COLUMN
STUDENT ANGLE

Computing Skills Beyond R for Fisheries Students

Caleb A. Aldridge | Mississippi State University, Department of Wildlife, Fisheries, and Aquaculture | Mississippi Cooperative Fish and Wildlife Research Unit, Mail Stop 9690, Mississippi State, MS 39762. E-mail:caa134@msstate.edu

There is a growing imbalance of computing skills among fisheries students and recent graduates. Computers are the interface for analyses (e.g., descriptive statistics in a spreadsheet) and communication (e.g., email messages and journal articles) of the modern biologist, but complementing skills go untaught or overlooked. A casual inquiry of course offerings, workshops, and informal training for computing skills show a focus on data analysis via statistical computing languages (e.g., R and Python). Statistical prowess takes a lot of hard, continuous work and learning a computer language is transformational, but it seems rare to find courses aimed at equipping students with the skills to improve reproducibility, collaboration, and communication. Here, I highlight a few of these "other" computing skills that I have found helpful. Incorporating them into my workflow has streamlined processes and helped me create stable records of my research processes. These other computing skills are not a silver bullet and require time and commitment when appropriating into your workflow. But, the return on investment can be huge for individuals, research groups, and the wider fisheries community. I encourage agencies, organizations, and academic departments to consider workshops and coursework to jumpstart learning and adoption of these other computing skills.

I, like Shamaskin (2016), am spending my time as a graduate student indoors, writing scripts for computer applications, rather than outdoors sampling fish communities or running mesocosm experiments at the field station. I love what I do, but it is a change from what drew me to the field. I make up for it on the weekends though, spending time exploring nearby creeks and public lands. That downtime is important to me for my physical, mental, emotional, and spiritual health. Your downtime is valuable too. That's why I want to highlight three "indoor" skills that have streamlined my workflow, allowing my weekends to be mine—even if I do occasionally choose to spend them inside reading the latest issue of *Fisheries*! Additionally, I believe that these skills are increasingly needed in an evolving and growing discipline, especially as data sets grow larger (i.e., "Big Data") and reproducibility remains an issue (Baker 2015).

Below I aim to highlight notebooks, typesetting, and version control in an accessible way. These types of software programs can help you build informative, even interactive deliverables from source code, produce highly professional documents without having to fiddle with page layouts, and keep a permanent record of changes to important documents. I consider each in turn and provide some freely available resources at the end of each section. I wrap up with a challenge to all of us to make our workflows more efficient, but, more importantly, meet the demand of reproducibility in science.

A couple of short disclaimers: (1) I'm not an expert with the following tools, but have become competent with them within a year or so. Hopefully, that comes as encouragement and not a loss in confidence; if I can do it, you can do it! (2) There are several alternative brands to the tools I highlight here. I'm not promoting one over others; I can only share what I know.

NOTEBOOKS

I use the statistical program R (available: https://www.r-project.org/)—a free and flexible program and language that can perform a wide range of analyses and produce publication-quality graphics—and R Studio (available: https://www.rstudio.com/)—an integrated development environment known for making R more user friendly with production-focused features. R and R Studio are popular with many scientists, fisheries students, and faculty (http://bit.ly/39FpsU0). However, it can sometimes be confusing or frustrating when viewing code you wrote 6 months earlier… "What does that function do again?"

Notebooks enable you to document your code (i.e., analysis) with great detail. They can be created in R Markdown (available: https://rmarkdown.rstudio.com/), and Jupyter (available: https://jupyter.org/) if you are keen to Python, fairly easily. In R Studio, creating a notebook (file extension `.Rmd`) is as simple as selecting "R Markdown" from the create file menu. This special file type enables you to combine R code (`.r`) with markdown (`.md`), a lightweight, text-based markup language, so that you can narrate and comment as you analyze. What's more, you can create deliverables, such as an annual report, by adding some YAML (**YAML** **A**in't **M**arkup **L**anguage) in the preamble section. Compiling the script into a final format, sc., `.html`, `.pdf`, `.doc`, or `.rtf`, is as easy as a click of a button or by entering *`render()`* in the console. The best parts are lower rates of user error (e.g., entered wrong mean total length into report) and no fiddling with table and figure layouts—they're automatically added from the script you wrote!

