Towards Transparency and Open Science

A Principled Perspective on Computational Reproducibility and Preregistration

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM

eingereicht an der
Lebenswissenschaftlichen Fakultät der
Humboldt-Universität zu Berlin
von

Aaron Peikert

Betreut von:
Prof. Dr. Andreas M. Brandmaier

Betreut von:
Prof. Dr. Ulman Lindenberger

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr. Julia von Blumenthal

Dekan der Lebenswissenschaftlichen Fakultät der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. Christian Ulrichs

Towards Transparency and Open Science

A Principled Perspective on Computational Reproducibility and Preregistration

Abstract: Psychology and other empirical sciences are in the middle of a crisis, as many researchers have become aware that many findings do not have as much empirical support as they once believed. Several causes of this crisis have been suggested: misuse of statistical methods, sociological biases, and weak theories. This dissertation proposes the following rationale: to some extent, imprecise theories are unavoidable, but they still can be subjected to an empirical test by employing induction. Data may be used to amend theories, allowing precise predictions that can be compared to reality. However, such a strategy comes at a cost. While induction is necessary, it causes overconfidence in empirical findings. When assessing findings, this overconfidence must be taken into account. The extent of the overconfidence depends on the properties of the inductive process. Some inductive processes can be made fully transparent, so their bias can be accounted for appropriately. I show that this is the case for induction that can be repeated at will on other data, highlighting the importance of computational reproducibility. Induction involving the researcher and their cognitive model can not be repeated; hence, the extent of overconfidence must be judged with uncertainty. I propose that reducing this uncertainty should be the objective of preregistration. Having explicated the goals of computational reproducibility and preregistration from a perspective of transparency about induction in the synopsis, I put forward recommendations for the practice of both in the articles published as part of this dissertation.

Zusammenfassung: Die Psychologie und andere empirische Wissenschaften befinden sich in einer Krise, da vielen Forschenden bewusst geworden ist, dass viele Erkenntnisse nicht so stark empirisch gestützt sind, wie sie einst glaubten. Es wurden mehrere Ursachen dieser Krise vorgeschlagen: Missbrauch statistischer Methoden, soziologische Verzerrungen und schwache Theorien. In dieser Dissertation gehe ich davon aus, dass ungenaue Theorien unvermeidlich sind, diese aber mithilfe von Induktion einer empirischen Prüfung unterzogen werden können. Anhand von Daten können Theorien ergänzt werden, sodass präzise Vorhersagen möglich sind, die sich mit der Realität vergleichen lassen. Eine solche Strategie ist jedoch mit Kosten verbunden. Induktion ist daher zwar notwendig, aber führt zu einem übermäßigen Vertrauen in empirische Befunde. Um empirische Ergebnisse adäquat zu bewerten, muss diese Verzerrung berücksichtigt werden. Das Ausmaß der Verzerrung hängt von den Eigenschaften des induktiven Prozesses ab. Einige induktive Prozesse können vollständig transparent gemacht werden, sodass ihre Verzerrung angemessen berücksichtigt werden kann. Ich zeige, dass dies bei Induktion der Fall ist, die beliebig mit anderen Daten wiederholt werden kann, was die Bedeutung von computergestützter Reproduzierbarkeit unterstreicht. Induktion, die die Forschenden und ihr kognitives Modell einbezieht, kann nicht beliebig wiederholt werden; daher kann die Verzerrung durch Induktion nur mit Unsicherheit beurteilt werden. Ich schlage vor, dass die Verringerung dieser Unsicherheit das Ziel von Präregistrierung sein sollte. Nachdem ich die Ziele von Reproduzierbarkeit und Präregistrierung unter dem Gesichtspunkt der Transparenz über Induktion präzisiert habe, gebe ich in den wissenschaftlichen Artikeln, die als Teil der Dissertation veröffentlicht wurden, Empfehlungen für die praktische Umsetzung beider Verfahren.

Für meine Mutter.

Acknowledgments

I thank my family for their unwavering support, Caroline Gahrmann for her loving kindness, Andreas Brandmaier for his trust in me and my work, Caspar van Lissa for his graciousness when I was still a stranger to him, my friends for all the fun we have had over the years, my colleagues at MPIB for their patience with me when I was unbearable, and last but not least those many other people for their kind gestures, which helped me in more ways than I can express here.

Insight must precede application.

Contents

Legal matters

Digital copy of the work

Acknowledgement

During the work on their dissertation, Aaron Peikert was a pre-doctoral fellow of the International Max Planck Research School on Computational Methods in Psychiatry and Ageing Research (IMPRS COMP2PSYCH) of the Max Planck Institute for Human Development, Berlin, Germany, and University College London, United Kingdom.

Used resources and aids

I used several digital aids to improve the writing (not the content), including grammerly.com, deepl.com, and mentor.duden.de. Also, to aid with the clarity of the writing, I asked Andreas Brandmaier, Carli Ochs, Leo Richter, Leonie Hagitte, and Caroline Gahrmann for feedback. Furthermore, I asked Leo Richter and Maximilian Ernst for help with mathematical notation and typesetting. Julia Delius has kindly agreed to proofread the journal articles at the time of submission and the synopsis of the thesis. All the errors that remain are solely my responsibility. I also prompted `ChatGPT` to write a song about reproducibility; while it was amusing, it was not funny enough to make an appearance in the text.

Declaration of independent work

I hereby declare that I completed the doctoral thesis independently based on the stated resources and aids. I have not applied for a doctoral degree elsewhere and do not have a corresponding doctoral degree. I have not submitted the doctoral thesis, or parts of it, to another academic institution and the thesis has not been accepted or rejected. I declare that I have acknowledged the Doctoral Degree Regulations which underlie the procedure of the Faculty of Life Sciences of Humboldt-Universität zu Berlin, as amended on 5th March 2015. Furthermore, I declare that no collaboration with commercial doctoral degree supervisors took place, and that the principles of Humboldt-Universität zu Berlin for ensuring good academic practice were abided by.

Aaron Peikert

Introduction

Psychology is a difficult science (Meehl, 1978). Although there is some disagreement on why exactly this is the case, I doubt there is disagreement about the claim itself. Highlighting the difficulties and trying to overcome them are no recent trends, though they have been invigorated by the so-called replication crisis in psychology; a crisis that has also begun to ripple through other empirical sciences (Ioannidis, 2005; Open Science Collaboration, 2015). As psychology grapples with this crisis of confidence in its empirical findings, several causes have been identified and respective remedies have been suggested. The proposed countermeasures can be broadly categorized into those that aim to increase the correct use of statistical methods (e.g., Bakan, 1966; Benjamin et al., 2018; Cohen, 1994; Gigerenzer, 2004; Wagenmakers et al., 2011) and those that are designed to counteract sociological and psychological biases (Bakker et al., 2012; John et al., 2012; Rosenthal, 1979; e.g., Simmons et al., 2011).

In my view, both categories try to address a lack of transparency about the inductive process in the empirical test of a theory (Lee & Pawitan, 2021). Testing a theory empirically is often viewed as deductive since the theory is making statements about future observations. Empirical scientists, however, often simultaneously engage in induction by deriving general statements from past observations. I will argue that the inductive element in empirical tests leads to overconfidence in the empirical results if unaccounted for. The extent of this bias depends on properties of the inductive process. The induction, therefore, must be transparent to other researchers in order for them to be able to judge the empirical support of a theory.

The above distinction between statistical and sociological countermeasures arises from two sources of inductive bias. On the one hand, we have inductive processes that are well-defined in the form of statistical methods, while on the other, researchers also engage in more informal inductive behavior outside of well-defined models. These different kinds of inductive behavior require different countermeasures. When rigorously applied, statistical methods make the inductive bias quantifiable, while open science measures reduce some uncertainty about the remaining informal sources of inductive bias. To understand why transparency is crucial, it is important to comprehend how integral formal and informal induction are for empirical sciences. Any science must be able to communicate how it generates its knowledge. However, transparency has an outstanding role in psychology and other empirical sciences because empirical statements lose their value without transparency about the inductive processes involved. Therefore, transparency is more than a virtue that may improve empirical sciences somewhat, rather it is an indispensable property.

In order to function as an empirical science, psychology must be able to make statements about the world that can be compared to the actual conditions of the world. In psychology, this is not a purely deductive endeavor (Hitchcock & Sober, 2004; Meehl, 1990). Very few psychological theories are precise enough to derive testable statements (Fried, 2020; van Rooij & Baggio, 2020). While it is tempting to claim that a theory makes deductively testable claims (Lee & Pawitan, 2021) by, for example, implying a mean difference between two groups, such a statement is not testable on its own. Making inferences about a mean difference requires knowledge about the variance. Either the variance is known, then a purely deductive test is possible, or it has to be estimated from data, making induction part of the empirical test. Consider a placebo and an experimental group; there it is possible to hypothesize and test a mean difference using a simple t-test. The decision to reject the null hypothesis (groups have equal means) depends on the observed variance, besides true mean difference and sample size. However, the variance needs to be induced from the data. So the threshold of the deductive decision depends on a quantity that must be induced.

Induction is necessary and, in the present case, harmless for psychology as a science. In this case, it is innocuous because the bias from inducing the variance can be accounted for. A t-test accounts for the estimation of variance by having somewhat wider tails than a z-test which assumes that the variance is known, i.e., the threshold of significance is higher for a t-test (the critical value for a one-sided test with α-level set to 1% and degrees of freedom set to 10 is $t \approx 2.76$ vs. $z \approx 2.33$). It has been widely known for decades that even without this correction, the z-test is a good approximation when sample sizes are large (Student, 1908), i.e., the inductive bias vanishes with increasing sample size. Induction is also necessary because it is virtually impossible to ask psychologists to specify every detail, such as the variance, a priori from their theory. If they had to, there would currently be very little psychological theory that could be subjected to an empirical test (Muthukrishna & Henrich, 2019).

In other words, induction gives theories some leeway to be imprecise and contain “blank” spaces, filled in later based on observation. It allows researchers to focus on the essential statements of their theories and choose to leave some parts (such as auxiliary assumptions) to be determined inductively. In some ways, it is the empirical researchers' answer to the Duhem–Quine problem (Duhem, 1976; van Orman Quine, 1976), which states that any empirical test of a theory is testing the conjunction of theory, auxiliary assumptions, and conjectures (Meehl, 1978, 1990). Auxiliary assumptions are necessary to test a theory but do not follow from the theory itself. If such assumptions do not hold, they may lead to empirical falsification even though the theory holds. However, if an auxiliary assumption is induced, it cannot be falsified by the same data that induced it. Inducing such assumptions, therefore, effectively removes auxiliary assumptions from the conjunction that is exposed to falsification. Since empirical researchers cannot always derive every assumption from their theory, avoiding refutation because of those assumptions is a desirable property.

By the same token, entire theories may escape refutation by replacing every ill-fitting statement deduced from theory with statements induced from data. If applied to auxiliary assumptions, such a strategy of changing a theory post hoc in light of facts has been called "Lakatosian Defense" (Meehl, 1990). If pushed to the limit, we arrive at a "theory" governed by the data. Such a theory, full of empirically induced statements, is almost empty of statements that have been empirically verified. The data used for induction cannot refute these statements, so they have never been subjected to an empirical test.

So what is to be thought of such as yet untested theory? Researchers and philosophers of science differ considerably in their opinion about how to appraise theories, e.g., judging the long-term performance (if they are frequentists), degrees of belief (if they are Bayesians), or probativeness (if they are severe testers, Mayo, 2018, p. 14) of a hypothesis. Whatever measure they subscribe to, they would agree on a low appraisal of an untested theory.

Empirical researchers thus find themselves in a predicament. On the one hand, they need induction to test their imprecise theories. On the other hand, induction may render any test of a hypothesis ineffective. Therefore, I argue that the problem is not induction but making transparent where and to what extent induction is used in the inferential process. The replication crisis can be traced to a misjudgment of how much induction has been going on in psychology and hence, how well-tested the empirical claims, as reported in the literature, actually are. Therefore, the questions of this dissertation are *what* must be made transparent, and *how* to best make it transparent?

The first question (the *what*) is theoretical in nature. It is addressed in this synopsis, which supplies the theoretical framework of the articles written as part of the dissertation. Under this framework, induction is split into a process that can be formally analyzed (statistical methods) and a part that is much more difficult to judge in current research practice (sociological factors).

Based on this distinction, my research articles that are part of this dissertation answer the second question (the *how*). I argue that transparency in statistical methods is enabled by computational reproducibility, while transparency about sociological factors is facilitated by preregistration. The conceptualization of computational reproducibility and preregistration from a transparency perspective is supplemented by practical guidance on how researchers can implement these approaches.

What makes transparency necessary?

The need for transparency is closely tied to the use of induction in the empirical test of a theory. There has been a long and vigorous debate about what it means to test a theory empirically (Popper, 2002, Ch. 5, Experience as a Method). I do not attempt to rehash the debate about what constitutes an empirical test. However, I aim to examine the role of transparency in two frameworks that lend themselves to investigate the sources of inductive bias. The first framework motivates transparency when an empirical test aims to evaluate a theory's verisimilitude ("closeness to truth"). The second framework motivates transparency under a science that wants to select a theory according to its expected predictive performance.

Both frameworks show how unaccounted induction leads to overconfidence in empirical results and imply some theoretical tools to disclose induction so that this overconfidence can be assessed and controlled. Because both frameworks are quite technical, they are followed by a more conceptual summary of these ideas. These sections provide the basis for understanding how computational reproducibility and preregistration enable a proper assessment of an empirical test on a conceptual level.

An information-theoretic perspective

Information theory provides a rigorous mathematical measure that can be understood as the verisimilitude of a theory (Niiniluoto, 1998). The distance to the truth can be formalized in terms of how much information about the truth is lost when the theory is used to model reality (Rosenkrantz, 1980). Expressed mathematically, assume the existence of a function $f(x)$ that yields the likelihood of observing the state of the world x where f represents the ground truth. The quantity of interest is how much information is lost if we use $g(x)$, our theory as description of the world, instead of $f(x)$, the reality, over all possible states $\mathcal{X}$. Expressed as lost bits of information, a measure known as Kullback–Leibler divergence (Kullback & Leibler, 1951), we get:

$$\mathcal{L}_{KL}(f,g) = \int_{\mathcal{X}} f(x) \log\left(f(x)\right) \mathrm{d}x - \int_{\mathcal{X}} f(x) \log\left(g(x)\right) \mathrm{d}x \tag{1}$$

$$= \mathbb{E}_{X\sim f}\left[\log(f(x))\right] - \mathbb{E}_{X\sim f}\left[\log(g(x))\right] \ . \tag{2}$$

Many readers will recognize that this information-theoretic setup and the derivation below closely follow Burnham & Anderson (2002), Chapter 7.2, in their derivation of the Akaike Information Criterion (Akaike, 1971) in its general form. What is of interest here is not the derivation but how this conceptualization may help us to understand what happens when data is simultaneously used to induce quantities of a theory and to test the theory.

Note that in practice, $\mathcal{L}_{KL}(f,g)$ is unknowable, since the full truth is unobservable. However, this fact does not impede us from getting closer to the truth, because we can still compare the competing theories relative to each other. Because the expectation for f remains constant (left-hand expectation in Eq. (2)), we only need to estimate the relative expected loss of information (right-hand expectation in Eq. (2)) to make a comparative judgment. To make a relative judgment about several competing theories, it suffices to estimate for any theory $g(x)$:

$$\mathbb{E}_{X\sim f}\left[\log(g(x))\right] \ . \tag{3}$$

To allow quantities to be induced, we must assume that the theory is parameterized, e.g., $g(x|\theta)$. That means that the theory implies a family of possible probability distributions that may describe reality. This parameterization captures the idea that some assumptions necessary for a theory to make testable statements are arbitrary. Of those arbitrary assumptions, we want to find those that fit the reality with the least amount of information lost. The best parameterization is achieved by:

$$\theta_* = \arg\min_{\theta} \mathcal{L}_{KL}(f, g(\cdot|\theta))\,. \tag{4}$$

The inference goal for comparing other theories to $g(x)$ is, therefore:

$$\mathbb{E}_{X\sim f}\left[\log(g(x|\theta_*))\right] \ . \tag{5}$$

Of course, we usually do not know θ_*. That is why it is necessary to induce it from data, denoted as $\hat{\theta}(y)$, where Y are n independent samples from $X \sim f$.

The crucial point is to understand what happens when θ cannot be derived deductively but must be substituted inductively with an estimate $\hat{\theta}(y)$. Any estimated parameters $\hat{\theta}(y)$ would almost surely not be equal to θ_* (assuming θ may take an infinite number of values, i.e., is continuous). It follows, almost surely, that information is lost:

$$\mathcal{L}_{KL}(f, g(\cdot|\hat{\theta}(y)) > \mathcal{L}_{KL}(f, g(\cdot|\theta_*)) \, , \tag{6}$$

or

$$\mathbb{E}_{X\sim f}\left[\log(g(x|\theta_*))\right] > \mathbb{E}_{X\sim f}[\log(g(x|\hat{\theta}(y)))] \, , \tag{7}$$

ignoring the constant.

That is to say, any induced estimate will be suboptimal. The inference goal, however, is to compare the theory g to reality f, not to evaluate the estimates of $\hat{\theta}$. The point is to make a statement about the theory, not to make a statement about the data in light of the theory. If the estimate of $\hat{\theta}(y)$, i.e., the inductive process, is unbiased in the sense that it converges towards θ_*, we may form an expectation over the data Y:

$$\mathbb{E}_{Y\sim f^n}\mathbb{E}_{X\sim f}[\log(g(x|\hat{\theta}(y)))] \, . \tag{8}$$

Forming this expectation over data is a crucial step; it requires thinking beyond the observed data, of all the data we could have observed. There are two ways to get at this expectation. One is the use of Taylor series expansion, which follows in this section, and another is cross-validation, discussed in the next section.

