Hien Nguyen (Ed.)

Statistics and Data Science

Research School on Statistics and Data Science, RSSDS 2019
Melbourne, VIC, Australia, July 24–26, 2019
Proceedings

Springer

RSSDS2019

Communications in Computer and Information Science 1150

More information about this series at http://www.springer.com/series/7899

Hien Nguyen (Ed.)

Statistics and Data Science

Research School on Statistics and Data Science, RSSDS 2019
Melbourne, VIC, Australia, July 24–26, 2019
Proceedings

Editor
Hien Nguyen
La Trobe University
Bundoora, VIC, Australia

Preface

Welcome to the proceedings of the RSSDS: Research School in Statistics and Data Science, held during July 26–28, 2019, in Melbourne, Australia. This was the third edition in a series of workshops; the first and second of which were called the S4D: International Research Summer School in Statistics and Data Science, held in 2017 and 2018, at the University of Caen, in Normandy, France.

The workshop was organized as a collaboration between the French and Australian statistical research communities, and was sponsored by AFRAN, ANR, Inria, La Trobe University, the region of Normandy, and the University of Caen. The workshop brought together academics, researchers, and industry practitioners of statistics and data science, to discuss numerous advances in the disciplines and their impact on the sciences and society. Attendees and presenters at the workshop covered numerous topics, including data analysis, data science, data mining, data visualization, bioinformatics, machine learning, neural networks, statistics, and probability.

This year, RSSDS received 23 submissions. After a thorough peer-review process, 11 English papers were selected for inclusion in these proceedings, which implies an acceptance rate of 47.83%. These 11 papers were presented at the workshop via poster presentations. In addition to these 11 papers, 7 invited English papers were solicited from the invited speakers of the workshop. These 7 invited papers were subjected only to a technical editing process. In total, the proceedings contain 18 high-quality English papers on various topics from data science to statistics.

The high-quality program would not have been possible without the authors who chose RSSDS 2019 as their preferred venue for their publications. Furthermore, the workshop would not have been successful if not for the work of the Program Committee members and Organizing Committee members, who put a tremendous amount of effort into organizing the event, and soliciting and reviewing the research papers that make up the program.

We hope that you enjoy reading and benefit from the proceedings of RSSDS 2019.

October 2019   Hien Nguyen

Organization

RSSDS 2019 was organized via a collaborative effort between staff and students from La Trobe University, Melbourne, Australia, along with fellow Australian academics, and French and international collaborators.

Organizing Committees

Workshop Co-chairs

Hien Nguyen	La Trobe University, Australia
Natalie Karavarsamis	La Trobe University, Australia

Program Co-chairs

Faicel Chamroukhi	University of Caen, France
Florence Forbes	Inria Grenoble Rhône-Alpes, France

Program Committee

Daniel Ahfock	The University of Queensland, Australia
Julyan Arbel	Inria Grenoble Rhône-Alpes, France
Daniel Fryer	The University of Queensland, Australia
Charles Gray	La Trobe University, Australia
Paul Kabaila	La Trobe University, Australia
Sharon Lee	University of Adelaide, Australia
Steffen Maeland	University of Bergen, Norway
Geoffrey McLachlan	The University of Queensland, Australia
Shu-Kay Ng	Griffith University, Australia
Tin Nguyen	University of Caen, France
Andriy Olenko	La Trobe University, Australia
Kok-Leong Ong	La Trobe University, Australia
Luke Prendergast	La Trobe University, Australia
Kai Qin	Swinburne University, Australia
Inga Strumke	University of Oslo, Norway
Emi Tanaka	The University of Sydney, Australia
Dianhui Wang	La Trobe University, Australia
Benjamin Wong	Monash University, Australia

Organizers

Organized and hosted by

La Trobe University, Melbourne, Australia
In Cooperation with

The University of Caen, Inria, and Springer

Sponsors

Sponsored by

AFRAN, ANR, and the region of Normandy

Contents

Truth, Proof, and Reproducibility: There's No Counter-Attack for the Codeless

Charles T. Gray[1]([✉]) [iD] and Ben Marwick[2] [iD]

[1] La Trobe University, Melbourne, Australia
charlestigray@gmail.com
[2] University of Washington, Seattle, USA
bmarwick@uw.edu

Abstract. Current concerns about reproducibility in many research communities can be traced back to a high value placed on empirical reproducibility of the physical details of scientific experiments and observations. For example, the detailed descriptions by 17th century scientist Robert Boyle of his vacuum pump experiments are often held to be the ideal of reproducibility as a cornerstone of scientific practice. Victoria Stodden has claimed that the computer is an analog for Boyle's pump – another kind of scientific instrument that needs detailed descriptions of how it generates results. In the place of Boyle's hand-written notes, we now expect code in open source programming languages to be available to enable others to reproduce and extend computational experiments. In this paper we show that there is another genealogy for reproducibility, starting at least from Euclid, in the production of proofs in mathematics. Proofs have a distinctive quality of being necessarily reproducible, and are the cornerstone of mathematical science. However, the task of the modern mathematical scientist has drifted from that of blackboard rhetorician, where the craft of proof reigned, to a scientific workflow that now more closely resembles that of an experimental scientist. So, what is proof in modern mathematics? And, if proof is unattainable in other fields, what is due scientific diligence in a computational experimental environment? How do we measure truth in the context of uncertainty? Adopting a manner of Lakatosian conversant conjecture between two mathematicians, we examine how proof informs our practice of computational statistical inquiry. We propose that a reorientation of mathematical science is necessary so that its reproducibility can be readily assessed.

Keywords: Metaresearch · Reproducibility · Mathematics

In David Auburn's Pulitzer prize-winning 2000 play *Proof*, a young mathematician, Catherine, struggles to prove to another mathematician, Hal, that her argument is not a reproduction of the intellectual work of her deceased father, a professor [2]. Her handwriting similar to her father's, there is no way to discern her proof from his. But if Catherine were a computational scientist, we would have a very different story. We reimagine Hal challenging Catherine for different mathematical questions and the reproducibility of her solutions. We consider simple to complex mathematical questions that can be answered at the blackboard, and then consider the scenario where Catherine must use a combination of mathematical and computational tools to answer a question in mathematical science. Via these scenarios, we question to what extent proof methodology continues to inform our choices as mathematical scientists become as much research software engineers[1] as they are mathematicians.

Mathematical science is the compendium of research that binds the Catherine's methodology of work indistinguishably from her father's. However, in computational science, we not only do not have a common language in the traditional sense, with programming languages such as Python, R, and C++ performing overlapping tasks, but our research workflows comprise tools and platforms and operating systems, such as Linux or Windows, as well. Many inadvertent reasons conspire so that scientists are arriving at similar problems with different approaches to data management and version control. Code scripts, arguably the most immediately analogous to mathematical proof, are but one of the many components that make up the outputs of computational science.

