**Tailoring GitHub for Ecology**

**Abstract**

Ecology is transitioning from an anecdotal science into a data driven science. Biologists are finding that they are needing skills to manage, analyze, and store data effectively. Data, such as code and text, require thoughtful management practices to keep these files organized and available to other researchers. We developed a modern structure using GitHub so that files may be available to all members of the team, other collaborators, and publishers. We do this by 1) evaluating our previous repository structure and workflow; 2) creating a new and consistent structure and workflow among all project repositories; 3) and establishing and maintaining a file naming convention which encompasses any file that could be in a repository. This repository structure takes into account the need to manage “living data” and the necessity for ecological efforts to be transparent.

**Introduction**

Traditional field ecology is currently experiencing a data revolution. Advancing technology drives the need to collect and analyze regularly updated data for natural resource efforts, (i.e., “living data”, Yenni et al., 2018). These developments require scientists to work effectively with these “living data” (Box 1), despite they are rarely trained to do so (Lowndes et al., 2017). A recent NSF (National Science Foundation) survey (Lowndes et al., 2017) demonstrated that of the 704 scientists who participated in the survey, “data skills” (e.g. multi-step workflows, ability to store, share and publish data) was identified as the largest unmet need (Barone, Williams, and Micklos, 2017; Lowndes et al., 2017). Ecologists are also increasingly needing to write code as part of their field, lab, and modeling research (Mislan et al., 2016). The necessity to write efficient code, store the code, and increase analysis reproducibility is frequently mandated by publishers (Alarid‑Escudero et al., 2019).

The LCR (Lone Cabbage Reef) project is a large ecological effort to restore an oyster reef to historic elevation levels so that it may endure sea level rise and river discharge. Our project generates data from multiple sources including observations of oyster populations and measurements by field biologists and continuous autonomous water quality data via sensors. These data are updated at different frequencies and require specific attention to be processed (Moreno et al., 2020 hopefully). However, once they are processed there is a need to store these data so that they made be used among project team members and collaborators. We use GitHub version control software to keep track of regularly updated data, and to keep the multiple working projects using these data, organized. This paper describes our approach to address concerns regarding standardizing naming conventions, GitHub repository structures, and data availability to differing LCR project repositories to increase reproducibility and transparency.

#### Box. 1 Terminology

GitHub- an online version control software, free, and accessible to anyone with Internet ([www.github.com](http://www.github.com))

living data- data that are being analyzed while also still being collected (Yenni et al., 2018)

version control- a system which allows the users to track iterative changes to code and text (Blischak et al., 2016)

project repository- term used to identify one type of analysis that is conducted on an LCR project dataset

README.md- a markdown file which includes information about folder and files contained in the repository

user- any person using GitHub

team member- specifically referring to an LCR project collaborator

admin- specific members of the LCR project tasked to monitor project repository’s pull requests in GitHub

pull request- a process in GitHub to submit changes and a message containing a description of the change, in order to track the version of the change, an additionally an admin can review and accept the changes

`master` branch- the first branch created with a GitHub repository, in the contexts of the LCR project, it is the most up to date branch with the most user limitations for pull requests, this branch requires admin reviews

branches- essentially copies of the GitHub repository `master` branch, in the context of the LCR project each collaborator specifically make edits to their own branch prior to a pull request to the `master` branch

merge conflicts- when branches have competing commits during a pull request, needing to be resolved by an admin (<https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-merge-conflicts>)

dark data- data which is not easily found by potential users (Heidorn, 2008)

