Advancing Reproducible Research by Publishing R Markdown Notebooks as Interactive Sandboxes Using the learnr Package

by Chak Hau Michael Tso, Michael Hollaway, Rebecca Killick, Peter Henrys, Don Monteith, John Watkins, and Gordon Blair

Abstract Various R packages and best practices have played a pivotal role to promote the Findability, Accessibility, Interoperability, and Reuse (FAIR) principles of open science. For example, (1) well-documented R scripts and notebooks with rich narratives are deposited at a trusted data centre, (2) R Markdown interactive notebooks can be run on-demand as a web service, and (3) R Shiny web apps provide nice user interfaces to explore research outputs. However, notebooks require users to go through the entire analysis, while Shiny apps do not expose the underlying code and require extra work for UI design. We propose using the 'learnr' package to expose certain code chunks in R Markdown so that users can readily experiment with them in guided, editable, isolated, executable, and resettable code sandboxes. Our approach does not replace the existing use of notebooks and Shiny apps, but it adds another level of abstraction between them to promote reproducible science.

Introduction

There has been considerable recognition to the need to promote open and reproducible science in the past decade. The FAIR principles (Wilkinson et al., 2016; Stall et al., 2019) (https://www.go-fair.org/fair-principles/) of reproducble research are now known to most scientists. While significant advances has been made through the adoption of various best practices and policies (e.g. requirements from funders and publishers to archive data and source code, metadata standards), there remains considerable barriers to further advance open science and meet reproducible science needs. One of such issues the availability of various levels of abstraction of the same underlying analysis and code base to collaborate and engage with different stakeholders of diverse needs (Blair et al., 2019; Hollaway et al., 2020). For complex analysis or analysis that utilize a more advanced computing environment, it is essential to provide the capability to allow users to interact with the analysis at a higher level.

Existing approach to reproducible research focuses on either documenting an entire analysis or allows user-friendly interaction. Within the R ecosystem, R scripts and notebooks allow reserachers to work together and others to view the entire workflow, while R Shiny apps (Chang et al., 2019) allows rapid showcase of methods and research outcomes to users with less experience. R Shiny has been widely adopted to share research output and engage stakeholders since its conception in 2013. A recent review (Kasprzak et al., 2021) shows that bioinformatics is the subject with the most Shiny apps published in journals while earth and envrironmental science ranks second. Shiny apps are especially helpful to create reproducible analysis (e.g. examples in Hollaway et al., 2020) and explore different scenarios (e.g. Whateley et al., 2015; Mose et al., 2018). Finally, the interactivity of Shiny apps makes it a excellent tool for teaching(e.g. Williams and Williams, 2017; Field, 2020). However, not all users fit nicely into this dichotomy. Some users may only want to adopt a small fraction of an analysis for their work, while others may simply want to modify a few parts of the analysis in order to test alternative hypothesis. Current use of notebooks do not seem to support such diverse needs as notebook output *elements* (e.g. figures and tables) are not easily reproducible. This issue essentially applies to all coding languages.

One potential way to address the problem described above is to allow users to experiment with the code in protected computing environment. This is not limited to creating instances for users to re-run the entire code. Rather, this can also be done by exposing specific parts of a notebook as editable and executable code boxes, as seen in many interactive tutorial webpages for various coding languages. Recenty, while discussing next steps for fostering reproducible research in artificial intelligence, Carter et al. (2019) lists creating a protected computing environment ('data enclave' or 'sandbox') for reviewers to log in and explore as one of the solutions. In software engineering, a sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development including Web development and revision control. Making a sandbox environment available for users to test

and explore various changes to the code that leads to research outputs is a great step to further open science. Current practice of open science largely requires users to assemble the notebooks, scripts and data files provided in their own computing environment, which requires significant amount of time and effort. A sandbox environment can greatly reduce such barriers and if such sandboxes are available as a web service, users can explore and interact with the code that generates the research outputs at the convience of their own web browser on demand.

