PeerJ Preprints NOT PEER-REVIEWED

Software vs. data in the context of citation

Daniel S. Katz[1], Kyle E. Niemeyer[2], Arfon M. Smith[3], William L. Anderson[4], Carl Boettiger[5], Konrad Hinsen[6], Rob Hooft[7], Michael Hucka[8], Allen Lee[9], Frank Löffler[10], Tom Pollard[11], and Fernando Rios[12]

[1]**National Center for Supercomputing Applications & Electrical and Computer Engineering Department & School of Information Sciences, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; d.katz@ieee.org; ORCID: 0000-0001-5934-7525**
[2]**School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, Oregon, USA; kyle.niemeyer@oregonstate.edu; ORCID: 0000-0003-4425-7097**
[3]**Space Telescope Science Institute, Baltimore, Maryland, USA; arfon@stsci.edu; ORCID: 0000-0002-3957-2474**
[4]**School of Information, University of Texas at Austin, Austin, Texas, USA; band@acm.org; ORCID: 0000-0003-3200-7947**
[5]**Department of Environmental Science, Policy, and Management, University of California, Berkeley, California, USA; cboettig@berkeley.edu; ORCID: 0000-0002-1642-628X**
[6]**Centre de Biophysique Moléculaire (CNRS), Orléans, France; konrad.hinsen@cnrs.fr; ORCID: 0000-0003-0330-9428**
[7]**Dutch Techcentre for Life Sciences; Utrecht, The Netherlands; rob.hooft@dtls.nl; ORCID: 0000-0001-6825-9439**
[8]**Computing and Mathematical Sciences, California Institute of Technology, Pasadena, California, USA; mhucka@caltech.edu; ORCID: 0000-0001-9105-5960**
[9]**Center for Behavior, Institutions & the Environment, Biosocial Complexity Initiative, Arizona State University, Tempe, Arizona, USA; allen.lee@asu.edu; ORCID: 0000-0002-6523-6079**
[10]**Center for Computation & Technology, Louisiana State University, Baton Rouge, Louisiana, USA; knarf@cct.lsu.edu; ORCID: 0000-0001-6643-6323**
[11]**Institute for Medical Engineering & Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA; tpollard@mit.edu; ORCID: 0000-0002-5676-7898**
[12]**Data Management Services, The Sheridan Libraries, Johns Hopkins University, Baltimore, Maryland, USA; rios@jhu.edu; ORCID: 0000-0001-6262-3260**

Corresponding author:
Daniel S. Katz[1]

Email address: d.katz@ieee.org

ABSTRACT

Software is data, but it is not just data. While “data” in computing and information science can refer to anything that can be processed by a computer, software is a special kind of data that can be a creative, executable tool that operates on data. However, software and data are similar in that they both traditionally have not been cited in publications. This paper discusses the differences between software and data in the context of citation, by providing examples and referring to evidence in the form of citations.

INTRODUCTION

This preprint is a snapshot of the GitHub repository (Katz et al., 2016b). The repository is intended to be used to discuss and document the differences between software and data in the context of citation in the research record. This preprint allows the current state of this work to be cited.

The repository was created as part of the FORCE11 Software Citation Working Group (FORCE11 Software Citation Working Group, 2016) writing the FORCE11 Software Citation Principles (Smith et al., 2016a), and then the editors submitting them to PeerJ Computer Science (Smith et al., 2016b) and responding to reviewer comments.

We start with the idea that software, while similar to data in terms of not traditionally having been cited in publications, is also different than data. In the context of research (e.g., in science), the term "data" usually refers to electronic records of observations made in the course of a research study ("raw data") or to information derived from such observations by some form of processing ("processed data"), as well as the output of simulation or modeling software ("simulated data"). In the following, we use the term "data" in this specific sense.

The confusion about the distinction between software and data comes in part from the much wider sense that the term "data" has in computing and information science, where it refers to anything that can be processed by a computer. In that sense, software is just a special kind of data.

The remainder of this document gives examples of these differences. The format of this document is a series of statements (as subsections), each with an explanation, including or followed by evidence in the form of citations.

Updating this document

To add a new difference between software and other forms of data, please submit a pull request in the repository (Katz et al., 2016b). Similarly, to add or update a citation or to add a new explanation, please also do this via a pull request. To discuss a difference (for example, you don't think it is correct), please open a new issue or discuss via an existing issue, again in the repository (Katz et al., 2016b). If you do add text in a pull request, also add yourself as an additional author in that same request. Once there is a significant set of changes, this preprint will be updated.

