IJDC | *General Article*

Improving the Reproducibility of LaTeX Documents by Enriching Figures with Embedded Scripts and Data

Christian T. Jacobs
Defence Science and Technology Laboratory (Dstl)

Abstract

The introduction of open access data policies by research councils, the enforcement of best practices, and the deployment of persistent online repositories have enabled datasets that support results in scientific papers to become more widely accessible. Unfortunately, despite this advancement in the curation/publishing workflow, the data-driven figures within a paper often remain difficult to reproduce. Plotting or analysis scripts rarely accompany the manuscript or any associated software release; and even if they do, it may be unclear exactly which version was used. Furthermore, the precise commands and parameters used to execute the scripts are often not included in a README file or in the paper itself. This paper introduces a new open source digital curation tool, Pynea, for improving the reproducibility of LaTeX documents. Each figure within a document is enriched by automatically embedding the plotting script and data files required to generate it, such that it can be regenerated by readers of the paper in the future. The command used to execute the plotting script is also added to the figure's metadata, along with details of the specific version of the script used (if the script is tracked with the Git version control system). If the document is to be recompiled with a figure that has since changed, or had its plotting script or data files modified, the figure is regenerated such that the author can be confident that the latest version of the figure and its dependencies are included.

Received 06 April 2019 | *Revision received* 30 June 2019 | *Accepted* 12 August 2019

Correspondence should be addressed to Dr Christian T. Jacobs, Defence Science and Technology Laboratory (Dstl), Porton Down, Salisbury, Wiltshire, SP4 0JQ, United Kingdom, Email: cjacobs@dstl.gov.uk

Introduction

Motivation

Journal articles and conference proceedings remain a popular means of disseminating important results, backed up by data-driven figures or plots, to a vast audience. However, ensuring that these figures can be readily reproduced by others, through the availability of the underpinning software and data, is a crucial aspect of the workflow (de Leeuw, 2001) that has largely been overlooked in the past. This has occasionally led to a lack of confidence in results by peers and the potential for the credibility of the data to be called into question (Allison, Brown, George & Kaiser, 2016; Donoho, Maleki, Rahman, Shahram & Stodden, 2009; Stodden et al., 2013). The scientific research and publication process is undergoing significant reform in response to this so-called reproducibility crisis (Baker, 2016; LeVeque, Mitchell & Stodden, 2012). Indeed, recent efforts by funding bodies, publishers, and educational establishments are attempting to address this issue by facilitating the implementation of reproducible research practices.

The requirement to deposit data files in openly-accessible repositories, such as those provided by the UK Data Service[1], Figshare[2] or Zenodo[3], is now mandated by many funding bodies such as the Economic and Social Research Council (ESRC) (Economic & Social Research Council (ESRC), 2017) and the Engineering and Physical Sciences Research Council (EPSRC)[4]. These stricter policies and approaches will hopefully further the longevity of the data that supports funded publications. Similarly, the release of the software under open source licences also increases the transparency of the scientific workflow and also its potential uptake with new users (Easterbrook, 2014; von Krogh & Spaeth, 2007). Educational resources, such as the Open Science Massive Open Online Course (MOOC)[5], are raising awareness of various techniques and tools available to researchers, and demonstrating how to use them effectively, to facilitate a more open and transparent scientific workflow. Finally, dedicated software journals, such as SoftwareX[6] and The Journal of Open Source Software[7], are a relatively new phenomenon in the realm of scientific publishing. Not only do they offer a route for researchers to obtain recognition for this key component of the research workflow, they encourage the submission of the software itself to persistent online repositories; this in turn can be readily accomplished using tools such as PyRDM (Jacobs, Avdis, Gorman & Piggott, 2014) and dvn (Leeper, 2014). As a result, the longevity of the software's existence is greatly enhanced, enabling its future use by other researchers around the world.

