Taylor & Francis
by informa

# Code and Software at Taylor & Francis

Matt Cannon – Associate Director of Open Science Programmes

# Code and Software background

Code is currently called out as a type of data in our data sharing policy framework

However, most authors and our editors/internal checks focus on data availability rather than code or software availability.

Work is underway to remove code and software as an example of a type of data, and have a standalone policy

# Initiatives

Taylor & Francis staff have contributed to a range of community projects on software citations and production guidance

Taking an active role in setting guidelines on how code and software should be handled in publishing workflows

scientific data

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › scientific data › comment › article

Comment | Open access | Published: 26 September 2023

**Journal Production Guidance for Software and Data Citations**

Shelley Stall ✉, Geoffrey Bilder, Matthew Cannon, Neil Chue Hong, Scott Edmunds, Christopher C. Erdmann, Michael Evans, Rosemary Farmer, Patricia Feeney, Michael Friedman, Matthew Giampoala, R. Brooks Hanson, Melissa Harrison, Dimitris Karaiskos, Daniel S. Katz, Viviana Letizia, Vincent Lizzi, Catriona MacCallum, August Muench, Kate Perry, Howard Ratner, Uwe Schindler, Brian Sedora, Martina Stockhause, ... Timothy Clark + Show authors

*Scientific Data* **10**, Article number: 656 (2023) | Cite this article

**9599** Accesses | **13** Citations | **40** Altmetric | Metrics

F1000Research Search

BROWSE GATEWAYS & COLLECTIONS HOW TO PUBLISH ⌄ ABOUT ⌄

Home » Browse » Recognizing the value of software: a software citation guide

Check for updates

METHOD ARTICLE

**REVISED** **Recognizing the value of software: a software citation guide**

[version 2; peer review: 2 approved]

Previously titled: "The importance of software citation"

Daniel S. Katz [1], Neil P. Chue Hong [2], Tim Clark[3], August Muench [4], Shelley Stall [5], Daina Bouquin[6], Matthew Cannon [7], Scott Edmunds[8], Telli Faez[9], Patricia Feeney[10], Martin Fenner[11], Michael Friedman [12], Gerry Grenier [13], Melissa Harrison [14], Joerg Heber[15], Adam Leary [16], Catriona MacCallum [17], Hollydawn Murray[18], Erika Pastrana[19], Katherine Perry [20], Douglas Schuster[21], Martina Stockhause [22], Jake Yeston[23]

Author details

# Current usage

Teams have been working to implement best practice from these community projects

For example, citation best practice required work by production teams to create software reference examples in unsupported reference styles

**Software**

When citing software that you've created or used in your research, you should include the following details:

- **Creator(s):** the authors or project that developed the software.
- **Title:** the name of the software.
- **Publication venue:** the venue of the software, preferably an archive or repository that provides persistent identifiers.
- **Date:** the date the software was published. This is the date associated with a release or version of the software, or "n.d." if the date is unknown.
- **Identifier:** a resolvable pointer to the software, preferentially, a persistent identifier (PID) that resolves to a landing page containing descriptive metadata about the software, similar to how a Digital Object Identifier (DOI) for a paper that points to a page about the paper rather than directly to a representation of the paper, such as the PDF. DOIs are preferable, and other examples of PIDs include Handles, RRIDs, ASCL IDs, swMath IDs, Software Heritage IDs, ARKs, etc. If there is no PID for the software, a URL to where the software exists may be the best identifier available.

If relevant, it may also be useful to include the following additional information:

- **Version:** the identifier for the version of the software being referenced. If the version is unidentified or unknown, the date of access should be used.
- **Type:** some citation styles (e.g., APA), require a bracketed description of the citation (e.g., Computer software) to be included.

For further guidance we recommend Software Citation Checklist for Developers, which includes information on how to obtain a PID or choose a software license for software you've developed.