LIST OF DIFFERENCES

Software is executable, data is not

A commonsense definition of software is that it is "a set of instructions that direct a computer to do a specific task" (Chun, 2005). On the other hand, data is simply a collection of facts or measurements (real or simulated). In other words, software is functionally active, while data is passive. Of course, software (in form) can be considered data as well, especially to functional programmers familiar with LISP and other languages with homoiconicity (Wikipedia, 2016b). However, from the point of view of conducting research with software, the main difference is that software is associated with action: knowledge creation, information transformation, visualization, etc. An action can be thought of a functional transformation between two states of data: a "before" (e.g., input files, parameter settings, unstructured or tacit information) to an "after" state (e.g., output files, transformed data, structured knowledge). That is, software generally performs a function upon something (e.g., software processes data), while data generally has a function performed upon it (e.g., data is processed by software). If we accept the definitions of software and data given at the beginning of this section, then (at least in scientific research), the difference between data and software can be summarized by the statement of Matthews et al. (2010): "we are more interested in what software does rather than what software is."

Data provides evidence, software provides a tool

Software exists to perform a task, while data does not. Software is fundamentally a logical construct, while data is fundamentally an empirical observation. Software can be used to express or explain processes and concepts, oftentimes with data as input. These differences have important consequences for how each may be re-used in the future: software may be used by any researchers seeking to apply the same methods, data by any researchers seeking evidence about the same facts.

Software is a creative work, scientific data are facts or observations

In particular, software is generally subject to copyright protection as a creative work that can continue to evolve over time, while scientific data is frequently considered outside the domain of copyright as it is comprised of contextual facts about the world (you cannot copyright the height of Mt. Everest.) Major scientific data repositories (e.g., Dryad, figshare) automatically apply licenses suited to data that may not be suited to software.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2630v1 | CC BY 4.0 Open Access | rec: 10 Dec 2016, publ: 10 Dec 2016

Evidence: Can I apply a Creative Commons license to software? (Creative Commons, 2016); Non-software licenses (Choose an open source license, 2016)

Software suffers from a different type of bit rot than data: It is frequently built to use other software, leading to complex dependencies, and these dependent software packages also frequently change

In general, software must be constantly maintained and updated in order to continue to function as both the hardware and software environments on which it depends change. Operating systems, software and system libraries, programming language toolchains and other compile-time and run-time dependencies all evolve as their respective maintainers and developers find and fix bugs, and as user requirements demand new features and capabilities. This is sometimes called "software rot" (Wikipedia, 2016c) and other times called "bit rot." On the other hand, bit rot for data, or data degradation (Wikipedia, 2016a), is generally thought of as changes in the underlying hardware or storage media that holds the bits, or changes in the software capable of interpreting the data. This definition of bit rot also affects software since software is actually stored as a set of bits on a filesystem, but software bit rot is generally thought of as a higher level concern than data- or file-level bit rot.

The lifetime of software is generally not as long as that of data

The lifetime of software can reach 20 years or more, especially for well-maintained projects. The life of software can end if the task it was supposed to do is not needed anymore, or if another software does it in a better way. Data, on the other hand, often represents the results of an experiment. It might become less interesting with time, but it cannot be replaced as it is connected to one particular experiment at that particular time. In this sense, software is replaceable (by other software), while data is usually not.

A 1995 NRC Report Preserving Scientific Data on Our Physical Universe (National Research Council, 1995) provides the following recommendations regarding retention criteria and the appraisal process (p. 40): "As a general rule, all observational data that are nonredundant, useful, and documented well enough for most primary uses should be permanently maintained. Laboratory data sets are candidates for long-term preservation if there is no realistic chance of repeating the experiment, or if the cost and intellectual effort required to collect and validate the data were so great that the long-term retention is clearly justified. For both observational and experimental data, the following retention criteria should be used to determine whether a data set should be saved: uniqueness, adequacy of documentation (metadata), availability of hardware to read the data records, cost of replacement, and evaluation by peer review. Complete metadata should define the content, format or representation, structure, and context of a data set."

While software is often replaced by newer software, computational models and data analyses can be important digital artifacts that should be preserved (Rollins et al., 2014) along with datasets in order to properly verify or reproduce (Peng, 2011) published findings. Long-term preservation of the software used in computational science is a wicked problem as outlined in the final report from the Preserving.exe: Toward a National Strategy for Preservation Software 2013 meeting (Library of Congress, 2013). At that time, best practices to facilitate reproducibility of computational science involve archiving of the following, in durable, plaintext formats:

1. the software itself, in source code form in a trusted digital repository
2. structured or unstructured narrative documentation (e.g., the ODD protocol (Grimm et al., 2013)) specifically covering key components of the software
3. descriptive provenance metadata on the software dependencies needed to compile and run the software as well as any input data dependencies

though these practices may change as virtualization and containerization become more common.

