Not just for programmers: A friendly guide on the versatility/benefits of GitHub for accelerating collaborative research in Ecology and Evolution

This manuscript (permalink) was automatically generated from SORTEE-Github-Hackathon/manuscript@9f38d4e on March 28, 2022.

Authors

- **Dylan G. E. Gomes**
  0000-0002-2642-3728 · dylangomes
  Cooperative Institute for Marine Resources Studies, Hatfield Marine Science Center, Oregon State University, Newport, OR, United States

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

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

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

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

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

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

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

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

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

- **Allison D. Binley**
  0000-0001-8790-9935 · adbinley · AllisonBinley
  Department of Biology, Carleton University, Ottawa, ON K1S 5B6, Canada

Abstract

Ecological and evolutionary researchers have become critically dependent on computional code to achieve scientific elaboration. With this, the use of efficient forms to collaborate, share and reproduce 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. While scientists in natural sciences have been encouraged to use GitHub, its adoption is not widespread due to the lack of domain-specific information and guidelines. To help ecology and evolutionary researchers adopt useful features from Github in their own workflows, we review thirteen 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. Because not all GitHub features are essentially useful for every researcher, we allocate existing development and collaborative workflows within Github – such as, tracking tasks, discussing edits to a manuscript, and submitting code to a repository – to the most common research development roles in ecology and evolution (project manager, coauthor, code contributor). As more ecologists and evolutionary biologists migrate their workflows to GitHub, the broader scientific community stands to benefit from disciplines that push the boundaries of collaborative, transparent, and open research.

Introduction

Most scientists, including ecologists and evolutionary biologists, have become critically dependent on computational tools in their research [see 1]. Researchers now write and use code as part of their research workflow to perform a wide-variety of tasks facilitating scientific elaboration, ranging from data management, data analysis, study replication, to the application and the development of tools for hypothesis testing. This strong code-dependent workflow imposes steep requirements towards an efficient, well-documented process in the publication and collaboration of maintenaible scientific code [2]. To facilitate this process, scientists have been increasingly borrowing and adopting tools from information system 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] [**doi?** 10.1038/538127a]. 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-faciliting tools, and thus may find practical barriers when applying adequate standards in the maintenance of their scientific code [1,3,4]. 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 with practical workflows aiming at facilitating their scientific code and management process.

With over 73 million registered users, GitHub is the most common web-platform used for documentation and collaboration on computer code [5]. GitHub builds on the Git version control system, providing 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, as well as publishing their own. Through its version control, users have a detailed, chronological record on the files and directories stored in their repositories [6]. This workflow provides a strong and clear advantage over receiving, processing and sending files back-and-forth, a process that can easily become challenging and time-consuming in projects extending in time and in the number of collaborators [7]. 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. [8]

Git is the version control system that enable all the collaborative tools available on GitHub. In Git, versions of files and directories are uniquely identified as “commits”, allowing one to identify and track modifications line-by-line. Although the understanding of basic concepts of Git (such as commit, push, pull, checkout; see (Box 1)[#box-1-definitions]) is necessary, the GitHub web-based platform and its integrated development environments (such as the GitHub Desktop) allow users to manage their repositories without using command-line sessions. We do not focus on Git in this study, but we recommend users to explore the many resources providing a detailed information on Git, such as journal articles [8,9], videos and books [10/].

The voluminous user-community and the numerous resources providing pedagogical instruction material on how to use GitHub streamlined its widespread use (CITE). Nevertheless, although researchers in ecology and evolutionary biology (EEB) have been encouraged to adopt GitHub in part of their research process [11,12], its use is still not widespread. Because GitHub and its features have been created centered on collaboration for software development in information systems [13], first-time users from domains without formal training in information technology can face challenging learning curves. Morever, domain-specific perspectives and resources providing tractable examples and practical guidance for researchers in ecology and evolution on GitHub are scarce (but see CITE, CITE, CITE). An increased availability of data and code managenent standards – of which GitHub is one increasingly important component – make research more reproducible and collaborative [14]. More importantly, a widespread, 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 scientific research [15].

This manuscript is the result from an academic hackathon held during the 2021 conference for the Society for Open, Reliable, and Transparent Ecology and Evolutionary Biology (SORTEE). We convened a group of ~30 EEB researchers with varying levels of familiarity with using GitHub as part of their research projects to showcase and discuss how existing features can contribute to the documentation and collaboration in ecological and evolutionary 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 their research more collaborative and transparent. Here, we outline thirteen practical uses of (?) GitHub features that can help towards a more collaborative, transparent and reproducible EEB research. We also provide critical perspectives on features that could be improved and catered towards research development.

