COMMUNICATIONS PHYSICS

COMMENT

https://doi.org/10.1038/s42005-020-00403-4 OPEN

Creating an executable paper is a journey through Open Science

Jana Lasser[1,2]✉

Executable papers take transparency and openness in research communication one step further. In this comment, an early career researcher reports her experience of creating an executable paper as a journey through Open Science.

Open Science practices are taking an increasingly central role in the way we conduct research, from accessible research data to transparency in the methodologies used to analyze them. To this end, the novel "executable paper" format offers a transparent and reproducible way to communicate research. Executable papers are dynamic pieces of software that combine text, raw data, and the code used for the analysis, and that a reader can interact with. In this commentary, I introduce the executable paper format and highlight its advantages for research communication. Drawing from my personal experience, I offer practical advice on how to create an executable paper by using Open Source tools such as Python and Jupyter Notebooks, and how to make it accessible by publishing it on open repositories.

Enhanced by the current Covid-19 pandemic, Open Science practices have taken an even more central role in the way we conduct research:[1] the sequenced genome of the virus has been made publicly available[2] at an unprecedented speed, hundreds of scientists worldwide base their research on the case number data publicly released by the John Hopkins university[3], data on global mobility patterns has been made available by Apple[4] and Google[5], and preprints repositories allow researchers to immediately disseminate results. Science, health, and government-related organizations have been calling for openness and timeliness[6] in sharing research results since the very beginning of the epidemic. As such, researchers across the globe are able to play a very active role in the development of strategies to curb the spread of the virus, and new insights proliferate rapidly. While the transparent sharing of data and results is the first fundamental step to open research, how data has been analyzed to reach results should be as transparent. The UK's response to the outbreak of SARS-CoV2 is a prime example of the pitfalls of intransparent research practices. While scientific evidence informing the initial decision to delay physical distancing measures has never been published[7], - the simulation code[8] supporting the sudden implementation of lock-down measures was made available only after measures were taken, still turning out to be intransparent and not designed to be re-used by external users[9]. Similarly, unreliable and flawed data that were not publicly[10] accessible affected dramatically the World Health Organization's drug research policies on invermectin and hydroxychloroquine[11],

[1] Complexity Science Hub Vienna, Josefstädterstraße 39, 1080 Vienna, Austria. [2] Section for Science of Complex Systems, Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, 1090 Vienna, Austria. ✉email: lasser@csh.ac.at

culminating in the recent retraction from the Lancet of the paper claiming adverse effects of hydroxychloroquine on Covid-19 patients[12]. While the present Covid-19 pandemics has magnified the issue of open and transparent research, the reproducibility crisis is a well known issue within the wider academic community[13,14], and how this could be tackled by implementing Open Science and open research practices is a current matter of debate[14,15].

As researchers, we aim at having a direct and positive impact on our society through our work. However, in order to inform policy, research has to be open, transparent, and reproducible. This does not stop at the data being available, but includes transparency of the methods employed to reach conclusions, as well as an efficient dissemination of the results[16].

A way to support data availability and methods transparency is the "executable paper" format - i.e. formatting a publication as a dynamic piece of software that combines text, raw data, and the code used for the analysis, and that a reader can interact with. While the reader still has to trust that data have been collected properly, being able to directly reproduce the analysis is already more than many classical publications currently allow. As classical journals do not yet accept executable papers as submissions, the executable paper fulfills the role of an extended preprint that exists next to a peer reviewed publication in a journal. Recently, I have been exploring the executable paper format as part of my Wikimedia Open Science fellowship by creating one from scratch for my last work on pattern formation in salt deserts[17]. My executable publication's latest release can be found on GitHub[18]. In the following, I share my experience of creating this executable paper. I will introduce the tools I used (and why), touch on the challenges I encountered (and how I solved them), and outline how working on the executable paper benefited my research practice.

What is an executable paper?

Before I start to dive into the details, I want to explain what an executable paper is (or at least what I think it should be). In principle the idea is to enable the reader to reproduce each and every step taken to arrive at a publication's conclusions, from the raw data to the polished plot. Therefore the paper should be linked to a repository containing the raw data and should support the necessary functionality to make the research process transparent and reproducible. Specifically, the executable paper has to display nicely formatted text, including references and links (just like a journal article), as well as figures, plots, and possibly also videos and interactive elements. Additionally, the analysis code used for creating plots from data should be displayed and interpreted by the executable paper, also allowing for user input to modify the analysis. Lastly, the executable paper should be composed of completely Open Source components and hosted under a free license to be easy to share and reuse.