# Proposals and next steps

We are preparing to introduce a code and software sharing policy at a journal level

Would allow journals to choose to add code and software sharing expectations for authors

For data sharing we have five policy levels; planning Code and Software to be less complicated:

Two levels of policy:

- Encouraged code and software sharing policy
- Required code and software sharing

# Considerations

- This is independent of data sharing policy
    - Allows journals to require data sharing, but encourage code/software sharing; vice versa
- Will encourage/require a "software availability statement";
    - This will be separate from DAS to ease with checking by non-specialists
- Will add questions to the submission process to ensure authors have followed the policy
- Will add questions to the internal checklists we use to check new submissions
- New author guidance pages

# New Resources

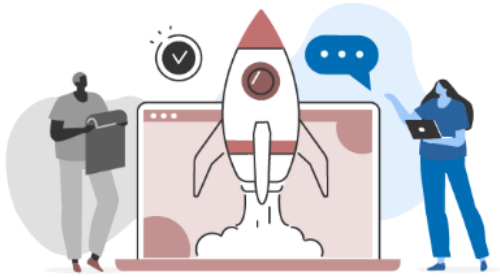
## Sharing your research code and software

Increase the visibility and impact of the code you develop during your research

Jump to section | Why is code sharing important? | How to share your code and software | Citing code and software | Peer review

### What are code and software?

Code is a human readable executable command for a computer. Lines of code are put together to make software programs. Code and Software are increasingly used in academic research in a range of ways:



**Some researchers write their own *analysis code*. These are created to help researchers create or analyse datasets.**

- There are different languages for writing code, such as C, C , C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala, and TypeScript
- This may occur on online collaboration platforms like Github

**Some researchers may use software tools created by others. These tools may be more professional, but have similar uses of creating or analyzing datasets.**

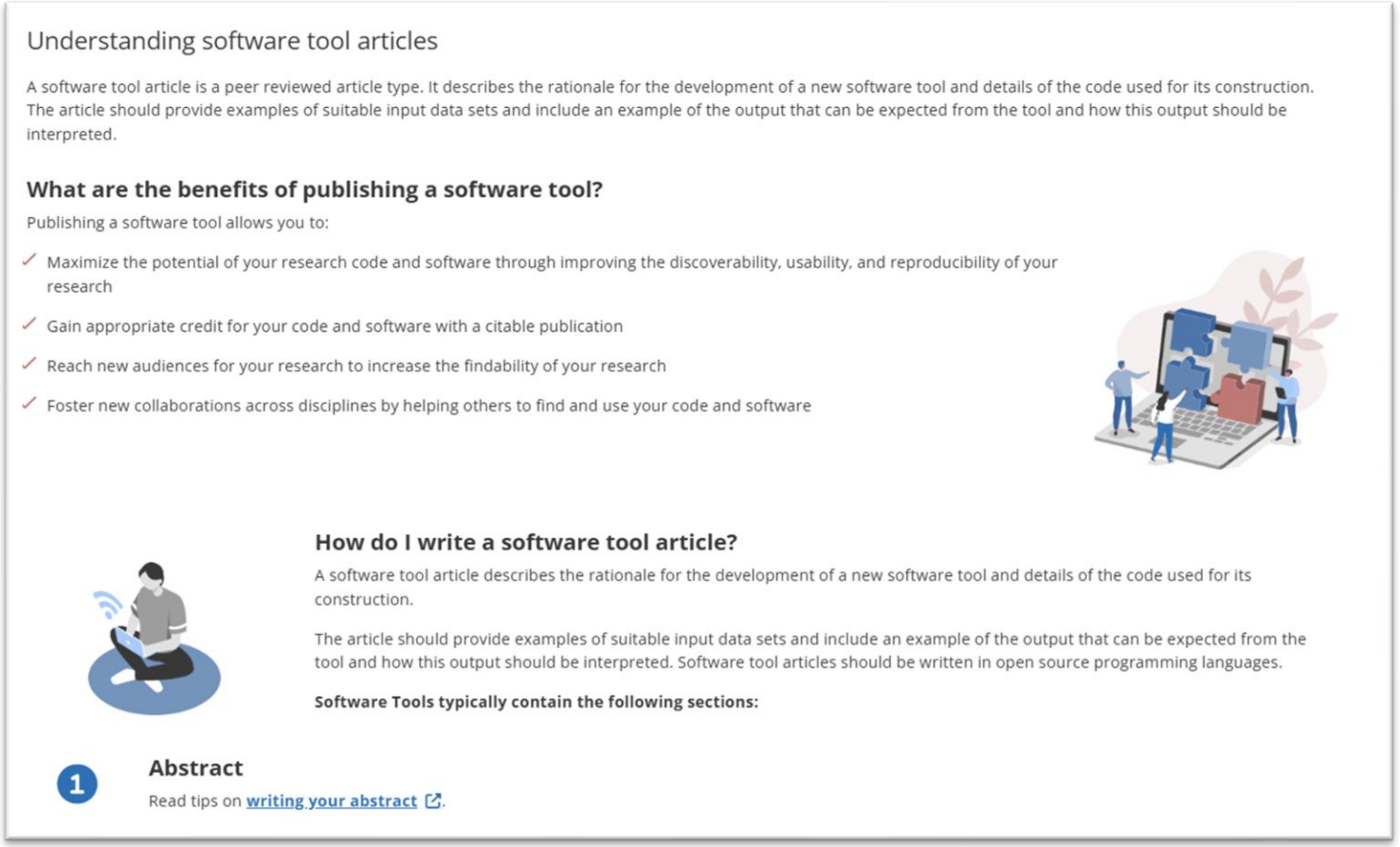
- Some researcher's activity is about creating software tools to help other researchers..
- These tools may be shared *open source*, which is where the tool is shared with a license that allows other researchers to use it without fees or permission.