We usually favor procedures to induce $\hat{\theta}$ that promise unbiased estimates for the observed likelihood, given that their assumptions are met. The observed likelihood is, therefore, often available, e.g., in maximum likelihood estimation. With slight abuse of notation, let $\log(g(y|\theta)) \equiv \sum_{i=1}^{n} \log(g(y_i|\theta))$, so that $\mathbb{E}_{y\sim f^n} \log(g(y|\hat{\theta}(y)))$ refers to the observed likelihood.

The expectation over the data together with Taylor series expansion yields

$$\mathbb{E}_{Y\sim f^n}\mathbb{E}_{X\sim f}[\log(g(x|\widehat{\theta}(y)))] \approx \mathbb{E}_{Y\sim f^n}[\log(g(y|\widehat{\theta}(y)))] - tr[J(\theta_*)I(\theta_*)^{-1}]\,, \tag{9}$$

where J is the Fisher information matrix with regard to g, and I for f, respectively. For more details about this derivation, see Burnham & Anderson (2002), Chapter 7.2.

The observed likelihood $\log(g(y|\widehat{\theta}(y)))$ is, therefore, a biased estimate of the distance to the truth. We may conclude that substituting deduced quantities by induced estimates leads to some overconfidence about how close one is to the truth. This overconfidence is directly related to how much induction a model entails. This bias is often called the complexity or capacity of a model, i.e., how much the data are influencing the results, hence, in how much detail the model may represent the data (Goodfellow et al., 2016, Chapter. 5.2; Mikkelson, 2001). I therefore denote it as as $\mathcal{C}$, i.e.,

$$\mathbb{E}_{X\sim f}\mathbb{E}_{Y\sim f^n}[log(g(x|\widehat{\theta}(y)))] \approx \mathbb{E}_{Y\sim f^n}[\log(g(y|\widehat{\theta}(y)))] + \mathcal{C}\,. \tag{10}$$

If we want to induce quantities and correctly appraise a theory on the same data, we must know how much we have to correct our appraisal for how the data influences the theory. Fortunately, it is possible to approximate the complexity of a model under some conditions. Since θ_* is unknown, one condition is that we know the properties of the inductive process that generated $\widehat{\theta}$. We can then formally analyze the behavior and derive a mathematical expression for $\mathcal{C}$. Corrections for a large class of statistical models, most famously the class of linear models, are well known, e.g., adjusted R^2 (Olkin & Pratt, 1958), Stein's Unbiased Risk Estimator (Stein, 1956, 1981), Mallow's C_p (Boisbunon et al., 2014; Olkin & Pratt, 1958) and information criteria (Gelman et al., 2014; Konoshi & Kitagawa, 1996).

A future-performance perspective

In addition to closeness to truth, there is another line of argumentation about why transparency about the process of induction is important. Instead of verisimilitude, one might be concerned with future performance (Yarkoni & Westfall, 2017). That is, how well does a theory do in predicting novel facts? Please note that the information-theoretic setup above has not appealed to the expected performance on unseen data. Verisimilitude and expected performance are different motivations for transparency, though they can be linked. In the future-performance setup, we do not appeal to ground truth (I drop $\sim f$, though the samples still follow some distribution), replace the Kullback–Leibler divergence with an arbitrary loss function, and no longer require $g(x)$ to return a likelihood (L stands for an arbitrary loss function):

$$\mathcal{L}(x, g(x|\theta)) = \mathbb{E}_x L(x, g(x|\theta))\,. \tag{11}$$

Again, the loss observed in the sample used to estimate $\hat{\theta}$ (Soch et al., 2020, Chapter 1.5.8) can be defined:

$$\mathbb{E}_y L(y, g(y|\hat{\theta}(y))) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, g(y_i|\hat{\theta}(y)))\,. \tag{12}$$

However, what we are interested in is not how well the theory did on data that informed it, but on future, yet unseen, data:

$$\mathbb{E}_x \mathbb{E}_y L(x, g(x|\hat{\theta}(y)))\,. \tag{13}$$

This expectation over what is often called training and test data is termed generalization error or expected prediction error (Bengio & Grandvalet, 2004). Note that both training (y) and test (x) data vary in this expectation. Therefore, it is closely related to the expectation over data shown in Eq. (8) in the information-theoretic setup (Stone, 1977).

Instead of using the Taylor series expansion, we can repeatedly sample data and repeat the inductive process. That is, we use cross-validation where the data are partitioned in n subsets of size $n-1$ and the inductive process is repeated on each subset. For each subset, the resulting model is then compared to the complement that was not used for induction, which is indicated by $y_{-i} = y \setminus \{y_i\}$.

$$\mathbb{E}_x \mathbb{E}_y L(x, g(x|\hat{\theta}(y))) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, g(y_i|\hat{\theta}(y_{-i}))) \tag{14}$$

As stated earlier, using cross-validation, it is possible to estimate $\mathbb{E}_{X\sim f}\mathbb{E}_{Y\sim f^n}[log(g(x|\hat{\theta}(y)))]$ as well; this connects the information-theoretic setup with this approach (Stone, 1974, 1977). To make the link to the first approach even more clear, we may give an alternative definition of complexity as the expected difference between observed prediction error and expected prediction error (Hauenstein et al., 2018):

$$\mathcal{C} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, g(y_i|\hat{\theta}(y_{-i}))) - L(y_i, g(y_i|\hat{\theta}(y))) \tag{15}$$

Instead of a formal analysis to derive $\mathcal{C}$, we can simply repeat the inductive process, i.e., use cross-validation. The former requires the derivatives with regard to the parameters, so I call it parametric estimator of complexity. The latter eschews the need for that, so I call it non-parametric estimator of complexity. Only requiring that a process can be repeated on other data drastically expands the set of inductive processes for which we can estimate the inductive bias.

A conceptual perspective

Now that we have established the need for transparency about the inductive process, we can glance over of the more technical details to clarify what we have to make transparent. It bears repeating that simply revealing what has been done is not enough. Merely showing the inductive results instead of the process that generated them is insufficient to appraise the theory. On a conceptual level, we want to compare the following:

$$\mathcal{L}(\text{Theory}, \text{Reality}) , \tag{16}$$

where $\mathcal{L}$ stands for the loss function, i.e., how to compare predictions and reality. To allow for induction, we replace theory with a model (not necessarily a statistical one) or, put differently, a multitude of implications about the data from the theory:

$$\mathcal{L}(\text{Model(Reality)}, \text{Reality}) . \tag{17}$$

The idea is that the version of our theory gets chosen that best fits reality. However, it is necessary to rely on a limited sample of reality. This is misleading because these two factors, induction and limited sample size, interact. Choosing the best version of the theory based on a sample is almost surely suboptimal. Therefore, the observed loss is an overconfident estimate of the loss in the future and closeness to the truth:

$$\mathcal{L}(\text{Model(Reality)}, \text{Reality}) > \mathcal{L}(\text{Model(Sample)}, \text{Sample}) . \tag{18}$$

The observed loss is thus an underestimation that has to be corrected for by considering the complexity of the model:

$$\mathcal{L}(\text{Model(Reality)}, \text{Reality}) \approx \mathcal{L}(\text{Model(Sample)}, \text{Sample}) + \mathcal{C}(\text{Model})\,. \quad (19)$$

Transparency is necessary because induction leaves researchers overly optimistic regarding their theories' fit to the data ($\mathcal{C}$). The extent of this optimism depends on the inductive process, not merely its results. Specifically, it depends on the complexity, i.e., the ability of the inductive process to adapt to data. The ability to adapt to data is independent of the specific data that were observed. Complexity is a function of the model, i.e., $\mathcal{C}(\text{Model})$, not of the data. Without knowing the inductive process, researchers cannot judge the overconfidence, so the inductive process ought to be made transparent.

How to establish transparency?

The above sections aimed to motivate the observation that the apparent fit of a theory to data is often overly optimistic if it has inductive elements. This observation is only useful if we know the extent of this optimism. However, both setups show that the extent of the optimism is closely related to the complexity of the inductive process ($\mathcal{C}(\text{Model})$) and suggest two starting points for making this bias transparent. The first requires a formal analysis of the statistical model and the inductive process it was embedded in, to compute the complexity (parametric, e.g., using information criteria). The second merely requires that the process is repeatable (nonparametric, e.g., using cross-validation). Both approaches require researchers to make the inductive process transparent rather than merely publishing the results.

Even a casual consideration of the above formalization should strike anyone who has ever worked with empirical data as unrealistic. This flexibility is often called researchers' degrees of freedom (Simmons et al., 2011). Using researchers' degrees of freedom opportunistically is an obvious problem (Wicherts et al., 2016; Yarkoni, 2022). However, no one can expect researchers to be inductive only in formally analyzable or strictly repeatable ways. The point is to set the goalpost and have a yardstick to measure how well we can judge empirical support. Without induction, there is no bias to correct for. With only formal induction, the bias can be quantified. The issue of how to deal with informal induction remains.

As a first step, it can be noted that one may split $\mathcal{C}$ in complexity that can be formally described and a second part that can only be evaluated subjectively. Formal description, in the strict sense, means that the complexity can be mathematically derived. In a looser sense, that the process can be repeated at will. If a researcher employs a linear model, the complexity is calculable parametrically. Even if it were not calculable, the linear model could be fitted on a large set of other data to assess its expected inductive behavior nonparametrically. However, suppose the researcher reconsiders the model based on their results and their internal cognitive model (including prior knowledge, expectations, cognitive biases). For example, based on worse-than-expected results and their intuition, they decided to add a predictor to the model. In that case, the complexity cannot be formally judged because the researcher is an intractable part of the inductive process. The researcher cannot be asked to repeat the process on all possible data sets, nor is their behavior mathematically well-defined. However, it is clear that the resulting linear model is more complex than a linear regression formally implies:

$$\mathcal{C}(\text{Model}) = \mathcal{C}(\text{Model}_{\text{formal}}) + \mathcal{C}(\text{Model}_{\text{informal}})\,.$$

That is not to say that there is no basis for judging the informal induction. The inductive decision can be thought of as reasonable, thus found unlikely that just about any variable would be added if the data suggests it. Or the opposite, it might not seem well justified on theoretical grounds and deemed a purely data-driven decision, which implies higher complexity. What can be said, however, is that this judgment is debatable and, therefore, subjective.

It is without question that researchers sometimes engage in inductive behavior that is neither formally analyzable nor repeatable. This fact implies that for these situations, the complexity and, hence, the optimism bias cannot be fully quantified. Though full transparency remains out of reach, researchers may do their best to provide a good basis for assessing complexity. However, judging the informal complexity $\mathcal{C}(\text{informal})$ must remain a subjective exercise.

The imperative to enable proper judgment of theories is simple: induce only what is necessary, and what is induced should, if possible, be done formally. Otherwise, the supposedly objective test of the theory using hard data must be judged more subjectively than necessary.

To summarize, I separate complexity $\mathcal{C}$ according to the transparency that can theoretically be achieved. Formal inductive processes allow full transparency in the sense that $\mathcal{C}$ can be objectively quantified. Informal inductive processes allow only limited transparency and must be subjectively judged.

Therefore, there is a theoretical bound that limits transparency about the inductive process. Some things cannot be made transparent in principle because some induction happens informally, and the complexity cannot be estimated with certainty. However, there is another bound that restricts transparency further. That is how well the inductive process, be it formal or informal, is communicated to the intended recipient. To achieve transparency about the inductive process and ultimately about the empirical support of a theory, both boundaries have to be increased.

In the following, I propose preregistration as a means to move induction into the formal domain (pushing the first boundary) and computational reproducibility to make formal induction transparent (pushing the second boundary). I argue that both practices should ultimately provide sufficient transparency to judge empirical support for theories, although they may have secondary benefits for science.

Transparency about statistical models: Computational reproducibility

Assume that a researcher only engaged in formal induction while testing a theory empirically. If other researchers want to judge this empirical test properly, the inductive process must be communicated to them. In other words, formal induction is needed, but it still must be made transparent in practice. Such transparency can be achieved by computational reproducibility. Computational reproducibility is usually defined as the ability to recreate the same results from the same data set (Claerbout & Karrenbach, 1992; Peikert, van Lissa, et al., 2021; Peikert & Brandmaier, 2021). However, from the perspective of transparency about the inductive process, it may be possible to further refine our view on computational reproducibility.

Consider software that delivers the same results upon input of the same data, i.e., it enables reproducible analyses. However, place two restrictions on the software. First, while able to compute results based on data, the software cannot be understood by human researchers, e.g., because it is an executable program, of which the source code may not be openly available. Second, when the data set changes, the software does not work anymore. The first condition rules out that the software can be analyzed formally. The second condition prevents changes of the data and observation of the results. Therefore, there are no means to assess the complexity of the inductive process and the empirical support cannot be judged. It follows that to provide transparency about the complexity of an inductive process, the definition of computational reproducibility must be broadened, i.e., at least one of the above conditions needs to be ruled out..

Suppose the second condition is relaxed; then there is a “black box” that is repeatable on other data. In that case, there still is transparency about the inductive process because the complexity of the inductive process can be estimated, i.e., using cross-validation. Relaxing the first condition, missing transparency for human researchers, most likely leads to relaxation of the second condition, repeatability on other data, as well. If humans can understand the inductive process, complexity can either be calculated analytically or the process can be reimplemented to work on similar data. Therefore, the traditional definition of computational reproducibility has to be extended. Further, it is required that the process can be repeated on other similar data.

Therefore, to provide transparency, computational reproducibility has to satisfy two requirements. First, computational reproducibility must ensure that the same data lead to the same results. Second, computational reproducibility must make the inductive process repeatable on similar data.

These two requirements are not easy to meet in practice. In Peikert & Brandmaier (2021), I proposed a workflow that unifies both requirements under the objective of automating the full process from data to manuscript. Implementing computational reproducibility by automating the process from data to results fulfills both conditions needed for transparency while reducing the effort that must be invested. Verifying that the results are actually produced by the inductive process and data is then a task that a computer may fulfill without human intervention. Removing the human from the loop facilitates that the data can be easily substituted by similar data. Given enough computing power, automatic reproducibility allows one to scale the inductive process to many data sets and therefore enables the assessment of the complexity $\mathcal{C}$.

While this workflow provides transparency and facilitates judgement of the inductive process by recipients of the work (readers, collaborators, editors, etc.), it asks a lot of the researchers who create it. To address this problem and simplify the workflow's application in practice, I developed the R package `repro` (Peikert, Brandmaier, et al., 2021). I further refined and simplified the workflow to make it more accessible in Peikert, van Lissa, et al. (2021) and contributed to van Lissa et al. (2021)'s manuscript and the software `worcs` presented there.

These works address the problem of automating reproducibility by dividing it into four subproblems. First, it must be unambiguous what results are generated by which inductive process. This can be ensured by employing dynamic document creation (Aust & Barth, 2022; Knuth, 1984; Xie, 2015, 2019). Second, the version of the software that is used to generate the results must be known. This can be ensured by documenting the version using software management (Merkel, 2014; Wiebels & Moreau, 2021). Third, the version of the author's written code (and possibly text) must be tracked using version control systems (Chacon & Straub, 2014; van Lissa et al., 2021). Fourth, how the data relates to software and computing infrastructure must be managed by workflow automation (e.g., Feldman, 1979; Kim et al., 2022; Kinsman et al., 2021).

Transparency about human researchers: Preregistration

Computational reproducibility enables transparency about formal inductive processes. However, informal inductive processes prevent complete transparency. Therefore, it is prudent to replace informal induction by formal induction wherever possible. Nonetheless, even if a researcher has made every effort to only employ formal inductive reasoning in a reproducible manner, they must still persuade their readers that they have not engaged in informal induction.

The problem is that formal inductive reasoning can be part of an informal process. A perfectly reproducible linear regression is, from the outside, indistinguishable from one that was cherry-picked from hundreds of possible regressions. However, the expectations regarding verisimilitude and future performance should vary substantially between the cherry-picked and the simple regression.

So what would constitute a persuasive argument that the data has not influenced the results above and beyond the complexity of the formal inductive process? Researchers could simply try to explain the inductive process after the fact. This task is not easy because one needs to know what they did, as well as what they did not do, e.g., that they did not cherry-pick, and what they would have done had the data looked different. This goes back to the idea that simply knowing the outcome of an inductive process is not enough to judge its complexity. For researchers to judge the complexity, the process must be transparent to them, i.e., how would the results change if the data looked different.

Trusting a post hoc explanation of an inductive process is not entirely unreasonable. However, post hoc, formal and informal induction are indistinguishable unless the account is exhaustive. Again, the report should not only be comprehensive for the data that were observed but also for all possible data. This task is challenging to accomplish in practice. After all, researchers would have to publish the process that led to the induction, i.e., their thoughts and full mental model. Dissolving the distinction between formal and informal induction would mean surrendering the advantage a formal inductive process provides; instead of quantifying complexity, the complexity must be judged with considerable uncertainty.

A clear distinction between formal and informal induction can be provided by specifying the formal inductive process before any data is available. Such a practice is called registration (Rice & Moher, 2019) or preregistration (Nosek et al., 2018). Registration provides a strong argument that the data have not influenced the results more than the complexity of the formal inductive process implies. If the data do not yet exist, they cannot influence the results ($\mathcal{C} = 0$). Note that registration is about the flow of information, i.e., the data used for induction, and not the temporal order. Information can simply not travel backward in time, which makes registration before data acquisition so appealing.

Having no access to the data when specifying the analysis forces the researcher to think about the process of induction. They have to reason about all possible data patterns they might encounter. While this makes registration so difficult, it is exactly the information we need and from which the complexity of the inductive process may be derived.