As I started working on my executable paper, I quickly found out that the whole concept of executable papers was not as established as I thought. There was a short outburst of activity back in 2011, driven by Elsevier[19], but other than that little turns up when searching for the term online. Other than a couple of templates[20], very short guidelines[21], or very detailed but not very broadly applicable how-to's[22], there is currently not much to build on.

A variation of the executable paper is the "reproducible paper"[23], a publication centered around a simulation code that can be run by the reader to generate data and perform their analysis, thus reproducing the whole research process down to the results. The main difference with the executable paper is that, while the executable paper relies on empirical data, the reproducible paper's code generates the data, thus putting a larger emphasis on the description of the code.

Along a similar line, the large crowd of Jupyter Notebook[24] users is driving the development and dissemination of executable documents containing scientific research. There is a growing gallery[25] of amazing examples of scientific publications and books written in Jupyter Notebooks, including a notebook that walks the reader through the process of detecting gravitational waves from LIGO data[26]. Therefore I decided to adopt the Jupyter Notebook as the basic tool for my executable paper.

The document

The Jupyter Notebook document is written in Markdown[27] – a very easy-to-learn markup language that allows one to quickly format text in an easy-to-read and web-friendly way. Important for me as a physicist: Jupyter Notebooks are also able to render equations using LaTeX notation. Since I usually write my publications in LaTeX, this allows me to copy-paste most of the text and equations over to the executable format. Figure 1 shows a small excerpt of the math rendering in the executable paper.

Next to the text, the document is composed of code cells that contain and interpret code from a variety of languages. In Fig. 2 is an example of such a code cell, in which a data set is loaded and displayed as a table. Personally, I believe that being able to see the first few lines of a data set can be very useful, as it immediately gives an intuition of what information the data set contains, and what are the typical values.

The code displayed in this specific cell is Python[29]. I chose Python for my executable document because it has a large user base and a large variety of well-documented and well-maintained libraries for scientific computing. Another advantage of Python is that with a little bit of effort and by writing *readable* instead of *elegant* code, it can be read almost as a description of what the code is actually doing. This allows readers who are unfamiliar with the syntax to follow the steps of the analysis and validate them by running the code themselves. Jupyter Notebooks also support other languages[30], such as R[31], haskell[32], or javascript[33]. In general, the format of the Jupyter Notebook makes the analysis more open and accessible, by allowing readers that are potentially unfamiliar with the programming language to run any code without locally installing any additional software (more on that later under "hosting").

Hosting

A fundamental part of producing an executable paper is making all material, from the actual paper to the data, available to readers

Taking the average evaporation rate E as the natural velocity scale for the system, we set the characteristic length and time as $L = \phi D/E$ and $T = \phi^2 D/E^2$, respectively. Non-dimensionalization of Eqs. 1-3 then gives

$$\nabla \cdot U = 0 \quad (4)$$

$$U = -\nabla P - \mathrm{Ra} S \hat{Z} \quad (5)$$

$$\frac{\partial S}{\partial \tau} = \nabla^2 S - U \cdot \nabla S \quad (6)$$

Fig. 1 Screenshot of a text cell containing symbolic math notations in a Jupyter Notebook. Equations are typeset using LaTeX notation and mathjax[28] and can be referenced using hyperlinks.

```
[29]: # load the spatially resolved concentration data from a
      # dissected experiment
      lab_C_data = pd.read_csv('data/concentration_statistics_experiment.csv',skiprows=[1])
      lab_C_data.columns = ['x','z','water','sand','salt','C']

      lab_C_data.head()
```

[29]:

	x	z	water	sand	salt	C
0	0	0	0.18911	0.65014	0.03111	14.12678
1	4	0	0.28452	1.01313	0.00373	1.29402
2	9	0	0.32237	1.12396	0.03642	10.15078
3	14	0	0.27866	0.90894	0.02253	7.48033
4	19	0	0.21689	0.81797	0.02666	10.94642

Fig. 2 Screenshot of a code cell in which a data table is read and displayed in a Jupyter Notebook. The reader can see which data file is read into the notebook and which columns it has. Executing the code cell reads the data table into a variable and displays the first five rows of the table below the code cell.

by hosting it on an accessible platform. In the future, journals adopting this kind of format will take over all the tasks associated with hosting. So far, however, creating an executable paper presents two challenges in terms of hosting: first all the resources (code, images, and data) need to be accessible to the reader. Second, since the executable paper is not a static artefact, but rather an interactive and dynamic piece of software, it needs to be hosted such that the user can interact with the document, preferably without having to install additional software.