Unfortunately, despite the increasing availability of the underpinning software and data, the process of re-generating a figure in a scientific manuscript is often still fraught with difficulty. For example, the precise commands and parameters passed to the plotting script to produce a particular figure are frequently omitted from README files or the

[1] UK Data Service: https://ukdataservice.ac.uk/
[2] figshare: https://figshare.com/
[3] Zenodo: https://zenodo.org/
[4] EPSRC website - Expectations: https://epsrc.ukri.org/about/standards/researchdata/expectations/
[5] Open Science MOOC: https://opensciencemooc.eu/
[6] SoftwareX: https://www.journals.elsevier.com/softwarex
[7] The Journal of Open Source Software: https://joss.theoj.org/

manuscript itself. Similarly, even if the manuscript includes a ‘Data Statement’ with links to all the relevant datasets, it may be unclear which one was used in a given figure. Furthermore, the individual plotting scripts themselves are often not published along with the software used to generate the data itself; and even when they are included, they often do not come with a version number, making it difficult to ascertain whether the correct version is being used when attempting to regenerate a figure. These crucial components of the analysis and curation workflow are, unfortunately, frequently overlooked (Vandewalle, 2012) making the process of reproducing a figure tedious, error-prone, or practically impossible. Addressing this issue requires better support for manuscript authors, particularly through the use of software tools that make the process of reproducing a figure as straight-forward as possible.

Improving the Reproducibility of Figures

LaTeX is a document preparation system (Lamport, 1994) that is especially popular within the scientific community. When an author wishes to submit their manuscript to a journal, the LaTeX ‘source’/markup can be compiled, typeset and rendered as a single manuscript in Portable Document Format (PDF) (Adobe Systems Incorporated, 2006) which is supported by most major publishers. Indeed, the PDF is extremely prevalent within the publishing industry because it offers a feature-rich, cross-platform environment. It also includes the ability to embed files within a manuscript, and to enrich the manuscript itself via metadata fields. By exploiting this wide range of functionality it is possible to develop tools that can facilitate the reproducibility of the figures within the manuscript itself.

Tools such as Sweave (Leisch, 2002) and knitr (Gandrud, 2015; Xie, 2015) are capable of incorporating executable scripts into a LaTeX document in an effort to enhance its reproducibility (see e.g. (Hinsen, 2011, 2015) for other approaches towards enhancing a document’s reproducibility). When the LaTeX markup is compiled, these scripts are replaced by their output (e.g. a figure or a numerical value) in the final PDF document. However, the underlying data dependencies and scripts would also need to be made available to ensure that the figures are reproducible by the wider scientific community in addition to the document’s author. It may be more desirable to embed the scripts and data within the figures, and subsequently embed those figures into the final PDF file, in order to form a standalone reproducible document.

This paper introduces a new digital curation tool, called Pynea (currently in the prototype stage), for improving the reproducibility of data-driven figures within LaTeX documents. By specifying the paths to the plotting script and data files required to generate a given figure within the LaTeX document’s markup, Pynea automatically regenerates the figure with these files embedded such that it can be reproduced in the future. The specific command used to execute the plotting script is also included within the figure’s metadata. Moreover, if the plotting script is tracked with the Git version control system (Chacon & Straub, 2014) then the specific version of that script is also noted within the metadata. The figures are then subsequently embedded within the final PDF document to achieve a standalone document with improved reproducibility.

```
\begin{figure}
  \includegraphics{/home/christian/my_paper/images/plot.pdf}
  \pyneascript{/home/christian/my_scripts/plot.py}
  \pyneacommand{python3 plot.py}
  \pyneadata{/home/christian/my_data/data1.txt /home/christian
      ↪ /my_data/data2.txt}
\end{figure}
```

Figure 1. Example usage of Pynea's macros applied to a single figure environment in a LaTeX source file. The path to the figure itself is specified first, followed by three macro commands to specify the path to the script, the command used to execute it, and the paths to its two data files.

Report Structure

The 'Method' section presents the software architecture behind Pynea, including a discussion of the design choices made, any limitations of the approach, and information regarding how to obtain the software. An example application that showcases the key features of Pynea is provided in the 'Application' section. The paper finishes with some conclusions and recommendations.

Method

Pynea is a standalone Python application that comprises several stages. Put briefly, it first ensures that any figures to be included in the LaTeX manuscript are up-to-date and embeds the relevant plotting/analysis script (hereafter referred to as the script) and data files. It then invokes a suitable LaTeX compiler, such as pdflatex[8] (Thành, 2000), to collate the figures and generate the final PDF manuscript. Pynea subsequently attaches the figure files themselves to the manuscript. However, as a prerequisite step, the user must specify the paths/locations to the plotting script, the command used to run it, and also any data files it depends on. This is achieved by adding Pynea-specific macros to the LaTeX markup, demonstrated in Figure 1, which are defined as follows:

- `\pyneascript{plot.py}` tells Pynea that the figure was generated using a script called `plot.py`. Note that scripts written in other languages can also be executed, as long as the figure output is in PDF format.
- `\pyneacommand{python3 plot.py}` tells Pynea that the script is to be executed using Python 3. Note that any paths provided here are relative to the directory containing the script file.
- `\pyneadata{data1.dat data2.dat}` tells Pynea that the script depends on two data files, `data1.dat` and `data2.dat`.