commit message- a written text of why a pull request is being submitted

#### End of Box 1

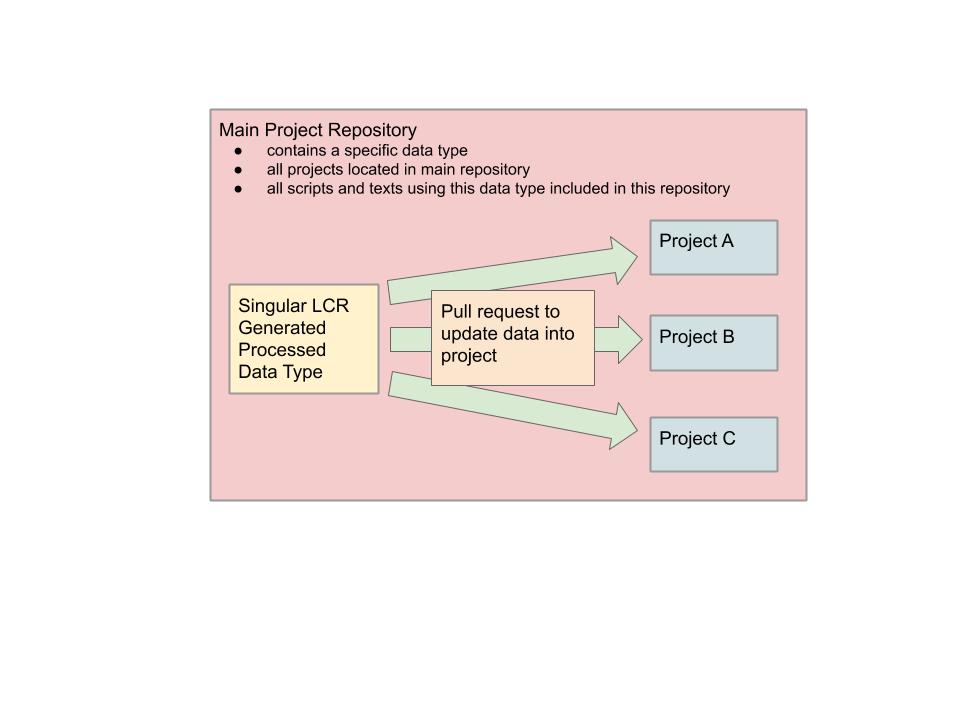
**GitHub and version and control**

Scripts and text can be difficult to track changes in collaborative efforts (Blischak et al., 2016). To keep track of these changes version control systems are generally implemented. Version control allows the user to track changes by requiring the user to write comments for each change. The version control software will then create a unique version identifier to track each change and allow the user to revert back to those changes if needed (Noble, 2009). This is especially useful when a user decides to share their code or text with other collaborators. When the user receives new comments from the code/text they have shared, they are able to see what has been changed in relation to the original document. The user will then be able to implement those changes through a version control system.

In the LCR project, code collaboration is a common practice. Code writing collaboration in our project ultimately ended in several files of the same script but each file was a slight iteration of each other (e.g, rscript\_1.R, rscript\_2.R) . It became confusing which script was the most up to date, and which script should be used for analysis. Using version control stores each iteration of every file, keeping repositories free of duplicate code files. We utilize GitHub to keep a record of which files have been changed, who has changed them, and why they were changed. Using a version control software has saved us timed in trying to determine when and why certain script changes were implemented.

**Challenges working in one repository**

As the LCR project started to generate a consistent stream of data, it became apparent that its GitHub structure was becoming increasing more difficult to maintain and manage. One of the main complaints was that it was difficult to find files and locate their data source. Collaborators working in the main GitHub repository were not always following repository guidelines, however the guidelines at the time did not address many of our newfound needs such as how to account for multiple working projects using the same data. Without having proper guidelines, the main GitHub repository quickly began to grow and expand (Figure 1). Our main GitHub repository started to grow projects inside of itself, leading to a confusion of what was in the repository was which data and scripts were used for each of the different projects. We soon came to realize that the GitHub repository structure we had employed was not effective in keeping our files or projects organized. We want to note that this described repository structure could work for ecological efforts, but for our project it became too cluttered and unclear to continue using this structure.

Figure 1- Visualization of our main project repository structure and various projects in the same repository

**Creating a new GitHub repository structure and workflow**

The main goal of our new GitHub repository structure is to keep different projects separate but to have one common data source in which they may be able to update their data from. We extracted the different projects inside our main GitHub repository and created individual project repositories. Each project repository follows the same guidelines for folder structure (https://github.com/LCRoysterproject/repo\_structure). These new project repositories also include descriptions in their README.md about the folders and files inside of them. These README.md’s are essential in maintaining transparency of what each script does and what their outputs are. Our project team members are required to make updates to README.md’s as they create new files for scripts and text.

These individualized project repositories are self-sustaining and only team members actively working on that project have access to the repository. These repositories are self-sustaining in the sense that they are independent from one another, and their scripts are not influenced by other project repositories’ scripts. Because some of these project repositories need to access the most up to date LCR project generated data (e.g., water quality, oyster measurements, oyster counts) it was imperative that our workflow included a way that the project team member could access these data. All LCR generated data are processed and then stored in a master data repository. This data repository may contain relevant data which could be used among different project repositories (Figure 2). This master data repository also contains commonly used scripts and text. Every LCR team member has access to this master data repository. In the event in which a team member needs to update the data for their analysis, they are able to do so without limitations or approval. These data are also protected from being wrongfully edited or deleted by GitHub branch permissions. Only LCR project admins are allowed to make updates or changes to the master data repository, which adds an extra layer of security to this repository.