In this paper, we describe a rapid approach to create and publish "interactive sandboxes" R Shiny apps from R Markdown documents using the **learnr** package, with the aim to bridge the gap between typical R Markdown notebook and typical Shiny apps in terms of levels of abstraction. While code and markdown documents gives full details of the code, standard R Shiny apps has too much limitations on users to interact with the code and users often cannot see the underlying code. Our approach allows users to interact with selected parts of the code in an isolated manner, by specifying certain code chunks in a R Markdown document as executable code boxes.

The *learnr* R package

learnr (Schloerke et al., 2020) is an R package developed by RStudio to rapidly create interactive tutorials. It follows the general *R Markdown* (the file has .Rmd extensions, https://rmarkdown.rstudio.com/index.html) architecture and essentially creates a pre-rendered Shiny document similar to the way Shiny UI components can be added to any R Markdown documents. Pre-rendered Shiny documents (https://rmarkdown.rstudio.com/authoring_shiny_prerendered.HTML) is a key enabling technology for the **learnr** package since it allows users to specify the execution context in each code chunk of a R Markdown document that is used to render a R Shiny web app. Its use circumvents the need of a full document render for each end user browser session so that this type of R Shiny apps can load quickly. To create a **learnr** tutorial in RStudio after **learnr** is installed, the user chooses a **learnr** R Markdown template from a list after clicking the 'create new R Markdown document' button. This template is not different from other .Rmd files, except it requires additional chunk arguments to control the sandbox appearances. The two main features of the **learnr** package are the "exercise" and "quiz" options. The former allow users to directly type in code, execute it, and see its results to test their knowlege while the latter allow other question types such as multiple choice. Both of these options include auto-graders, hints, and instructor feedback options. Additonal overall options include setting time limits and an option to forbid users to skip sections. Like any Shiny apps, **learnr** apps can be easily embedded to other webpages, as seen in this example (Higgins, 2021).

Although the **learnr** package has existed for a few years now, it is relatively not well known to scientists as a potential use of R Shiny Apps and it has mostly been used for simple tutorial apps designed for R beginners. We propose a novel application of the **learnr** package to advance reproducible research, which we outline in the next section.

Approach: Using *learnr* for reproducible research 'sandboxes'

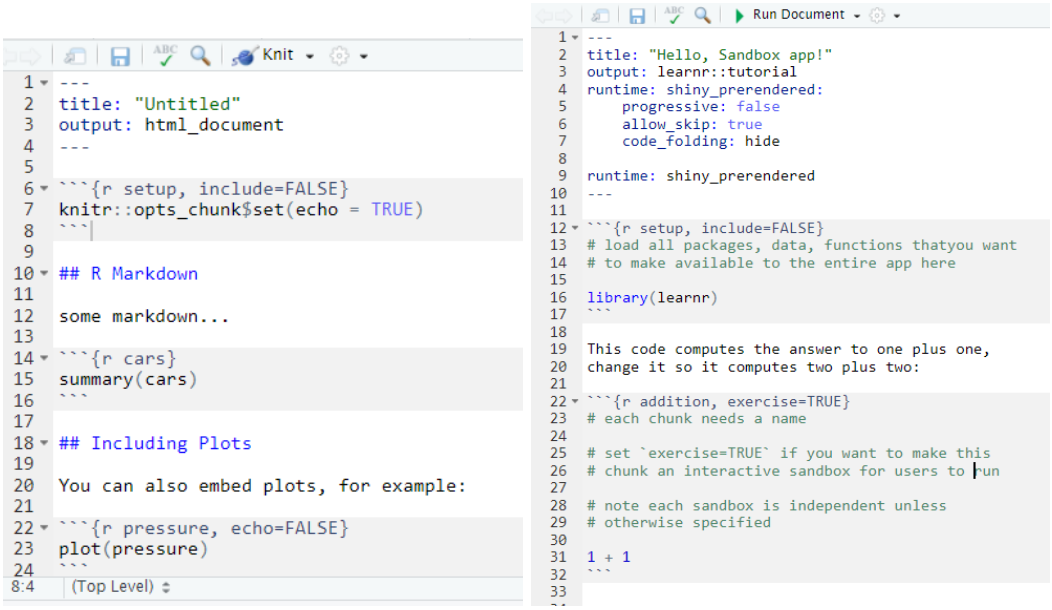
learnr allows users to create executable code boxes. Our approach is to publish R notebooks and serve parts of the notebooks as interactive sandboxes to allow users to re-create certain elements of a published notebook containing research outputs. We do not use the auto-graders and any quiz-like functionality of **learnr** while keeping the sandboxes. Notebook authors can go through their notebook and select the code chunks that they would allow users to experiment, while the others are rendered as static code snippets.