Notebooks become handy when working with those not familiar with the R language—perhaps your advisor, collaborators, reviewers, or students. Plus, this creates a reproducible document that can be easily revisited. Take your future self as an example. Imagine several months after you've submitted your manuscript to your committee, or even a journal (go you!), someone asks you to justify why you chose a certain procedure. The likely reason they're asking is that the manuscript lacked detail or a key reference(s). Keeping a notebook allows you to guard against this anxiety and the frustration of trying to recall or track down that reference(s). As you are exploring data and model fitting, notebooks help you keep detailed comments and references at the line level. I cannot overstate how important it is to

© 2020 American Fisheries Society
DOI: 10.1002/fsh.10392

```r
# Comments usually come in the form of:

# ------------------------- #
# Block Comments...
# ------------------------- #

# Line comments...
2 + 2  # and inline comments

# ------------------------- #
# EXAMPLE
# ------------------------- #

if (!require(pacman)) {  # check to see if package is installed
    install.packages("pacman")  # if package isn't already, install it
    library(pacman)  # call installed package
}

# The function below does the same as above but for multiple of packages
pacman::p_load(ggplot, dplyr, lme4)

# One can also
## use 'levels' of
### number signs

# ************************* #
# Main Point: Take time to comment,
# no matter how you style it.
# ************************* #
```

Figure 1. R code with comments.

comment thoroughly as you write your code (Figure 1). You can see an example notebook output and more details at http://bit.ly/2TywbcW.

Free resources:

R Studio. 2014. R Markdown reference guide. R Studio, Inc., Boston. Available: http://bit.ly/2VYHX20 (CC BY 4.0).

R Studio. 2016. R Markdown cheat sheet. R Studio, Inc., Boston. Available: http://bit.ly/39Dzc17 (CC BY-SA 4.0).

Xie, Y., J. J. Allaire, and G. Grolmund. 2019. R Markdown: The definitive guide. CRC Press, Taylor & Francis Group, LLC., Boca Raton, Florida. Available: http://bit.ly/2TJH9Lm (CC BY-NC-SA 4.0).

