


The logo for Peer Community In Archaeology features a circular arrangement of various archaeological icons such as a microscope, a skull, a spear, a map, and a compass, all rendered in a dark, sketchy style. The text 'Peer Community In Archaeology' is positioned to the right of the icons, with 'Peer Community In' in a smaller font and 'Archaeology' in a larger, bold font. The entire graphic is set against a background of light blue dots and lines.

# Peer Community In Archaeology

## Sharing Research Code in Archaeology

**James Allison**  based on peer reviews by **Thomas Rose, Joe Roe** and 1 anonymous reviewer

Sophie C Schmidt; Simon Maddison (2023) Percolation Package - From script sharing to package publication. Zenodo, ver. 3, peer-reviewed and recommended by Peer Community in Archaeology. <https://doi.org/10.5281/zenodo.7966497>

Submitted: 24 May 2023, Recommended: 23 November 2023

### Cite this recommendation as:

Allison, J. (2023) Sharing Research Code in Archaeology. *Peer Community in Archaeology*, 100333. [10.24072/pci.archaeo.100333](https://doi.org/10.24072/pci.archaeo.100333)

Published: 23 November 2023

Copyright: This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

---

The paper “Percolation Package – From Script Sharing to Package Publication” by Sophie C. Schmidt and Simon Maddison (2023) describes the development of an R package designed to apply Percolation Analysis to archaeological spatial data. In an earlier publication, Maddison and Schmidt (2020) describe Percolation Analysis and provide case studies that demonstrate its usefulness at different spatial scales. In the current paper, the authors use their experience collaborating to develop the R package as part of a broader argument for the importance of code sharing to the research process.

The paper begins by describing the development process of the R package, beginning with borrowing code from a geographer, refining it to fit archaeological case studies, and then collaborating to further refine and systematize the code into an R package that is more easily reusable by other researchers. As the review by Joe Roe noted, a strength of the paper is “presenting the development process as it actually happens rather than in an idealized form.” The authors also include a section about the lessons learned from their experience.

Moving on from the anecdotal data of their own experience, the authors also explore code sharing practices in archaeology by briefly examining two datasets. One dataset comes from “open-archaeo” (<https://open-archaeo.info/>), an on-line list of open-source archaeological software maintained by Zack Batist. The other dataset includes articles published between 2018 and 2023 in the Journal of Computer Applications in Archaeology. Schmidt and Maddison find that these two datasets provide contrasting views of code sharing in archaeology: many of the resources in the open-archaeo list are housed on Github, lack persistent object identifiers, and many are not easily findable (other than through the open-archaeo list). Research software attached to the published articles, on the other hand, is more easily findable either as a supplement to the published article, or in a repository with a DOI.

The examination of code sharing in archaeology through these two datasets is preliminary and incomplete, but it does show that further research into archaeologists’ code-writing and code-sharing practices could be useful. Archaeologists often create software tools to facilitate their research, but how often? How often is

research software shared with published articles? How much attention is given to documentation or making the software usable for other researchers? What are best (or good) practices for sharing code to make it findable and usable? Schmidt and Maddison's paper provides partial answers to these questions, but a more thorough study of code sharing in archaeology would be useful. Differences among journals in how often they publish articles with shared code, or the effects of age, gender, nationality, or context of employment on attitudes toward code sharing seem like obvious factors for a future study to consider.

Shared code that is easy to find and easy to use benefits the researchers who adopt code written by others, but code authors also have much to gain by sharing. Properly shared code becomes a citable research product, and the act of code sharing can lead to productive research collaborations, as Schmidt and Maddison describe from their own experience. The strength of this paper is the attention it brings to current code-sharing practices in archaeology. I hope the paper will also help improve code sharing in archaeology by inspiring more archaeologists to share their research code so other researchers can find and use (and cite) it.

### **References:**

Maddison, M.S. and Schmidt, S.C. (2020). Percolation Analysis – Archaeological Applications at Widely Different Spatial Scales. *Journal of Computer Applications in Archaeology*, 3(1), p.269–287.

<https://doi.org/10.5334/jcaa.54>

Schmidt, S. C., and Maddison, M. S. (2023). Percolation Package - From script sharing to package publication, Zenodo, 7966497, ver. 3 peer-reviewed and recommended by Peer Community in Archaeology. <https://doi.org/10.5281/zenodo.7966497>

## **Reviews**

### **Evaluation round #1**

DOI or URL of the preprint: <https://doi.org/10.5281/zenodo.7966498>

Version of the preprint: 1

### **Authors' reply, 25 September 2023**

We have revised the paper taking on board all of the comments and feedback from the reviewers. These were really useful, and we believe that we have addressed all of them.

### **Decision by [James Allison](#) , posted 26 June 2023, validated 27 June 2023**

#### **Suggested revisions to Percolation Package - From script sharing to package publication**

This article combines a case study of the development of an R package implementing Percolation Analysis with general observations about the importance of code sharing in archaeology. The case study details a process beginning with borrowing code from researchers outside archaeology, refining it to fit archaeological case studies, and collaboration between the authors to further refine and systematize the code into an R package that is more easily reusable by other researchers.

