Keeping modelling notebooks with TRACE: Good for you and good for environmental research and management support

Daniel Ayllón, Steven Railsback, Cara Gallagher, Jacqueline Augusiak, Hans Baveco, Uta Berger, Sandrine Charles, Romina Martin, Andreas Focks, Nika Galic, et al.

Keeping modelling notebooks with TRACE: Good for you and good for environmental research and management support

Daniel Ayllón [a,*], Steven F. Railsback [b], Cara Gallagher [c], Jacqueline Augusiak [d], Hans Baveco [e], Uta Berger [f], Sandrine Charles [g], Romina Martin [h], Andreas Focks [e], Nika Galic [i], Chun Liu [j], E. Emiel van Loon [k], Jacob Nabe-Nielsen [c], Cyril Piou [l], J. Gareth Polhill [m], Thomas G. Preuss [n], Viktoriia Radchuk [o], Amelie Schmolke [p], Julita Stadnicka-Michalak [q], Pernille Thorbek [r], Volker Grimm [s,t]

[a] Complutense University of Madrid (UCM), Faculty of Biology, Department of Biodiversity, Ecology and Evolution. Calle José Antonio Novais 12, 28040, Madrid, Spain daniel.ayllon@bio.ucm.es

[b] Lang Railsback & Associates, 250 California Ave., Arcata, CA 95521, USA Steve@LangRailsback.com

[c] Department of Bioscience, Aarhus University, Frederiksborgvej 399, 4000 Roskilde, Denmark cgallagher@bios.au.dk, jnn@bios.au.dk

[d] Charles River Laboratories Den Bosch B.V., Dept. of Discovery and Environmental Sciences, Hambakenwetering 7, 5231 DD 's-Hertogenbosch, the Netherlands jaugusiak@gmail.com

[e] Wageningen Environmental Research, Wageningen University & Research, Droevendaalsesteeg 3a, 6708PB Wageningen, the Netherlands hans.baveco@wur.nl, andreas.focks@wur.nl

[f] TU Dresden, Institute of Forest Growth and Computer Sciences, Pienner Straße 8, 01737 Tharandt, Germany uta.berger@tu-dresden.de

[g] Université de Lyon, Université Lyon 1, CNRS, UMR 5558, Laboratoire de Biométrie - Biologie Evolutive, 43 boulevard du 11 novembre 1918, F-69622 Villeurbanne Cedex, France sandrine.charles@univ-lyon1.fr

[h] Stockholm Resilience Centre, Stockholm University, 10691 Stockholm, Sweden romina.martin@su.se

[i] Syngenta Crop Protection LLC., Greensboro, NC-27408, USA nika.galic@syngenta.com

[j] Syngenta, Herbicide Bioscience, Jealott's Hill International Research Centre, Bracknell, RG42 6EY, UK chun.liu@syngenta.com

[k] Institute for Biodiversity and Ecosystem Dynamics, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, the Netherlands E.E.vanLoon@uva.nl

[l] CIRAD, UMR CBGP, INRA, IRD, Montpellier SupAgro, Univ. Montpellier, F-34398 Montpellier, France cyril.piou@cirad.fr

[m] The James Hutton Institute, Craigiebuckler, Aberdeen, AB15 8QH, UK gary.polhill@hutton.ac.uk

[n] Bayer AG, Alfred Nobel Str. 50, 40789 Monheim, Germany thomas.preuss1@bayer.com

[o] Leibniz Institute for Zoo and Wildlife Research (IZW), Alfred-Kowalke-Straße 17, 10315 Berlin, Germany radchuk@izw-berlin.de

[p] Waterborne Environmental, Inc., 897B Harrison Street SE, Leesburg, VA 20175, USA schmolkea@waterborne-env.com

[q] Eawag, Swiss Federal Institute of Aquatic Science and Technology, 8600 Dübendorf, Switzerland julita.stadnicka@eawag.ch

[r] BASF SE, APD/EE, Speyerer Strasse 2, 67117 Limburgerhof, Germany pernille.thorbek@basf.com

This document is the accepted manuscript version of the following article:
Ayllón, D., Railsback, S. F., Gallagher, C., Augusiak, J., Baveco, H., Berger, U., … Grimm, V. (2021). Keeping modelling notebooks with TRACE: good for you and good for environmental research and management support. Environmental Modelling and Software, 136, 104932 (12 pp.). https://doi.org/10.1016/j.envsoft.2020.104932

[s] Helmholtz Centre for Environmental Research – UFZ, Permoserstr. 15, 04318 Leipzig, Germany volker.grimm@ufz.de

[t] University of Potsdam, Institute for Biochemistry and Biology, Maulbeerallee 2, 14469 Potsdam, Germany

* **Corresponding author:** Daniel Ayllón. Complutense University of Madrid (UCM), Faculty of Biology, Department of Biodiversity, Ecology and Evolution. Calle José Antonio Novais 12, 28040, Madrid, Spain. Telephone: +34 913944951. E-mail: daniel.ayllon@bio.ucm.es

Content

Abstract

The acceptance and usefulness of simulation models are often limited by the efficiency, transparency, reproducibility, and reliability of the modelling process. We address these issues by suggesting that modellers (1) "trace" the iterative modelling process by keeping a modelling notebook corresponding to the laboratory notebooks used by empirical researchers, (2) use a standardized notebook structure and terminology based on the existing TRACE documentation framework, and (3) use their notebooks to compile TRACE documents that

supplement publications and reports. These practices have benefits for model developers, users, and stakeholders: improved and efficient model design, analysis, testing, and application; increased model acceptance and reuse; and replicability and reproducibility of the model and the simulation experiments. Using TRACE terminology and structure in modelling notebooks facilitates production of TRACE documents. We explain the rationale of TRACE, provide example TRACE documents, and suggest strategies for keeping “TRACE Modelling Notebooks.”

Keywords: model documentation, standards, modelling cycle, reproducible research, environmental modelling, scientific communication

"The act of writing in the notebook causes the scientist to stop and think about what is being done in the laboratory. It is in this way an essential part of 'doing good science'."

Kanare (1985, p. 1)

1. Introduction

Modelling has become an essential tool for environmental and ecological research and for management support (e.g., EFSA, 2014, 2018; Stillman et al., 2015; Elsawah et al., 2017; Ayllón et al., 2018; Badham et al., 2019; Schuhwirth et al., 2019). Simulation experiments are used to explore and understand the behaviour of a model and to make inferences about the corresponding real system. However, a major limitation in the current practice of environmental modelling is that simulation experiments are usually not documented well enough, particularly for the purpose of relating and discussing insights to management practice (Schmolke et al., 2010, Schuwirth et al., 2019). Most of the work testing, evaluating, analyzing, and actually using a model usually remains undocumented, which limits the transparency and hence credibility of the results while making inefficiencies more likely.

In empirical research, the use of laboratory notebooks to document experiments is routine and often standardized and demanded by third parties (Kanare, 1985; Lee, 2003; Nickla and Boehm, 2011), but there is no such notebook culture for simulation experiments in ecology and other environmental sciences. This lack is unfortunate because keeping modelling notebooks is as valuable for simulation experiments as it is for empirical experiments. Keeping a notebook on a daily basis forces us not only to document settings and results, but also to write narratives about what we have learned. Writing by itself improves our science, as pointed out by the sociologist Niklas Luhmann: "*Without writing, one cannot think; at least not in a sophisticated, connective way*" (translated from German).

Many simulation modellers keep some kind of paper or electronic notebook, but usually at their own discretion, as they are not specifically trained to maintain these types of records. Consequently, their notebooks are not standardized, often incomplete, and not easily understood by others or even, after some time, by the notebook keepers themselves. This situation is regrettable as keeping and using a proper notebook provides many benefits for both the individual modeller and for environmental research and management based on modelling.

First, keeping a record of the whole modelling process contributes to increased efficiency during model development and when re-using a model in new applications. Without keeping track of the workflow, the modelling process often becomes less efficient because resources influencing coding decisions (e.g., web pages, papers, code snippets) are

forgotten, analyses must be repeated, and mistakes, unproductive approaches, or unsuccessful trials are made or used repeatedly (Grimm et al., 2014).

Second, modelling notebooks streamline the generation of transparent supporting documentation that facilitates the acceptance and increases impact of developed models. The design of models and simulation experiments often looks ad hoc because the series of experiments and thoughts leading to the current design have not been documented (Schmolke et al., 2010). Without a modelling notebook, early model design decisions, assessments of model quality and realism, and rationales for simulation experiment design are hard to document as the project wraps up. Chances of a model being used for decision support are considerably increased if the rationale for important assumptions is transparent and there is documentation of how alternative assumptions were evaluated.

Third, precise descriptions of model design, analysis, and application enable reproducible research. Replication or reproduction of published results can be difficult or even impossible because simulation experiment details have been forgotten or lost. Many studies have recently drawn attention to problems with model replicability and reproducibility in computation-based science (Peng, 2011; Sandve et al., 2013; Donkin et al., 2017; Rougier et al., 2017; Miłkowski et al., 2018; Monks et al., 2018). Inaccurate or imprecise description of the model, analysis workflow, or simulation experiments are among the main reasons published simulation experiments cannot be replicated or reproduced (Crook et al., 2013; Rougier et al., 2017). Modelling notebooks are an important tool for avoiding such problems.