Note that requiring induction to be provably formal (i.e., analytically analyzable or repeatable), does not imply that induction itself is limited in its extent. In other words, researchers may still be very unsure how the data will look and therefore rely on induction to fill in the blanks. However, they must be clear about what they are going to induce. A formal inductive process can be designed to accommodate a wide range of data patterns, e.g., for a novel research question or a vague theory. Several strategies reflect such uncertainties. For example, suppose it is unknown which variables from a broad set of variables are involved in a process. In that case, a variable selection mechanism can be specified. Or suppose it is unclear what functional form to expect between predictor and outcome. In that case, many statistical models exist that allow almost arbitrary functional forms (Rissanen, 1984), e.g., smoothing splines (Craven & Wahba, 1978), random forests (Ho, 1998), or neural networks (Amari, 1993). If the theory does not suffice to define an outlier, this decision can be made depending on the data, etc.

The idea of registering the inductive process without knowledge of the data to demarcate formal from informal induction suffers from two problems. First, how can this be accomplished in a way that is practical and fits into the well-established practices of the scientific community? Second, how does one account for informal induction, e.g., when, despite all considerations, the data does not behave as anticipated or the registered method is deemed unsuitable for other reasons?

In Peikert, van Lissa, et al. (2021), I address the first problem and propose a practical yet rigorous form of registration called preregistration as code (PAC). In a PAC, researchers write the intended analyses as computer code, initially based on simulated data. They include this code in a reproducible dynamic document written in the style of a traditional academic manuscript. This version, with "mock" results based on simulated data, serves as the registration. When the data have been collected, the results are updated to reflect the actual observations.

To answer the second question of how to account for informal induction, I investigate the objective of registration and how it may be separated from the objective of confirmatory science in Peikert & Brandmaier (2023). In particular, I propose to formalize the objective of registration as a reduction of uncertainty about theoretical risk. Being able to account for uncertainty is indispensable for the question of how to deal with informal induction. If induction cannot be repeated at will nor formally analyzed, there is uncertainty about the complexity of the inductive process. Theoretical risk is conceptually related to the discussed concept of complexity but put into the context of Bayesian philosophy of science. Specifically, instead of the arbitrary loss functions considered here, the loss functions in Peikert & Brandmaier (2023) are restricted to those that apply to binary evidence, and that satisfy the statistical relevancy condition (the loss function rewards the observation of evidence in favor of the theory if the evidence is more likely under the theory). For this class of loss functions, I show how to account for uncertainty caused by informal induction and that a reduction in uncertainty due to preregistration is universally beneficial for those loss functions.

Discussion

This dissertation proposes transparency about the inductive process as an indispensable property of empirical sciences. First, I address the question of *what* has to be made transparent. The need for transparency is directly related to the use of induction in empirical sciences. Induction is often necessary to test vague theories empirically. However, induction introduces a bias that leads to overconfidence. The extent of this overconfidence is a function of the inductive process. Specifically, I propose that transparency must concern the extent to which data may influence the results and which is captured by the term complexity. I show that the complexity of an inductive process can be quantified if the process can be analyzed or repeated on similar data. I call such an inductive process "formal." However, researchers often engage in informal induction, where complexity must be judged subjectively and with some uncertainty. Second, I explain *how* computational reproducibility and preregistration provide transparency about complexity. Preregistration allows a distinction between formal and informal induction, while computational reproducibility communicates the process of formal induction.

This dissertation advances the theory and application of registration and computational reproducibility. Based on theoretical considerations, I refined the requirements of computational reproducibility, so that not only the same data must produce the same results but also that the inductive process may be repeated on other similar data. I developed a workflow for computational reproducibility that satisfies these requirements (Peikert, van Lissa, et al., 2021; Peikert & Brandmaier, 2021) and built tools to apply it (Peikert, Brandmaier, et al., 2021; van Lissa et al., 2021). Based on the computational reproducible workflow, I proposed preregistration as code (PAC) as an advancement of the practice of registration (Peikert, van Lissa, et al., 2021) and developed the reduction of uncertainty about the theoretical risk as a formal objective of registration (Peikert & Brandmaier, 2023).

From theory to model

My consideration of transparency centers around the necessity to allow for induction to test imprecise theories. I find it unconvincing that with extra effort, researchers will develop remarkably better and more precise theories from thin air. Instead, I find it more plausible that psychological sciences will refine theories using data to make testable statements (Brandmaier et al., 2016) and account for the bias the induction entails. I conceptualize induction to derive testable statements as choosing among a set of implications of the theory. Such a set of possible implications and the process of choosing among them according to a loss function is what I understand as a model. It is not entirely clear how to select the process of selecting the best fitting version of the theory. Should the process of induction depend on the theory? Is the application of mathematically or computationally convenient methods, as is usual in statistical modeling, sufficient? Does the arbitrariness of the selection process need to be accounted for?

In the conceptualization of complexity, the assumption of unbiasedness was implicit (the inductive process converges towards θ_*), i.e., representing an ideal inductive process. Such an assumption implies a perfect match of the set of implications of the theories and the statistical method. Furthermore, it assumes a certain optimality of the statistical method. Such an optimality requirement is difficult to verify. Most proofs regarding optimality make restrictive assumptions, i.e., optimality depends on the data and relies on large sample behavior.

In practice, researchers are restricted by the statistical models that are available to them. A certain statistical method may imply a set of possible implications that only partially overlap with the theory. Developing more powerful inductive algorithms is a very active field of research in the machine learning community (Brandmaier et al., 2013). However, having increasingly powerful tools for induction simply means that ever more vague theories may be tested. It does not necessarily lead to the development of statistical methods that fit the theories researchers are interested in.

In the end, more than one inductive process may perhaps be applicable, and no criterion makes one preferable over the other, i.e., from the perspective of the researcher, the decision is arbitrary. I have dealt with arbitrary assumptions by choosing an inductive process and forming the expectation over all possible data. A possible direction for future research is to explore whether forming the expectation over inductive processes is similarly fruitful. One suggestion in such a direction is a method called multiverse analysis (Steegen et al., 2016), where many different data analytic decisions are systematically explored.

Comparing predictions to reality

To complicate matters further, any statistical model optimizes some specific loss function. However, the choice of loss function may be highly consequential for the appraisal of a theory. Not only is this choice consequential, but it is easily imaginable that researchers cannot agree on a metric to judge the theory.

One important example of loss functions that are necessarily subjective are prior beliefs. Thus, even researchers who agreed on a general class of metrics, e.g., the posterior probability of a theory, would appraise theories differently. Allowing loss functions to vary between researchers also implies that the complexity of the inductive process varies. It follows that the correction for complexity cannot be done by the authors for all their readers universally if the loss function and complexity may depend on the researcher assessing the theory. This again highlights the importance of making the process transparent. If the process is transparent, the readers may change the loss function and calculate complexity accordingly.

Future research

This dissertation proposes a narrow definition of transparency. I have not addressed essential parts of the scientific process, among them: How should researchers decide which theory to test? How should they go about collecting data concerning the chosen theory? When are they ready to publish their conclusions? How should these conclusions be evaluated? Which conclusions warrant publication? Of those, which warrant widespread attention of the scientific community? Which must be further substantiated?

I expect that the conceptual frameworks I have employed to address the question of transparency about the inductive process may also be enormously fruitful in that context. Applying statistical theory to the philosophy of science has a long tradition. However, I hope we might abandon the tradition of only analyzing scientific practices long after they have been established. Instead, the aim should be to analyze new practices of science, such as computational reproducibility or preregistration, as they emerge. That way, theoretical insight may shape application.

References

Akaike, H. (1971). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csáki (Eds.), *2nd International Symposium on Information Theory* (pp. 267–281).

Amari, S. (1993). A universal theorem on learning curves. *Neural Networks*, *6*(2), 161–166. https://doi.org/10.1016/0893-6080(93)90013-M

Aust, F., & Barth, M. (2022). *papaja: Prepare reproducible APA journal articles with R Markdown.* https://github.com/crsh/papaja

Bakan, D. (1966). The test of significance in psychological research. *Psychological Bulletin*, *66*(6), 423. https://doi.org/10.1037/h0020412

Bakker, M., van Dijk, A., & Wicherts, J. M. (2012). The rules of the game called psychological science. *Perspectives on Psychological Science*, *7*(6), 543–554. https://doi.org/10.1177/1745691612459060

Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research*, *5*, 1089–1105.

Benjamin, D. J., Berger, J. O., Johannesson, M., Nosek, B. A., Wagenmakers, E.-J., Berk, R., Bollen, K. A., Brembs, B., Brown, L., Camerer, C., Cesarini, D., Chambers, C. D., Clyde, M., Cook, T. D., De Boeck, P., Dienes, Z., Dreber, A., Easwaran, K., Efferson, C., … Johnson, V. E. (2018). Redefine statistical significance. *Nature Human Behaviour*, *2*(1), 6–10. https://doi.org/10.1038/s41562-017-0189-z

Boisbunon, A., Canu, S., Fourdrinier, D., Strawderman, W., & Wells, M. T. (2014). Akaike's information criterion, C_p and estimators of loss for elliptically symmetric distributions: AIC and C_p as estimators of loss. *International Statistical Review*, *82*(3), 422–439. https://doi.org/10.1111/insr.12052

Brandmaier, A. M., Prindle, J. J., McArdle, J. J., & Lindenberger, U. (2016). Theory-guided exploration with structural equation model forests. *Psychological Methods*, *21*(4), Article 566. https://doi.org/10.1037/met0000090

Brandmaier, A. M., von Oertzen, T., McArdle, J. J., & Lindenberger, U. (2013). Structural Equation Model Trees. *Psychological Methods*, *18*(1), 71–86. https://doi.org/10.1037/a0030001

Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed). Springer.

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.

Claerbout, J. F., & Karrenbach, M. (1992). Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992* (pp. 601–604). Society of Exploration Geophysicists. https://doi.org/10.1190/1.1822162

Cohen, J. (1994). The earth is round ($p < .05$). *American Psychologist*, *49*(12), 997–1003. https://doi.org/10.1037/0003-066X.49.12.997

Craven, P., & Wahba, G. (1978). Smoothing noisy data with spline functions. *Numerische Mathematik*, *31*(4), 377–403. https://doi.org/10.1007/BF01404567

Duhem, P. (1976). Physical theory and experiment. In S. G. Harding (Ed.), *Can theories be refuted? Essays on the Duhem-Quine thesis* (pp. 1–40). Springer. https://doi.org/10.1007/978-94-010-1863-0_1

Feldman, S. I. (1979). Make — a program for maintaining computer programs. *Software: Practice and Experience*, *9*(4), 255–265. https://doi.org/10.1002/spe.4380090402

Fried, E. I. (2020). Theories and models: What they are, what they are for, and what they are about. *Psychological Inquiry*, *31*(4), 336–344. https://doi.org/10.1080/1047840X.2020.1854011

Gelman, A., Hwang, J., & Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, *24*(6), 997–1016. https://doi.org/10.1007/s11222-013-9416-2

Gigerenzer, G. (2004). Mindless statistics. *The Journal of Socio-Economics*, *33*(5), 587–606. https://doi.org/10.1016/j.socec.2004.09.033

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hauenstein, S., Wood, S. N., & Dormann, C. F. (2018). Computing AIC for black-box models using generalized degrees of freedom: A comparison with cross-validation. *Communications in Statistics — Simulation and Computation*, *47*(5), 1382–1396. https://doi.org/10.1080/03610918.2017.1315728

Hitchcock, C., & Sober, E. (2004). Prediction versus accommodation and the risk of overfitting. *The British Journal for the Philosophy of Science*, *55*(1), 1–34. https://doi.org/10.1093/bjps/55.1.1

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844. https://doi.org/10.1109/34.709601

Ioannidis, J. P. A. (2005). Why most published research findings are false. *PLOS Medicine*, *2*(8), Article e124. https://doi.org/10.1371/journal.pmed.0020124

John, L. K., Loewenstein, G., & Prelec, D. (2012). Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychological Science*, *23*(5), 524–532. https://doi.org/10.1177/0956797611430953

Kim, A. Y., Herrmann, V., Barreto, R., Calkins, B., Gonzalez-Akre, E., Johnson, D. J., Jordan, J. A., Magee, L., McGregor, I. R., Montero, N., Novak, K., Rogers, T., Shue, J., & Anderson-Teixeira, K. J. (2022). Implementing GitHub Actions continuous integration to reduce error rates in ecological data collection. *Methods in Ecology and Evolution*, *13*(11), 2572–2585. https://doi.org/10.1111/2041-210X.13982

Kinsman, T., Wessel, M., Gerosa, M. A., & Treude, C. (2021). How do software developers use GitHub Actions to automate their workflows? *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 420–431. https://doi.org/10.1109/MSR52588.2021.00054

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Konoshi, S., & Kitagawa, G. (1996). Generalised information criteria in model selection. *Biometrika*, *83*(4), 875–890. https://doi.org/10.1093/biomet/83.4.875

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694

Lee, Y., & Pawitan, Y. (2021). Popper's falsification and corroboration from the statistical perspectives. In Z. Parusniková & D. Merritt (Eds.), *Karl Popper's Science and Philosophy* (pp. 121–147). Springer International Publishing. https://doi.org/10.1007/978-3-030-67036-8_7

Mayo, D. G. (2018). *Statistical inference as severe testing: How to get beyond the statistics wars*. Cambridge University Press.

Meehl, P. E. (1978). Theoretical risks and tabular asterisks: Sir Karl, Sir Ronald, and the slow progress of soft psychology. *Journal of Consulting and Clinical Psychology*, *46*(4), 806–834. https://doi.org/10.1037/0022-006X.46.4.806

Meehl, P. E. (1990). Appraising and amending theories: The strategy of Lakatosian defense and two principles that warrant it. *Psychological Inquiry*, *1*(2), 108–141. https://doi.org/10.1207/s15327965pli0102_1

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *2014*(239). http://dl.acm.org/citation.cfm?id=2600239.2600241

Mikkelson, G. M. (2001). Complexity and verisimilitude: Realism for ecology. *Biology and Philosophy*, *16*(4), 533–546. https://doi.org/10.1023/A:1011905829922

Muthukrishna, M., & Henrich, J. (2019). A problem in theory. *Nature Human Behaviour*, *3*(3), 221–229. https://doi.org/10.1038/s41562-018-0522-1

Niiniluoto, I. (1998). Verisimilitude: The third period. *The British Journal for the Philosophy of Science*, *49*(1), 1–29. https://doi.org/10.1093/bjps/49.1.1

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *The Annals of Mathematical Statistics*, *29*(1), 201–211. https://doi.org/10.1214/aoms/1177706717

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), Article aac4716. https://doi.org/10.1126/science.aac4716

Peikert, A., & Brandmaier, A. M. (2021). A reproducible data analysis workflow with R Markdown, Git, Make, and Docker. *Quantitative and Computational Methods in Behavioral Sciences*, *1*, Article e3763. https://doi.org/10.5964/qcmb.3763

Peikert, A., & Brandmaier, A. M. (2023). *Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization.* PsyArXiv. https://doi.org/10.31234/osf.io/cs8wb

Peikert, A., Brandmaier, A. M., & van Lissa, C. J. (2021). *Repro: Automated setup of reproducible workflows and their dependencies* [Manual]. https://github.com/aaronpeikert/repro

Peikert, A., van Lissa, C. J., & Brandmaier, A. M. (2021). Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once. *Psych*, *3*(4), 836–867. https://doi.org/10.3390/psych3040053

Popper, K. R. (2002). *The logic of scientific discovery.* Routledge.