[8] pdfTeX - TeX Users Group: http://www.pdftex.org/

Following this prerequisite step, Pynea can be run at the command line with the user specifying the path to the LaTeX source file. Figure 2 illustrates the key tasks that are subsequently performed.

The example LaTeX source file, denoted `example.tex` in Figure 2, is first parsed using the third-party TexSoup[9] module. This extracts information about each figure, namely the expected path to the figure file itself (`plot.pdf` in Figure 2) and the information provided in the three macros.

Each figure is then considered in turn and may be regenerated prior to the LaTeX manuscript being compiled. This regeneration step ensures that the most recent version of the plotting script and data files have been used to produce it. Furthermore, the user can be confident that the same command specified in the LaTeX file is the one that has been used to execute the script and produce the figure. After a figure has been regenerated, it is enriched by using the PyMuPDF[10] module to embed the following files and metadata for future reference:

- The script (*embedded file*).
- The data files (*embedded files*).
- The plotting script's execution command (*metadata*).
- If the script is under Git version control, the version of the script that was used. This takes the form of a SHA-1 hash from Git's revision log. (*metadata*).

All of these embedded resources help to improve each figure's reproducibility. However, in the age of Big Data (Chen & Zhang, 2014), regenerating a figure can potentially be a computationally- and memory-intensive operation. A figure is therefore regenerated only if the file does not already exist, if the script or data files have been modified in the Git repository (determined using the GitPython[11] module), or if the command used to execute the script has changed. If the script or data files are not under Git version control, Pynea errs on the side of caution and regenerates the figure each time. If the figure has not been modified, the existing version is used. Note also that only PDF figures are currently supported by Pynea such that figures in other formats are always left unchanged.

After all supported figures have been enriched with their script file, data files, and command metadata, the LaTeX compiler is invoked to produce the manuscript in PDF format. However, while the figures will be rendered in their placeholders as expected, the compilation stage ignores the embedded files. For this reason, the raw figure files are also attached separately to the manuscript as a final step in Pynea to ensure the script and data files remain accessible. The final PDF manuscript therefore has two layers of embedded files:

1. The inner layer comprises the scripts and data files (`plot.py`, `data1.dat`, `data2.dat` in Figure 2) embedded in each figure file.
2. The outer layer comprises the enriched figure files (`plot.pdf` in Figure 2) attached to the PDF manuscript.

[9] GitHub - alvinwan/TexSoup: https://github.com/alvinwan/TexSoup
[10] GitHub - rk700/PyMuPDF: https://github.com/rk700/PyMuPDF
[11] GitHub - gitpython-developers/GitPython: https://github.com/gitpython-developers/GitPython

