Delft University of Technology

Comment on “Most computational hydrology is not reproducible, so is it really science?” by Christopher Hutton et al.

Let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel ourselves

Hut, R. W.; van de Giesen, N. C.; Drost, N

DOI
10.1002/2017WR020665

Publication date
2017

Document Version
Final published version

Published in
Water Resources Research

Citation (APA)
Hut, R. W., van de Giesen, N. C., & Drost, N. (2017). Comment on “Most computational hydrology is not reproducible, so is it really science?” by Christopher Hutton et al. Let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel ourselves. *Water Resources Research*, *53*(5), 4524-4526. https://doi.org/10.1002/2017WR020665

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

COMMENT

10.1002/2017WR020665

This article is a companion to *Hutton et al.* [2016], doi:10.1002/2016WR019285.

Key Points:
- Storing software code and data is not enough to guarantee reproducibility
- We suggest to use container technology to guarantee reproducibility
- Work closely together with Research Software Engineers (RSEs)

Correspondence to:
R. W. Hut,
r.w.hut@tudelft.nl

Citation:
Hut, R. W., N. C. van de Giesen, and N. Drost (2017), Comment on "Most computational hydrology is not reproducible, so is it really science?" by Christopher Hutton et al.: Let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel ourselves, *Water Resour. Res.*, *53*, 4524–4526, doi:10.1002/2017WR020665.

Received 2 MAR 2017
Accepted 26 MAR 2017
Accepted article online 31 MAR 2017
Published online 11 MAY 2017

Comment on "Most computational hydrology is not reproducible, so is it really science?" by Christopher Hutton et al.: Let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel ourselves

R. W. Hut[1], N. C. van de Giesen[1], and N. Drost[2]

[1]Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, Netherlands, [2]Netherlands eScienceCenter, Amsterdam, Netherlands

Abstract The suggestions by *Hutton et al.* might not be enough to guarantee reproducible computational hydrology. Archiving software code and research data alone will not be enough. We add to the suggestion of *Hutton et al.* that hydrologists not only document their (computer) work, but that hydrologists use the latest best practices in designing research software, most notably the use of containers and open interfaces. To make sure hydrologists know of these best practices, we urge close collaboration with Research Software Engineers (RSEs).

1. Introduction

With an eye-catching title and a well thought out analysis, *Hutton et al.* [2016] raise the question "Most computational hydrology is not reproducible, so is it really science?" Luckily for most computational hydrologists, they conclude that computational hydrology is not in itself unscientific, but rather that the current practices within computational hydrology lead to unscientific behavior. Please note that the unscientific behavior that *Hutton et al.* point at arises from historical limitations on computer resources that are no longer valid, but that are still reflected in the scientific culture within hydrology, and from the incentive structure for publishing academic articles. Like *Hutton et al.*, we do not intend to accuse individual scientists of deliberate unscientific actions).

As a solution *Hutton et al.* propose a list of actions computational hydrologists (and journal editors) should take to move toward a more reproducible, more scientific, field of computational hydrology. We strongly agree with every point on the action list of *Hutton et al.* On some issues we believe that the solutions proposed by *Hutton et al.* might not go far enough to achieve the desired result of more scientific rigor in computational hydrology. We would like to expand on the list of *Hutton et al.* with three actions that are complementary. Specifically, we suggest two practical tools to operationalize their suggestions, using recent advances in computer science.

2. Reproducibility Across Systems With Containers

Demanding that code and work flows to generate scientific results are well documented and available may not be enough to reproduce hydrological model output. Software libraries, most notably those made by (geo)scientists, are notoriously hard to install and when asked with a command to "install CDO" [*CDO*, 2015], different scientists will end up with different versions of CDO on their computer. These differences in versions may make enough of a difference to break the resulting software. Or, worse, generate subtly different results. With the large number of libraries needed for most scientific work flows, the chances of the resulting system failing or failing to reproduce the result go up dramatically. Ideally, with hydrological articles, we want that reviewers in particular and readers in general can run the analyses on the same environment (both hardware and software) as the original analyses. This is of course unfeasible, but virtual machines (made popular in cloud computing) could provide a solution. It is then important that the

© 2017. The Authors.
This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

provenance, i.e., a record of how the virtual machine itself was created, is available. The recent advent of software containers, most notably the Docker platform, can provide the solution.

