Percolation Package - From script sharing to package publication

- Sophie C. Schmidt, Freie Universität Berlin, s.c.schmidt@fu-berlin.de ORCID: 0000-0003-4696-2101
- Simon Maddison, University College London, simon.maddison@ucl.ac.uk ORCID:0000-0001-5006-6604

Abstract

In this paper we trace the development of an R-package starting with the adaptation of code from a different field, via scripts shared between colleagues, to a published package that is being successfully used by researchers world-wide. Our aim is to encourage researchers to prepare their custom analysis code in a way that is useful and easily reusable for others.
The paper starts with a short background as to the scientific aims of the software package before relating the collaboration journey and the detailed processes involved. We then move to a review of code sharing in archaeology and the lessons we have learned in publishing code for others to use and share.

Introduction

In this paper we describe the development of a suite of R scripts for use within archaeology, building on work from a different domain. This was originally undertaken for a specific university research project, but the results were sufficiently interesting to warrant dissemination at conferences. It is an example of Research Software, which is discussed in more detail later. Presentation at a CAA international event stimulated a conversation between the authors, which led to code sharing. This in turn led to collaboration on extending the code's application scope, and to the decision to prepare an R-package for publication and open use.

To set the scene, we start with a brief description of the scientific aims of the code package. We then relate the journey of collaboration and the detailed processes involved, and in particular the goals in wishing to publish it in the form of a package. We then discuss the lessons that we have learned in going through this process.

We move on to review how code is currently shared in archaeology, drawing on the list open-archaeo and articles published in the Journal of Computer Applications in Archaeology. We present and discuss the results of this analysis within the context of good practices for sharing Research Software.

In conclusion we summarise all these lessons and evangelise our view that publication of Research Software for open use and sharing is a worthwhile and effective goal for others to follow.

The scientific aims of the package

Percolation Analysis is a well established process for identifying clusters amongst spatially distributed points. The percolation package (Maddison and Schmidt 2020), published as {percopackage}, evolved from work undertaken at UCL by Simon Maddison on the distribution of Hillforts. The aim was to find an approach to identify regional groupings, independent of geographical and political boundaries of any period. Percolation Analysis provided a conceptually simple technique and had recently been used in geography by Elsa Arcaute at UCL for identifying and classifying urban density through road junctions (Arcaute et al. 2016). This had been adopted within the Institute of Archaeology for identifying clusters of Domesday settlements in England by Stuart Brookes (Arcaute et al. forthcoming), and from that was taken up for the study of Hillfort distribution by Simon, see Fig. 1b. This work was applied to data from the Atlas of Hillforts in Britain and Ireland (https://hillforts.arch.ox.ac.uk/) which was being compiled at that time and was subsequently published online in 2017 (Lock and Ralston 2017, see Fig. 1a).

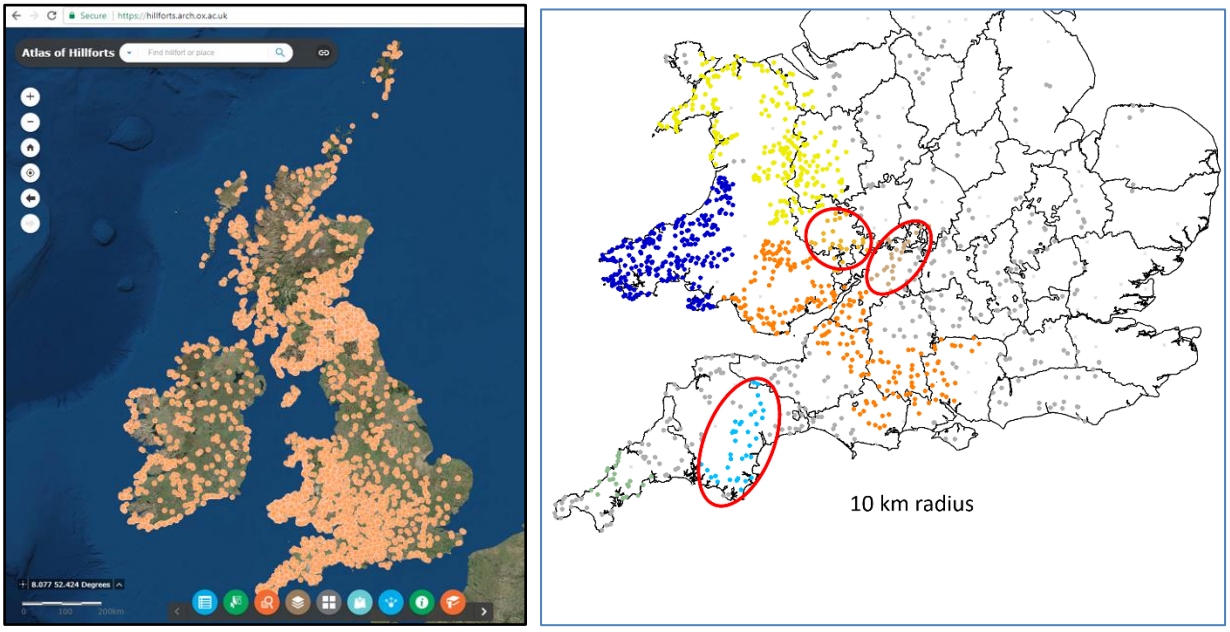

Fig. 1. a) screenshot of the online hillforts atlas (https://hillforts.arch.ox.ac.uk/), b) percolation results of hillforts at 10km radius, taken from Maddison (2016)

The geographical work was undertaken using a C program to efficiently manage the very large databases involved. Elsa kindly produced a simple program in R (R Core team 2023) that could be readily adapted and developed for the exploratory work on other, smaller archaeological datasets. This adapted program suite was written to be readily used by others and the work was presented at CAA, where it caught Sophie Schmidt's interest for a project she was working on.