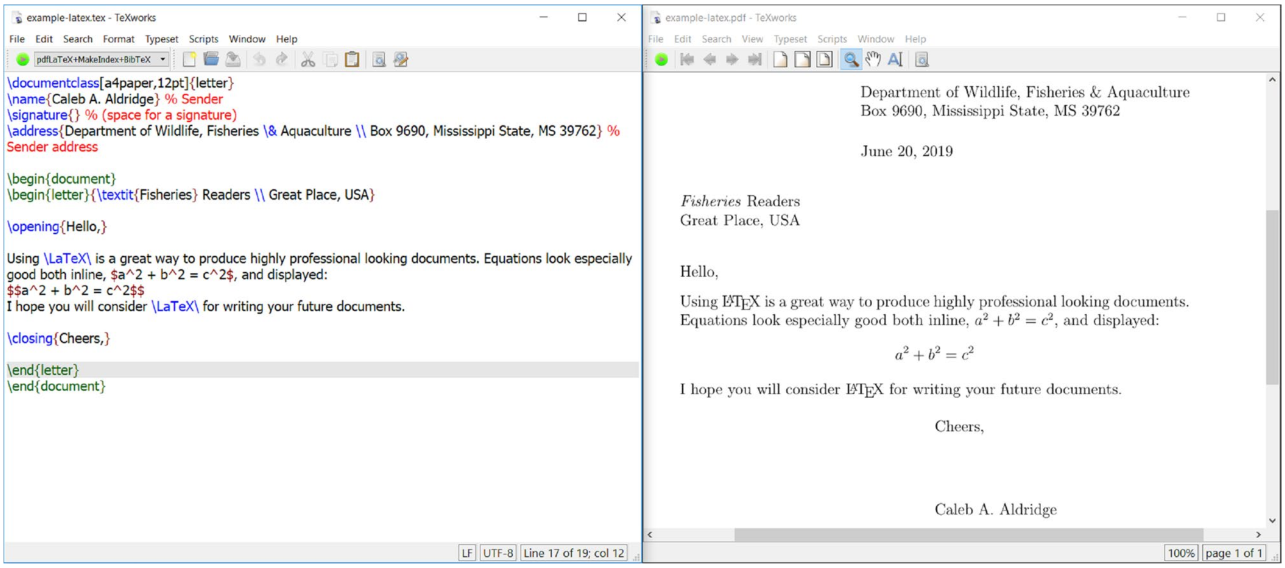
TYPESETTING

Imagine writing a manuscript (e.g., your thesis) without having to stress about the formatting and arrangement. Or, not worrying if adding in that extra table will produce something akin to a Picasso painting! Enter typesetting. Typesetting is different from word processing (e.g., Microsoft Word), to which you're probably accustomed. Typesetting uses markup commands for formatting and arrangement. (We've already encountered a couple of markup languages above, markdown and YAML). This means you write commands and text in your favorite text editor or development environment, then the script is sent to a compiler to produce the document. This is fundamentally the same process as writing an R script and sending it to the console or when you *render()* an .Rmd file.

One of the most popular typesetting languages is LaTeX (available: https://www.latex-project.org/). With LaTeX, pronounced *LAH-tekh*, you define the document class (type), any required packages, add sections, text, and equations, then compile for a professional looking document (Figure 2). LaTeX is free and text-based, making files (i.e., .tex) small, portable, and stable. This is great for lengthy documents (e.g., your thesis), and LaTeX allows you to painlessly cite while writing via a .bib file—a bibliography file format that can be exported by most reference management systems like EndNote, Mendeley, and Zotero. Free LaTeX compilers allow you to format citations and references in your favorite style (e.g., APA, MLA) or for a publisher/journal (e.g., Wiley/AFS.). Compilers can export your document in several file types (.html, .pdf, .xml, or .rtf) for easy sharing. Additionally, LaTeX can be integrated with other software, such as in R Markdown documents, for a high level of automation.

I won't lie, LaTeX has a pretty steep learning curve. I'd only consider myself "weekend hobbyist" level, even after writing half a dozen documents using it. Luckily, there are a few people that have developed nice tools and tutorials that make the transition from a word processor to typesetting much more amiable. First, I'll point to LyX (available: https://www.lyx.org/). LyX is an open source document processor specifically built for writing with LaTeX in a graphical interface. It can be a little confusing to work with at first, but it will quickly become preferable to word processors. Second, is an online graphical interface called AUTHOREA (available: https://www.authorea.com/). The basic plan (free) allows users to write seamlessly in plain-text, markdown, LaTeX, and insert code blocks. AUTHOREA also comes with nice features such as collaborative writing, version control, journal-specific formatting, and citation uploading and searching. I explored AUTHOREA in writing this document and have to say that I'm impressed. Both programs allow you to retain .tex files for safe storage.

I find one of the best ways to use LaTeX is within an R Markdown document. The advantage is that R users do not have to learn a completely new language, just some of the more common commands and syntax. What makes the integration of LaTeX and R so great is that you can produce statistics, tables, and figures within a nicely formatted document. Since R


example-latex.tex - TeXworks
File Edit Search Format Typeset Scripts Window Help
pdfLaTeX+MakeIndex+BibTeX
example-latex.pdf - TeXworks
File Edit Search View Typeset Scripts Window Help

```latex
\documentclass[a4paper,12pt]{letter}
\name{Caleb A. Aldridge} % Sender
\signature{} % (space for a signature)
\address{Department of Wildlife, Fisheries \& Aquaculture \\ Box 9690, Mississippi State, MS 39762} % Sender address

\begin{document}
\begin{letter}{\textit{Fisheries} Readers \\ Great Place, USA}

\opening{Hello,}

Using \LaTeX\ is a great way to produce highly professional looking documents. Equations look especially good both inline, $a^2 + b^2 = c^2$, and displayed:
$$a^2 + b^2 = c^2$$
I hope you will consider \LaTeX\ for writing your future documents.

\closing{Cheers,}

\end{letter}
\end{document}
```

LF | UTF-8 | Line 17 of 19; col 12

Department of Wildlife, Fisheries & Aquaculture
Box 9690, Mississippi State, MS 39762

June 20, 2019

Fisheries Readers
Great Place, USA

Hello,

Using LaTeX is a great way to produce highly professional looking documents. Equations look especially good both inline, $a^2 + b^2 = c^2$, and displayed:

$$a^2 + b^2 = c^2$$

I hope you will consider LaTeX for writing your future documents.