Despite these benefits, simulation modellers have not yet developed a strong culture of keeping notebooks. Moreover, even if such a culture existed, some of its benefits—increased transparency, reproducibility, and credibility—would still be limited because they require communication with others and notebooks contain too much unfiltered information to be communicated efficiently. To be useful to a model's "clients", information in the modelling notebook must be filtered and distilled into a document of appropriate format and detail.

But what is appropriate document format and detail for a useful summary of a modelling notebook? In fact, there is already a standard for such documents: TRACE (TRAnsparent and Comprehensive model Evaludation[1]; Schmolke et al., 2010; Grimm et al., 2014). TRACE provides a standardized terminology and structure for compiling, in a printable document intended for others to use, information about the formulation, implementation, testing, analysis, evaluation/validation, and application of ecological and

[1] 'Evaludation' merges model evaluation and validation and refers to the process of assessing the quality of all aspects of a model and its development – see section 3.

other simulation models. The TRACE standard keeps modellers from having to invent their own format and provides guidance on what information is important to include. Standard formats make communication more efficient and coherent, as we know from the standard structure of scientific publications and standards for describing agent-based (the ODD protocol; Grimm et al., 2006, 2010, 2020a; Railsback and Grimm, 2019) or species distribution models (e.g., Zurell et al., 2020).

Because TRACE already provides a standard format for distilling information from notebooks, all we need to fully realize the benefits of both is a way to efficiently link notebooks to TRACE. To provide this link and therefore (1) facilitate the establishment of a culture of keeping modelling notebooks and (2) promote a standard for summarizing and communicating their contents, we propose a standard format for modelling notebooks that uses TRACE terminology and standards.

There are important differences between TRACE documents and modelling notebooks. TRACE documents are designed to be published as supplements to scientific articles or reports (Schmolke et al., 2010; Grimm et al., 2014), with the goal of making decisions and research more transparent, robust, and trustworthy (Grimm et al., 2020b). In contrast, modelling notebooks are primarily for the modellers themselves, kept as a routine record of the work conducted during the entire modelling cycle. But providing the information for a TRACE document is yet another benefit of keeping a modelling notebook, and this benefit should be a major consideration in designing the notebook.

We first briefly review and summarize literature on laboratory notebooks. This provides insight into why and how modelling notebooks should be kept, but we also discuss important differences between empirical and simulation experiments and, hence, laboratory and modelling notebooks. Then we explain TRACE and its rationale, and introduce the corresponding proposed structure of modelling notebooks.

Next, we give general and practical recommendations for keeping modelling notebooks. Our article particularly targets beginners and junior modellers, faculty teaching modelling classes, and modellers not in the habit of keeping notebooks. We therefore provide (1) guidelines about the kind of notes and details to be included in the modelling notebook in relation to each TRACE element, (2) general recommendations on keeping the notebook and designing the workflow, and (3) information on a range of available notebook tools to help modellers select one appropriate to their experience and skills. Consequently, the article goes into some detail about the TRACE protocol. Readers interested in practical instructions on how to use it are directed to section 4. The experience of those who follow our

recommendations is that keeping a modelling notebook typically takes about 15 minutes per day.

We conclude with a vision of the future practice and role of ecological and environmental modelling if a culture of keeping modelling notebooks and producing TRACE documents is established.

2. Documenting experiments: Laboratory and modelling notebooks

A laboratory notebook is usually a bound book where the experimenter takes notes on the planning, design, execution, analysis, and interpretation of experiments (Kanare, 1985). Alternative names are “laboratory journal” or “research journal”. The classical laboratory notebook contains the purpose of the experiment, all information that is needed to repeat the experiment, and the data observed (or just the metadata) or the reference to where these data are stored, and a short interpretation and comment on the results. Notes should be concise and clear enough that they can be read and fully understood by others. Notebook entries form the basis of scientific publications, but they also describe all kinds of experiments that are not published but nevertheless important to document, including auxiliary or preparatory experiments and experiments that failed.

One important function of a modelling notebook is similar to that of a laboratory notebook: the documentation of experimental procedures that allows replication and explanation of how experiments were designed, executed, and interpreted. However, laboratory and modelling notebooks differ in various ways. One important difference is the format. The paper-based laboratory notebook is still the most widely used form of recording in laboratories worldwide, despite no longer being the most efficient system because most data generated in laboratories is now digital (Dirnagl and Przesdzing, 2016; Kanza et al., 2017). For modelling notebooks, a paper-based format is inefficient. An electronic format offers many advantages over the traditional paper notebook, including: (1) legibility of handwriting is not an issue, (2) sharing with others is easy, (3) the notebook can be kept in the same (virtual) place as all other project files, (4) synchronization through the cloud allows notebooks to be used on multiple devices, (5) documents are searchable, (6) external files can be directly linked to the notebook, and (7) graphs, pictures, tables, code, and other documents (e.g., spreadsheets, PDFs, presentations) can be inserted, cross-referenced, and annotated.

Another difference between laboratory and modelling notebooks results from simulation experiments being “ephemeral” compared to empirical experiments: simulations are usually fast and easily repeated, in principle. Therefore, simulation experiments can investigate substantially more factors with substantially more treatments than field or laboratory experiments (Kleijnen, 2015). Modellers often run thousands of simulations, so documenting each experimental treatment separately would be impossible and not meaningful. The task then is not to document each experimental treatment, but the overall “design of experiment” (Lorscheid et al., 2012). While empirical research sometimes has the same challenge (e.g., in automated laboratories), modellers especially need techniques for documenting large experiments.

While the “design of experiment” approach is straightforward in systematic, scientific-based analyses such as sensitivity analysis, heuristic analyses are more challenging to document (Railsback and Grimm, 2019, Chapter 22). Heuristic analyses are important steps in initial testing of model behaviour as well as in robustness analysis (Sect. 4.2.7, below). Heuristic analyses, often informally referred to as “playing with” the model, frequently result in important design decisions (just as preliminary experiments do in the empirical laboratory). However, if such analyses are not documented the design will look ad hoc and may not convince model users. Recording these heuristic analyses in a notebook is especially important because details and results of such simulation experiments are otherwise very likely to be lost; the more flexible and less well-defined a model analysis, the more important it is to record its design, results, and consequences.

Besides experimental treatments and results, notebooks should document the details of the materials and methods used. In modelling, this means documenting the development of a model, including its design and underlying rationale. Model development is usually an iterative process that includes starting with simple model versions (“prototypes”), learning from them, and then systematically improving the model’s design until it is considered realistic enough for its intended purpose. This “modelling cycle” (Grimm and Railsback, 2005) produces important details throughout, as we discuss in Sect. 4.1; documenting these details in a notebook makes iteration through the cycle more efficient and less subject to errors.

3. Documenting the modelling cycle: Model "evaludation" and the TRACE documentation framework

Modelling notebooks are intended primarily for the modeller's own use: they are for recording any information that may be useful later, no matter how extensive or detailed. The problem we address is that *some* but not *all* of this information is essential for documenting the model's usefulness and reliability for future users and decision-makers: how do we determine what information from a modelling notebook needs to be turned into public documentation, and what format should that documentation have?

These questions have been answered in the "evaludation" framework for assessing good modelling practice (Augusiak et al., 2014) and, in turn, the TRACE model documentation framework (Grimm et al., 2014), which are both explicitly based on the modelling cycle (Figure 1). TRACE was first developed as a standard format for documenting all elements of iterative model development ("TRAnsparent and Comprehensive Ecological modelling"; Schmolke et al., 2010). Its purpose was to create transparency and quality assurance and thereby help decision makers and other stakeholders understand the conditions under which a simulation model can be used to support their decisions.

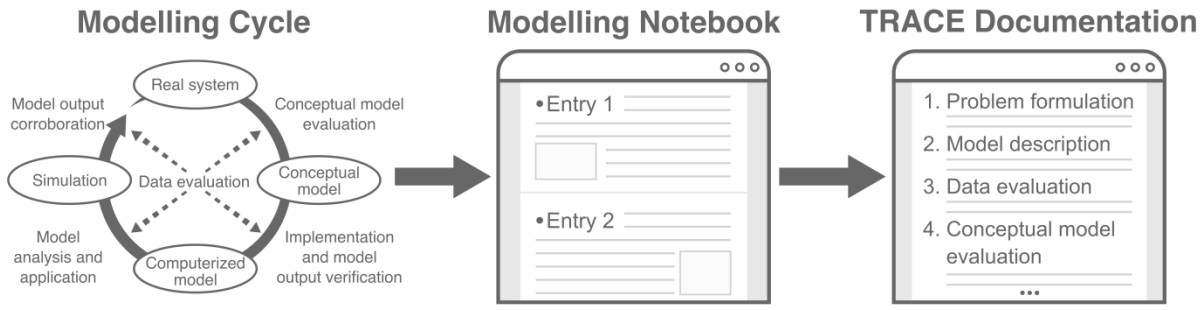

Fig. 1. Daily modelling activities are related to the iterative modelling cycle and documented in the modelling notebook. TRACE documentation can be compiled from the daily notebook entries at any stage of the project, in particular when a publication or report is generated.