Collaboration

This personal meeting of the co-authors at the CAA 2018 conference in Tübingen spurred the next phase in the development of the package. After discussing their relevant papers and

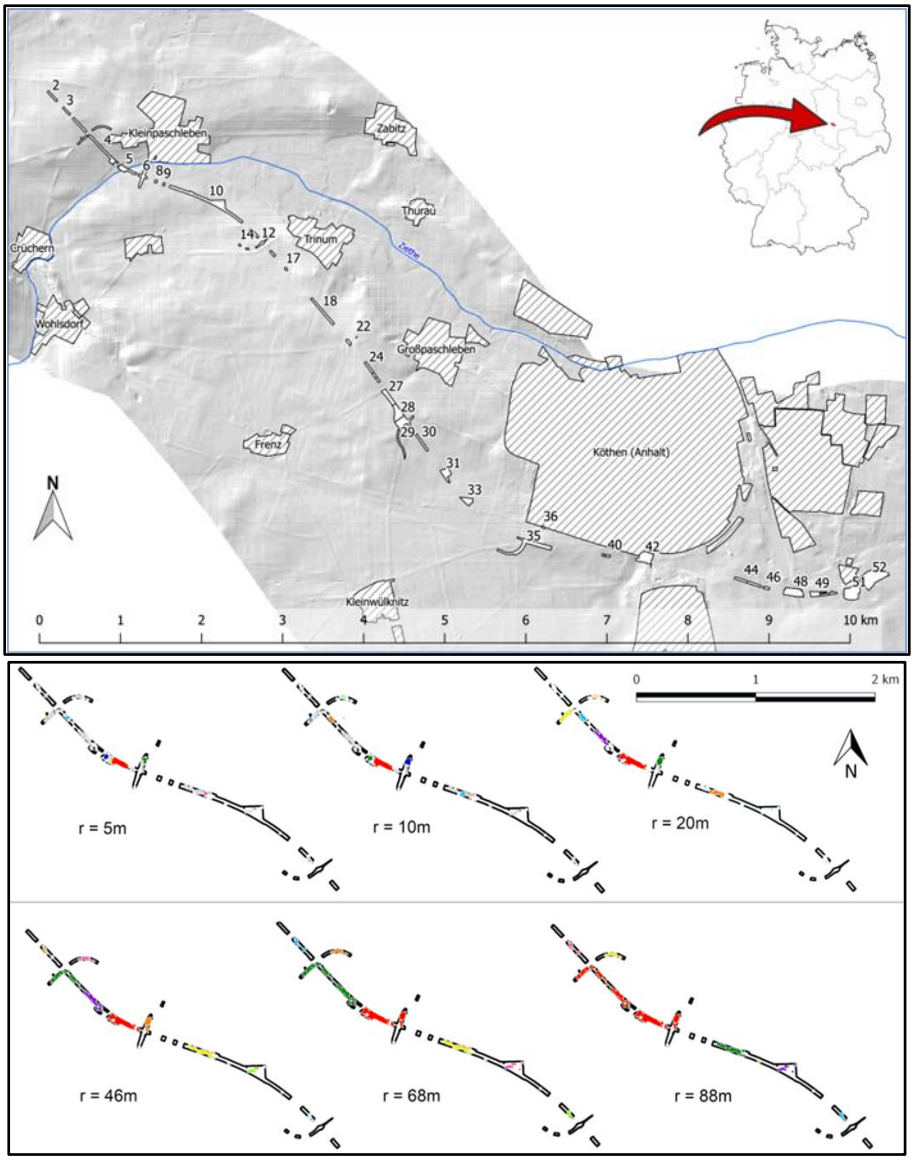

Fig 2. a) site of strip excavation of road B6n near Köthen (Maddison and Schmidt 2020, Fig. 11), b) overview of percolation results of features excavated along the transect at varying radii (Maddison and Schmidt 2020, Fig. 15)

research aims, Simon shared his four scripts with Sophie, who was able to reuse the code to analyse her own data concerning features along a strip excavation in Saxony-Anhalt, Germany (see Fig. 2).

At this point the code had some comments and a three-page description that enabled a basic understanding of the program within a couple of hours. In re-using the code on a dataset of a very different geographic scale (compare the scales of Fig. 1 and Fig. 2), some problems arose that needed to be addressed in order to extend the program's range of application. During the next two years Sophie successfully applied the code to her project work and a paper was prepared, in which the performance of the code applied to these rather different datasets and a comparison with different clustering methods was evaluated (Maddison and Schmidt 2020).

Detailed Process

The collaboration between the authors was organised rather traditionally, by emailing versions of the R scripts back and forth. On one side git and a private Github repository were used for version control. The package named ‘percopackage’ was created and the scripts converted into functions, which were documented using the package {roxygen2} (Wickham et al. 2022). Roxygen automatically creates manuals and organises the NAMESPACE as well as parts of the DESCRIPTION file.
The DESCRIPTION file is a txt which contains the metadata of the package. Name, description, authors and dependencies are the most important ones. It is the defining file for the package. The NAMESPACE file defines the functions that are to be usable by the person installing the package. This file makes a package readily usable (Wickham and Bryan 2023).

A vignette, which explains the package functions in more detail, and an example dataset for users to try out the functions were added. These are not necessarily needed for an R package to run, but greatly improve usability. The dataset provided was a file of the hillfort locations used in Simon’s research, derived from the Atlas database (Lock and Ralston 2017).

