Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution

This manuscript (permalink) was automatically generated from SORTEE-Github-Hackathon/manuscript@55dfe2a on June 9, 2022.

Authors

- **Robert Crystal-Ornelas**
  0000-0002-6339-1139 · robcrystalornelas · rob_c_ornelas
  Earth and Environmental Sciences Area, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

- **Brandon P.M. Edwards**
  0000-0003-0865-3076 · BrandonEdwards
  Department of Biology, Carleton University, Ottawa, ON K1S 5B6, Canada

- **Katherine Hébert**
  0000-0001-7866-6775 · katherinehebert · hebert_kat
  Département de biologie, Université de Sherbrooke, Québec, Canada

- **Friederike Hillemann**
  0000-0002-8992-0676 · fhillemann
  Department of Human Behavior, Ecology and Culture, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany

- **Emma J. Hudgins**
  0000-0002-8402-5111 · emmajhudgins
  Department of Biology, Carleton University, Ottawa, ON K1S 5B6, Canada

- **Luna L. Sánchez Reyes**
  0000-0001-7668-2528 · LunaSare · LunaSare
  School of Natural Sciences, University of California, Merced, USA

- **Eric R. Scott**
  0000-0002-7430-7879 · Aariq
  Department of Wildlife Ecology and Conservation, University of Florida, Gainesville, FL, USA

- **Matthew J. Grainger**
  0000-0001-8426-6495 · DrMattG · Ed_pheasant
  Terrestrial Biodiversity, Norwegian Institute for Nature Research - NINA, Postbox 5685 Torgarden, 7485 Trondheim, Norway

- **Vivienne Foroughirad**
  0000-0002-8656-7440 · vjf2 · vforoughirad
  Department of Biology, Georgetown University, Washington, DC, USA

- **Allison D. Binley**
  0000-0001-8790-9935 · adbinley · AllisonBinley

Department of Biology, Carleton University, Ottawa, ON K1S 5B6, Canada

- **Cole B. Brookson**
  0000-0003-1237-4096 · colebrookson
  Department of Biological Sciences, University of Alberta, Edmonton, AB, Canada

- **Kaitlyn M. Gaynor**
  0000-0002-5747-0543 · kaitlyngaynor · kaitlyngaynor
  Departments of Zoology and Botany, University of British Columbia, Vancouver, BC, Canada; National Center for Ecological Analysis and Synthesis, Santa Barbara, CA 93101, USA

- **Saeed Shafiei Sabet**
  0000-0001-5919-2527 · shafieisabets · SaeedSHSABET
  Fisheries Department, Faculty of Natural Resources, University of Guilan, Sowmeh Sara, Iran

- **Ali Güncan**
  0000-0003-1765-648X · Aguncan · aliguncan
  Department of Plant Protection, Faculty of Agriculture, Ordu University, 52200, Ordu, Turkey

- **Friederike Hillemann**
  0000-0002-8992-0676 · fhillemann
  Department of Human Behavior, Ecology and Culture, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany

- **Helen Weierbach**
  0000-0001-6348-9120 · helenweierbach · HWeierbach
  Earth and Environmental Sciences Area, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

- **Dylan G. E. Gomes**
  0000-0002-2642-3728 · dylangomes
  (Current) National Academy of Sciences NRC Research Associateship Program, Northwest Fisheries Science Center, National Marine Fisheries Service, National Oceanic and Atmospheric Administration, Seattle, WA, USA 98112; (Former) Cooperative Institute for Marine Resources Studies, Hatfield Marine Science Center, Oregon State University, Newport, OR, United States

- **Pedro Henrique Pereira Braga**
  0000-0002-1308-1562 · pedrohbraga · pedrohp_braga
  Department of Biology, Concordia University, 7141 Sherbrooke Street West, Montreal, QC H4B 1R6, Canada

Abstract

Researchers in ecology and evolutionary biology are increasingly dependent on computational code to conduct research. With the growing role of data science in research, the use of efficient methods to share, reproduce, and collaborate on code has become fundamental. GitHub is an online, cloud-based service that can help researchers to track, organize, discuss, share, and collaborate on software and code. Despite these benefits, the use of GitHub by EEB researchers is not widespread due to the lack of domain-specific information and guidelines. To help EEB researchers adopt useful features from GitHub in their own workflows, we review twelve practical ways to use the platform. We outline features ranging from low to high technical difficulty: storing code, managing projects, coding collaboratively, conducting peer review, and writing a manuscript. Given that members of a research team may have different technical skills and responsibilities, we describe how the optimal use of GitHub features may vary among members of a research collaboration. As more ecologists and evolutionary biologists establish their workflows using GitHub, the field can continue to push the boundaries of collaborative, transparent, and open research.

Introduction

Most scientists, including ecologists and evolutionary biologists, are increasingly dependent on computational tools in their research[1]. Researchers write and use code as part of their scientific workflow to perform a wide-variety of tasks ranging from data management, data analysis, and study replication, to the application and the development of tools for hypothesis testing. To maintain code for scientific collaboration requires an efficient and well-documented work-flow[2]. To facilitate this process, scientists have been increasingly adopting tools from information and systems technology, such as cloud-based services for documentation and version control (*e.g.*, from the Google Suite, with Docs, Sheets, and Drive; the Microsoft Suite, with Word, Excel, and OneDrive; and GitHub;[3]). However, most researchers lack exposure to adequate software development practices and are required to dedicate valuable time and effort to self-teach the use of research-facilitating tools, and thus may be limited in their ability to adhere to adequate standards of scientific code quality and maintenance[1,4,5]. Here, we review and discuss one of the most used web-based platforms for computational version control and collaboration, GitHub, and provide researchers in ecology and evolutionary biology (EEB) with practical workflows aimed at facilitating their scientific code and management process.

With over 73 million registered users, as of 2022, GitHub is the most widely-used web platform for collaborating on computer code[6]. GitHub provides a simplified but powerful web interface that allows users to participate in projects, contribute code, report and discuss software bugs, discover existing code and data, and publish new code. Through version control, users have a detailed, chronological record of the files and directories stored in their repositories (see Box 1;[7]). This workflow provides a strong and clear advantage over receiving, processing and sending files back-and-forth (e.g. via email), a process that can easily become challenging and time-consuming in projects extending in time and in the number of collaborators[8]. Through the combination of version control management and the network- and collaboration-based features, GitHub can broadly facilitate openly available source code alongside concomitant collaborative development[9].

Git is the version control software system that enables all the collaborative tools available on GitHub. Although the understanding of basic concepts of Git (such as commit, push, pull, checkout; see Box 1) is necessary, the GitHub web-based platform and its integrated development environments (such as the GitHub Desktop) allow users to perform most repository and data management operations without opening Git command-line sessions. Still, GitHub users are encouraged to explore and improve their proficiency in Git when feeling comfortable, so they can feel confident in performing more flexible, complex operations (such as integrating changes from one branch into another branch with `git merge` and `git rebase`). Examples of extensive explanations on Git can be found in journal articles[9,10], video tutorials[11], and books[12].

The expansive GitHub user-community and the numerous resources on how to use GitHub have boosted its growing popularity[7,9,12,13]. Nevertheless, although multiple articles have encouraged researchers in EEB to adopt GitHub as part of their research process[3,14], its use is still not widespread. First-time users without formal training in information technology may face steep learning curves because GitHub and its features have been centered on collaboration for software development in information systems[15]. Moreover, domain-specific perspectives and resources providing tractable examples and practical guidance for researchers in EEB on GitHub are scarce (but see https://ourcodingclub.github.io; https://www.openscapes.org). A common adoption of GitHub for collaborating on a variety of research tasks can ultimately enable EEB researchers to spend less time on creating novel processes for collaboration and more time on their research[16]. More importantly, increasing the availability of data and code management standards – of which GitHub is one increasingly important component – make research more reproducible and collaborative[17].

