A reproducible manuscript workflow with a Quarto template

**Richard A. Erickson,\* Althea A. Archer, Michael N. Fienen**

**R.A. Erickson**  
U.S. Geological Survey, Upper Midwest Environmental Sciences Center, La Crosse, WI  
**A.A. Archer**  
U.S. Geological Survey, Water Resources Mission Area, Reston, VA  
**M.N. Fienen**  
U.S. Geological Survey, Upper Midwest Water Science Center, Madison, WI  
\*Corresponding author: (rerickson@usgs.gov)

# Abstract

Scientists and resource managers increasingly use Markdown-based tools to create reproducible reports and manuscripts. We present a Quarto template and demonstrate how this template may be used for a journal, the *Journal of Fish and Wildlife Management*, in our article. This template may also be readily adapted to other journals that use Microsoft Word-based workflows and for other product types such as annual reports. We also provide a high-level overview of Quarto and other Markdown-based workflows as part of the document. Lastly, we provide examples of some features of the Quarto publishing system that may be helpful for authors when customizing Quarto template for specific journal formatting requirements and other product types.

**Keywords:** Findable, Accessible, Interoperable, and Reuseable (FAIR); markdown; reproducible science; journal template; Quarto

# Introduction

Modern computing tools allow scientists and practitioners to be efficient, transparent, and provide reproducible results (Erickson et al. 2021; Braga et al. 2023). For example, ecologists are moving away from point-and-click statistical methods and using scripting languages with code (Borregaard and Hart 2016). These changes not only make scientists more productive but also confer additional benefits. First, scripting allows methods to be reused by their creator. For example, scientists commonly complete the same or similar analyses on a regular basis (such as U.S. Fish and Wildlife Service 2023a), and scripting allows methods to be easily adapted and often quickly rerun with minimal formatting. Second, sharing the scripts allows others to reuse methods and recreate results when data are shared. Third, scripting provides direct evidence about what options were, or were not, selected for an analysis and data cleaning. All three of these features increase transparency in the scientific process.

The broader scientific community increasingly has begun to understand the importance of reproducible results because the perception of a “reproducibility crisis” exists among most of the scientists surveyed across scientific fields (Baker 2016; Fanelli 2018). This lack of reproducibility also exists in natural resource fields. For example, ArchMiller et al. (2020) determined only 17% of studies randomly selected from the *Journal of Wildlife Management* and the *Wildlife Society Bulletin* could have their analyses reproduced. In hydrology, lack of reproducibility has raised questions of the validity of the field as a whole (Hutton et al. 2016) and has motivated exploration of challenges from other fields (Fienen and Bakker 2016). Thus, scientists increasingly seek to produce reproducible results, especially for computationally intensive projects (Borregaard and Hart 2016).

Others have provided suggestions and methods for creating reproducible results in natural resource and related fields (such as Borregaard and Hart 2016; Braga et al. 2023). Additionally, Findable, Accessible, Interoperable, and Reusable (FAIR) principles exist to help scientists share and produce reproducible science (for example, Reiser et al. 2018). Likewise, professional societies and government agencies have produced guidelines similar to FAIR for reproducible science (for example, refer to Table 1 in Erickson et al. (2021) that lists example organizations).

Many scientists use scripting languages like Python or R for their statistical methods and analyses. For example, Erickson and Rattner (2020) determined that a plurality of authors in the journal *Environmental Toxicology and Chemistry* from 2019 used the R computing language for analysis in their papers. However, we have observed that fewer scientists use similar tools in a reproducible “research workflow” for writing manuscripts even though scripting environments are now enriched with code-based manuscript services, such as Quarto, knitr, and Bookdown, which are primarily Markdown-based programs (Baumer and Udwin 2015; Figure 1). Advantages of Markdown-based programs over direct typesetting products like LaTeX, Word, or Pages include simplicity, readability, transparency, and embedded computation (Baumer and Udwin 2015). Specifically, code-based manuscript writing can include the option to readily incorporate formatting from templates, ease of changing templates and reference formatting, automatic populating of references based upon Digital Object Identifiers (DOIs), and the ability to easily handle complex formulas (specifically embedded LaTeX syntax for formulas). Furthermore, these writing tools can be especially powerful when embedded with the code and data used in the document, which allows authors to create reproducible manuscripts that directly populate in-text tables and figures.