Cheers,

Caleb A. Aldridge

100% | page 1 of 1

Figure 2. A LaTeX formatted document in the text editor (left) and the compiled document (right).

is object-oriented, the tables and figures are updated when new data is added. No need to rewrite code, export figures, and fiddle with margins. However, defining the document format can become cumbersome for highly customized formats and may require more advanced knowledge of R Markdown, LaTeX, and YAML, along with some specific R and LaTeX packages (e.g., `bookdown`).

Free resources:

Chang, W. 2014. EX 2 cheat sheet. Available: http://bit.ly/33472d7

CTAN (Comprehensive TeX Archive Network). 2019. Starting out with TeX, LaTeX, and friends. Available: http://bit.ly/3aJmCOb

Xie, Y., J. J. Allaire, and G. Grolmund. 2019. R Markdown: The definitive guide. CRC Press, Taylor & Francis Group, LLC., Boca Raton, Florida. Available: http://bit.ly/2TJH9Lm (CC BY-NC-SA 4.0).

VERSION CONTROL

I've saved the best for last. You may have noticed that I have made somewhat of a big deal about files being text-based. That's because, in addition to being small, portable, and stable, they can be "tracked" using a version control system (VCS). Using a VCS, such as Git (Git Team 2019) or Subversion (Apache Software Foundation 2019), allows you to save "snapshots" of changes you make to files into a repository (i.e., database) over the life of the project. Microsoft Word's "Track Changes" is somewhat analogous—except VCS has unlimited "undo" ability. Additionally, a VCS allows multiple collaborators to make changes to a document *simultaneously* (i.e., branching) and "merge" each collaborator's edits, without having to keep track of multiple files and versions. We all know how frustrating that can be (Figure 3). What's more, changes are committed with metadata allowing one to view author and time of changes. If not already apparent, using a VCS can help keep a project organized, reproducible, and shareable.

Getting started with a VCS can be intimidating. But, the most common commands (e.g., `git init`, `git status`, `git add`, and `git commit`) can be learned quickly and used efficiently to record file changes. Plus, with many good resources available via the Internet and seamless integration with friendly user interfaces (e.g., R Studio) you can add this skill to your toolbelt quickly. I keep a Git cheat sheet (link below) hanging above my desk and have never really needed any other commands.

Free resources:

Bryan, J. 2019. Happy Git and GitHub for the useR. Available from: https://happygitwithr.com/ (CCBY-NC 4.0).

Chacon, S., and B. Straub. 2014. Git Pro (Second Edition). Apress. Available from: http://bit.ly/3aH5uso (CC BY-NC-SA 3.0).

Collins-Sussman, B., B. W. Fitzpatrick, and C. M. Pilato. 2011. Version control with Subversion: For Subversion 1.7: (Compiled from r5930). Available: http://bit.ly/2TRBaEA (CC BY 2.0).

GitHub Training Team. n.d. Git cheat sheet. GitHub. Available: https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf.

Stephens, N. 2019. Using version control with RStudio. R Studio Support. Available: https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN.

CONCLUSION

At this point you might be thinking, "Weird flex, you geeky nerd, but why is all this even that important?" Hopefully, my appeal to your selfish gene hooked you and you have invested the few minutes to land here. But I now want to take a more serious turn and appeal to the scientist in you. These tools are about reproducibility and readying ourselves in a rapidly evolving discipline. Reproducibility is at the foundation of our scientific processes (Powers and Hampton 2018) and ecological and environmental sciences are increasingly adopting data-driven methods (i.e., Big Data and machine learning; Hampton et al. 2013; Whittier et al. 2016). Our field is also innovating many types of information (e.g., public comments, harvest records, and population simulations), but it is uncommon that a typical university program teaches students the skills necessary to handle such advances (Dudley and Butte 2009; Noble 2009; Brewer and Smith 2011; Smith 2015; Hampton et al. 2017). On top of that, science in general has come under scrutiny in recent years as irreproducibility has been exposed, which adds pressure to develop computing skills for open science (i.e., increased openness [transparency], integrity, and reproducibility of research; Baker 2015;