Box 1: Definitions

- **repository**: A collection of files (e.g. a directory) tracked by git. Commonly shortened to "repo"

- **commit**: Commits are like snapshots in the development of a project. 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, make a commit (e.g. "generate and include fig1 in results"), and after several commits to **push** those commits to the remote GitHub **repository**.

- **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**: Development branches can be created at any point in time and work on each branch can continue independently. This 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 arise when conflicting edits are pushed to the same 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**.

- **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 (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**.

- **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 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 are made. Some automated tests may be run and review may be required before integrating your changes.

- **merge**: Combining **commits** from two different branches together into one **branch**

Box 2

Role	Description	GitHub Feature	Use Case
Project Manager	Owner of the GitHub repository for a project. This is often, but not necessarily the main author of code and text for a manuscript or other academic project.	Permissions	The Project Manager can manage permissions of the repository to give push/pull access to contributors within their organization. They can protect the main branch from push access, if desired, to be sure all contributions are reviewed as pull requests.
		Issues	Project managers can use issues to track tasks and assign them to co-authors and code contributors. Tasks may including writing, code features, and responding to reviewer comments.
		README	The project README can contain contributor guidelines, a link to a project website (e.g. made with GitHub pages), and licence information.
		Releases	Releases mark milestones in the progression of a project (e.g. version of manuscript submitted to a journal). If a repository is linked to Zenodo, releases are automatically archived and given a unique DOI.
Co-author	Contributes writing, but has little to no experience with programming. Even if familiar with programming, they are unfamiliar with git and GitHub.	GitHub Pages	The Project Manager can use GitHub pages to share reports and figures with Co-Authors who can access up-to-date information through the project website.
		Discussions	Can be used as a forum to discuss proposed changes in a manuscript.
		Markdown Editors	Can contribute text as Markdown using a web interface through GitHub (Add File button) or through extensions like HackMD
Code contributor	Anyone who contributes code to the project.	Pull Requests	Pull Requests are a natural way to contribute code to the project. These changes can be reviewed by the project manager or code reviewers before integrating them.
Code reviewer	Anyone asked to review project code (e.g. a co-author or a peer-reviewer). They are familiar with coding, but not necessarily with git or GitHub (but they are willing to learn).	Issues	Reviewers can highlight specific lines of code and create issues reffering to them with suggestions
		Pull Requests	Reviewers can either be assigned to review pull requests (e.g. from code contributors) or can make pull requests with recommended changes.
Community	Anyone who wishes to use or adapt data or code for their own analysis	Fork	Anyone can use this feature to create a linked copy of the repository on their own GitHub account.
		Discussions	Community members can be directed to the Discussions tab as a place to ask clarifying questions about data and code.
		GitHub Pages	Can be used to create a public-facing website for the project that can be accessed by community members.

GitHub in EcoEvo examples (Part 1)

Storing and archiving version-controlled data

contributors to this section: Dylan Gomes, Emma Hudgins

Many researchers often start their use of GitHub to backup their working data and code to a remote server (Just push and pull, see Box 1, from their own repo). This saves the user time from backing up data and code on their own portable devices, such as hard drives. This also offers some peace of mind, as this information is retrievable even if one's laptop ends up at the bottom of a lake. Thus, an additional benefit of this 'cloud' storage is that one's GitHub repository can be accessed by any machine with internet access, allowing the user to be more mobile if they wish to both work from home and the office from different computers. Each time a user pushes changes to their repository, GitHub tracks what these changes are and stores this history. This feature allows for version control, such that users can re-visit previous versions of data and code. This is particularly useful if a mistake has been made where a user has unknowingly overwritten or deleted information that would otherwise be irretrievable without GitHub having saved that information.

An even easier way to start using Github is for the archival of cleaned code and data, often accompanying preprinting, manuscript submission, or manuscript acceptance. Many users prefer to host a separate, cleaned repository that they make public when they complete a papr, while keeping

the original folders as either a private GitHub repository, or on another cloud storage service such as OneDrive, Dropbox, etc. One benefit of using GitHub for this service is that it can integrate with a website called Zenodo, a free, long-term data archiving service funded by CERN. After linking your GitHub account to Zenodo and turning on archiving, any time a release (Box 1) is made, a snapshot of the entire repository is archived in Zenodo with a versioned, citable DOI (see `### Making code citable` below for more information). DOI's for data and code are increasingly being required by journals for paper acceptance (e.g., Journal of Applied Ecology), and Zenodo provides a free alternative to other hosting services (such as Dryad).