On the biggest scale, there is research infrastructure. These are software programmes that are relied upon by other researchers to do their work. For example: SPSS.

# New Article Type: Software Tool Articles

Short Description:
A Software Tool Article should include the rationale for the development of the tool and details of the code used for its construction.

The article should provide examples of suitable input data sets and include an example of the output that can be expected from the tool and how this output should be interpreted.



Understanding software tool articles

A software tool article is a peer reviewed article type. It describes the rationale for the development of a new software tool and details of the code used for its construction. The article should provide examples of suitable input data sets and include an example of the output that can be expected from the tool and how this output should be interpreted.

**What are the benefits of publishing a software tool?**

Publishing a software tool allows you to:

- ✓ Maximize the potential of your research code and software through improving the discoverability, usability, and reproducibility of your research
- ✓ Gain appropriate credit for your code and software with a citable publication
- ✓ Reach new audiences for your research to increase the findability of your research
- ✓ Foster new collaborations across disciplines by helping others to find and use your code and software

**How do I write a software tool article?**

A software tool article describes the rationale for the development of a new software tool and details of the code used for its construction.

The article should provide examples of suitable input data sets and include an example of the output that can be expected from the tool and how this output should be interpreted. Software tool articles should be written in open source programming languages.

**Software Tools typically contain the following sections:**

1 **Abstract**

Read tips on writing your abstract.

# Software Tool - examples

INTERNATIONAL JOURNAL OF SUSTAINABLE ENERGY
2026, VOL. 45, NO. 1, 2673270
https://doi.org/10.1080/14786451.2026.2673270

Taylor & Francis by informa

SOFTWARE TOOL ARTICLE

OPEN ACCESS Check for updates

## ShinyEnet: an in-house simulation software for data-driven waste-to-energy gasification and pyrolysis

Dejan Brkić[a,b], Judita Buchlovská Nagyová[a], Tomáš Martinovič[a], Renáta Praksová[a], Lukáš Vojáček[a], Jan Najser[c], Jan Kielar[c], Václav Bolom[d], Martin Marek[c,e], Michal Běloch[a] and Pavel Praks[a]

[a]VSB - Technical University of Ostrava, IT4Innovations, Ostrava, Czech Republic; [b]University of Niš, Faculty of Electronic Engineering, Niš, Serbia; [c]VSB - Technical University of Ostrava, ENET Centre, Ostrava, Czech Republic; [d]ABB Operation Center, IAPI – Power systems, EUOPC, Ostrava, Czech Republic; [e]College of Polytechnics Jihlava, Department of Technical Studies, Jihlava, Czech Republic

