# Approachable case studies support learning and reproducibility in data science: An example from evolutionary biology

Luna L. Sanchez Reyes*
School of Natural Sciences, University of California, Merced
and
Emily Jane McTavish
School of Natural Sciences, University of California, Merced

February 14, 2022

**Abstract**

Research reproducibility is essential for scientific development. Yet, rates of reproducibility are low, especially in the natural sciences. As increasingly more research is relying on computing tools and software, efforts for improving reproducibility rates have focused on making available research workflows as computer code, as well as raw and processed data in computer readable form. However, research products that are digitally available are not necessarily friendly for learners and interested parties with little to no experience in the field. This renders research products unapproachable, which counteracts availability, and hinders reproducibility short and long term. To improve long term adoption of reproducible workflows in research, they need to be made approachable for learners, the researchers of the future.

Using an example within evolutionary biology, we identify aspects of research workflows that make them unapproachable to the general audience: use of highly specialized computing jargon and programming techniques; high cognitive load; unspecified, unclear or lengthy goals; content-focused descriptions instead of user-focused; inflexible learning environment; unapproachable (cold or intimidating) language; and little to no diversity of representation of information. Then, we propose a set of principles to improve the unapproachable aspects of research workflows, and illustrate their application in a case study from evolutionary biology that we used as teaching material. Finally, we elaborate on the general application of these principles for documenting research workflows and products, to provide present learners and future researchers with tools for successful scientific reproducibility.

*Keywords:* open science, R, phylogenetics, Open Tree of Life

*The authors gratefully acknowledge "Sustaining the Open Tree of Life", NSF ABI No. 1759838, and ABI No. 1759846.

# Introduction

Research reproducibility –the extent to which consistent results are obtained when a scientific experiment or research workflow is repeated (Curating for Reproducibility Consortium 2017)– is a key aspect of the advancement of science, as it constitutes a minimum standard that allows understanding research products (e.g., methods, data, analysis, results, etc.; **?**) to determine their reliability and generality, and eventually build up scientific knowledge and applications based on those products (King 1995, Peng 2011, Powers & Hampton 2019). In the natural sciences, rates of reproducibility are low (Ioannidis 2005, Prinz et al. 2011), which has elicited concerns about a crisis in the field (Baker 2016).

In response, the scientific community has been developing new principles and standards to incentivize cultural changes that support a long term improvement of reproducibility rates in the natural sciences (Peng 2015, Wilkinson et al. 2016, Miyakawa 2020). A standard for reproducibility that has received much attention is availability, which we define as a property denoting that a research product can be reached (acquired, copied, analyzed, processed and/or reused) at no financial, legal or technical cost (Arnold et al. 2019), and without geographic, demographic, social or temporal barriers for the population (Fecher & Friesike 2014).

In this paper, we argue that research products that are digitally available are often unapproachable in practice (Box 1), because they are not friendly for learners and interested parties with different levels of experience in the field. Research products that are unapproachable counteract availability, and hinder reproducibility short and long term. To support long term adoption of reproducible practices in the natural sciences, research workflows need to be made approachable for learners, the researchers of the future.

To elaborate on our thesis, we designed a case study within the research field of phylogenetics, a discipline within evolutionary biology. We use our case study to identify barriers that have made research workflows largely unapproachable to a general audience in the natural sciences. Then, we propose some principles for researchers to address these barriers, and create research workflows that are reproducible by a larger audience. The principles proposed here can be generalized and integrated into the undergraduate and graduate school STEM curriculum, either for courses specialized in reproducibility or within other subject areas, as a necessary component of successful and impactful science.

## A case study from phylogenetics

Phylogenetics is a key discipline within evolutionary biology (Dobzhansky 1973). It focuses on investigating the history of shared ancestry of living and extinct organisms using biological data, and represents this evolutionary history with a diagram known as a phylogeny or phylogenetic tree (because it grows through time and appears to have branches; Figure 1). Phylogenies provide the basis to study and understand all biological processes in an evolutionary context (Dobzhansky 1973) Hence, it appears that improving reproducibility rates in phylogenetics has the potential to positively impact research across the natural sciences.