This paper is the result of an academic hackathon held during the 2021 conference for the Society for Open, Reliable, and Transparent Ecology and Evolutionary Biology (SORTEE, https://www.sortee.org). We convened a group of around thirty researchers in EEB with varying levels of familiarity with using GitHub as part of their research projects to showcase and discuss how existing features can contribute to documentation and collaboration in EEB research. During the hackathon, we identified the need for a formal discussion on how EEB researchers can benefit from GitHub and its features to make research more collaborative and transparent. Here, we outline twelve practical ways that EEB researchers can use GitHub features for more collaborative, transparent, and reproducible science. We also provide critical perspectives on features that could be improved and catered towards research development.

Box 1

Glossary
repository: Commonly shortened to "repo", a repository is a collection of files (*e.g.*, a directory) tracked by Git. Repositories are managed by an owner and can be listed as "public" where the repository will be visible to all GitHub users or "private", where the repository is visible only to authorized users.
fork: A fork is a copy of a **repository** hosted on GitHub. If a repository is public, then anyone can make a fork. Even if they do not have access to **push** to the original repository, they can make a fork and edit it independently. Forks are linked to the original GitHub repository and "upstream" changes (i.e. those in the original repository) can be **merged** to keep the fork up to date with the original project. Changes made in the fork can be integrated into the original project via **pull requests**.
clone: Cloning a **repository** is a way of making a local copy (i.e. on your computer) of a GitHub **repository**. If you have access to **push** to a **repository**, this can be a first step to contributing to a project.
branch: A git branch is an alternative line of development for a project (repository). Branches allow to add new features or modifications to the project without affecting the main part of the project. Development branches can be created at any point in time and work on each branch can continue independently. Branching is useful for testing out new ideas (both code and text) which may or may not eventually get integrated into the main branch of the project. Branches can also be used to isolate contributions of multiple contributors. Each person working on their own branch eliminates problems that may arise if conflicting edits are **pushed** to the same remote branch. Changes in a development branch can be **merged** into the main branch via **pull requests**. Branches can only be made by those who are given access to the project **repository**.
commit: Commits are snapshots of the development of a project. In Git, versions of files and directories are uniquely identified as "commits", allowing one to identify and track modifications line-by-line. Commits can include changes in multiple files and must include a brief commit message describing the changes made. A typical workflow is to make some related changes in files, add a commit message (e.g. "generate and include fig1 in results"), and after several commits **push** those commits to the remote (*i.e.*, cloud-based) GitHub **repository**.
push/pull: When **commits** are made in a project locally, they must be synced with the remote GitHub repository by **pushing** them. Changes on a GitHub repository can then be **pulled** to keep your local version of the project up-to-date.
pull request: A pull request is a request that the owner(s) of a GitHub repository integrate changes you've made on either a **branch** in the repository or in your own **fork**. When you initiate a pull request, you must provide a description of what changes were made. Some automated tests may be run and review may be required before integrating your changes into the main **branch**.
merge: Combining **commits** from two different branches together into one **branch**
release: At any point a release can be made on GitHub to mark a significant milestone in the progression of a **repository**. While this GitHub feature is designed with releases of new versions of software in mind (*e.g.*, v1.0.0), it can also be used to create a snapshot of a repository at significant stages like submission, revision, and acceptance of an associated manuscript.
community: A forum where GitHub users can ask for advice, offer solutions to questions, and share ideas (https://github.community/).

12 Practical ways of using GitHub in EEB

Storing a research compendium

In EEB, a research compendium includes all computational materials related to research production, including data, code for analyses and protocols. Having multiple copies of these files is important to ensure that research can continue should one copy of a file be accidentally modified or deleted. Many researchers begin using GitHub as a means to backup their research compendium[18] to a remote server (pull-commit-push, see Box 1). The GitHub repository serves as a centralized backup and also allows users to synchronize files and work on any device with internet access. Because GitHub is built on git, version control features are also available, allowing users to re-visit or even restore previous versions of the repository. Storing a research compendium as a GitHub repository has additional benefits such as facilitating collaboration, integration with data and code archiving services, and contribution to open science, all discussed in sections below.

Unfortunately, GitHub repositories have file size limits, so they may not be appropriate for synchronizing large data files. Commits (see Box 1]) greater than 50 MB receive a warning and commits larger than 100 MB are blocked ([19]). Therefore, it is recommended that large datasets are stored only locally or using other cloud services.

Virtual lab notebook

Lab notebooks are the virtual or physical notebooks used to help researchers at any career stage keep track of their research methods or laboratory policies[20]. Digital lab notebooks stored in the cloud provide clear benefits given the ease with which documents can be shared with new employees and updated as policy changes or experimental methods are modified[21]. Increasingly, researchers are leveraging GitHub's underlying version control system to maintain and share digital lab notebooks[3]. At least for aspects of a research project that involve writing code, a GitHub repository is a form of a lab notebook; when changes are made to files in a version controlled repository, the author of those changes makes a commit (Box 1) accompanied by a message describing the changes and the reason for them. Later, the entire history of commits and their messages are viewable and can be audited similar to a physical lab notebook[8]. GitHub issues (Box 1) can be used to prioritize lab objectives and goals, as well as track any status updates. Some EEB labs have even turned their lab notebooks into shareable websites (https://scheuerell-lab.github.io/lab-book; https://github.com/HuckleyLab/how_we_work) as a centralized location for all lab resources.

Project management

Modern EEB research is highly collaborative, bringing together multidisciplinary teams from various institutions. On GitHub, collaborators can share feedback, brainstorm ideas, and troubleshoot problems (Figure 1). GitHub has an "Issues" feature that allows for discrete tasks and sub-tasks to be identified, assigned to team members, and categorized with custom labels. The new GitHub "Discussion" feature serves as a message board for conversation. Scripts, commit messages, and pull requests can be linked directly to issues and discussions, providing a clear record of project workflow. The use of GitHub for all project-related conversation and planning, rather than e-mail or messaging tools, makes it easier to keep track of progress throughout the lifespan of a project. Unlike emails and messages which can get lost as more new tasks arise, GitHub issues are intentionally closed by repository administrators hiding the issue from view (closed issues remain accessible but not immediately visible). Fortunately, it is not essential for all team members to have proficiency in git or programming, as users can interact with Issues and Discussions via web browser or e-mail (e-mail responses still get tracked as comments on the focal GitHub issue). For larger projects with many

team members and tens or hundreds of GitHub issues to sort through, project management software like ZenHub, can help prioritize issues and pull requests. ZenHub's web interface includes a GitHub Issue visualizer where users can organize issues into high priority or backlogged tasks and link issues together when they are related to a shared project goal or milestone. GitHub is currently beta testing a similar project management feature called GitHub Projects[22]. GitHub can also be integrated with other project management software such as Slack (https://slack.github.com/) so that teams are notified through a Slack group chat when updates are made to a repository.

Educational materials

GitHub supports a broad set of mechanisms for hosting educational materials. The entire process of running a course, workshop, or lecture, can all be done openly on GitHub including material development, web hosting, and delivery, and even the submission and grading of assignments. While there are other purpose-built platforms for this, GitHub provides a free, open-source alternative.