If Catherine were a contemporary computational mathematician, she would not only struggle to reproduce another person's work, but she would likely struggle to reproduce her own. She may be overwhelmed by the diversity of research outputs [5], and find that she needs to rewrite her work to unpick what she did with specific computational functions under specific software package releases. The language of mathematical science has changed from something we write, to something we collect. In order to diligently answer scientific questions computationally, the mathematician must now consider her work within that of a research compendium. In this paper we ask: how can we extend the certainty afforded by a mathematical proof further down the research workflow into the 'mangle of practice' [31]? We show that communities of researchers in many scientific disciplines have converged on a toolkit that borrows heavily from software engineering to robustly provides many points to verify certainty, from transparency via version control, to stress testing of algorithms. We focus on unit testing as a strong measure of certainty.

---

[1] We might argue here we employ the term *research software engineer* (RSE) as Katz and McHenry would define *Super RSE*s, developers who 'work with and support researchers, and also work in teams of RSEs who research and develop their own software, support it, grow it, sustain it, etc.' [20]. Or choose the more ambiguous Research Software Engineers Association definition of RSEs as people in academia who 'combine expertise in programming with an intricate understanding of research' [45].

1   The Technological Shift in Mathematical Inquiry

The task of a mathematical scientist in the pre-computer age was largely that of a blackboard rhetorician, where the craft of proof reigned. For a proof such as that featured in Auburn's play, the argument can often be included in the article, or as a supplementary file. This allows the reader to fully reproduce the author's reasoning, by tracing the flow of argument through the notation. As computers have become ubiquitous in research, mathematical scientists have seen their workflow shift to one that now more closely resembles that of a generic scientist, concerned with diligent analysis of observational and experimental data, mediated by computers [30]. But the answer to the question of what constitutes a diligent attempt to answer a scientific question examined in a computationally intensive analysis, is unclear, and remains defined by the era of the blackboard mathematician.

So, what is proof in mathematics, when experimental and computer-assisted methods are common? And, beyond mathematics, in fields where literal proofs are unattainable, what counts as an equivalent form of scientific certainty in a computational experimental environment? How do we measure truth in the context of uncertainty? Among the histories of science we can trace three efforts to tackle these questions. First is the empirical effort, most prominently represented by Robert Boyle (1627–1691), known for his vacuum pump experiments [34]. Boyle documented his experiments in such detail and to an extent that was uncommon at the time. He was motivated by a rejection of the secrecy common in science at his time, and by a belief in the importance of written communication of experimental expertise (as a supplement to direct witnessing of experimental procedures). Boyle's distinctive approach of extensive documentation is often cited by modern advocates of computational reproducibility [35]. Making computer code openly available to the research community is argued to be the modern equivalent of Boyle's exhaustive reporting of his equipment, materials, and procedures [22].

A second effort to firming up certainty in scientific work, concerned with statistical integrity, can be traced at least as far back as Charles Babbage (1791–1871), mathematician and inventor of some of the first mechanical computers. In his 1830 book 'Reflections on the Decline of Science in England, and on Some of Its Causes' he criticised some of his contemporaries, characterising them as 'trimmers' and 'cooks' [14]. Trimmers, he wrote, were guilty of smoothing of irregularities to make the data look extremely accurate and precise. Cooks retained only those results that fit their theory and discarded the rest [26]. These practices are now called data-dredging, or p-hacking, where data are manipulated or removed from an analysis until a desirable effect or p-value is obtained [17].

A third effort follows the history of formal logic through to the time when an equivalence between philosophical logic and computation was noted. This observation is called the Curry-Howard isomorphism or the proofs-as-programs interpretation. First stated in 1959, this correspondence proposed that proofs in some areas of mathematics, such as type theory, are exactly programs from a particular programming language [37]. The bridging concepts come from intuitionistic

logic and typed lambda calculi, which have lead to the design of computational formal proof management systems such as the Coq language. This language is designed to write mathematical definitions, execute algorithms and theorems, and check proofs [3]. This correspondence has not been extensively discussed in the context of reproducibility, but we believe it has relevance and is motivating beyond mathematics. Our view is that this logic-programming correspondence can be extended in a relaxed way beyond mathematics in proofs to scientific claims in general, such that computational languages can express those claims in ways that can establish a high degree of certainty.

Questions of confidence in scientific results are far from restricted to the domains of mathematics or computers; indeed, science is undergoing a broad reexamination under what is categorised as a crisis of inference [12]. How we reproduce scientific results is being examined across a range of disciplines [6,38]. An early answer to some of these questions is that authors should make available the code that generated the results in their paper [11,36]. These recommendations mark the emergence of a concern for computational reproducibility in mathematics. This paper extends this argument for computational reproducibility further into the workflow of modern statistical inquiry, expanding and drawing on solutions proposed by methods that privilege computational reproducibility.

Systemic problems are now being recognised in the practice of conventional applied statistics, with a tendency towards *dichotomania* [1] that reduces complex and nuanced questions to Boolean statements of `TRUE` or `FALSE`. This has diluted the trust the can be placed in scientific results, and led to a crisis of replication, where results can not easily be reproduced [12] and questionable research practices [13] proliferate.

As the conventions of statistics are called into question, it stands to reason that the research practices of the discipline of statistics itself require examination. For those practicing statistical computing, a conversation is emerging about what constitutes best practice [43]. But best practice may be unrealistic, especially for those applying statistics from fields where their background has afforded limited computational training. And thus the question is becoming reframed in terms of *good enough* standards [44] we can reasonably request of statistical practitioners. By extension, we must reconsider how we prepare students in data-analytic degree programs.