Recognizing **learnr** documents are themselves R Shiny web apps, our approach essentially allows the publication of notebooks in the form of web apps. However, unlike a typical R Shiny web app, users do not need to prepare a seprate UI (i.e. user interface) layout. Advanced users can modify the site appearance by supplying custom design in .css files.

Here, we first show the skeleton of a R Markdown (.Rmd) file for a **learnr** document (Figure 1). Notice that it is very similar to a typical .Rmd file where there is a mixture of narratives written in markdown and R code chunks, in addtional to a YAML header. However, there are a couple of important exceptions, namely the use of the `exercise` chunk option (i.e. editable and executable code boxes) and different output type in the YAML header.

Next, we outline the steps an author needs to take to publish R notebooks as interactive sandboxes:

1. All research output is included in the form of a well-documented R Markdown document



(a) A typical .Rmd file **(b)** A sandbox app .Rmd file

Figure 1: A comparison of minimal examples of a typical .Rmd document and a .Rmd document for an interactive sandbox app.

2. Open a new **learnr** R Markdown template. Copy the content of the original notebook
3. For the code chunks that you would like to become sandboxes, add `exercise=TRUE`. Make sure it has a unique chunk name. It may look something like this:

```
```{r fig2,warning=FALSE,exercise=TRUE,exercise.lines=30,fig.fullwidth=TRUE}
```

4. Before any interactive code chunks, call the first code chunk 'setup'. This will pre-load everything that will be used later.
5. Check whether you would like to link any of the interactive code snippets (by default each of them are independent, and only depends on the output of the 'setup' chunk) You may want to modify your code chunks accordingly.
6. Done! Knit the notebook to view outputs as an interactive webpage. Publish it just like a Shiny app.

The entire process took us a few hours of effort and can be incorporated to the proofreading of a R Markdown document. However, we note that as in any preparation of research output or web development several iterations are often needed and the time required increases accordingly as the compmlexity of the analysis increases.

In our implementation in DataLabs (https://datalab.datalabs.ceh.ac.uk/), the environment and folder to create the research is made available to the Shiny app in a read-only fashion. Therefore, the authors do not have to worry about versions of packages of data or a different software setup. Using DataLabs straightforward visual tools to publish R Shiny apps, we can publish an R Markdown notebook with interactive code snippets to reproduce certain parts of research readily in a few clicks.

Deployment

In general, **learnr** tutorial apps can be published the same way as R Shiny web apps in Shiny servers, such as the ones provided by cloud service providers or https://shinyapps.io. The **learnr** package vignettes provide additional help on deployment.

We also describe our deployment of these apps in DataLabs, a UK NERC virtual research environment that is being developed. DataLabs is a collobarative virtual research environment (Hollaway et al., 2020) (https://datalab.datalabs.ceh.ac.uk/) for environmental scientist to work together where data, software, and methods are all centrally located in projects. DataLabs provide a space for scientists from different domains (data science, statisticians, environmental science and computer science) to
```