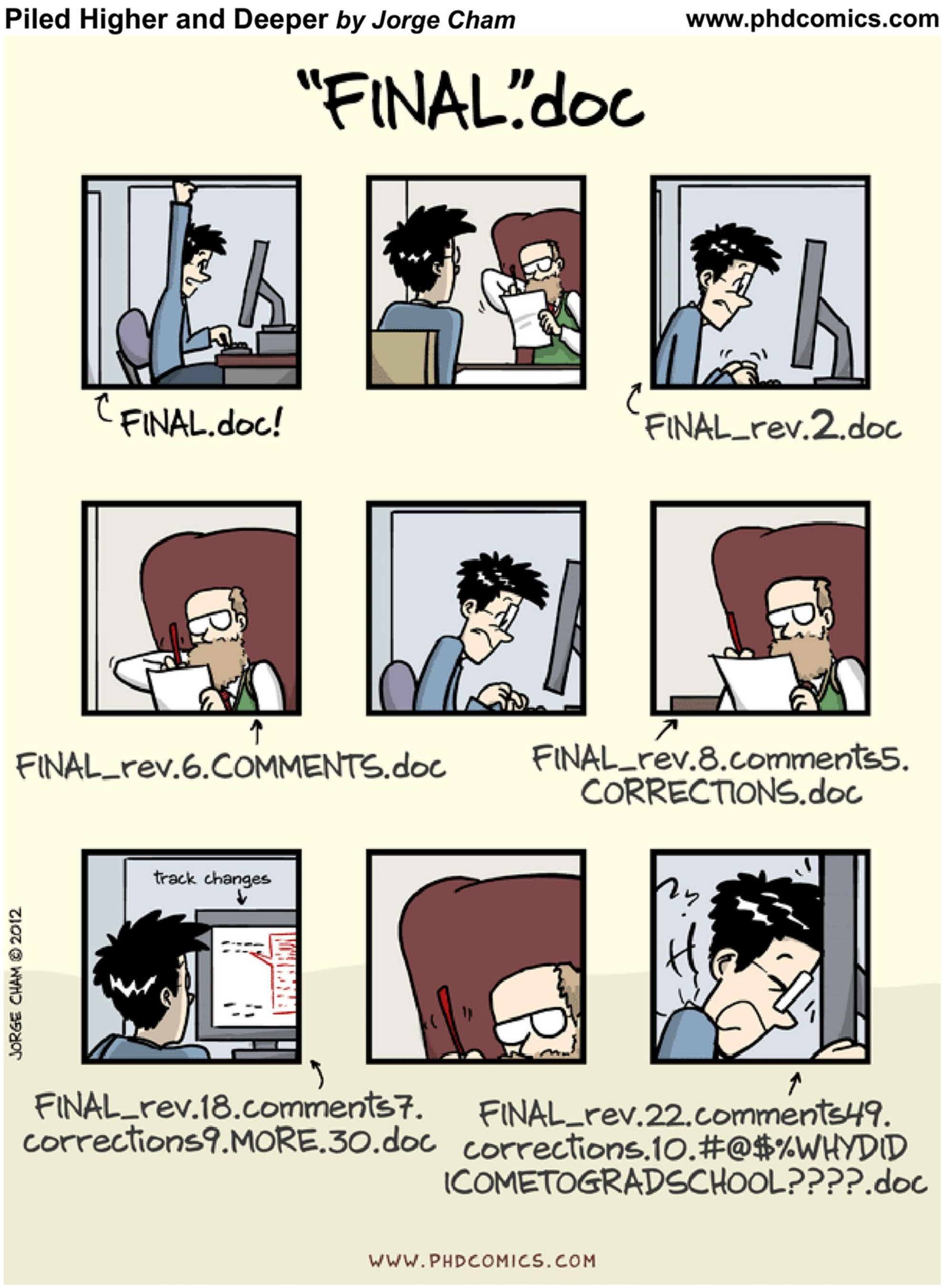

Piled Higher and Deeper by Jorge Cham
www.phdcomics.com
"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Figure 3. Cham, Jorge. 2012. "notFinal.doc". http://bit.ly/38Aljzz

Baker and Dolgin 2017; Hampton et al. 2017; Lowndes et al. 2017).

I encourage you to dive deeper. I believe that Lowndes et al. (2017), Hampton et al. (2017), Munafò et al. (2017), and Lewis et al. (2018) best frame the issues we are facing as students and young professionals; Dudley and Butte (2009) and Noble (2009) give good practical advice on staying organized while "indoors;" and Allesina and Wilmes (2019), along with the numerous other resources I list, give step-by-step instructions on how to integrate these and additional tools into your workflow. But don't stop there; discuss these issues and shortcomings in education with your peers, colleagues, and faculty members. There are many challenges and problems on the horizon, and we need to be prepared to solve them. Let's not trip over our own feet.