ACKNOWLEDGMENTS

The initial version of the text in this document was a list in an early version of (Smith et al., 2016b), which was developed by the FORCE11 Software Citation Working Group. The members of this working group were an initial set of volunteers who were joined by members of a related WSSSPE2 working

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2630v1 | CC BY 4.0 Open Access | rec: 10 Dec 2016, publ: 10 Dec 2016

group (Katz et al., 2016a). During the review process, it became clear having this text in that document was a distraction for the reviewers, so it was extracted into a repository (Katz et al., 2016b) where it continued to develop. As such, both FORCE11 (The Future of Research Communication and eScholarship; https://www.force11.org) and WSSSPE (Working towards Sustainable Software for Science: Practice and Experiences; http://wssspe.researchcomputing.org.uk) are acknowledged as instigating this discussion.

REFERENCES

Choose an open source license (2016). Non-software licenses. URL: http://choosealicense.com/non-software/ [Accessed: 2016-12-05].

Chun, W. H. K. (2005). On software, or the persistence of visual knowledge. *Grey Room*, pages 26–51. DOI:10.1162/1526381043320741.

Creative Commons (2016). Can i apply a creative commons license to software? URL: https://wiki.creativecommons.org/index.php/Frequently_Asked_Questions#Can_I_apply_a_Creative_Commons_license_to_software.3F [Accessed: 2016-12-05].

FORCE11 Software Citation Working Group (2016). FORCE11 software citation working group GitHub repository. URL: https://github.com/force11/force11-scwg [Accessed: 2016-12-05].

Grimm, V., Polhill, G., and Touza, J. (2013). Documenting social simulation models: The ODD protocol as a standard. In Edmonds, B. and Meyer, R., editors, *Simulating Social Complexity: A Handbook*, pages 117–133. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI:10.1007/978-3-540-93813-2_7.

Katz, D. S., Choi, S.-C. T., Wilkins-Diehr, N., Chue Hong, N., Venters, C. C., Howison, J., Seinstra, F. J., Jones, M., Cranston, K., Clune, T. L., de Val-Borro, M., and Littauer, R. (2016a). Report on the second workshop on sustainable software for science: Practice and experiences (WSSSPE2). *Journal Open Research Software*, 4(1):e7. DOI:10.5334/jors.85.

Katz, D. S., Niemeyer, K. E., Smith, A. M., Anderson, W. L., Boettiger, C., Hinsen, K., Hucka, M., Lee, A., Löffler, F., Pollard, T., and Rios, F. (2016b). Software vs. data GitHub repository. URL: https://github.com/danielskatz/software-vs-data [Accessed: 2016-12-05].

Library of Congress (2013). Preserving.exe: Toward a national strategy for software preservation. URL: http://www.digitalpreservation.gov/multimedia/documents/PreservingEXE_report_final101813.pdf [Accessed: 2016-12-05].

Matthews, B., Shaon, A., Bicarregui, J., and Jones, C. (2010). A framework for software preservation. *International Journal of Digital Curation*, 5(1):91?105. DOI:10.2218/ijdc.v5i1.145.

National Research Council (1995). *Preserving Scientific Data on Our Physical Universe: A New Strategy for Archiving the Nation's Scientific Information Resources*. National Academies Press. URL: http://www.nap.edu/catalog/4871.html [Accessed: 2016-12-05].

Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227. DOI:10.1126/science.1213847.

Rollins, N. D., Barton, C. M., Bergin, S., Janssen, M. A., and Lee, A. (2014). A computational model library for publishing model documentation and code. *Environmental Modelling & Software*, 61:59 – 64. DOI:10.1016/j.envsoft.2014.06.022.

Smith, A. M., Katz, D. S., Niemeyer, K. E., and FORCE11 Software Citation Working Group (2016a). Software citation principles. URL: https://www.force11.org/software-citation-principles [Accessed: 2016-12-05].

Smith, A. M., Katz, D. S., Niemeyer, K. E., and FORCE11 Software Citation Working Group (2016b). Software citation principles. *PeerJ Computer Science*, 2:e86. DOI: 10.7717/peerj-cs.86.

Wikipedia (2016a). Data degradation. URL: https://en.wikipedia.org/wiki/Data_degradation [Accessed: 2016-12-05].

Wikipedia (2016b). Homoiconicity. URL: https://en.wikipedia.org/wiki/Homoiconicity [Accessed: 2016-12-05].

Wikipedia (2016c). Software rot. URL: https://en.wikipedia.org/wiki/Software_rot [Accessed: 2016-12-05].

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2630v1 | CC BY 4.0 Open Access | rec: 10 Dec 2016, publ: 10 Dec 2016