work together and draw on each other's expertise. It includes an easy-to-use user interface where users can publish R Shiny apps with a few clicks, and this applies to these notebooks with interactive code chunks as well. Importantly, when provisioning a instance of R Shiny, this is deployed in a Docker container with read-only access to the project datastore being used for analysis. This allows an unprecended level of transparency as parts of the analysis are readily exposed for users to experiment from the exact environments, datasets (can be large and includes many files), and versions of software that created the analysis. The use of Docker deployed onto a Kubernetes infrastructure allows strict limits to be placed on what visitors can do through the use of resource constraints and tools such as **RAppArmor** (Ooms, 2013). While access to project files is read-only, some author discretion is still advised to ensure that visitors should not be able to view or list any private code or data. We also note that future relases of **learnr** will contain external exercise evaluators, so that the code sandboxes can be executed by an independent engine (such as Google Cloud) and give the benefit of not having to rely on **RAppArmor**.

Example: GB rainfall paper

To demonstrate our concept, we have turned a R Markdown notebook for a paper we have submitted for publication into a **learnr** site (https://cptecn-sandboxdemo.datalabs.ceh.ac.uk/) using the procedures described in the previous sections. The paper investigates the effect of weather and rainfall types on rainfall chemistry in the UK. As can be seen in Figure 2, the code chunks to generate certain parts of the paper is exposed. But unlike a static notebook site, the code chunk is not only available for copy and paste but allows users to modify and run on-demand. This makes it very straightforward for user to experiment with various changes of the original analysis, thereby promoting transparancy and trust.

Since **learnr** apps are R Markdown documents, Shiny UI elements can be easily added. We repeat one of the examples by replacing the interactive code box by a simple selector, with minimal modification of the code itself. This approach to publish Shiny apps requires significantly less work than typical R Shiny web apps since no UI design is needed and researchers can rapidly turn an R Markdown document to a R Shiny web app. For some cases, the use of certain datasets may require a license, as in this example. A pop-up box is shown when the site is loaded and visitors are required to check the boxes to acknowledge the use of the approapriate data licenses (an alternative is to require users to register and load a token file) before they can proceed.

Evaluation

The main strength of our approach is that it fills nicely the gap of existing approaches in terms of levels of abstraction. While code and markdown documents gives full details of the code, standard R Shiny apps has too much limitations on users to interact with the code (Figure 3) and users often cannot see the underlying code. Recently, it has become popular to publish 'live' Jupyter notebooks on Binder and Google Colab. While this is a great contribution to open science, users are still required to go run and go through the entire notebook step-by-step and it can be easy to break it if users change something in between. Our approach allows users to interact with portions of the code in a guided and isolated manner, without the need to understand all the other parts of a notebook or the fear to break it (Table 1). We emphasize that R scripts/notebooks and R Shiny apps work well for their intended uses, but our approach adds an additional level of accessibility to users.

The openness and ease-to-access our approach provides can benefit many different stakeholders (Table 2). Researchers can more rapidly reproduce *parts* of the analysis of their choice without studying the entire notebook or installing software or downloading all the data. They can quickly test alternative hypothesis and stimulate scientific discussions. For funders, encouraging the use of this approach means less time is needed for future projects to pick up results from previous work. And since this is based on **learnr** which is originally designed as a tutorial tool, this approach will no doubt speed up the process to train other users to use similar methods. Overall, it promotes open science and make a better value of public funds.

An obvious limitation of our approach is that it does not work well for ideal conditions where other R file formats are designed for. For instance, R scripts and R notebooks are much better suited for more complex analysis for users to adopt to their own problems. Meanwhile, R Shiny web apps provides a much richer user experience and is most suited when the exposed code is generally not useful to stakeholders. Nevertheless, as discussed above, our approach is designed for users to reproduce *elements* of an analysis. The user should evaluate these options carefully, paying special attention to the needs of intended users.

