Promoting analysis reproducibility with accessibility: An example in evolutionary biology

Abstract

Reproducibility is essential for scientific development. Efforts to increase computational reproducibility have focused on increasing availability of code and data. However, availability does not imply accessibility, and the latter is infrequently addressed, even if it is key to achieve full workflow automatization and reproducibility. Using an example in evolutionary biology, we identify factors that have specifically affected accessibility in the natural sciences, and ways researchers can address it to ensure reproducible and automatic workflows. The Open Tree of Life project (OpenTree) has developed a platform that facilitates availability of results from evolutionary biology research. However, baseline required computational knowledge and skills to access them are often not found among the target users. While documentation is available, it is often written using highly specialized language that is also inaccessible for the average target user. We present a set of principles to generate documentation that improves accessibility of code and documentation and present a tutorial were we apply those principles.

Keywords: open science, education, R, phylogenetics, tutorials

1 Introduction

Reproducibility –the extent to which consistent results are obtained when a scientific experiment is repeated (Curating for Reproducibility Consortium (2021))– is a key aspect for the advancement of science, as it constitutes a minimum standard to understand scientific results, to determine their reliability and generality, and eventually to build more scientific knowledge upon those results (King (1995), Peng (2011), Powers & Hampton (2019)). Reproducibility rates in the natural sciences are low (Ioannidis (2005), Prinz et al. (2011)), prompting concerns about a reproducibility crisis in the field (Baker (2016)). The scientific community has united to incentivize cultural changes that will improve reproducibility rates long term, such as transparency, availability, and workflow automatization, to name a few (Peng (2015)). We argue that accesibility is a key aspect that must accompany availability (Box 1) in order to achieve full workflow automatization and reproducibility.

Box 1. Availability does not imply accessibility. One example we really like is the marshmallows in an office story. The marshmallow bags were there, availabe for anyone in the office to eat the marshmallows inside. But the marshmallows stayed there for days, weeks even. And it was not until someone opened the bag and put the marshmallows in a tray, that people started actually eating them. They were gone in a matter of hours. Code is not marshmallows. But the point is that by sharing your code and documenting it might not be enough for the general researcher to reproduce results. Researchers will certainly be able to eventually figure it out, but the time needed to do so might not be worthy for them, or might not be something they can invest in, even if it would be useful for them long term, the short term investment is too intense.

We focus on identifying factors that have specifically affected accessibility in the natural sciences, and ways researchers can address it to ensure reproducible and automatic workflows. We use an example in evolutionary biology. This example focuses on understanding the shared ancestry among organisms through evolutionary trees. These trees provide the basis to study and understand biological processes (Dobzhansky (1973)). Hence, improving reproducibility and availability of phylogenetic research is relevant for many aspects of biological research. The Open Tree of Life project (OpenTree) has developed a platform that

facilitates availability of results from phylogenetic research, by standardizing and storing phylogenetic data with the goal of synthesizing a single phylogenetic tree encompassing all life (OpenTreeOfLife et al. (2019)). All data in OpenTree is open access and available programmatically through its many Application Programming Interface (API) services (OpenTreeOfLife et al. (2021)). Additionally, R packages (Michonneau et al. (2016)) and Python libraries (Mctavish et al. (2021)) have been developed as wrappers for OpenTree API services to make them available to a wider programming audience. The R and Python OpenTree wrappers have been utilized by computer-literate individuals, to seamlessly establish reproducible workflows to use and reuse expert phylogenetic knowledge for biological research (Sánchez-Reyes & O'Meara (2019)) and education (Nguyen et al. (2020), Jodie Wiggins and Phylotastic Team (2021), Matt Wilkins and Galactic Polymath (2021)).