Virtual lab notebook

contributors to this section: commits as a way to record daily progress issues as a way to keep track of short-term objectives/goals, and progress towards them

Classroom teaching / educational materials

contributors to this section: Cole Brookson

GitHub provides a large variety of uses for hosting teaching/educational materials. In fact, through taking advantage of the suite of GitHub features, the entire process of running a course, workshop, or even just a lecture, can all be done openly on GitHub. As a matter of gross simplification, organizing a course (for example) could be broken down into: 1) developing the material (i.e. slides, examples, relevant readings, labs, etc.), 2) hosting the course on some online platform for students to access, 3) delivering the content, and 4) dealing with student submissions and the subsequent grade returns. While of course there are other purpose-built platforms for this type of activity, few of them provide the usability at the price point GitHub does. First, developing your course material, from slides to labs and everything in between, can be done on GitHub, out in the open, where others can see, review and offer feedback on your process! Making presentations can be done through most major high-level programming languages such as R, with `RMarkdown`, Python, with `python-ppt`, and Julia, with `Remark.jl`. Since all these programs work via code bases, they can be version-controlled through git and GitHub. Once you've made all the content for your course, hosting a course website can be done through GitHub pages, and there are lots of templates available to borrow from. This way, not only can the course content be available to your enrolled students, but also to anyone interested in the course material. Since the course material can be easily housed on a GitHub pages website, it is then simple enough to deliver the content via that website, and/or a GitHub organization with template repositories for assignments etc. Student submissions are perhaps the least seamless component, but for assignments submitted as code files (i.e. `.R` & `.Rmd` as two of the most common) and/or `.pdf` files, GitHub has a new and far-from-perfect but still useful tool GitHub classroom where instructors can host private assignments, and even build custom autograding tests, that will autograde assignments!

The previous section is meant to highlight the myriad tools GitHub can provide to centralize the delivery of educational materials. While most instructors will likely choose to pick from this selection and end up having a mix of tools to deliver their content to students, it is still valuable to utilize some of these, if only for the reason that it can encourage students to even *begin* learning about version control through interacting with git/GitHub, however minimally, through the course. There are (as always) no "points" awarded for using ALL GitHub materials ALL the time, but if a central tenant of a given course or educational unit is to introduce or give students experience to version control and the tools that working professionals in the biological sciences use, then adopting a few of these tools can be a great way to do so.

Matthew D. Beckman, Mine Çetinkaya-Rundel, Nicholas J. Horton, Colin W. Rundel, Adam J. Sullivan & Maria Tackett (2021) Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses, Journal of Statistics and Data Science Education, 29:sup1, S132-S144, DOI: 10.1080/10691898.2020.1848485

GitHub in EcoEvo examples (Part 2)

Project management

Contributors to this section: Kaitlyn Gaynor, Rob Crystal-Ornelas

GitHub can be a powerful tool for team-based project management, allowing collaborators to share feedback, brainstorm ideas, and troubleshoot problems (Figure 2). The “Issues” feature of GitHub 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. This is because unlike emails and messages which can get lost as more new tasks arise, GitHub issues exist until they are intentionally closed by repository administrators. 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 as well as link issues togther when they are related to a shared project goal or milestone. GitHub is currently beta testing a similar project management feature called GitHub Projects. GitHub can bGitHub can also be integrated with other project management software like Slack or Zenhub.

Building website

Contributors to this section: Rob Crystal-Ornelas, Emma Hudgins
It is now common for many scientists to have personal, project, or lab websites (hereafter, personal websites) to share and promote their work.
There are many options for creating and hosting websites.
Some sites are built through a point-and-click user interface that requires no coding experience, but these services tend to have monthly or annual fees (e.g., Wix, Squarespace, Wordpress).
GitHub Pages allows users with a GitHub account to easily create a website, hosted by GitHub, from one of their many website templates [12].
It is also possible to fork any public website hosted on GitHub in order to use it as a template.
When creating a website with GitHub Pages, all content is stored in a GitHub repository, the content is written in markdown (e.g., https://github.com/SORTEE-Github-Hackathon/main-website), and a website is automatically rendered in HTML from the markdown documents (e.g., https://sortee-github-hackathon.github.io/main-website/).
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 [6].
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 [16]).
We emphasize that despite the many benefits of using GitHub pages (free hosting, templates, customization), this avenue for creating a website will be more time intensive than the out of the box platforms mentioned above and requires consideration of tradeoffs offered by website creation

services.
For more advance GitHub users, Jekyll and Hugo are both “static website generators”, which also include template libraries for websites that can be hosted freely via GitHub pages. Both of these tools require some additional learning because they are deployed via the computer’s terminal or command line, still they are a great resource for creating free, eye-catching websites.