we have observed interest (circa 2021-2023) in Markdown-based writing documents from our colleagues and collaborators who are natural resource managers or scientists supporting these managers, in North America and globally. For example, the U.S. Geological Survey has a Markdown Community of Practice, and the authors of this paper have presented tutorials to this group. Additionally, this group includes members from the U.S. Fish and Wildlife Service and other agencies in the U.S. Department of the Interior. We have also discussed using Markdown-based writing tools in online conversations with others such as the U.S. Fish and Wildlife Service’s “Great Lakes R Working Group.” These examples motivated our creation of this paper and corresponding template.

For readers who are unfamiliar with Markdown-based writings tools, we provide a brief history here. Stanford University computer scientist Donald Knuth had trouble typesetting his mathematical equations in the late 1970s and early 1980s and this led him to create the TeX language (Knuth 1984). Building upon TeX, Leslie Lamport created LaTeX as a set of macros to make TeX easier to use (Lamport 1985). Statisticians who used LaTeX sought to readily embed statistical code, results, and figures with LaTeX documents. Thus, to meet this need, the program Sweave was created to “weave” S into LaTeX (at the time, many statisticians used the S language, which inspired the R language; Leisch 2002).

Although useful, we have observed that TeX, LaTeX, and Sweave can be be complicated to use, and Sweave can be especially cumbersome because it requires the source file to be compiled multiple times for a single output. TeX, LaTeX, and Sweave also may be difficult for natural resource scientists to learn, because they require new software and languages outside of the more common R or Python environments. Furthermore, in the four and half decades since Tex was created and four decades since LaTeX was created, Markdown usage became ubiquitous through its integration with version control tools such as Git and common Microsoft Office tools such as Teams and Word, which include many common Markdown shortcuts for formatting text (for example, typing \* then space on a new line in either program will start a bulleted list, which is syntax from the Markdown language). Others noted these difficulties as well; thus, Yihui Xie and others created a Markdown-based program for use with R, known as R Markdown (Xie et al. 2018), that overcomes many of these limitations. R Markdown works by using the knitr package to “knit” (or compile) the R Markdown into an output such as a Hyptertext Markup Language files, ending with .html; Microsoft Word Document, ending with .docx; or Portable Document Format, ending with .pdf (Xie 2014). Another Markdown offshoot is the ROxygen2 language, which allows users in the R environment to directly convert R script files (ending with .R rather than R Markdown files ending with .Rmd) into R Markdown, which is then “knit” to output formats mentioned in the previous sentence (Wickham et al. 2022). The broader R community received R Markdown well (such as reviews by Baumer and Udwin 2015), but this language also has some limitations. Most notably, the language requires R and has some minor quirks. The popularity and limitations of R Markdown led a group including Yihui Xie to create a more general program, Quarto, that works natively with many languages rather than only R (Allaire et al. 2022).

The remaining purpose of our paper is to present our Quarto template for this journal, which can be fully accessed at Erickson et al. (2024). We demonstrate some useful features including how to include figures, tables, accessibility components, and the quarto-utils package for automatic bibliography generation (Fienen and Erickson 2024). We also discuss how these tools may be applied to other situations. Although our examples tend to focus on R, Quarto works with many languages including LaTeX (for example, to typeset equations and format outputs), Python, and Julia (Figure 1).

