Title	Version
IBL Code Reproducibility Policy	0.0.1

Authors	
Gaelle Chapuis, Niccolò Bonacchi, Sebastian Bruijns, Eric DeWitt, Liam Paninski	
Editors	**Working Group(s)**
Niccolò Bonacchi, Mayo Faulkner, Olivier Winter	Code Reproducibility
Abstract	

- We want IBL to set a strong example on reproducibility.
- We want to minimize bugs in our published papers and shared code.
- Writing code with an eye towards reproducibility / code review will help elevate the quality of the code and improve our trainees' software engineering skills
- We want to build software tools that are well-documented and easy to support after the author of the code graduates or moves on to another position

Associated documents:

- **IBL Code Reproducibility Guidelines**
- **Code review Checklist**
- IBL Publication Policy
- IBL Data Release Policy
- IBL Data Release Guidelines
- IBL Personal Project Policy
- Software deployment

Title	Version
IBL Code Reproducibility Policy	0.0.1

Table of contents

Title	Version
IBL Code Reproducibility Policy	0.0.1

Section 0: Drivers and Background

Motivation and drivers:

- IBL depends on code! Our major outputs are datasets, scientific results, and new theories and analysis methods. All of these fundamentally depend directly on underlying code. Therefore, to ensure high-quality IBL outputs, we need to ensure high-quality code.
- Collaborative science requires that scientists can understand, use, and rely on the code written by others.
- The IBL is a flagship organization, dedicated to open science, and the community will pay close attention to our results and our processes. We really want to avoid any embarrassing errors. More positively, the IBL can play a strong role in leading the way improving the “usual way of doing things.”
- Our SAB has advised us to improve our software development practices and develop a proper policy to address the above issues.

Current typical scientific coding practice is as follows: a researcher (typically without training in best software practices) writes code to fit their individual needs, checks that the results pass sanity checks and seem “reasonable,” and publishes after checking the results for statistical validity. When a paper is published, the code is sometimes posted to a repository, often without sufficient documentation to reuse it or verify that it behaves as intended. Often code is not maintained actively and may go stale.

This process will typically miss bugs that produce statistically significant results caused by faults in the algorithm, data processing, or misuse of a software package. Further, a bug can prevent a true result from being discovered! This process also leads to various downstream frustrations in reproducing the code results, or reading, using, or adapting the code for other uses.

So, how can we reduce bugs and improve code in the IBL?

1. Adopt some basic best practices for coding in the IBL
2. Use code review!

What are coding best practices?

- Documentation of code as you write—even before you write—helps ensure that the purpose of the code is clear to you and others
- Testing code to make sure it performs what you intend it to

- Committing code regularly to a repository so you can track changes (and find errors) as well as collaborate

What is code review?

- Code review is analogous to paper review: we review papers before they go out with the IBL stamp of approval. We should do the same for the code that underlies the data+analysis in the paper.
- The goal is *not* perfect code. “Instead of seeking perfection, what a reviewer should seek is continuous improvement.” from https://google.github.io/eng-practices/review/reviewer/standard.html
- This is standard practice in industry; we would like to import best practices where feasible

“Won’t this slow down progress?” Short answer: not if done right. Long answer:

- If the code reviewer catches a bug that was missed, this will actually speed up progress (since we avoid wasting time using buggy code).
- Code review does take some time and effort (just like reviewing a paper) - but again, we accept the cost/benefit tradeoff of paper review, so why not code review?
- If code review is done as recommended - progressively, over the course of code development - not all at once at the end, which is highly non-recommended - then the code review would not impose a barrier on submitting a paper (since most of the code would already be reviewed well before the paper is even written).
- The reviewer’s role is to catch errors or bits of code that are hard to understand and suggest improvements, not force researchers to write code they don’t want to. The main goal is to get some additional eyeballs on our code, to reduce the number of bugs - not to slow down progress.