Making code citable

Contributors to this section: Rob Crystal-Ornelas, Emma Hudgins, Dylan Gomes
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 [8,12]. Also, unlike the 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 [17]. Fortunately, both long-term repositories mentioned above (Zenodo and Figshare) have integrations with GitHub which facilitates archiving a snapshot of all repository content with the click of a button.

Linking one’s GitHub repository with Zenodo, etc. to achieve a DOI helps work become findable, gives proper attribution, and that can ensure long-term stability (Hampton et al. 2015). Thus, when researchers wish to include data and code with their publications, they ought to reference a DOI from a long-term storage site, rather than a URL from GitHub (which can change or be deleted). Additionally, referencing a DOI for data and code is preferable to submitting these as supplementary materials to the journal, as supplementary materials are more difficult to find and reuse (i.e. often not centralized and searchable in a database) and not necessarily permanent (as most journals offer no guarantee of long-term storage).

Many researchers believe that their code is not useful because their analysis is context-specific and not designed for re-use like software. However, there are many reasons to share data and code beyond re-use. Even if code is rough, it shows the exact steps taken to conduct an analysis, and therefore provides the most detailed look into how to reproduce a given analysis [18]. This is important in light of the reproducibility crisis [https://doi.org/10.1038/533452a] 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 our publications leaves future scientists with many fewer resources from which to understand the world.

The standard GitHub licensing options are best suited for software. If your code is intended only for your specific analysis, consider a Creative Commons License. The Choose a License website can offer further guidance. If you wish to allow anyone to re-use your code, consider a CC0 1.0 public domain dedication. If you wish to receive attribution for any reuse of your code, consider a CC BY 4.0 license, which requires attribution upon reuse. If you have build 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. If you do not wish to receive attribution and are open to private use, consider the MIT license.

Collaborative (code) editing

Contributors to this section: Kaitlyn Gaynor, Rob Crystal-Ornelas

From its inception, one of the primary uses of GitHub has been for collaborative coding. We acknowledge that there are important differences between the average software developer and ecology/evolution researcher using GitHub, and that not all GitHub collaboration features are optimal for research purposes. However, core features of git like forking and branching can allow for simultaneous coding on different versions of the same research project, and alternative versions can be easily discussed and resolved with GitHub. While a complete review of these features is beyond the scope of our paper, there are many free resources for learning how to use these collaborative features of GitHub [19]. (e.g. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/incorporating-changes-from-a-pull-request/merging-a-pull-request) It is often best to develop comfort with features like pull requests and merges on "practice" repositories with colleagues before integrating these tools fully into a collaborative workflow.

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 can also help advisors to identify errors early in the process, and inform further training and mentorship to fill gaps in skills.

Writing manuscript

contributors to this section:
Caveat that GitHub has been called out for not being so user-friendly for manuscript development (Ram 2013). But getting better? Tools that link with GitHub have been developed with synchronous writing in mind. HackMD provides a collaborative writing platform based on Markdown that integrates with GitHub.
We used this platform early on in the process of writing this manuscript to generate an outline.

Later, we wrote the manuscript by Manubot, which is one of the important collaborative manuscript platform uses Markdown for writing and GitHub for storing and tracking changes over time [20]

Recently, more manuscripts were written on GitHub via Manubot(see examples https://manubot.org/catalog/)

Peer-Review

Peer review of research software by 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 (Box 1). 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 issues (Box 1) can also be used to organize and discuss reviewer suggestions and to assign them to co-authors (See example here). 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 (Box 1).

GitHub in EcoEvo examples (Part 3)

Open science discussion

Contributors to this section: Freddy Hillemann, Allison Binley 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, using 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.

Github is a tool for managing and sharing components of any research project, that can help accelerate progress towards Open Science goals [21], from developing the analysis through to publication and ensuring reproducibility. 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. Collaborators 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 [**doi.org/10.1371/journal.pbio.3000763?**]. 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 that aids the direct communication with repository owners, as well as the Github Community forum for more general questions and sharing of expertise.