The OpenTree project demonstrates that efforts to increase reproducibility and availability have also increased the baseline required computational knowledge and skills in phylogenetic research. These computational skills requirements are likely increasing across all natural sciences. Computing is not traditionally a core skill taught to biologists and naturalists. However, many students are now being trained in R as a statistics and data analysis environment. In order for reproducible computational tools to be adopted for research, they need to be much more accessible to reseachers. Data and code availability is a core requirement for reproducibility research (Peng (2011), Sandve et al. (2013), Powers & Hampton (2019)). However, the utility of data resources is limited by the technical challenges of accessing the data. In order to motivate reproducible research, that gap needs to be bridged. In this work we focus on improving accessibility of code examples and documentation. In particular, we identify the necessity for documentation that is written down using language that is common to the target audience to facilitate examination, application, and adoption of code by the wider audience.

We present a set of principles to generate documentation that improves accessibility of code and documentation. We applied these principles to a series of tutorials and vignettes for the OpenTree project.

2 Methods

Identifying hurdles to accesibility

Good primary documentation for code 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. As opposed to code, primary documentation is written in natural language (i.e., any known human language, e.g., English, Spanish, Chinese) and usually makes use of highly specialized computational jargon (computationally specific concepts, words, and phrases) as well as formal language, which often slows down or even obstructs examination, application, and adoption of code by external individuals. Because primary documentation is considered a professional document, acceptance of the research by the scientific community could be reduced if a more informal language is used. Secondary types of documentation, such as vignettes and tutorials, demonstrate additional cases of individual function usage, and describe analysis workflows in more detail, as well as function associations to generate a specific analysis and results. While secondary documentation has become more common practice, it is still often generated using highly specialized language.

Adressing hurdles to accesibility: the principles

1. Demonstrate integration of function usage with motivating examples

We examined available primary documentation for the OpenTree API R wrapper (the package 'rotl'), and designed a workflow that visits as many functions as possible, and demonstrate uses commonly requested by OpenTree users that are not demonstrated elsewhere. 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 facilitates the translation of functions to specfic use cases in biology.

For the tutorial demonstrated here, we used the commonly requested use case of obtaining a phylogenetic tree for all lineages within a specific taxonomic rank.

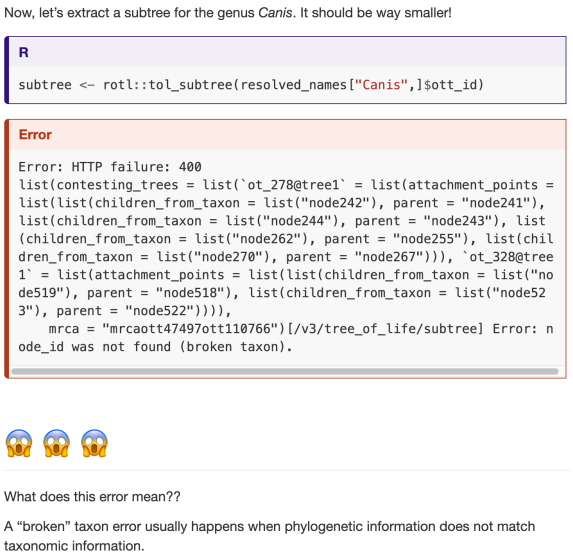

Figure 1: Snapshot of a section of the tutorial website, where we demonstrate a common error.

2. Demonstrate errors and warnings thoroughly

Primary documentation focuses on demonstrating usage function with examples that work seamlessly, without errors. We argue that the opposite is needed to support user adoption of reproducibile workflows: demonstrate examples that do not work as expected and exemplify ways to address them (Figure 1). 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.

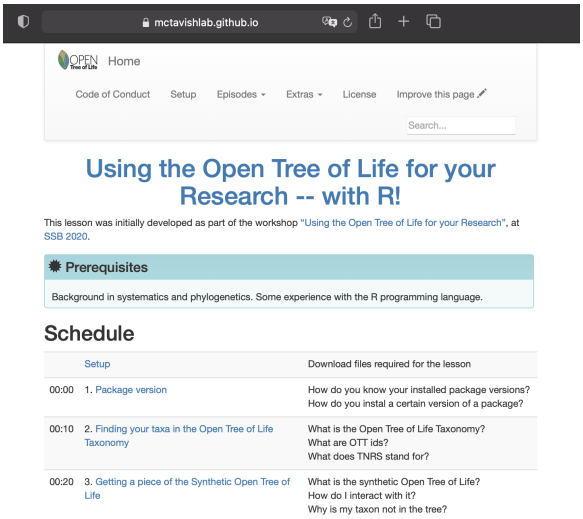

Figure 2: Snapshot of the home to our tutorial website, showing part of the schedule.