Proofs, derivations, verification, all form the work of mathematics. How do we make mathematical arguments in a computational[2] environment? In constructing mathematical arguments, we posit that we require an additional core element: unit testing for data analysis. We propose an expansion of the spectrum of reproducibility, Fig. 1, to include unit testing for data analytic algorithms facilitated by a tool such as `testthat::` [41], for answering mathematical research

---

[2] We focus in this manuscript on R packages, but the reader is invited to consider these as examples rather than definitive guidance. The same arguments hold for other languages, such as Python, and associated tools.
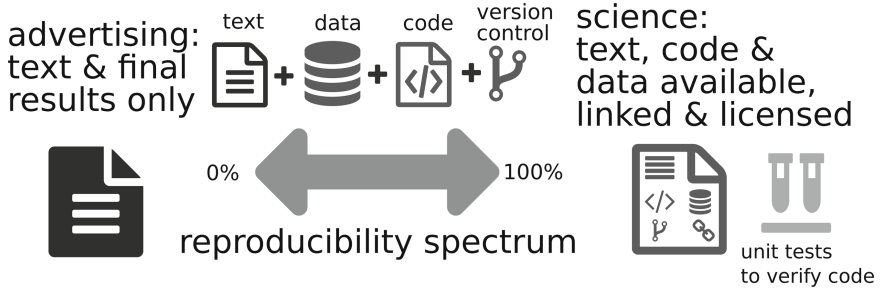
Fig. 1. We propose updating this spectrum of reproducibility [25] with unit tests for data analysis. In addition to the advertising, the **formal** scientific argument put forward, many *informal* and traditionally hidden scientific outputs comprise the compendium of research that produces the results. Given the underutilised nature of unit tests, we suggest there is further work to be done to facilitate the adoption of *good enough* [44] research software engineering practices for answering mathematical questions computationally. The informal components of mathematical research compendium are shaded grey. This figure has been adapted with permission [33] and is licensed under CC-BY 2.0.

questions computationally. In order to motivate this practice, we turn to the purest of sciences, mathematical proof.

2   Truth in Mathematics

The titular proof [2] of Auburn's play is a mathematical argument, a formalised essay in mathematical science. The creator of the proof, Catherine, is questioned by Hal, who is capable of following the argument; that is, Hal can *replicate* an approximation of the type of thought process that leads to a *reproduction* of the argument presented in the proof.

In Fig. 1, we have coloured the components, black **formal** argument, and grey *informal* work, of mathematics Hal would need to reproduce the proof. In order to verify the results, Hal would need to follow the formal argument, to understand what was written in the proof, but also need to do informal work, to understand the links between concepts for verification.

Hal would come to the problem with a different background and education to Catherine. Although work is necessary for the verification of the results, the reproduction of the reasoning, the work required would be different for Hal and Catherine, based on their respective relevant preparation. However, the language of mathematics carries enough uniformity that Hal can fill in the work he requires to understand the result, from reasoning and mathematical texts. If Catherine were asking a mathematical question computationally, the presentation of the results carries not millennia of development of methodology, as does the noble craft of mathematics, but less than a century of frequently disconnected developments separated by disparate disciplines.

We begin with traditional mathematics and end with answering questions in computational mathematics. To this aim, we adopt, in the manner of Lakatos' *Proofs and Refutations*' conversant conjecture, scenarios between Hal and Catherine, where Hal challenges Catherine over her authorship of the proof. In each scenario, we imagine the challenge would play out for different ways of answering mathematical questions. We argue the thinking work of mathematical science is not as immediately inferable in a computational experimental environment, and that the roots of mathematical science in proof lead to an overconfidence that science is as readily reproducible as a proof.

2.1   Prove It!

Let us suppose Catherine claimed she could demonstrate a property about the order[3] on natural numbers, $\mathbb{N} = \{1, 2, 3, \dots\}$, the counting numbers.

The order on a set of numbers is **dense** if, for any two numbers we can find a number in between. More formally, we say an ordered set P is dense if, for all $x < y$ in P, there exists z in P such that $x < z < y$.

Catherine presents the following argument that the order on $\mathbb{N}$ is not dense. In this case she chooses a type of *indirect* proof, an *existence* proof [4], where she presents a counterexample demonstrating that the density property is not true for all cases for $\mathbb{N}$.

Proof. **The order on $\mathbb{N}$ is not dense.** Let us, in the spirit of Lewis Carroll[4], be contrary and suppose, by way of contradiction, that the order on $\mathbb{N}$, *is* dense. Then for any two numbers, x and y, in $\mathbb{N}$ such that $x < y$, I should be able to find a distinct number z in $\mathbb{N}$ between them, that is $x < z < y$. But, consider the numbers 3 and 4. Let $x = 3$ and $y = 4$, then $x < y$. There is no distinct number, z, that exists between x and y. Since this rule must be true for any two numbers $x < y$ in the order to be dense, we have shown the order on natural numbers $\mathbb{N}$ is not dense.   □

A standard way to prove something is *not* true, is to assume it *is* true, and derive a contradiction [9]. Arguably, this reasoning goes to the heart of the problem of *dichotomania* lamented by 800 scientists in a recent protest paper about the misinterpretation of statistics in *Nature* [1]. A null hypothesis test of a difference between two groups will assume the opposite of what we suspect is true; we believe there to be a difference between two groups and take a sample from each of the groups and perform a test. This test assumes there is no difference, null, between the two groups and that any observed differences in sampling are due to

---

[3] Let P be a set. An *order* on P is a binary relation $\leqslant$ on P such that, for all $x, y, z \in P$: we have $x \leqslant x$; with $x \leqslant y$ and $y \leqslant x$ imply $x = y$; and, finally, $x \leqslant y$ and $y \leqslant z$ imply $x \leqslant z$. We then say $\leqslant$ is reflexive, antisymmetric, and transitive, for each of these properties, respectively [8].

[4] Lewis Carroll, author of *Alice in Wonderland*, is a writing pseudonym used by Charles Lutwidge Dogson, born in 1832, who taught mathematics at Christ Church, Oxford [7].

random chance. The calculation returned, the p-value, is the likelihood we would observe the difference under those null assumptions. Crucially, the calculation returned is probabilistic, a number between 0 and 1, not a `TRUE` or `FALSE`, the logic of a proof by contradiction. The logic does not apply to a situation where, within a single group of people, some people might be resistant to treatment, and some might not be, say, and we have estimated a likelihood of the efficacy of the treatment. Dichotomania is the common misinterpretation of a probabilistic response in a dichotomous framework; scientists are unwittingly framing null hypothesis significance testing in terms of a proof by contradiction.

In order to illustrate our central point, we now turn to a direct argument, rather than the indirect approach of contradiction, in order to examine the process of the making of a proof. In both the case of the direct, and indirect proofs, however, Hal could challenge Catherine, as he did in the play.

"Your dad might have written it and explained it to you later. I'm not saying he did, I'm just saying there's no proof that you wrote this" [2].

2.2 The Steps in the Making of a Proof

Let us now suppose Catherine's proof instead demonstrated a density property on the order on the real numbers,

$$\mathbb{R} = \{\ldots, -3, \ldots, -3.3, \ldots, 0, \ldots, 1, \ldots, 100.23, \ldots\},$$

i.e., the whole numbers, and the decimals between them. Catherine claims the order on $\mathbb{R}$ is dense, which is to say, if we choose any two distinct numbers in the real numbers, we can find a distinct number between them.