Making presentations can be done through most major high-level programming languages such as R, with `RMarkdown`[23], Python, with `python-ppt` (https://python-pptx.readthedocs.io), and Julia, with `Remark.jl` (https://juliapackages.com/p/remark). Since all these programs work via code bases, they can be version-controlled through git and GitHub. Once content is made, hosting a course website can be done through GitHub Pages, and there are lots of templates available (e.g., see https://github.com/topics/course-website). This way, the course content can be available to enrolled students, as well as a global pool of learners and teachers interested in the course material. Content can then be delivered via the course website, and/or a GitHub organization with, for example, template repositories for assignments. Student submissions are perhaps the most challenging component, but the new GitHub Classroom tool (https://classroom.github.com) allows instructors to host private assignments submitted as code files (.R or .Rmd) or .pdf files, and even build custom autograding tests.

Instructors may only need to incorporate some of these tools given the time required to implement various GitHub features, however it is still valuable to do so and to encourage students to begin learning about version control through interacting with git/GitHub, however minimally, through the course. Especially if a central tenet of a given course or educational unit is to introduce or give students experience with version control, reproducibility, and the tools that working professionals in EEB use, then adopting a few of these tools can be a great way to do so.

Creating a website

Personal or lab websites are one method of improving the dissemination of research findings and coordination of research efforts, but most EEB researchers have little experience in building or hosting webpages. GitHub Pages (https://pages.github.com) is a feature that, when activated for a repository, renders content written in markdown (e.g., https://github.com/SORTEE-Github-Hackathon/main-website) as an HTML web page with a URL (e.g., https://sortee-github-hackathon.github.io/main-website/). This can be useful for sharing reports (e.g. written with RMarkdown and rendered to HTML) with collaborators, or for more complex projects like personal, project, or lab websites. GitHub also offers website templates and it is possible to create a website by forking (Box 1) a repository hosting a GitHub Pages site as a starting point. Aside from free hosting services, another benefit is that GitHub Pages are autogenerated, meaning that when content is modified in the associated GitHub repository, the website instantly updates[7]. Though the templates are useful for quickly starting up a new website, users are able to fully customize their Pages websites (for technical details of customizing GitHub Pages site see[24]). We emphasize that despite the many benefits of using GitHub pages (free hosting, templates, customization), this avenue for creating a website will often be more time intensive than the out-of-the-box platforms. "Static websites" (i.e.,

websites where every user sees the same content) can be created using Jekyll (https://jekyllrb.com) and Hugo (https://gohugo.io), which also include template libraries for websites that can be hosted freely via GitHub pages. Several R packages such as distill (https://github.com/rstudio/distill) and blogdown (https://bookdown.org/yihui/blogdown) have made the interface to these website generators more accessible. In addition, Quarto (https://quarto.org) has options for creating websites (https://quarto.org/docs/websites) as well as the code-free website generator wowchemy (https://wowchemy.com) which uses Hugo.

Archiving citable code and data

GitHub makes it easy to store and share a variety of data files in the cloud. If a repository is made "public" the URL to the repository can be shared freely with others. However, for a variety of reasons (e.g., privately owned company, ability to make repositories private, accounts can be deleted at will) GitHub is not considered a long-term data or code repository like Zenodo and Figshare[3,9,**tbl:compare?**]. Also, unlike long-term repositories, GitHub does not issue Digital Object Identifiers (DOIs) for content uploaded to their servers. DOIs are persistent and unique alpha-numeric IDs assigned to research products like papers, code, and data. DOIs allows tracking and citing research products. For this reason, scientists who share code and data through GitHub are strongly encouraged to also submit GitHub repository content to a long-term data archive[25]. Fortunately, both long-term repositories mentioned above (Zenodo and Figshare) have integrations with GitHub. After linking one's GitHub repository to Zenodo or Figshare, every time a release (Box 1) is made, a snapshot of the entire repository is archived with a versioned, citable DOI. Linking one's GitHub repository with Zenodo, etc. to obtain a DOI helps work become findable, gives proper attribution, and can ensure long-term stability[26].

Many researchers believe that their code is not useful because their analysis is context-specific and not explicitly designed for re-use like software. However, there are many reasons to share data and code beyond re-use. Even if code lacks detailed annotation, it shows the exact steps taken to conduct an analysis, and therefore provides the most detailed look into how to reproduce a given analysis[27]. This is important in light of the reproducibility crisis[28] and will become increasingly important to the collective scientific enterprise as advances in computing power and accessibility unlock the ability to conduct 'big data' meta research with data that has already been collected by others. Failing to include data and code with publications leaves future scientists with fewer open resources from which to understand the world.

An important aspect of making code citable and reusable is adding an appropriate licence. Code without a licence is (by default) actually provided more protection from reuse than code with an open licence. The standard GitHub licensing options are best suited for software, so depending on how you want your code to be used you can choose one of the other license options. For example, if your code is intended only for your specific analysis, consider a Creative Commons License. If you wish to receive attribution for any reuse of your code, consider a CC BY 4.0 license. If you have built an app, tool, package, or other product that you would like others to use and would like attribution for any reuse of your code, consider the GNU General Public License v3. This license also prohibits the re-user from making their re-used version private. To help navigate through the potential licenses available to you and their attributes the Choose a License (https://choosealicense.com/non-software/) website can offer further guidance.

Collaborative (code) editing and asynchronous working

GitHub facilitates collaborative coding and asynchronous working because researchers can seamlessly access and contribute to data and code regardless of disparities in schedules or location. Project work can also be clearly split among team members, giving them the flexibility to contribute when it best

fits their schedule. We acknowledge that there are important differences between the average software developer and EEB researcher using GitHub, and that not all of GitHub's features are useful for research workflows (Table 1). However, git's core features such as forking and branching (Box 1) allow for simultaneous coding on different versions of the same research project, and alternative versions can be discussed and resolved with GitHub. When EEB projects are contained on the GitHub platform, researchers can stay abreast of progress made by other collaborators without the need for meetings or emails. Version control[29] features also allow users to make progress and changes without worrying about irreparably writing over someone else's work. By enabling more comprehensive remote collaboration, GitHub encourages the exchange of ideas among researchers at different institutions and in different countries, which can serve to improve the quality of the research itself by providing open access to data and code. In academic settings, GitHub can also facilitate interactions between research advisors and advisees, providing a platform for students or other trainees to share in-progress code, and flag specific challenges or questions for their supervisors or mentors. Periodic code review[30] can also help advisors identify errors early in the process, and inform further training and mentorship to fill gaps in skills.

Writing a manuscript

Beyond supporting collaboration at the level of code, GitHub can be used for collaboratively writing manuscripts. Writing a manuscript in GitHub and storing it with associated data and code all in the same repository increases scientific reproducibility because files associated with a manuscript can be found in one place. Co-authors can contribute new text to a manuscript or suggest revisions through GitHub's robust pull request feature (Box 1) which provides a line-by-line view of all proposed changes. Further, authors can make use of the Discussions tab to suggest relevant papers to be cited, and can raise issues during the writing process that can be assigned to collaborators (Table 1).

While GitHub is not considered as user-friendly for manuscript development as conventional text processors[8], it has been substantially improved with recent tools. Manuscripts can be written on GitHub with Markdown. Add-ons like HackMD (https://hackmd.io), can enable real-time collaboration for individual Markdown documents. Many tools exist for extending Markdown and Pandoc to add formatting features necessary for scientific writing like in-text citations and figure and table cross references.