Rice, D. B., & Moher, D. (2019). Curtailing the use of preregistration: A misused term. *Perspectives on Psychological Science*, *14*(6), 1105–1108. https://doi.org/10.1177/1745691619858427

Rissanen, J. (1984). Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, *30*(4), 629–636. https://doi.org/10.1109/TIT.1984.1056936

Rosenkrantz, R. D. (1980). Measuring truthlikeness. *Synthese*, *45*(3), 463–487. https://doi.org/10.1007/BF02221788

Rosenthal, R. (1979). The file drawer problem and tolerance for null results. *Psychological Bulletin*, *86*(3), 638–641. https://doi.org/10.1037/0033-2909.86.3.638

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, *22*(11), 1359–1366. https://doi.org/10.1177/0956797611417632

Soch, J., Faulkenberry, T. J., Petrykowski, K., & Allefeld, C. (2020). Law of the unconscious statistician. In *The book of statistical proofs*. Zenodo. https://doi.org/10.5281/ZENODO.4305950

Steegen, S., Tuerlinckx, F., Gelman, A., & Vanpaemel, W. (2016). Increasing transparency through a multiverse analysis. *Perspectives on Psychological Science*, *11*(5), 702–712. https://doi.org/10.1177/1745691616658637

Stein, C. M. (1956). Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, *3.1*, 197–207.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. In *The Annals of Statistics* (Vol. 9, pp. 1135–1151). Institute of Mathematical Statistics. https://doi.org/10.1214/aos/1176345632

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, *36*(2), 111–147. https://doi.org/j.2517-6161.1974.tb00994.x

Stone, M. (1977). An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 44–47. https://doi.org/j.2517-6161.1977.tb01603.x

Student. (1908). The Probable Error of a Mean. *Biometrika*, *6*(1), 1. https://doi.org/10.2307/2331554

van Lissa, C. J., Brandmaier, A. M., Brinkman, L., Lamprecht, A.-L., Peikert, A., Struiksma, M. E., & Vreede, B. M. I. (2021). WORCS: A workflow for open reproducible code in science. *Data Science*, *4*(1), 29–49. https://doi.org/10.3233/DS-210031

van Orman Quine, W. (1976). Two dogmas of empiricism. In S. G. Harding (Ed.), *Can theories be refuted? Essays on the Duhem-Quine thesis* (pp. 41–64). Springer Netherlands. https://doi.org/10.1007/978-94-010-1863-0_2

van Rooij, I., & Baggio, G. (2020). Theory development requires an epistemological sea change. *Psychological Inquiry*, *31*(4), 321–325. https://doi.org/10.1080/1047840X.2020.1853477

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., & van der Maas, H. L. J. (2011). Why psychologists must change the way they analyze their data: The case of psi: Comment on Bem (2011). *Journal of Personality and Social Psychology*, *100*(3), 426–432. https://doi.org/10.1037/a0022790

Wicherts, J. M., Veldkamp, C. L. S., Augusteijn, H. E. M., Bakker, M., van Aert, R. C. M., & van Assen, M. A. L. M. (2016). Degrees of freedom in planning, running, analyzing, and reporting psychological studies: A checklist to avoid p-hacking. *Frontiers in Psychology*, *7*. https://doi.org/fpsyg.2016.01832

Wiebels, K., & Moreau, D. (2021). Leveraging containers for reproducible psychological research. *Advances in Methods and Practices in Psychological Science*, *4*(2), 25152459211017853. https://doi.org/10.1177/25152459211017853

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.name/knitr/

Xie, Y. (2019). *knitr: A general-purpose package for dynamic report generation in R* (Version 1.22) [Computer software]. https://yihui.name/knitr/

Yarkoni, T. (2022). The generalizability crisis. *Behavioral and Brain Sciences*, *45*, Article e1. https://doi.org/10.1017/S0140525X20001685

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*(6), 1100–1122. https://doi.org/10.1177/1745691617693393

Articles

In the following, I have reprinted the articles published as part of the dissertation as they were made accessible by the journals, as allowed by the respective licenses. Note that the following articles were written collaboratively with the indicated coauthors. The annex to § 6, para. 2 of the Doctoral Degree Regulations of the Faculty of Life Sciences, amended on 05.03.2015, *University Gazette of Humboldt-Universität zu Berlin 12* will be submitted with the initiation of the doctoral degree procedure.

A reproducible data analysis workflow With R Markdown, Git, Make, and Docker

Peikert, A., & Brandmaier, A. M. (2021). A Reproducible Data Analysis Workflow With R Markdown, Git, Make, and Docker. *Quantitative and Computational Methods in Behavioral Sciences*, *1*, e3763. https://doi.org/10.5964/qcmb.3763

Quantitative and Computational Methods in Behavioral Sciences

PsychOpen GOLD

Method Dissemination Articles

Check for updates

A Reproducible Data Analysis Workflow With R Markdown, Git, Make, and Docker

Aaron Peikert [1,2], Andreas M. Brandmaier [1,3]

[1] *Center for Lifespan Psychology, Max Planck Institute for Human Development, Berlin, Germany.* **[2]** *Department of Psychology, Humboldt-Universität zu Berlin, Berlin, Germany.* **[3]** *Max Planck UCL Centre for Computational Psychiatry and Ageing Research, Berlin, Germany.*

Quantitative and Computational Methods in Behavioral Sciences, 2021, Article e3763, https://doi.org/10.5964/qcmb.3763

Received: 2020-05-27 • **Accepted:** 2021-01-25 • **Published (VoR):** 2021-05-11

Corresponding Author: Andreas M. Brandmaier, Max Planck Institute for Human Development, Lentzeallee 94, 14195 Berlin, Germany. E-mail: brandmaier@mpib-berlin.mpg.de

Supplementary Materials: Materials [see Index of Supplementary Materials]

Abstract

In this tutorial, we describe a workflow to ensure long-term reproducibility of R-based data analyses. The workflow leverages established tools and practices from software engineering. It combines the benefits of various open-source software tools including R Markdown, Git, Make, and Docker, whose interplay ensures seamless integration of version management, dynamic report generation conforming to various journal styles, and full cross-platform and long-term computational reproducibility. The workflow ensures meeting the primary goals that 1) the reporting of statistical results is consistent with the actual statistical results (dynamic report generation), 2) the analysis exactly reproduces at a later point in time even if the computing platform or software is changed (computational reproducibility), and 3) changes at any time (during development and post-publication) are tracked, tagged, and documented while earlier versions of both data and code remain accessible. While the research community increasingly recognizes dynamic document generation and version management as tools to ensure reproducibility, we demonstrate with practical examples that these alone are not sufficient to ensure long-term computational reproducibility. Combining containerization, dependence management, version management, and dynamic document generation, the proposed workflow increases scientific productivity by facilitating later reproducibility and reuse of code and data.

Keywords

reproducibility, R, version management, dynamic document generation, dependency management, containerization, open science

In this tutorial, we describe a workflow to ensure long-term and cross-platform reproducibility of data analyses in R (R Core Team, 2020). Reproducibility is the ability to obtain identical results from the same statistical analysis and the same data. For us, statistical results are only reproducible if their generating, computational workflow is reported completely and transparently, and remains permanently available, such that the workflow can be re-run by a different person or later in time, and that the results remain identical to those initially reported (Claerbout & Karrenbach, 1992; Heroux, Barba, Parashar, Stodden, & Taufer, 2018; The Turing Way Community et al., 2019). The need to ensure reproducibility directly follows from commonly accepted rules of good scientific practice (such as the guidelines of the German Research Foundation; Deutsche Forschungsgemeinschaft, 2019). Ensuring reproducibility is a prerequisite for replicability (the ability to reach consistent conclusions from the same analysis and *new* data), and a means to increase the trustworthiness of empirical results (Epskamp, 2019). Transparency and accessibility are central scientific values, and open, reproducible projects will increase the efficiency and veracity of knowledge accumulation (Nosek & Bar-Anan, 2012).

Here, we combine four software tools, whose interplay can guarantee full computational reproducibility of data analyses and their reporting. There are various ideas on how to enhance reproducibility (Piccolo & Frampton, 2016), four of which we believe to be particularly important: dynamic document generation: (Rule et al., 2019), version control (Barba, 2016), dependency management (Askren et al., 2016), and containerization (Clyburne-Sherin, Fei, & Green, in press). We argue that only a workflow using all four concepts in unison can guarantee confidence in reproducing a scientific report (see The Turing Way Community et al., 2019 for similar arguments). Various implementations of these concepts exist, but we consider the following four best suited for analyses centered on the R environment (R Core Team, 2020) but also allowing for external dependencies: R Markdown (Xie, Allaire, & Grolemund, 2018) for dynamic document generation, Git (Chacon & Straub, 2014) for version control, Make (Feldman, 1979) for dependency management, and Docker (Merkel, 2014) for containerization. Each of these software solutions serves a valuable meta-scientific goal (reproducibility) and increases the researchers' productivity. They are all very flexible and powerful, so their complete mastery requires a significant amount of practice. However, for our purposes, it is sufficient to master a valuable minimal subset of functions to ensure the reproducibility of scientific analyses. We recommend using RStudio, an integrated development environment (IDE) for R, which provides simplified access to essential features of some of the tools.

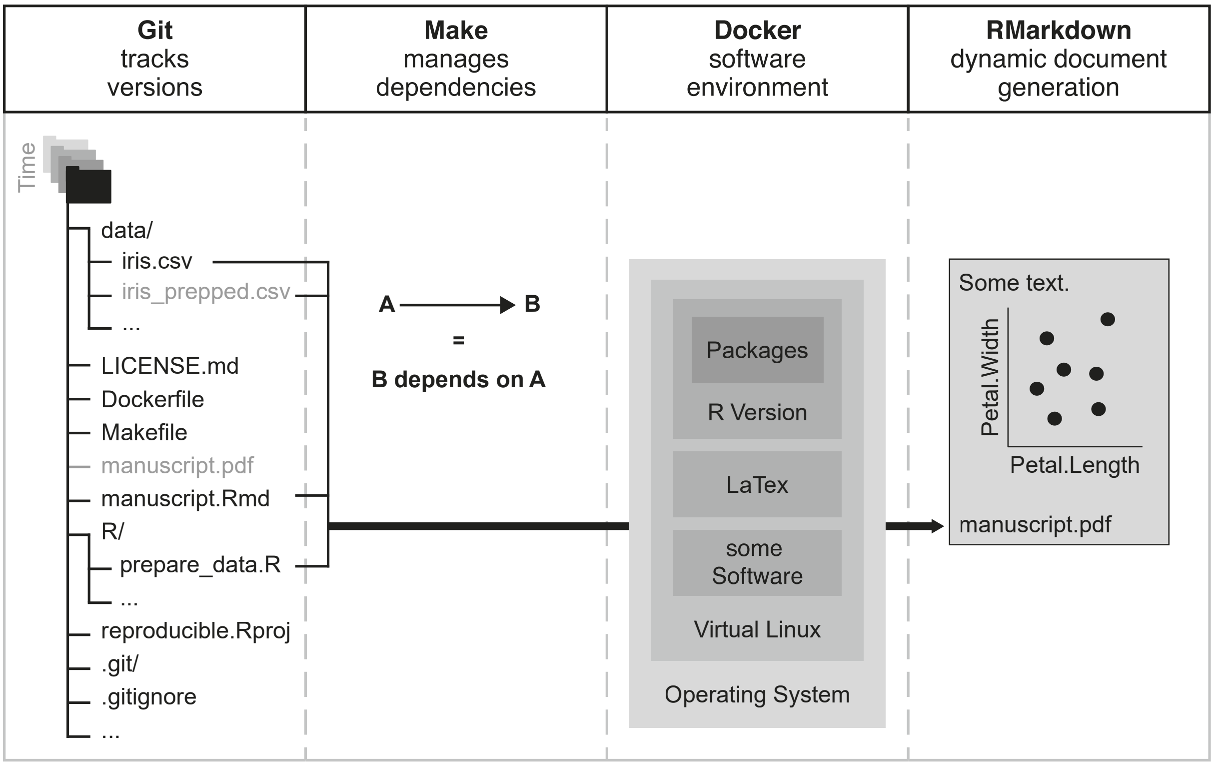
Components of the Reproducible Workflow

The Reproducible Workflow in a Nutshell

Figure 1 gives an overview of how the four components of our workflow interact to ensure computational reproducibility. Before we describe the four components in more detail, we begin with a minimal description of the roles of each component. In the remainder of this tutorial, we will further detail each of the four components of our workflow.

Figure 1

Schematic Illustration of the Interplay of the Four Components Central to the Reproducible Workflow



Note. Git tracks changes to the project over time; Make manages dependencies among the files; Docker provides a container, in which the final report is built using dynamic document generation in R Markdown. Git = Version Control; Make = Dependency Management; Docker = Containerization; R Markdown = Dynamic Document Generation

The first component is version control. Version control manages changes to files (e.g., data and code) over time so that you can recall specific versions of files later or revert the entire project to a past state. Version control offers snapshots of your workflow at different time points identified by a unique identifier. How different parts of an analysis and a corresponding report relate to each other and in what order they need to be executed is documented using dependency management. The arrows in Figure 1 visualize dependencies, such as an analysis depending on the availability of a particular data file. Third, all computer code (such as a statistical analysis in R) is executed in a virtual environment that guarantees exact reproduction of results independent of the host operating system, the locally installed R version, and installed package versions. Finally, dynamic document generation (also known as the literate programming paradigm) interweaves human-readable code and computed results (such as point estimates, *p* values, or confidence intervals) to eliminate inconsistency errors such as those arising from copy-and-paste errors.

Dynamic Document Generation

The translation of computational results into a human-readable summary, for example into a technical report, a presentation, or a manuscript, is time-consuming and error-prone. Typical errors result from copy-and-paste mistakes, erroneous rounding, or missed updates of the manuscript when the associated computer code and computed results have changed. In order to create not only fully reproducible results but also fully reproducible reports, we resort to the literate programming paradigm (Knuth, 1984), in which human-readable language and computer code are mixed to create dynamic documents whose order follows the logic of thought rather than the order of the computer. R Markdown is a simple markup language to create dynamic documents with embedded chunks of R code that can be exported to standard formats such as documents (docx, pdf, rtf, epub), presentations (ppt, html) or websites (html) using the **knitr** package (Xie, 2015, 2019). Several packages extend the functionality of **knitr**. Of particular note are the **papaja** package (Aust & Barth, 2018), which offers additional functions to enable American Psychological Association (APA) style document formatting, including a journal-style final typeset format, and the **stargazer** package (Hlavac, 2018), which provides journal-ready tables and reports of statistical models. Figure 2 illustrates R Markdown syntax using the **papaja** package and Figure 3 shows the resulting rendered document.

Figure 2

Exemplary Excerpt of an R Markdown File

```
### Dynamic Document Demonstration

```{r setup, echo=FALSE}
library("knitr")
library("papaja")
```

This is a simple analysis of the `sleep` dataset (Student, 1908) taken from
`help(t.test)`.

```{r t-test}
data("sleep")
result <- t.test(extra ~ group, data = sleep, paired = TRUE)
```

The difference in means of hours slept between the groups
was `r ifelse(result$p.value >= .05, "**not**", "")`
significantly different from zero (`r apa_print.htest(result)$full_result`).
```

Note. This excerpt of an R Markdown file shows a combination of executable R code, which will be dynamically rendered to content on document creation, and English manuscript text. Code is either given in separate chunks (shown in grey background delimited by triple backticks) or inline (single backticks). The resulting document is shown in Figure 3.

Figure 3

Rendered Result of the Source Code Shown in Figure 2

Dynamic Document Demonstration

This is a simple analysis of the `sleep` dataset (Student, 1908) taken from `help(t.test)`.

```
data("sleep")
result <- t.test(extra ~ group, data = sleep, paired = TRUE)
```

The difference in means of hours slept between the groups was significantly different from zero ($M_d = -1.58$, 95% CI $[-2.46, -0.70]$, $t(9) = -4.06$, $p = .003$).

PsychOpen GOLD

Version Control

Fundamentally, reproducibility means that computational results remain identical if neither the script nor the data have changed. It is often not trivial to find out whether any element in a project has changed over time and if so, to "go back in time." The Git program enables you to do both. A good mental model for Git is that it takes a sequence of snapshots of all files it is supposed to track. In the language of Git, these snapshots are "commits." A commit represents a complete copy of the state of all tracked files. Each commit has a short, unique identifier (a hash code) and a human-readable description (commit message). Going back to one state is as easy as traversing the history of all commits and switching the repository to a given previous state; it is possible to visually compare changes between different versions. The collection of all snapshots is called a "repository," which ideally tracks your entire R project.

A typical Git workflow in the terminal looks like this:

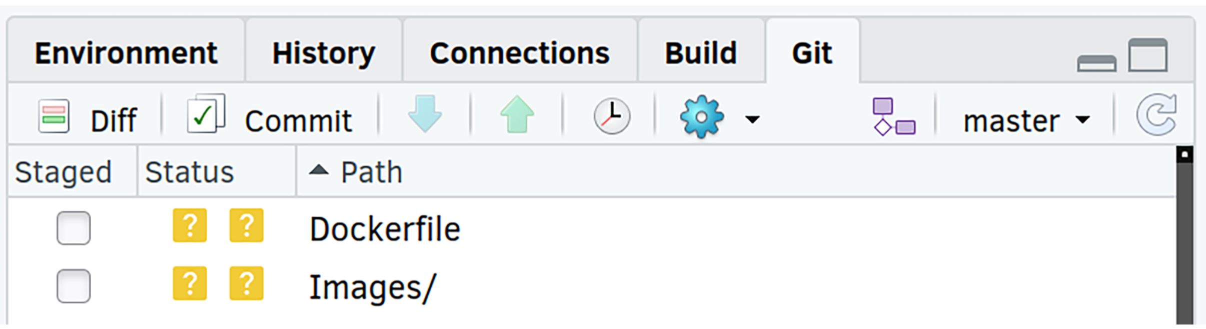
```
# -- type this on the command line --

git init # to initialize Git in the current directory
git add ./data/iris.csv ./R/analysis.R # track specific files
git commit -m "added data and analysis" # take snapshot with comment
# once script or data were changed, take a new snapshot
git commit -a -m "completed data collection" # add and commit all changes
```

To keep track of all changes on your local computer, you only need to use `git add` and `git commit` or `git commit -a` to add and commit at the same time. Adding a file means to save its changes on the next commit. These commands need to be executed in the terminal, which you can access from within RStudio (`Shift + Alt + R`). RStudio also offers a graphical user interface for Git. For most basic operations, this interface is convenient and sufficient (see Figure 4).

Figure 4

Git Pane Providing Easy Access to Basic Functions in RStudio



In a given Git project, you can inspect all changes (`git log`) and examine any previous state by stating the identifier of the commit to `git checkout`:

```
# -- type this on the command line --

# inspect all changes
git log
# revert local directory to previous version with hash '77db06f78e'
git checkout 77db06f78e
```

Git also makes it particularly easy to share and collaborate on a project with other researchers. A popular service for sharing materials via Git is GitHub. Alternatively, institutions can host an equally feature-rich open-source service called GitLab, avoiding the reliance on commercial service providers. At the time of writing, sharing repositories on GitHub with the public is free, private repositories (only visible to persons you invite) are free for researchers or have limited features. After creating a user account, one can create a new repository and GitHub provides information on how to upload your repository from the terminal, for example, for our repository (here with user name "aaronpeikert" and repository name "reproducible-research"):

```
# -- type this on the command line --

# link remote github repository to local directory
git remote add origin https://github.com/aaronpeikert/reproducible-research.git
# push all changes from local repository to the remote repository
git push -u origin master
```

`git push` or the green upward arrow in the Git pane (see Figure 4) uploads local updates. To download the remote Git repository on another computer, type into the terminal:

```
# -- type this on the command line --

git clone https://github.com/aaronpeikert/reproducible-research.git
```

Git and GitHub can do even more to support you when collaborating with fellow researchers, for example, by providing a web interface to track issues and their status (open/closed/resolved) and further means to manage and merge multiple, parallel versions of code (such as branches, pull requests, or merges), but this is beyond the scope of this tutorial. In particular, GitHub's issue management can be leveraged as a post-publication platform to discuss manuscripts and their results (to comment on our paper, please add an issue to the GitHub repository of our paper, see Supplementary Materials). Another benefit of using Git and GitHub is that experimentation is highly encouraged since you can go back to any state quickly. Even when you lose access to the file on your computer, everything can be backed up on a remote Git server (like GitHub or GitLab).