Catherine would construct her proof in the manner laid out in the introductory monograph *When is a Proof?* [9], in Table 1, provided to undergraduate mathematics majors at La Trobe University. These steps comprise **formal** and *informal* mathematical work, showing that mathematical *work* comprises more than the *advertising*, as it is labelled in the reproducibility spectrum presented in Fig. 1. In the case of pure mathematics, the advertising would be the paper that outlines the proof, the formal mathematical argument, but the informal work is left out.

Catherine presents the following proof to Hal to show the order on real numbers, $\mathbb{R}$, is dense.

Proof. The order on $\mathbb{R}$ is dense. Let $x < y$ in $\mathbb{R}$. Let[5] $z := (x + y)/2$. To see that $x < z < y$, we begin with $x < y$, so, $x + x < x + y$ and $x + y < y + y$, which gives,

---

[5] In mathematics, we read $:=$ as 'be defined as', $\implies$ as 'implies', and $<$ as 'less than but not equal to'.

Table 1. The steps in the making of a proof from Brian A. Davey's primer, *When is a Proof?* [9]. The formal steps that contribute to the final proof are in **bold**, the hidden informal work, in *italics*. These steps are summarised in terms of $p \implies q$ in the final column of the table.

Step -1	**Translate the statement to be proved into ordinary English and look up appropriate definitions**	
Step 0	**Write down what you are asked to prove. Where appropriate, isolate the assumptions, p, and the conclusion, q**	$p \implies q$
Step 1	**Write down the assumptions, p: "Let … "**	**Assume p**
Step 2	**Expand Step 1 by writing out definitions: "i.e., … "**	**Define p**
Step 3	*Write down the conclusion, q, which is to be proved: "To prove: …"*	*State q*
Step 4	*Expand Step 3 by writing out definitions: "i.e., …"*	*Define q*
Step 5	*Use your head: do some algebraic manipulations, draw a diagram, try to find the relationship between the assumptions and the conclusion*	*Work*
Step 6	**Rewrite your exploration from Steps 3, 4 and 5 into a proof. Justify each statement in your proof**	**Formalise work**
Step 7	**The last line of the proof**	**"Hence q."**

$$x + x < x + y < y + y$$

$$\implies \frac{x+x}{2} < \frac{x+y}{2} < \frac{y+y}{2}$$

$$\implies \frac{2x}{2} < \frac{x+y}{2} < \frac{2y}{2}$$

$$\implies x < \frac{x+y}{2} < y$$

$$\implies x < z < y,$$

since $z = (x + y)/2$, as required.

Catherine presents the formal proof, the science that in Fig. 1 is described as the advertising, a subcomponent, of the compendium of research she created in order to arrive at this argument. Hal wishes to verify the results and investigate whether Catherine merely reproduced her father's reasoning. In the case of proof, what is published is the formal argument, but as the steps in Table 1, this is not all of what makes a proof. We could think of the steps presented in Table 1 in terms of a mathematical statement $p \implies q$, which we read as p *implies q*, as given in the final column of the table. We now revisit the proof Catherine offered in terms of these steps.

We begin, step 0; we state what we wish to prove, $p \implies q$, in plain English. We wish to show the real numbers, $\mathbb{R}$, are dense; i.e., for all $x < y$ in $\mathbb{R}$, there exists z such that $x < z < y$.

Step 1, we **assume** p is true. We assume we have two distinct numbers x and y in $\mathbb{R}$ with $x < y$; i.e., x is less than y, and x is not equal to y. Step 2, nothing to define as we are familiar with $<$ and $\mathbb{R}$.

Step 3, we *state* what we wish to prove, q; the order on $\mathbb{R}$ is dense. Step 4, i.e., we need to show there exists z in $\mathbb{R}$ such that $x < z < y$. Now, Catherine has offered a solution $z := (x + y)/2$ that Hal wishes to verify.

Step 5, Suppose Hal asks, what if both x and y are negative numbers? Is it still true that $x < z < y$? Hal might verify his understanding of $+$ by thinking about positive and negative numbers as steps taken to the left or the right. In Fig. 2, Hal considers the case where both numbers are negative, $x, y < 0$. In this case, we have x steps to left, and y steps to the left, which we imagine as arrows of appropriate length. If we lay both arrows end to end, we see the number of combined steps to the left. If we consider the half-way point of x and y laid beside each other, $(x + y)/2$, we see this falls between where the arrow heads of x and y fall.
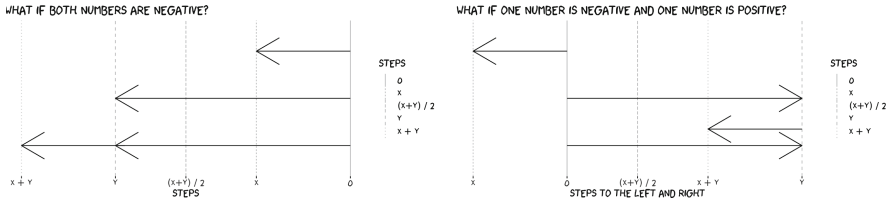
Fig. 2. On the left, Hal might begin to verify his understanding of $+$ by first considering the case where both numbers are negative, $x, y < 0$. In this case, we might think of $+$ as combining x steps to the left with y steps to the left. The halfway point $(x + y)/2$, falls in the middle of the two arrows laid side by side, which also falls between where the two ends of the arrows fall. On the right, Hal considers the case where $x < 0$, $y > 0$ and $|x| < |y|$. Here $x + y$ can be thought of as y steps to the right and then x steps to the left. Again, the halfway point $(x + y)/2$ falls halfway between the tips of the two arrows above.

Now Hal can flip the arrows in the opposite directions to construct an argument for if both numbers were positive, $x, y > 0$.

But then Hal asks in Fig. 2, what if one number were positive and one number were negative? Is $(x + y)/2$ still halfway between? Let us assume, as mathematicians say, without loss of generality that the magnitude of x is strictly less than y, that is $|x| < |y|$, the number of steps in x is less than the number of steps of y. Hal now considered where one would end up if one took y steps to the right and then x steps to left. He checks that he does not need to consider two cases, as he would end up in the same place if he took x steps to the left and then y steps to the right. Again, $(x + y)/2$ falls between where he would start and where he would end.

Now Hal has verified his understanding of $+$, which may or may not be the way that Catherine arrived at her result, but after this work he is capable of fully

reproducing the mathematical result presented. He reads the proof Catherine has provided, and verifies Steps 6, and Step 7. Catherine has proved that the order on $\mathbb{R}$ is dense. With this proof, as with the proof presented in Sect. 2.1, Hal cannot disqualify the possibility that Catherine merely reproduced her father's work.