We wrote this manuscript using Manubot[31], a collaborative manuscript platform that uses Markdown for writing and GitHub for storing and tracking changes to a manuscript over time. Manubot uses a GitHub Actions-based typesetting system to compile individual Markdown files stored in a GitHub repository into a single LaTeX document, which can be displayed in Word, HTML, or PDF formats. The resulting manuscript can also be compiled using a journal's .tex template to match their formatting requirements. Since this tool reruns the entire manuscript compilation process with any change to the underlying repository, it can also accomodate continuous integration of code updates into figures and tables with additional GitHub Actions (e.g. https://github.com/SORTEE-Github-Hackathon/manuscript/tree/main/.github/workflows). Manubot also allows for straightforward citation management based on URLs or DOIs. For more examples of Manubot being used for manuscripts, see[31].

Since Manubot works on documents in a distributed format, it can be difficult to edit manuscripts for overall flow with only this tool. We employed hypothes.is (https://hypothes.is) to write comments on the HTML manuscript document produced by Manubot, which we then addressed by committing changes to the underlying Markdown files via pull requests. Other authors can reply to the comments to indicate agreement or disagreement, and to note when changes have been made. Other tools can also be used for version control of scientific manuscripts including R Markdown via the bookdown package (https://bookdown.org/), jupyter notebooks[32] and a relatively new tool, Quarto (https://quarto.org).

Peer-Review

Peer review plays a critical role in the scientific evaluation of EEB research. GitHub provides an open and transparent platform that can be used for either directly providing feedback on research products or building a to-do list from reviewer comments. Peer review of research software by the open software development community rOpenSci (https://ropensci.org/software-review/) and of research software and associated manuscripts by the Journal of Open Source Software (https://joss.readthedocs.io/en/latest/submitting.html) requires that submitted work is hosted on GitHub and their review processes make use of GitHub issues. rOpenSci's efforts have resulted in many well-used R packages for ecology research including rfishbase[33] and taxize[34].

GitHub can also be used as a hub for reviewers and authors during the peer review process of an ordinary research manuscript. If the code associated with a manuscript is made available at the time of submission (e.g. via a link to a GitHub repository in a Data Availability Statement), peer-reviewers may be able to offer more helpful suggestions on written methods and may even make comments on the code itself, potentially catching bugs or errors before publication. GitHub can also be used as a hub for reviewers and authors during the peer review process of an ordinary research manuscript. GitHub issues can be used to organize and discuss reviewer suggestions and to assign them to co-authors (See example in https://github.com/BrunaLab/HeliconiaDemography/issues?q=is%3Aissue+label%3A%22reviewer+comment%22+). When reviewer comments are posted as separate issues, authors can comment on the issues to discuss possible changes and assign themselves to indicate which comments they intend to handle. Co-authors can then integrate their edits and responses to reviewers using pull requests.

Some specialized journals, such as the the Journal of Open Source Software (https://joss.readthedocs.io/en/latest/submitting.html), require that submitted work is hosted on GitHub and their review processes make use of GitHub issues. Peer review of research software by rOpenSci (https://ropensci.org/software-review/) also makes use of many features of GitHub to streamline and automate parts of the review process. rOpenSci's efforts have resulted in many well-used R packages for ecology research including rfishbase[33] and taxize[34]. Although the use of GitHub in peer-review is currently restricted to these specialized cases, they may provide a model for peer-review in more traditional EEB journals for manuscripts that make use of code.

Open science discussion

Research papers are condensed outputs that hide the underlying intellectual and computational workflows, including the treatment of the raw data and analytical steps. Granting readers access to code and other documentation of the analysis allows them to retrace and comprehend analytical decisions. GitHub provides a platform to access all aspects of the project, and can be linked to platforms that create citable DOIs, rather than just the final manuscript. While often thought of as storage for data and code, GitHub repositories can also be used to publish a time-stamped preregistration of research plans and hypotheses.

Conventional research practices typically rely on one or two people running and checking the data analyses, while most coauthors (and readers of the subsequent publication) see only the final results and a verbal description of the analytical steps. In the developmental stages, collaborators can directly see the code for the analysis, manipulate and explore the data themselves, and check for errors. Cynically, there is also more insurance against nefarious colleagues that may be tempted to distort results[35]. Collaborators as well as pre-publication and post-publication reviewers are better positioned to discover questionable findings if they have full and transparent access to the project.

This transparency can similarly be extended beyond coauthors to the entire scientific community. Publishing the data and reproducible workflows along with the manuscript allows any reader to review the analysis and reproduce the experiment[36]. Supplying code for (novel) methods that are proposed or used also reduces barriers to knowledge and can greatly improve the ability of others to build on existing work, resulting in greater proliferation and accessibility for a broader audience. GitHub even provides a useful Discussions Forum (https://docs.github.com/en/discussions) that aids the direct communication with repository owners, as well as the GitHub Community (https://github.community/) forum for more general questions and sharing of expertise (Table 1.

Project continuity

The development of research software continues, and so does the need to consider project continuity, especially in EEB where graduate students, research assistants, and postdoctoral fellows often hold relatively short-term positions[37]. Often with these types of short-term research projects, once the contract expires, the research software upkeep tends to fall off as the researchers move on to new projects. Additionally, if the research software is kept on only the researcher's personal devices, it becomes increasingly difficult to access the software and code for future uses. As projects or contracts end, there should be a handover period of this software in order for the next cohort of researchers to reuse what was already developed[8,37].

GitHub facilitates project continuity among research software and research code by providing tools that make this handover period easier. As we have already mentioned, using Git for code in Ecology and Evolution can allow for a "paper trail" of sorts to be created for the research software, thus allowing for future users of the code access to the entire history of the project[14]. Additionally, GitHub allows for repositories and organizations to have designated Code Owners[38]; these code owners can change through time allowing for the transition of research software from one cohort of researchers to the next[26]. There are also multiple means by which to archive software dependency information with GitHub. Dependencies can be described in README files, added to environment.yml and/or an environment.txt file, or stored in a repository using tools such as renv (https://rstudio.github.io/renv/articles/renv.html). This ensures that as packages continue to develop with new syntax, code will still run as it did when it was developed.

Within EEB projects, tasks are often divided among contributors taking various roles (see CRediT taxonomy, https://casrai.org/credit/). The creation of project repositories is commonly the purview of those involved in the software, formal analysis, and/or visualization components of the project through their roles as code writers. However, the structural components of a typical GitHub repository and the derived EEB-specific templates can provide functional ways for other collaborators not contributing to code to be engaged in aspects of repository design in a way that improves institutional memory and facilitates project continuity. These other collaborators can offer many contributions to repository design and development, and their active involvement can both aid authors ability to act as guarantors of the project, and the clarity and reproducibility of the project for future users. In (Figure 2), we highlight several elements of good repository structure, and the various ways that contributors may interact with them.

GitHub Organizations

GitHub Organizations offer a shared virtual space that allows a team to work in different repositories, while remaining tied together under a larger figurehead, such as a laboratory, a department, an organization, or a large project involving several teams. Organizations are well-suited to ensure larger projects with many steps or moving parts are constrained to one virtual space, where outputs and sub-projects can be easily accessed and located without relying on any one individual. Because the repositories are grouped in one virtual space, members can reference and contribute to each other's

work without necessarily being part of the same repository, broadening the accessibility and longevity of code and writing contributions.

Contributors can be assembled into teams within an organization, which allows administrators to assign roles and tasks to groups of people. Whereas access to repositories is usually assigned to individual contributors, Organizations facilitate the management of access permissions by allowing each team to be granted access to certain repositories, and not to others. This ensures that more sensitive repositories remain as restricted as needed, while repositories with greater general interest can be easily accessible to many members at once.