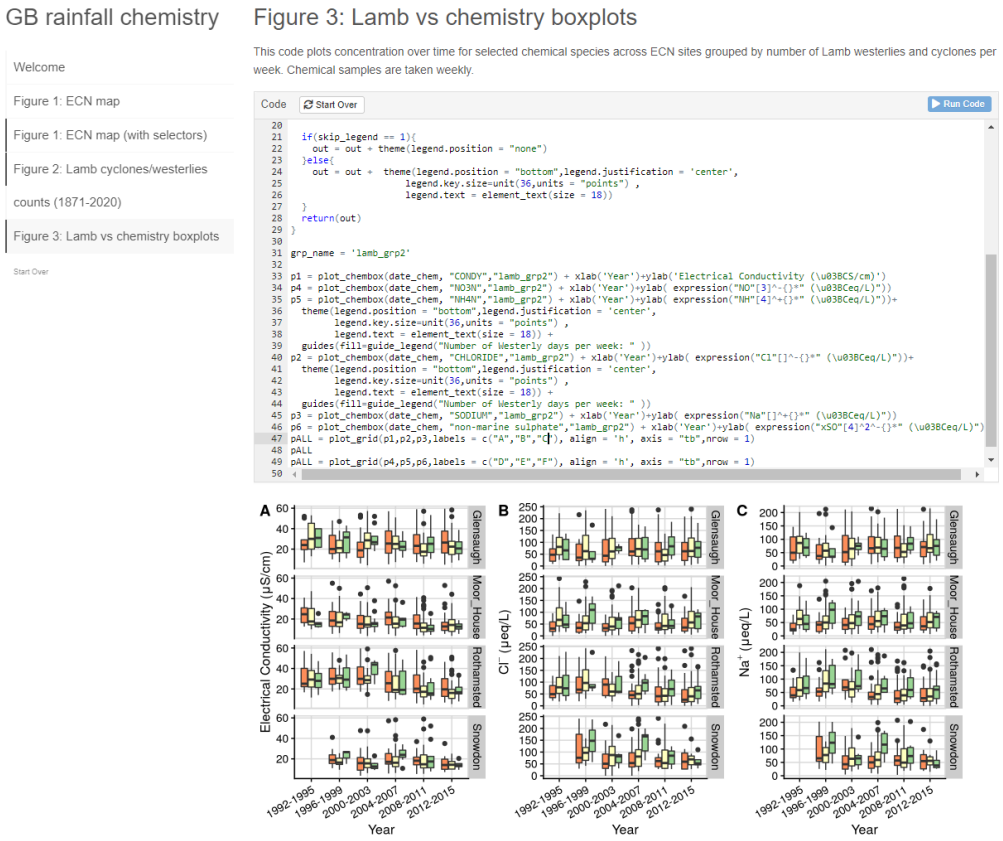


Figure 2: A screenshot of the GB rainfall interactive notebook site. The main feature is the code box. When the site loads, the code that generates published version of the figure is in the box and published version of the figure is below it. Users can make edits and re-run the code in the code box and the figure will update accordingly. Users can use the "Start Over" button to see the published version of the code at any point without refreshing the entire site.

Serving notebooks as a web service will inevitably face provenance issues. It is surely beneficial if the author's institution can host these interactive notebooks for a few years after its publication (and that of its related publications). In the future, publishers and data centres may consider providing services to provide longer term provenance of serving these interactive notebooks online. As for any web apps, funding for the computation servers can be a potential issue. This work uses DataLabs computation time which is part of the UK research funding that develops it. However, a more rigorous funding model may be needed in the future to ensure provenance of these notebooks.

Our approach focuses on improving reproducibility by exposing parts of R script for users to run them live on a R Shiny web app, leveraging the option to render R Markdown documents as R Shiny web apps and the **learnr** package. It focuses on the R scripts and R Markdown documents. Users, however, may want to improve reproducibility from the opposite direction, namely to allow outputs from an R Shiny web app to be reproducible outside of the Shiny context. For such a requirement, we recommend the use of the **shinymeta** (Cheng and Sievert, 2021) package, which allows users to capture the underlying code of selected output elements and allows users to download it as well as the underlying data to re-create the output in their own R instance. The **shinymeta** approach can be more involved and requires more effort than **learnr** so we think it is more suitable for users that are focusing their effort on the R Shiny app (particularly the UI). In summary, these two approaches complements each other and we recommend users to consider them to improve reproducibility of their work.