All three reviewers have generally positive views of the paper, but they also offer some suggestions for revision. I agree that both the case study and the general argument are important, but also agree with many of the suggestions for revision. The manuscript would particularly benefit from better integration between the case study and the general discussion of code sharing. As Rose's review asks, "what are the lessons learned

from the case study...”? The article would benefit from making the connections between the sections more explicit.

I also agree with the suggestion that the authors should reconsider submitting the package to CRAN, although that would probably require considerable effort, and I would not see it as a necessary revision to make the paper publishable. The reviewers raise a number of other issues that I do not need to repeat, but most of the suggested revisions are minor and should be easy to accommodate.

I encourage the authors to consider the reviewers’ comments and to revise the paper accordingly.

### Reviewed by anonymous reviewer 1, 20 June 2023

The topic of the paper is very interesting and up to date. In this article the authors deal with issues that are fundamental to share scripts, algorithms and packages that can be useful for different researches, and thus to make different research experiences comparable with each other and repeatable.

The analysis proposed and the observations developed on its basis are interesting and a useful starting point to begin a true organisation of sharing quantitative investigations tools of many types.

Maybe for a future contribution it would be interesting to have statistics about the developments of the tendency in sharing code through the years, in order to understand or hypothesise which policy is more fruitful for this. Sharing code can be one of the bases for a true dialogue among researchers.

Regarding the form, I feel that the first paragraphs (“Inception” and “Collaboration”) could be made clearer and more fluent. In particular, this could be accomplished through the introduction of elements that tie together the sentences, which at the moment seem rather juxtaposed paratactically.

Lastly, one of the main issues for code sharing is properly stressed in the article: that is, the long term support of platforms for sharing. Github seems to show some important weaknesses, from this point of view, but it also seems to be the most used platform. A proposal (maybe in the “Conclusions” paragraph) to solve this issue would be useful and fit this paper.

### Reviewed by Joe Roe, 07 June 2023

This is an innovative paper describing the process of developing an R package from the inception of an idea to publication. It is refreshingly brief and readable, and I appreciate the authors openness in presenting the development process as it actually happens rather than in an idealised form. They neatly shows that the package format is a natural progression of more informal means of code-sharing—not something to be scared of—and I share the hope that this will encourage colleagues to set their code free.

There is just one point on which I have reservations – that the package is not on CRAN. I’m aware that there is a debate within the R community about whether CRAN’s submission requirements processes are too strict, but the fact is that it remains the accepted standard repository for R packages; I don’t think most people would call a package “published” until it’s on CRAN. Both leading journals in the R development world—the *Journal of Statistical Software* (<https://www.jstatsoft.org/authors>) and the *R Journal* (<https://journal.r-project.org/>)-require that packages are on CRAN or Bioconductor before a description can be published. The submission requirements and automated quality checks, while sometimes annoying for the developer, are there so that *users* can count on packages being reliable, well-documented and not likely to stop working in future versions. I think that the authors would agree that this is very important for scientific software.

Similarly, I can’t agree that “CRAN is not necessary for the easy use of a package”. The existence of a single comprehensive repository (CRAN) with an easy user interface (`install.packages()`) is often cited as one of the main reasons R has become so widely used in academia,<sup>[1][2][3]</sup> especially when compared to alternatives like Python, whose dependency management is more oriented towards the needs of software engineers. In my experience of teaching R, not being able to find and install a package the “normal way” is a *major* obstacle for newer users, no matter how clearly you try to point to the existence of `remotes::install_github()`. It also

makes it harder to use the package in an analysis formatted as a package-based compendium (an approach popularised in no small part thanks to one of the authors!), because it cannot be listed as a formal dependency. These issues seem especially relevant given that the authors themselves (rightly) emphasise that scientific software must be “findable and accessible” if it is to be useful. Like it or not, in the current R world, findable and accessible means CRAN.

Honestly, in my experience, the CRAN submission process is not nearly as bad as is often made out, especially if you use the scaffolding provided by the devtools and usethis packages. I would strongly encourage the authors to at least try and, if they think it could be useful, I'd be happy to review the package code for any potential issues beforehand. Otherwise, the authors might consider including their package in R-universe (<https://r-universe.dev>), a CRAN alternative that addresses some (but not all) of the above-mentioned problems.