# Methods

## Quarto workflow

This tutorial focuses on Microsoft Word document outputs because many journals, including the *Journal of Fish and Wildlife Management*, use a Word-based workflow. The Quarto documentation provides a generic journal format (https://quarto.org/docs/journals/, Accessed January 2024), and a Quarto journals GitHub repository, hosts specific templates for other journals (https://github.com/quarto-journals/, Accessed January 2024). When writing a Quarto document, the first step is to create a Quarto file that ends with .qmd extensions. Common editors that work with Quarto files include Posit RStudio (v2023.06 or newer; Posit Team 2023) and Microsoft Visual Studio Code (Microsoft 2023). RStudio also offers the benefits of creating new Quarto files from templates for journal articles, presentations, and other types of documents directly within the editor. Additionally, Quarto contains tools for generating empty templates outside of RStudio using command line tools for editing in programs such as Visual Studio Code or similar editor programs (refer to the program documentation such as https://quarto.org/docs/journals/templates.html for addition details of these features, accessed December 2023). Alternatively, readers may use an existing Quarto file and edit that file. For example, people can edit the Quarto file used to generate this paper that is hosted at Erickson et al. (2024) to write their own manuscripts for journal articles or other product types such as annual reports.

Within the Quarto file, the header contains metadata in a YAML-style format. The Quarto file for this article contains the following metadata:

* The title listing the article’s title.
* The author listing the article’s author or authors.
* The date the Quarto file was last rendered to create the Word file.
* The output format options including the output type of docx, an option to set the figure caption locations (fig-cap-location) to the bottom, an option to keep the LaTeX output files (keep-tex: true), and the Word file used as the style template for the file (reference-doc: jfwm\_template\_template.docx).
* The csl file listing article’s citation style; for the *Journal of Fish and Wildlife Management*, this is the “Council of Science Editors, author-date format indents” with minor modifications by the authors of this paper for the *Journal of Fish and Wildlife Management* (council-of-science-editors-author-date-jfwm.csl). “Council of Science Editors, author-date format indents” refers to the “Council of Science Editors” formatting style using the Council of Science Editors style with authors and dates (such as “example fact *Author et al. 2023*” and in contrast to other formats numeric such as “example fact [1]”) and indented reference section paragraphs.
* The name and location of the reference file (bibliography: references.bib).
* Code to include LaTeX-based headers (header-includes:) for double spacing (\usepackage{setspace}\doublespacing) and line numbers (\usepackage{lineno} \linenumbers).
* Editor options (editor\_options) telling the Markdown editor (markdown) to use a soft line wrap after each sentence (wrap: sentence).
* A list of “no citation” references (nocite) tells the Markdown to include citation items even if the item is not cited in the document. We include this because we had to manually format some inline citations to match the journal’s style.

After the header, the user writes their manuscript using Markdown code with optional LaTeX code for equations. Code can be embedded within “chunks” that are defined by their language (such as the lower case r for R or python for Python). These code chunks can either be displayed or hidden in the final document. References are added to the BibTeX file, either manually or using a script described later by pulling DOIs. Figures and tables can be added as needed, whether built locally with code chunks or called in from other sources. While writing, the author can preview the results by compiling the file to create a Word document. This also serves as a check for code mistakes that might prevent the Quarto file from compiling. The end result is a Word file that may be submitted to the a journal with minimal manual formatting required.

## Background knowledge, tips, and other Quarto observations

Writing Quarto documents requires a basic knowledge of Markdown. The Quarto documentation (available at <https://quarto.org/>, accessed September 2023) contains tutorials to help people get started. Additionally, the Quarto document (Erickson et al. 2024) used to generate this document contains many examples for those who like to dive in and learn by doing. Writing equations requires an understanding of some of the basics of LaTeX but much less than writing entire documents in LaTeX (many online tutorials exist such as the TeX User Group’s, http://tug.ctan.org/info/short-math-guide/short-math-guide.pdf, accessed December 2023).