In addition to its original purpose as a documentation framework, TRACE was found to also facilitate good modelling practice. Augusiak et al. (2014) suggested a new structure and terminology for the modelling cycle, centred around "evaludation", defined as "the entire process of establishing model quality and credibility throughout all stages of model development, analysis, and application" (Augusiak et al., 2014, p.121). Grimm et al. (2014)

adopted this new terminology for the current version of TRACE, which now stands for "TRAnsparent and Comprehensive model Evaludation" and is defined as: "a tool for planning, performing, and documenting good modelling practice. TRACE documents should provide convincing evidence that a model was thoughtfully designed, correctly implemented, thoroughly tested, well understood, and appropriately used for its intended purpose." (Grimm et al., 2014, p.129). This makes the TRACE framework useful as a target for notebook keeping: knowing that they will create a TRACE document from their modelling notebooks tells modellers what needs to be recorded and guides them to produce useful, reproducible simulation experiments.

TRACE documents are meant to provide comprehensive documentation of models that can be submitted as supplementary material with scientific publications, reports, or dossiers where models are presented to support decision making. TRACE provides a standard format for organizing and documenting the elements of model evaludation so that (1) modellers know where to present what kind of information, and (2) model users and evaluators know exactly where to look for this information, guided by tables of contents and executive summaries. At the same time, TRACE provides a checklist for modellers, which helps them to make sure that they thoroughly addressed and documented issues that affect the quality and usefulness of a model. A full TRACE document consists of eight elements: two elements related to model development (*problem formulation* and *model description*) and six related to model evaludation, which largely correspond to the elements of the modelling cycle (Grimm and Railsback, 2005): *data evaluation*, *conceptual model evaluation*, *implementation verification*, *model output verification*, *model analysis* and *model output corroboration* (Table 1). The *model analysis* element includes the model's application to its intended purpose, such as answering a research question, evaluating alternative management scenarios, or assessing environmental risk.

TRACE element / MN entry tag	**MN keyword**	**Provides supporting information on**
1. Problem formulation	Model purpose; Research questions	"The decision-making context in which the model will be used; the types of model clients or stakeholders addressed; a precise specification of the question(s) that should be answered with the model, including a specification of necessary model outputs; and a statement of the domain of applicability of the model, including the extent of acceptable extrapolations."

2. Model description	Model development; Design decisions	"The model. Provide a detailed written model description. For individual/agent-based and other simulation models, the ODD protocol is recommended as standard format. For complex submodels it should include concise explanations of the underlying rationale. Model users should learn what the model is, how it works, and what guided its design."
3. Data evaluation	Parameterization; Patterns	"The quality and sources of numerical and qualitative data used to parameterize the model, both directly and inversely via calibration, and of the observed patterns that were used to design the overall model structure. This critical evaluation will allow model users to assess the scope and the uncertainty of the data and knowledge on which the model is based."
4. Conceptual model evaluation	Conceptual design decisions	"The simplifying assumptions underlying a model's design, both with regard to empirical knowledge and general, basic principles. This critical evaluation allows model users to understand that model design was not ad hoc but based on carefully scrutinized considerations."
5. Implementation verification	Debugging	(1) "Whether the computer code implementing the model has been thoroughly tested for programming errors."
	Software verification/ Testing	(2) "Whether the implemented model performs as indicated by the model description."
	Usability tools design	(3) "How the software has been designed and documented to provide necessary usability tools (interfaces, automation of experiments, etc.) and to facilitate future installation, modification, and maintenance."
6. Model output verification	Output verification/ Goodness-of-fit	(1) "How well model output matches observations."
	Calibration; Tests on environmental drivers	(2) "How much calibration and effects of environmental drivers were involved in obtaining good fits of model output and data."
7. Model analysis and application	Sensitivity analysis; Uncertainty analysis	(1) "How sensitive model output is to changes in model parameters."
	Robustness analysis; Simulation experiment	(2) "How well the emergence of model output has been understood."
8. Model output corroboration	Output corroboration / Validation	"How model predictions compare to independent data and patterns that were not used, and preferably not even known, while the model was developed, parameterized, and verified. By documenting model output corroboration, model users learn about evidence which, in addition to model output verification, indicates that the model is structurally realistic so that its predictions can be trusted to some degree."

Table 1. Structure, terminology, and contents of TRACE documents, and their link to entries in the modelling notebook (MN). The third column, describing the information provided by each TRACE element, includes literal definitions provided by Grimm et al. (2014).

With TRACE as a standard for the modelling cycle documentation that will be extracted from a notebook, it makes sense to design the notebook to facilitate distillation of its contents into TRACE. In fact, the TRACE framework provides a coherent standard format and terminology for modelling notebooks, and organizing a modelling notebook using the TRACE terminology, in turn, makes the compilation of TRACE documents easier and more efficient. Extracting the information and data needed to compile a TRACE document is relatively straightforward if the standardized TRACE terminology is used to tag entries in the notebook (Schmolke et al., 2010; Augusiak et al., 2014; Grimm et al., 2014; Fig. 2).

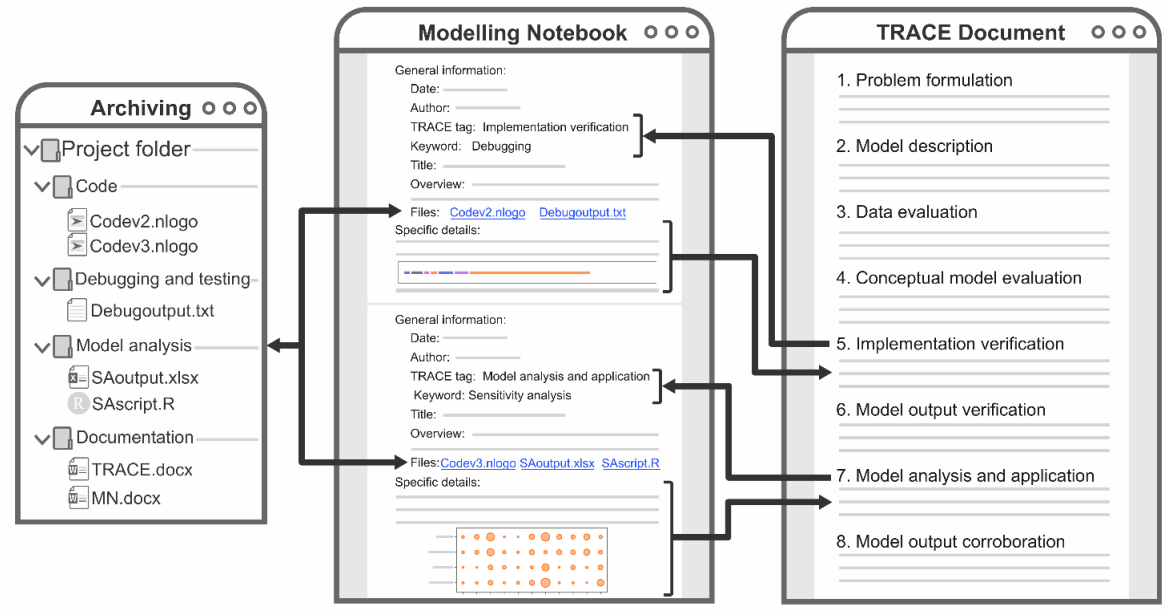

Fig. 2. Schematic figure explaining the relation between files in a modelling project, a modelling notebook, and TRACE. Notebook entries are added chronologically with tags provided by the terminology from the TRACE framework. Standardized tags and keywords, in turn, facilitate the compilation of a TRACE document from the specific details included in the notebook entries. Entries provide hyperlinks to all files related to a modelling task, indicating their location in the archiving system; TRACE terminology should also be used to organize and name files and folders.

4. Proposed structure of TRACE modelling notebooks

We suggest a TRACE-based approach to keeping modelling notebooks. We first list the types of information that should be entered in a notebook and then present detailed examples of the information to be provided for each modelling task in Tables 2 and 3.

4.1. What should be in a modelling notebook?

There are several ways of organizing a modelling notebook. Modellers can employ a fully chronological format analogous to that of the laboratory notebook, with tags using TRACE terminology and with entries chronological irrespective of their TRACE category (Box 1). Alternatively, the notebook can have the same structure as a TRACE document with entries made chronologically under the relevant TRACE element. For any organization, it is fundamental to make entries chronological to make the notebook a "master log" of the daily work. (Work on multiple tasks on the same day should of course be logged with separate entries.)

Fundamental elements of any modelling notebook are a log of daily, dated entries reporting what was done on the project and why, and what was accomplished; and the name, location, and a brief description of all files relevant to the project (Box 1). An additional important element is a "to-do" list of both critical issues to be addressed as soon as the task is resumed and non-critical issues to be addressed when one has time.