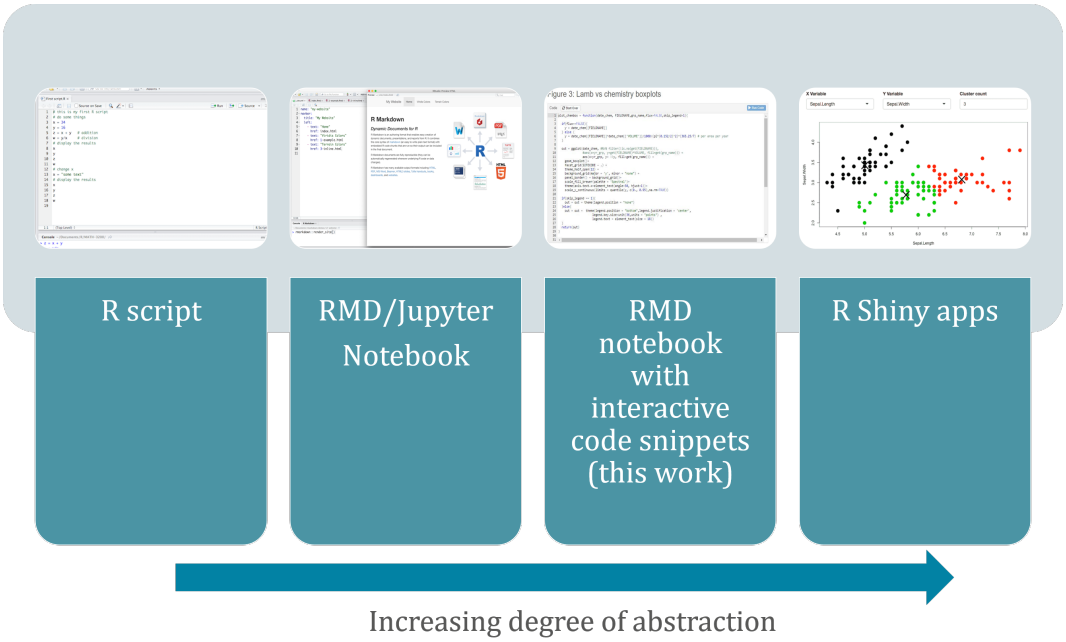


Figure 3: The various levels of abstraction of various types of R documents. Our approach fills nicely the gap between R Markdown or Jupyter notebooks and R Shiny apps.

Table 1: Advantages of the proposed approach to various stakeholders

Stakeholders	Advantages
Authors	• Very little extra work required in additional to writing R markdown document. • No experience to generate web interfaces required. • Much greater impact in research output.
Other researchers (those wanting to try or compare the method)	• A much more enriched experience to try methods and data and to test alternative hypothesis and scenarios. • No need to download data and scripts/notebooks and install packages to try a method. • More efficient to learn the new method.
Other researchers (those curious about the results)	• Try running different scenarios quickly than the published ones without the hassle of full knowledge of the code, downloading the code and data, and setting up the software environment. • Quickly reset to the published version of code snippet. No need to worry about breaking the code.
Data Centres	• A new avenue to demonstrate impact to funders if end users try methods or datasets hosted by them in sandboxes.
Funders	• Better value of investment if even small parts of a research is readily reproducible. • Time saving to fund related work that builds on research documented this way.
Wider research community and general public)	• Promotes trust and confidence in research through transparency.