As a general observation, when Quarto documents are rendered to Word documents, the Pandoc software (Dominici 2014) that renders the document is limited in formatting and style options compared to Hypertext Markup Language or Portable Document Format files because of technical reasons beyond the scope of this paper. Hence, readers may find some solutions using search engines that will not work for this Word-based template. For example, cross-references for figures and tables do not work with this template. Likewise, we had to use nocite and manually type in some references due to formatting challenges arising from certain characters such as semicolons. Additionally, readers familiar with Quarto may find some quirky examples in the Quarto template to help meet the journal specific formatting style. For example, the authors’ names are all listed on one line rather than using multiple entries as one might expect in the metadata section.

Integrating code into Quarto documents also helps to create reproducible results. Code may be included directly within the file, or R script files may be called into the Quarto file and “knit” into the final outputs using the read\_chunk() function from the knitr package (Xie 2014). For example, a simple regression might be run or data plotted and then the outputs included in the file. Including the code ensures the manuscript always has the most recent results and statistics if the input files are changed. Additionally, code outputs can be included and displayed inline with R (this is currently under development with Python, as noted in <https://github.com/quarto-dev/quarto-cli/pull/6190>, accessed September 2023). For example, when describing a study or writing a monitoring report, one might write “We observed many birds (*n* = 4,341) at the refuge with a mean of 184.7 birds per day.” With this example, inline code would allow these numbers to updated dynamically if the source data or code changed when the file is rerendered. Hence, if the authors updated their input numbers in either the source data files or code, the numbers in text would change. This change may occur because the authors updated their data while writing the manuscript, or because the authors update their report on a regularly basis (such as a report for an annual monitoring program).

Another tip for using Quarto to render Word documents is to use the flextable package in R (Gohel and Skintzos 2023) for creating tables rather than the popular kable() function from the knitr package (Xie 2014) and kableExtra package (Zhu 2021) because the flextable package more readily allows tables to be formatted in Word-based outputs. For example, we demonstrate how the flextable package may be used to create simple tables in Table 1 and tables with multiple columns and rows in Table 2. This table has multicolumn headers and automatically drops duplicate row names. These two aesthetic features are commonly used by scientists when describing data and can be automatically created using the flextable package in R. These features exist with kable-type tables, but do not render correctly in Word document outputs.

Finally, to edit the text style, change the jfwm\_template.docx file using Word rather than directly editing the output rendered from Quarto code. This may require practitioners to become familiar with Word templates. For example, in the *Journal of Fish and Wildlife Management*, the first paragraph of every section lacks indentation, but the second and subsequent paragraphs in a section use a tab indent. Changing this formatting requires changing the paragraph template settings in the template Word document. Further modifications can be made to change the template by updating the Styles definitions (such as “Header 1” or “Body Text”) in the template Word document. These changes would then be shown on the output Quarto-based Word document after the template Word document is saved and the Quarto (.qmd) file is re-rendered to create a Word document.

## Tips to improve accessibility

Using Quarto as a templating workflow does not necessarily guarantee that the resulting rendered document will be 508-compliant or accessible; however, there are some additional guidelines that will improve overall accessibility:

1. Use the built-in header system (e.g., # for Header 1, ## for Header 2, etc.) to create screen-readable organization in the rendered document.
2. Add “alt-text” to your figure elements. Alt-text is text added to provide a visual description of the figure or image and is read by screenreaders. If you are using a chunk of code to create the figure, you can include a chunk argument fig.alt="..." that will automatically create alt-text for that output. Alternatively, if you are directly adding a figure to your document, you can include alt-text within the square brackets: ![Alt text description](image\_pathway.png)