The first challenge is easily solved by using Git and GitHub. Putting the text and code under version control in a Git repository presents additional advantages: it allows to save the history of changes, which can be used to revert back to earlier versions of the document if needed. It also allows for collaborative working on text and code by managing changes from different collaborators. Uploading the code on a remote repository at GitHub is a minor additional step, and has the benefit of creating a backup in a remote server. Git also allows for an easy setup of a project description through the README file, and the creation of a license. Furthermore, it is easy and fast to create a release of the materials and add a persistent DOI, for example by using Zenodo[34]. Ultimately, publishing an executable paper on GitHub is equivalent to publishing a preprint on your own homepage or on a preprint server.

As regards the second challenge, most people would agree that getting a programming language to work locally on one's machine is far from being straightforward. Operating systems would interfere with development environments or compilers, and even producing a simple "hello world" program with a new language could easily take up to an hour. Fortunately, for interpreted programming languages (such as Python) online hosting solutions such as binder[35] are available. GitHub allows for easy integration of binder with a simple badge (just like a license badge). This way, readers can execute the code without the need to download any data or install any software. Large projects (such as my executable paper[18]) can take a couple of minutes to load on binder, but that is a minor tradeoff when compared to the need to locally install a new programming language. So now the technical aspects are covered, I will say a few words about the challenges I encountered when creating the content of the executable paper.

Clean and shareable data

To fulfill the aim of enabling a reader to follow my analysis from the first raw data point to the polished plot, I need to make my raw data accessible. This means raw data has to be well-ordered, well-described, and uploaded somewhere publicly accessible.

Formatting data so that they are accessible is actually extremely time consuming, even when one would consider themself as a well-organized person. In particular, my publication draws on data from experiments, field studies, and simulations, all of which features different formats, sizes, and analysis pipelines that needed to be made consistent. While going through the process, I realized I had made countless implicit assumptions that guided the data processing. Therefore, a part of making my data and analysis open, included giving data sets more telling names, documenting meta-information, polishing, and annotating my analysis scripts.

Particularly relevant points turned out to be (i) using telling and unique file names that describe the data content, and possibly the dates when the data was collected or processed; (ii) documenting all relevant meta-information in a dedicated readme file any time data are modified; (iii) storing a copy of the original raw data; and (iv) having multiple and synchronized backups of the processed data.

Beside being fundamental in terms of Open Science practice, formatting data can bring further benefits. For example, I could publish my reformatted data as six independent datasets on PANGAEA[36–41], and write a dedicated descriptive publication[42]. I definitely learned a lot throughout the process and I am applying what I learned to my new research projects - in the hope that next time it will not take me so much time to get my data in order.

Choosing the right level of detail

Making every single analysis step transparent and reproducible is desirable for the sake of openness. Nevertheless, as for every scientific publication, the level of detail needs to be carefully balanced with the narrative flow in order to ensure readability. To this end, the executable paper format gives more options than a regular publication: Jupyter Notebooks offer collapsing code cells and the possibility to add links to more detailed descriptions or separate analysis scripts. Analysis scripts and additional methods descriptions can also be presented in the supplement of a regular publication. Nevertheless, the strength of the executable paper is to seamlessly collect and combine all relevant information in one master document. This way, following the details necessary to reproduce the research does not require multiple switches between different documents and software, thus becoming more transparent.

In my specific case, I tried to strike a balance between brevity and reproducibility by merging a few important parts of the original appendix and methods into the main part of the article. In addition, I added many links that lead to more detailed explanations. For the more complicated data analysis I created a

separate script that can be downloaded with the main repository and can be re-run to analyze the data from scratch. Since alongside raw data I also provide the already analyzed data, re-running that script is a reader's choice and not a must.

Interactive elements

In my opinion, one of the major benefits of executable papers in terms of supporting the openness of the research process, is the possibility to extend the analysis beyond what is typically reported in a traditional publication. Usually, only a single (often "the successful") way of analyzing the data is described in a paper, and things like sensitivity analysis, or the results obtained with different parameters (for example a cutoff value) are reported in the supplementary information, or omitted altogether. Conversely, an executable paper allows for user input in the code cells, that is, plots can be interactively modified and re-computed, taking the reader input into account. If a certain choice was made for a parameter, the reader can modify the chosen value and check to what extent the conclusions drawn from the analysis still hold.

In my executable paper, I incorporated interactive elements as a sensitivity analysis within the key figure of the manuscript by allowing the user to try out different approaches to calculate the permeability of desert soil. I chose this specific parameter because it has a large impact on the results and because there are many different and ambiguous ways to calculate it. In this way, the figure shows the main result, and at the same time that the choices for the analysis only have a minor impact on the outcome (except for the very extreme cases). I believe such a solidity check not only strengthens the conclusions in the eyes of the reader, but is also important for an author to be more confident about the choices made during the analysis process.