The paper was published in the Journal of the CAA in 2020. The GitHub package is now open as well (https://github.com/SCSchmidt/percopackage). However, Github is not a publishing platform and does not currently support long term stability. The package was therefore released on the Open Science Framework (OSF, https://osf.io), which offers DOIs for research output. This way the package is citable and reliably findable.

Any package that is in a tar.gz-format can be installed in R manually and packages on Github can be installed using {remotes} (Csárdi et al. 2023). For these reasons we took a simpler approach by publishing the package on Github and OSF. The package - DOI (https://doi.org/10.17605/OSF.IO/7EXTC) was included in the paper. As it was linked to the paper, the package was also peer reviewed as part of the publication process, which was a really effective way to have it tried and critically evaluated. The paper itself was a useful promotional vehicle and we can now see people using the package and citing our paper as well

(for example: Díaz-Rodríguez and Fábregas-Valcarce 2022, Hu et al. 2022, Pajdla 2023, Verhagen 2023).

Although it has not yet been released on the Comprehensive R Archive Network (CRAN, https://cran.r-project.org/), it is our intention to do this. CRAN offers significant advantages for users and is great for easy installation and seamless integration into R, but does require a further tranche of work. We have some significant enhancements to the package in planning and we intend to complete these first.

Lessons learned

Traditionally, script sharing needs personal contact, either at an institution, at a conference or by contacting the author of the paper who stated for example “the code of this program can be requested”. Creating something new from a shared script needs either a really good understanding of it or, which we think more likely, a prolonged engagement with the original author, asking questions and proposing adaptations.

As a package, the original code was developed to have more flexible input schemas and therefore not only a broader variety of use applications, but also more comfortable handling. The code can now be adopted without contact with the original authors. As it is licenced and citable, no legal issues arise.

Other points that come out of our experiences are:

Comment your code properly from the very beginning; it makes it much easier to adapt later. It is a very old dictum, everyone says this, and it remains true. It is also an important and very valuable part of the code creation process.

Testing is essential from the start. The original software was rigorously tested during development, and particularly by successive versions of the source hillfort data, which was very much work in progress at the time, with many minor errors. Different sources of data, used for comparative analysis, were also useful in this process. Collaboration opened a new dimension in testing and is a valuable element in development.

Going from script to function requires some thought about standard inputs and possible applications outside of the original research question. For both of these, a collaboration with someone working on similar (but distinct) issues is very helpful.

It also makes the process easier, as tasks can be split. In this case, Simon remained the main author of the actual implementation of the working code, while Sophie did the restructuring to create a package, documentation and vignette.

It is not as hard as it initially seemed. Creating a package can seem daunting, but the guidance provided by Wickham and Bryan, for example, is very helpful and detailed.

Another topic we learned more about during this project was the way code can and should be shared within a scientific community. So called Research Software is being developed for most disciplines today. It may describe anything from an elaborate program suite to a simple code script that has been developed with the aim of generating, processing or analysing research data (Hettrick et al. 2014).

Good practice for code sharing

Putting the software online is only one step in making it available and useful to others. The FAIR (Findable, Accessible, Interoperable and Reusable) Guiding Principles for Data by Wilkinson et al. (2016) have been adapted to the publication of Research Software (Lamprecht et al. 2019, Goedicke and Lucke 2020), creating the FAIR Principles for Research Software (FAIR4RS Principles, Barker et al. 2022). Importantly it has been considered to contribute to the quality of research software (Homburg et al. 2020). To summarise, the guidelines for FAIR Research Software (Barker et al. 2022) suggest that:

Findable and Accessible: Software should be easy to find and access. Unique and persistent identifiers (PIDs) that allow for differentiation between different versions of the same software should be assigned to the software. Only this way can any (FAIR) metadata describing the software include the identifiers of the software they describe. Access should be given via standardised, open and free protocols.

Interoperable: The software should be able to exchange data and/or metadata with other software to enable its integration into workflows or other programs (e.g. dependency management).

Reusable: Software can be executed as well as understood, modified, built upon, or incorporated into other software. This includes an up-to-date description of the software, as well as an open licence, which allows for its reuse.

The full FAIR recommendations are extensive; in our view one needs to be pragmatic depending on the scope and complexity of the Research Software in question.

We will now consider how code is being actually shared in the archaeological discipline and if and to what extent these principles are being employed.

Code Sharing in Archaeology

At the outset of writing this paper we believed that with the increase in computational approaches within archaeological research, requiring more complex programming code, the need for sharing programs intensifies in order to keep the approaches scientific and reproducible. Creating a package or plugin as described above is one possible way of distributing the code. The following aims at contextualising our approach and discovering if and to what extent other methods are being used in archaeological research.

Particularly in older publications, code is often published in a printed form. Since 2008, research software engineers have been using Github, Gitlab or similar collaborative platforms, and there are of course websites, blogs, journal supplementary materials, bespoke repositories, such as CRAN or PyPI, or general repositories such as osf.io or zenodo.org, where code is being published. However, we now consider how widespread this practice actually is. Also, as addressed before, publishing is just one step in sharing code in a reusable way, so we also need to consider what community standards there are to make other researchers aware of published code.

Free and Open Source solutions are often advertised with links to the code at talks and presentations at conferences, such as ArcheoFOSS (https://www.archeofoss.org/) or CAA. At CAA especially the 'Little Minion' sessions aim at demonstrating coding solutions. A useful collection of Free and Open Source Software (FOSS) tools for archaeology is open-archaeo (https://open-archaeo.info/), where short descriptions of programs, some information on the maintainers, and a link to the software are collected and tagged with keywords. This catalogue can be also used to understand code sharing practices in archaeology. A second dataset was collected from articles in the Journal of the CAA (JCAA) and analysed with the same aims.