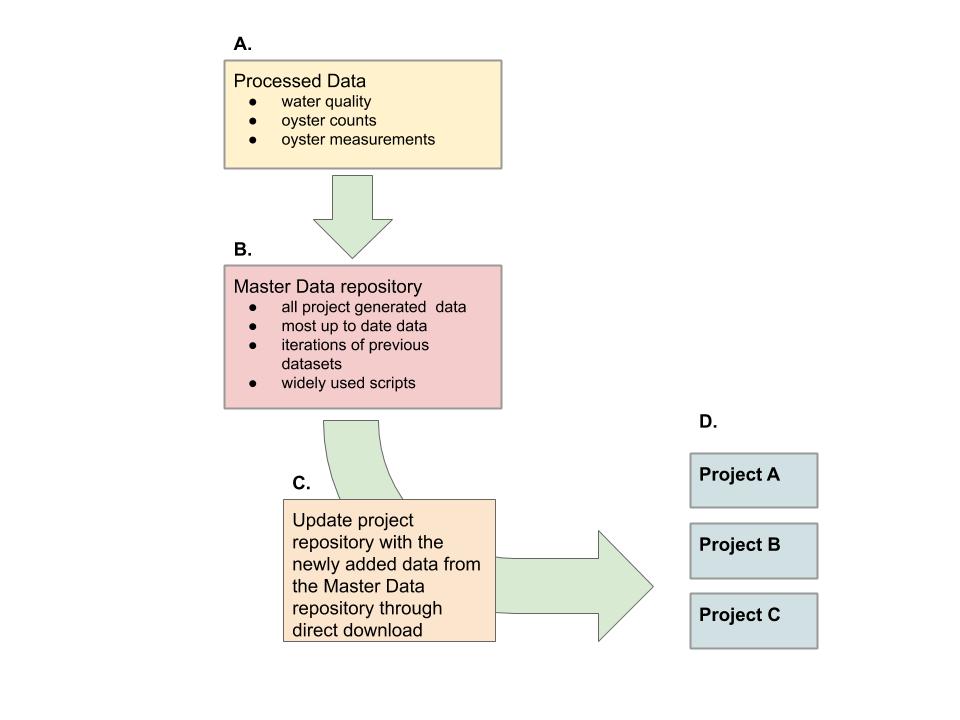


Figure 2- New LCR project workflow which describes how project repositories update to work with newly added data. A. Data are processed and cleaned via MySQL or scripts, B. Data are updated to the master data repository, reviewed, and approved by LCR project admins, C. Project repositories update their repository data by downloading the data directly from the master data repository, D) Project repositories conducting individual analysis on LCR datasets

**GitHub permissions and branch workflow**

GitHub has settings which can limit who can edit or modify a repository’s branches (Yasset Perez-Riverol et al., 2016). For each LCR project repository, which has collaborators, we establish a protected `master` branch, and open collaborator branches. Collaborators are able to edit and modify their own branches however they please, but they are not able to update or modify the `master` branch unless approved by a project admin via a pull request (Box 1). Project admins are expected to review a pull request rigorously and work with the collaborator if there are any discrepancies in the pull request. Using a system that checks the work of collaborators has helped us reduce errors in code, text, and data and can be implemented across many ecological efforts.

Furthermore, GitHub has the functionality to make repositories public or private. Whether a repository is public or private is ultimately up to the admins of that project repository. Public repositories are open and searchable to the public. Private repositories are only initially viewable to the creator of the repository. Additionally, in the LCR project we also limit the users who have access to any given repository. All users have access to the master data repository, however they do not have access to other collaborator repositories, unless an LCR admin grants them access. We allow some repositories to be public and protected, and actively worked on repositories to be private. GitHub allows project managers to change the status of any repository, to private or public, at any time. These types of repository functionalities can allow many ecologists to actively work on their research while protecting their data, analyses, and findings.

**Naming conventions for repository, files, and folders**

Proper file naming conventions help users understand the contents of the file without having to click on it. For scripts, naming conventions exists in which if a script file creates a function or a certain output the output file should also be named the same (<https://style.tidyverse.org/package-files.html#names-1>). In the LCR project we created a set of consistent set of guidelines for filenames (Table 1). The overall guidance to naming files is to keep the cases consistent, in our case lowercase, and to keep the structure of naming the file the same. We use an overall naming guidance of study, location, and project summary, in that order, to name files. These guidelines help each collaborator really think about (A why they are creating a new file; and (B what does this file ultimately intend to do. These guidelines have allowed the LCR project to also increase its transparency among other ecologists. We hope that these file naming guidelines it will help other ecologists in pursuit of standardizing their filenames and increase their file transparency.