A journey through Open Science

I think that executable papers would greatly improve research standards, especially if they were to be accepted and reviewed by journals just like regular publications. In fact, the executable paper format enforces publishing data and analysis code in a clean and well-documented way alongside the article. Preparing such polished material takes time, but also gives an additional chance of catching mistakes, and leaves it ready-to-use and reuse for future research. Moreover, the additional transparency makes it hard to "hide" sloppiness in the analysis process or even fraudulent practices.

Since – just like regular readers – reviewers of executable papers would not need to install additional software, the technological barrier to reviewing executable papers is minimal. Actually, weaving together into a seamless environment all components of the research process has the potential to reduce review time: reviewers would not need to attempt reproducing the calculation themselves, as they could check the code straight away, and well explained and documented methods would probably prevent many misunderstandings. Of course, reviewers should have a minimum of knowledge about coding and data standards to be able to judge the quality of all aspects of the publication.

Similar to publishing preprints, publishing an executable paper ahead of a classical journal publication bears the risk of other scientists trying to "steal" ideas. But just like preprints, a DOI can be assigned to executable papers which can be used to prove the origin of the idea in case of a dispute. Similar to publishing a data set in a public repository, publishing data in an executable paper enables other people to use such data for their own research. While this might seem scary to some, I think that it has many benefits: publishing data broadens the context in which the research is useful to other researchers, resulting in more citations or even new possibilities for collaborations.

Creating an executable paper takes a significant amount of time, but I can only recommend the experience. It gave me new confidence about my own research, and forced me to revisit choices I had done in the past, sometimes in a rush and with less understanding than I have today. Creating an executable publication also gave me the chance to improve my research practice by using many open tools and workflows I already knew about, and pulling them together into a more open and transparent way of academic working.

In retrospective, creating an executable paper is a journey through almost all aspects of Open Science: from open and accessible data to open and transparent tools and software, ending with an open publication composed of open and transparent content.

Data availability

All data referenced in this article is available under a CC BY 4.0 license at PANGAEA[36–41].

Code availability

Code referenced in this article is available under a CC-BY-SA 4.0 license at GitHub[18].

Received: 18 June 2020; Accepted: 10 July 2020;
Published online: 19 August 2020

References

1. China coronavirus: how many papers have been published? https://www.nature.com/articles/d41586-020-00253-8 (2020).
2. GISAID - Initiative. https://www.gisaid.org/ (2020).
3. Maps & Trends - Johns Hopkins Coronavirus Resource Center. https://coronavirus.jhu.edu/data (2020).
4. Apple. COVID 19 - Mobility Trends Reports. *Apple* https://www.apple.com/covid19/mobility (2020).
5. Google. COVID-19 Community Mobility Report. *Google* https://www.google.com/covid19/mobility?hl=en.
6. Calling all coronavirus researchers: keep sharing, stay open. *Nature* **578**, 7–7 (2020).
7. Yong, E. The U.K.'s Coronavirus 'Herd Immunity' Debacle. *The Atlantic* https://www.theatlantic.com/health/archive/2020/03/coronavirus-pandemic-herd-immunity-uk-boris-johnson/608065/ (2020).
8. Ferguson, N., Nedjati-Giliani, G. & Laydon, D. *CovidSim Microsimulation Model* (MRC Centre for Global Infectious Disease Analysis, 2020).
9. Boland, H. & Zolfagharifard, E. Coding that led to lockdown was 'totally unreliable' and a 'buggy mess', say experts. *The Telegraph* https://www.telegraph.co.uk/technology/2020/05/16/coding-led-lockdown-totally-unreliable-buggy-mess-say-experts/ (2020).
10. Campbell, D., Perraudin, F., Davis, N. & Weaver, M. Calls for inquiry as UK reports highest Covid-19 death toll in Europe. *The Guardian* https://www.theguardian.com/world/2020/may/05/uk-coronavirus-death-toll-rises-above-32000-to-highest-in-europe (2020).
11. Davey, M. Unreliable data: how doubt snowballed over Covid-19 drug research that swept the world. *The Guardian* https://www.theguardian.com/world/2020/jun/04/unreliable-data-doubt-snowballed-covid-19-drug-research-surgisphere-coronavirus-hydroxychloroquine (2020).
12. Mehra, M. R., Desai, S. S., Ruschitzka, F. & Patel, A. N. RETRACTED: Hydroxychloroquine or chloroquine with or without a macrolide for treatment of COVID-19: a multinational registry analysis. *The Lancet* https://doi.org/10.1016/S0140-6736(20)31180-6 (2020).
13. Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature* **533**, 452 (2016).
14. Reproducibility of Scientific Results (Stanford Encyclopedia of Philosophy). https://plato.stanford.edu/entries/scientific-reproducibility/ (2018).
15. Munafò, M. Open science and research reproducibility. *Ecancermedicalscience* **10**, ed56 (2016).
16. Chen, X. et al. Open is not enough. *Nat. Phys.* **15**, 113–119 (2019).
17. Lasser, J., Nield, J. M., Ernst, M., Karius, V. & Goehring, L. Salt polygons are caused by convection. *ArXiv190203600* (2019).
18. Jana Lasser. *JanaLasser/salt-polygons-are-caused-by-convection: final version for open science fellowship project 'executable papers'* (Zenodo). https://doi.org/10.5281/zenodo.3831237 (2020).
19. Executable Papers - improving the article format in computer science - News - Elsevier. https://www.journals.elsevier.com/the-journal-of-logic-and-algebraic-programming/news/introducing-executable-papers.