To explore barriers to approachability in phylogenetics, we develop a case study that touches on three common problems within the field: standardizing organism names in phylogenies, obtaining current phylogenetic knowledge for a group of organisms, and summarizing this phylogenetic knowledge in a meaningful way. To address these problems, we propose a research workflow that relies on resources from the Open Tree of Life (OpenTree), an open source project that provides digital availability of phylogenetic results from published, peer-reviewed research, which is considered as vetted and state-of-the-art knowledge in the field. OpenTree phylogenies are stored in a public database, the Phylesystem (McTavish et al. 2015), and are downloadable as various computer-readable file types, which is key for reusability and reproducible workflows (Wilson et al. 2017). OpenTree also provides access to a single naming standard for organisms (taxonomic standard) that is applied to the stored phylogenies (Rees & Cranston 2017), which are then used to summarize a single phylogenetic tree encompassing all life (Open Tree Of Life et al. 2019).

All of OpenTree resources are free of cost to any user, and are available for download and use through its Graphical User Interface (aka, a website or application that allows users to access computer functionalities, in this case OpenTree resources, with mouse or keyboard clicks). However, reducing as many manual steps as possible in research workflows is key for reproducibility, as manual data manipulation scales poorly and is prone to error (Bakken 2019). OpenTree's resources are also programmatically available through its Application Programming Interface services (APIs; aka computer code that implements computer functionalities, in this case OpenTree resources, that can be used by programmers to build more functionalities), which provide scalability and reproducibility (Open Tree Of Life et al. 2016). However, this comes at a high cost for the user, which requires considerable more computer programming experience and literacy to be able to successfully use APIs. The R and Python programming languages are open source and free of cost, and represent two of the most widely used programming languages in the sciences today (Baker 2017). The rotl R package (Michonneau et al. 2016) and the OpenTree Python module (McTavish et al. 2021) are now available as wrappers for OpenTree's API services. As such, these packages should contribute to making OpenTree resources more accessible to a wider programming user audience.

However, while learners in the natural sciences have been engaging independently with R and Python programming languages, computer programming is not traditionally a core skill formally taught to biologists and naturalists (Sayres et al. 2018, Wright et al. 2019, Williams et al. 2019). As computers continue to play a larger role in most scientific disciplines (Piccolo & Frampton 2016), higher baseline computational skills are required across all natural sciences. Thus, efforts to increase reproducibility rates in the natural sciences must consider the specifics of scientific workflows that rely on usage of computer tools and programming languages (Peng 2011, Sandve et al. 2013, Powers & Hampton 2019).

In the next section que describe the barriers to approachability that we identified on our case study. We address these barriers by developing teaching materials that are available at https://mctavishlab.github.io/R_OpenTree_tutorials/.

# Identifying barriers to approachable research workflows

Cite carpentries and Mosi

Good primary documentation for code is thorough. It describes general usage of individual functions, the components and variables a function can take, and it should be accompanied with function usage examples on how to apply it (Karimzadeh & Hoffman 2018). As opposed to code, primary documentation is written in natural language (i.e., any known human language, e.g., English, Spanish, Chinese). Primary documentation is

R

```
ape::plot.phylo(my_tree, cex = 2) # or just plot(my_tree, cex = 2)
```

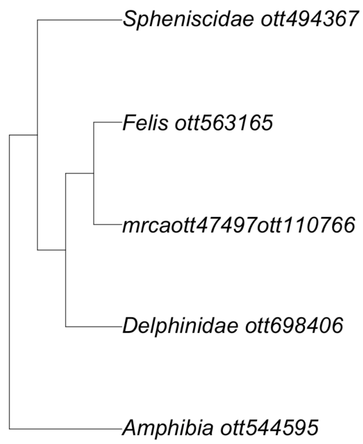


Figure 1: A phylogenetic tree from our tutorial. It was extracted using OpenTree of Life resources (Open Tree Of Life et al. 2019) wrapped in the `rotl` R package (Michonneau et al. 2016).

viewed as a key element for success of a piece of code (Karimzadeh & Hoffman 2018), which might be why it is also usually written using highly specialized computational jargon (i.e., computationally specific concepts, words, and phrases) as well as formal scientific language. While this might be important for formal acceptance of the code by the scientific and academic community, it often slows down or even obstructs examination, application, and adoption of code by the general audience (Ball 2017).