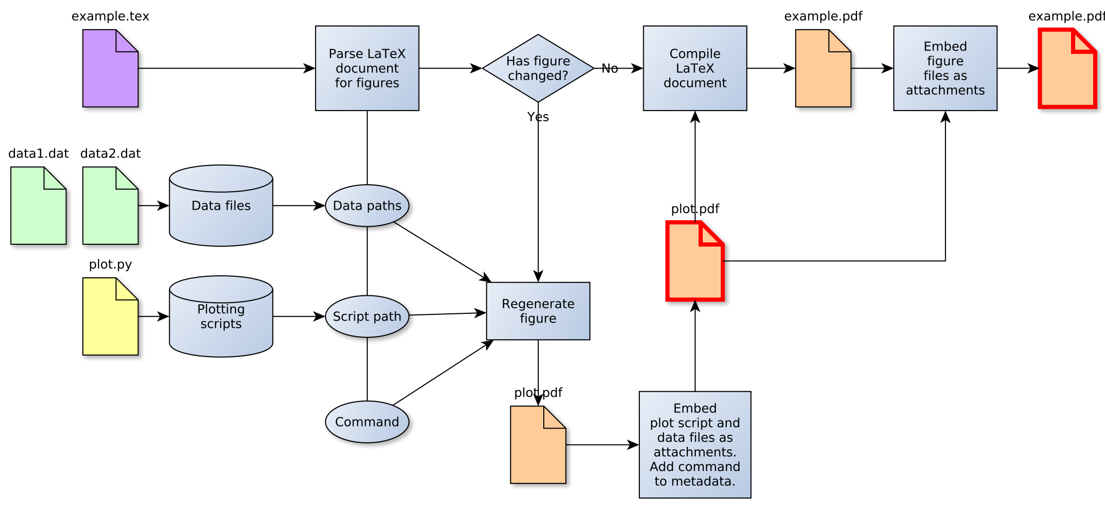


Figure 2. A flowchart outlining the key stages of Pynea. A LaTeX document containing only a single figure is assumed here for simplicity, but Pynea is capable of handling multiple figures. A red outline emphasises the fact that the file is enriched with embedded files.

Opening the resulting PDF manuscript in a suitable PDF reader, such as Adobe Reader, will display the embedded files in the outer layer. These can be extracted/saved manually. Note that embedding the script and data files in the figures themselves, rather than directly in the PDF manuscript, allows the figures to be reused elsewhere in a reproducible fashion (e.g. in slidedecks for conference presentations).

It is worth noting that Big Data poses a challenge for the approach taken by Pynea, since the embedding of large data files would make the manuscript inconvenient to download, share and store, especially if the reader does not wish to reproduce the figures within. When an author is ready to compile the final version of the manuscript, it may be more appropriate to instead submit the data dependencies (identified by the file names provided in the LaTeX source) to a dedicated online repository and embed the repository's Digital Object Identifier (DOI)[12] and version information in the figures' metadata. Readers of the manuscript can then download the data separately, should they wish to do so. However, care must be taken during this curation step in order to guarantee that the version of the data accessible to readers is the same version used by the author to generate the manuscript's figures. Pynea would ideally automate this curation stage for the author, facilitated by tools such as PyRDM (Jacobs et al., 2014) which can interface with repository hosting services. Such an option of using online repositories is not possible in the current prototype of Pynea, but forms part of the project's roadmap discussed in the Conclusions section.

Pynea is freely available to download under the open-source MIT licence from GitHub[13].

[12] Digital Object Identifier System: https://www.doi.org/
[13] GitHub - ctjacobs/pynea: https://github.com/ctjacobs/pynea

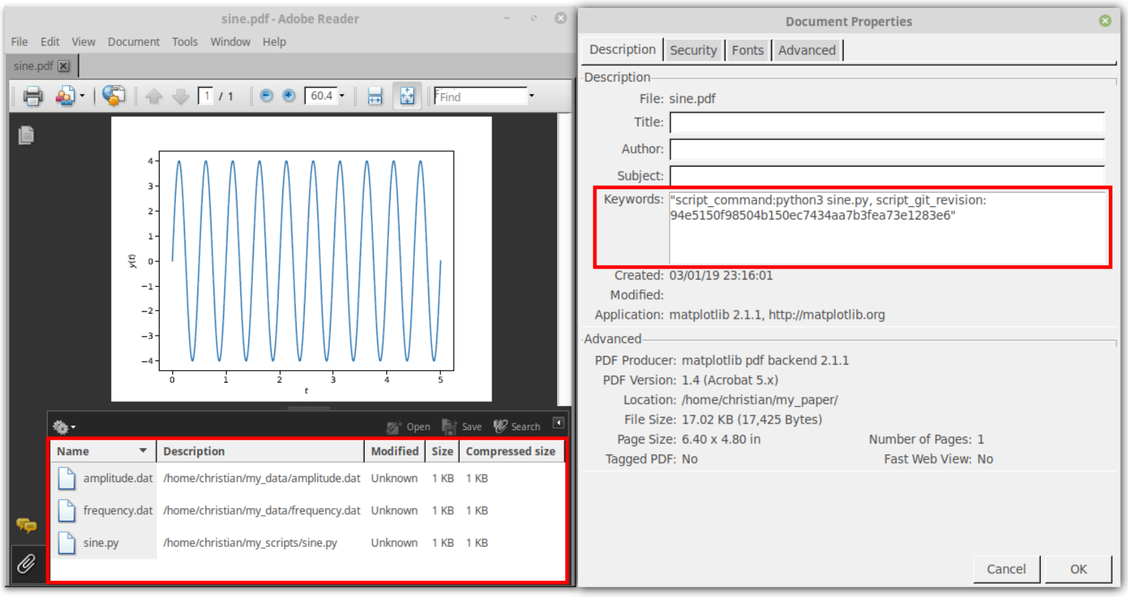
```
\documentclass{article}
\usepackage{graphicx}
\usepackage{pynea}

\begin{document}

\section{Introduction}
This is an example of a LaTeX document containing an enriched figure
   ↪ file.

\begin{figure}[!ht]
   \begin{center}
      \includegraphics[width=\columnwidth]{/home/christian/my_paper/
         ↪ sine.pdf}

      \pyneascript{/home/christian/my_scripts/sine.py}
      \pyneacommand{python3 sine.py}
      \pyneadata{/home/christian/my_data/amplitude.dat /home/
         ↪ christian/my_data/frequency.dat}

      \caption{A sine wave.}
   \end{center}
\end{figure}

\end{document}
```