**ABSTRACT**

ShinyEnet is an open-source software tool for modelling waste-to-energy processes, including gasification and pyrolysis. Developed at IT4Innovations, the National Supercomputing Centre of the Czech Republic at VSB - Technical University of Ostrava, it utilizes operational data from experimental facility at the Centre for Energy and Environmental Technologies—Explorer (CEETe), also part of the same university. The software models a modular, mobile, and scalable system that converts waste into gaseous or liquid fuels. ShinyEnet supports dynamic simulation, including component-failure cases, optimization, and scenario analysis. The platform thus facilitates development and assessment of compact and mobile waste-to-energy units and provides tools for addressing municipal waste-management challenges. ShinyEnet is based on real operational datasets, with continuously updated data currently available for the pyrolysis process via real-time monitoring system. The interactive web application is implemented using open-source Python libraries and employs validated historical data and machine-learning models to simulate and optimize system performance.

**ARTICLE HISTORY**
Received 14 March 2026
Accepted 8 May 2026

**KEYWORDS**
Machine learning; hydrogen; pyrolysis; syngas; software; waste-to-energy

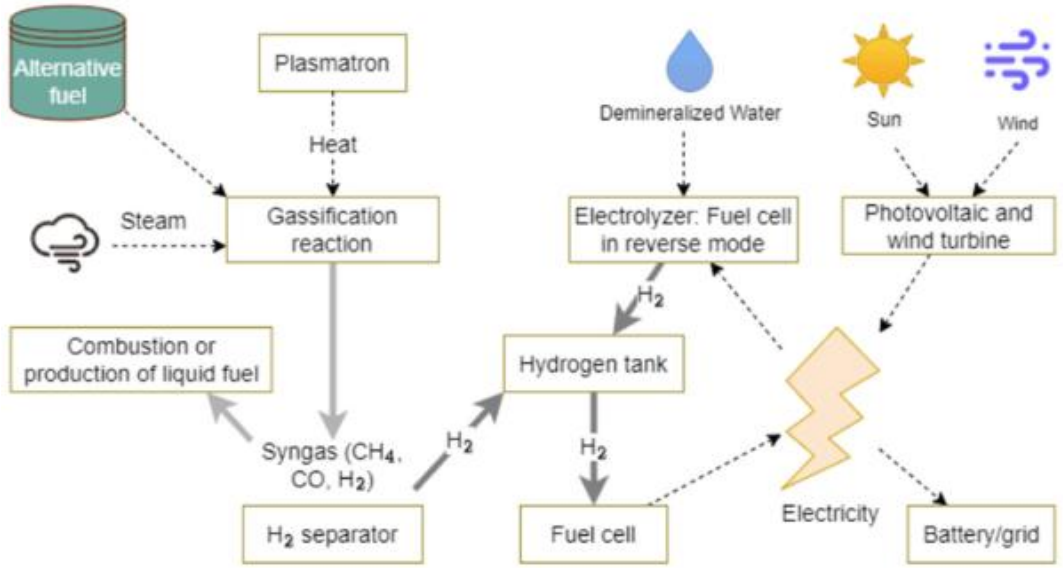



**Figure 1.** Simplified scheme for gasification process used in the ShinyEnet software.

- ShinyEnet is an open-source software tool for modelling waste-to-energy processes
- The software models a modular, mobile, and scalable system that converts waste into gaseous or liquid fuels. ShinyEnet supports dynamic simulation, including component-failure cases, optimization, and scenario analysis.
- ShinyEnet is based on real operational datasets, with continuously updated data currently available for the pyrolysis process via real-time monitoring system. The interactive web application is implemented using open-source Python libraries and employs validated historical data and machine-learning models to simulate and optimize system performance.

Brkić, D., Buchlovská Nagyová, J., Martinovič, T., Praksová, R., Vojáček, L., Najser, J., … Praks, P. (2026). ShinyEnet: an in-house simulation software for data-driven waste-to-energy gasification and pyrolysis. *International Journal of Sustainable Energy*, *45*(1). https://doi.org/10.1080/14786451.2026.2673270

Taylor & Francis
by informa

Thank you for listening!

Please get in touch:
Matthew.Cannon@tandf.co.uk