Some principles of an inclusive syllabus:

- high cognitive load/ skill level; (literate programming)

- unspecified, unclear or lengthy goals;(literate programming)

- content-focused descriptions instead of user-focused; (demonstrate errors)

- inflexible learning environment; (make it stable so users can learn whenever is best for them)

- unapproachable (cold or intimidating) language, use of highly specialized computing jargon; (use friendly and informal but respectful language)

- and little to no diversity of representation of information (say the same thing with different images, metaphors, stories)

# Best practices for approachable research workflows

### a. Literate programming: Demonstrate code usage with integrative examples

Pedagogical research shows that active learning practices are one the most effective ways to take on abstract subjects (Freeman et al. 2014). Programming computer languages are quite abstract and cognitive load can be greatly reduced for learners by applying an active learning strategy such as linking its usage to a "real world" or "human" application (Felder & Brent 2009).

A story-like narrative that links pieces of code together and invites learners to try the code, can lead learners to remember what they are doing and why they are doing it. This "literate programming" paradigm (Knuth 1984, Fritzson et al. 2002) makes code more approachable, as it integrates narratives with computer code in the same document, supporting learners in actively following the code usage, supporting memory and understanding (Piccolo & Frampton 2016).

We propose that documents developed with "literate programming" can be made more accessible by choosing narratives that are relatable to a more general audience. An easy way to do this in biology is choosing as model organism a charismatic taxon that a research group is more interested in studying. For the non-specialized user base, a highly charismatic group such as dinosaurs will do the trick.

We examined available primary documentation for the package `rotl`, and designed a narrative that required the usage of as many functions as possible. We demonstrate code applications that are commonly requested by OpenTree users, but that are not demonstrated in the primary documentation of the R package. By framing the function workflow using highly requested uses, the documentation acquires a narrative arc that is easier to follow and remember by users. This can also facilitate the application of code to other use cases in biology of interest for the users.

### b. Demonstrate errors and warnings thoroughly

A practice that has become more and more widespread in programming-language pedagogical practices is the use of typos and mistakes to normalize them for learners, and show them how to solve them when they are outside the classroom (Shannon & Summet 2015). Yet, this is rarely done on written pedagogical materials. Primary documentation focuses on demonstrating usage function with examples that work seamlessly, without errors. We argue that the opposite is needed to support adoption of reproducible workflows and support long term independence in learners (Gaspar & Langevin 2007). We demonstrate examples that do not work as expected and exemplify ways to address them (Figure 2).

We identify inputs that would give a wide range of warnings and errors, focusing on demonstrating these cases. This helps users to not be afraid of errors and warnings, but instead to use them to their advantage. We also identify effects of warnings and errors downstream of the workflow.

We identify ways to evaluate inputs to know if they will produce an error, and design alternatives on what to do when faced with an error or warning, and demonstrate these alternatives. One of the most essential skills in programming is interpreting and moving forward from errors. Many finely honed tutorials do not trigger errors, which precludes helping students to develop the tools to understand and address errors when they do encounter them, as they inevitably will. In our tutorial, we focus on explaining the meaning and downstream of warnings and errors, and showcase ways to detect them before they are triggered (i.e., before using an input that would elicit a warning or error). This has two pedagogical benefits: 1) it provides users/students with the means to troubleshoot their own warnings and errors, and 2) it allows users/students to understand with more depth what the function is doing.

### c. Avoid jargon and expert language

Besides avoiding formal language, and incorporating elements of pop culture, such as picture character icons known as “emojis“, to make the language more familiar to a broader target audience (see Figure 2), we made an effort to specifically complement the primary documentation by identifying computational concepts that were assumed or were not ex-