As an example, GitHub Organizations are particularly well-suited to house documents and projects within a laboratory, such as research compendia, codes of conduct, protocols, training documents, and other such documents that evolve collaboratively over time and are relevant to many colleagues. In this way, teams can have full ownership of repositories within an organization, while ensuring that these materials stay accessible to the laboratory after people have moved on or lost their copies of project data. This application extends to research centres, which may include several distinct projects that remain linked under a given institution, such as the German Centre for Integrative Biodiversity Research (iDiv, https://github.com/idiv-biodiversity). Of course, the utility of this tool goes beyond laboratories - they are useful to structure the organization, presentation, and outcomes of working groups such as the hackathon which inspired this paper (SORTEE-Github-Hackathon, https://github.com/SORTEE-Github-Hackathon) by keeping track of all materials as ideas develop and take shape in one virtual space. Organizations are also convenient for hosting a set of related learning materials such as a set of lectures or workshops, such as the Québec Centre for Biodiversity Science R Workshop Series (QCBSRworkshops, https://github.com/QCBSRworkshops) or the University of Edinburgh's Coding Club (Coding Club, https://github.com/ourcodingclub), which may be updated by an ever-evolving group of contributors over time.

Using GitHub organizations as a research group or even for a handful of individuals working on a group of projects can be incredibly useful for all involved. GitHub organizations are relatively easy to set up, and especially easy to manage as membership to the organization changes through time. Not only is it a useful way to store repositories of lab-related research products, but it is also incredibly helpful for storing "living documents" that may be edited frequently, and may be linked to a lab website (that could also be generated via a repository that lives within the organization). The use of the "Teams" feature can allow certain groups to have varying levels of access to repositories in the organization with a select group having push access to some repositories but not others. This can manifest in a group working on some common dataset(s) to have push access to the handful of repositories used for processing sequence data, while another group of students/researchers may have push access to an entirely different set of repositories. The organization structure also allows for easy tracking of issues, projects, and discussions related to the research group, and provides group leads an easy birds-eye view of the progress going on across multiple projects.

Discussion

The promise of GitHub for EEB researchers

There have been many calls for researchers outside of the software development community to join the 73 million GitHub users for their collaborative research[3,25,39]. This call comes in light of both the continual shift toward open-science and increasing computational and data requirements in EEB. Until now, resources and practical guidance specifically focused on using GitHub within the EEB community have been dispersed in blog posts and video tutorials (Box 2 Box[box:tips?]). We felt these resources have been extremely useful for us to learn to use GitHub in our own work, and that a collation of the

main ideas into one medium, while adding on our personal perspectives, would be of use to the EEB community.

In this paper, we described 12 tractable ways that EEB researchers can leverage GitHub to enable more transparent and collaborative research (Figure 2). Many of the examples are specifically meant for first-time GitHub users and can likely be adopted with just a little practice (storing data, creating virtual notebooks, making code citable). For example, storing code and data and making it citable generally just involve creating a repository on GitHub, pushing code to the repository, and then going through the necessary steps (e.g., connecting a GitHub repository with Zenodo; see below) for creating a DOI for the repository. These actions are often covered in any introductory tutorial on GitHub and take little effort to implement. On the other hand, some other examples we described here, including material development, web hosting, and delivery, may require a greater time commitment, but have the potential to make EEB research more open, accessible, and collaborative than ever before. Managing full research projects or research labs on GitHub will require careful thought as to how to delegate tasks such as reviewing pull requests or creating issues, as well as thought as to how modular to make the research project or research lab (i.e., which repository will be used for what, and how many repositories are needed). For example, collaboratively writing a paper using GitHub, as we have done here, will involve a learning curve for co-authors less familiar with the intricacies of GitHub, and also require overhead to set up the repository using GitHub actions. Despite the many potential applications of GitHub to EEB research, we acknowledge that there will still be many times when researchers might look to other platforms for research collaboration.

Other platforms for collaboration

Though we see GitHub as a useful tool for collaboration in EEB, we describe two use cases where, to our knowledge, GitHub's features still fall short of the type of highly collaborative work emblematic of EEB research. First, real-time collaborative editing (e.g., as on a shared Google Doc or a Word document stored on Dropbox) is not possible on GitHub. There are websites outside of the GitHub ecosystem that are built on top of the GitHub architecture that allow real-time collaborative editing (e.g., hackMD (https://hackmd.io/) or replit (https://replit.com/)). We used HackMD at two key points in writing our manuscript when real-time co-writing was essential: when taking meeting notes and writing the outline of our paper. Second, we looked to other software when working on figures and tables. Though creating tables and figures on GitHub using markdown or other scripting languages is possible, we found that it was not practical at the early brainstorming stages. We needed to rapidly iterate on figure and table design, share feedback through comments, and merge/reorder ideas when necessary. For these reasons, we used Google Slides for working on figures and Google Sheets for working on tables. As our figures and tables moved towards more finalized forms, some co-authors chose to create the tables and figures using R and Markdown which could then be tracked using the same version control system as the rest of manuscript.

Why aren't more EEB researchers using GitHub?

Though GitHub has been available as a platform for more than a decade, its uptake among EEB researchers, especially as a tool for collaboration, has been slow. Here we discuss four potential barriers to GitHub use in EEB.

First, there may be a hesitation to use GitHub due to the somewhat steep learning curve to using the platform combined with limited instruction available through traditional university courses. When GitHub is taught within an EEB context, it is usually accompanying coursework in topics such as statistical programming, and some students may find it overwhelming to juggle learning git alongside scripting languages, statistical theory, and file system navigation, especially when many may also be new to using command-line interfaces in general. Instructors likewise may confuse the expected

digital literacy of younger students with computational fluency, even when modern technology increasingly abstracts many relevant concepts through search optimization and preponderant IDEs (Integrated Development Environments), or 'point-and-click' user interfaces.

Second, while many EEB researchers take advantage of GitHub for individual use, collaborative use may lag due to how researchers traditionally divide labor within projects. Despite broad utility, GitHub remains a tool predominantly used by computer scientists and software developers. EEB researchers may take the view that GitHub is a platform that only needs to be used by individuals writing code, and may silo those aspects of projects to a single individual. Those assumptions may obscure the utility of GitHub for tasks other than traditional data analysis and code development, or how GitHub can facilitate the integration of code with non-coding aspects of projects through the practice of repository design. However, we emphasize that there are opportunities for collaboration using GitHub by researchers of all skill levels or time constraints (Table 1); for example, project stakeholders can provide a list of use-cases or highlight important conceptual components of a project using the issues or discussions functionality of GitHub.

A third barrier may come from general reluctance to share data and code publicly, or technical and logistical issues. The default public nature of GitHub usage can add additional pressure to students and scientists learning to use the platform. We also note that large file storage is discouraged (and limited) on GitHub but add-ons do exist (e.g. https://git-lfs.github.com/) that permit data storage, and increasing integration between platforms (e.g.[40]) allows data to be stored away from GitHub and linked dynamically. We suspect a major additional barrier to EEB researchers is a distinct lack of GitHub help documents for non-English researchers in ecology and evolution, meaning that EEB researchers potentially miss the opportunity to fully understand the importance of version control, reproducibility, and other benefits of GitHub.

Lastly, GitHub has both free and paid plans. When projects get highly collaborative they may have to pay for additional GitHub support. The acquisition of GitHub by Microsoft has led to some concerns over the future of free plans and several biodiversity data managers have begun to switch to Open Source Git services (e.g. Bitbucket and GitLab). At this point however, there is little practical difference for EEB researchers between the paid and free GitHub plans.