1. **Table of contents**. A table with hyperlinks to each entry at the beginning of the notebook. The TOC can be a chronological index with entries listed by date. In addition to the TOC, it is advisable to include a **topical index** organized by the tags and keywords in Table 1 to hyperlink each TRACE element and modelling task to related entries.
2. **Master catalogue.** A list of the locations of files most relevant to the project, with a description of the file and folder taxonomy.
3. **Work log**. The main body of the notebook, composed of daily, dated entries. Each entry includes:
(a) **General information** (common to all entries): (i) *date* of the entry, (ii) *author* of the entry, (iii) *TRACE tag* indicating the TRACE element the entry is linked to (Table 1), (iv) *keyword* indicating the specific modelling task within the TRACE element (Table 1), (v) *title*, (vi) *overview* of what has been done and what has been accomplished, and (vii) *files* linked to the entry (e.g., program code, script used to generate the experiment, spreadsheet containing parameter values, model input files, output files from experiments, summary files).
(b) **Specific details**, which depend on the specific modelling task (Tables 2 and 3).

Box 1. Proposed structure for modelling notebooks.

Different approaches to producing public documentation are possible; you can either: (1) document everything in the notebook as you work, and then later prepare final documents like ODD, TRACE, project reports, or a software user guide from the notebook; or (2) write those documents as part of the workflow, entering information directly in them instead of in the notebook. In the second case, the notebook should document both (a) where you were working, i.e., what task you worked on, what document/code and section you worked in, the version of the document/code that first contained this work and where it was archived, and what was left to be done or fixed; and (b) any material you did not save anywhere else, and consider unlikely to be included in a document but important for staying organized and efficient. It is worth highlighting once again that modelling is an iterative process, so modellers typically switch from task to task and go back to earlier tasks when they need to modify something. The modelling notebook is important because it allows and even facilitates such iterative work. Entries should be written so they contain all the information needed to resume work efficiently.

Tagging the entries in the modelling notebook using TRACE terminology provides the link between the notebook and TRACE documents (Fig. 2). Lower-level tags (keywords; Table 1) refine the information about the specific task conducted within TRACE's broad categories (e.g., "sensitivity analysis" within the "model analysis" element); informative, concise titles can further subdivide and organize entries. This TRACE-based workflow organization can make a project more efficient even if a TRACE document is never prepared.

Each entry should start with a general overview of the work done, before getting into details. It is critical to provide hyperlinks to (or at least names of) all files related to the entry. The file list should include the version of the model and its corresponding code, the code and input files (Box 1), any other relevant files not directly included in the notebook, and comments on those files. All such files should be archived each time a substantial task is completed to ensure that the exact files used for a particular analysis can be extracted from the archive, using the notebook as an index of the archive. A master catalogue at the beginning of the notebook should provide an overview of this archive (Box 1).

Finally, the specific details logged in an entry depend on the modelling task. This information will be rather descriptive for some tasks (e.g., problem formulation, parameterization, or conceptual model design), and rather technical for those that involve

running simulation experiments or tests (e.g., calibration, sensitivity analysis, or model output corroboration). While these details can be incorporated in the notebook, it will often make more sense to document them in the files related to the entry. For example: papers from which ideas or data were extracted can be directly annotated in their PDF; code tests are most easily documented with notes in the computer files where they were analysed; the design and analysis of results from simulation experiments can be documented in the scripts used to run them. If documentation is kept outside of the notebook, it is essential to write a concise summary of documentation files: their purposes, how they are used, and where they are archived. If applicable, external files may also be tagged by their date of creation or change, to verify their match with the notebook entry.

To sum up: (1) the notebook is a daily log of your modelling work, should be based on quick, short and concise notes, and thus is not a major additional workload; (2) the notebook need not be redundant with other documents that each describe some of the modelling work, but rather can be an index of all the work done on each modelling task and can document material that does not belong elsewhere or will be intentionally excluded from public documents; (3) whatever you do every day, log it in the notebook and tag it with TRACE tasks and keywords; (4) keeping a notebook this way facilitates iterative work; and (5) the notebook must be a master catalogue of the project's file archive, which itself often includes essential documentation.

4.2. Details for specific modelling tasks

In this section, we present details and checklists of the specific information that should be logged for each task of the modelling cycle, following the structure and organization of TRACE documents (the keywords in Table 1 are italicized here). The rationale of each TRACE category is explained in the text while the specifics of how to fill out the notebook are in Tables 2 and 3, which also provide examples of specific kinds of information to log. As explained earlier, information does not need to be repeated in the modelling notebook if recorded in files and documents appended and linked to it.

1. *Problem formulation*

The first element of TRACE describes the specific research or decision-making context in which the model is to be employed, with reference to potential users and the type of audience addressed. This element also specifies the precise basic or applied question(s) about an environmental system that could be answered with the model and the outputs the model will

provide to address such questions. The exact question or problem to be addressed with a model usually changes in early iterations of the modelling cycle. Sometimes initial questions are too simple or too complex or too vague, or do not support management or implementation decisions directly enough. Finding the right questions is an essential part of any modelling project, and this process must be carefully documented in the notebook (Table 2).

2. *Model description*

A TRACE document includes a complete written description of the final model, which should enable a full understanding and independent replication of the model. A "written" description can include equations and algorithms but is for people who are not necessarily modellers, and certainly not for computers. Since the ODD protocol (a standard for describing agent-based and other types of simulation models; Grimm et al., 2020a) is recommended to describe models in TRACE documents (Grimm et al., 2014), the ODD format, or an equivalent comprehensive format, is also recommended for the notebook. The model description need not be in the modelling notebook; indeed, it is typically more convenient to write it in another document and link it to the notebook. However, the process of model development is typically not reported in the model description but is important to document. Therefore, a model's notebook should describe its main features as they are implemented and tested, including tests of alternative model structures or submodel implementations (Table 2). The notebook should also document the approaches tried and abandoned and why they did not work, so time is not wasted on them in the future.

3. *Data evaluation*

This TRACE element documents the modeller's assessment of the quality of quantitative and qualitative data: parameter values, input data (spatial data, time series), and patterns observed in data; i.e., all data used for model design, parameterization, calibration, and model output corroboration (Table 2). Note that methods used to inversely parameterize the model via calibration belong in the "Model Output Verification" element. Data evaluation allows model users to identify data sources (e.g., original data, expert knowledge, literature review) and should provide a direct assessment of data variability and uncertainty.

4. *Conceptual model evaluation*

In this TRACE element, the modeller evaluates the simplifying assumptions underlying a model's design. This evaluation explicitly lists, discusses, and justifies in the notebook the model's most important conceptual decisions as they are made: selection of entities, relevant processes and essential structures, spatial and temporal scales, imposed vs. emergent system

properties, use of stochasticity, spatial heterogeneity and environmental drivers, etc. (Table 2). The extent to which the model is built upon existing theories, concepts, or earlier models should be also documented here.

MN entry tag MN keyword	**Document in the notebook/appended files:**
Problem formulation	
Research question(s) / Model purpose	A summary of the background of the modelling project. Preliminary notes regarding the question, problem, or hypothesis to be addressed, and alternative pathways to solve it. The relevant outcomes of discussions with teammates, advisors, clients, and stakeholders or other potential end users, and what these outcomes mean for the further direction of the modelling project.
Model description	
Model development	Written description of the model's structure and elements as they are implemented and tested.
Design decision(s)	All technical details and data used for tests that contrast alternative model structures or submodel implementations.
Data evaluation	
Parameterization / Patterns	All data sources, specifying where the data came from, when and where the data were collected, under which conditions, and by whom; thus, include the literature references or other sources, together with any relevant information that comes with it, e.g. where exactly in a publication or report you found the data. Often, listing the original data will make sense. Why any data or information was rejected. This information is important but unlikely to be reported elsewhere. The exact steps taken (if any procedure or software was used) in preparing data, e.g. all equations and scripts used. Any (potential) problems with the availability of input data and patterns, or during the parameterization process.
Conceptual model evaluation	
Conceptual design	Description of background information (with references) used to derive the initial conceptual model at the beginning of modelling project.

decision(s)	Any line of thought or literature research that leads or may lead to a specific assumption, working hypothesis, or method used in the model. The basis for the elements of the conceptual model can be statistical relationships, theories, probabilistic empirical rules based on expert knowledge, or, most often, elements of existing models addressing similar questions and systems. The process, during the design of the model's structure, taken to get to the final decisions, including the choices that were considered but rejected and why they were.

Table 2. Specific details to be provided in the modelling notebook (MN) or appended files to document problem formulation, model description, data evaluation, and conceptual model evaluation.

5. *Implementation verification*

This TRACE element is focused on (1) checking the computer code for errors, bugs, and oversights (*debugging*), (2) assessing whether the code actually implements the model as intended or described (*software verification*), and (3) documenting how the software's design (e.g., its interface, collection and visualization of outputs, automation of simulation experiments, runtime-error reporting; *usability tools design*) makes it usable for its purpose. In contrast to the first four elements, the modelling notebook entries for this and the following elements will be more technical than descriptive (Table 3).

6. *Model output verification*

This TRACE element deals with the evaluation of (1) how well model outputs reproduce observed patterns (*output verification*) and (2) the extent to which calibration and effects of environmental drivers were involved in obtaining good fits between model outputs and data (*calibration* and *tests on environmental drivers*). Model users need to know how much calibration was involved to make the model reproduce observed patterns and whether the fulfilment of verification criteria was driven by an in-depth study of the influence of environmental drivers (e.g., weather, climatic conditions, chemical disturbances, food availability; see Becher et al., 2014 for an example). Thus, calibration and other formal tests should be fully documented in the modelling notebook or linked files (Table 3).