If the methods and analyses used are fully available during the peer-review process, they can be as critically evaluated as the rest of the manuscript. *link to peer review section?*. However, fear of being scooped feeds into cultural reluctance to do science openly due to worries of intellectual property. A solution would be to allow sharing a private link with a citable DOI for peer review, and update the DOI post-acceptance or post-publication. Updating DOIs is currently yet a missing feature of github, but implemented in other open-access repositories such as Zenodo, which was developed under the European OpenAIRE program and is being hosted by CERN. Alternatively, github users can to keep their data and code repository private until publication, and then make it public.

Project continuity

Contributors to this section: BPME, VF

The development of research software continues to be on the rise, and with that comes the need to consider the continuity of the research software. This is particularly relevant for software developed for relatively short-term research projects, such as projects developed by graduate students or postdoctoral fellows [22]. Often with these projects, once the research 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 hard drive, it becomes increasingly difficult to access the software and code for future uses.

When the project owner is finished with the project, or their contract expires, there generally should be a handover period of this software in order for the next cohort of researchers to reuse what was already developed [[22]][7]. 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 owners of the code access to the entire history of the project [11]. Additionally, GitHub allows for repositories and organizations to have designated Code Owners [23]; these code owners can change through time allowing for the transition of research software from one cohort of researchers to the next [24].

Within EEB projects, tasks are often divided among contributors taking various roles (see CRediT taxonomy). 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 non-code writers to be engaged in aspects of repository design in a way that improves institutional memory and facilitates project continuity. Non-code writers 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.

Asynchronous working

Contributors to this section: Ali, Allison Binley

Recently, asynchronous communication tools were boosted the team works. Github served as an excellent environment for asynchronous communication and collaboration for especially remote team projects. Researchers can easily collaborate without being in same place and time.

One of the most useful aspects of Github is its propensity to facilitate remote and asynchronous collaboration. Researchers can seemlessly access and contribute to data and code regardless of disparities in schedules or location. This is particularly important given the increase in remote work in recent years, but the benefits can also extend far beyond the “work from home” model. Improving remote collaboration can encourage the exchange of ideas among researchers at different institutions and in different countries, which can serve to improve the quality of the research itself. Researchers can work directly with experts from all over the world, who have access to the same data and code as they do.

However, Github has something to offer even for team members on a project that work in the same office. Researchers can easily stay abreast of progress made by other collaborators without the need for meetings or emails. Collaborative project work can also be clearly split between team members, giving them the flexibility to contribute when it best fits their schedule. The version control features also allow users to make progress and changes without worrying about irreparably writing over someone else’s work.

could link this back to the “Project Management” section or even “Collaborative (code) editing”.It actually seems to me like this entire section could potentially be combined with “Collaborative (code) editing”

GitHub organizations

Contributors to this section: Katherine Hébert, Cole Brookson

Whether experiments are done in a wetlab, data are gathered in a field site, or analyses are run in a shared office, even conceptually distinct projects are often carried out in a common physical space. GitHub Organisations 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

organisation, or a large project involving several teams. Organisations 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 organisation, 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, students or 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 upgraded their computers). 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). Of course, the utility of this tool goes beyond laboratories - they are useful to structure the organisation, presentation, and outcomes of working groups such as the hackathon which inspired this paper (SORTEE-Github-Hackathon) by keeping track of all materials as ideas develop and take shape in one virtual space. Organisations 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) or the University of Edinburgh's Coding Club (Coding Club), which may be updated by an ever-evolving group of contributors over time.

Utilizing 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's 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 repos in the organization with a select group having push access to some repos but not others. This can manifest in a group working on some common dataset(s) (e.g. some genetic data) 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 repos. The organization structure also allows for easy tracking of issues, projects, and discussions related to the research group, and provides PIs/group leads an easy birds-eye view of the progress going on across multiple projects.

As well, organizations provide a convenient location for students to archive the code for their projects, for use/reference by future students in the research group, thus providing a type of knowledge communication that may not exist otherwise. Indeed, providing new students with access to the organization and ideally a template repository for lab projects can soften the burden on those new to the software, in that it provides them with examples to work off of, and an online location to ask for help from their labmates and/or advisors through tools like projects, discussions, and issues.

Additional uses for GitHub in EcoEvo research

Contributors to this section: RCO, Ali

There are many more ways that EEB researchers can use GitHub for accelerating research collaborations, and we briefly highlight several here.
First, there are increasing calls for ecological data to be more Findable, Accessible, Interoperable, and Reusable (FAIR) [25,26].
A key component of data reusability is standardizing the ways (e.g., variable names, file formats) that research data are archived in long-term repositories.
Recently, community-led data standardization efforts are taking place on GitHub [17], where documents and templates can be version controlled and commented on by the user community e.g., ESS-DIVE's GitHub Community Space.