Figure 3. The contents of the LaTeX source file, `paper.tex`, used in the sine wave application.

Application

To demonstrate the outcomes of Pynea, consider a Python script which plots a time-dependent sine wave $y(t) = A\sin(2\pi f t)$, where A is the peak amplitude and f is the frequency. The plotting script reads in data from two files, `amplitude.dat` and `frequency.dat`, for the quantities A and f respectively. The resulting figure is included within a manuscript using the LaTeX markup in Figure 3.

After Pynea is run, Figure 4 shows how the script and data files are successfully embedded in the `sine.pdf` figure via the Attachments tab/feature. User-defined fields are not permitted within a PDF's metadata. Therefore, Pynea adds an entry in the `keywords` metadata field instead (also shown in Figure 4), which includes the Git revision of the script and the Python command used to execute it. Similarly, this figure file is attached to the final PDF manuscript as demonstrated in Figure 5.



Name	Description	Modified	Size	Compressed size
amplitude.dat	/home/christian/my_data/amplitude.dat	Unknown	1 KB	1 KB
frequency.dat	/home/christian/my_data/frequency.dat	Unknown	1 KB	1 KB
sine.py	/home/christian/my_scripts/sine.py	Unknown	1 KB	1 KB

Description
File: sine.pdf
Title:
Author:
Subject:
Keywords: "script_command:python3 sine.py, script_git_revision: 94e5150f98504b150ec7434aa7b3fea73e1283e6"
Created: 03/01/19 23:16:01
Modified:
Application: matplotlib 2.1.1, http://matplotlib.org

Advanced
PDF Producer: matplotlib pdf backend 2.1.1
PDF Version: 1.4 (Acrobat 5.x)
Location: /home/christian/my_paper/
File Size: 17.02 KB (17,425 Bytes)
Page Size: 6.40 x 4.80 in   Number of Pages: 1
Tagged PDF: No   Fast Web View: No

