Response to the reviewers

March 10, 2023

Dear Dr. Aaron Elison,

We would like to thank you and the reviewers for the thoughtful comments and for the opportunity to submit a revised version of our manuscript "Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution", for consideration for publication in Methods in Ecology and Evolution.

We have addressed all the comments that you and the reviewers provided. We provide a point-by-point detailed response discussing each comment or suggestion. Our responses are mirror-indented, while the editor's and reviewers' comments are numbered. To facilitate the review, we have also included indented extracts from the text, which are indented and displayed in a gray colour. The line numbers in our responses correspond to those in the revised manuscript.

In the response to your recommendations, we have made several changes to the manuscript.

We have revised all sections to rewrite several sentences to decrease their length, improve the clarity and decrease the complexity of sentences. We removed one of the use cases to simplify the narrative, and improved the readability of figures and tables. We have also included additional references to support and to clarify certain sections.

We thank you again for your time and for the opportunity to submit this revised version of the manuscript.

We hope that our revision has further improved this manuscript and that it now meets your expectations. We very much look forward to hearing from you.

Sincerely,

Pedro Henrique Pereira Braga, on behalf of my co-authors

Comments from the Editor (E1)

As a long-term but sporadic GitHub user myself, I agree with most of the points and cases presented in the paper.

E1.1. I agree with Reviewer 1 that these could be streamlined somewhat into a smaller number of more generalizable cases. They also are a bit heavily weighted towards more experienced users. “Simple” operations like push, pull, and commit are rarely understandable to new users, who routinely forget to commit their actions. The GitHub Desktop app makes many of the operations and use of GitHub somewhat easier, and is a good entree into GitHub and Git for novices. More could be made of the desktop app.

Response: Thank you for these comments. We agree that the use cases and the jargon could be simplified to make the text more readable and attractive. To address these recommendations, we: (1) added a brief paragraph in the introduction describing basic Git and Github workflows (between lines 98 and 115), directing readers to other references containing more technical and in-depth information about these platforms; (2) decreased the number of use cases (see response to R1.1 and response to R1.2.); (3) added information on how GitHub Desktop and RStudio can be used as GUI interfaces for Git into Box 1 and into the Introduction.

E1.2. In my own experience, the “public” nature of repos is a major barrier to the use of GitHub for exactly the reason you suggest - EEB practioners remain very uncomfortable with sharing data pre-publication (and often even post-publication). Some additional emphasis on the ability to make repos “private” should be included beyond the Boxes.

Response: Thank you for this suggestion! We added a discussion on how repositories can be kept private, as well as emphasized that there should not be a trade-off between privacy, using GitHub for collaboration, and open science practices.

These changes have been added to two sections. To the “Storing and sharing research compendia” section (between lines 153 and 159), we modified the text to the form below:

> GitHub allows administrative control over who can view and modify repositories, and the process of cloning, forking and suggesting changes through pull requests (see Box 1) ensures code owners have full control over code and documents. Repositories can be changed from public to private, allowing data storage and management without sacrificing the privacy that may be necessary for certain compendia. With this, researchers can grant collaborators read and/or write access to files in private repositories to pursue pre-publication analyses or writing in privacy.

To the “Open science discussion” section (between lines 367 and 372), we modified the following paragraph to this form:

> The desire or need for privacy during the developmental stages of a manuscript or of a larger research project is common in EEB, and this is often perceived as a major barrier to doing science openly. Because GitHub repositories can be made private or public at any time, there is no need to choose privacy over open science, or vice-versa. Repositories can be kept private until their contents are ready to be shared publicly, as might occur when a research article is published or when an embargo is lifted.

E1.3. Another reason that many EEBers don't use GitHub appears to be related to how we develop code for projects. Unless we are developing packages that others are likely to use, we tend to develop one-off scripts or functions that, once they work and are run, are never returned to. Version control is not used widely, or even needed in these cases (and my own work in software engineering provenance software has similarly had little uptake for the same reasons). I recognize that this makes it difficult to document the processes involved, but if the goal is simply to analyze the data at hand and push out another paper, why deal with the overhead of version control? At root, this is a sociological phenomenon, not a scientific one. Within your ms., consider saying something about how even one-off scripts may need to be documented, archived, and assigned a DOI to conform to evolving archiving requirements of journals.