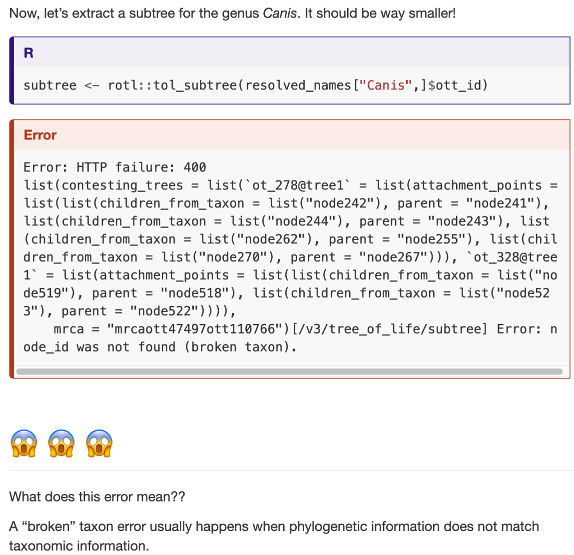


Figure 2: Snapshot of a section of the tutorial website, where we demonstrate a common error.

plained in depth. We vetted the tutorials through feedback from workshop participants as well as individual users. We choose examples that are charismatic for the audience. For example, when we presented the tutorial for a team specialized in Amphibians, we tailored the examples using frogs and their allies.

### d. Make it stable through time

We published the tutorials on a public, free license, free of cost, and free for use and reuse repository and persistent website (Sánchez Reyes, Luna L and McTavish, Emily Jane and Holder, Mark T 2021). The tutorial is available for the users to go back to any time they need it, and to be passed on to other users (Figure 3).

We created a main version of the tutorial that is stable. Any updates to the tutorial are published as new versions, or tutorials for new workshops (Wilson 2006, 2022) Versions presented at workshops are a copy from the original repository. They represent a temporally stable snapshot of functions and workflows presented during a workshop.

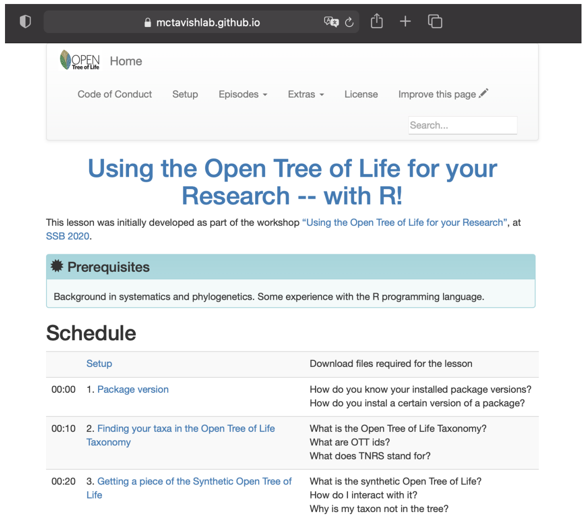


Figure 3: Snapshot of the home to our tutorial website, showing part of the schedule.

# Conclusion

Response form the community has been invaluable in gauging success of our teaching materials. Senior researchers often comment on the usefulness of the tutorials for their research, as well as how they have supported students in using the R packages with less help from them as PIs.

Making accessible reproducible workflows has several advantages:

* save explanation/training time when analyses are run again by students and collaborators; * save research time for yourself when analyses are run again with more data, a* different dataset, a different organism or biological model; * scientific efforts can build off of each other.

Ultimately, the long term improvement of reproducibility rates in science will depend on our ability to intentionally integrate the subject of reproducibility into the undergraduate curriculum, so college learners and future researchers have the basis to develop the fundamental skills needed to successfully create reproducible scientific workflows and materials.

Some universities have been incorporating the subject in their classes (see University of Washington Libraries - (2022), NIGMS Career Curriculum Development - (2015)). The focus of these resources has been for students to develop skills to document their work. The principles identified and outlined here can be used to set learning goals and outcomes on new reproducibility syllabi.

The principles to create tutorials described here facilitate adoption of software and analysis workflows among researchers at different academic levels, from undergrads to established researchers. It will also help close the gap between students that had access to computational resources (and computational training) from an early age and students that did not. Late access to computational resources and training can occur due to lack of economic resources, often occurring in households from underrepresented communities and minorities (Google Inc. & Gallup Inc. 2016, Warner et al. 2021). It can also be due to gender-biased parental and community pressures, in which male individuals are more often encouraged to perform activities related to computers, while female individuals are discouraged (Warner et al. 2021). These principles can be used to improve not only reproducibility practices, but also software adoption in the natural sciences.