7. Model analysis and application

This TRACE element concerns (1) analysis of output uncertainty and sensitivity to inputs (*uncertainty* and *sensitivity analysis*), and (2) further analyses (e.g., *robustness analysis*) to better understand what mechanisms drive key model results. This element also includes *model*

application through controlled simulation experiments: using the model to address its original purpose.

Sensitivity analysis (SA) explores the model's response to changes in certain model components—typically parameters, but also input data, initial conditions, or spatial configuration (Ligmann-Zielinska et al., 2020). Uncertainty analysis (UA) is to understand how the uncertainty in parameter values and the model's sensitivity to parameters interact to cause uncertainty in model outputs. Therefore, for UA the parameter values must not only cover the full parameter space but also reproduce the expected probability distribution of parameters (Railsback and Grimm, 2019, Chapter 23).

The aim of Robustness Analysis (RA) is to assess how robust the explanation provided by a model is to major changes in its structure and parameter values; RA does so by exploring the conditions under which the model's ability to explain certain observations break down, i.e., a key pattern is no longer reproduced (Grimm and Berger, 2016). Unlike SA, RA is not a body of formal techniques but is currently a collection of heuristics (Grimm and Berger, 2016). Examples of RA are: (1) analysing unrealistic scenarios that cannot occur in nature or, in general, "extreme" scenarios; (2) exploring simplified versions of the model or, in contrast, adding complexity to it; and (3) detecting tipping points within the parameter space. Because RA is less formalized, it is more important to record its methods fully.

Of course, a key element of model analysis is documenting the use of the model for its intended purpose, i.e., model application. Once a model is complete, it can be applied by different users to different situations, which leads to the important question of how to keep modelling notebooks and produce TRACE documents when model developers and model users are not the same persons or organizations. TRACE documents for different applications of the same model can be identical for elements concerning model development, while model application elements differ. For each model application, the TRACE document must be updated to describe and justify the application's simulation experiments. A technical solution for doing so is to copy the original document, give a new name that refers to a new version, and indicate both old text which refers to earlier applications and new text about the current application by using different colours. The same solution was suggested for different versions of ODD model descriptions (Grimm et al. 2020, Supplement S4). Alternatively, the TRACE document could be version-controlled.

All simulation experiments, whether for model exploration, analysis, or application, should be described as empirical experiments are, by stating their purpose and providing all details required to replicate them. Together with the description of technical details listed in

Table 3, it is critical to enter in the notebook a summary of each experiment's results and the conclusions drawn from them.

8. *Model output corroboration*

This element documents any comparisons of model predictions to independent data and patterns that were not used during model development, parameterization, or verification. These independent data and patterns can be considered as secondary predictions, because the model was not designed to reproduce them. Thus, output corroboration provides model users evidence that the model is structurally realistic and that its predictions can be trusted. The information to be provided in the notebook or linked documents and files is similar to that for model output verification (Table 3).

MN entry tag MN keyword	**Document in the notebook/appended files:**
Implementation verification	
Debugging	All debugging tests performed to check for, diagnose, and fix mistakes as the program is written, indicating the code version being debugged, and describing which hypotheses were made and how they were tested, how data were collected and analysed. Every significant mistake found in the code and how it was fixed.
Software verification / Testing	Every verification test performed on each module/submodel or on the full model, indicating whether the test is conducted on the final or on an intermediate version of the module/full model, what tests (e.g., stress tests, test programs, statistical analysis of file output, independent reimplementation of submodels; see Railsback and Grimm, 2019, Chapter 6) are applied, the experiment settings, parameter ranges, and other technical details. Briefly describe test results and conclusions. If existing software is used for the analysis of test outputs, its version and exact steps taken need to be documented.
Usability tools design	All decisions regarding what results to observe and how (e.g., graphical displays, output files) through the cycle of building, testing, and using the model. Tools used to automate simulation experiments (e.g., external libraries, extensions, R packages). If relevant, design decisions about the format of input files (e.g., time series of environmental variables, maps), runtime-error reporting (e.g., format and media), and model usability.
Model output verification	

Output verification/ Goodness-of-fit	A brief overview of methods or formal tests used to assess model accuracy, and why they were chosen, including literature references. List the patterns used to verify the outputs of the model (their full description belongs to the "Data Evaluation" element).
	A description of quantitative or qualitative criteria used to decide whether a certain pattern was matched by the model.
	A summary of how well model outputs matched the patterns used for calibration or model development.
	If applicable, a brief overview of the software used to perform the analyses, and steps taken to prepare observed data for the analyses.
Calibration	The parameters that were calibrated and the reason why they were chosen for calibration.
	How parameters were calibrated, including: (1) whether parameters were calibrated in the sub- or full model, and independently or simultaneously, (2) the range of values tested for each parameter and method used to sample the entire parameter space, (3) initial conditions and simulation settings (e.g., simulation length, spatial landscape, time series of environmental drivers, values of non-calibrated parameters), (4) empirical patterns to be matched by the model, (5) fitting criteria or metrics used to quantify how well the model output matches the data (e.g., sum of squared standardized errors) and strategy (e.g., best-fit, categorical calibration), and (6) other technical details such as number of replicates of each parameter set and the software used to implement the parameter space sampling algorithm or to analyse model fit.
	A summary of results, including tables and figures if necessary.
Tests on environmental drivers	All tests performed to assess the effects of driving environmental factors on goodness-of-fit of model outputs, indicating the default and alternative settings of tested factors, and how environmental driver input differed from default input (if applicable).

Model analysis and application

Sensitivity analysis	Analyses performed to explore the sensitivity of outputs to parameters and other components. Describe which model components (e.g., parameters, initial conditions, input data, model configuration, submodels) were evaluated for sensitivity, under what conditions, and which outputs were analysed.
	For parameter sensitivity analysis (the most common SA), summarize the experimental design, indicating: (1) whether a local or global analysis was performed; (2) which parameters were assessed, over what values/ranges, or what parameter space sampling method was used in the case of global analyses; (3) the analysis technique employed, in detail; (4) settings of the other model components when this information is relevant, and (5) other technical details, such as number of replicates and the software used for statistical analysis (with links to the relevant files).
Uncertainty analysis	Information analogous to that provided for SA, but additionally, for each parameter analysed: (1) the distribution of its values (type, shape, and distribution parameters) that describes the uncertainty it is believed to have, (2) the source of these probability distributions, and (3) the algorithms used to draw random parameter values from the distributions. Statistical tests (and software) used to analyse the distribution of model results.
Robustness analysis	The purpose and rationale of the simulation experiment(s) performed.

	The pattern(s) tested.
	The element(s) of the model that were modified, for example whether and how the model's structure, environmental settings, or process representation were simplified or made more complex, or whether parameter values were varied and over what ranges.
Model application	The purpose and rationale of each simulation experiment. Justification of scenarios tested.
	If not fixed between experiments, the model structure and spatial and temporal settings.
	Other simulation settings (e.g., time step, simulation length, stop conditions, number of replicates).
	The list of model inputs that are varied (e.g., parameters, initial values of state variables, time series of environmental drivers, external maps).
	Any regime shifts, events or scheduled model forcing implemented in the simulations (providing information about how those events are scheduled, at which predefined times, to what values parameter sets/input data are changed, etc.).
	The outputs analysed, and analysis methods and results.
Model output corroboration	
Output corroboration / Validation	The empirical observations or theoretical patterns that model results were compared to (including sources, references, etc.).
	The simulation settings (e.g., simulation length, spatial configuration, environmental input, initial conditions, parameter values) and number of replicates performed.
	The tests and criteria used to assess whether observations or patterns were reproduced by the model.
	If applicable, the software used to perform the analyses, and steps taken to prepare observed data for the analyses.
	A brief description of how well model outputs match each pattern, and the consequences of these results for the modelling project.

Table 3. Information to be provided in the modelling notebook (MN) or appended files to document implementation verification, model output verification, model analysis, and model output corroboration.

9. Software development

As discussed above, one of a modelling notebook's primary purposes is recording information that will not be in a final product but is still important for project efficiency and success.

Software development produces many kinds of such information. Consequently, notes on the entire process of code design, implementation, and, perhaps, runtime optimization should be logged in the notebook, describing:

Code design decisions. (1) The implementation approaches chosen (e.g., platforms or languages, numerical methods), briefly explaining the reason why they were preferred over alternative ones. It is especially useful to document any code designs that were tried and abandoned, to remind the coder why the design did not work. (2) Any strategies for avoiding code dependency problems. (3) Strategies for future code maintenance.

Code implementation. (1) Information about how routines are programmed (e.g., the source code itself, or a link to a file and procedure names) and notes on specific implementations of functions or model controls. Citations for code imported or adapted from elsewhere should also be included here. (2) Notes about any (potential) problems with the model, specific submodel(s), input data, etc, or ideas about possible further extensions or directions of the model. Code implementation entries would often be accompanied by "model description" entries, describing the purpose and rationale of implemented elements.

Optimization and profiling. Code profiling and optimization methods and results, including what parts of the code were tested and changed and how each change affected execution speed. Once again, it is also important to document what potential code improvements were proven unhelpful.