Further, one can reduce the likelihood of dead code accumulating (e.g., lines that have been commented out) because it is safe to simply remove unneeded code blocks and track their removal in Git.

GitHub allows you to archive and label a specific version of your repository in the form of a release. A release tags a particular commit with an arbitrary label, for example, as "submission,"[1] "preprint," or "published,"[2] and archives also "binary" products of your code, for example, the resulting pdf of the manuscript or the docker image (see Section "Containerization"). From such a release, zenodo.org or figshare.com can create a DOI, making it easier to reference and retrieve it (see the GitHub Guide[3]).

Dependency Tracking and Management

Even when you have obtained a given version of a project with the aim to reproduce reported results, and you can confirm that this version is unchanged, you may not know exactly how to reproduce the results because it may be unclear which scripts or commands must be executed in which order. This is particularly the case when complex preprocessing pipelines are part of the computation or there are dependencies on external programs. Handling such dependencies is easy with Make because it allows you to manage dependencies by creating (computational) recipes to create or recreate files.

Fundamentally, a `Makefile` is a list of recipes. Each recipe has a target (the name of the recipe) followed by a colon, a list of dependent targets or files, and finally a list of system commands to create the target. This is similar to a cooking recipe where the name of the dish appears first, then the required ingredients and finally the steps to follow to prepare the dish. If any of the dependencies have changed since the last time the target was built, the recipe's commands are executed to recreate the target file. We illustrate the use of Makefiles with an example. Assume the final product is a manuscript (`manuscript.pdf`). This manuscript is written in R Markdown (`manuscript.Rmd`) and includes dynamically generated plots from a raw data file (`data/iris.csv`) that needs to be preprocessed first using a separate script (`R/prepare_data.R`) into a prepared data file (`iris_prepped.csv`). You find a graphical representation of this example in Figure 1. A `Makefile` for these dependencies may look like this:

1) We created an release for the submission: https://github.com/aaronpeikert/reproducible-research/releases/tag/v0.1.1-submission

2) We created a release for the final version: https://github.com/aaronpeikert/reproducible-research/releases/latest

3) Retrieved from https://guides.github.com/activities/citable-code/

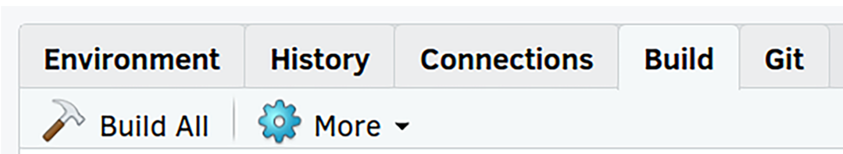
```
# -- this is a Makefile --
# indent by tabs, not spaces
all: manuscript.pdf

manuscript.pdf: data/iris_prepped.csv manuscript.Rmd
  Rscript -e 'rmarkdown::render("manuscript.Rmd")'
data/iris_prepped.csv: R/prepare_data.R data/iris.csv
  Rscript -e 'source("R/prepare_data.R")'
```

The first line after the comment is the first (default) target called "all," which depends on manuscript.pdf, which itself is a target. If Make is called without an argument, the first target is built. To create `manuscript.pdf` (the second target in the file), the file `manuscript.Rmd` needs to be rendered, which depends on `data/iris_prepped.csv`. This dependency is itself a target (the third target in the file). To create `data/iris_prepped.csv, R/prepare_data.R and data/iris.csv` are needed. If you type `make manuscript.pdf`, Make first checks whether the dependencies do exist and, if not, creates them. Here, if `data/iris_prepped.csv` does not exist, Make creates it by executing the third target (running the preprocessing script `R/prepare_data.R`). Also, if one of the dependencies of a target is newer than the target itself, Make updates everything that directly or indirectly depends on the target. Here, if the original data (`data/iris.csv`) is newer than the preprocessed data (`data/iris_prepped.csv`) and thus was possibly modified since the preprocessing was done the last time, Make will attempt to recreate `data/iris_prepped.csv` first before recreating `manuscript.pdf`. If there is a dependency missing, and there is no target to make it, Make stops with an error message. This way Make ensures that all four dependencies of the manuscript (the raw data `data/iris.csv`, the data preparation script `R/prepare_data.R`, the prepared data `data/iris_prepped.csv` and the manuscript source file `manuscript.Rmd`) are correctly resolved. It is a convention to have the first target named `all`, which creates the entire project. Subsequently, the command `make` without any argument automatically creates everything possible in the project. The button `Build All` from within RStudio triggers this process (see Figure 5).

Figure 5

Build Pane in RStudio With Access to Makefile Target "All"



PsychOpen GOLD

If you have followed our workflow as presented thus far, you are (almost) only three commands away from fully reproducing the authors' version of our paper. You simply have to type the following commands on the command line:

```
# -- type this on the command line --

# (1) obtain a local copy of the remote repository
git clone https://github.com/aaronpeikert/reproducible-research.git
# (2) enter the project directory
cd reproducible-research
# (3) run the analysis/data preparation etc. with the local R installation
make all
```

However, if you execute the above on your system, there is a good chance that you cannot reproduce our manuscript and the `make all` command results in an error. Successful reproducibility relies on the crucial assumption that your computational environment is identical or sufficiently compatible to the original one, that is, all required software dependencies need to be installed (e.g., R and all additional R Packages) and no updates or other changes to the computational environment must break or alter the original analysis. As we will shortly see, ensuring full computational reproducibility requires one further level of documentation, that is, documentation and reproduction of the computational environment.

Containerization

Docker is a tool that allows encapsulation, sharing, and re-creation of a computational environment on most operating systems (Windows, macOS, & Linux). Docker achieves these goals by setting up a virtual computer, on which it can execute commands (e.g., installing software). It then saves the resulting state of the virtual computer in what is called an "image." This image can be started and execute commands on the virtual computer, for example, running `Rscript` or `make`. A running instance of an image is called a container. An image can be transferred and executed on any machine that has Docker installed. Regardless of the machine that is executing the container, the computational environment is identical for the programs running inside the container. The most important advantage over traditional virtual machines is that containers are lightweight: they start rapidly, run with little overhead, and do not need much storage space. Docker achieves this by reusing large parts of the host's operating system.

With the following example, we demonstrate the importance of documenting and (re-)storing the computational environment. Generally, with containers, we would like to safeguard against changes to the computational environment resulting in unexpected consequences, for example, changes in the functionality or default options in packages or even in the R environment itself. While the R programming language is considered stable and much effort is put into backward compatibility, even basic functions like `read.csv()` (to load data) or `sample()` (to randomly sample from a set) sometimes change their behaviour from one version to another. For example, to ensure reproducibility of analyses based on a computer's pseudo-random number generator (PRNG), it is good practice to rely on fixed PRNG seeds, which are numeric values that set the PRNG into a deterministic state, that is, the sequence of pseudo-random numbers reproduces exactly. Consider the following R code to randomly draw five numbers between 1 and 10:

```
# -- R code --
set.seed(1234)
sample(1:10, 5)
```

The usual expectation is that this code delivers the same pseudo-random five numbers regardless of the operating system or R Version (because of `set.seed()`). Using Docker, we can start an image which contains the R (Version 3.5.0), and execute the code there.

```
R.version$version.string
set.seed(1234)
sample(1:10, 5)
```

This outputs:

```
## [1] "R version 3.5.0 (2018-04-23)"

## [1] 2 6 5 8 9
```

When executing the code in an image with a more recent version of R (Version 3.6.1), the function returns a different sample despite the identical random seed:

```
R.version$version.string
set.seed(1234)
sample(1:10, 5)
```

This outputs:

```
## [1] "R version 3.6.1 (2019-07-05)"

## [1] 10 6 5 4 1
```

Note, that this is intended behaviour as it is the result of a bugfix in the random number generator implemented as of R (Version 3.6.0). Now, such changes may strictly render analyses run on previous R versions not reproducible if they contain, for example, multiple imputations, bootstrapping, simulations studies, graphics with random jitter, Bayesian estimations using sampling algorithms (such as Markov Chain Monte Carlo), or similar techniques that involve random sampling. We would like to illustrate this with a more concrete example (the full R code to reproduce this non-reproducibility is provided in the GitHub repository of this manuscript). We ran a linear regression model on a simulated dataset with two variables `x` and `y` with R's `lm()` function regressing `x` on `y`. Using the **boot** package (Canty & Ripley, 2019), we bootstrapped the 95% confidence intervals around the regression coefficient estimate with 1,000 bootstrap samples to evaluate whether the estimated confidence interval included zero. To make the analysis reproducible, we set a random seed. We ran this once in R (Version 3.5.0):

```
R.version$version.string
set.seed(seed)
results <- boot(data=simdata, statistic=bs,
                      R=1000, formula=y~1+x)

# get beta estimates' confidence intervals
round(confint(results, type = "bca", parm = 2), 4) # parm = 2 -> b
```

```
## [1] "R version 3.5.0 (2018-04-23)"

## 2.5 % 97.5 %
## 0.0097 0.3842
```

Subsequently, we ran the identical script with the identical seed in R (Version 3.6.1):

```
R.version$version.string
set.seed(seed)
results <- boot(data=simdata, statistic=bs,
                      R=1000, formula=y~1+x)

# get beta estimates' confidence intervals
round(confint(results, type = "bca", parm = 2), 4) # parm = 2 -> b
```

```
## [1] "R version 3.6.1 (2019-07-05)"

## 2.5 % 97.5 %
## -0.0005 0.3748
```

As we see from these R outputs, the latter of the estimated confidence intervals does include zero while the former does not. Please note that one could discuss deeper issues with null hypothesis significance testing here, but with this example, we would simply like to stress that computational reproducibility in the strict sense requires capturing the full computational environment.

Only rarely does an analysis depend on base R only. Typically, a considerable number of packages is required that each may depend on multiple other packages. Each update of each package in this dependency hierarchy and updates to base R itself will increase the likelihood of breaking reproducibility (the resulting frustration is sometimes referred to as *dependency hell*). The whole endeavour of reproducibility is therefore at stake every time an update is rolled out. To ensure long-term reproducibility, our workflow replicates the original computational environment of an analysis exactly. Note, that we do not intend to advocate that software should not be updated; updates typically promote bugfixes and provide new functionality; our point is that full computational reproducibility is only achieved if the software versions used originally are precisely documented. Among other things, this makes it possible to trace back update histories to discover which change in which package caused the non-reproducibility. Quite to the contrary, with containerization, it gets easier than ever to safely update to new versions just by changing the R version number of the Docker image (and reverting back if this update breaks code). This convenience is possible because of the efforts of the Rocker project (Boettiger & Eddelbuettel, 2017), which provides Docker images pre-configured with an installation of selected R versions. These packages are taken from MRAN (Revolution Analytics, 2019), a repository for R packages fixed to the last date on which the R version of the image was the most recent. Building upon these Rocker images, researchers can easily build their own Docker images with all required R packages. The rocker project also provides images that include RStudio (`rocker/rstudio`), the **tidyverse** package (`rocker/tidyverse`) and the **R Markdown** package with LaTeX (`rocker/verse`). Because our workflow relies on R Markdown, we suggest using the `rocker/verse` image (which also contains `rstudio` and `tidyverse`). These images are stored on Dockerhub (https://hub.docker.com/).

Building on a basic Rocker image, we can specify further software dependencies in a Dockerfile. For example, the basis for this manuscript's Docker image is the following Dockerfile:

```
# -- this is a Dockerfile --

# Define the R version to be installed from rocker project
FROM rocker/verse:3.6.1
# install CRAN R packages: pacman, here, and pander
RUN install2.r --error --skipinstalled\
  pacman here pander
# install additional R packages from github: papaja and wordcountaddin
# the package version fixed by hash (user/package@hash)
RUN installGithub.r\
  crsh/papaja@b6cd70f benmarwick/wordcountaddin@fdf70d9
# set the working directory inside the container
WORKDIR /home/rstudio
```

The `FROM` statement specifies which Docker image to use, in this case, the `rocker/verse` image with the tag 3.6.1 (referring to the R Version 3.6.1). The `RUN` statement describes a command to execute, in this case, to run an R script `install2.r` which is available on all Rocker images, to install the specified packages (here, **pacman**,

here and **pander**). A Dockerfile allows more than one `RUN` statement, executing arbitrary system commands. Those `RUN` statements can install dependencies that are not an R package, for example, other programming languages like python or Matlab. The `WORKDIR` statement is not strictly necessary but simplifies commands. The command `docker build -t image-name` creates an image named `image-name` from the Dockerfile in the project. A way to identify the dependencies automatically and generate a docker image from them is provided in the liftR package (Xiao, 2019).

The flexibility to fully control the software environment is of particular interest for software infrastructures where users cannot install software because of limited access rights, for example, on cloud computing platforms or high-performance computing clusters. However, Docker needs unrestricted access rights to the system, which are rarely granted on high-performance computing clusters. For this case, Singularity provides a fully compatible alternative (see Section “Linux”) that can be executed with limited access rights.

There are two ways to share a Docker image; either by sharing the Dockerfile that creates the image or by sharing the image itself, for example, through a service like Dockerhub. While both ways guarantee a replicable computational environment, sharing the Dockerfile is more transparent and more space-saving; in our workflow, we can use Git to track changes in the Dockerfile (such as updates to dependencies). A possible downside is that in order to create an image from a Dockerfile, all software repositories need to be still available. Hence, to guarantee long term reproducibility, it is best to archive the complete binary image at major points of the projects’ progress, for example, on publication (ideally, using a release tag; see Section “Version Control” for details).

There are two options to execute commands in a container. Both options are based on the `docker run` command. The first way is to run a command inside the container. The call takes the form:

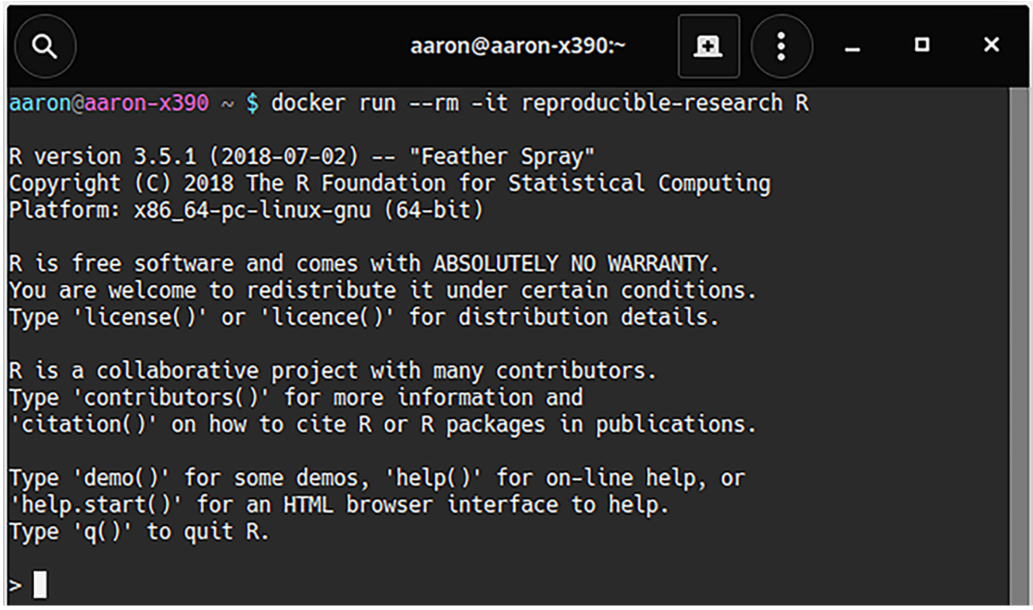
```
# -- type this on the command line --

# execute a command in a container image; do not save the state
# of the container; accept inputs from and return outputs to terminal
docker run --rm -it <IMAGENAME> <COMMAND>
```

The `--rm` flag means that the state of the container after the command will have finished is not going to be saved. The `-it` flag tells Docker to run the command interactively, that is, to accept keyboard inputs and return outputs to the terminal. For example, this is the command to start an interactive R session inside a Docker image called `reproducible-research` (see Figure 6 for a screenshot):

Figure 6

R Terminal Running Inside Docker




```
# -- type this on the command line --

# start an interactive R session in the
# container named 'reproducible-research'
docker run --rm -it reproducible-research R
```

The second option is to start the container in the background and to interact with the container via the web browser and the RStudio server instance running in it. In order to do so, you need to supply a password to log into the RStudio server (`-e PASSWORD=<YOUR_PASS>`) and open a local network service on a specified port (`-p 127.0.0.1:8787:8787`).