Response: This is a great point. We have added the following sentences to the end of the "Why aren't more EEB researchers using GitHub" section (between lines 497 and 501) :

> Other scientists may simply lack the time or incentives to document and version control their code if the code is unlikely to be reused beyond their analysis. However, we (and others, e.g., Gomes et al., 2022) argue that for open science and collaboration to be successful, code owners should document, and version control their code, despite uncertainty about future use.

E1.4. I am surprised you didn't include integration between GitHub and Overleaf. Overleaf is also a good collaborative latex-based text processor, and provides syncing with GitHub (and Dropbox) in a way that will be more familiar to beginning users. (Actually, I eventually found it buried in Table 1)

Response: Thank you for this recommendation! We now discuss the possibility of integrating GitHub into existing workflows on Overleaf, Google Drive, and DropBox among other services to collaboratively write scientific manuscripts, in the section "Writing a manuscript" (between lines 307 and 312). We refer to Table 1 for further details on these integrations. These changes appear as below:

> Incorporating GitHub into the process of writing a manuscript does not necessarily mean pivoting to an entirely new workflow. For instance, authors who prefer writing in LaTeX or Markdown can synchronize Overleaf (Using Git and GitHub), HackMD (https://hackmd.io), and other platforms directly with a GitHub repository. Similar GitHub integrations are available for projects stored in DropBox, Google Drive, among other popular tools familiar to many scientists and their collaborators (Table 1).

E1.5. Finally, DropBox, SharePoint, and OneDrive are file-sharing systems that include version control, are widely supported, especially in colleges and universities, and are much easier to use for those who might be uncomfortable learning a system such as GitHub. Other than the ability to clone and fork, why should one choose GitHub over, for example, DropBox or SharePoint? More should be made of this in text, as opposed to burying it in Table 1, which is referenced only once in text (line 232) and only in the context of other longterm archives (line 231). Indeed, I suggest a somewhat expanded section of the ms. dedicated to a detailed exploration of the comparisons made in this Table. Otherwise, the ms. comes across as more proselytizing and less analytical/convincing.

We think this is a good point, as many readers may not immediately see the value in a time investment to learn a new tool when they are already comfortable with tools that meet most of their needs. To keep with the recommendations to decrease the complexity and the length of the manuscript, we refrained from creating a new section. To address your recommendation, we identify specific points in the manuscript where we discuss how GitHub enables different and better types of collaboration and open science in relation to alternatives (such as Dropbox or Google Suite). We added this discussion earlier in the manuscript, and now refer more often to Table 1. These changes are listed below:

In the first paragraph of the introduction, we made the text more concise and also modified the following sentences to incorporate these suggestions (between lines 62 and 69):

> Maintaining code for scientific collaboration requires an efficient and well-documented workflow (J. M. Perkel, 2020). To facilitate this process, scientists have been adopting tools from information and systems technology, such as cloud-based services for documentation and version control (J. Perkel, 2016). These include Google Suite, Microsoft Suite, DropBox, and GitHub. Within the spectrum of cloud-based services for collaboration, GitHub is uniquely positioned (Table 1) to benefit scientists because it is specifically designed to store, track changes, and enable collaboration on computer code—

> fundamental components of modern research.

To the second paragraph (between lines 78 and 86), we incorporated these suggestions in the following sentences:

> GitHub also integrates several communication features, such as Github Issues, Github Discussions, Github Pages, which allows users to engage in discussions, to plan and collaborate on code, and publish information to a webpage (see Box 1). These features are an improvement over traditional practices of directly sharing files through email, other cloud-based services or physical storage units, which can become challenging and time-consuming in long-term and collaborative projects (Ram, 2013). By making it easier to integrate versioning, communication, collaboration with research code and data, Github helps facilitate open science practices in research projects (Perez-Riverol et al., 2016).

We added the following sentences to the "Storing a research compendia" section (between lines 146 and 152):

> Many researchers begin using GitHub to backup their research compendium (Marwick et al., 2018) into a centralized, readily-available remote server (see Box 1). This practice has the advantages of facilitating collaboration, integrating data and code archiving, allowing file versions to be accessed and restored, and further contributing to open science practices (Borghi & Van Gulick, 2022). Changes made to files in version-controlled repositories are accompanied by authored descriptions of modifications (Box 1).

In the first paragraph of discussion we incorporated these recommendations in the sentences below (between lines 435 and 439):

> While tools like Google Suite and DropBox enable rapid sharing and collaboration of some research documents, GitHub brings together features that directly integrate open science practices. These include linking data, code and findings to public discussions, tracking edits to files for review, and managing complex research projects with many collaborators and goals.