5. Schedules, tools, and recommendations for keeping modelling notebooks

5.1. Schedules

We recommend making one or more entries to the modelling notebook daily; otherwise, important details can be forgotten or remembered in a distorted way. In addition, daily entries have an emotional benefit, as a daily log provides a written record of all the small, slow advances, which helps put seemingly minor accomplishments into perspective and increase the modeller's confidence on the modelling project. Within a day, entries can be made each time a task (e.g., running an experiment) is completed or an important outcome has been produced. A good strategy is to create an explicit list of "triggers": well-defined events at which progress is recorded. Example triggers for new notebook entries are (1) a non-trivial

information search is made and relevant results found; (2) an explicit design decision is made; (3) a measurement is taken (e.g. when profiling or testing the effect of a parameter value); (4) debugging tests for subtle problems are performed; (5) a piece of code or an entire submodel is added, changed or optimized; or (6) an atypical model behaviour or a problem that needs investigation is noticed. The list of triggers should grow with experience: a new trigger should be generated any time you wish you had recorded something that you did not.

5.2. Tools

There is a wide range of complementary tools for keeping a modelling notebook. Which ones are best for you will depend on standards used in your team and your own expertise and skills. We list some tools that are currently common, knowing that some will become outdated and new ones will appear.

5.2.1 Word processors

Word processing software is the simplest tool for keeping a modelling notebook. Word processor features that could be useful for keeping a notebook include text formatting and editing, autosaving, opening files from other software, the ability to create and use templates; the ability to insert hyperlinks, equations, other documents, images, videos and other visual content; and collaboration options (see basic characteristics of common word processors in Wikipedia contributors, 2019).

5.2.2 Spreadsheets

Like word processors, spreadsheet programs are ubiquitous, simple, and relatively suitable for keeping a notebook. The tabular format facilitates organization, e.g., with one log entry per line and separate columns for date, TRACE task, model version, notes, to-do items, and links to other documents that were modified (see basic characteristics of common spreadsheet software in Wikipedia contributors, 2020a). Spreadsheet notebooks can be sorted to, e.g., assemble all the work on one task or all the unfinished to-do items.

5.2.3 Note-taking software

There is a growing number of note-taking packages and apps that are easy to set up and use. Most of them can store notes in the cloud and synchronize them across multiple devices, and some let users upload files, embed and edit external files or programming code, record audio, snap pictures, and clip pages from the Internet (see the basic and advanced features of the many available options in Wikipedia contributors, 2020b).

Microsoft OneNote and *Evernote* are probably the most popular digital note-taking packages and, because of their wide variety of features, they are useful for modelling notebooks. *Microsoft OneNote* and *Evernote* are programs for free-form information gathering, organizing, and archiving, and they enable multi-user collaboration. Information is saved in pages organized into sections within notebooks, and the notes can be tagged, annotated, edited, searched, given attachments, and exported. Notes include not only text, tables, pictures, and drawings, but also hyperlinks, multimedia recordings, and images captured from cameras or websites. They allow offline data editing with later synchronization and merging, which allows working on multiple machines and operating systems, and enables collaboration among multiple users in a shared notebook even when they are offline.

Another interesting note-taking tool for keeping modelling notebooks is *Org-mode*. It is an outline processor within the Emacs editor, designed for keeping notes, maintaining to-do lists, and project planning with a plain-text markup language (see https://orgmode.org/). It has tools that also facilitate reproducible research as Org files can include fully functional source code blocks, which can be evaluated in place and their results can be captured in the file. Therefore, a self-contained document combining problem formulation, original data, analyses, and conclusions can be created in a way that can be reproduced by any reader using the same software tools.

5.2.4 Cloud-based collaborative document-editing tools

Online real-time document-editing collaboration tools are useful when several modellers are working on the same project and share a notebook. In this context, these tools are critical for cooperative work, streamlining workflows, and eliminating inefficiencies. They allow collaborators of the modelling project to view, edit, and work simultaneously on their modelling notebook as they work on separate modelling tasks. *Google Docs* is currently probably the most widely used collaborative document tool; however, the 2020 increase in telecommuting has led to the development of numerous others, such as *Microsoft Office Online*, *Dropbox Paper*, *Bit.ai*, *Zoho Docs*, and *Framasoft*.

5.2.5 Documentation generators

These are programming tools that assemble comments written in the code files into a documentation file. They allow cross referencing of documentation and code, so the document makes it easier to access and understand the code. If the programmer is good at documenting code design and optimization, debugging, etc., in code comments, these tools can be useful for assembling that documentation in the modelling notebook or a separate

document. Examples include *Doxygen* and *Javadoc*, and the basic features of available software are described in Wikipedia contributors (2020c).

5.2.6 Computational notebooks

These are interactive computing environments that enable users to produce notebook documents containing a complete record of a computation, including the computer code, interactive widgets, plots, descriptive texts, equations, and multimedia resources. The idea for the computational notebook can be traced back to the literate programming concept (Knuth, 1984) but has only become popular with the rise of data science. Currently the most widely adopted systems are *Mathematica*, *R Markdown*, and, especially, *Jupyter* Notebooks. Each of these computing environments allows for both performing analyses and combining code, results in multiple formats, and explanatory text into a self-contained computational narrative, which can be shared with and explored and rerun by other scientists, facilitating reproducible computational research (Perkel, 2018, Rule et al., 2019). Each notebook document keeps a historical (and dated) record of the analysis being performed and, in some cases, it can be version controlled. Computational notebooks are powerful for performing and documenting model analyses and simulation experiments, but as we have discussed throughout this paper, those are only a part of the modelling cycle and the model evaludation framework. Remember that critical details of all elements of model development (from problem formulation to conceptual model evaluation) must also be documented, either in the computational notebook or in a linked document.

5.2.7 Version control systems

Version control systems (VCSs) help software developers manage changes to source code and its documentation over time. Version control systems keep track of and uniquely identify modifications to the code, as requested by a user, from correction of a small typo to a complete redesign. VCSs generally work by maintaining a “repository” to which code updates are “committed.” When a change is committed, the user includes text describing the code modifications. Committing small incremental changes allows the software developers to go back to any previous version at any time. Hence, if a mistake is made by a contributor, earlier versions of the code can be compared to help fix the bug while minimizing disruption to other collaborators. In addition, since VCSs track every change made by each contributor, they prevent simultaneous work from conflicting. Thus, a good VCS provides file backups, synchronization, both short- and long-term undo, change tracking, and ownership, sandboxing (testing environments isolated from the repository), branching (creating a copy of the

repository that can be modified without altering the main branch) and merging (integrating two branches into one), bug tracking, and reporting. Currently, *SVN*, *Mercurial*, and, especially, *Git* are widely adopted VCSs.

Perez-Riverol et al. (2016) provide useful guidelines for using version control in scientific research. As pointed out by Rule et al. (2019), version control can complement notebook keeping because it provides the capability, when the inevitable bugs are found, to determine which model analyses or simulation experiments it affected. Online platforms such as *Github, Bitbucket*, and *Gitlab* and user interface software for these platforms (e.g., *Github desktop* and *Gitkraken*) make VCSs much easier to use. Simple structured texts (e.g., using Markdown syntax) stored in the same repository as the model code and scripts for analyses can keep track of the whole history of the modelling process and hence serve as a modelling notebook.

5.3. Recommendations

We provide the following recommendations for keeping modelling notebooks. The recommendations are primarily for inexperienced modellers and students and their instructors, but potentially valuable to any modeller not yet in the habit of keeping a notebook.

Choose the right tools. Use tools (see previous section) that suit your skills and experience as well as those of your collaborators.

Use the TRACE terminology. Using the terminology (tags and keywords) provided in Table 1 will help you organize the notebook, remember what tasks remain in the modelling cycle, and facilitate production of TRACE documents to support your work and publications. This terminology should be also used to name and organize project files.

Keep it readable. Remember that the modelling notebook must be understood not only by you but by any collaborator in the modelling project. It must be readable by people, not by machines. In particular, TRACE documents need to be useful to clients without profound modelling knowledge.

Document as you proceed, not afterwards. Otherwise, relevant information and details can be forgotten and thus never documented (see Section *Schedules*).

Treat your notebook as append-only. Add each new entry at the end of the log to keep them in chronological order, and do not edit previous entries as the modelling project evolves. The notebook's purpose is to document the modelling experience, not to have a perfect finished

piece of documentation. Documentation is cleaned up during production of a TRACE document or other formal products.

Never modify or remove anything. You never know what information can be useful in the future, or if seemingly incorrect information could be actually correct. Instead, when you modify or replace part of the model or analysis, note in the new notebook entry which old entry it updates. You can add also a note in the old entry indicating that it has been updated and when.

Create templates for each TRACE element/specific modelling task. Create templates that can be easily accessed and loaded to speed up the documentation process.

Insert graphs, equations, videos, images, and other visual content. Visual information is often easier to follow and understand than written descriptions.

Embed external documents. Sometimes it can be useful to embed external documents (e.g., a spreadsheet with parameter values, an R script, a code snippet, etc.) instead of hyperlinking them (e.g., if the notebook is meant to be shared). In this case, make sure that linked documents cannot alter the notebook.

Take advantage of OCR search. Optical Character Recognition can be used to search and retrieve information from hand-written images (scans or pictures from old paper notebooks, notes from meetings, etc.), figures, or graphs.

