be modified and maintained.

What can researchers do?

Reuse existing code. Every new line of code costs time and effort to write, test, debug and maintain. Reusing existing code from lab colleagues or collaborators can reduce this cost. Community and industry tools are also very helpful and must be appropriately cited whenever used (see Related links).

Often, existing tools solve the problem at hand approximately but not exactly. In other cases, tools may initially be suitable but become less so as the scientific question evolves. Unlike hardware, software tools are easy to modify. Community open-source software, in particular, provides an opportunity to extend and tailor existing code. Contributing to community efforts also alleviates the responsibility of maintaining code beyond the lifetime of a scientific project.

[1]*Computation in Neural Circuits Group, Max Planck Institute for Brain Research, Frankfurt, Germany.*

[2]*School of Life Sciences, Technical University of Munich, Freising, Germany.*

✉*e-mail: gjorgjieva@brain.mpg.de*

https://doi.org/10.1038/s41583-021-00450-y

Break code down. Code is most readable when organized into pieces that can be digested and understood independently. These pieces are abstractions: concepts from which unnecessary details have been removed. All programming languages support abstractions through syntactic constructs such as functions, methods, namespaces, classes, modules and scripts. When used adequately, these constructs enable future readers of the code to navigate it, by understanding each individual piece's concept, function and interactions with other pieces of code.

Document code. Just as scientific protocols and progress must be recorded in lab notebooks, code must too be documented. We highlight two essential tools: annotation using comments and version control.

Making code readable does not require extensive commentary. When code is carefully broken down, and each piece is given a descriptive name, the code becomes self-documenting. Additionally, comments can support abstractions by documenting inputs and outputs and explaining their roles. Comments are most useful when they provide context for the reader by describing how a piece of code can be used, why it is necessary or why it is implemented in a particular way. Literate programming tools, such as Jupyter notebook and Markdown, can complement traditional code comments.

Version control systems, such as Git, track changes to code. Although fundamental for openly sharing code and collaborating, they are also useful to individual users because they allow reverting changes. As a scientific project evolves and code undergoes multiple iterations, reviewing the change history of code and reverting it to previous versions whenever needed becomes invaluable. Further, the ability to restore previous versions liberates the writer from the temptation of keeping old code around, minimizing implementation clutter.

These are just a few suggestions for improving the quality of scientific software. Many other good practices and principles exist, including unit testing, test-driven development, continuous integration, software patterns and pair programming[6,7]. Individual researchers can engage with additional self-learning resources and improve their computational literacy through more formal training.

What can the community do?

Multiple measures exist to ensure the quality of scientific manuscripts, including writing workshops, writing mentorship from project leaders and detailed reviewer feedback. Similarly, when a student needs to learn a new lab technique, they may attend a summer school. As code is fundamental to produce scientific results, shouldn't similar structures for quality assurance and training exist for writing effective and readable code?

Early career neuroscientists often learn computational skills through self-learning resources and informal interactions with colleagues. Although these avenues are not to be disregarded, labs, institutions and funding agencies can build other structures to provide more thorough training and support. Code clubs are a simple self-organized format that can provide a sustained pace to learn good coding practices[8]. They can be interactive group reviews of existing code or tutorials that introduce new techniques to the participants. Some institutions also provide code clinics: one-to-one sessions where computationally skilled researchers provide concrete advice on improving code. In addition, online or in-person workshops can be organized with software companies and volunteer organizations, such as The Carpentries[7], and tailored to the participants' level. Initiatives like these are more likely to be embraced by project leaders and institutions if appropriate funding mechanisms are available.

The community must acknowledge the critical place of code in the future of neuroscience and act to improve coding practices. The first steps are already underway with an increased presence of 'research software engineers' in research institutions and some journals encouraging code reviews during peer review[9]. As this recognition grows, training in code writing will likely become commonplace.

Neuroscience has been rapidly advanced by methods that have rendered it an impressively data-rich field — so rich that automated processing tools are now indispensable and ubiquitous. We must recognize the importance of these tools and invest in their quality. Research can only be as good as the tools we use for it. The road ahead includes advancing the computational literacy and good coding practices of neuroscientists at all levels, from graduate students and established researchers to academic institutions and scientific journals, and the community as a whole.

1. Hettrick, S. It's impossible to conduct research without software, say 7 out of 10 UK researchers. *Software Sustainability Institute*, https://software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers (2014).
2. Miller, G. A scientist's nightmare: software problem leads to five retractions. *Science* **314**, 1856–1857 (2006).
3. Mumford J., et al. Keep calm and scan on. *Organization for Human Brain Mapping*, https://www.ohbmbrainmappingblog.com/blog/keep-calm-and-scan-on (2016).
4. Eglen, S., Marwick, B., Halchenko, Y. et al. Toward standard practices for sharing computer code and programs in neuroscience. *Nat. Neurosci.* **20**, 770–773 (2017).
5. Singh Chawla, D. Critiqued coronavirus simulation gets thumbs up from code-checking efforts. *Nature* **582**, 323–324 (2020).
6. Martin, R. C. Clean code: a handbook of agile software craftsmanship. *Pearson Education* (2009).
7. Wilson, G. et al. Good enough practices in scientific computing. *PLoS Comput. Biol.* **13**, e1005510 (2017).
8. Hagan, A. K. et al. Ten simple rules to increase computational skills among biologists with Code Clubs. *PLoS Comput. Biol.* **16**, e1008119 (2020).
9. Reviewing computational methods. *Nat. Methods* **12**, 1099 (2015).

Acknowledgements
The authors gratefully acknowledge S. Eglen, J. Kirchner and G. Laurent for helpful feedback.

Competing interests
The authors declare no competing interests.

RELATED LINKS

FENS (Federation of European Neuroscience Societies) Job Market: https://www.fens.org/News-Activities/Jobs/?pid=509%7C508&key=python%7Cmatlab%7Cprogramming
Git: https://git-scm.com/
Guide for reproducible research: https://the-turing-way.netlify.app/reproducible-research/reproducible-research.html
How to cite and describe software: https://www.software.ac.uk/how-cite-software
Jupyter: https://jupyter.org/
Nature journals' reporting standards and availability: https://www.nature.com/nature-research/editorial-policies/reporting-standards
NEST: https://www.nest-simulator.org/publications/index.php
Research software engineers: https://researchsoftware.org/
The Carpentries: https://carpentries.org/