Ecologists who write code often use the R programming language, and the rOpenSci community has a well-established software peer review process that involves both rOpenSci's staff software engineers and the broader R user community.
Their software review GitHub repository provides instructions for submitting an R package for review as well as guidelines for code reviewers.
rOpenSci's efforts have resulted in many well-used R packages for ecology research including rfishbase [27] and taxize [28].

Lastly, GitHub gists let users create and share snippets of code, notes, and files quickly.
Rather than create an entire GitHub repository for saving a small code chunk you want to use in a presentation or share with a colleague, GitHub gists provide a lightweight way to write, save, and share code.
Gists are associated with your Github account and can be public or private.
Though gists lack all the features embedded in a GitHub repository, gists can still be forked, starred, downloaded, and easily added into a website or blog post.

Discussion

General pargrah on what GitHub can enable in EcoEvo

Contributors to this section: Rob, Brandon

There have been many calls for researchers outside of the software development community to join the 73 million GitHub users for their collaborative research.
This call comes in light of the continual shift toward a more open-science framework across several fields, and as computational and data requirements increase in several fields. Until now, resources and practical guidance specifically focused on using GitHub within the EEB community have been dispersed in blog posts and video tutorials. We felt these resources have been extremely useful for us to learn to use Github in our own work, and we felt that a collation of the main ideas into one medium, while adding on our personal perspectives, would be of use in the EEB community.

In this paper, we describe 13 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 several hours of practice (e.g., 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 for creating a DOI for the repository. These actions are generally covered in any introduction to Github tutorial and take little overhead to make it work. On the other hand, some other examples we describe may require a greater time commitment, but have the potential to make EEB research even more open, accessible,

and collaborative than ever before. For example, 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). Additionally, collaboratively writing a manuscript using Github, such as what we have done with this, will involve a learning curve for folks less familiar with the intricacies of Github, and also require overheard 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.

When to look to other platforms for collaboration

Contributors to this section: Rob

Though we see GitHub as a useful tool for collaboration in EEB, we describe 2 use cases where, to our knowledge, GitHub's features still fall short of the type of highly collaborative work emblamatic of EEB research.

First, because of the underlying version control system which tracks "pushed" changes through "commits", 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 now websites outside of the GitHub ecosystem that are built on top of the GitHub architecture that allow real-time collaboartive editing (e.g., hackMD.

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.

Despite the extra layer of coordination required to use GitHub to write a paper, many journals are changing their manuscript formatting requirements to allow for manuscript format output from GitHub [29].

Second, we looked to other software when working on figures and tables. 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 toward 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

Contributors to this section: Saeed, Vivienne

Though GitHub has been available as a platform for more than a decade at time of writing, its uptake among EEB researchers, especially as a tool for collaboration, has been slow. Some research groups choose to take advantage of alternative platforms with similar capabilities, such as GitLab or Bitbucket, but many have yet to integrate these types of project management software into their lab ecosystems at all.

Some attribute this hesitation to the steep learning curve combined with limited instruction available through traditional university courses. When use of 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. The default public nature of GitHub usage can add additional pressure to students learning to use the platform.

While many quantitative ecologists and evolutionary biologists 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, and 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 have obscured the utility of GitHub for tasks other than traditional data analysis and software development, or how GitHub can facilitate the integration of code with non-coding aspects of projects through the practice of repository design. Additional hesitancy may come from general reluctance to share data and code publicly, or technical and logistical issues related to storage of large data files and lack of integration with other research platforms.

Our own limitations in writing this paper

Contributors to this section: Ali

Our own limitations since we are mostly writing from the EcoEvo perspective/ additional github limitation Reliance on R since we are generally in EcoEvo Discussion of free vs. paid plans. When projects get highly collaborative may have to add / pay for accounts. At this point, little difference between paid and free.

File size limitations make it hard to sync entire folders with the cloud, especially if your repo is private. One trick to increase the size limit associated with a repository is to attach additional large files to the release once the repository is finalized and ready for re-use.

While we have written this manuscript collaboratively via GitHub and GitHub Actions, there was a substantial learning curve, and we are all early career researchers who are highly motivated to learn these skills. This approach is likely less practical when collaborating with people at later career stages and a greater range of computational skills. Real-time collaboration via GitHub is also not possible without relying on additional tools such as hackmd, so there is a longer delay to receive feedback from other collaborators, and a greater likelihood of conflicts.