Keep automatic backups, and use the notebook to remind you to archive other files. Back up the notebook to avoid losing its information. Use backup technology you are comfortable with, but make sure you can always access the current version if you use multiple devices. This access is critical for collaborative teams (see collaborative tools in the previous section). The notebook should remind you to record which code and document versions you worked on, which can remind you to archive those files regularly.

Use VCS on your modelling notebook document. If you are comfortable with VCS, use it to back up and track modifications to your notebook.

Keep an efficient folder structure. Design a smart archiving system at the first stages of work. Keep files that are linked to the notebook organized, which will make it easier to reference, hyperlink, back up, and version control them. Use the terminology provided in Table 1 (tags and keywords) to organize and name folders.

Design a reproducible research workflow. When performing large or repeated simulation experiments, develop, document, and automate end-to-end workflows from raw inputs to publication-ready outputs (Kitzes et al., 2018, Essawy et al., 2020).

5.4. How to produce a TRACE document from a modelling notebook

Compiling a TRACE document of any modelling project can increase the chances that the work will be accepted (for publication, decision-making, or other purposes), used as intended, and reused for future projects. This is the step that distils information useful to others from the notebook. Grimm et al. (2014) provided a template and a short user guide on the TRACE framework. Each element in the TRACE document starts with an executive summary and then, if needed, a table of contents. TRACE documents are not meant to be read from cover to cover but selectively, so it is critical that the table of contents facilitates navigation. Similarly, a hyperlinked subject index is also worth producing. When compiling a TRACE document from the modelling notebook, emphasis is of course on the final model version used to obtain the results reported in a publication or report. In some cases, it may also be important to describe the evolution of key submodels and design decisions. Two example TRACE documents are provided in Supplement S1.

The key steps in compiling a TRACE document from a modelling notebook are: (1) Extract notebook entries by their TRACE tags, focusing on the model version used for the final results. The notebook software should make this easy. (2) Retrieve corresponding information from linked files. (3) Organize the documentation for each TRACE tag into a coherent section on its modelling tasks (using the keywords in Table 1). (4) Edit the text, figures, and tables into a coherent format. (5) Check the TRACE document for completeness. Steps 1 and 2 (and even 3) are greatly simplified by using TRACE to organize the modelling notebook (see section 4.1).

TRACE, like any standard, serves as a checklist of tasks for model development, testing, analysis, and application, and thereby provides quality assurance. Often the process of compiling a TRACE document reveals that important tests or analyses are missing; they can then be completed and recorded in the notebook. However, another important benefit of using TRACE terminology in modelling notebooks is being exposed to this checklist sooner instead of later.

While keeping a notebook facilitates the compilation of a TRACE document, this is just one of its many benefits—keeping a notebook is beneficial whether or not a TRACE document is produced. However, some kind of distillation process is necessary to compile a useful snapshot of a project's status at, e.g., the time when major deliverables are produced. Using TRACE organization, instead of a fully chronological format, can facilitate such a process. Some modellers might therefore choose, as pointed out in Section 4.1., to keep a

modelling notebook that has the structure of a TRACE document. Entries are thus chronological only within each of the TRACE elements. This approach has the benefit of having notebook entries grouped by the elements of the modelling cycle, but makes it harder to follow the incremental development of a model and its rationale. There are certainly multiple strategies for organizing information over the course of a project, all likely to be more efficient than the lack of such a strategy.

6. Discussion

We propose a standard terminology and document structure, based on the TRACE framework, to keep notebooks that document the development process of a modelling project, and describe tools and routines that can make project documentation easier. Keeping a notebook has direct and indirect benefits that exceed its costs in time and effort. We have emphasized why modellers should make the effort of keeping an organized notebook:
(1) The main benefit is increased efficiency of the workflow by facilitating iterative work; task management (e.g., knowing what has and has not been done); rapid access to the information, outcomes, files, and documentation generated along the project; and reuse of successful methods while avoiding time lost by repeating unsuccessful approaches.
(2) It increases long-term productivity by helping modellers apply successful methods to new data and contexts and even to reuse processes or code for new projects.
(3) It facilitates collaborative work in projects involving multiple modellers and software developers, for example by tracking the work done by each team member and giving members access to each others ’work; however, collaborative documentation can require more sophisticated procedures and tools, such as documentation standards and tools to deal with parallel editions.
(4) It enables reproducible—and therefore more credible—computational research by fully documenting simulation experiments, including the exact input, code, and quantitative analyses, and all technical details of the experiments, which are likely to be lost if not recorded promptly. While reproducibility in complex simulation studies can be achieved by tools that encapsulate the end-to-end workflow, from raw data to final publication-ready outputs (e.g., containerized virtual environments), documentation that fully describes the analysis is fundamental (Essawy et al., 2020).
(5) Importantly, it allows modellers to easily assemble and produce TRACE documentation of their model (i.e., “writing your TRACE document in 15 minutes per day”). A TRACE

document extracted and distilled from the notebook supports a model, and publications and decisions based on the model, by documenting the entire modelling cycle in a standard format intended for public use.

(6) Last and perhaps most importantly, as indicated in the motto of this article, keeping a notebook forces modellers to continuously reflect upon lessons learned, sharpen their questions, question assumptions, develop their stories, and make scientific writing an integral part of the daily work.

As environmental science becomes more computational, modellers need to merge the management practices of traditional science with those of data and computer science and software development. Environmental computational research must not only be reproducible but also adhere to high standards of modelling practice (e.g., EFSA, 2014). Environmental modellers must provide convincing evidence that their simulation models are thoughtfully designed, correctly implemented, thoroughly tested and validated, and that model limitations are well understood. Providing such evidence is the purpose of the TRACE documentation framework and TRACE modelling notebooks. To meet these high-level standards, modellers can borrow methods, techniques and tools for software development and software quality control from software engineers, and adopt data science principles to streamline the analytic workflow (see Lowndes et al. 2017). But the real challenge goes even further and lies in the thorough documentation of the entire modelling cycle.

Science based on modelling would be markedly improved by an established culture of keeping modelling notebooks that are routinely turned into TRACE documents in publications and reports. Transparency, reproducibility, and reliability would reach a new level, coherence within and across disciplines (Ayllón et al., 2018) would be increased, and theory development (Lorscheid et al., 2019) would be facilitated. Instead of developing models from scratch, based on ad hoc design decisions, modellers would learn from each other by speaking the same language (Vincenot et al., 2018) and using the same checklists. The most efficient way to establish such a new culture in environmental modelling is to introduce the basic principles and best practices of keeping modelling notebooks in modelling curricula. We consider early adoption of this new culture an important (but not the only) step to ensure good modelling practice in future environmental modelling. To this end, beginning modellers must be trained to keep notebooks just as students are in the field and wet laboratory.

We suggested here a particular structure and practice for keeping modelling notebooks. How well these suggestions work, for self-taught beginners, modelling students

and instructors, and also for more experienced modellers, remains to be tested. As with the ODD protocol (Grimm et al. 2006; 2010; 2020), we hope to learn from the experience of notebook and TRACE users and welcome users to provide feedback by contacting the lead author. To allow us to track, improve, and update our guidance and recommendations, we ask users to add this text to the Methods section of relevant publications: "*Model development, implementation, testing, analysis, and application was documented in a modelling notebook according to Ayllón et al. (2020), and a corresponding TRACE document (Schmolke et al., 2010: Grimm et al., 2014). The TRACE document is available in the Supplementary Material and provides evidence that the model was thoughtfully designed, correctly implemented, thoroughly tested, well understood, and appropriately used for its intended purpose*."

To conclude, as a relatively new scientific approach, simulation modelling has continuously evolving techniques; however, common documentation standards are independent of techniques and in fact are made more important by the rapid pace of technological change. The modelling cycle (Grimm and Railsback, 2005) summarises the key steps of model development, ODD (Grimm et al., 2006, 2010, 2020a) provides a protocol for describing a model, TRACE (Grimm et al., 2014) sets guidance for documenting model development, testing, analysis, and application, and finally the modelling notebook format we propose here is based on and designed to support all these standards. These four components provide a complete framework for organizing and documenting modelling projects, and facilitate good modelling practice throughout.

Acknowledgements

D. Ayllón was financially supported by the Spanish Ministry of Economy, Industry and Competitiveness through the research project CGL2017-84269-P. S. Charles is participating under the umbrella of the Graduate School H_2O'Lyon (ANR-17-EURE-0018) and "Université de Lyon" (UdL), as part of the program "Investissements d'Avenir" run by "Agence Nationale de la Recherche" (ANR). C. Piou participated under the funding from ANR-JCJC PEPPER (ANR-18-CE32-0010-01). J. G. Polhill receives funding from the Scottish Government Rural Affairs Food and Environment Strategic Research Programme. We thank four anonymous reviewers for helpful comments that improved the quality of the manuscript.

7. References

Ayllón, D., Grimm, V., Attinger, S., Hauhs, M., Simmer, C., Vereecken, H., Lischeid, G. 2018. Cross-disciplinary links in environmental systems science: Current state and claimed needs identified in a meta-review of process models. Science of the Total Environment 622-623: 954-973.

Augusiak, J., Van Den Brink, P.J., Grimm, V. 2014. Merging validation and evaluation of ecological models to 'evaludation': A review of terminology and a practical approach. Ecological Modelling 280: 117-128.