And finally, in the discussion, under "Why aren't more EEB researchers using Github", we added (between lines 475 and 476):

> Additionally, with the availability of software licenses for tools like Google Suite or Microsoft Office, researchers may be reluctant to spend valuable time learning another tool.

We also note that GitHub falls short in certain features that exist or work better in other platforms, within the "Other platforms section" (between lines 463 and 468):

> First, real-time document editing is still best performed on other platforms, such as cloud-stored documents from Microsoft Word, Google Docs, and HackMD (https://hackmd.io/). Second, operations that are dependent on software requiring graphical user interfaces might not be easily achievable in GitHub, such as designing and manipulating figures or tables. While creating tables and figures can be done through code, users may choose other software to collaboratively brainstorm figures and tables, like Google Slides or Google Sheets (but see GitHub Discussions).

Recommendations from Reviewer 1 (R1)

This paper clearly communicates the enthusiasm of the authors for integrating GitHub in the EEB research process. Advantages in areas of collaboration, transparent and reproducible science are clear and nicely discussed. This is a well written paper.

R1.1. I would, however, recommend some restructuring and shortening. As it stands, this paper is too long and some of the use cases are too nerdy to spark enthusiasm in somebody who is not already using GitHub. Breaking it into 13 use cases leads to some repetition and those use cases may be combined more effectively for convincing a new user.

Thank you for this recommendation! We agree that the many sections could be restructured to become more concise and clearer. To address this recommendation, we did the following: (1) to reduce repetition and jargon in the use cases sections, we added a paragraph to the introduction (between lines 98 and 115) explaining basic Git and Github workflows. We also cited resources for learning how to use tools. These changes allowed us to focus on the usefulness of these tools for research in ecology and evolution; (2) we removed the "laboratory notebooks" section and merged part of its contents into the other sections section, which allowed us to further shorten the text; and, (3) we performed an extensive review of the text to clarify and shorten sentences.

R1.2. The title and abstract gave me the impression that the goal is to convince EEB researcher to start using GitHub. If that's the case, it might be better to tailor the use cases to that entry level and use less GitHub specific lingo. Advanced usage may be mentioned but not detailed as much as it is currently done. E.g. collaboratively writing a paper in GitHub is probably out of the question for most. Most GitHub options for communication, discussion, issue tracking are still somewhat esoteric for most non-programmers.

We thank you for this suggestion and we agree that the narrative could be simplified to convince researchers to adopt GitHub for certain of their practices.

Following our modifications to address R1.1 and E1.1, we reduced jargon, simplified the narrative, and focused more in the advantages of the features GitHub provide to improve the research workflow in ecology and evolution. We also clarify to the reader that the objective of the manuscript is not to instruct how these tools work. We still mention certain advanced tools, but we refer the reader to additional learning resources instead of describing them in the manuscript.

R1.3. Although the abstract states 'We outline features ranging from low to high technical difficulty' the paper reads a bit like a laundry list of what GitHub can do (in fact, the word 'can' is used about 140 times, which makes for tedious read). Figure 2 helps sort through this laundry list and defines the technical difficulty. It might be better to clearly lay out where anybody can start using GitHub effectively in the text. And the emphasis is on 'effectively'. Most people are not likely to learn a new piece of software if it does not promise to reduce effort and time. So, defining tasks where GitHub shines in terms of return on investment maybe the better approach to convincing new users and then only mention the advance use cases with some pointers to further reading, but not going into too much detail.