Lack of GitHub help documents for non-English researchers in ecology and evolution leads them to miss the opportunity to fully understand the importance version control as well as the other benefits of GitHub.

Using GitHub is a good start, but lots of practices to make repo more user friendly

Contributors to this section: Ali, Emma

10 Tips for getting started in GitHub
1. **Check for an existing solution to your problem.** The Github Help webpage 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 common Github issues.
2- **Consider taking free courses** such as those from Software Carpentries and sharing these courses with your lab members or colleagues.
3. **Take the advantage of Github as an asynchronous working tool for team-based projects** See the repository for this paper as an example of a collaborative manuscript that includes discussions, issues, and a website via GitHub.

10 Tips for getting started in GitHub
4. **The GitHub Learning Lab** 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** 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, statistics course and [6].
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 your curricula**. Some gradute 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 commiting with GUI (Graphical user interface) tools** like GitHub Desktop, git-gui, RStudio, Visual Studio Code, Atom, GitKraken tools instead 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 to disentangle and manage multiple notifications from distinct GitHub projects.

Conclusion

We provide 13 practical ways that Ecologists and Evolutionary Biologists can incorporate GitHub into their workflows.
GitHub is still an emerging platform for many working in EEB, and so we include definitions (Box 1) and key user groups (Box 2) that can help researchers prioritize which GitHub skills to learn first.
Some of the ways we outline for using GitHub 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 immediate collaboration, adoption of GitHub 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 applications of GitHub within EEB 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 2020.

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.

References

1. **How do scientists develop and use scientific software?**
Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, Greg Wilson
2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (2009-05) https://doi.org/bw966x
DOI: 10.1109/secse.2009.5069155

2. **Challenge to scientists: does your ten-year-old code still run?**
Jeffrey M Perkel
Nature (2020-08-24) https://doi.org/gg89cr
DOI: 10.1038/d41586-020-02462-7 · PMID: 32839567

3. **A survey of the practice of computational science**
Prakash Prabhu, Yun Zhang, Soumyadeep Ghosh, David I August, Jialu Huang, Stephen Beard, Hanjun Kim, Taewook Oh, Thomas B Jablin, Nick P Johnson, … David Walker
State of the Practice Reports on - SC '11 (2011) https://doi.org/bdpsrp
DOI: 10.1145/2063348.2063374

4. **Best Practices for Scientific Computing**
Greg Wilson, DA Aruliah, CTitus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, … Paul Wilson
PLoS Biology (2014-01-07) https://doi.org/qtt
DOI: 10.1371/journal.pbio.1001745 · PMID: 24415924 · PMCID: PMC3886731

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

6. **Excuse Me, Do You Have a Moment to Talk About Version Control?**
Jennifer Bryan
The American Statistician (2018-01-02) https://doi.org/gdhzdp
DOI: 10.1080/00031305.2017.1399928

7. **Git can facilitate greater reproducibility and increased transparency in science**
Karthik Ram
Source Code for Biology and Medicine (2013-02-28) https://doi.org/krv
DOI: 10.1186/1751-0473-8-7 · PMID: 23448176 · PMCID: PMC3639880

8. **Ten Simple Rules for Taking Advantage of Git and GitHub**
Yasset Perez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J Eglen, Daniel S Katz, … Juan Antonio Vizcaíno
PLOS Computational Biology (2016-07-14) https://doi.org/gbrb39
DOI: 10.1371/journal.pcbi.1004947 · PMID: 27415786 · PMCID: PMC4945047

9. **A Quick Introduction to Version Control with Git and GitHub**
John D Blischak, Emily R Davenport, Greg Wilson
PLOS Computational Biology (2016-01-19) https://doi.org/gbqsnf
DOI: 10.1371/journal.pcbi.1004668 · PMID: 26785377 · PMCID: PMC4718703

10. **Let's Git started | Happy Git and GitHub for the useR**
Jenny Bryan Hester the STAT 545 TAs, Jim

https://happygitwithr.com/

11. **Our path to better science in less time using open data science tools**
Julia SStewart Lowndes, Benjamin D Best, Courtney Scarborough, Jamie C Afflerbach, Melanie R Frazier, Casey C O'Hara, Ning Jiang, Benjamin S Halpern
Nature Ecology & Evolution (2017-05-23) https://doi.org/gc4jb3
DOI: 10.1038/s41559-017-0160 · PMID: 28812630

12. **Democratic databases: science on GitHub**
Jeffrey Perkel
Nature (2016-10-03) https://doi.org/gdz6dq
DOI: 10.1038/538127a · PMID: 27708327