Additional resources:

Allesina, S., and M. Wilmes. 2019. Computing skills for biologists: a toolbox. Princeton University Press, Princeton, New Jersey. Available: https://press.princeton.edu/titles/13268.html.

Cooper, N., and P.-Y. Hsing. 2017. A guide to reproducible code in ecology and evolution. British Ecological Society, London. Available: http://bit.ly/2wG83MA (CC BY 4.0).

Harrison, K. 2018. Data management. British Ecological Society, London. Available: http://bit.ly/2wG83MA (CC BY 4.0).

Wickham, H., and G. Grolemund. 2016. R for data science: import, tidy, transform, visualize, and model data. O'Reilly Media, Inc., Sebastopol, California. Available: http://bit.ly/339PwUX (CC BY-NC-ND 3.0 US).

ORCID

Caleb A. Aldridge https://orcid.org/0000-0002-7655-9535

REFERENCES

Apache Software Foundation. 2019. Apache Subversion, Version 1.12.0. Available: https://subversion.apache.org

Baker, M. 2015. Over half of psychology studies fail reproducibility test. Nature News (August 27). Available: https://go.nature.com/39BCbaz

Baker, M., and E. Dolgin. 2017. Cancer reproducibility project releases first results. Nature 541:269–270.

Brewer, C. A., and D. Smith. 2011. Vision and change in undergraduate biology education: a call to action. American Association for the Advancement of Science, Washington, D.C.

Dudley, J. T., and A. J. Butte. 2009. A quick guide for developing effective bioinformatics programming skills. PLoS Computational Biology 5:e1000589.

Git Team. 2019. Git Version Control System, Version 2.22.0. Available: https://git-scm.com/.

Hampton, S. E., M. B. Jones, L. A. Wasser, M. P. Schildhauer, S. R. Supp, J. Brun, R. R. Hernandez, C. Boettiger, S. L. Collins, L. J. Gross, D. S. Fernández, A. Budden, E. P. White, T. K. Teal, S. G. Labou, and J. E. Aukema. 2017. Skills and knowledge for data-intensive environmental research. BioScience 67:546–557.

Hampton, S. E., C. A. Strasser, J. J. Tewksbury, W. K. Gram, A. E. Budden, A. L. Batcheller, C. S. Duke, and J. H. Porter. 2013. Big data and the future of ecology. Frontiers in Ecology and the Environment 11:156–162.

Lewis, K. P., E. V. Wal, and D. A. Fifield. 2018. Wildlife biology, big data, and reproducible research. Wildlife Society Bulletin 42:172–179.

Lowndes, J. S. S., B. D. Best, C. Scarborough, J. C. Afflerbach, M. R. Frazier, C. C. O'Hara, N. Jiang, and B. S. Halpern. 2017. Our path to better science in less time using open data science tools. Nature Ecology & Evolution 1:0160 Available: https://doi.org/doi:10.1038/s41559-017-0160.

Munafò, M. R., B. A. Nosek, D. V. M. Bishop, K. S. Button, C. D. Chambers, N. P. du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. P. A. Ioannidis. 2017. A manifesto for reproducible science. Nature Human Behaviour 1:0021 Available: https://doi.org/doi:10.1038/s41562-016-0021.

Noble, W. S. 2009. A quick guide to organizing computational biology projects. PLoS Computational Biology 5:e1000424.

Powers, S. M., and S. E. Hampton. 2018. Open science reproducibility, and transparency in ecology. Ecological Applications 29:e01822.

Shamaskin, A. 2016. The end parenthesis: how a fisheries student tackled programming. Fisheries 41:714–715.

Smith, D. 2015. Vision and change in undergraduate biology education: chronicling change, inspiring the future. American Association for the Advancement of Science, Washington, D.C..

The LaTeX Project. N.D. EX A document preparation system. Available: https://www.latex-project.org/

Whittier, J., N. Sievert, A. Loftus, J. M. Defilippi, R. M. Krogman, J. Ojala, T. Litts, J. Kopaska, and N. Eiden. 2016. Leveraging BIG data from BIG databases to answer BIG questions. Fisheries 41:417–419. AFS