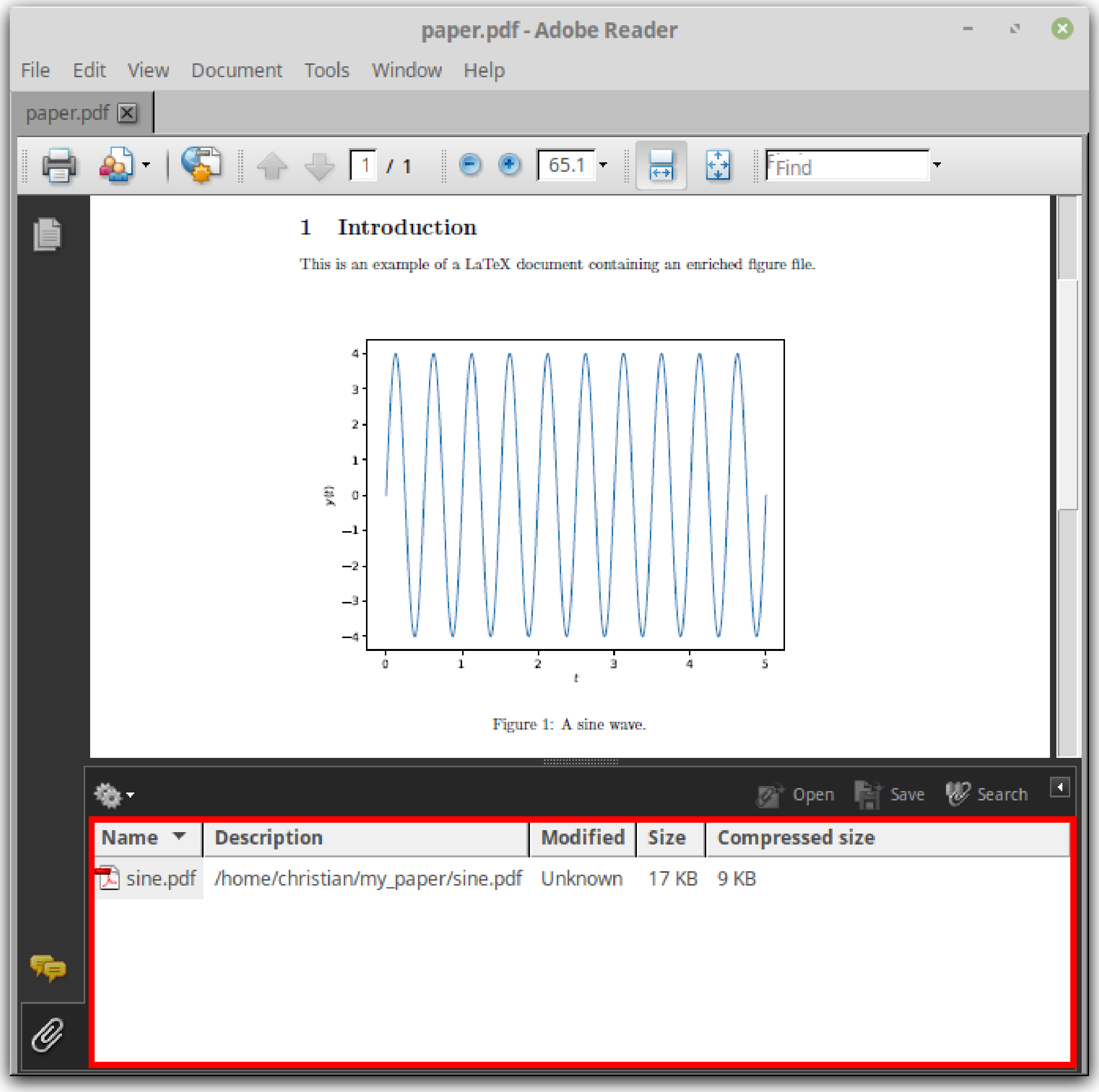
Figure 4. The figure file, `sine.pdf`, viewed using Adobe Reader. The script and data files are listed beneath the sine wave plot in the Attachments tab. The metadata is also shown in the Document Properties dialog window on the right-hand side.

Conclusions

The creation of digital curation tools, such as Pynea, is a crucial step towards a more open and reproducible scientific workflow. However, the success of such tools will be largely dependent on their uptake and support by academia, industry, and publishers. Furthermore, a number of issues and caveats were identified during the development of this work which will need to be addressed to improve usability. For example, one caveat is the inability to create user-defined/custom metadata fields through PyMuPDF. Pynea works around this by appending the relevant metadata to the `keywords` field, but this is of course not ideal. Moreover, few figure formats allow for the embedding of files in a straight-forward manner, and it may not always be practical to save every figure as a PDF.

Restrictions that may be imposed by publishing bodies regarding the inclusion of embedded files can present additional challenges. For example, the embedding of large data files will in turn increase the figure's file size, for which there may be an upper limit imposed by the publisher. Furthermore, when a journal article is being typeset using the publisher's own template, embedded files may automatically be stripped out; particularly if PDF files are 'flattened', converted to other formats, or rendered on the journal's website using a custom/bespoke PDF viewer, which may not support embedded files. Pynea also requires the use of custom macros, which may not be readily available in the publisher's template, although this will be less of an issue if the publisher manually embeds the enriched figure files provided by the author to the final typeset manuscript instead.

Pynea is currently in a prototype stage. The roadmap for Pynea includes the addition of a suite of tests in order to verify the functionality of Pynea and cover a range of use cases. Users are encouraged to submit any bug reports via Pynea's GitHub repository, and collaboratively submit improvements to the codebase via merge requests.



1 Introduction

This is an example of a LaTeX document containing an enriched figure file.

Figure 1: A sine wave.