## quarto-utils

The quarto-utils software is a Python package that assists Quarto authors in compiling and formatting references. By leveraging the https://doi.org web interface, authors are able to place references in their documents by only providing a DOI in place of a reference in the text. Then using functionality in quarto-utils, the code will search the .qmd document for DOI codes, retrieve a BibTeX representation of the reference from doi.org, add the BibTeX string to a bibliography file, and convert the DOI reference in the .qmd file to a valid reference call. The author identifies a reference in the same way they would call out an existing reference using [] delimiters, but rather than providing a key that references the .bib file (such as [@Prince1999]), they would use either \_doi:thedoinumber or \\_doi:thedoinumber in its place. The quarto-utils code searches for \_doi: or \\_doi: and parses for the DOI. The DOI can take the form of just the number (such as 10.1999/prince.albums) or the full uniform resource locator (such as https://doi.org/10.1999/prince.albums). If invalid DOI codes are provided, quarto-utils will report that it was not able to find a reference for a given string and will leave the .qmd file unchanged. This feature is another method to streamline the writing process. Many bibliographic database software programs exist, but authors may find it more efficient to simply search the internet for a reference they want to cite, copy the DOI, paste it into their .qmd file, and keep writing. The BibTeX file generated by quarto-utils may require manual formatting, and occasionally, automatically generated outputs prevent Quarto from rendering files until special characters (such as non-American Standard Code for Information Interchange) are removed. Also, citations listed on doi.org may be incorrect and sometimes require addition curation and formatting.

# Results

We have demonstrated how to use Quarto to prepare manuscripts for journals that use Word-based workflows, such the *Journal of Fish and Wildlife Management* or the *Journal of Wildlife Management*, and do not include LaTeX-based workflows as submission options. Our example paper included figures (such as Figure 1) and tables (such as Table 1 and Table 2). We have also described the coauthors’ connections to the Upper Midwest (Table 2). These figures and tables to demonstrated how to construct a table with formatting changes from default settings. The methods of this paper provided tips and suggestions for using Quarto to format documents to match a specific journal’s style. Last, we presented use a Python package, quarto-utils (Fienen and Erickson 2024) for helping author automate their reference management.

# Discussion

Script-based writing programs such as Quarto offer advantages compared to traditional word processing programs by allowing writers to focus on writing rather than formatting (Baumer and Udwin 2015). For example, scientists can readily change formatting if they need to change reports (such as completion reports or partner summaries) to journal articles or change formatting across journal articles because only style files need to be changed rather than reformatting the entire document. Additionally, reports that are completed and rerun on a regular basis can be formatted to automatically include numbers in the text to avoid tedious and error prone copying and pasting of values such as the examples previously shown in this document where numbers are reinserted each time the source file is recompiled. When integrated with statistical programs such as Python or R and publicly released, files created by these programs also more readily support FAIR principles because the details (and code) necessary to recreate results are embedded within the Quarto document. The FAIR principles are increasingly being used by science agencies for conducting and documenting research and monitoring programs. For example, the U.S. Geological Survey includes FAIR principles as part of their road map for science (Lightsom et al. 2022) and U.S. Fish and Wildlife Service planning documents also includes FAIR principles (U.S. Fish and Wildlife Service 2023b).

Thus, we created an example of how to prepare manuscripts in a Quarto document for following the style guide for the *Journal of Fish and Wildlife Management*. Our example could readily be adapted for other journals with similar Word-based workflows such as *Environmental Toxicology and Chemistry* or the *Journal of Wildlife Management*. Aside from directly helping authors who write journal articles or other products such as annual reports, we also hope our publication will raise awareness for fish and wildlife managers about this tool. For example, many reports that are published weekly, monthly, quarterly, annually, or on a similar interval would benefit from not only using a computer script to perform the statistical analysis, data formatting, plotting, or other number crunching but also in preparing the document itself. The example described in this paper and fully provided in Erickson et al. (2024) could also serve as a starting place for these documents.

# Conflict of Interest

The authors report no conflicts of interest.