Even in these relatively simple proofs, Step 5, the informal work of verification and understanding vastly outweighs what goes into the formal proof. But these toy examples belie a process of redefinition and re-examination, as illustrated in the discussion within a hypothetical mathematics classroom that forms the narrative of Lakatos' *Proofs and Refutations* [21]. We now move to a recently published proof to illustrate this process of redefinition.

In the Combat Conditions of New Mathematics. Suppose, now, that Catherine's proof were for the theorem pertaining to quasi-primal algebras, presented in the recent publication 'The homomorphism lattice induced by a finite algebra' [10] in *Order*, a mathematics journal devoted to 'original research on the theory and application of ordered sets'. In addition to the informal work demonstrated by the proof that the order on $\mathbb{R}$ is dense, the making of this proof involved a redefinition of the result proved, through a process writing several proofs. In terms of Table 1, initially a result was considered, $p \implies q$. A proof was written for this result. At this point the mathematicians realised, however, that the converse could be shown, that is, $q \implies p$. And so, a proof was generated for a new result, $p \iff q$. In the case of this proof, the act of writing the proof itself redefined the result in question. In the combat conditions of new mathematics, the process of writing a proof is doing mathematical science, and involves a great deal more work than is presented in the advertising of the science.

Hal may require graduate-level knowledge of abstract algebra to reproduce this proof, but as a professional mathematician, this is not a great leap. More challenging the proof may be, but the process of reproduction would be similar. Even if this were the proof, Hal would not know if Catherine merely reproduced, as he did, her father's proof.

But what if Catherine were posing her mathematical question computationally? Would Hal be able to reproduce her results?

2.3 Is Computational Mathematics Mired in Proof Methodology?

When we are exploring and answering mathematical questions in a computational environment, we consider some aspects of our work to be **formal** and some *informal*. But in omitting the greyed *informal* work in Fig. 1, are we still approaching compendia of research from the perspective of a blackboard mathematician?

Given we use statistics in most science, arguably most scientific questions are posed, to some extent, mathematically. The output format, a published paper, remains similar to mathematics of the pre-computer age. But the informal work

of answering mathematical questions has changed significantly. Now that much work is done computationally, there are multiple research outputs that comprise the compendium of science that produces the published paper.

Let us now suppose that Catherine had a statistical estimator for a population parameter of interest. That is, Catherine has an equation that, given some data, she can approximate some value about the population, such as an overall average. Let us further suppose, as is increasingly common, that she does not have a closed-form solution, meaning she cannot write out a mathematical argument in the traditional sense. Instead, she demonstrates the estimator's performance through simulation studies.

Now suppose Hal challenges Catherine to prove that she created the science that produced the paper. Given what is on the piece of paper, how can Hal know that Catherine's code does what she said it does? It is unclear what assumptions were made, about, say, sample size and distribution. How can Hal verify her results? Through adopting research software engineering principles, Catherine can facilitate a process akin to proofs and refutations, the redefinition described in the Sect. 2.2, The combat conditions of new mathematics. The process of redefinition is transcribed by version control, but further to this, the software itself provides a modular framework, such as a theorem in mathematics, for future work to scaffold and extend. New software can be developed that either extends, or redefines the existing software. One analogous way this is occurring is in the rise of metapackages, such as `tidyverse::` [42] and `metaverse::` [39], that collect software to solve particular problems in an opinionated [29] manner, that guide the end-user to what the creators consider to be good enough practice. This is analogous to classes of mathematics, such as group theory or analysis, that collect results, theorems, that rely upon each other, and where certain underlying assumptions, such as the *Axiom of Choice*[6], are made. Indeed, as Martin-Löf proposed a shift in terminology from computer science to computing science, they make the following remark.

> It has made programming an activity akin in rigour and beauty to that of proving mathematical theorems [23].

How are contemporary researchers answering mathematical questions? Alex Hayes, current maintainer and one of the many authors of `broom::` [32], an open source R package that amalgamates hundreds of contributions towards providing a suite of tools that *tidily*[7] [15] extract statistical model information from R

---

[6] Turning to the bible of algebra, *Lattices and Order* [8], we learn the *Axiom of Choice* 'asserts that it is possible to find a map which picks one element from each member of a family of non-empty sets'.

[7] From Wickham's *Tidy data* [15], we describe data as *tidy* if

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

.

algorithms, recently noted the underdeveloped nature of the implementation of statistical algorithms [16]:

> In practice, most people end up writing a reference implementation and checking that the reference implementation closely matches the pseudocode of their algorithm. Then they declare this implementation correct. How trustworthy this approach is depends on the clarity of the connection between the algorithm pseudocode and the reference implementation.

This is not to carp upon diligent scientists; we need to do far more to support the software engineering principles we expect from those who answer mathematical questions computationally [28]. Mathematicians are trained to provide enough work such that the hidden steps illustrated in italics in Table 1 can be reproduced by their target audience. The detail of mathematical work shown is tempered for level of the audience, but the same process described in bold in Table 1 is the same. But, does the workflow Alex describes above equip the target audience with enough information such that they can understand all the details of the entire argument put forward?

Code has the appearance of being highly logical, it's easy to assume it's infallible; and whilst the logic of the code is robust, the pipeline that carries the algorithm to implementation may be susceptible to compromising factors, with typos being just one example of inadvertent error.

Because code appears so logical, we assume it is analogous to proof for our intended audience to follow. But we were trained to leave out the informal messy thinking work associated with mathematics; trusting the formal argument provides enough information to verify and reproduce the mathematics. Does our code do what we think it does? In addition to providing the research outputs in the spectrum of reproducibility, Fig. 1, we posit mathematical science should adopt the software development practice of unit testing, to ensure the mathematical results can be verified and reproduced.

3   Testing

Testing is the software engineering tool that is provides a key piece of the correspondence between scientific claim and programming. Just as the Curry-Howard isomorphism expresses proofs-as-programs to link mathematics and programming, we argue that tests are the link between scientific claims more generally and programming. In a test the researcher isolates a scientifically meaningful part of their code, and creates a witness so that others can easily see that the code does what the researcher intends it to do. In this section we consider a 'vital' [40] research output, testing, that it is unlikely the mathematical scientist has been trained in. There are many such under-formalised skills represented in Fig. 1[8]. In 2016, a quarter of packages on R package archives CRAN, Biocon-

---

[8] Indeed, the natural consequence of questioning how we practice mathematical science is how we train the next generation of practitioners. Important, however this may be, this is beyond the scope of this manuscript.

ductor, and rOpenSci, included tests, a repository by repository breakdown of this is shown in Table 2.

Table 2. Percentage of R packages in repositories that have unit tests included. These results are from Jim Hester's presentation on covr:: in September 2016 [18].