20. Mohammad Akhlaghi/reproducible-paper. *GitLab* https://gitlab.com/makhlaghi/reproducible-paper.
21. How to make a paper reproducible? | Reproducible Research. https://reproducibleresearch.net/how-to-make-a-paper-reproducible/.
22. Executable Papers - CodaLab Worksheets Documentation. https://codalab-worksheets.readthedocs.io/en/latest/Executable-Papers/.
23. Reproducible paper template - Summary. https://savannah.nongnu.org/projects/reproduce/.
24. Project Jupyter. https://www.jupyter.org.
25. A gallery of interesting Jupyter Notebooks. *GitHub* https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks.
26. Min, RK, LIGO binder. *GitHub*https://github.com/minrk/ligo-binder (2020).
27. Markdown. *Wikipedia* (2020).
28. MathJax | Beautiful math in all browsers. https://www.mathjax.org/.
29. The Python Language Reference — Python 3.8.3 documentation. https://docs.python.org/3/reference/.
30. Jupyter Kernels. *GitHub* https://github.com/jupyter/jupyter/wiki/Jupyter-kernels.
31. R: The R Project for Statistical Computing. https://www.r-project.org/.
32. Haskell Language. https://www.haskell.org/.
33. JavaScript. *MDN Web Docs* https://developer.mozilla.org/en-US/docs/Web/JavaScript.
34. Zenodo - Research. Shared. https://zenodo.org/.
35. The Binder Project. https://mybinder.org/.
36. Lasser, J. & Goehring, L. Subsurface salt concentration profiles and pore water density measurements from Owens Lake, central California, measured in 2018. *PANGAEA* https://doi.org/10.1594/PANGAEA.911059 (2020).
37. Lasser, J. & Goehring, L. Grain size distributions of sand samples from Owens Lake and Badwater Basin in central California, collected in 2016 and 2018. *PANGAEA* https://doi.org/10.1594/PANGAEA.910996 (2020).
38. Lasser, J. & Karius, V. Chemical characterization of salt samples from Owens Lake and Badwater Basin, central California, collected in 2016 and 2018. *PANGAEA* https://doi.org/10.1594/PANGAEA.911239 (2020).
39. Lasser, J., Goehring, L. & Nield, J. M. Images and Videos from Owens Lake and Badwater Basin in central California, taken in 2016 and 2018. *PANGAEA* https://doi.org/10.1594/PANGAEA.911054 (2020).
40. Nield, J. M., Lasser, J. & Goehring, L. Temperature and humidity time-series from Owens Lake, central California, measured during one week in November 2016. *PANGAEA* https://doi.org/10.1594/PANGAEA.911139 (2020).
41. Nield, J. M., Lasser, J. & Goehring, L. TLS surface scans from Owens Lake and Badwater Basin, central California, measured in 2016 and 2018. *PANGAEA* https://doi.org/10.1594/PANGAEA.911233 (2020).
42. Lasser, J., Nield, J. M. & Goehring, L. Surface and subsurface characterisation of salt pans expressing polygonal patterns. *Earth Syst. Sci. Data* https://doi.org/10.5194/essd-2020-86 (2020).

Acknowledgements

The work presented in this article was made possible by the Wikimedia foundation, Stifterverband, and Volkswagen foundation.

Author contributions

J.L. is the sole author of this article, conceived the idea, created the executable paper and wrote the manuscript.

Competing interests

The author declares no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.L.