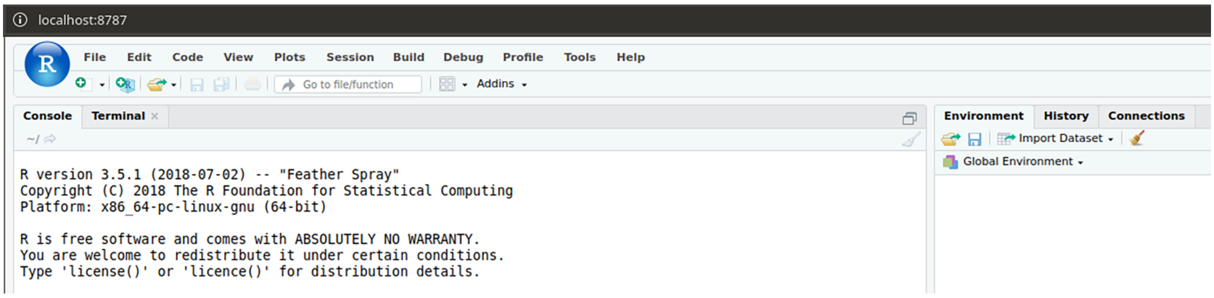
```
docker run -e PASSWORD=<YOUR_PASS> -p 127.0.0.1:8787:8787 image-name
```

The address to connect to the RStudio server is your IP address (or `localhost` on Linux) in this scheme: `<IPADDRESS>:8787`. This offers a fully functioning RStudio instance that runs in the image but is accessible through a local web browser.

Figure 7 shows a screenshot of Rstudio running inside Docker accessed from a local web browser.

Figure 7

RStudio Running Inside Docker



By default, programs inside the container cannot access files on the local computer, thus requiring an explicit link to a local folder to enable access (and on macOS and Windows this also has to be allowed in the settings):

```
docker run -v /folder/on/your/computer:/folder/in/docker
```

The main directory for RStudio inside the container is `/home/rstudio`, so the complete call to start RStudio inside a Docker container may look like this in the local terminal:

```
# start docker in the background, open a local web service with a virtual
# Rstudio instance and enable access to selected local directories
docker run --rm -it -e PASSWORD=<YOUR_PASS> -p 8787:8787 -v
/path/to/project:/home/rstudio reproducible-research
```

Figure 7 shows a screenshot of the result.

Since Docker commands tend to grow long and become tedious to type manually, we recommend using some automatic way to generate them. Fortunately, one can use Make to automatically generate the `docker` commands, for example, the (simplified) `Makefile` for this paper allows the command after `$(run)` to be conditionally passed through Docker if one types `make DOCKER=TRUE` (otherwise, they are run locally):

```
# -- this is a Makefile --
# indent by tabs, not spaces

# set local variables for later use
project := $(notdir $(CURDIR))
current_dir := $(CURDIR)
home_dir := $(current_dir)
uid = --user $(shell id -u)

# determine if DOCKER=TRUE was given
# if so, run everything in docker
# if not, run everything locally
```

```
ifeq ($(DOCKER),TRUE)
  run:=docker run --rm --user $(uid) -v $(home_dir):/home/rstudio $(project)
  current_dir=/home/rstudio
endif

# default target is target 'manuscript.pdf'
all: manuscript.pdf

# build the docker container
build: Dockerfile
  docker build -t $(project) .

# build manuscript.pdf from Rmd file
# run in docker if DOCKER=TRUE else locally
manuscript.pdf: manuscript.Rmd reproducible-research.bib
  $(run) Rscript -e 'rmarkdown::render("$(current_dir)/$<")'
```

Installing and Setting Up the Workflow

Other than on R, RStudio, and R Markdown, our workflow relies on three pieces of software from outside the R environment: Git, Make, and Docker. The smoothness of the installation process of these software packages varies across operating systems. For example, on macOS, Make is always available, whereas Linux systems are typically shipped with both Git and Make. In the following section, we share what we consider the easiest way to install those packages across common operating systems. However, installation processes may be subject to change, and we advise readers to also consult the documentations of the packages or see our collection of links to tutorials and installation instructions on our GitHub repository.

Windows

Windows systems typically require the biggest efforts to install all necessary pieces of software. Note, that you must have either Windows Pro, Enterprise, Education, or Server installed, as Microsoft prevents the use of Docker on Windows Home (see Section "Related Approaches" for alternatives to Docker in case you cannot avoid Windows Home). There is a package manager for Windows called Chocolatey, which you can install from: https://chocolatey.org/install. Chocolatey provides all software packages needed for our workflow in one place. Having installed Chocolatey (and restarted the computer), all dependencies can be installed in an **admin terminal** (Windows key, then type `cmd`, right-click *Run as administrator*) via:

```
# -- type this on the command line --

# install Docker, Make, and Git using Chocolatey
choco install -y git make docker-desktop
```

To use `docker` you need to start Docker Desktop. In the settings of Docker Desktop, you have to allow the sharing of your drive. Docker on Windows requires an unusual path (e.g., `C:\Users\aaron\Documents\reproducible-research` becomes `/c/Users/aaron/Documents/reproducible-research`). Therefore, you currently need to hand-edit the `Makefile` and set `current_path` to the project directory and use `make all DOCKER=TRUE WINDOWS=TRUE`. We hope that future releases of Docker for Windows will not require that workaround.

macOS

As Make already ships with macOS, you only need Git and Docker. We suggest using the package manager Homebrew, which you can install from https://docs.brew.sh/Installation, to install Docker (Git will be installed during the installation of Homebrew):

```
# -- type this on the command line --

# install Docker via Homebrew
brew cask install docker
```

To use `docker`, you need to start Docker Desktop. In the settings of Docker you have to allow the sharing of your drive.

Linux

There is a host of different Linux distributions and almost as many package managers. Still, to our knowledge, there is no (recent) Linux edition, that does not include Git, Make and Docker. For example, in Ubuntu Linux, installation is straightforward using the shipped package manager:

```
# -- type this on the command line --

# install Docker via advanced package tool
apt install git make docker
```

For other distributions, replace `apt install` with your package manager's equivalent. You may need elevated rights for the installation; in this case, add `sudo` before the installation command. `docker` also needs elevated rights to run; therefore, we recommend adding the local user to the `docker` group, following the documentation of Docker.

An alternative to Docker on Linux is Singularity (Kurtzer, Sochat, & Bauer, 2017). To use it, just replace any docker calls with `singularity docker` because Singularity fully supports docker images. A possible advantage is that Singularity works well in high-performance computing environments and on old Linux versions, the downside is that Singularity is currently only available on Linux.

Project Organization

Finally, we conclude with some notes on project organization, which we think makes migrating projects to a reproducible workflow easier. The first step towards reproducibility is to create an R script or R Markdown file as the primary entry point for the analysis that runs on a local computer without error and performs the main statistical analyses. Next, one needs to make sure that all files relevant to the analysis can be moved to another computer. To this end, it is recommended that all files reside within one folder (or enclosed subfolders within it) and all paths are relative to that folder because absolute paths are specific to a given computer. A robust solution to the problem of making sure that file access does not break across computing platforms are RStudio projects and the **here** package (Müller, 2017) to manage file access. The **here** package solves two common issues with relative paths. First, it takes care of the fact that path separator characters vary across operating systems (typically, slash or backslash). Second, it solves the issue that anchor points of relative paths may differ depending on context. For example, knitr interprets paths relative to the dynamic document, whereas R has a current working directory that may change over the course of an R session. The **here** package provides consistent paths relative to the project directory. The following three examples refer to local files ranging from absolute paths with system-specific path separators (bad) to relative paths using the **here** package:

```
# -- R code --

# BAD because the path is specific to the computer/user
iris <- read.csv("/home/aaron/reproducible-research/data/iris.csv")
# GOOD because it is a relative path, but slash depends on OS
iris <- read.csv("data/iris.csv")
# BETTER because truly compatible across OS
iris <- read.csv(here("data", "iris.csv"))
```

The folder where all the files reside that you need for an analysis (code and data), is referred to as a "project" (or sometimes as a "research compendium"). Working with projects is particularly convenient with RStudio. It is useful to organize a data analysis project in a way that strictly segregates (raw) data and code by placing them in directories called `data` and `R` (see Section 4 in Marwick, Boettiger, & Mullen, 2018); there are also tools that automatize the standardized creation of folder structures such as workflowr (Blischak, Carbonetto, & Stephens, 2019).

Sometimes external requirements make it impossible for the data to be stored and shared with the scripts. In most of the cases we have seen, these are either space constraints or privacy considerations. In these cases, unrestricted reproducibility is not guaranteed. If splitting data and scripts is unavoidable, we recommend validating all data files using checksums (also called a "hash," e.g., using the functions provided in package **digest**; Eddelbuettel et al., 2019) before analyzing them. A checksum is a short fixed-length fingerprint (often displayed in the hexadecimal system) of a file with the purpose of verifying the integrity of a digital object. Fingerprints are computed from digital objects such that they change with high probability if data is changed only a little. To use checksum validation, checksums for all data files must be created and stored at the time of the original analysis. At the time of reproduction, the current checksum must be compared with the stored checksum to ensure data integrity.

```
# -- R code --

# create a dummy data.frame with two columns
x <- data.frame(VAR1=c(1,2,3,4),VAR2=c(0,4,6,9) )
# compute checksum using md5
checksum <- digest::digest(x, "md5")
if (checksum != "5ba412f5a26f43842971dd74954fcdeb"){
  warning("Mismatch between original and current data file!")
}
```

Use Case: Reproducing an Analysis

We provide a reproducible analysis as a working example via GitHub. We encourage interested readers to try to reproduce this example as a practical exercise. The example shows a minimalistic analysis of the Considerations of Future Consequences (CFC) Scale. The analysis demonstrates a complete implementation of our workflow including downloads of external data, comparison of their integrity using a checksum, and a confirmatory factor analysis on the first few items using the R package **lavaan** (Rosseel, 2012). Once all required tools are installed on a computer, the following four command-line commands are sufficient to reproduce our demo analysis:

```
# -- type this on the command line --

# (1) obtain a local copy of the remote repository
git clone https://github.com/aaronpeikert/workflow-showcase.git
# (2) enter the project directory
cd workflow-showcase
# (3) build the docker container
make build
# (4) run the analysis and produce the final PDF inside the container
make all DOCKER=TRUE
```

Summary

The overarching goal of this paper was to provide a complete workflow that allows confidence in the reproducibility of R-based data analyses. Analyses following our workflow can be reproduced with four commands (here shown for this manuscript):

```
# -- type this on the command line --

# (1) obtain a local copy of the remote repository
git clone https://github.com/aaronpeikert/reproducible-research.git
# (2) enter the project directory
cd reproducible-research
# (3) build the docker container
make build
# (4) run the analysis and produce the final PDF inside the container
make all DOCKER=TRUE
```

The workflow enables the reproduction of a scientific report exactly without regard to the local operating system, locally installed software, time, or interim changes to the project files. To that end, the proposed workflow relies on tools that have been the foundation of reliable software development for years or even decades. As a by-product, it makes transparent how statistical results depend on the software that created them and, by virtue of this transparency, facilitates later reuse by other researchers.

Each tool in the workflow reduces the chances of non-reproducibility. Dynamic reporting with R Markdown guarantees consistency between computational results and their reporting; version control with Git ensures permanence and consistency across multiple versions of data and code; dependency management with Make provides defined entry-points while mapping out dependencies between all components of a project; containerization with Docker guarantees full computational reproducibility. We believe that the proposed combination of tools does not limit researchers but enables them to operate on a solid basis to deliver transparent and sustainable research.

Related Approaches

While our approach was designed to scale well with the complexity of a computationally intense project, we realize that this flexibility may not be straightforward to integrate into researchers' everyday workflow. There are various R packages that implement parts of our workflow and, thus, lower the threshold for adoption when the full flexibility provided by our workflow is not needed. The use of R Markdown within a project, tracked with Git can be simplified with the workflowR package (Blischak et al., 2019). The drake package (Landau, 2018) is directly inspired by Make and takes an R-centric approach, making it especially suited for projects only involving R, but it can also handle external dependencies. The liftR package (Xiao, 2019) and the holepunch package (Ram, 2019) automatize the use of Docker. The former is perfectly compatible with the described workflow, and we recommend it to users who are not comfortable with command-line use of Docker. **holepunch** uses binder (Jupyter et al., 2018) to move the analysis to the cloud, so that no local installation of Docker is required. **holepunch** is well suited for simple analyses with low computational demands because binder's memory and computing time is limited. There are several alternatives to Docker that manage dependencies on R packages. renv (Ushey, 2020) is a way to freeze package version via local copies of packages in the project, but it does not guarantee a given base R version or system dependencies beyond R. Similar approaches are taken by jetpack (Kane, 2019), miniCRAN (de Vries, 2019) and checkpoint (Microsoft Corporation, 2019). The package reprex (reproducible example, Bryan, Hester, Robinson, & Wickham, 2019) is also worth noting, but its scope is limited. A particularly noteworthy approach is the **worcs** package (van Lissa, Brinkman, et al., 2020; van Lissa, Peikert et al., 2020), which is an R project template that creates a standardized file structure for code and data supporting version management with Git, package management with **renv** and dynamic document generation with R Markdown. We acknowledge that worcs is much easier to install and provides a one-click solution for the creation of reproducible projects. It achieves a high standard of reproducibility but does not guarantee full computational reproducibility and is limited to dependency management within the R environment.

Other than these tools, which ease the process of creating workflows like ours does, we have noticed an increased interest in changing the way research is published and used (Perkel, 2018), with the emergence of *life code* (Perkel, 2019) and *continuous integration* (Beaulieu-Jones & Greene, 2017; Yenni et al., 2018). These techniques give us a glimpse of a paradigm shift from static to dynamic, interactive, and living publications that is yet to happen.

Limitations

We are aware that implementing the proposed workflow is not straightforward, and the difficulty of its implementation may vary by platform. For example, the installation of all tools is already easier on POSIX-compatible platforms such as Unix, Linux, or macOS (but not Windows). However, once a reproducible workflow is established as a default, it can be used with minimal changes for every R project.

In our own experience, it is often not possible to convince all co-authors to switch to a different document processing environment, such as R Markdown. That is, we have experienced the case that after writing up the first draft in R Markdown, we eventually had to generate a Word file that, from then on, was used as static file serving as a basis for multiple iterations among the co-authors. Retaining reproducibility in such situations requires tedious manual synchronization of files across formats. This annoyance may be reduced with the redoc package (Ross, 2019), which enables a bidirectional synchronization between Word and R Markdown. Conversions between R Markdown and Word retain all changes and support Word's track-changes feature. Hence, R Markdown users can share a Word file with their collaborators, receive their changes in this file and transform it back to R Markdown.

Sharing Reproducible Workflows

How can one best share a reproducible workflow? We believe that, ideally, a non-commercial public service provider should be found that guarantees permanent and reliable hosting of reproducible workflows, such as the Open Science Framework (Foster & Deardorff, 2017). An independent provider mirroring and complementing the services offered by GitHub, Docker Hub, and MRAN would be desirable. Second, to ensure that other users are legally able to benefit from the shared materials, authors must choose an appropriate license. Typically, there is no single license that works for code, data, and media (such as text or figures). We encourage authors to choose appropriate license forms that do not hinder others from freely downloading, using, and modifying the shared workflows and materials while, at the same time, ensuring recognition for the time and effort invested in creating the workflow in the first place. In our experience, the Creative Commons—Attribution license (CC-BY) is often appropriate for sharing texts, R Markdown files, generated figures, and other media, whereas scripts and any other computer code are often best shared under the MIT license (or similar permissive licenses). Both licenses assure maximal freedom for future users while requiring the attribution of the original authors in derivative work. These licenses are also in line with the recommendations by the Reproducible Research Standard (Stodden, 2009; Stodden et al., 2016). A great resource to choose a license is choosealicense.com, however, no resource, including our recommendation, replaces legal advice. To facilitate an inclusive environment, we recommend naming all contributors and including a Code of Conduct[4] in your project.

Outlook

The proposed workflow leverages various existing tools that are partly integrated into RStudio already. Parts of the proposed workflow have been integrated into stand-alone packages (such as **worcs**, van Lissa, Peikert et al., 2020; **workflowr**, Blischak et al., 2019; or **holepunch**, Ram, 2019), which we recommend to beginners; in particular, **worcs** is a step-by-step procedure with best practices for Open Science from preregistration to publication. Still those approaches do either not guarantee full computational reproducibility or rely on proprietary service providers. We hope that as awareness of the challenges of computational reproducibility increases, the growing demand for unified and open solutions will lead to better integration of existing tools and services so that reproducible workflows become a standard in psychological research.

4) For example, https://www.contributor-covenant.org/version/2/0/code_of_conduct/

Funding: The authors have no funding to report.

Acknowledgments: We are grateful to Julia Delius for her helpful assistance in language and style editing. We thank Michèle Nuijten and the anonymous reviewer, and all contributors to our GitHub repository for helpful feedback on the paper.

Competing Interests: The authors have declared that no competing interests exist.

Supplementary Materials

This paper is fully reproducible using the workflow described here. All materials for doing that, are provide via the GitHub repository (Peikert & Brandmaier, 2020). This paper is based on the commit identified by hash "c4213f6". For adding issues, providing feedback or any other type of comment or questions regarding the workflow described in the paper please add an issue to the GitHub repository (https://github.com/aaronpeikert/reproducible-research/issues).