Repository	Tests	Total	
CRAN	2091	9772	21%
Bioconductor	449	1258	36%
rOpenSci	84	146	58%
	2624	11,176	24%

Now, Hayes advises people against using untested software [16]. It is alarming that, by this logic, we would be **insane** to use *three quarters* of packages available. But Hayes continues, 'You have two jobs. The first job is to write correct code. The second job is to convince users that you have written correct code' [16]. The disconnect here suggests a failure to communicate broadly the importance of testing of algorithms in the dissemination of research. As researchers, we believe our science is as reproducible as a traditional mathematical proof; however, the growing literature of the replication crisis demonstrates we have not succeeded in rendering our science reproducible.

rOpenSci's review system recommends using the `covr::` [19] package to measure how the code behaves with different expected outputs. From the creator of `covr::`, we obtain the following definition of test coverage.

Test coverage is the proportion of the source code that is executed when running these tests [19].

3.1   What Is a Test?

Tests demonstrations that a given input produces an expected output. They are grouped contextually in a file; the context being a certain aspect of the algorithm that should be tested [40]. An example of a context for a test is the question, does a given function return the expected result for different inputs? Each test comprises a collection of expectations. Each expectation runs a function or functions from the package, and checks the returned output is as expected. In this case, we have a test for the `expect_equal` function: one expectation checks the function successfully runs when given equal inputs, and another expectation checks that the function fails when passed two non-equal inputs.

An example test from the `testthat::` [41] contains two expectations.

```
  test_that("basically principles of equality hold", {
 expect_success(expect_equal(1, 1))
 expect_failure(expect_equal(1, 2))
})
```

3.2   How Good Are We at *good Enough* testing?

A response to the replication crisis has been to examine *questionable research practices* [13], frequently borne of tradition and convention within different disciplines, deviate from evidence-based best-practice research methodology. We suggest it is a questionable research practice to draw conclusions about the efficacy of statistical estimators from untested code.

Given only a quarter of R packages have unit tests associated with them, we are falling short of best practice in scientific computing [43]. In a recent assessment of what constitutes *good enough* practice in scientific computing [44], unit testing was not included. However, for mathematical science, where the algorithms implemented and the code written is often complex, we suggest that unit testing should be considered good enough practice, in spite of the additional learning curve. With the backdrop of the replication crisis, it is crucial we have confidence in the algorithms we implement.

3.3   Analysis of Testing Code in R Packages

So, what packages have tests? We provide a preliminary analysis of tests in CRAN packages in Fig. 3. The code and data used to generate the results presented here are openly available at https://github.com/softloud/proof.

We provide analysis for packages associated with CRAN task view [46], opinionated [29] collections of R packages that are relevant to a particular type of statistical analysis, maintained voluntarily by experts in their respective fields [46]. CRAN task views provide a convenient taxonomy of R packages for a preliminary exploratory analysis of patterns of test use among R package authors.

Packages listed in a task view are may be interpreted by users as more stable and trustworthy than other packages, because they have passed some kind of inspection by maintainer of the task view who listed the package (however the review and curation process is not open or documented). And yet, even amongst the 4105 packages associated with task views, 1524 packages were without tests; 37 per cent of packages associated with CRAN task view were without tests.

The proportion of task view packages with tests has fallen over the last decade. This does not seem surprising given the uptake of R amongst communities of researchers in applied sciences with little formal programming and computer science training, such as psychology and ecology.