Box 2

10 Tips for getting started in GitHub
1. **Check for an existing solution to your problem.** The GitHub Help webpage (https://docs.github.com/en) contains extensive and detailed documents with helpful screenshots. It is a good starting point for handling an issue, and has troubleshooting tips for specific problems. Alternatively, consider Tweeting your issue. There is a large community of GitHub users around the world who have likely faced analogous problems and may be able to provide quick solutions. Third, try to follow blogs (e.g., https://github.blog), Twitter accounts or YouTube channels that regularly post practical solutions about the most widely-used web platform for common GitHub issues.
2. **Consider taking free courses.** such as those from Software Carpentry[41] and sharing these courses with your lab members or colleagues.
3. **Take advantage of GitHub as an asynchronous working tool for team-based projects.** See the repository for this paper (https://github.com/SORTEE-Github-Hackathon/manuscript) as an example of a collaborative manuscript that includes discussions, issues, and a website via GitHub.
4. **The GitHub Learning Lab (https://lab.github.com).** allows you to learn GitHub basics through short projects and tasks, and allows you to get feedback from their Learning Lab bot.
5. **Check out the following markdown cheatsheet (http://markdownguide.org/basic-syntax).** so that you can write clear metadata README files for your repositories.
6. **The Jenny Bryan universe of GitHub material.** provides a thorough and accessible introduction for a multitude of research-related uses for GitHub, and includes a book[42], statistics course[43] and academic article[7].

10 Tips for getting started in GitHub
7. **Don't be afraid of trial and error.** One of the best ways to learn GitHub is the trial and error method. Learning from your own mistakes can be the better way to master your GitHub abilities. In any case, GitHub has the advantage of making it easy to go back to any steps that you desire via version controlling if you make mistakes.
8. **If you are an educator, include lectures on reproducibility and tools for creating reproducible workflows in the curricula.**. Some graduate programs now include coursework on course Rmarkdown and GitHub. Getting students started with these tools earlier will prevent the resistance that comes from working with a less reproducible workflow for a longer period of time.
9. **Try to begin committing with GUI (Graphical user interface) tools.** like GitHub Desktop (https://desktop.github.com), git-gui (https://git-scm.com/docs/git-gui), RStudio (https://www.rstudio.com), Visual Studio Code (https://code.visualstudio.com), Atom (https://atom.io), GitKraken (https://www.gitkraken.com) tools instead of CLI (Command line interface) tools such as Terminal or Console for more advanced features.
10. **Get help deciphering GitHub Notifications.** Try using tools like Octobox (https://octobox.io) to disentangle and manage multiple notifications from distinct GitHub projects.

Conclusion

We provide 12 practical ways that Ecologists and Evolutionary Biologists can incorporate GitHub into their research workflows, and include definitions (Box 1) and key user groups (Figure 1) that can help researchers prioritize which GitHub skills to learn first. Some GitHub uses are highly collaborative (e.g., open science discussion and collaborative code editing) while others are focused on individual actions (e.g., storing code/data, building a website). Regardless of the degree of collaboration, GitHub use in ecology and evolution has the potential to make the field more open and transparent than ever before. Our paper provides the most comprehensive review of how EEB researchers can use GitHub to date, and we encourage EEB researchers at any career stage studying any topic to try GitHub as a platform for sharing and collaboration.

Acknowledgements

This manuscript arose from a hackathon at the Society for Open, Reliable, and Transparent Ecology and Evolution (SORTEE) virtual meeting in July 2021.

RCO was funded by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, Earth and Environmental Sciences Division, Data Management program under contract number DE-AC02-05CH11231.

Code and data availability

The source code and data for this manuscript are available at https://github.com/SORTEE-Github-Hackathon/manuscript.

Figures

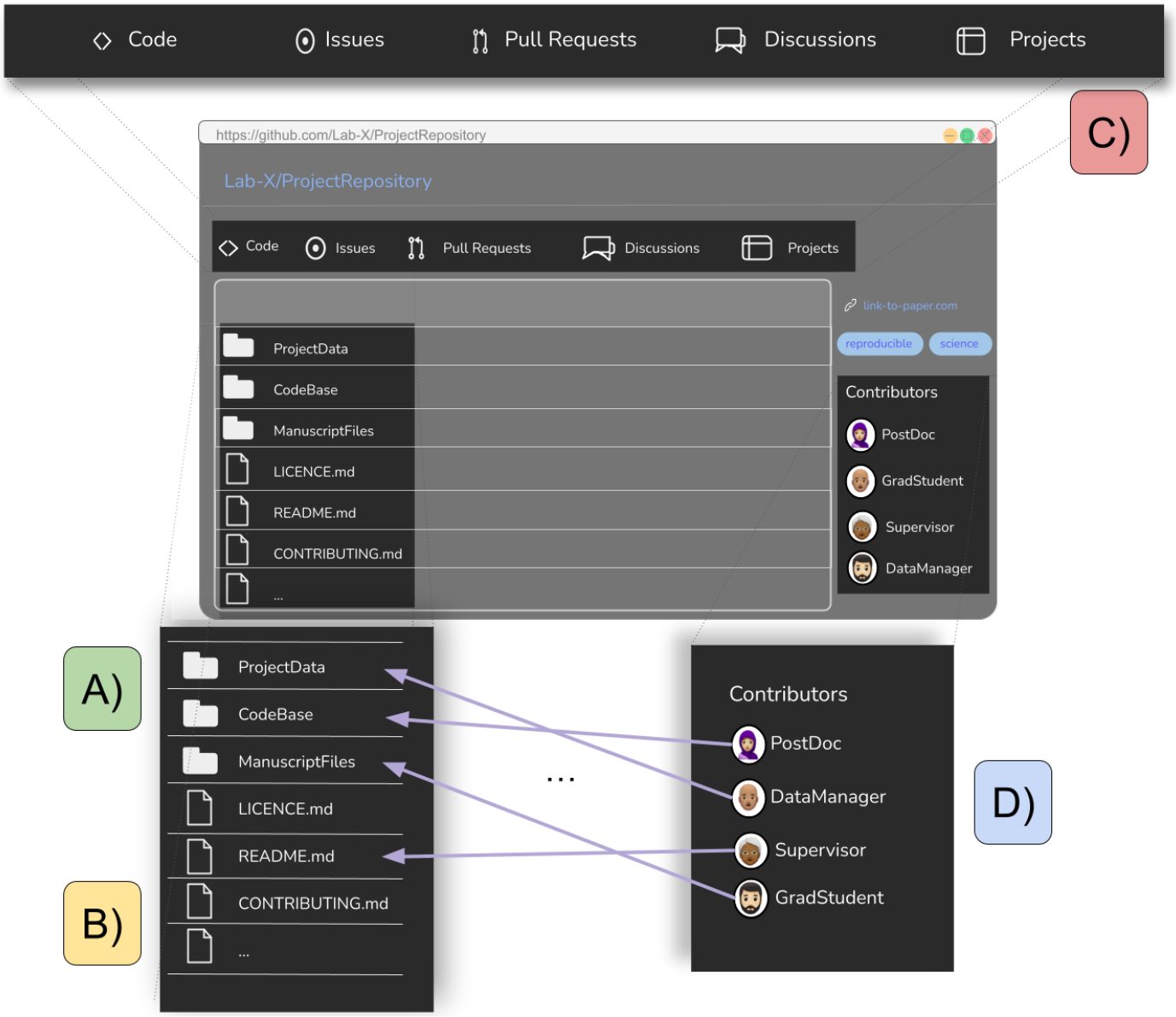

Figure 1: An overview of git's core features. A) Multi-faceted components allow for code writing, small data storage, manuscript writing, and project management to all be done in one place. B) Issues, Pull Requests, Discussions, and Projects allow for team members to ask for feedback, suggest fixes, discuss related ideas, and keep track of all the moving parts of a project. C) All collaborators on a project can be a part of a single repository, with varying push privileges and responsibilities. D) CONTRIBUTING.md, LICENCE.md, & README.md files can allow new team members or others wanting to use materials to understand the project components and learn how they can engage with the project and existing team members.