3. 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 used more familiar to the target audience (see Figure 1), we made an effort to specifically complement the primary documentation by identifying computational concepts that were assumed or were not explained in depth. We vetted the tutorials with an audience on workshops as well as individual user. 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.

4. 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 (Sanchez-Reyes, Luna L and McTavish, Emily Jane and Holder, Mark T (2021*a*,*b*)). The tutorial is available for the users to go back to it any time they need it, and to be passed on to other users (Figure 2).

Following the carpentries, we created a main version of the tutorial that is updated. Versions presented on workshops are a copy from the original repository, and represent a stable and temporal snapshot of the functions and workflows presented in the tutorial.

3 Results and Discussion

We explain the warnings and errors and design ways to avoid them, and detect them beforehand (i.e., before using an input that would give an error). We explain the consequences of warnings. We designed ways to access the different elements of the outputs. We have received emails from senior researchers thanking us for this materials, and students have been able to engage using the packages with less hep from the PIs.

The principles to create tutorials described here facilitate adoption of software and analysis wokflows among researchers at different academic levels, from undergrads to established researchers. It will also help closing 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 ocurring in households from underrepresented communities and minorites. It can also be due to gender-biased parental and community pressures, in which males are more often encouraged to perform activities related to computers, while females are discouraged. How to balance software acceptance VS. adoption? These principles can be used to aide not only reproducibility, but also software adoption in the natural sciences. Discuss: why address accessibility and not other apects of reproducibility?

4 Conclusion

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

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

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

Curating for Reproducibility Consortium (2021), '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.

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

Jodie Wiggins and Phylotastic Team (2021), 'Scientific data in your classroom'.
URL: *https://jwiggi18.github.io/phyloEd/*

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

Matt Wilkins and Galactic Polymath (2021), 'support K-16 education and scicomm'.
URL: *https://github.com/galacticpolymath/galacticEdTools*

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.

Nguyen, V. D., Nguyen, T. H., Tayeen, A. S. M., Laughinghouse IV, H. D., Sánchez-Reyes, L. L., Wiggins, J., Pontelli, E., Mozzherin, D., O'Meara, B. & Stoltzfus, A. (2020), 'Phylotastic: improving access to tree-of-life knowledge with flexible, on-the-fly delivery of trees', *Evolutionary Bioinformatics* **16**, 1176934319899384.

OpenTreeOfLife, Redelings, B., Cranston, K. A., Allman, J., Holder, M. T. & McTavish, E. J. (2021), 'Open Tree of Life APIs v. 3.0'.
URL: *https://github.com/OpenTreeOfLife/germinator/wiki/Open-Tree-of-Life-Web-APIs*

OpenTreeOfLife, Redelings, B., Reyes, L. L. S., Cranston, K. A., Allman, J., Holder, M. T. & McTavish, E. J. (2019), 'Open tree of life synthetic tree'.
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.

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.

Sánchez-Reyes, L. L. & O'Meara, B. C. (2019), 'Datelife: Leveraging databases and analytical tools to reveal the dated tree of life', *bioRxiv* p. 782094.

Sanchez-Reyes, Luna L and McTavish, Emily Jane and Holder, Mark T (2021*a*), 'McTavishLab/R_OpenTree_tutorials v0.9.1: Using the Open Tree of Life for your Research, with R'.
URL: *https://github.com/McTavishLab/R_OpenTree_tutorials*

Sanchez-Reyes, Luna L and McTavish, Emily Jane and Holder, Mark T (2021*b*), 'Using the Open Tree of Life for your Research, with R'.
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.