Index of Supplementary Materials

Peikert, A., & Brandmaier, A. M. (2020). *A reproducible data analysis workflow with R Markdown, Git, Make, and Docker* [Materials for reproducing the journal paper]. GitHub. https://github.com/aaronpeikert/reproducible-research/

References

Askren, M. K., McAllister-Day, T. K., Koh, N., Mestre, Z., Dines, J. N., Korman, B. A., ... Madhyastha, T. M. (2016). Using make for reproducible and parallel neuroimaging workflow and quality-assurance. *Frontiers in Neuroinformatics, 10*, Article 2. https://doi.org/10.3389/fninf.2016.00002

Aust, F., & Barth, M. (2018). *papaja: Create APA manuscripts with R Markdown* (Version 0.1.0.9842). Retrieved from https://github.com/crsh/papaja

Barba, L. A. (2016). The hard road to reproducibility. *Science, 354*(6308), 142. https://doi.org/10.1126/science.354.6308.142

Beaulieu-Jones, B. K., & Greene, C. S. (2017). Reproducibility of computational workflows is automated using continuous analysis. *Nature Biotechnology, 35*(4), 342-346. https://doi.org/10.1038/nbt.3780

Blischak, J., Carbonetto, P., & Stephens, M. (2019). *workflowr: A framework for reproducible and collaborative data science* (Version 1.4.0.9001). Retrieved from https://github.com/jdblischak/workflowr

Boettiger, C., & Eddelbuettel, D. (2017). An introduction to rocker: Docker containers for R. *The R Journal, 9*(2), 527-536. https://doi.org/10.32614/RJ-2017-065

Bryan, J., Hester, J., Robinson, D., & Wickham, H. (2019). *reprex: Prepare reproducible example code via the clipboard* (Version 0.3.0). Retrieved from https://CRAN.R-project.org/package=reprex

Canty, A., & Ripley, B. D. (2019). *boot: Bootstrap r (s-plus) functions* (Version 1.3-23). Retrieved from https://CRAN.R-project.org/package=boot

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). New York, NY, USA: Apress.

Claerbout, J. F., & Karrenbach, M. (1992). Electronic documents give reproducible research a new meaning. In *SEG technical program expanded abstracts 1992* (pp. 601–604). https://doi.org/10.1190/1.1822162

Clyburne-Sherin, A., Fei, X., & Green, S. A. (in press). Computational reproducibility via containers in social psychology. *Meta-Psychology.* https://doi.org/10.31234/osf.io/mf82t

Deutsche Forschungsgemeinschaft. (2019). *Leitlinien zur Sicherung guter wissenschaftlicher Praxis.* Retrieved from https://www.dfg.de/download/pdf/foerderung/rechtliche_rahmenbedingungen/gute_wissenschaftliche_praxis/kodex_gwp.pdf

de Vries, A. (2019). *miniCRAN: Create a mini version of cran containing only selected packages.* Retrieved from https://CRAN.R-project.org/package=miniCRAN

Eddelbuettel, D., Lucas, A., Tuszynski, J., Bengtsson, H., Urbanek, S., Frasca, M., ... Denney, B. (2019). *Digest: Create compact hash digests of r objects* (Version 0.6.21). Retrieved from https://CRAN.R-project.org/package=digest

Epskamp, S. (2019). Reproducibility and replicability in a fast-paced methodological world. *Advances in Methods and Practices in Psychological Science, 2*(2), 145-155. https://doi.org/10.1177/2515245919847421

Feldman, S. I. (1979). Make — A program for maintaining computer programs. *Software: Practice and Experience, 9*(4), 255-265. https://doi.org/10.1002/spe.4380090402

Foster, E. D., & Deardorff, A. (2017). Open Science Framework (OSF). *Journal of the Medical Library Association, 105*(2), 203-206. https://doi.org/10.5195/jmla.2017.88

Heroux, M. A., Barba, L., Parashar, M., Stodden, V., & Taufer, M. (2018). Toward a compatible reproducibility taxonomy for computational and computing sciences (No. SAND2018-11186). https://doi.org/10.2172/1481626

Hlavac, M. (2018). *stargazer: Well-formatted regression and summary statistics tables* (Version 5.2.2). Bratislava, Slovakia: Central European Labour Studies Institute (CELSI). Retrieved from https://CRAN.R-project.org/package=stargazer

Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., ... Willing, C. (2018). Binder 2.0—Reproducible, interactive, sharable environments for science at scale. In F. Akici, D. Lippa, D. Niederhut, & M. Pacer (Eds.), *Proceedings of the 17th Python in science conference* (pp. 113–120). https://doi.org/10.25080/Majora-4af1f417-011

Kane, A. (2019). *jetpack: A friendly package manager.* Retrieved from https://github.com/ankane/jetpack

Knuth, D. E. (1984). Literate programming. *The Computer Journal, 27*(2), 97-111. https://doi.org/10.1093/comjnl/27.2.97

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PLOS ONE, 12*(5), Article e0177459. https://doi.org/10.1371/journal.pone.0177459

Landau, W. M. (2018). The drake R package: A pipeline toolkit for reproducibility and high-performance computing. *Journal of Open Source Software, 3*(21), 550. https://doi.org/10.21105/joss.00550

Marwick, B., Boettiger, C., & Mullen, L. (2018). Packaging data analytical work reproducibly using R (and friends). *The American Statistician, 72*(1), 80-88. https://doi.org/10.1080/00031305.2017.1375986

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal, 2014*(239). Retrieved from http://dl.acm.org/citation.cfm?id=2600239.2600241

Microsoft Corporation, M. (2019). *checkpoint: Install packages from snapshots on the checkpoint server for reproducibility* (Version 0.4.6). Retrieved from https://CRAN.R-project.org/package=checkpoint

Müller, K. (2017). *here: A simpler way to find your files* (Version 0.1). Retrieved from https://CRAN.R-project.org/package=here

Nosek, B. A., & Bar-Anan, Y. (2012). Scientific utopia: I. Opening scientific communication. *Psychological Inquiry, 23*(3), 217-243. https://doi.org/10.1080/1047840X.2012.692215

Perkel, J. M. (2018). A toolkit for data transparency takes shape. *Nature, 560*, 513-515. https://doi.org/10.1038/d41586-018-05990-5

Perkel, J. M. (2019). Pioneering "live-code" article allows scientists to play with each other's results. *Nature, 567*, 17-18. https://doi.org/10.1038/d41586-019-00724-7

Piccolo, S. R., & Frampton, M. B. (2016). Tools and techniques for computational reproducibility. *GigaScience, 5*(1), Article 30. https://doi.org/10.1186/s13742-016-0135-4

Ram, K. (2019). *holepunch: Configure your R project for "binderhub".* Retrieved from https://github.com/karthik/holepunch

R Core Team. (2020). *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Revolution Analytics. (2019). *Reproducibility: Using fixed CRAN repository snapshots. MRAN. Microsoft R Application Network.* Retrieved from https://mran.microsoft.com/documents/rro/reproducibility

Ross, N. (2019). *redoc: Reversible reproducible documents.* Retrieved from https://github.com/noamross/redoc

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1-36. http://www.jstatsoft.org/v48/i02/

Rule, A., Birmingham, A., Zuniga, C., Altintas, I., Huang, S.-C., Knight, R., … Rose, P. W. (2019). Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLOS Computational Biology, 15*(7), Article e1007007. https://doi.org/10.1371/journal.pcbi.1007007

Stodden, V. (2009). *Enabling reproducible research: Open licensing for scientific innovation* (SSRN Scholarly Paper No. ID 1362040). Rochester, NY, USA: Social Science Research Network. Retrieved from https://papers.ssrn.com/abstract=1362040

Stodden, V., McNutt, M., Bailey, D. H., Deelman, E., Gil, Y., Hanson, B., … Taufer, M. (2016). Enhancing reproducibility for computational methods. *Science, 354*(6317), 1240-1241. https://doi.org/10.1126/science.aah6168

The Turing Way Community, Arnold, B., Bowler, L., Herterich, S. G. P., Higman, R., Krystalli, A., … Whitaker, K. (2019). *The Turing way: A handbook for reproducible data science* (v0.0.4). https://doi.org/10.5281/zenodo.3233986

Ushey, K. (2020). *renv: Project environments.* Retrieved from https://CRAN.R-project.org/package=renv

van Lissa, C. J., Brinkman, L., Vreede, B., Schoot, R. van de, Peikert, A., & Brandmaier, A. M. (2020). *WORCS: A workflow for open reproducible code in science.* https://doi.org/10.17605/OSF.IO/ZCVBS

van Lissa, C. J., Peikert, A., & Brandmaier, A. M. (2020). *worcs: Workflow for open reproducible code in science.* Retrieved from https://github.com/cjvanlissa/worcs

Xiao, N. (2019). *liftr: Containerize R markdown documents for continuous reproducibility.* Retrieved from https://CRAN.R-project.org/package=liftr

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Boca Raton, FL, USA: CRC press. Retrieved from https://yihui.name/knitr/

Xie, Y. (2019). *knitr: A general-purpose package for dynamic report generation in R* (Version 1.22). Retrieved from https://yihui.name/knitr/

Xie, Y., Allaire, J., & Grolemund, G. (2018). *R markdown: The definitive guide.* Boca Raton, FL, USA: CRC press. Retrieved from https://bookdown.org/yihui/rmarkdown

Yenni, G. M., Christensen, E. M., Bledsoe, E. K., Supp, S. R., Diaz, R. M., White, E. P., & Ernest, S. K. M. (2018). Developing a modern data workflow for living data. *bioRxiv,* Article 344804. https://doi.org/10.1101/344804

Reproducible research in R: A tutorial on how to do the same thing more than once

Peikert, A., van Lissa, C. J., & Brandmaier, A. M. (2021). Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once. *Psych*, *3*(4), 836–867. https://doi.org/10.3390/psych3040053

psych

MDPI

Tutorial

Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once

Aaron Peikert [1,2,*], Caspar J. van Lissa [3,4] and Andreas M. Brandmaier [1,5,6]

1. Center for Lifespan Psychology—Max Planck Institute for Human Development, Lentzeallee 94, 14195 Berlin, Germany; brandmaier@mpib-berlin.mpg.de
2. Humboldt-Universität zu Berlin, Unter den Linden 6, 10117 Berlin, Germany
3. Department of Methodology & Statistics—Utrecht University, Faculty of Social and Behavioral Sciences, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands; C.J.vanLissa@uu.nl
4. Open Science Community Utrecht, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
5. Max Planck UCL Centre for Computational Psychiatry and Ageing Research, Lentzeallee 94, 14195 Berlin, Germany, and London WC1B 5EH, UK
6. MSB Medical School Berlin, Rüdesheimer Str. 50, 14197 Berlin, Germany

* Correspondence: peikert@mpib-berlin.mpg.de

Simple Summary: Reproducibility has long been considered integral to the scientific method. An analysis is considered reproducible if an independent person can obtain the same results from the same data. Until recently, detailed descriptions of methods and analyses were the primary instrument for ensuring scientific reproducibility. Technological advancements now enable scientists to achieve a more comprehensive standard that allows anyone to access a digital research repository and reproduce all computational steps from raw data to final report, including all relevant statistical analyses, with a single command. This method has far-reaching implications for scientific archiving, reproducibility and replication, scientific productivity, and the credibility and reliability of scientific knowledge. One obstacle to the widespread use of this method is that the underlying tools are complex and not part of most researchers' basic training. This paper introduces `repro`, an R package that guides researchers through installation and use of the tools required to make a research project reproducible. We also suggest using the proposed workflow for the preregistration of study plans as reproducible computer code (Preregistration as Code; PAC). Since computer code represents the planned analyses exactly as they will be executed, it is more precise than natural language descriptions. PAC circumvents the shortcomings of ambiguous preregistrations that may result in undisclosed use of researcher degrees of freedom. Reproducibility, facilitated by automation, has a wide range of applications and could potentially accelerate scientific progress.

Abstract: Computational reproducibility is the ability to obtain identical results from the *same* data with the *same* computer code. It is a building block for transparent and cumulative science because it enables the originator and other researchers, on other computers and later in time, to reproduce and thus understand how results came about, while avoiding a variety of errors that may lead to erroneous reporting of statistical and computational results. In this tutorial, we demonstrate how the R package `repro` supports researchers in creating fully computationally reproducible research projects with tools from the software engineering community. Building upon this notion of fully automated reproducibility, we present several applications including the preregistration of research plans with code (Preregistration as Code, PAC). PAC eschews all ambiguity of traditional preregistration and offers several more advantages. Making technical advancements that serve reproducibility more widely accessible for researchers holds the potential to innovate the research process and to help it become more productive, credible, and reliable.

Keywords: open science; computational reproducibility; preregistration; R; R Markdown; Make; GitHub; Docker

check for updates

Citation: Peikert, A.; van Lissa, J.; Brandmaier, A.M. Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once. *Psych* **2021**, *3*, 836–867. https://doi.org/10.3390/psych3040053

Academic Editor: Alexander Robitzsch

Received: 11 October 2021
Accepted: 25 November 2021
Published: 9 December 2021

1. Introduction

Scientists increasingly strive to make research data, materials, and analysis code openly available. Sharing these digital research products can increase scientific efficiency by enabling researchers to learn from each other, reuse materials, and increase scientific reliability by facilitating the review and replication of published research results. To some extent, these potential benefits are contingent on whether these digital research products are reproducible. Reproducibility can be defined as the ability of anyone to obtain identical results from the *same* data with the *same* computer code (see [1] for details). The credibility of empirical research results hinges on their objectivity. Objectivity in this context means, "in principle it [the research finding] can be tested and understood by anybody." ([2] p. 22). Only a reproducible result meets these requirements. Therefore, reproducibility has long been considered integral to empirical research. Unfortunately, despite increasing commitment to open science practices and good will, many projects can not yet be reproduced by other research teams [3]. This is because there are various challenges to making a research project reproducible (e.g., missing software dependencies or ambiguous documentation of the exact computational steps taken), and there is a lack of best practices for overcoming these challenges (but see [4]). With technological advancement, however, it is now possible to make all digital products related to a research project available in a manner that enables automatic reproduction of the research project with minimal effort.

With this paper, we pursue two aims. First, we want to introduce researchers to a notion of automated reproducibility that requires no manual steps, apart from the initial setup of the software environment. Secondly, we discuss the implications of automated reproducibility for changing the general approach to research. With regard to the first goal, we discuss how to address four common threats to reproducibility, using tools originating from software engineering (see [1] for details), and present a tutorial on how to employ these tools to achieve automated reproducibility. A single tutorial cannot comprehensively introduce the reader to the detail of individual tools, but this tutorial is intended to help readers get started with a basic workflow. The tutorial is aimed at researchers who regularly write code to analyze their data and are willing to make relevant code, data, and materials available, either publicly or on request. Ideally, the reader has already created a dynamic document at some point in time (e.g., with R Markdown or Jupyter) and used some form of version control (e.g., Git). The R package `repro` supports researchers in setting up the required software and in adopting this workflow. We present automated reproducibility as a best practice; a goal that is not always fully achieved due to limited resources, technical restrictions, or practical considerations, but is worth striving for nonetheless.

In pursuit of the second aim, we present a strictly reproducible and unambiguous form of preregistration [5] that builds upon implementing this reproducible workflow, the so-called *Preregistration as Code* (PAC). PAC involves preregistering the intended analysis code and the major part of the final scientific report as a dynamic document, including typical sections like introduction, methods, and results. The resulting dynamic document closely resembles the final manuscript but uses simulated data to generate placeholder results (e.g., figures, tables, and statistics). Simulated data serve two functions, they allow to test the code for the planned analyses and for preregistering the exact presentation of the results. Once the empirical data are available, these replace the simulated data; the results are then updated automatically, and the discussion can be written to finalize the report.

Scientific organizations and funding bodies increasingly demand transparent sharing of digital research products, and researchers are increasingly willing to do so. However, although the sharing of such digital research products is a necessary condition for reproducibility, it is not a sufficient one. This was illustrated by an attempt to reproduce results from open materials in the journal *Cognition* [6]. Out of 35 published articles with open code and data, the results of 22 articles could be reproduced, but further assistance from the original authors was required in 11 of these cases. For 13 articles, at least one outcome could not be reproduced—even with the original authors' assistance. Another study of 62 registered reports found that only 41 had data available, and 37 had analysis scripts available [3]. The authors could execute only 31 of the scripts without error and reproduce the results of only 21 articles (within a reasonable time). These failed attempts to reproduce findings highlight the need for widely accepted reproducibility standards because open repositories do not routinely provide sufficient information to reproduce relevant computational and statistical results. If digital research products are available but not reproducible, their added value is limited.

This tutorial demonstrates how R users can make digital research products more reproducible, while striking a balance between rigor and ease-of-use. A rigorous standard increases the likelihood that a project will remain reproducible as long as possible. An easy-to-use standard, on the other hand, is more likely to be adopted. Our approach is to promote broad adoption of such practices by ensuring a “low threshold”, by making it easy to get started, while enabling a “high ceiling” by ensuring that they are compatible with more complex rigorous solutions. As researchers become more proficient in using the tools involved, they can thus further improve the reproducibility of their work.

We have structured the tutorial with a *learning-by-doing* approach in mind, such that readers can follow along on their own computers. We explicitly encourage readers to try out all R commands for themselves. Unless stated otherwise, all code blocks are meant to be run in the statistical programming language R ([7] tested with version 4.0.4).

2. Threats to Reproducibility and Appropriate Remedies

From our own experience with various research projects, we have identified the following common threats to reproducibility:

1. *Multiple inconsistent versions of code, data, or both*; for example, the data set may have changed over time because outliers were removed at a later stage or an item was later recoded; or, the analysis code may have been modified during the writing of a paper because a bug was removed at some point in time. It may then be unclear which version of code and data was used to produce some reported set of results.
2. *Copy-and-paste errors*; for example, results are often manually copied from a statistical computing language into a text processor; if a given analysis is re-run and results are manually updated in the text processor, this may inadvertently lead to inconsistencies between the reported result and the reproduced result.
3. *Undocumented or ambiguous order of computation*; for example, with multiple data and code files, it may be unclear which scripts should be executed in what order; or, some of the computational steps are documented (e.g., final analysis), but other steps were conducted manually without documentation (e.g., executing a command manually rather than in a script; copy-and-pasting results from one program to another).
4. *Ambiguous software dependencies*; for example, a given analysis may depend on a specific version of a specific software package, or rely on software that might not be available on a different computer, or no longer exist at all; or a different version of the same software may produce different results.