Summary and outlook

We have proposed and demonstrated a rapid approach to publish R Markdown notebooks as interactive sandboxes to allow users to experiment with changes with various elements of a research output. It provides an additional level of abstraction for users to interact with research outputs and the codes that generates down. Since it can be linked to the environment and data that generated the published output and has independent document object identifiers (DOI), it is a suitable candidate to preserve research workflow while exposing parts of it to allow rapid experimentation by users. Our work is a demonstration on how we may publish a notebook from virtual research environments such as DataLabs, with data, packages, and workflow pre-loaded in a coding envrionment, accompanied by rich narratives. While this paper outlines the approach using R, the same approach can benefit other coding languages such as Python. In fact, this can already be achieved as **learnr** can run Python chunks (as well as other execution engines **knitr** supports such as SAS and mySQL) as long as the users generate and host the document using R. This paper contributes to the vision

towards publishing interactive notebooks as standalone research outputs and the advancement of open science practices.

Data availability and acknowledgments

The GB rainfall example notebook is accessible via this URL (https://cptecn-sandboxdemo.datalabs.ceh.ac.uk/) and the R Markdown file is depositied in the NERC Environmental Information Data Centre (EIDC) (Tso, 2021). The DataLab code stack is available at https://github.com/NERC-CEH/datalab. We thank the DataLabs developers team (especially Iain Walmsley, UKCEH) for the assitance to deploy interactive R Markdown documents on DataLabs. This work is supported by NERC Grant NE/T006102/1, Methodologically Enhanced Virtual Labs for Early Warning of Significant or Catastrophic Change in Ecosystems: Changepoints for a Changing Planet, funded under the Constructing a Digital Environment Strategic Priority Fund. Additional support is provided by the UK Status, Change and Projections of the Environment (UK-SCAPE) programme started in 2018 and is funded by the Natural Environment Research Council (NERC) as National Capability (award number NE/R016429/1). The initial development work of DataLabs was supported by a NERC Capital bid as part of the Environmental Data Services (EDS).

Table 2: Advantages of the proposed approach to existing approaches

Technology	Potential Issues	How our approach can help?
R script		
	• Limited narrative. Needs to run all the scripts.	• Much richer narrative and interactive experience.
Static notebooks		
	• Needs to download the code, data and package to try it out.	• Can instantly try out the code in a controlled manner, using the published data/packages/software environment.
Web apps (e.g. R Shiny)		
	• While web apps helpful to some stakeholders, it can be too high-level to some. • Lots of extra to create web interface. • Does not expose the code to generate results.	• Users can interact with the code within the code snippet sandboxes themselves. • The published version of the code is shown to users. • Users can run the code snippets live.
Binder/Google Colab		
	• Users change the entire notebook. • Users need to run all cells about the section they are interested.	• A much more enriched and guided experience. • Users can choose to only run the sandboxes they are interested.

Bibliography

G. S. Blair, P. Henrys, A. Leeson, J. Watkins, E. Eastoe, S. Jarvis, and P. J. Young. Data Science of the Natural Environment: A Research Roadmap. *Frontiers in Environmental Science*, 7, aug

2019. ISSN 2296-665X. doi: 10.3389/fenvs.2019.00121. URL https://www.frontiersin.org/article/10.3389/fenvs.2019.00121/full. [p1]

R. E. Carter, Z. I. Attia, F. Lopez-Jimenez, and P. A. Friedman. Pragmatic considerations for fostering reproducible research in artificial intelligence. *npj Digital Medicine*, 2(1), May 2019. doi: 10.1038/s41746-019-0120-2. URL https://doi.org/10.1038/s41746-019-0120-2. [p1]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2019. URL https://CRAN.R-project.org/package=shiny. R package version 1.4.0. [p1]

J. Cheng and C. Sievert. *shinymeta: Export Domain Logic from Shiny using Meta-Programming*, 2021. URL https://CRAN.R-project.org/package=shinymeta. R package version 0.2.0.1. [p6]