# Acknowledgments

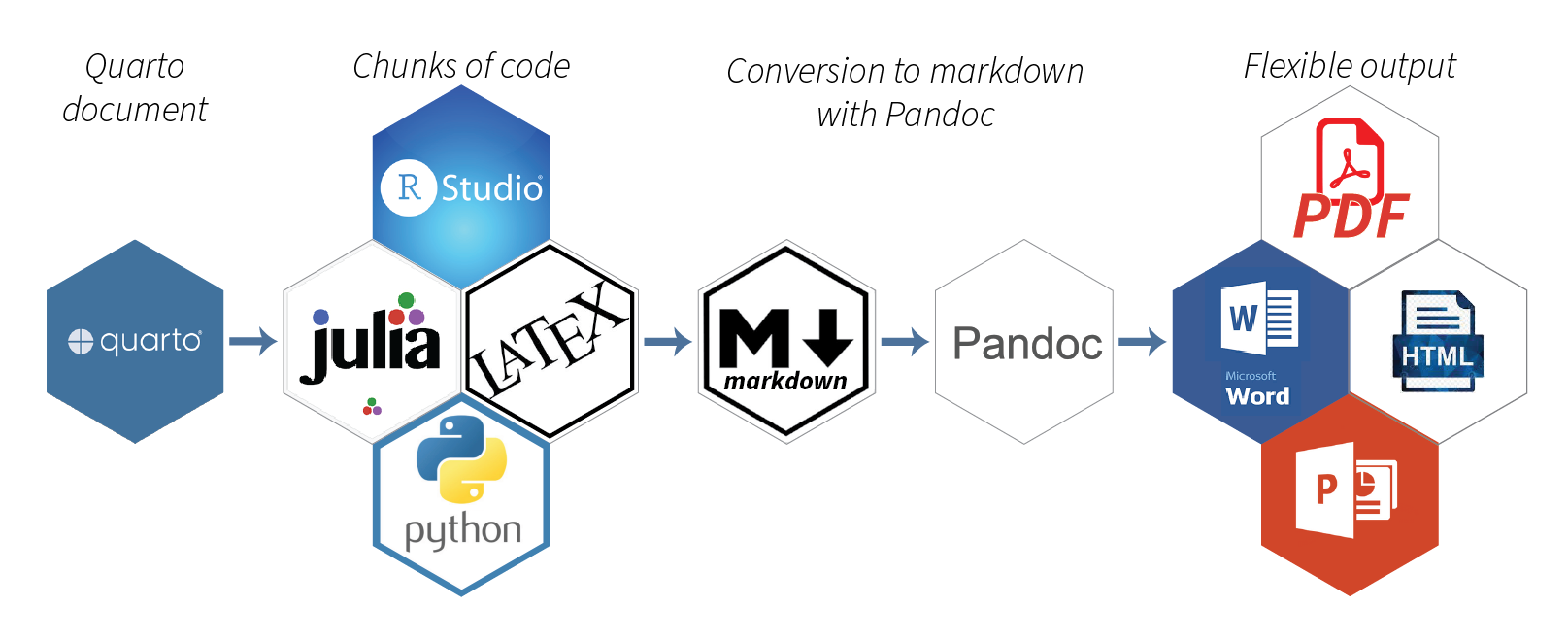
We thank all reviewers and editors, both the journals and internal U.S. Geological Survey, for their feedback that improved this manuscript. We thank the U.S. Geological Survey Biological Threats and Invasive Species Research Program and U.S. Geological Survey Water Resources Mission Area Integrated Information Dissemination Division for funding. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. The findings and opinions expressed in this manuscript are those of the authors and do not necessarily represent the views of the U.S. Fish and Wildlife Service.

# Data Availability

The template code for the Quarto document used to create this article is available in Erickson et al. (2024). The quarto-utils package is available at Fienen and Erickson (2024).

# Figures

Figure 1. Example of Quarto workflow. Within the Quarto document, users can add chunks of code in the R, Julia, or Python languages. In addition, users can directly embed LaTeX typesetting, which is often used for equations. Quarto then renders the document to an intermediate Markdown document with Pandoc. Pandoc then converts the Markdown document directly to one of the many output formats. These formats include such products as Portable Document Format (PDF), Microsoft Word, Microsoft PowerPoint, or Hypertext Markup Language (html).