Badham, J., Elsawah, S., Guillaume, J. H., Hamilton, S. H., Hunt, R. J., Jakeman, A. J., et al. 2019. Effective modeling for Integrated Water Resource Management: a guide to contextual practices by phases and steps and future opportunities. Environmental modelling & software 116: 40-56.

Becher, M.A., Grimm, V., Thorbek, P., Horn, J., Kennedy, P.J., Osborne, J.L. 2014. BEEHAVE: a systems model of honeybee colony dynamics and foraging to explore multifactorial causes of colony failure. Journal of Applied Ecology 51: 470-482.

Crook, S.M., Davison, A.P., Plesser, H.E. 2013. Learning from the Past: Approaches for Reproducibility in Computational Neuroscience. In Bower J. (Ed.), *20 Years of Computational Neuroscience*. Springer Series in Computational Neuroscience, vol 9. New York: Springer.

Dirnagl, U., Przesdzing, I. 2016. A pocket guide to electronic laboratory notebooks in the academic life sciences. F1000Research 5: 2.

Donkin, E., Dennis, P., Ustalakov, A., Warren, J., Clare, A. 2017. Replicating complex agent based models, a formidable task. Environmental Modelling & Software 92: 142-151.

EFSA. 2014. Scientific Opinion on good modelling practice in the context of mechanistic effect models for risk assessment of plant protection products. EFSA Journal 12(3):3589, 92 pp.

EFSA. 2018. Scientific Opinion on the state of the art of Toxicokinetic/Toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms. EFSA Journal 16(8):5377, 188 pp.

Elsawah, S., Pierce, S. A., Hamilton, S. H., Van Delden, H., Haase, D., Elmahdi, A., Jakeman, A. J. 2017. An overview of the system dynamics process for integrated modelling of socio-ecological systems: Lessons on good modelling practice from five case studies. Environmental Modelling & Software 93: 127-145.

Essawy, B.T., Goodall, J.L., Voce, D., Morsy, M.M., Sadler, J.M., Choi, Y.D., ... et al. 2020. A taxonomy for reproducible and replicable research in environmental modelling. Environmental Modelling & Software, 104753.

Grimm, V., Railsback, S.F. 2005. Individual-based Modeling and Ecology. Princeton University Press, Princeton, NJ.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., et al. 2006. A standard protocol for describing individual-based and agent-based models. Ecological Modelling 198: 115-126.

Grimm, V., Berger, U., Deangelis, D.L., Polhill, J.G., Giske, J., Railsback, S.F. 2010. The ODD protocol: A review and first update. Ecological Modelling 221: 2760-2768.

Grimm, V., Augusiak, J., Focks, A., Frank, B.M., Gabsi, F., Johnston, A.S.A., et al. 2014. Towards better modelling and decision support: Documenting model development, testing, and analysis using TRACE. Ecological Modelling 280: 129-139.

Grimm, V., Berger, U. 2016. Robustness analysis: Deconstructing computational models for ecological theory and applications. Ecological Modelling 326: 162-167.

Grimm V., Railsback S.F., Vincenot C.E., Berger U., Gallagher C., DeAngelis D.L., et al. 2020a. The ODD protocol for describing agent-based and other simulation models: a second update to improve clarity, replication, and structural realism. Journal of Artificial Societies and Social Simulation 23(2), 7.

Grimm, V., Johnston, A.S.A., Thulke, H.H., Forbes, V.E., Thorbek, P. 2020b. Three questions to ask before using model output for decision support. Nature Communications.

Kanare, H.M. 1985. Writing the Laboratory Notebook. American Chemical Society, Washington D.C.

Kanza, S., C. Willoughby, N. Gibbins, R. Whitby, J. G. Frey, J. Erjavec, et al. 2017. Electronic lab notebooks: can they replace paper? Journal of Cheminformatics 9: 31.

Kitzes, J., Turek, D., Deniz, F. (Eds.). 2018. The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences. University of California Press, Oakland, CA.

Kleijnen, J.P.C. 2015. Design and Analysis of Simulation Experiments, 2nd edition. International Series in Operations Research & Management Science 230. Springer International Publishing, Cham.

Knuth, D.E. 1984. Literate programming. The Computer Journal 27: 97-111.

Lorscheid, I., Heine, B.-O., Meyer, M. 2012. Opening the ‘black box ’of simulations: increased transparency and effective communication through the systematic design of experiments. Computational & Mathematical Organization Theory 18: 22-62.

Lorscheid, I., Berger, U., Grimm, V., Meyer, M. 2019. From cases to general principles – a call for theory development through agent-based modeling. Ecological Modelling 393: 153-156.

Lee, K.S. 2003. Good laboratory notebook practices. Drug Information Journal 37: 215-219.

Ligmann-Zielinska, A., Siebers, P.-O., Magliocca, N., Parker, D.C., Grimm, V., Du, J., et al. 2020. "One Size Does Not Fit All": A Roadmap of Purpose-Driven Mixed-Method

Pathways for Sensitivity Analysis of Agent-Based Models. Journal of Artificial Societies and Social Simulation, 23, 6.

Lowndes, J.S.S., Best, B.D., Scarborough, C., Afflerbach, J.C., Frazier, M.R., O'Hara, C.C, et al. 2017. Our path to better science in less time using open data science tools. Nature Ecology & Evolution 1: 0160.

Milkowski, M., Hensel, W.M., Hohol, M. 2018. Replicability or reproducibility? On the replication crisis in computational neuroscience and sharing only relevant detail. Journal of Computational Neuroscience 45: 163-172.

Monks, T., Currie, C.S., Onggo, B.S., Robinson, S., Kunc, M., & Taylor, S.J. 2018. Strengthening the reporting of empirical simulation studies: Introducing the STRESS guidelines. Journal of Simulation 13: 55-67.

Nickla, J.T., Boehm, M.B. 2011. Proper laboratory notebook practices: protecting your intellectual property. Journal of Neuroimmune Pharmacology 6: 4-9.

Peng, R.D. 2011. Reproducible Research in Computational Science. Science 334: 1226-1227.

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F.d.V., et al. 2016. Ten Simple Rules for Taking Advantage of Git and GitHub. PLoS Computational Biology 12: e1004947.

Perkel, J.M. 2018. Why Jupyter is data scientists 'computational notebook of choice. Nature 563: 145-146.

Railsback, S.F., Grimm, V. 2019. Agent-based and Individual-based Modeling: A Practical Introduction. Second edition. Princeton University Press, Princeton, N.J.

Rougier, N.P., Hinsen, K., Alexandre, F., Arildsen, T., Barba, L.A., Benureau, F.C., et al. 2017. Sustainable computational science: the ReScience initiative. PeerJ Computer Science 3: e142.

Rule, A., Birmingham, A., Zuniga, C., Altintas, I., Huang, S.-C., Knigh, R., et al. 2019. Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. PLoS Computational Biology 15: e1007007.

Sandve, G.K., Nekrutenko, A., Taylor, J., Hovig, E. 2013. Ten Simple Rules for Reproducible Computational Research. PLOS Computational Biology 9: e1003285.

Schmolke, A., Thorbek, P., DeAngelis, D.L., Grimm, V. 2010. Ecological models supporting environmental decision making: a strategy for the future. Trends in Ecology & Evolution 25: 479-486.

Schuwirth, N., Borgwardt, F., Domisch, S., Friedrichs, M., Kattwinkel, M., Kneis, D., et al. 2019. How to make ecological models useful for environmental management. Ecological Modelling 411: 108784.

Stillman, R.A., Railsback, S.F., Giske, J., Berger, U. Grimm, V. 2015. Making Predictions in a Changing World: The Benefits of Individual-Based Ecology. Bioscience 65: 140-150.

Vincenot, C.E. 2018. How new concepts become universal scientific approaches: insights from citation network analysis of agent-based complex systems science. Proceedings of the Royal Society B: Biological Sciences 285: 1874.

Wikipedia contributors. 2019. September 25. Comparison of word processors. In *Wikipedia*, The Free Encyclopedia. Retrieved 05:56, May 22, 2020, from https://en.wikipedia.org/w/index.php?title=Comparison_of_word_processors&oldid=917729583

Wikipedia contributors. 2020a. May 11. Comparison of spreadsheet software. In *Wikipedia*, The Free Encyclopedia. Retrieved 06:00, May 22, 2020, from https://en.wikipedia.org/w/index.php?title=Comparison_of_spreadsheet_software&oldid=956099192

Wikipedia contributors. 2020b. April 25. Comparison of note-taking software. In *Wikipedia*, The Free Encyclopedia. Retrieved 06:01, May 22, 2020, from https://en.wikipedia.org/w/index.php?title=Comparison_of_note-taking_software&oldid=953002478

Wikipedia contributors. 2020c. May 12. Comparison of documentation generators. In *Wikipedia*, The Free Encyclopedia. Retrieved 06:03, May 22, 2020, from https://en.wikipedia.org/w/index.php?title=Comparison_of_documentation_generators&oldid=956291012

Zurell, D., Franklin, J., König, C., Bouchet, P.J., Dormann, C.F., Elith, J., et al. 2020. A standard protocol for reporting species distribution models. Ecography. In press.