Section 1: Goals and scope

The goal of this policy is to encourage that:

a. Code produced within the IBL is shared as needed;
b. IBL code meets high standards of accuracy;
c. IBL code is well maintained and documented;
d. Published results from IBL are reproducible, even beyond the original author(s) leaving the IBL.

The scope of this policy covers:

a. Code written as part of an IBL member's registered project;
b. Code written as part of an IBL platform project; this includes the code forming the IBL main data architecture

Section 2: Code reproducibility Working Group (CRWG)

The CRWG will assume the following roles in overseeing IBL code forming and releases:

a. Draft and maintain a body of documentation of policies, guidelines, and advice for producing, reviewing, and publishing code.
b. Arbitrate disputes involving conducting the Code Review. Help assign code reviewers if the author of the code has trouble finding reviewers, checking workload is distributed fairly.

Section 3: Types of code

IBL libraries and codebase

This includes code from the main data gathering and analysis pipelines of the IBL.

This code is typically generated by IBL staff members as part of the main IBL endeavours.

Examples of such code are:

- Pipelines (e.g. ibllib), toolboxes (e.g. brainbox), task control (e.g. iblrig) code

IBL platform paper code

Definition: any code that underlies a result in a big platform publication (that is not included in an IBL library discussed above).

Small group or individual project code

Such code is generated by IBL members as part of a registered IBL personal project ('primary science paper').

Examples include:

- Specific analysis, model code

Section 4: Authors

The authors are those writing or amending the code. How to establish who wrote or amended the code is described in the Guidelines.

Section 5: Coding Guidelines

Members of the IBL should try to follow the best practices for coding outlines in the IBL Code Reproducibility Guidelines. This will facilitate collaboration within the IBL and reduce the effort required during code review.

Section 6: Code Review

Any code used to generate a figure or results in a published journal or conference paper bearing the name of the IBL must be internally reviewed prior to publication or placement in a public repository.

Internal code review is strongly encouraged (but not required) for code accompanying other types of communications such as e.g. posters or abstracts.

The work of the code reviewer(s) should be acknowledged where appropriate in the paper and/or the code repository / documentation. In some cases (e.g., if the reviewer suggests a new algorithm that significantly improves the results or changes the message of the paper) a reviewer may earn authorship.

How is the internal code review conducted?

- [**strongly recommended**] A code reviewer is assigned upon the project registration, and code reviews are conducted on a regular basis.

- [**mandatory**] A code review is conducted on the code to be published, and has to be sent to the Review board upon submission for internal review. The code review is part of the publication process (see Publication Policy), and without it publication cannot proceed.

The code writer has first responsibility for finding a reviewer. IBL members are preferred as code reviewers, but the reviewer can be chosen from outside IBL with a good justification. If the authors have difficulties finding a code reviewer, the Review board should help recruit a code reviewer (within IBL), and must notify the CRWG.

If a paper depends on results from a given set of code, this code must be reviewed upon submission for internal review. This should be considered the final deadline; however, code review **should be undertaken on an ongoing basis**, when a semantic unit is sufficiently mature - as noted above, we strongly encourage the author of the code not to wait until the entire package is "finished." This will make code review much more effective and will avoid slowdowns before submission of the manuscript.

Finally, "delta" reviews are permitted (i.e., if a previous code review checked one version of the code, then parts are added or changed, only these affected parts need to be reviewed) during the internal review process, to allow for late changes before e.g. conference deadlines. The author should contact the code reviewer and ask to be provided with a new review of the changes to be sent to the Review board in such cases.

Statement of scope and support / README:
A statement of scope and support should appear clearly in the repo README, and will guide the code reviewer's work; e.g., review of "personal code" should be reviewed differently than "core" code. This statement describes the use cases of the code and support available (e.g. if the code is intended to work solely on IBL data, or if it is a tool to be widely used by the community). The Code Reviewer is responsible for checking if this statement is adequate given the state of the code.

