Information and data architecture in environmental sciences: Folder structure, metadata, and project management through GitHub

Ensuring project reproducibility extends beyond just the tools you use; it encompasses a set of principles that should be applied from the very inception of your project. Often, our focus is on the hypothesis, experimental procedures, or manuscript writing. However, consider this: for all the involved tasks, from project creation to management and follow up steps, a computer is essential, whether it's for data storage, analysis, or writing a manuscript. Throughout this course, my aim is to help you integrate a higher level of planning into your workflow and guide you in making informed decisions about how, why, and where you store your project data.

Think of your project as a dynamic and ever-evolving entity. It will undergo continuous changes, and you might find yourself working on it for an extended period. For instance, I worked on my MSc thesis for 8 months, but later, when I began my PhD, I had to return to that project a year later to prepare a manuscript for peer review. Consider it a given that unexpected opportunities can arise during the journey, and you never know where they might lead you. And, remember, your past self can be your worst enemy. To make things easier for your future self, it's crucial to be prepared.

The initial step is to create a dedicated folder where all project-related files will reside. It's wise to anticipate the types of files you'll need. Ask yourself some basic questions (for instance, in the context of a BSc thesis):

* What data files will I generate or collect?
* What types of analyses or experiments will I perform?
* What documents will I write (e.g., drafts, reports, presentations)?
* Will there be code or scripts involved in data analysis?
* Are there any reference materials or literature to keep track of?

By addressing these questions proactively, you can establish an organized and robust foundation for your project, ensuring it remains manageable and reproducible, even as it evolves over time (Figueiredo L., et. al. 2022). When it comes to organization, it's essential to recognize that you won't be working alone. More often than not, you'll be part of a team, research group, or working under the guidance of your PI, all of whom will require access to project files.

Here are some key principles to consider:

1. **Transparent Communication**: It is absolutely essential to communicate your organization system clearly with your collaborators. Whether you're working within a research team, group, or under the guidance of your PI, everyone should have a thorough understanding of the naming conventions, folder structures, and overall file organization that you've put in place.
2. **Adopt a Consistent System**: While personal organizational systems may make sense to you, it's crucial to adopt a standardized system that is easily comprehensible, logically structured, and accessible to everyone involved. This minimizes confusion and ensures that all team members can efficiently access and work with project files.
3. **Documentation as a Guide**: Alongside your file and folder structure, provide comprehensive documentation or guidelines explaining how the system functions. Elaborate on the naming conventions, folder hierarchy, and the reasoning behind your choices. This documentation serves as a valuable reference for all collaborators.
4. **Openness to Feedback and Adaptation**: Be receptive to feedback from your collaborators. They may offer suggestions for enhancing the organizational system or encounter challenges that you hadn't foreseen. Be flexible and willing to adjust the system as needed to accommodate the team's requirements.

**Folder directories**

Try to think of a folder as babushka; first you have higher directories and as you move in it gets smaller (in size) and more specific. First folder can be named by the title of your thesis or your project. Then as you go to lower directories, think hierarchically. And always ask yourself; which is the goal of this folder and which organization system will serve better? As you go into the directory you need separate folders for different documents, i.e. for raw data, photos (in case you are running an experiment), script (for data analyses and cleaning) and for exports (results).

A screenshot of a computer

Description automatically generated

**Figure 1**: Example of a folder directory following the aforementioned principles.

**Folder and file onomatology**

Now that we've established the structural framework, let's delve into the specifics of how files and folders should be named. Suppose you're working on your thesis, which includes raw data and manuscript documents. Your initial file names are **data.csv, manuscript2023-final.docx**, and **clean-data.csv**. It is recommended to employ a naming convention that is both machine-readable and understandable for humans. For instance:

* **20231217\_project-name\_raw-data.csv**
* **20231217\_project-name\_manuscript\_V1.docx**
* **20231217\_project-name\_clean-data.csv**

In this naming system, the date is placed first, separated by an underscore, followed by the project name, and lastly, the type of file. Note that an underscore (\_) is used to separate elements within the metadata unit, and a dash (-) can be used within the metadata unit to distinguish words. Additionally, it's a good practice to include the initials of the first and last names to indicate the file's ownership. For example:

* 20231217\_project-name\_manuscript\_V1\_**PCH**.docx

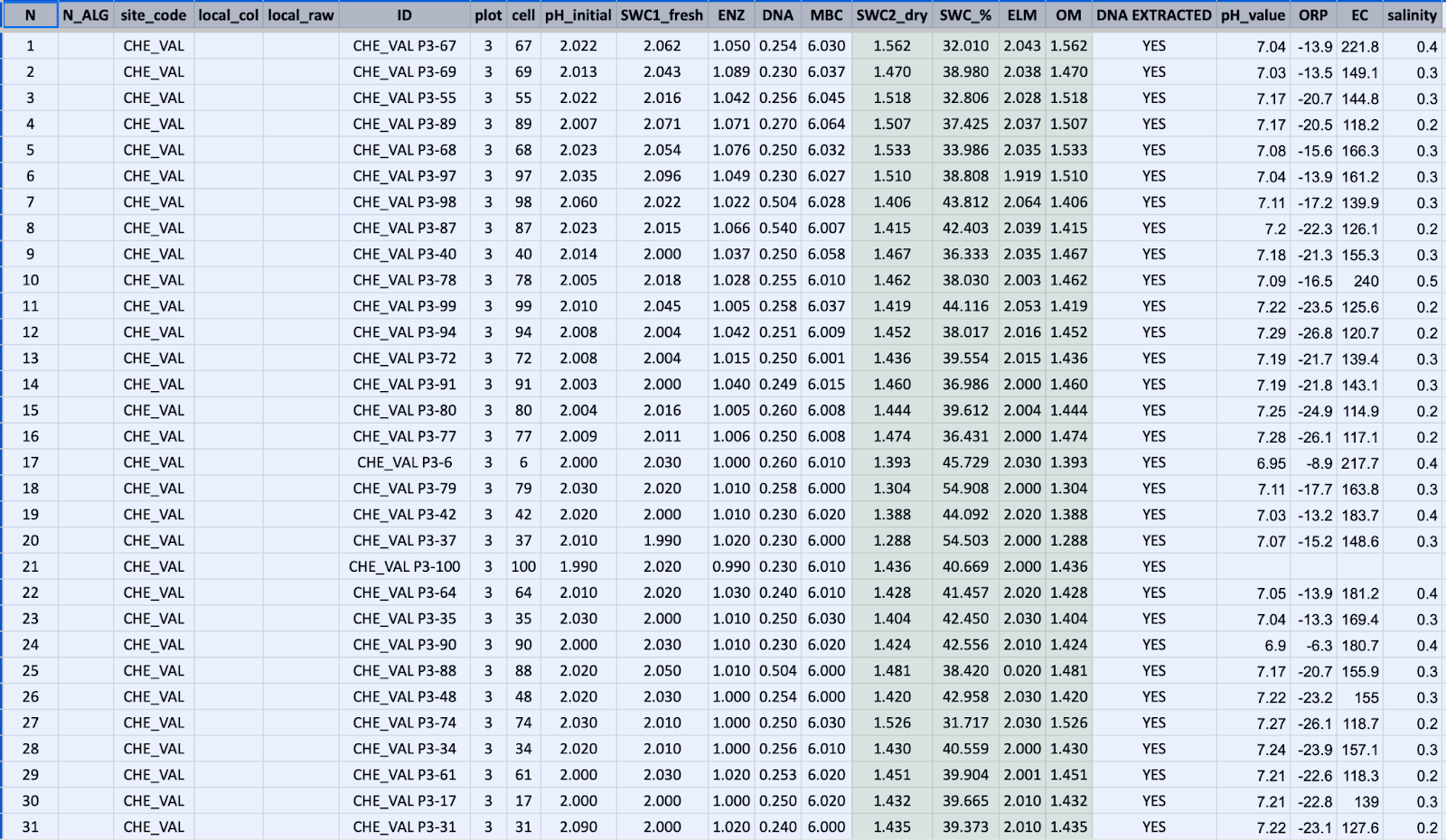
While there can be further discussions about date formats, for your specific scenario of a BSc thesis within a department without international collaborations, this naming convention should suffice.

**Metadata and README files**

Regardless of how descriptive the folder or file names may seem, it's essential to capture additional information for future reference. Metadata and README files serve as valuable supplementary data related to the actual content. Imagine you've organized your folders according to the principles we previously discussed and then share this folder with an external collaborator, or more likely, revisit it after several years. README files act as documentation, providing insights into the folder's structure and the types of analyses performed. For instance:

*‘File ECO4FUN\_species-diversity\_2020\_raw-data.csv contains the community composition raw data as were collected during the field season of 2020 by R.O.. I received the file by email on 2022-05-17, personally by R., on my gmail account. The file contains raw data and should NOT be modified. In case you need to modify the data, make a copy.’*

On the other hand, we refer to metadata as descriptive data for the raw data itself. Typically when working with excel sheets, metadata is a separate sheet. First column is a list with the column names of the actual data and second, is a description of each variable. 

  
**Figure 2:** Soil weights dataset. Notice that column names are both computer and human readable. There are no spaces nor special characters.

A screenshot of a computer

Description automatically generated  
**Figure 3:** Metadata for soil weights dataset.

**Version control systems (*git*)**

Version control, often implemented through tools like *Git*, is a system designed to track and manage changes in a project's files or codebase. This system proves to be particularly valuable for individuals and teams working with code or any set of files that undergo regular updates and modifications. In the context of coding, version control allows developers to:

1. **Track Changes**: It records every alteration made to the codebase, preserving a record of who made the changes and when.
2. **Manage Collaboration**: It facilitates collaborative work by enabling multiple contributors to work on the same codebase simultaneously. Version control systems merge changes from different team members efficiently.
3. **Revert to Previous States**: If a mistake is made or a new feature causes unexpected issues, version control provides the ability to revert the codebase to a previous, stable state.
4. **Branching and Merging**: Developers can create separate branches to work on new features or bug fixes independently. These branches can then be merged back into the main codebase when ready.
5. **Documentation**: Version control systems often include commit messages, which serve as documentation for why a particular change was made, helping team members understand the rationale behind each modification.
6. **Backup and Disaster Recovery**: Version control systems serve as a reliable backup mechanism, ensuring that project history and code are preserved even in the event of hardware failure or data loss.