A. Field. *adventr: Interactive R Tutorials to Accompany Field (2016), "An Adventure in Statistics"*, 2020. URL https://cran.r-project.org/web/packages/adventr/index.html. [p1]

P. Higgins. *Reproducible Medical Research with R*, 2021. URL https://bookdown.org/pdr_higgins/rmrwr/. [p2]

M. J. Hollaway, G. Dean, G. S. Blair, M. Brown, P. A. Henrys, and J. Watkins. Tackling the challenges of 21st-century open science and beyond: A data science lab approach. *Patterns*, 1(7):100103, Oct. 2020. doi: 10.1016/j.patter.2020.100103. URL https://doi.org/10.1016/j.patter.2020.100103. [p1, 3]

P. Kasprzak, L. Mitchell, O. Kravchuk, and A. Timmins. Six years of shiny in research - collaborative development of web tools in r. *R Journal*, (3):155–162, 2021. URL https://journal.r-project.org/archive/2021/RJ-2021-009/RJ-2021-009.pdf. [p1]

V. N. Mose, D. Western, and P. Tyrrell. Application of open source tools for biodiversity conservation and natural resource management in east africa. *Ecological Informatics*, 47:35–44, Sept. 2018. doi: 10.1016/j.ecoinf.2017.09.006. URL https://doi.org/10.1016/j.ecoinf.2017.09.006. [p1]

J. Ooms. The RAppArmor package: Enforcing security policies in R using dynamic sandboxing on linux. *Journal of Statistical Software*, 55(7):1–34, 2013. URL http://www.jstatsoft.org/v55/i07/. [p4]

B. Schloerke, J. Allaire, and B. Borges. *learnr: Interactive Tutorials for R*, 2020. URL https://CRAN.R-project.org/package=learnr. R package version 0.10.1. [p2]

S. Stall, L. Yarmey, J. Cutcher-Gershenfeld, B. Hanson, K. Lehnert, B. Nosek, M. Parsons, E. Robinson, and L. Wyborn. Make scientific data FAIR, 2019. ISSN 14764687. [p1]

C.-H. M. Tso. R shiny sandbox app demonstrator for advancing reproducible research. NERC Environmental Information Data Centre. (Model). https://doi.org/10.5285/df57b002-2a42-4a7d-854f-870dd867618c, 2021. URL https://doi.org/10.5285/df57b002-2a42-4a7d-854f-870dd867618c. [p8]

S. Whateley, J. D. Walker, and C. Brown. A web-based screening model for climate risk to water supply systems in the northeastern united states. *Environmental Modelling & Software*, 73:64–75, Nov. 2015. doi: 10.1016/j.envsoft.2015.08.001. URL https://doi.org/10.1016/j.envsoft.2015.08.001. [p1]

M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 2016. ISSN 20524463. doi: 10.1038/sdata.2016.18. [p1]

I. J. Williams and K. K. Williams. Using an R shiny to enhance the learning experience of confidence intervals. *Teaching Statistics*, 40(1):24–28, Nov. 2017. doi: 10.1111/test.12145. URL https://doi.org/10.1111/test.12145. [p1]

Chak Hau Michael Tso
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0002-2415-0826
mtso@ceh.ac.uk

Michael Hollaway
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0003-0386-2696
mhollaway@ceh.ac.uk

Rebecca Killick
Department of Statistics, Lancaster University
Flyde College
Lancaster LA1 4YF, United Kingdom
ORCiD: 0000-0003-0583-3960
r.killick@lancaster.ac.uk

Peter Henrys
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0003-4758-1482
pehn@ceh.ac.uk

Don Monteith
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
donm@ceh.ac.uk

John Watkins
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0002-3518-8918
jww@ceh.ac.uk

Gordon Blair
School of Computing and Communications, Lancaster University
InfoLab21
Lancaster LA1 4WA, United Kingdom
ORCiD: 0000-0001-6212-1906
g.blair@lancaster.ac.uk