Name	Description	Modified	Size	Compressed size
sine.pdf	/home/christian/my_paper/sine.pdf	Unknown	17 KB	9 KB

Figure 5. The compiled LaTeX document, `paper.pdf`, viewed using Adobe Reader. The individual figure file, which in turn contains the files shown in Figure 4, is listed in the Attachments tab.

Further improvements that could also be made to Pynea include the submission of data dependencies and plotting scripts to online repositories once the final version of the manuscript is ready to be compiled by the author. Instead of embedding the data and scripts directly into the compiled manuscript's figures, the repository's DOI and version information could be included in the metadata instead.

Acknowledgements

The author wishes to thank Neil Stapleton and an anonymous reviewer for their constructive feedback regarding this manuscript.

References

Adobe Systems Incorporated. (2006). *PDF Reference, sixth edition: Adobe Portable Document Format version 1.7* (6th). Adobe Systems Incorporated.

Allison, D. B., Brown, A. W., George, B. J. & Kaiser, K. A. (2016). Reproducibility: A tragedy of errors. *Nature*, *530*, 27–29. doi:10.1038/530027a

Baker, M. (2016). Is there a reproducibility crisis? *Nature*, *533*, 452–454. doi:10.1038/533452a

Chacon, S. & Straub, B. (2014). *Pro git* (2nd). Apress.

Chen, C. L. P. & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, *275*, 314–347. doi:10.1016/j.ins.2014.01.015

de Leeuw, J. (2001). *Reproducible research. The bottom line*. UCLA: Department of Statistics, UCLA. Retrieved from https://escholarship.org/uc/item/9050x4r4

Donoho, D. L., Maleki, A., Rahman, I. U., Shahram, M. & Stodden, V. (2009). Reproducible research in computational harmonic analysis. *Computing in Science Engineering*, *11*(1), 8–18. doi:10.1109/MCSE.2009.15

Easterbrook, S. M. (2014). Open code for open science? *Nature Geoscience*, *7*, 779–781. doi:10.1038/ngeo2283

Economic & Social Research Council (ESRC). (2017). *ESRC Research Funding Guide*. Economic & Social Research Council (ESRC). Retrieved from https://esrc.ukri.org/files/funding/guidance-for-applicants/research-funding-guide/

Gandrud, C. (2015). *Reproducible research with R and R Studio* (2nd). CRC Press.

Hinsen, K. (2011). A data and code model for reproducible research and executable papers. *Procedia Computer Science*, *4*, 579–588. doi:10.1016/j.procs.2011.04.061

Hinsen, K. (2015). ActivePapers: A platform for publishing and archiving computer-aided research. *F1000Research*, *3*. doi:10.12688/f1000research.5773.3

Jacobs, C. T., Avdis, A., Gorman, G. J. & Piggott, M. D. (2014). PyRDM: A Python-based library for automating the management and online publication of scientific software and data. *Journal of Open Research Software*, *2*, e28. doi:10.5334/jors.bj

Lamport, L. (1994). *LaTeX - User's Guide and Reference Manual* (2nd). Addison-Wesley Publishing Company.

Leeper, T. J. (2014). Archiving reproducible research with R and Dataverse. *The R Journal*, *6*(1), 151–158.

Leisch, F. (2002). Sweave, Part I: Mixing R and LaTeX. *R News*, *2*(3), 28–31.

LeVeque, R. J., Mitchell, I. M. & Stodden, V. (2012). Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science Engineering*, *14*(4), 13–17. doi:10.1109/MCSE.2012.38

Stodden, V., Bailey, D. H., Borwein, J., LeVeque, R. J., Rider, W. & Stein, W. (2013). Setting the default to reproducible: Reproducibility in computational and experimental mathematics. In *Proceedings of the icerm workshop on reproducibility in computational and experimental mathematics, 10–14 december 2012*.

Thành, H. T. (2000). *Micro-typographic extensions to the TeX typesetting system* (Doctoral dissertation, Masaryk University).

Vandewalle, P. (2012). Code sharing is associated with research impact in image processing. *Computing in Science Engineering*, *14*(4), 42–47. doi:10.1109/MCSE.2012.63

von Krogh, G. & Spaeth, S. (2007). The open source software phenomenon: Characteristics that promote research. *The Journal of Strategic Information Systems*, *16*(3), 236–253. doi:10.1016/j.jsis.2007.06.001

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd). CRC Press.