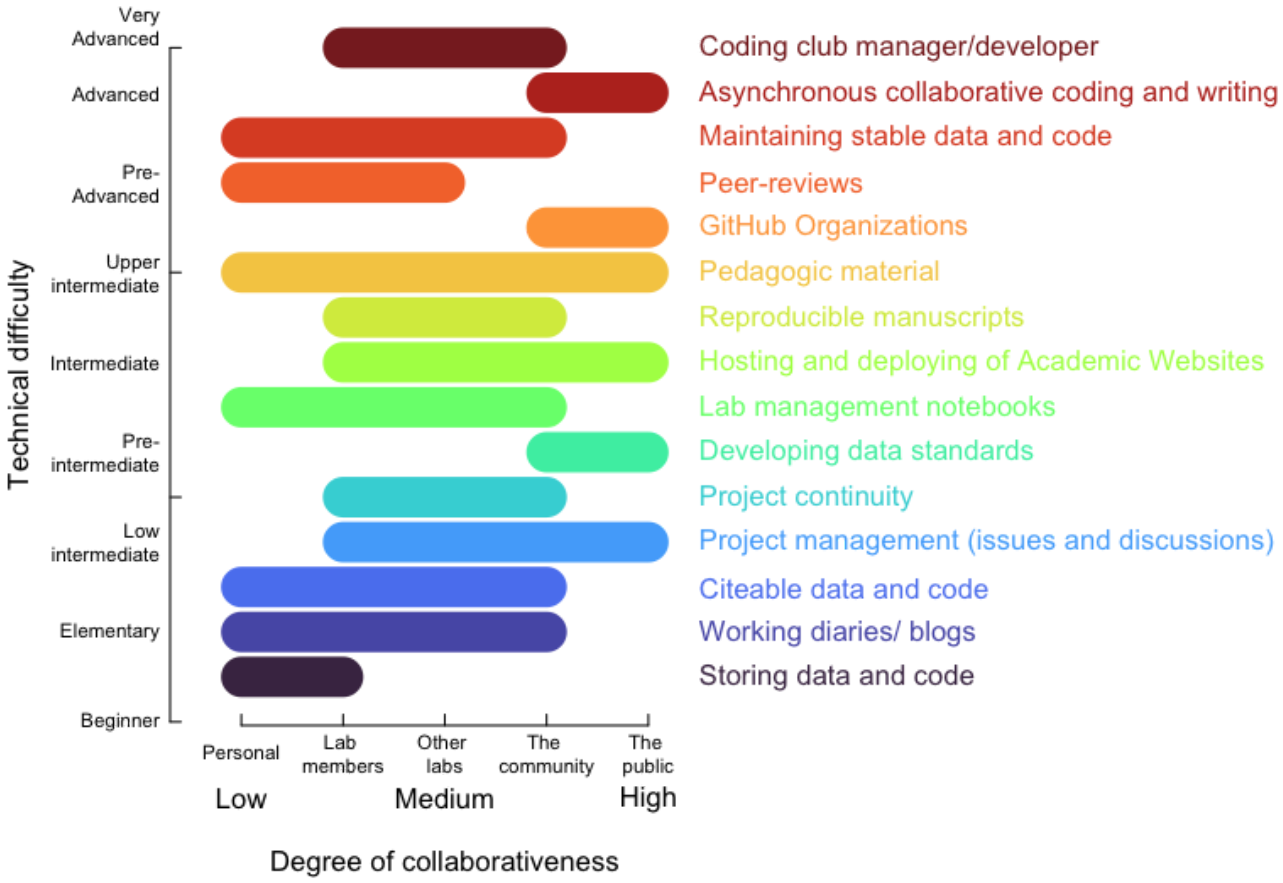

Figure 2: A summary of ways GitHub can be used showing technical difficulty and degree of collaboration for each. Activities higher on the vertical axis require usage knowledge of more GitHub features than activities lower on the axis. On the horizontal axis, each activity spans a region representing who is potentially involved with or benefits from each activity. For example, storing data and code mainly benefits individual researchers or members of a lab group while making data and code citable and reproducible benefit other labs and the larger community as well. Independently of a users knowledge level of GitHub features, there are ways to use GitHub that allow tapping unto one of the most salient benefits of the platform: facilitating and enhancing collaboration.

Tables

Table 1: A non-exhaustive collection of ideas for how various GitHub features could be utilized for a research project. Here we have categorized contributors/collaborators into five roles. A Project Manager owns the GitHub repository for a project, and leads the academic project (e.g., lead author of a manuscript). A co-author contributes to writing and other aspects of research, but may have limited or no experience with programming, git, and/or GitHub. A code contributor writes or edits analysis code for the project. A code reviewer could be a project collaborator or a peer reviewer who reviews project code. They are familiar with coding, but not necessarily with git or GitHub (but they are willing to learn). Finally, community members could be other researchers or non-researchers interested in reproducing results, re-using code or data, or communicating with researchers involved in the project. These roles are not mutually exclusive—a co-author could also be e code contributor and code reviewer, for example. For definitions of the GitHub features, see Box 1.

Role	GitHub repository	README	Issue	Discussion	Pull Request	Fork	GitHub Pages

Role	GitHub repository	README	Issue	Discussion	Pull Request	Fork	GitHub Pages
Project manager	Set contributor permissions, share code of conduct	Project descritption, citation, DOIs	Assign tasks to collaborators	Discuss project directions and goals	Approve and incorporate edits to code and/or writing		Share up-to-date reports, figures, or draft manuscript
Co-author	Edit Markdown text or add files		Propose changes involving code (e.g. analyses, figures)	Discuss proposed changes to manuscript			
Code contributor			Suggest code changes		Contribute changes to code, initiate code review		Contribute to project website
Code reviewer	Find all code related to a project		Highlight specific lines of code and make suggestions		Review or recommended changes in code		
Community			Suggest additional features and report bugs	Ask questions about data and code		Create a linked, editable copy of the repository	View project website

Table 2: a comparison of technologies...