13. **Social network of software development at GitHub**
William Leibzon
2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (2016-08) https://doi.org/gph5qt
DOI: 10.1109/asonam.2016.7752419

14. **A Beginner's Guide to Conducting Reproducible Research**
Jesse M Alston, Jessica A Rick
The Bulletin of the Ecological Society of America (2021-01-15) https://doi.org/gk5v4p
DOI: 10.1002/bes2.1801

15. **Foundational Practices of Research Data Management**
Kristin Briney, Heather Coates, Abigail Goben
Research Ideas and Outcomes (2020-07-27) https://doi.org/ghssbk
DOI: 10.3897/rio.6.e56508

16. **Building tools with GitHub: customize your workflow**
Chris Dawson
O'Reilly (2016)
ISBN: 9781491933503

17. **A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats**
Robert Crystal-Ornelas, Charuleka Varadharajan, Ben Bond-Lamberty, Kristin Boye, Madison Burrus, Shreyas Cholia, Michael Crow, Joan Damerow, Ranjeet Devarakonda, Kim S Ely, ... Deborah A Agarwal
Earth and Space Science (2021-08) https://doi.org/gmbs9c
DOI: 10.1029/2021ea001797

18. **Elevating The Status of Code in Ecology**
KAS Mislan, Jeffrey M Heer, Ethan P White
Trends in Ecology & Evolution (2016-01) https://doi.org/gg43mk
DOI: 10.1016/j.tree.2015.11.006 · PMID: 26704455

19. **Barely sufficient practices in scientific computing**
Graham Lee, Sebastian Bacon, Ian Bush, Laura Fortunato, David Gavaghan, Thibault Lestang, Caroline Morton, Martin Robinson, Philippe Rocca-Serra, Susanna-Assunta Sansone, Helena Webb
Patterns (2021-02) https://doi.org/gjpcb6
DOI: 10.1016/j.patter.2021.100206 · PMID: 33659915 · PMCID: PMC7892476

20. **Open collaborative writing with Manubot**

Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
PLOS Computational Biology (2019-06-24) https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007128
DOI: 10.1371/journal.pcbi.1007128

21. **The Open Knowledge Foundation: Open Data Means Better Science**
Jennifer C Molloy
PLoS Biology (2011-12-06) https://doi.org/g3b
DOI: 10.1371/journal.pbio.1001195 · PMID: 22162946 · PMCID: PMC3232214

22. **Sustainable Research Software Hand-Over**
J Fehr, C Himpe, S Rave, J Saak
Journal of Open Research Software (2021) https://doi.org/g4n4
DOI: 10.5334/jors.307

23. **About code owners**
GitHub Docs
https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners

24. **The Tao of open science for ecology**
Stephanie E Hampton, Sean S Anderson, Sarah C Bagby, Corinna Gries, Xueying Han, Edmund M Hart, Matthew B Jones, WChristopher Lenhardt, Andrew MacDonald, William K Michener, … Naupaka Zimmerman
Ecosphere (2015-07) https://doi.org/gdj5w6
DOI: 10.1890/es14-00402.1

25. **Ecological Data Should Not Be So Hard to Find and Reuse**
Timothée Poisot, Anne Bruneau, Andrew Gonzalez, Dominique Gravel, Pedro Peres-Neto
Trends in Ecology & Evolution (2019-06) https://doi.org/gg43mw
DOI: 10.1016/j.tree.2019.04.005 · PMID: 31056219

26. **The FAIR Guiding Principles for scientific data management and stewardship**
Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, … Barend Mons
Scientific Data (2016-03-15) https://doi.org/bdd4
DOI: 10.1038/sdata.2016.18 · PMID: 26978244 · PMCID: PMC4792175

27. **rfishbase: exploring, manipulating and visualizing FishBase data from R**
C Boettiger, DT Lang, PC Wainwright
Journal of Fish Biology (2012-11) https://doi.org/gh5x27
DOI: 10.1111/j.1095-8649.2012.03464.x · PMID: 23130696

28. **taxize: taxonomic search and retrieval in R**
Scott A Chamberlain, Eduard Szöcs
F1000Research (2013-10-28) https://doi.org/ggdskx
DOI: 10.12688/f1000research.2-191.v2 · PMID: 24555091 · PMCID: PMC3901538

29. **Synchronized editing: the future of collaborative writing**
Jeffrey M Perkel
Nature (2020-03-31) https://doi.org/ggqk8s
DOI: 10.1038/d41586-020-00916-6 · PMID: 32235940