**Tailoring GitHub for ecology**

**Introduction**

Traditional field ecology is currently experiencing a data revolution. Environmental crises have also propagated ecologists to a rapid shift into “big science” ecology (Hampton et al., 2013). Currently, there is a lack of training in basic data management, curation, and workflow design among scientists working with “big science” data (Lowndes et al., 2017, Wilson et al., 2016). 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). It is also becoming more apparent that ecologists need to write code as part of their field, lab, and modeling research (Mislan et al., 2016). The need to write efficient code, store the code, and increase project analysis reproducibility is becoming more of a requirement for publications (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

big science- originally termed by Alvin Wienberg in 1961, as a sizeable investment, involving international and collaborative efforts among scientists (Wienberg, 1961)

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 being conducted on a LCR project dataset

README.md- a markdown file which includes information about folder and files in a 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 GitHub repository’s pull requests

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

#### End of Box 1

**GitHub and version and control**

Scripts and text can make it 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 recording messages 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, the user will be able to implement those changes through a version control system.

In the LCR project, code writing 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.

**GitHub permissions and branch workflow**

**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. 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.

**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 repository follows the same guidelines for folder 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 change and update their 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. 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 restrictions. 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.

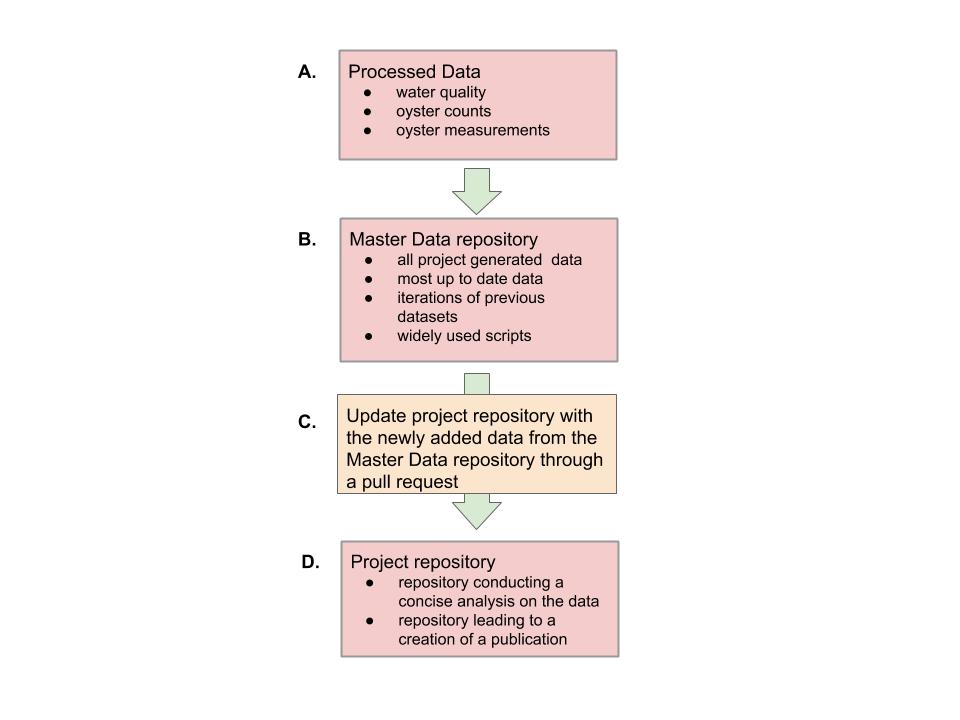


Figure 1- New LCR project workflow which describes how project repositories update to work with newly added data. A. Data is processed and cleaned, B. Data is updated to the master data repository, tentatively approved by project admins, C. Team prohect

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

Proper file naming conventions helps the user understand the contents inside the file. For coding scripts naming conventions exists if the file creates a function or a certain output the file should also be named that way (<https://style.tidyverse.org/package-files.html#names-1>). For our project we created a set of consistent set of guidelines to name files.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

Figure 2- Table of naming conventions

**Discussion**