Checklist verification by the code reviewer
The reviewer will check whether sufficient unit tests are in place + passed, code coverage is assessed and sufficiently high, and documentation / API is adequate. If a new algorithm is introduced or implemented, the reviewer should check the correctness (and optionally comment on the efficiency of the implementation). See checklist here: https://docs.google.com/document/d/1ums99_gYKeifuQCIveq7_UPQ_bB0Opjhp9zM4cn_cc8/edit?usp=sharing

Can a code reviewer reject a paper - i.e., stop it from being submitted or posted?

If a critical error is found that invalidates the result of the paper, yes.

However, we expect this to be a rare case. More generally, the role of the code reviewer is analogous to the role of the internal paper reviewer: the reviewer will make suggestions about

how to improve the code, and ideally the code author will use this feedback to make improvements.

Section 7: Code Sharing

Any code used to generate a figure or results in a published journal or conference paper bearing the name of the IBL **must be publicly shared at the latest upon paper publication or upon depositing the paper in a public repository** (e.g. Biorxiv). We recommend working with an open GitHub repository throughout the duration of the project, to facilitate more open collaboration (see Guidelines), but this is not required.

Code used in other forms of communication, such as abstracts or posters, are not required to be shared publicly (and may thus be exempted from code review) but we encourage sharing in these cases as well.

In the case of a member leaving the IBL, the code developed as part of IBL projects has to be deposited in a repository accessible to all IBL members 1 month prior to the member's departure, and meet the minimum requirements below. See Guidelines for recommended repositories.

Minimum requirements for sharing code:

- The code must be placed in a public source control repository (e.g. GitHub)
- The data, code, and paper should all use the same variable names / notation where possible
- The code and data dependencies must be versioned to allow execution despite changes in libraries, language, etc.
- The code has to be released for each update of biorxiv or arxiv (but only the most recent version needs to be supported; see section on Code Maintenance).

Section 8: Code Maintenance

Maintaining code entails fixing bugs, clarifying documentation, commiting patches, and/or improving the organization of code in a source repository. Code authors are responsible for maintaining the code as long as they remain affiliated with the IBL. In cases of multiple publication versions, only the most recent version of the associated code needs to be maintained. In each case, the environment has to be sufficiently described (requirements file, listing dependencies with versions, etc).

What happens beyond the original author(s) leaving the IBL?

General:

- In cases where a code maintainer is required, it is the responsibility of the author to find a suitable code maintainer.
- The new code maintainer should be introduced by the author to the relevant body (working group or task force) upon being appointed, at least 1 month prior to the author leaving the IBL.
- If a code maintainer has not been assigned by that time, the author should notify the relevant body, who will be charged to establish one.

Specific:

- Code associated with personal projects need not be maintained once the author leaves the IBL; code maintenance in these cases will be up to the author.
- Code associated with platform papers will be maintained by another IBL member (usually a member of the associated task force), for a maximal duration of 6 months after the authors has left the IBL. The relevant body for platform paper code maintenance is the task force associated with the platform paper.
- Pipeline code will be maintained by another IBL member (usually a Software Developer staff) once the author leaves the IBL, throughout the time of the internal pipeline usage and for a maximal duration of 1 year after publication if the pipeline is no longer actively used internally. The relevant body for pipeline code maintenance is the Data Architecture WG.

Section 9: Specific cases for IBL platform project code

Codebases developed as part of platform projects are more visible and must satisfy a couple additional requirements.

Internal code review

IBL libraries and pipeline code must be reviewed *continuously* by IBL staff members (rather than solely upon publication), and be at least checked and tested as is asked from any other code.

Code sharing

Code used in platform publications will be frozen upon release, alongside with the data and library associated which will also be frozen. There will be no continuous integration of such code once frozen.

Title	Version
IBL Code Reproducibility Policy	0.0.1

External user testing (platform papers)

- Platform papers must satisfy a higher level of ease of reproducibility and user experience. These factors will be highly project-dependent; requirements and expectations for each project will be determined on an ad hoc basis by a User Experience team.
- The Code Reproducibility WG should be contacted if a User Experience team cannot be assembled for a given platform paper.