Minor notes:

- The package is referred to inconsistently as both “the percolation package” and “percopackage”
- The authors write that they “publish[ed] the binary package on Github and OSF”, but unless I'm missing something neither hosts binary versions of the package, only the source and a tarball.
- The citation format is inconsistent
- URLs and DOIs are used inconsistently, sometimes with http://, sometimes without. The bibliography also mixes http and https for DOI links (https should be preferred).
- Is there a particular reason to cite the forthcoming second edition of Wickham & Bryan's *R Packages*, rather than the published first edition?
- I appreciate the citation to my own work with Zack Batist; could I perhaps ask you to include the full title and venue as given here? <https://github.com/zackbatist/openarchaeo-collaboration#citation>

References:

- [1] <https://journal.r-project.org/archive/2020/RJ-2020-028/index.html>
- [2] <https://www.nature.com/articles/517109a>
- [3] <https://academic.oup.com/jrssig/article/15/4/14/7029433>

## Reviewed by **Thomas Rose**, 24 June 2023

This short manuscript provides a case study of how sharing of code via email within a collaboration eventually turned into the development and publication of an R package. This account is followed by a brief general discussion on how code is shared in archaeology with a strong focus on <https://open-archaeo.info/> and a brief excursion to the FAIR principles. It concludes that publishing code might be time-consuming, especially regarding a proper documentation of the code, but overall speeds up the scientific progress because the code is easier to re-use and can be understood by users without the need to get in contact with the developer. Consequently, it recommends code publication over individual code sharing.

- The manuscript provides a welcome and important case study that hopefully incentivizes its readers and the community to rethink the current predominant practice of sharing code individually or as (almost) undocumented annexes to publications in favor for a proper publication of the code as such, with additional publications to promote it. However the manuscript requires significantly improvement in several aspects. The case study remains rather anecdotal and has little to no relevance for the further discussion. In addition, the relation between this part and the data compiled from <https://open-archaeo.info/> including the brief mentioning of the FAIR principles is not clear. Important publications about software publication and citation are missing. And while it might be evident for

readers from the field that the discussion focuses exclusively on code written in R, this is a too narrow approach, especially as the key message of the paper is independent from the programming language. Therefore, I recommend to revise the manuscript with respect to the following points:

- What are the lessons learned from the case study for similar processes? Which recommendations could be given to peers in a similar situation? This part focuses on the important step of turning scripts into something publishable and re-usable. I am sure the authors learned from this experience more than that proper documentation of code is worth the effort. Turning the lessons learned into some kind of recommendation to the community would significantly increase the value of the paper. Likewise, the discussion should be extended to include also this step and not only the question whether code should be published or not.
- Increase the linkage between the first part (“from code sharing to publication”) and the second part (“current code publication practices”) or restructure the manuscript. At the moment, both are standing rather on their own, with the first part contributing very little to the discussion. The same holds true for the turn from code sharing over the small literature study to the recommendations based on the FAIR principles: The common thread in the line of the arguments is difficult to recognize.
- Widen the wording of the manuscript to be inclusive of other languages than R. For example, rather than focusing exclusively on CRAN, PyPi and CRAN could be discussed together as repositories because PyPi is practically the Python-equivalent of CRAN. The overall message of the paper is independent from the programming language. This should be reflected in the wording.
- Potential further aspects to consider in the discussion:
  - Benefits for the developer of code by making it citable (e.g. getting cited)
  - Reproducibility and replicability: Published and properly cited code increases both without the need to including the code in every single publication that relies on it.
  - Importance of code quality: Having code published is nice but how to ensure as developer and assess a user its quality, i.e. that it is actually running and giving the correct output?
- What exactly are the hurdles and “costly tests” that made publication of the package on CRAN unattractive? Maintaining a CRAN-published package, I fail to understand the reservations of the authors here. The testing required in the CRAN policies should be done anyway to ensure compatibility of the package on a wide range of operating systems and ensure that the package runs stable. These processes and publication on CRAN itself is greatly facilitated by the devtools package, as detailed in “R Packages” by Wickham and Bryan (cited by the authors). In addition, publication on CRAN comes with the benefit of regular automated checks for compatibility with newer versions of R and of CRAN-published packages, the published package might depend on.  
Further, I was surprised that Zenodo is not mentioned. Especially the webhook from GitHub to Zenodo in combination with the automatic retrieval of metadata from the CFF file provides a fully automated way of software publication/archiving.
- The small study on how code is published in articles (“Sophie drew a sample of 30 research articles and case studies ...”) is lacking any foundation for the stated claims. Which papers were included? How were they chosen? How representative are 30 publications deemed for the field?
- The section regarding the FAIR principles must be rewritten, taking into account the recommendations of the FAIR4RS working group of the RDA (<https://www.nature.com/articles/s41597-022-01710-x>). In addition, these ideas are not original by the authors and hence should be properly cited.

- The conclusions invoke the impression that documentation is exclusively done for the sake of software publication. This needs to be rephrased. Proper documentation provides a massive benefit to the developer. Proper documentation (and testing) should be regarded as an essential part of the work. Otherwise, the code will become inaccessible also to the developer sooner or later.
- The software mentioned in the manuscript, including the R packages, must be properly cited. For the R packages, the respective citations can be gathered directly from them by citation("package"). For general guidance on how to cite software and the importance of proper software citations, see <https://doi.org/10.7717/peerj-cs.86> and <https://doi.org/10.12688/f1000research.26932.2>
- The authors switch from full name to first name when referring to themselves in the last paragraph before "Collaboration". Please be consistent.