For the following analyses code and data are available online under https://github.com/SCSchmidt/CAA2023_code_sharing/ and https://doi.org/10.5281/zenodo.8307960 . They were created in R (R Core Team) with the packages: {here} (Müller 2020), {stringr} (Wickham 2022), {forcats} (Wickham 2023), {dplyr} (Wickham et al. 2023), {tidyr} (Wickham et al. 2023b), {ggplot2} (Wickham 2016) and {viridis} (Garnier et al. 2021).

Analysis of Open-Archaeo data

Joe Roe and Zack Batist have already shown an analysis of open-archaeo data for the CAA conference in 2021 (Batist and Roe 2021). Drawing on the open-archaeo dataset and their analysis we can see that most of the resources in open-archaeo are actually packages and libraries that either extend existing software or are standalone software solutions (see Fig. 3a). The research software is most often on GitHub and in some cases further promoted: Those tools are linked to a scientific paper, a website, a blogpost or similar, which explicitly „advertise" their program. Of 488 entries, some are on several platforms, mostly with overlap to Github (335, see Fig. 3b). About 117 are linked to a website, nine have been linked to a paper and 18 to a DOI (with some overlap), 19 are on CRAN, two on PyPI – R and Python being the most common platforms (Batist and Roe 2021). The scientific publication and the archiving of a released version with a repository that gives a persistent identifier is most common for packages and libraries. All in all though, the open-archaeo data suggests that there is a lot of material online which is rarely or not referenced in other publication outlets and may be hard to find for potential users. This also reveals what a valuable contribution open-archaeo constitutes to the archaeological community.

Nonetheless, open-archaeo is a collection of online resources and an online platform may not always link to a journal publication or similar. Another question that may be difficult to answer

via open-archaeo data is how analysis code for research papers is being treated. Therefore, a further angle was taken as well, and articles published in the Journal of the CAA (JCAA) analysed. The JCAA recommends using repositories for Open Code and Open Methods

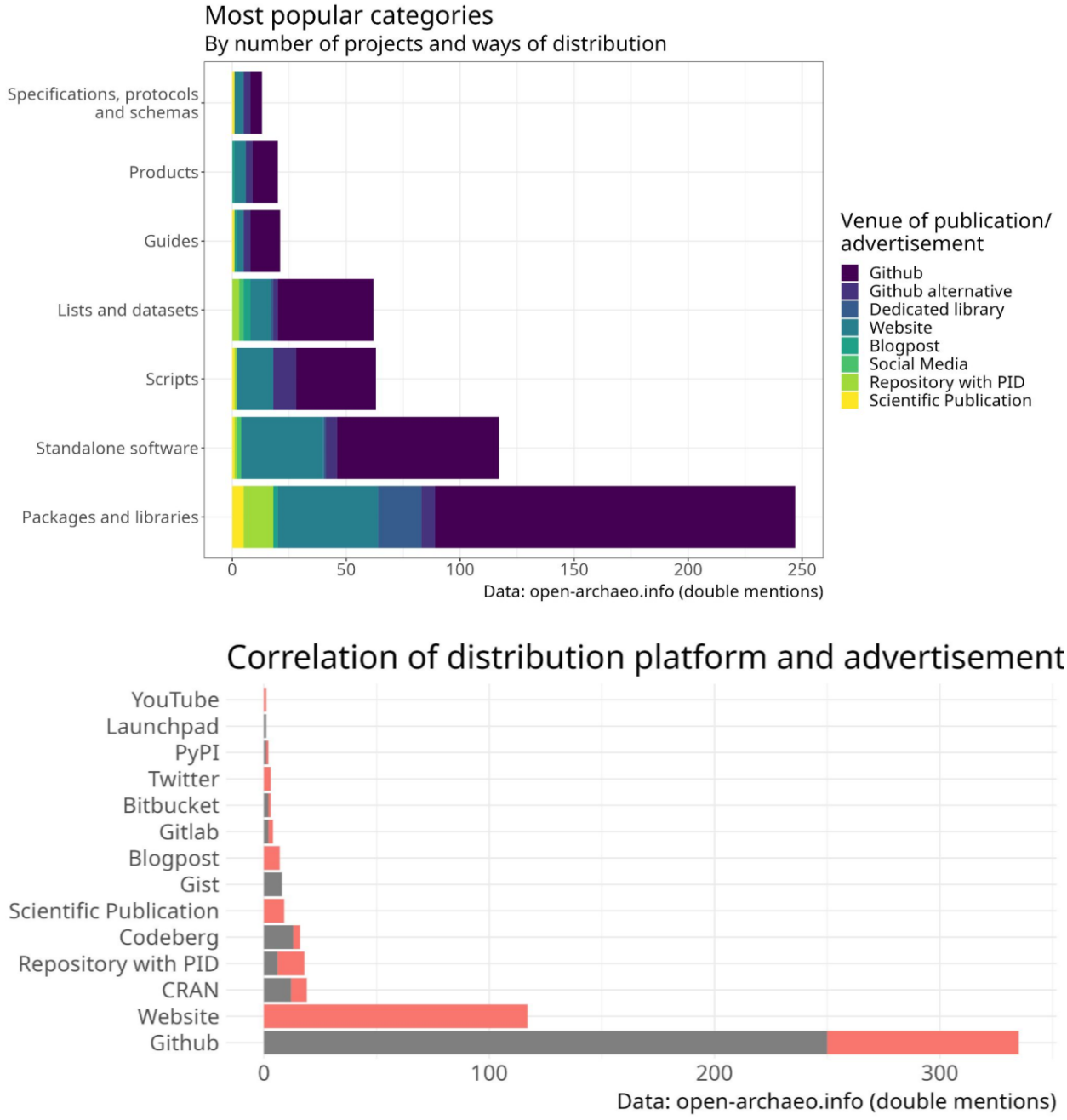

Fig. 3 a) Most popular kinds of software on open-archaeo.info (adapted from Batist and Roe 2021) and b) Correlation of advertisement and distributing software listed on open-archaeo.info. Red: software that is linked to a paper, website or blog, indicating some advertisement of the software product.