We appreciate this suggestion. While the attribution of a type of return of investment index to use cases could be interesting, we believe that such value can be highly biased, subjective and vary much among fields, laboratories, research culture, and individuals. For instance, while collaborative writing of a manuscript using GitHub may be seen as a strong time investment and an advanced usage, there is an increasing number of research articles using GitHub to write manuscripts, such as Halie et al. ([arXiv:2102.01521](https://arxiv.org/abs/2102.01521), 2021), Halie et al. ([mSystems.00233-21](https://journals.asm.org/doi/10.1128/mSystems.00233-21), 2021), Lordan et al. ([mSystems.00122-21](https://journals.asm.org/doi/10.1128/mSystems.00122-21), 2021), Rando et al. (CEUR.2976.2, 2022), Sabatini et al. ([Global Ecology and Biogeography](doi.org/10.1111/geb.13346), 2021), Le et al. ([Cell Systems](https://www.cell.com/cell-systems/pdf/S2405-4712(21)00285-4.pdf), 2021), and many more (see https://manubot.org/catalog/; in addition to our manuscript). In these cases, most co-authors had little experience with effectively integrating Github tools to write a manuscript within the platform, and had to commit to learn and adopt proposed frameworks to achieve project completion. It is also difficult to measure return on investment on innovative tools that only recently took off, such as the usage of Github Actions in ecological research frameworks for data management, processing, and analysis [*e.g.*, in Sabatini et al. (Global Ecology and Biogeography, 2021), McIntire et al. (Ecology Letters, 2022), Micheletti et al. (Frontiers of Ecology and Evolution, 2021)]. We reiterate our appreciation for this suggestion, however we think that the quantification of a ratio between investment of time and resources and returns when learning research aiding skills requires an in-depth individual study.

In this context, we note that our metric of cognitive requirement represented in Figure 2 considers the "usefulness" or cognitive requirement for learning GitHub (see also our response to R2.2). Moreover, we have taken the advice of reviewer 1, and reduced and reordered the use cases (see our responses to R1.4 and R1.1).

R1.4. In my experience, the project continuity is actually very high on the importance list for researchers, i.e., knowing that the code and data will be findable by the next student. This includes the discussion of organizing and managing teams, keeping lab information in one place etc. Followed by code versioning and the ability to go back to older versions. Interest in website development is picking up because it really is simple to do in GitHub, and the information can be maintained by several people (i.e., a lab group).

Thank you for this recommendation. We moved the "Project continuity" section to be presented as the second use case, immediately after introducing storing and research compedia.

R1.5. Line 346 delete second 'can': who can also can change through time

Corrected! Thank you for catching this!

R1.6. Line 389 it should be 'each other's work': contribute to each other work without necessarily

Corrected! Thank you once again!

R1.7. Line 457 missing 'collaborator'? especially when many may be

We have added "prospective users" to this sentence. It now is "*It can be challenging to learn Git alongside scripting languages, statistical theory, and file system navigation, especially when many prospective users may be inexperienced with programming.*"

R1.8. Line 477: not sure what this sentence is saying: 'requiring the complementation of other tools to fully integration project files and GitHub repositories'

We agree that this sentence needed clarification. We have simplified it and it reads now as: "*Moreover, additional tools may be required to fully integrate project files and GitHub repositories [...]*". Thank you once again!

Recommendations from Reviewer 2 (R2)

The manuscript presents how ecologists and evolutionary biologists (EEB) can use collaborative software development project manage-

ment tools. The authors present a high-level view of Github and the Git version control system that is at its core. As the authors detail, the tools that Github provides has broad potential to be leveraged within the EEB community. The paper is well written and helps to make Github tools and related concepts legible to EEB audiences. On the whole, I see this as potentially being an impactful paper for improving EEB research. I have the following Major and Minor comments for the authors:

R2.1. Figure 1 was more confusing to me than helpful. It presents a view of a Github interface in order to detail generalized features; however, it is edited/abstracted so much that it doesn't map easily to the interface as it would be viewed by a reader of the paper. To improve this I would reduce the level of abstraction of the web interface.

Thank you for this recommendation! We have decreased the complexity and clarified Figure 1. We changed it to bright-mode instead of dark-mode, blurred out irrelevant content, added three clear bounding boxes with their respective labels to highlight the repository resources, the file system organization and commit history, and the contributors.

R2.2. Figure 2 was useful, especially as it summarized across multiple areas that a reader make in-roads into using Github. Please clarify how the "Technical Difficulty" was assessed. Was this based on the impression of the working group? Was it quantified using different types of required knowledge (e.g., programming, software design, working in the Terminal, etc.)?

Thank you for catching this. We added a supplementary material containing appendices, tables and code that detail how the indices for degrees of collaboration and of difficulty were estimated for each use case. We also added citations to these elements in the text.

R2.3. I would find it helpful to give a brief (1-2 sentence) history of Git (L85-89). Namely, that it was developed as an aid for software development within distributed groups of software engineers and was developed within an open-source framework so that it could be improved by the community. This provides more context as to why Github (as it extends Git) is a useful tool for collaboration "by design".