# Tables

Table 1. Steps to use the Quarto template associated with this manuscript for a journal such as the *Journal of Fish and Wildlife Management*.

| order | step |
| --- | --- |
| 1 | Obtain template |
| 2 | Write text using Markdown |
| 3 (optional) | Run Python script to populate bibliography file |
| 4 | Render document to Word |
| 5 (optional) | Break Quarto-Word link and do final formatting manually |
| 6 | Submit to journal |

Table 2. Author trivia to demonstrate a table. The table awkwardly includes wide and long form data to demonstrate how to use the flextable package.

|  | Minnesota Connection | | Wisconsin Academic Connection |
| --- | --- | --- | --- |
| Author | Internship | Undergraduate | University of Wisconsin (UW) System |
| RAE | University of Minnesota | -- | Undergraduate, UW-Stevens Point |
| -- | -- | Internship, UW-Madison |
| -- | -- | Graduate faculty, UW-La Crosse |
| AAA | University of Minnesota | Gustavus Adolphus College | Graduate Certificate, UW-Madison |
| MNF | -- | Macalester College | Graduate faculty, UW-Madison |

# Supplemental Material

**Supplement S1** Lightsom FL, Hutchison VB, Bishop B, Debrewer LM, Govoni DL, Latysh N, Stall S. 2022. Opportunities to improve alignment with the FAIR principles for U.S. Geological Survey data. U.S. Geological Survey Open File Report 2022-1043. U.S. Geological Survey, Reston, VA. doi:10.3133/ofr20221043. **Supplement S2** U.S. Fish and Wildlife Service. 2023a. Budget justifications and performance information fiscal year 2024. U.S. Department of the Interior, Washington, D.C. USA. **Supplement S3** U.S. Fish and Wildlife Service. 2023b. Waterfowl population status, 2023. U.S. Department of the Interior, Washington, D.C. USA.

# References

Allaire JJ, Teague C, Xie Y, Dervieux C. 2022. Quarto. doi:[10.5281/ZENODO.5960048](https://doi.org/10.5281/ZENODO.5960048).

ArchMiller AA, Johnson AD, Nolan J, Edwards M, Elliott LH, Ferguson JM, Iannarilli F, Vélez J, Vitense K, Johnson DH, et al. 2020. Computational reproducibility in The Wildlife Society’s flagship journals. Journal of Wildlife Management. 84(5):1012–1017. doi:[10.1002/jwmg.21855](https://doi.org/10.1002/jwmg.21855).

Baker M. 2016. 1,500 scientists lift the lid on reproducibility. Nature. 533(7604):452–454. doi:[10.1038/533452a](https://doi.org/10.1038/533452a).

Baumer B, Udwin D. 2015. R Markdown. Wiley Interdisciplinary Reviews: Computational Statistics. 7(3):167–177. doi:[10.1002/wics.1348](https://doi.org/10.1002/wics.1348).

Borregaard M, Hart E. 2016. Towards a more reproducible ecology. Ecography. 39(4). doi:[10.1111/ecog.02493](https://doi.org/10.1111/ecog.02493).

Braga PHP, Hébert K, Hudgins EJ, Scott ER, Edwards BPM, Reyes LLS, Grainger MJ, Foroughirad V, Hillemann F, Binley AD, et al. 2023. Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution, journal = Methods in Ecology and Evolution. 14(6):1364–1380. doi:[10.1111/2041-210x.14108](https://doi.org/10.1111/2041-210x.14108).

Dominici M. 2014. An overview of Pandoc. TUGboat. 35(1):44–50. <https://www.tug.org/TUGboat/tb35-1/tb109dominici.pdf>.

Erickson RA, Archer AA, Fienen MN. 2024. Journal of Fish and Wildlife Management Quarto template, U.S. Geological Survey software release. Reston, VA. [accessed January 2024]. <https://code.usgs.gov/umesc/quant-ecology/jfwm-quarto-template>.

Erickson RA, Burnett JL, Wiltermuth MT, Bulliner EA, Hsu L. 2021. Paths to computational fluency for natural resource educators, researchers, and managers. Natural Resource Modeling. 34(3). doi:[10.1111/nrm.12318](https://doi.org/10.1111/nrm.12318).