(https://journal.caa-international.org/reproducibility). It is one of the main publishing venues for computational archaeology and so particularly interesting to analyse a CAA publication. It is not by far, though, the only journal in which relevant articles are being published (for R see Schmidt and Marwick 2020, Fig. 4 and https://github.com/benmarwick/ctv-archaeology).

Analysis of JCAA articles

There are at the time of submission 68 research articles and case studies published in JCAA between 2018 and 2023. They were checked manually for references to program code used in the analysis and the way this code may have been published. The search was restricted to case studies and research articles, as it seemed more probable that programming was needed for the creation of the published analysis than for reviews, position papers or editorials. Of the 68 articles, in 32 cases it can be assumed program code was produced for the research (the database under https://doi.org/10.5281/zenodo.8307960 consists of 58 articles, as articles of a theoretical nature were not included). Of those, 22 articles contained links to published code, two of which did not work (though they were DOIs) and one article printed SPSS code. These publications from the last five years may capture current trends within the CAA community, but as of yet not long-term developments. The documentation of the submitted code was quickly and qualitatively checked, and the following variables noted: the format (*comments only*, *added README,* or *manuals*); the length (*good*, *some*, *little*); whether the code was licensed and how metadata was provided (*plenty* if a number of structured entries, *some* if some basic information was given, *little* is usually just author and publication date).

The articles for 2023 cannot yet be considered representative of the year and the peak in 2020 (see Fig. 4 and 6) is most probably related to the pandemic induced reduction of publications in the years 2021 and 2022. The number and percentage of publications using persistent identifiers to link to code seems to be growing, with the exception of 2021 (see Fig. 4). Zenodo

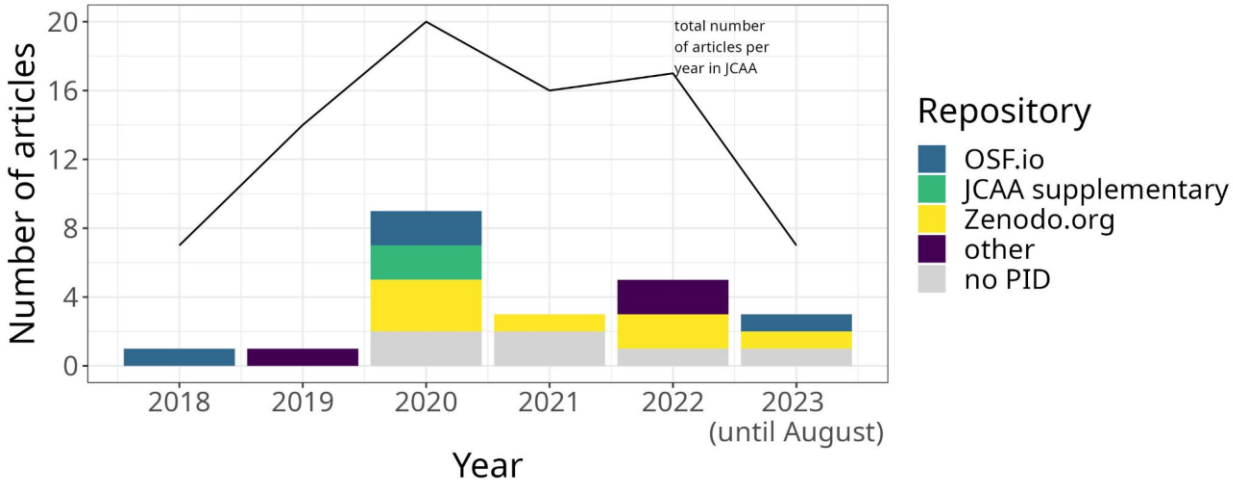

Fig. 4: Number of articles published in JCAA (black line) and the number of times a repository was used to gain a Persistent Identifier (PID) for the publication of code. Grey indicates code publication without a PID.

has been used most often to create a persistent identifier, though OSF.io and the supplementary materials of JCAA have been employed as well. “Other” refers to an university repository, printed code and one not resolving DOI. It is noteworthy that no kind of persistent identifier other than DOI was used for code in this sample of JCAA articles.

In most cases where research software is published together with the paper, the code is organised in some way, e. g. in the form of a research compendium, package or program (package denoting any kind of extension on existing software, whereas program is a stand-alone, see Fig. 5). Of those, most have a README as documentation, which are in some cases more elaborate than others. Manuals are available only for programs and packages. About half of the time (10 times) the code is organised in a compendium (denoting a folder structure with code, data, README or some other kind of documentation), seven times only scripts were published.

Licences were given in all but seven cases, most often MIT, GPL-3.0 or a CC-BY variant. Metadata, structured information on the published code to improve results of human and machine search, seems to be provided more in recent years (see Fig. 6). The material published in the Zenodo repository is more often associated with providing at least some metadata than research software in other repositories (see Table 1). Zenodo is also quite often used in combination with Github (six times, OSF.io: two times, Github being used 15 times altogether). It offers an easy integration and publication of Github repositories (https://docs.github.com/en/repositories/archiving-a-github-repository/referencing-and-citing-content).