## SUPPLEMENTARY MATERIAL

**Title:** Website and GitHub repository containing the complete teaching materials developed and demonstrated here.

**GitHub repository link:** `https://github.com/McTavishLab/R_OpenTree_tutorials`

**Website link:** `https://mctavishlab.github.io/R_OpenTree_tutorials`

# References


Arnold, B., Bowler, L., Gibson, S., Herterich, P., Higman, R., Krystalli, A., Morley, A., O'Reilly, M., Whitaker, K. et al. (2019), 'The turing way: a handbook for reproducible data science (version v1.0.1)', *Zenodo* .

Baker, M. (2016), 'Is there a reproducibility crisis?', *Nature* **533**(26), 353–66.

Baker, M. (2017), 'Scientific Computing: Code Alert', *Nature* **541**(7638), 563–565.

Bakken, S. (2019), 'The journey to transparency, reproducibility, and replicability'.

Ball, P. (2017), 'It's not just you: science papers are getting harder to read', *Nature* **30**.

Curating for Reproducibility Consortium (2017), 'Defining "reproducibility"'.
**URL:** *https://cure.web.unc.edu/defining-reproducibility/*

Dobzhansky, T. (1973), 'Nothing in biology makes sense except in the light of evolution', *The American Biology Teacher* **35**(3), 125–129.

Fecher, B. & Friesike, S. (2014), *Open Science: One Term, Five Schools of Thought*, Springer International Publishing, pp. 17–47.

Felder, R. M. & Brent, R. (2009), 'Active learning: An Introduction', *ASQ Higher Education Brief* **2**(4), 1–5.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H. & Wenderoth, M. P. (2014), 'Active learning increases student performance in science, engineering, and mathematics', *Proceedings of the National Academy of Sciences* **111**(23), 8410–8415.

Fritzson, P., Gunnarsson, J. & Jirstrand, M. (2002), Mathmodelica - an extensible modeling and simulation environment with integrated graphics and literate programming, *in* '2nd International Modelica Conference, March 18-19, Munich, Germany'.

Gaspar, A. & Langevin, S. (2007), Restoring "coding with intention" in introductory programming courses, *in* 'Proceedings of the 8th ACM SIGITE conference on Information technology education', pp. 91–98.

Google Inc. & Gallup Inc. (2016), 'Diversity gaps in computer science: exploring the underrepresentation of girls, blacks and hispanics', *Retrieved from http://goo.gl/PG34aH* (Additional reports from Google's Computer Science Education Research are available at g.co/cseduresearch.).

Ioannidis, J. P. (2005), 'Why most published research findings are false', *PLoS Medicine* **2**(8), e124.

Karimzadeh, M. & Hoffman, M. M. (2018), 'Top considerations for creating bioinformatics software documentation', *Briefings in Bioinformatics* **19**(4), 693–699.

King, G. (1995), 'Replication, replication', *PS: Political Science & Politics* **28**(3), 444–452.

Knuth, D. E. (1984), 'Literate programming', *The Computer Journal* **27**(2), 97–111.

McTavish, E. J., Hinchliff, C. E., Allman, J. F., Brown, J. W., Cranston, K. A., Holder, M. T., Rees, J. A. & Smith, S. A. (2015), 'Phylesystem: a git-based data store for community-curated phylogenetic estimates', *Bioinformatics* **31**(17), 2794–2800.

McTavish, E. J., Sánchez Reyes, L. L. & Holder, M. T. (2021), 'OpenTree: A Python Package for Accessing and Analyzing Data from the Open Tree of Life', *Systematic Biology* .
**URL:** *https://doi.org/10.1093/sysbio/syab033*

Michonneau, F., Brown, J. W. & Winter, D. J. (2016), 'rotl: an R package to interact with the Open Tree of Life data', *Methods in Ecology and Evolution* **7**(12), 1476–1481.

Miyakawa, T. (2020), 'No raw data, no science: another possible source of the reproducibility crisis'.