Figure 4 shows that there is wide variation in test coverage. Even the largest and fastest growing CRAN task views have very different proportions of packages with tests (Survival, about 0.23, compared to Web Technologies about 0.66). We find few clear patterns in the presence of tests over time, between different CRAN task views, and with metadata such as the number of authors, the size of the package and the centrality of the package (as measured by the union of the number of reverse dependencies and reverse imports). Based on these data, we suggest there is much work to be done in developing methods and opinionated tools that guide users towards good enough practices.
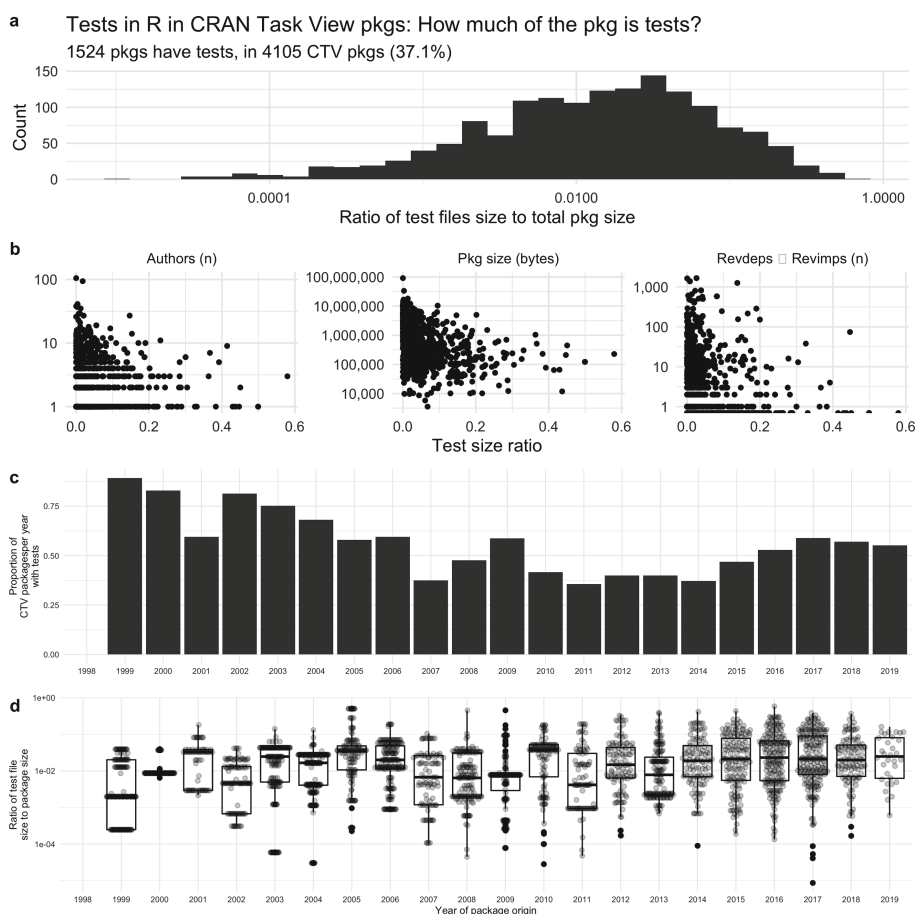
Fig. 3. This panel shows some basic details of tests in R packages listed in CRAN task views [46]. The measure of interest, *test size ratio*, was calculated by dividing the test file size with the overall package source file size from the unofficial CRAN mirror on GitHub. This is a rough indicator of test coverage, future work should consider more precise metrics such as those produced by the covr:: package. (a) the distribution of the ratio of test file size to total package size, test size ratio. (b) scatter plots demonstrate the relationship between test size ratio and number of authors, overall package size, and number of packages imported and calling the package, respectively. (c) the proportion of all task view packages that contain tests over time. (d) boxplot detailing the distribution of file size ratio over time.
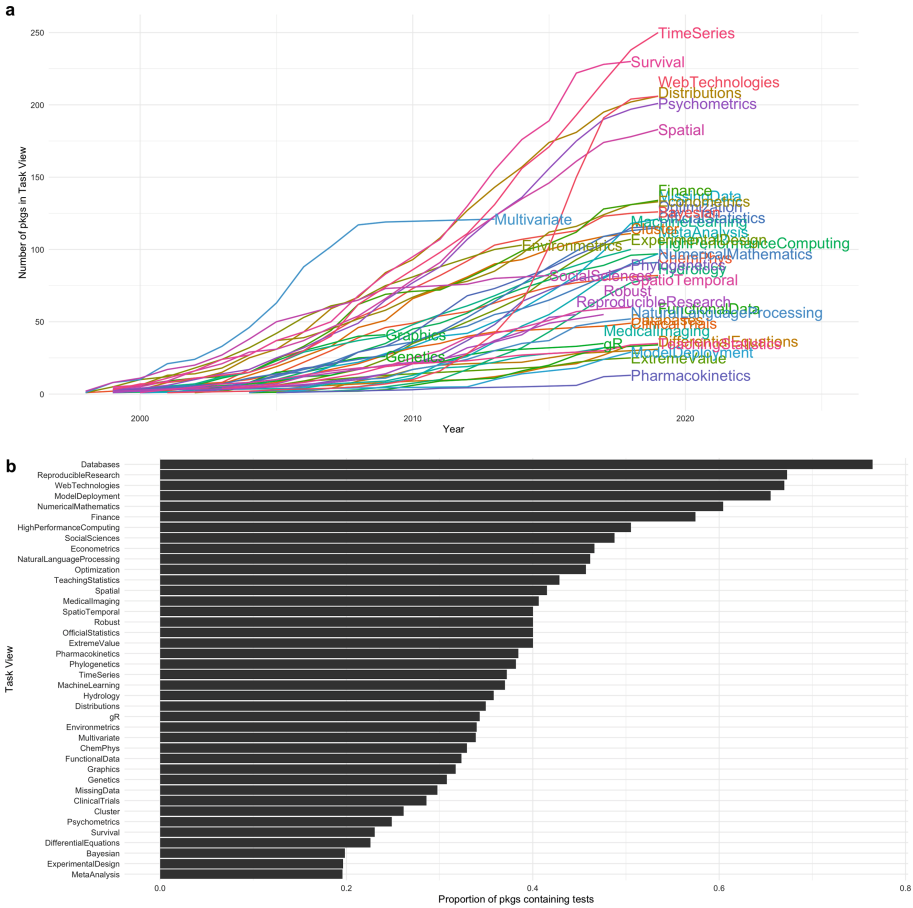
Fig. 4. (a) shows the change in the number of packages in each CRAN task view over time. (b) shows the proportion of packages in each CRAN task view that have tests.

4   Tempered Uncertainty and Computational Proof

It's easy to lie with statistics, but it's even easier without them [27]. In a computational experimental setting, we often cannot achieve the satisfying precision offered by a proof. We can, however, adopt good enough practices in sharing and testing code to increase confidence in our scientific conclusions. Given the prevalence of generalised linear models, we can think of the practice of much science as the interpretation of

$$y \approx bx,$$

where: x represents what we know about the data; y, the observed response of interest that we wish to investigate how it responds to x; and b, the *how* it responds, approximated unknown. It may not be possible to provide the rigour of

a closed-form mathematical solution, but we can aim to temper the uncertainty, and bolster confidence, in computational arguments via automated testing, version control, and other computational outputs.

We suggest there is much work to be done in developing good enough practices [44] we can ask mathematical scientists to adopt. For example, we do not have a chance to discuss in this manuscript the role of markdown and html reporting in reproducible science. Indeed, the question of good enough practice can be posed for each research output. Less than offering answers, this manuscript seeks more to suggest there is a rich line of inquiry [28] in the relationship between scientific truth, mathematical proof, and computational reproducibility and rigour.

4.1   Coda

Returning to Catherine and Hal from Auburn's *Proof* [2], we can now imagine her as computational mathematician who provides a compendium of reproducible research. To demonstrate the rigour of her computational work, she would provide unit tests for the algorithms she had implemented. Catherine would share her work openly via her GitHub or similar repository, where the development of her ideas would be timestamped and recorded. The structure of her research compendium of would be automatically standardised via a tool such as `rrtools::` [24]. At publication, her compendium would be deposited on a trustworthy, DOI-issuing repository for others to link to and cite.

And she would feel safe asking questions about good enough practice [44], and how to avoid questionable research practices [13], because there is an understanding in the community that no one is trained in all these things, so we are all always learning.

There would be no struggle, as there was in Auburn's play, to show that the mathematician who created these research outputs was Catherine. But that wouldn't matter - she and Hal would be having far too much fun collaborating on the next question.

References

1. Amrhein, V., Greenland, S., McShane, B.: Scientists rise up against statistical significance. Nature **567**(7748), 305 (2019). https://doi.org/10.1038/d41586-019-00857-9
2. Auburn, D.: Proof: A Play. Farrar, Straus and Giroux, New York (2001). Google-Books-ID: 6AUtQVhrY90C
3. Bertot, Y.: A short presentation of Coq. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 12–16. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_3
4. Brown, S.: Partial unpacking and indirect proofs: a study of students' productive use of the symbolic proof scheme. In: Proceedings of the 16th Annual Conference on Research in Undergraduate Mathematics Education, vol. 2, pp. 47–54 (2013)
5. Bryan, J.: Excuse me, do you have a moment to talk about version control? Am. Stat. **72**(1), 20–27 (2018). https://doi.org/10.1080/00031305.2017.1399928