Repository / Metadata	**none**	**little**	**some**	**plenty**
Zenodo.org	0	0	6	1
OSF.io	0	1	3	0
JCAA supplementary	1	1	0	0
diverse	1	0	0	1
No PID giving repo	1	3	1	1

Table 1: correlation of repository and metadata provided

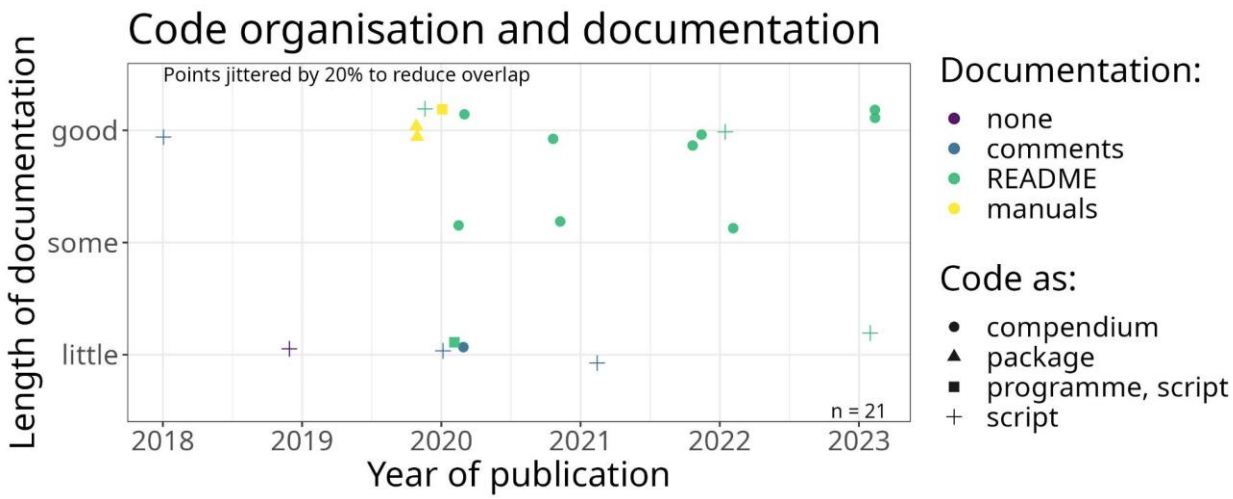

Fig. 5: Jittered scatter plot to show the way code has been published ("Code as"), the kind of documentation available and the length of this publication.

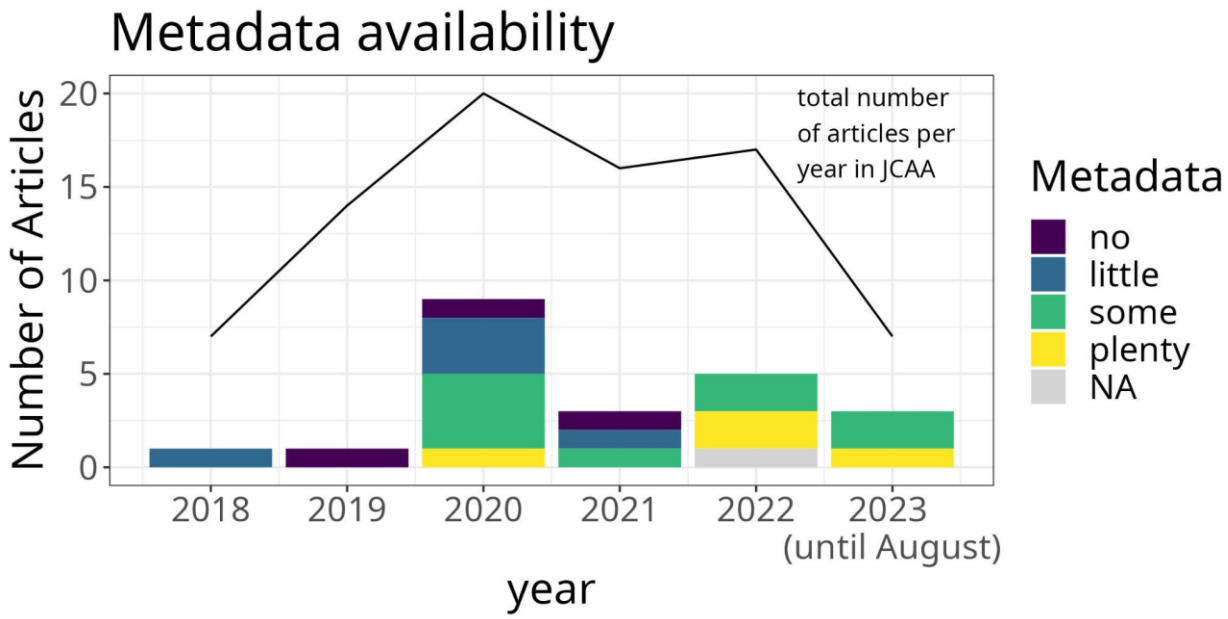

Fig. 6: Amount of metadata provided per year (no: no additional information, little: usually author, date of publication, some: adds 2-3 more structured points of information, plenty: more than that, NA: DOI did not resolve).

This small study of JCAA case studies and research articles shows that archaeologists are aware of methods for making research software FAIR. They utilise persistent identifiers for publishing code, though structured metadata is still rarely employed to its full potential. Documentation of code varies, especially not just the width and breadth of code comments, but rather providing additional information to make it easier for other researchers to gauge the usefulness of this software for their own project. Commendable is the use of research compendia, structuring analysis data and code. These are quite common, showing the aim of

making research reproducible, which may also be linked to JCAA's reproducibility statement (https://journal.caa-international.org/reproducibility), encouraging authors to provide code and data.