Erickson RA, Rattner BA. 2020. Moving beyond p 0.05 in ecotoxicology: A guide for practitioners. Environmental Toxicology and Chemistry. 39(9):1657–1669. doi:[10.1002/etc.4800](https://doi.org/10.1002/etc.4800).

Fanelli D. 2018. Is science really facing a reproducibility crisis, and do we need it to? Proceedings of the National Academy of Sciences. 115(11):2628–2631.

Fienen MN, Bakker M. 2016. HESS opinions: Repeatable research: What hydrologists can learn from the duke cancer research scandal. Hydrology and Earth System Sciences. 20(9):3739–3743. doi:[10.5194/hess-20-3739-2016](https://doi.org/10.5194/hess-20-3739-2016).

Fienen MN, Erickson RA. 2024. Quarto-utis, U.S. Geological Survey software release. Reston, VA. [accessed January 2024]. <https://github.com/DOI-USGS/quarto-utils>.

Gohel D, Skintzos P. 2023. Flextable: Functions for tabular reporting. <https://CRAN.R-project.org/package=flextable>.

Hutton C, Wagener T, Freer J, Han D, Duffy C, Arheimer B. 2016. Most computational hydrology is not reproducible, so is it really science? Water Resources Research. 52(10):7548–7555. doi:[10.1002/2016wr019285](https://doi.org/10.1002/2016wr019285).

Knuth DE. 1984. TEX: The Program. Reading, MA: Addison–Wesley.

Lamport L. 1985. LaTeX–A Document Preparation System. Reading, MA: Addison–Wesley.

Leisch F. 2002. [Sweave: Dynamic generation of statistical reports using literate data analysis](https://doi.org/10.1007/978-3-642-57489-4_89). In: Compstat. Physica-Verlag HD. p. 575–580.

Lightsom FL, Hutchison VB, Bishop B, Debrewer LM, Govoni DL, Latysh N, Stall S. 2022. [Opportunities to improve alignment with the FAIR Principles for US Geological Survey data](https://doi.org/10.3133/ofr20221043). US Geological Survey.

Microsoft. 2023. Visual Studio Code. Seattle, WA. [accessed January 2024]. <https://code.visualstudio.com/>.

Posit Team. 2023. RStudio: Integrated Development Environment for R. Boston, MA: Posit Software, PBC. <http://www.posit.co/>.

Reiser L, Harper L, Freeling M, Han B, Luan S. 2018. FAIR: A call to make published data more findable, accessible, interoperable, and reusable. Molecular Plant. 11(9):1105–1108. doi:[10.1016/j.molp.2018.07.005](https://doi.org/10.1016/j.molp.2018.07.005).

U.S. Fish and Wildlife Service. 2023b. Budget justifications and performance information fiscal year 2024. U.S. Department of the Interior, Washington, D.C. USA. [accessed January 2024]. <https://www.fws.gov/media/fiscal-year-2024-fish-and-wildlife-service-budget-justification>.

U.S. Fish and Wildlife Service. 2023a. Waterfowl population status, 2023. U.S. Department of the Interior, Washington, D.C. USA. [accessed January 2024]. <https://www.fws.gov/media/waterfowl-population-status-2023>.

Wickham H, Danenberg P, Csárdi G, Eugster M. 2022. roxygen2: In-Line Documentation for R. <https://CRAN.R-project.org/package=roxygen2>.

Xie Y. 2014. knitr: A Comprehensive Tool for Reproducible Research in R. In: Stodden V, Leisch F, Peng RD, editors. Implementing Reproducible Computational Research. Chapman; Hall/CRC. <https://yihui.org/knitr/>.

Xie Y, Allaire JJ, Grolemund G. 2018. [R Markdown](https://doi.org/10.1201/9781138359444). Chapman; Hall/CRC.

Zhu H. 2021. kableExtra: Construct Complex Table with ’kable’ and Pipe Syntax. <https://CRAN.R-project.org/package=kableExtra>.