We now briefly discuss the origin of Git and its expansion to applications beyond software development to help manage collaborations and community contributions, in the introduction (between lines 87 and 97), as seen below:

> Git is the version control system that enables the collaborative

> tools available on GitHub. Git was initially developed as a fast, lightweight, and open-source system to allow software engineers to efficiently develop and collaborate on projects (Git - A Short History of Git). Since its launch in 2005, Git has become the leading version control system in software development and in other disciplines that require collaboration and community contributions, such as in scientific research (Spinellis, 2012). To understand how GitHub keeps track of changes to files and folders, it is recommended to have knowledge of basic concepts of Git (such as commit, push, pull, and checkout; see Box 1). However, the GitHub web-based platform and its integrated development environments (such as GitHub Desktop) allow users to perform most repository and data management operations without using the command console, making these functionalities available even to users who are less familiar with software development.

R2.4. I suggest adding a discussion on dependency testing within or following the paragraph on Automation (Lines 373-382). This is a project "ecosystem" phenomenon that comes from collaboration, where you build your project on the work of someone else. As projects change over time, they can alter other projects. Software engineers have been working on this challenge for a long time in distributed teams where different parts of software are being built by different programmers. Checks can be done automatically within the software engineering framework (see Pasquier et al. 2017 https://www.nature.com/articles/sdata2017114). Beyond detection, any major changes can be detected and presented without additional work from the user via features like badges (https://shields.io/) within the project page (e.g., README).

Thank you for this recommendation. We recognize that these are developments that have been increasingly encouraged in biological research. We now provide brief comments on how automation can aid unit testing of research code (between lines 375 and 389), and that automation can also help the interpretation, integration and usage of data and software across different sources. These modifications are contextualized in the extract below:

> Automation frameworks can streamline many stages of the scientific process, including data collection and data validation (e.g., Micheletti et al., 2021; Yenni et al., 2019), data analysis (e.g., Beaulieu-Jones & Greene, 2017), unit testing of research code (e.g., Sarma et al., 2016), archiving and deployment of data, code and reports (e.g., this manuscript, White et al., 2018), and the interpretation, integration and usage of data and software across different sources (see Pasquier et al., 2017). In this context, small modifications to code and data can be frequently committed and

automatically tested, as in continuous integration and continuous deployment practices (Meyer, 2014). This allows for early detection and correction of errors, potentially improving confidence in scientific development by minimizing software errors (see Soergel, 2015). In addition to increasing scientific rigour and confidence in ecological software (Scheller et al., 2010), automation can help advance more rapidly sharing ecological data and making sure the data are high quality (Dietze et al., 2018). Integrating automation workflows has been highly encouraged in areas of EEB, including predictive ecology (McIntire et al., 2022), long-term ecological studies (Ernest et al., 2018; Yenni et al., 2019), and management of species at risk information (Naujokaitis-Lewis et al., 2022).

R2.5. Related to Major Comments 1 and 4, given that the focus of the paper is toward integration into EEB scientific research workflows, I suggest adding another figure (or replacing Figure 1) that details a generalized scientific research workflow (e.g., Munafò et al. 2017 https://www.nature.com/articles/s41562-016-0021) and how Github supports/augments that workflow. This could relate to or build on Table 2, which a useful summarization of important features as they relate to different research participants.

We thank you for this comment. We discussed and attempted to associate the use cases to hypothetical versions of the scientific method and academic roles. Despite enjoying the idea of a figure resembling Figure 1 from Munafò et al. 2017 (Nature), we have not been able to produce a figure that would not feel polluted or redundant with Figure 2. We have attempted this with a circular flow, stacked bar chart, or a network graph. The issue is that most use cases are related to many stages of the scientific process. The exception are "Academic websites" and "Educational materials", which may contribute to research development, but more marginally in relation to the other use cases. We appreciate the suggestion and would be happy to come to back to it if we can work on an alternative to this problem.

R2.6. The sentence from Line 66-67 is missing a close parenthesis.

Thank you for catching this. We closed the brackets.

R2.7. There appears to be a "fourth" barrier at Line 478. I suggest splitting the paragraph from Line 472 to 483 into two paragraphs and expanding on both "reluctance to share data" and "language-specific resources" as barriers to adoption.

Indeed! We have numbered all five barriers to using GitHub to clarify this section.

R2.8. Line 570 “could also be e” should be “could also be a”.

You are right! We have corrected this issue. Thank you once again!