Github is used for version control, collaboration and publication of code about two thirds of the time. It is good practice to archive a Github repository for accessibility and long-term stability of the code. A repository version can e. g. be released via Zenodo or OSF.io and Github used for eventual further development and version control. About half of the people using Github employed such a strategy.

The open-archaeo data shows that archaeological research software engineers use Github as much as any other professional software developers. Here however, the percentage of persistent identifiers used is much lower. Both might be due to the way the data was collected, as it was gathered manually from collaborative software development platforms (Batist and Roe 2021). There may be links missing to other resources, such as to studies, in which software was employed. No research software within the open-archaeo dataset links to a JCAA article, which also shows the limitations of this small study. Links to other online resources such as websites or bespoke repositories for software publication (CRAN, PyPI) are given more often than within the JCAA datasets.

The two datasets exemplify two different areas that come together in research software. The online list open-archaeo links to the platforms that host the software development process. This process is open-ended, as software routinely continues to be further developed, adapted and enhanced. PIDs are rare, but metadata is usually more amply added, as the findability of a software within the library is valued. In contrast, the Journal of the CAA focuses on research results. Here PIDs are more common, to ensure the exact version of the code (and data) used to generate these results can be accessed later: The reproducibility of the results is often the main reason for sharing the code. Within these publications, though, metadata provision seems less standardised, maybe because the paper itself is regarded as the main venue through which the code is supposed to be found.

Conclusions

In this article we elaborated on code sharing practices in archaeology. It was inspired by the development of an R package, which we describe in the beginning and of which we share lessons learned. The second part is an analysis of two datasets that show two different stages of scientific research software: development and use for creating research results. Here we will summarise the main points and further discuss the advantages of properly shared FAIR research software.

The process described above exemplifies how script sharing and co-creation "traditionally" works: personal contact between researchers leads to a collaboration, in which different versions are mailed back and forth. The script adaptation for own use requires communication between the script author and the person implementing. In our case this led to a discussion of

flexibility of input and use cases, which inspired the creation of the package. The package creation was a time costly endeavour, especially the documentation. This reflected shortcomings in the original code writing process, and illustrates the point that it should be done properly in the first place, however limited the original aims might have been.

To make a program, package or library useful to others, it is a good idea to consider standard inputs and possible applications outside of the original research question. For this, collaboration is a very valuable way of working towards publication, in terms of diversity of application, testing through unfamiliar datasets and sharing of work. With this contribution we wanted to show that it is not as hard as it might seem and that it is an important way of adding to the corpus of research tools.

As stressed throughout the paper, there are benefits to a thorough publication. The FAIR standards are on one hand a way to enable others to more easily gain from the effort put into the code creation in the first place and on the other hand give benefits to the developer as well. They create long-term stability and accessibility to the code and facilitate re-use and further development.

By making the research software citable the developer can be cited as its author. Citations are a currency of scientific research and can, in this case, showcase how research software development can be part of the skill set of modern researchers. For a sample of archaeological articles from the years 2010-2017 from the web of science, it could be shown that researchers that cite R Core are on average more often cited than those that do not (Schmidt and Marwick 2020, 24).

Just as other researchers may rely on an analysis published as a research paper, they are able to reuse code already published. Also, a researcher who continues his research on a topic does not need to re-publish his code with each paper when he can refer to his FAIR program version. The process of publication may also improve the code. Bespoke code repositories such as CRAN or PyPI have a code checking process and the publication together with a paper can lead to the code being peer reviewed as well. In general, developers may be more careful in their work when they know it will be published at some point. Reviews of published software can be a valuable source of checking for the quality of the program as well (Homburg et al. 2020).

It seems that in the CAA community there are people aware of these benefits who move towards a more formal publication of software than an upload to Github. We want to encourage others to follow their example and release the version of a software used for an article to a PID issuing repository, such as Zenodo or OSF.io.

To return to our package: Following the work to bring the package to publication, the code is now much easier to use and no contact with the authors is needed, which means time is saved by the person using the code. This supports the thesis by Schmidt and Marwick (2020), that code sharing may speed up scientific progress in archaeology. Also, the contribution is citable and there is no need to ask permission to reuse the program. This greatly improves usability, as can be seen by its application in different papers since 2020. We therefore strongly suggest,

that though sharing of scripts is a valid approach that already speeds up the adoption and adaptation of new methods, once the code is developed further, it should be distributed in a structured and easy to implement format suited to the environment it is run in (e.g. R package, QGIS plugin or similar) and uploaded to the domain repository (such as CRAN, PyPI or QGIS plugin repository).

Reproducibility

Code and Data for the analysis of code sharing practices in archaeology are available on https://github.com/SCSchmidt/CAA2023_code_sharing and https://doi.org/10.5281/zenodo.8307960.

The discussed R-package for percolation analysis, 'percopackage', can be found under https://github.com/SCSchmidt/percopackage and https://doi.org/10.17605/OSF.IO/7EXTC.

Acknowledgements

Thanks to Joe Roe and Zack Baptist for sharing their analysis and data for open-archaeo.info on Github and agreeing to letting us analyse it further. Thanks also to the very constructive comments of the reviewers.

Funding

Sophie Schmidt's travel costs to Amsterdam for the CAA conference were financed by the Berlin Graduate School of Ancient Studies and CAA international.

Conflict of interest disclosure

The authors declare that they comply with the PCI rule of having no financial conflicts of interest in relation to the content of this article.