6. Camerer, C.F., et al.: Evaluating replicability of laboratory experiments in economics. Science **351**(6280), 1433–1436 (2016). https://doi.org/10.1126/science.aaf0918
7. Carroll, L.: The Annotated Alice: The, Definitive Edition, updated, subsequent edn. W. W. Norton & Company, New York (1999)
8. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (2002). Google-Books-ID: vVVTxeuiyvQC
9. Davey, B.A.: When is a Proof?, 2nd edn. La Trobe University, Bundoora (2009)
10. Davey, B.A., Gray, C.T., Pitkethly, J.G.: The homomorphism lattice induced by a finite algebra. Order **35**(2), 193–214 (2018). https://doi.org/10.1007/s11083-017-9426-3
11. Donoho, D.L.: An invitation to reproducible computational research. Biostatistics **11**(3), 385–388 (2010). https://doi.org/10.1093/biostatistics/kxq028
12. Fidler, F., Wilcox, J.: Reproducibility of scientific results. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2018 edn. (2018)
13. Fraser, H., Parker, T., Nakagawa, S., Barnett, A., Fidler, F.: Questionable research practices in ecology and evolution. PLOS One **13**(7), e0200303 (2018). https://doi.org/10.1371/journal.pone.0200303
14. Haack, S.: Defending Science - within Reason: Between Scientism and Cynicism. Prometheus Books, Buffalo (2011). Google-Books-ID: RhXxaPTc_EYC
15. Wickham, H.: Tidy data. J. Stat. Softw. **59**(1), 1–23 (2014). https://doi.org/10.18637/jss.v059.i10
16. Hayes, A.: testing statistical software - aleatoric, July 2019. https://www.alexpghayes.com/blog/testing-statistical-software/
17. Head, M.L., Holman, L., Lanfear, R., Kahn, A.T., Jennions, M.D.: The extent and consequences of p-hacking in science. PLOS Biol. **13**(3), e1002106 (2015). https://doi.org/10.1371/journal.pbio.1002106
18. Hester, J.: covr: Bringing test coverage to R, January 2016. https://www.rstudio.com/resources/webinars/covr-bringing-test-coverage-to-r/
19. Hester, J.: covr: Test Coverage for Packages (2018). https://CRAN.R-project.org/package=covr
20. Katz, D.S., McHenry, K.: Super RSEs: Combining research and service in three dimensions of Research Software Engineering, July 2019. https://danielskatzblog.wordpress.com/2019/07/12/
21. Lakatos, I.: Proofs and Refutations: The Logic of Mathematical Discovery, reissue edn. Cambridge University Press, Cambridge (2015)
22. LeVeque, R.J., Mitchell, I.M., Stodden, V.: Reproducible research for scientific computing: tools and strategies for changing the culture. Comput. Sci. Eng. **14** (2012). https://doi.org/10.1109/mcse.2012.38
23. Martin-Löf, P.: Constructive mathematics and computer programming. In: Cohen, L.J., Łoś, J., Pfeiffer, H., Podewski, K.P. (eds.) Studies in Logic and the Foundations of Mathematics, Logic, Methodology and Philosophy of Science VI, vol. 104, pp. 153–175. Elsevier (1982). https://doi.org/10.1016/S0049-237X(09)70189-2
24. Marwick, B.: rrtools: Creates a reproducible research compendium (2018). https://github.com/benmarwick/rrtools
25. Marwick, B., Boettiger, C., Mullen, L.: Packaging data analytical work reproducibly using R (and friends). Technical report e3192v2, PeerJ Inc., March 2018. https://doi.org/10.7287/peerj.preprints.3192v2, https://peerj.com/preprints/3192
26. Merton, R.K.: On Social Structure and Science. University of Chicago Press, Chicago (1996). Google-Books-ID: j94XiVDwAZEC

27. Murray, C.: How to accuse the other guy of lying with statistics. Stat. Sci. **20**(3), 239–241 (2005). https://www.jstor.org/stable/20061179
28. Nowogrodzki, A.: How to support open-source software and stay sane. Nature **571**, 133 (2019). https://doi.org/10.1038/d41586-019-02046-0
29. Parker, H.: Opinionated analysis development. preprint (2017). https://doi.org/10.7287/peerj.preprints.3210v1
30. Peng, R.D.: Reproducible research in computational science. Science **334**(6060), 1226–1227 (2011). https://doi.org/10.1126/science.1213847
31. Pickering, A.: The Mangle of Practice: Time, Agency, and Science. University of Chicago Press, Chicago (2010)
32. Robinson, D., Hayes, A.: broom: Convert Statistical Analysis Objects into Tidy Tibbles (2019). https://CRAN.R-project.org/package=broom
33. Rodriguez-Sanchez, F., Pérez-Luque, A.J., Bartomeus, I., Varela, S.: Ciencia reproducible: qué, por qué, cómo. Revista Ecosistemas **25**(2), 83–92-92 (2016). https://doi.org/10.7818/re.2014.25-2.00, https://www.revistaecosistemas.net/index.php/ecosistemas/article/view/1178
34. Shapin, S., Schaffer, S.: Leviathan and the Air-Pump: Hobbes, Boyle, and the Experimental Life (New in paper), vol. 32. Princeton University Press, Princeton (2011)
35. Stodden, V.: What scientific idea is ready for retirement? (2014). https://www.edge.org/response-detail/25340.%202014
36. Stodden, V., Borwein, J., Bailey, D.H.: "Setting the default to reproducible" in computational science research. SIAM News **46**(5), 4–6 (2013)
37. Sørensen, M.H., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism, vol. 149. Elsevier, Amsterdam (2006)
38. Wallach, J.D., Boyack, K.W., Ioannidis, J.P.A.: Reproducible research practices, transparency, and open access data in the biomedical literature, 2015–2017. PLOS Biol. **16**(11), e2006930 (2018). https://doi.org/10.1371/journal.pbio.2006930
39. Westgate, M., et al.: metaverse: Workflows for evidence synthesis projects (2019). https://github.com/rmetaverse/metaverse, r package version 0.0.1
40. Wickham, H.: R Packages: Organize, Test, Document, and Share Your Code. O'Reilly Media, Sebastopol (2015). https://books.google.com.au/books?id=DqSxBwAAQBAJ
41. Wickham, H.: testthat: Get Started with Testing (2011)
42. Wickham, H.: tidyverse: Easily Install and Load the 'Tidyverse' (2017). https://CRAN.R-project.org/package=tidyverse
43. Wilson, G., et al.: Best practices for scientific computing. PLoS Biol. **12**(1), e1001745 (2014). https://doi.org/10.1371/journal.pbio.1001745
44. Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., Teal, T.K.: Good enough practices in scientific computing. PLOS Comput. Biol. **13**(6), e1005510 (2017). https://doi.org/10.1371/journal.pcbi.1005510
45. Wyatt, C.: Research Software Engineers Association (2019). https://rse.ac.uk/
46. Zeileis, A.: CRAN task views. R News **5**(1), 39–40 (2005). https://CRAN.R-project.org/doc/Rnews/