Guild	Software	Version control	Backup (cloud)	Passive collaboration	Active real-time collaboration	Free $	Permanent (DOI)	Storage limits	GitHub Integration

Multi-tool	Git Hub	yes	yes	yes	NA	Broadly limited free version. Advanced features are provided for free to students and education professionals.	A DOI can only be obtained when integrating to other services that can mint DOI (e.g. Zenodo, OSF).	100 MB per file, 500 MB per private repository (2 GB for paid accounts). 100 GB for public repositories. Larger files (up to 2 GB) can be attached to releases	NA
Multi-tool	OSF	yes	yes	yes	yes	yes	yes	25 GB for private projects, up to 5GB per file, plus partner add-ons, 50GB for public projects	yes
Long-term (public) data repositories	PANGAEA	yes	yes	yes	NA	yes	yes	10 GB free	NA
Long-term (public) data repositories	Zenodo	afterpublished	afterpublished	NA	NA	yes	yes	50 GB per dataset	yes
Long-term (public) data repositories	Dryad	afterpublished	afterpublished	NA	NA	some journals cover cost	yes	300 GB per publication	Can link to individual files (not entire reopository); not really integrated
Long-term (public) data repositories	Figshare	yes	yes	yes	NA	yes	yes	20 GB free, up to 5 TB	yes

Temporary (personal) drive storage	Google Drive	yes	yes	yes	yes	limited free version & paid	NA	15 GB free, up to 100 GB with Google One	yes
Temporary (personal) drive storage	Box	limited	yes	?	?	NA	NA	Unlimited total size for subscription	yes
Temporary (personal) drive storage	DropBox	limited	yes	yes	yes	limited free version & paid	NA	2 GB free	yes
Temporary (personal) drive storage	OneDrive and the Office Suite	yes	yes	yes	yes	limited free version & paid	NA	5 GB free, up to 1TB paid	yes
Collaborative code/text editors	Overleaf (online latex editor)	yes	yes	yes	NA	NA	NA	1MB for individual .tex, 50MB for individual files, unlimited project size	yes
Collaborative code/text editors	Jupyter Notebook	yes	?	yes	with Colab	yes	NA	via Binder: no hard limit, but suggests no files >100 MB, can also store on GitHub or Google Colab	yes
Collaborative code/text editors	HackMD	yes	yes	yes	yes	yes	NA	3 documents free, private invitee limits	yes

References

1. Hannay, J. E. *et al.* How do scientists develop and use scientific software? in *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering* (IEEE, 2009). doi:10.1109/secse.2009.5069155.

2. Perkel, J. M. Challenge to scientists: does your ten-year-old code still run? *Nature* **584**, 656–658 (2020).

3. Perkel, J. Democratic databases: science on GitHub. *Nature* **538**, 127–128 (2016).

4. Prabhu, P. *et al.* A survey of the practice of computational science. in *State of the Practice Reports on - SC '11* (ACM Press, 2011). doi:10.1145/2063348.2063374.

5. Wilson, G. *et al.* Best Practices for Scientific Computing. *PLoS Biol* **12**, e1001745 (2014).

6. Build software better, together. *GitHub* https://github.com.

7. Bryan, J. Excuse Me, Do You Have a Moment to Talk About Version Control? *The American Statistician* **72**, 20–27 (2018).

8. Ram, K. Git can facilitate greater reproducibility and increased transparency in science. *Source Code Biol Med* **8**, (2013).

9. Perez-Riverol, Y. *et al.* Ten Simple Rules for Taking Advantage of Git and GitHub. *PLoS Comput Biol* **12**, e1004947 (2016).

10. Blischak, J. D., Davenport, E. R. & Wilson, G. A Quick Introduction to Version Control with Git and GitHub. *PLoS Comput Biol* **12**, e1004668 (2016).

11. *Learn Git In 15 Minutes*.

12. Hester, J. B., the STAT 545 TAs, Jim. *Let's Git started | Happy Git and GitHub for the useR*.

13. Coding Club: A Positive Peer-Learning Community. https://ourcodingclub.github.io/.

14. Lowndes, J. S. S. *et al.* Our path to better science in less time using open data science tools. *Nat Ecol Evol* **1**, (2017).

15. Leibzon, W. Social network of software development at GitHub. in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (IEEE, 2016). doi:10.1109/asonam.2016.7752419.

16. Briney, K., Coates, H. & Goben, A. Foundational Practices of Research Data Management. *RIO* **6**, (2020).

17. Alston, J. M. & Rick, J. A. A Beginner's Guide to Conducting Reproducible Research. *Bull. Ecol. Soc. Am.* **102**, (2021).

18. Marwick, B., Boettiger, C. & Mullen, L. Packaging Data Analytical Work Reproducibly Using R (and Friends). *The American Statistician* **72**, 80–88 (2018).

19. About large files on GitHub. *GitHub Docs* https://ghdocs-prod.azurewebsites.net/en/repositories/working-with-files/managing-large-files/about-large-files-on-github.

20. Kanza, S. *et al.* Electronic lab notebooks: can they replace paper? *J Cheminform* **9**, (2017).

21. Schnell, S. Ten Simple Rules for a Computational Biologist's Laboratory Notebook. *PLoS Comput Biol* **11**, e1004385 (2015).

22. About projects (beta). *GitHub Docs* https://ghdocs-prod.azurewebsites.net/en/issues/trying-out-the-new-projects-experience/about-projects.

23. Xie, Y., Allaire, J. J. & Grolemund, G. *R Markdown: the definitive guide*. (CRC Press, Taylor and Francis Group, 2019).

24. Dawson, C. *Building tools with GitHub: customize your workflow*. (O'Reilly, 2016).

25. Crystal-Ornelas, R. *et al.* A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats. *Earth Space Sci* **8**, (2021).

26. Hampton, S. E. *et al.* The Tao of open science for ecology. *Ecosphere* **6**, art120 (2015).

27. Mislan, K. A. S., Heer, J. M. & White, E. P. Elevating The Status of Code in Ecology. *Trends in Ecology & Evolution* **31**, 4–7 (2016).

28. Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature* **533**, 452–454 (2016).

29. Crystal-Ornelas, R. *et al. Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution*. https://SORTEE-Github-Hackathon.github.io/manuscript/ (2022).

30. Song, X., Goldstein, S. C. & Sakr, M. Using Peer Code Review as an Educational Tool. in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (ACM, 2020). doi:10.1145/3341525.3387370.

31. Himmelstein, D. S. *et al.* Open collaborative writing with Manubot. *PLoS Comput Biol* **15**, e1007128 (2019).

32. Lasser, J. Creating an executable paper is a journey through Open Science. *Commun Phys* **3**, (2020).

33. Boettiger, C., Lang, D. T. & Wainwright, P. C. rfishbase: exploring, manipulating and visualizing FishBase data from R. *Journal of Fish Biology* **81**, 2030–2039 (2012).

34. Chamberlain, S. A. & Szöcs, E. taxize: taxonomic search and retrieval in R. *F1000Res* **2**, 191 (2013).

35. January 18, P. L. B. K. & Pm, 2022. 2:51. #PruittData and the Ethics of Data in Science. *Ecology for the Masses* https://ecologyforthemasses.com/2020/02/04/pruittdata-and-the-ethics-of-data-in-science/ (2020).

36. Culina, A., van den Berg, I., Evans, S. & Sánchez-Tójar, A. Low availability of code in ecology: A call for urgent action. *PLoS Biol* **18**, e3000763 (2020).

37. Fehr, J., Himpe, C., Rave, S. & Saak, J. Sustainable Research Software Hand-Over. *JORS* **9**, 5 (2021).

38. About code owners. *GitHub Docs* https://ghdocs-prod.azurewebsites.net/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners.

39. Anbaroğlu, B. A collaborative GIS programming course using GitHub Classroom. *Transactions in GIS* **25**, 3132–3158 (2021).

40. Connect GitHub to a Project - OSF Support. https://help.osf.io/article/211-connect-github-to-a-project.

41. Madicken Munk *et al.* swcarpentry/git-novice: Software Carpentry: Version Control with Git, June 2019. (2019) doi:10.5281/zenodo.3264950.

42. Hester, J. B., the STAT 545 TAs, Jim. *Let's Git started | Happy Git and GitHub for the useR*.

43. Bryan, J. & TAs, T. S. 545. *STAT 545*.