Bibliography

Arcaute, E, Molinero, C, Hatna, E, Murcio, R, Vargas-Ruiz, C, Masucci, AP and Batty, M. (2016). Cities and regions in Britain through hierarchical percolation. *Royal Society Open Science*, 3(4): 150691. DOI: https://doi.org/10.1098/rsos.150691

Arcaute, E., Brookes, S., Brown, T., Lake, M. & Reynolds, A. (forthcoming). Case studies in percolation analysis: the distribution of English settlement in the 11th and 19th centuries compared. *Journal of Archaeological Science.*

Barker, M., Chue Hong, N.P., Katz, D.S. *et al.*(2022). Introducing the FAIR Principles for research software. *Sci Data* **9**, 622 https://doi.org/10.1038/s41597-022-01710-x

Batist, Z., and Roe, J. (2021). *Open archaeology: a survey of collaborative software engineering in archaeological research*. Presented at Computer Applications & Quantitative Methods in Archaeology, Limassol (Virtual), 14–18 June 2021. https://github.com/zackbatist/caa2021-openarchaeo (last accessed 2023-08-24)

Csárdi G, Hester J, Wickham H, Chang W, Morgan M, Tenenbaum D (2023). remotes: R Package Installation from Remote Repositories, Including 'GitHub'_. R package version 2.4.2.1, https://CRAN.R-project.org/package=remotes

Díaz-Rodríguez, M., Fábregas-Valcarce, R. (2022). Evaluating the effectiveness of three spatial cluster analysis methods on Palaeolithic site location distributions in Galicia, NW Iberian Peninsula, *Journal of Archaeological Science*: Reports 41. DOI; https://doi.org/10.1016/j.jasrep.2021.103323

Garnier, S., Ross, N., Rudis, R., Camargo, A. P., Sciaini, M. and Scherer, C. (2021). *Viridis - Colorblind-Friendly Color Maps for R*. R package version 0.6.2.

Goedicke, M., Lucke, U. (2020). *Nationale Forschungsdateninfrastruktur für und mit Computer Science (NFDIxCS)*. https://www.dfg.de/download/pdf/foerderung/programme/nfdi/nfdi_konferenz_2020/nfdixcs_abstract.pdf [21.08.2023]

Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D., et al. (2014). *UK Research Software Survey 2014.* Zenodo, 4.12.2014. https://doi.org/10.5281/zenodo.14809

Homburg, T., Klammt, A., Mara, H., Clemens Schmid, C., Schmidt, S.C., Thiery, F., Trognitz, M. (2020). Recommendations for the review of archaeological research software. *Archäologische Informationen* 43, 357- 370, https://doi.org/10.11588/ai.2020.1.81423

Hu, X., Wang, Y., Wang, H., Shi, Y. (2022). Hierarchical Structure of the Central Areas of Megacities Based on the Percolation Theory—The Example of Lujiazui, Shanghai. *Sustainability* 14(16):9981. https://doi.org/10.3390/su14169981

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E. et al. (2019). Towards FAIR principles for research software. *Data Science,* 3(1), 37-59. https://doi.org/10.3233/DS-190026

Lock, G. and Ralston, I. *Atlas of Hillforts of Britain and Ireland*, https://hillforts.arch.ox.ac.uk/, (last accessed 2023-07-01)

Maddison, S. (2016) The Spatial Distribution of Iron Age Hillforts in the British Isles. Unpublished dissertation (MSc). Institute of Archaeology, University College London.

Maddison, S. and Schmidt, S.C. (2020). Percolation Analysis – Archaeological Applications at Widely Different Spatial Scales. *Journal of Computer Applications in Archaeology*, 3(1), 269–287. DOI: https://doi.org/10.5334/jcaa.54

Müller K. (2020). *here: A Simpler Way to Find Your Files*. R package version 1.0.1, https://CRAN.R-project.org/package=here

R Core Team (2023). R: *A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

Schmidt, S. C. and Marwick, B. (2020). Tool-Driven Revolutions in Archaeological Science. *Journal of Computer Applications in Archaeology*, 3(1), 18–32. DOI: https://doi.org/10.5334/jcaa.29

Verhagen, P. (2023). Centrality on the periphery: an analysis of rural settlement hierarchy in the Dutch part of the Roman *limes*. *Archaeological and Anthropological Sciences* 15, 45. DOI: https://doi.org/10.1007/s12520-023-01745-0

Wickham H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org

Wickham H. (2022). *stringr: Simple, Consistent Wrappers for Common String Operations*. https://stringr.tidyverse.org, https://github.com/tidyverse/stringr

Wickham H, Danenberg P, Csárdi G, Eugster M (2022). roxygen2: In-Line Documentation for R. R package version 7.2.3, https://CRAN.R-project.org/package=roxygen2

Wickham H. (2023). *forcats: Tools for Working with Categorical Variables (Factors)*. https://forcats.tidyverse.org/ , https://github.com/tidyverse/forcats

Wickham, H. and Bryan, J. (2023). R Packages (2e). L*earn how to create a package, the fundamental unit of shareable, reusable, and reproducible R code*, https://r-pkgs.org/ (last accessed 2023-08-24)

Wickham H., François R., Henry L., Müller K., Vaughan D. (2023). *dplyr: A Grammar of Data Manipulation*. https://dplyr.tidyverse.org, https://github.com/tidyverse/dplyr

Wickham H., Vaughan D., Girlich M. (2023b). *tidyr: Tidy Messy Data*. R package version 1.3.0, https://CRAN.R-project.org/package=tidyr

Wilkinson, M., Dumontier, M., Aalbersberg, I. et al. (2016) The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3, 160018. DOI: https://doi.org/10.1038/sdata.2016.18