NIGMS Career Curriculum Development - (2015), 'Rigor & Reproducibility, National Institute of General Medical Sciences'.
**URL:** *https://www.nigms.nih.gov/training/instpredoc/Pages/admin-supplements-prev.aspx*

Open Tree Of Life, Redelings, B., Cranston, K. A., Allman, J., Holder, M. T. & McTavish, E. J. (2016), 'Open Tree of Life APIs v3.0', *Open Tree of Life Project* (Online Resources).
**URL:** *https://github.com/OpenTreeOfLife/germinator/wiki/Open-Tree-of-Life-Web-APIs*

Open Tree Of Life, Redelings, B., Sánchez Reyes, L. L., Cranston, K. A., Allman, J., Holder, M. T. & McTavish, E. J. (2019), 'Open tree of life synthetic tree v12.3', *Zenodo* .
**URL:** *https://doi.org/10.5281/zenodo.3937742*

Peng, R. (2015), 'The reproducibility crisis in science: A statistical counterattack', *Significance* **12**(3), 30–32.

Peng, R. D. (2011), 'Reproducible research in computational science', *Science* **334**(6060), 1226–1227.

Piccolo, S. R. & Frampton, M. B. (2016), 'Tools and techniques for computational reproducibility', *Gigascience* **5**(1), s13742–016.

Powers, S. M. & Hampton, S. E. (2019), 'Open science, reproducibility, and transparency in ecology', *Ecological Applications* **29**(1), e01822.

Prinz, F., Schlange, T. & Asadullah, K. (2011), 'Believe it or not: how much can we rely on published data on potential drug targets?', *Nature Reviews Drug discovery* **10**(9), 712–712.

Rees, J. A. & Cranston, K. (2017), 'Automated assembly of a reference taxonomy for phylogenetic data synthesis', *Biodiversity Data Journal* (5).

Sánchez Reyes, Luna L and McTavish, Emily Jane and Holder, Mark T (2021), 'Using the Open Tree of Life for your Research, with R v0.9.1', *Open Tree of Life* (Online Resources).
**URL:** *https://mctavishlab.github.io/R_OpenTree_tutorials/*

Sandve, G. K., Nekrutenko, A., Taylor, J. & Hovig, E. (2013), 'Ten simple rules for reproducible computational research', *PLoS Computational Biology* **9**(10), e1003285.

Sayres, M. A. W., Hauser, C., Sierk, M., Robic, S., Rosenwald, A. G., Smith, T. M., Triplett, E. W., Williams, J. J., Dinsdale, E., Morgan, W. R. et al. (2018), 'Bioinformatics core competencies for undergraduate life sciences education', *PloS One* **13**(6), e0196878.

Shannon, A. & Summet, V. (2015), 'Live coding in introductory computer science courses', *Journal of Computing Sciences in Colleges* **31**(2), 158–164.

University of Washington Libraries - (2022), 'Teaching Reproducibility'.
**URL:** *https://guides.lib.uw.edu/research/reproducibility/teaching*

Warner, J. R., Childs, J., Fletcher, C. L., Martin, N. D. & Kennedy, M. (2021), Quantifying disparities in computing education: Access, participation, and intersectionality, *in* 'Proceedings of the 52nd ACM Technical Symposium on Computer Science Education', pp. 619–625.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E. et al. (2016), 'The fair guiding principles for scientific data management and stewardship', *Scientific Data* **3**(1), 1–9.

Williams, J. J., Drew, J. C., Galindo-Gonzalez, S., Robic, S., Dinsdale, E., Morgan, W. R., Triplett, E. W., Burnette III, J. M., Donovan, S. S., Fowlks, E. R. et al. (2019), 'Barriers to integration of bioinformatics into undergraduate life sciences education: A national study of us life sciences faculty uncover significant barriers to integrating bioinformatics into undergraduate instruction', *PLoS One* **14**(11), e0224288.

Wilson, G. (2006), 'Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive', *Computing in Science & Engineering* **8**(6), 66–69.

Wilson, G. (2022), 'The Carpentries', *Website* .
**URL:** *http://software-carpentry.org*

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L. & Teal, T. K. (2017), 'Good enough practices in scientific computing', *PLoS Computational Biology* **13**(6), e1005510.

Wright, A. M., Schwartz, R. S., Oaks, J. R., Newman, C. E. & Flanagan, S. P. (2019), 'The why, when, and how of computing in biology classrooms', *F1000Research* **8**.