|  |  |  |  |
| --- | --- | --- | --- |
| File Type | Naming Convention | Example | Definition |
| Project Repository | study\_location\_projectsummary | bird\_bb\_monitoring | Big Bend camera and survey bird monitoring project |
| Scripts | lowercase, no uppercase (camelcase) nor all caps, all names with separate words need to include a underscore ( \_ ) and no spaces, no dates in the names unless it helps with the descriptions of the conten script file names should be descriptive and concise. Scripts that have a single output should be named in a similar fashion to its filetype output. | discharge\_1941\_2018\_quantile.R | R script which reports quantiles from river discharge from 1941 to 2018 |
| Figures | study\_location\_type\_summary.filetype | oys\_lco8a\_map\_transect.tiff | oyster transect on reef element LCO8A map in a tiff image |
| Tables | study\_location\_summary.filetype | wq\_lcr\_inshore\_vs\_offshore.csv | LCR water quality inshore and offshore comparison |
| Data | every dataset file is required to be in lowercase, no uppercase (camelcase) nor all caps, all names with separate words need to include a underscore ( \_ ) with **no spaces**, no dates in the names unless it helps with the descriptions of the content | discharge\_1941\_2018.csv | River discharge data from 1941 to 2018 in a text file |

Table 1- Table of naming conventions for file types, example, and description of the example

**Transparency**

GitHub has options to increase transparency for an ecological project. Hosting a public project repository on Github can improve the probability of researchers and the public to find the repository, and possibly collaborate. Commit messages, through a pull request, are easily seen and located in GitHub, and allow for collaborators to understand any change submitted to the repository. Transparent repositories are unlikely to be “scooped” by another researcher that can claim the data and the analysis is theirs, through the continuous stream of commit messages leading to the final product. An additional benefit to a transparent repository is that many eyes will be available to evaluate code and text, which can increase the time is takes to debug an issue. Generally, most scientists with related research interested are more willing to collaborate original scientist than compete with them (Prlic and Procter, 2012).

**Conclusion**

Ecology is an ever-growing science requiring a constant need to keep up to date with the latest technological advancements in field data collection. With that, it is necessary to create a cautious and manageable plan to store and maintain ecological “living data”. These data provide unique challenges in management and reproducibility. Using a well-established GitHub repository for “living data” leads to effective and easy data sharing (White et al., 2013), which increases the transparency of the effort leading to a greater impact (Piwowar, Day and Fridsma, 2007).

We employed a repository structure workflow on the recommendations of the University of Florida Academic Research Consulting & Services (ARCS). The repository structure we developed for the LCR project solves many of the challenges we were facing while working with our own generated “living data” and multiple working projects. Separating the multiple working projects into their own repositories keeps the projects simple and organized. There is also little confusion on what is in the repository specifically since every repository is required to include a README.md describing every folder and file. A README.md file also shapes the first impression of a repository and increases the searchability of repositories in GitHub (Azriadi Prana et al., 2019), which may lead to greater transparency and collaborative efforts (Jones, 2013).

Advantages of our approach include that GitHub and free and accessible to anyone with internet. There are also many training programs which can teach a user how to utilize GitHub efficiently for their project (e.g., <https://guides.github.com/activities/hello-world/>, lab.github.com). Another advantage in using GitHub is that if a repository is accidently deleted, a user has 90 days to retrieve the repository. A benefit in using our described approach is that it can be applied to any ecological effort which has a consistent stream of data by allowing a master data repository to be accessible to all team members while still protecting the repository from adverse or unintentional changes. Additionally, this type of workflow can easily be maintained by a small group of ecologists with basic GitHub workflow training (Yenni et al., 2018).

Some disadvantages to our approach include that ecological projects, conducted in smaller teams, normally do not have the funding to hire full-time data manager, (Hampton et al., 2013), making it a timely effort to curate an efficient data management workflow, initially. Data management planning is also typically and generally underutilized and underappreciated in ecological project designs (Michener and Jones, 2012). It is up to the project team member to manage the data, take the initiative, and adhere their GitHub data management workflow. It is also important to note that even though GitHub has been discussed in detailed as an effective way of project organization and transparency, it might not be a one stop solution for reproducibility in science (Ram, 2013).

The approach we have described in this paper is meant to be a guideline for ecological efforts who desire to make their project organized through concise workflows, standardized naming conventions, and well documented README.md’s. Our hope is that this paper can serve as a mechanism for designing version control software, such as GitHub, to meet the needs of an ecological project with a continuous stream of data and multiple working analyses and projects. Increasing transparency through managing a well-documented repository, through README.md’s, may lead to useful future collaborations. The investment in creating a thorough data management workflow in GitHub will help decrease the time is takes to effectively reproduce analysis by reducing the time it takes to locate files and their outputs, which will allow ecologists more time to analyze their data and less time trying to manage it.