In essence, version control is an invaluable tool for maintaining order, collaboration, and traceability in projects where changes occur regularly. While it is especially prevalent in software development, its benefits extend to any domain.

Particularly within the field of ecology, *R*, as a programming language, increased in popularity and is a valuable tool for many ecologists around the globe. One of the everyday tasks of ecologists, from students to professors, is to clean or analyze data, by writing and exchanging *R*-scripts. For instance, in 2017, among 30,000 published articles in the field of ecology, 58.0 % reported the use of *R* as a statistical programming environment (Lai J., et. al. 2019). As a consequence, over the past few years, version control has gained immense popularity among ecologists, emerging as a powerful platform for facilitating collaboration and the distribution of *R* code (Braga et al., 2023).

**GitHub**

Ecology has become a highly quantitative science which mostly relies on data collection, data manipulation and statistical analyses. Furthermore ecology has become highly collaborative, expanding across continents, scientific cultures and institutions. If back in the day researchers had to learn collaborating with colleagues in the same faculty, today they have to work in a completely international environment around the globe. Nowadays ecologists, to address questions for global scale problems, are expanding their methodological palette by synthesizing past experiments or collaborating to conduct global experiments (Muscarella R.  and  Poorter L., 2022). On this wavelength, we have to expand our knowledge and adopt tools of other fields (i.e. computer science, mathematics, statistics) which sometimes is time consuming and without proper training (Braga et. al. 2023). There are a plethora of protocols and good practices on simple tasks but very often are neglected as soon as the goal for ecologists is to explain the natural environment. For example:

* Good practices on naming and organizing files and computer folders
* Write readme.md, .txt and markdown files to keep track on changes
* Learn to manipulate data (wide vs long format)
* Learn how to write a code (e.g. in *R* or python)
* How to deal and organize a big dataset (SQL)
* Implement complicated statistical models and machine learning (e.g. random forest)
* Open and reproducible scripts and datasets

Very often, ecologists, as they try to interpret the natural world, pay less attention to improving their workflow and their scripts, although their results are the outcome of coding (i.e. in *R*, python or julia). In 2017, among 30,000 published articles in the field of ecology, 58.0 % reported the use of *R* as a statistical programming environment (Lai J., et. al. 2019). *R*, as a programming language, increased in popularity due to its community, because it is free and easily reproducible.

GitHub is the graphic interface of *git* version control systems. It has been around since 2008, and in the last 5-10 years has become a favored tool among ecologists. This surge in its adoption within the field of ecology aligns with the broader shift towards a more quantitative, open, and reproducible approach to scientific research. It's no coincidence that a tool initially designed for software development has found a natural home among ecologists. GitHub offers several advantages to scientists, primarily by providing a robust platform to store, track changes, and facilitate collaborative work on computer code development. Despite initial skepticism that often accompanies emerging tools, GitHub has proven its worth by potentially replacing traditional cloud storage services, streamlining communication (and reducing the flood of emails), supporting educational endeavors, and fostering closer connections between users and *R* package developers. In light of these benefits, GitHub should be recognized as a valuable tool for enhancing collaboration and increasing overall research efficiency within the field of ecology. Moreover, in recent years, GitHub has taken another significant stride by introducing "GitHub Copilot," an AI-powered feature that operates within its environment to predict code and offer helpful suggestions.

**References**

* **Braga, P. H. P.,**  Hébert, K.,  Hudgins, E. J.,  Scott, E. R.,  Edwards, B. P. M.,  Sánchez Reyes, L. L.,  Grainger, M. J., Foroughirad, V.,  Hillemann, F.,  Binley, A. D.,  Brookson, C. B.,  Gaynor, K. M.,  Shafiei Sabet, S.,  Güncan, A., Weierbach, H.,  Gomes, D. G. E., &  Crystal-Ornelas, R. (2023).  Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution. *Methods in Ecology and Evolution*,  14,  1364–1380. <https://doi.org/10.1111/2041-210X.14108>
* **Figueiredo L,** Scherer C, Cabral JS (2022). A simple kit to use computational notebooks for more openness, reproducibility, and productivity in research. *PLOS Computational Biology*, 18(9): e1010356. <https://doi.org/10.1371/journal.pcbi.1010356>
* **Muscarella R, Poorter L** (2022). Ten simple rules for managing communications with a large number of coauthors. *PLoS Comput Biol,* 18(6): e1010185. <https://doi.org/10.1371/journal.pcbi.1010185>
* **Lai, J.,**  Lortie, C. J.,  Muenchen, R. A.,  Yang, J., and  Ma, K..  (2019).  Evaluating the popularity of R in ecology. *Ecosphere*  10(1):e02567. [10.1002/ecs2.2567](https://doi.org/10.1002/ecs2.2567)