A software container is a virtual machine minimized to run a single particular task, such as a hydrological model. A container ships with a setup on how to create the container, including which versions of which libraries to install. This guarantees that a single container, run on different machines, will always produce identical results. Containers were popularized in online applications where different task that serve a website (database, front-end, authorization, etc.) are each run in a separate container. Recently, the Open Containers Initiative is taking the lead in creating an open and free standard for building and running containers.

We suggest that journals start demanding that scientist not only provide the code and work flow that generate their results, but also a container-image that actually generates these results. With these container images, reviewers can more easily vouch that the results are valid and reproducible. And readers can more easily build on the work to generate novel results, speeding up the generation of hydrological knowledge.

3. Connectivity Between Components

Hutton et al. point out that software is not often shared or documented because scientists have little incentive to do so. They suggest, in our opinion correctly, using DOIs for software through services like Zenodo to make software citable and thus give (academic) credit to software creators. As *Hutton et al*. say: “many hydrologists may have written pieces of code that perform very similar tasks that could benefit from reuse.” We believe that sharing, documenting, and making code citable may not be enough to facilitate reuse.

Nowadays almost every website has a zoomable map to locate the business location. This is available to website builders, not only because Google Maps and Open Street maps are well documented, but because they have accessible Application Programming Interfaces (API’s) that allow website builders to use their functionality without having to understand every bit of the code involved. If hydrologists want to be able to use each other’s code, they should use API’s or equivalent methods to ensure that a colleague only needs to read the documentation and not the code itself, to be able to use its functionality. For online services, this means using API’s, for hydrological models, standard model interfaces such as OpenMI [*Donchyts et al*., 2010] or BMI [*Peckham et al*., 2013] can be used.

4. Working With Research Software Engineers

We hope that the two suggestions above will be added to the list of Hutton et al. to move computational hydrology toward a more scientific working method. More importantly, we would urge hydrologists to work directly with those skilled in creating research software. Though historically lacking any organization and even a proper name, recently, a community of Research Software Engineers (RSEs) has been formed [*RSE*, 2016]. As they themselves put it: “The people behind research software, combining expertise in programming with an intricate understanding of research.” Authors Hut and van de Giesen learned about open interfaces and container-images by working closely together with author Drost, who is a RSEs in the eWaterCycle project [*Hut et al*., 2015]. The experience of author Drost in building flexible modular simulation software for astrophysics [*Pelupessy et al*., 2013] was invaluable to the eWaterCycle project. Where Hutton et al. urge hydrologists to teach themselves and their students about new technologies, we would like to add the importance of working with RSEs with experience outside of hydrology. These cooperations prevent us from reinventing the wheel or, worse, designing square wheels, and allow us to incorporate the latest computer science technologies into computational hydrology.

Acknowledgments
This research is part of the eWaterCycle project. This work is in part funded by the research programme of the Netherlands Organisation for Scientific Research (NWO) under project number MP-272-13. This work is in part funded by the Netherlands eScience Center (http://www.nlesc.nl).

References

CDO (2015), Climate data operators. [Available at http://www.mpimet.mpg.de/cdo.]

Donchyts, G., S. Hummel, S. Vaneek, J. Groos, A. Harper, R. Knapen, J. Gregersen, P. Schade, A. Antonello, and P. Gijsbers (2010), OpenMI 2.0 What’s new, in *Proceedings of 2010 International Congress on Environmental Modelling and Software*, Int. Environ. Modell. and Software Soc., Ottawa, Canada. [Available at http://www.iemss.org/iemss2010/Volume2.pdf.]

Hut, R., N. Drost, E. Sutanudjaja, N. van de Giesen, and M. van Meersbergen (2015), eWaterCycle, EGU, Vienna, Austra. [Available at http://forecast.ewatercycle.org/.]

Hutton, C., T. Wagener, J. Freer, D. Han, C. Duffy, and B. Arheimer (2016), Most computational hydrology is not reproducible, so is it really science?, *Water Resour. Res.*, *52*, 7548–7555, doi:10.1002/2016WR019285.

Peckham, S. D., E. W. H. Hutton, and B. Norris (2013), A component-based approach to integrated modeling in the geosciences: The design of CSDMS, *Comput. Geosci.*, *53*, 3–12, doi:10.1016/j.cageo.2012.04.002.

Pelupessy, F. I., A. v. Elteren, N. d. Vries, S. L. W. McMillan, N. Drost, and S. F. P. Zwart (2013), The astrophysical multipurpose software environment, *Astron. Astrophys.*, *557*, A84, doi:10.1051/0004-6361/201321252.

RSE (2016), UK Research Software Engineer Association. [Available at http://rse.ac.uk/.]