We have developed a workflow that achieves long-term and cross-platform computational reproducibility of scientific data analyses. It leverages established tools and practices from software engineering and rests on four pillars that address the aforementioned causes of non-reproducibility [1]:

1. Version control
2. Dynamic document generation
3. Dependency tracking
4. Software management

The remainder of this section briefly explains why each of these four building blocks is needed and details their role in ensuring reproducibility. A more extensive treatment of these tools is given in Peikert and Brandmaier [1].

Version control prevents the ambiguity that arises when multiple versions of code and data are created in parallel during the lifetime of a research project. Version control allows a clear link between which results were generated by which version of code and data. This addresses the first threat to reproducibility, because results can only be said to be reproducible if it is clear which version of data and code produced them. We recommend using `Git` for version control, because of its widespread adoption in the R community.

`Git` tracks changes to all project-related files (e.g., materials, data, and code) over time. At any stage, individual files or the entire project can be compared to, or reverted to, an earlier version. Moreover, contributions (e.g., from collaborators) can be compared to and incorporated in the main version of the project. Version control thus reduces the risk of losing work and facilitates collaboration. `Git` is built around snapshots that represent the project state at a given point in time. These snapshots are called *commits* and work like a "save" action. Ideally, each commit has a message that succinctly describes these changes. It is good practice to make commits for concrete milestones (e.g., "Commented on Introduction", "Added SES as a covariate", "Address Reviewer 2's comment 3"). This makes it easier to revert specific changes than when multiple milestones are joined in one commit, e.g., "Changes made on 19/07/2021". Each commit refers back to its ancestor, and all commits are thus linked in a timeline. The entirety of commits (i.e., the version-controlled project) is called a repository. In `Git`, specific snapshots of a repository can be tagged, such that the user can clearly label which version of the project was used to create a preregistration, preprint, or final version of the manuscript as accepted by a journal. `Git` has additional features beyond basic version control, such as "branches" (parallel versions of a project that can later be merged again) to facilitate simultaneous collaboration. Vuorre and Curley [8] provide a more extensive treatment of how `Git` functions and how to use `Git` for research. Bryan [9] provides additional information on how to track `R Markdown` documents. Collaborating via `Git` is facilitated by uploading the repository to a cloud-based service. We recommend GitHub as a host for `Git` repositories because of its popularity among R users. GitHub has many tools that facilitate working with `Git`— particularly in project management and collaboration—but these are not central to achieving reproducibility.

Second, we rely on *dynamic document generation*. The traditional way of writing a scientific report based on a statistical data analysis uses two separate steps conducted in two different programs. The researcher writes text in a word processor, and conducts the analysis in another program. Results are then (manually) copied and pasted from one program to another, a process that often produces inconsistencies [10].

Dynamic document generation integrates both steps. Through dynamic document generation, code becomes an integral, although usually hidden, part of the manuscript, complementing the verbal description and allowing interested readers to gain a deeper understanding of the contents [11,12]. `R Markdown` uses `Markdown` for text formatting and R (or other programming languages) for writing the statistical analysis. `Markdown` is a lightweight text format in plain text with a minimal set of reserved symbols for formatting instructions. This way, `Markdown` does not need any specialized software for editing. It is userfriendly (unlike, for example, `LaTeX` [13]), works well with version control systems, and can be exported to various document formats, such as HTML websites, a Microsoft Word document, a typeset PDF file (for example, via LaTeX journal templates), or a Powerpoint presentation. `Markdown` can be used for all sorts of academic documents, ranging from simple sketches of ideas to scientific manuscripts [14] and presentations [15], or even résumés [16]. `R Markdown` extends regular `Markdown` by allowing users to include R code chunks (in fact, arbitrary computer code ([17] Chapter 15, Chapter 15, Other Languages)) into a `Markdown` document. Upon rendering the document, the code blocks are executed, and their output is dynamically inserted into the document. This allows the creation of (conditionally) formatted text, statistical results, and figures that are guaranteed to be up-to-date because they are created anew every time the document is rendered to its output format (e.g., presentation slides or a journal article). Xie et al. [17] provides an extensive yet practical introduction to most features of `R Markdown`.

While version control and dynamic document generation are becoming more common, we have argued that two more components are required and that each component alone is unlikely to guarantee reproducibility [1,4]. In practice, dependencies between project files (e.g., information on what script uses which data file and what script needs to be run first) or on external software (e.g., system libraries or components of the programming language, such as other R packages) are frequently unmentioned or not exhaustively and unambiguously documented.

Dependency tracking helps automatically resolve dependencies between project files. In essence, researchers provide a collection of *computational recipes*. A computational recipe describes how inputs are processed to deterministically create a specific output in a way that is automatically executable. The concept of computational recipes is central to our understanding of reproducibility because it enables a unified way to reproduce a project automatically. Similar to a collection of cooking recipes, we can have multiple products (*targets*) with different ingredients (*requirements*) and different steps of preparation (*recipes*). In the context of scientific data analysis, targets are typically the final scientific report (e.g., the one to be submitted to a journal) and possibly intermediate results (such as preprocessed data files, simulation results, and analysis results). A workflow that involves renaming variable names by hand in a graphical spreadsheet application, for example, is therefore incompatible with automated reproducibility. Another property of a computational recipe is that the same inputs should always result in the same outputs. For most computer code (given the same software is used), this property is fulfilled. However, one noteworthy exception is the generation of pseudo-random numbers. Whenever random numbers are used in a computation, it is only reproducible if the random number generator generates the same numbers. To ensure identical random numbers, users may fix the state of the random number generated with a so-called seed (e.g., `set.seed()` in R), but they also need to guarantee that the pseudo-random number generator is unchanged (see [1]).

We recommend using `Make` for dependency tracking because it is language independent. The following hypothetical example illustrates the utility of `Make` and a suitable `Makefile`. Consider a research project that contains a script to simulate data (`simulate.R`) and a scientific report of the simulation results written in `R Markdown` (`manuscript.Rmd`). A `Makefile` for this project could look like this:

```
1 manuscript.pdf: manuscript.Rmd simulated_data.csv
2 Rscript -e 'rmarkdown::render("manuscript.Rmd")'
3
4 simulated_data.csv: simulate.R
5 Rscript -e 'source("simulate.R")'
```

There are two targets, the final rendered report (`manuscript.pdf`, l. 1) and the simulation results (`simulation_results.csv`, l. 4). Each target is followed by a colon and a list of requirements. If a requirement is newer than the target, the recipe will be executed to rebuild the target. If a requirement does not exist, `Make` uses a recipe to build the requirement before building the target. Here, if one were to build the final `manuscript.pdf` by rendering the `R Markdown` with the command shown in l. 2, `Make` would check whether the file `simulation_results.csv` exists; if not, it would issue the command in l. 5 to run the simulation before rendering the manuscript. This ensures that the simulated data are present before the manuscript is built, and that the simulation is re-run and the manuscript is rebuilt if the simulation code was changed. `Make` therefore offers a standardized process to reproduce projects, regardless of the complexity or configuration of the project. Note that the Workflow for Open Reproducible Code in Science (WORCS) we presented elsewhere [4] does not explicitly contain this dependency tracking element, but its strict structure of only containing one definite `R Markdown` still makes dependencies between files unambiguous.

A version-controlled dynamic document with dependency tracking still relies on external software. Troubleshooting issues specific to a particular programming language or dependent tool typically requires considerable expertise and threatens reproducibility. *Software management* refers to the act of providing records of, or access to, all software packages and system libraries a project depends on. One comprehensive approach to software management is containerization. The central idea is that by "[..] packaging the key elements of the computational environment needed to run the desired software [makes] the software much easier to use, and the results easier to reproduce [...]" ([18] p. 174).

Docker is a popular tool for containerization. It manages software dependencies by constructing a virtual software environment independent of the host software environment. These so-called "Docker images" function like a virtual computer (i.e., a "sand box" a computational environment seperated from the host). A Docker image contains *all* software dependencies used in an analysis—not just R packages, but also R and Rstudio, and even the operating system. This is important because low level functionality may impact the workings of higher-order software like R, such as calls to random number generators or linear algebra libraries. All of the differences in computational results that could be caused by variations in the software used are hence eliminated.

Note that the software environment of the Docker image is completely separate from the software installed on your computer. This separation is excellent for reproducibility but takes some getting used to. For example, it is important to realize that software available on your local computer will *not* be accessible within the confines of the Docker image. Each dependency that you want to use within the Docker image must be explicitly added as a dependency. Furthermore, using Docker may require you to install software on an operating system that may not be familiar to you. The images supplied by the rocker project [19], for example, are based on Linux.

There are two ways to build a Docker image. First, users can manually install whatever software they like from within the virtual environment. Such a manually build environment can still be ported to all computers that support Docker. However, we prefer the second way of building images automatically from a textual description called Dockerfile. Because the Dockerfile clearly describes how which software is installed, the installation process can be repeated automatically. Users can therefore quickly change the software environment, for example, update to another R version or given package version. Packaging all required software in such an image requires considerable amounts of storage space. Two major strategies help to keep the storage requirements reasonable. One is to rely on pre-made images that are maintained by a community for particular purposes. For example, there are pre-made images that only include what is necessary for R, based on Ubuntu containers [19]. Users can then install whatever they need in addition to what is provided by these pre-compiled images. The image that was used for this article uses 1.35 GiB of disk space. The image for this project includes Ubuntu, R, RStudio, LaTeX as well as a variety of R packages like tidyverse [20] and all its dependent packages, amounting to 192 R packages.

A second strategy is to save a so-called Dockerfile, which contains only a textual description of all commands that need to be executed to recreate the software environment. Dockerfiles are tiny (the Dockerfile (https://github.com/aaronpeikert/repro-tutorial/blob/main/Dockerfile accessed on 11 October 2021) for this project has a size of only 1.55 KiB). However, they rely on the assumption that all software repositories that provide the dependent operating systems, pieces of software, and R packages will continue to remain accessible and provide historic software versions. For proper archiving, we therefore recommend storing a complete image of the software environment, in addition to the Dockerfile. A more comprehensive overview of the use of containerization in research projects is given by Wiebels and Moreau [21]. Note that WORCS, which we presented elsewhere [4] relies on the R package renv [22] for software management. Although renv is more lightweight and easier to use than Docker, it is not as comprehensive because it only takes snapshots of the R packages instead of all software used.

To summarize, the workflow by Peikert and Brandmaier [1] requires four components (see Figure 1) dynamic document generation (using R Markdown), version control (using Git), dependency tracking (using Make), and software management (using Docker). While

R Markdown and Git are well integrated into the R environment through RStudio, Make and Docker require a level of expertise that is often beyond the training of scholars outside the field of information technology. This presents a considerable obstacle to the acceptance and implementation of the workflow. To overcome this obstacle, we have developed the R package repro that supports scholars in setting up, maintaining, and reproducing research projects in R. Importantly, a reproducible research project created with repro does not have the repro package itself as a dependency. These projects will remain reproducible irrespective of whether repro remains accessible in future. Users do not need to have repro installed to reproduce a project; in fact, they do not even need to have R installed because the entire project can be rebuilt inside a container with R installed. In the remainder, we will walk you through the creation of a reproducible research project with the package repro.

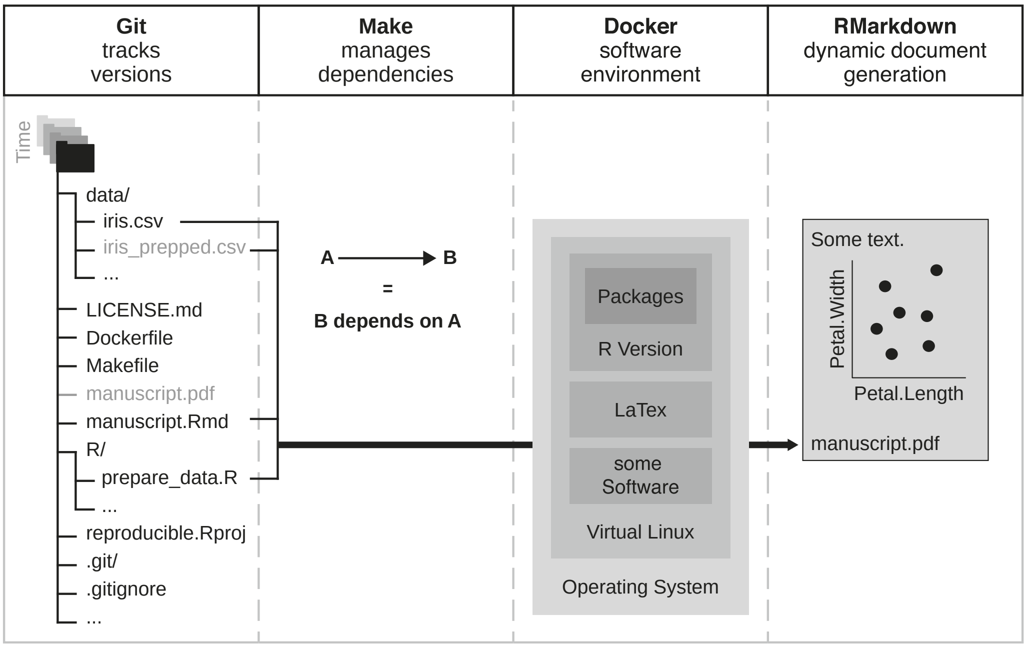


Figure 1. Schematic illustration of the interplay of the four components (in dashed columns) central to the reproducible workflow: version control (Git), dependency tracking (Make), software management (Docker), and dynamic document generation (R Markdown). Git tracks changes to the project over time. Make manages dependencies among the files. Docker provides a container in which the final report is built using dynamic document generation in R Markdown. Adapted from Peikert and Brandmaier [1] licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0 accessed on 4 December 2021).

3. Creating Reproducible Research Projects

One impediment to the widespread adoption of a standard for reproducible research is that learning to use the required tools can be quite time-intensive. To lower the threshold, the R package repro introduces helper functions that simplify the use of complicated and powerful tools. The repro package follows the format of theusethis (https://usethis.r-lib.org) [23] package, which provides helper functions to simplify the development of R packages. The repro package provides similar helper functions, but focuses on reproducibility-specific utilities. These helper functions guide end-users in the use of reproducibility tools, provide feedback about what the computer is doing and suggest what the user should do next. We hope this makes reproducibility tools more accessible by enabling beginner-level users to detect their system's state accurately and act correspondingly ([24] Chapter 8: "Automation and Situation Awareness"). These wrappers are merely a support system; as users learn to use the underlying tools, they can rely less on repro and use these tools directly to solve more complex problems.

This tutorial assumes that the user will be working predominantly in R with the help of RStudio. It describes basic steps that we expect to be relevant for small-scale psychological research projects that do not rely on external software or multistage data processing (for those requirements see Section 4). Of course, your specific situation might involve additional, more specialized steps. After completing the tutorial, you should be able to customize your workflow accordingly.

The first step is to install the required software. We assume that you have installed R ([7] version 4.0.4) and RStudio ([25] version 1.4) already but the tutorial will guide you in detail through the installation of other necessary software with the help of the R package `repro` (https://github.com/aaronpeikert/repro accessed on 4 December 2021) [26]. In case you have either not installed R and RStudio or are unsure if they are up-to-date, you might want to consult our installation advice in the Online Supplementary Material (https://github.com/aaronpeikert/repro-tutorial/blob/main/install.md accessed on 4 December 2021) that covers the installation of all software necessary for this tutorial in three steps. The installation advice (https://github.com/aaronpeikert/repro-tutorial/blob/main/install.md accessed on 11 October 2021) may also help Windows users who have problems installing `Docker`.

Unfortunately, Docker requires administrator rights to run, which may not be available to all researchers. We recommend `renv` [22] in cases where no administrator rights can be obtained but can not detail its use in this document. `renv` tracks which R package is installed from which source in which version in a so-called `lockfile`. This lockfile is then used to reinstall the same packages on other computers or later in time. For a more thorough discussion, see Lissa et al. [4].

Start RStudio and install the package `repro`[26]. It will assist you while you follow the tutorial.

```r
# repro is not on CRAN yet
options(
repos = c(aaronpeikert = 'https://aaronpeikert.r-universe.dev',
CRAN = 'https://cloud.r-project.org')
)
install.packages('repro')
```

To verify that you have indeed installed and set up the required software for this workflow, you can use the “check functions”. These also illlustrate how `repro` assists the user in setting up a reproducible workflow. In the example below, we use the double-colon operator to explicitly indicate which functions originate in the `repro` package. If the package is loaded (using `library("repro")`), it is not necessary to use this double-colon notation.

```r
# `package::function()` → use function from package without `library(package)`
repro::check_git()
```

```
## v Git is installed, don’t worry.
```

```r
repro::check_make()
```

```
## v Make is installed, don’t worry.
```

```r
repro::check_docker()
```

```
## v Docker is installed, don’t worry.
```

These functions check whether specific dependencies are available on the user’s system, and if not, explain what further action is needed to obtain it. Sometimes they ask the user to take action; for example, the following happens if you are a Windows user who does not have Git installed:

```
repro::check_git()
```

```
## x Git is not installed.

## i We recommend Chocolately for Windows users.

## x Chocolately is not installed.

## * To install it, follow directions on:
##   'https://chocolatey.org/docs/installation'

## i Use an administrator terminal to install chocolately.

## * Restart your computer.

## * Run 'choco install -y git' in an admin terminal to install Git.
```

The messages from `repro` try to help the user solve problems. They are adjusted to your specific operating system and installed dependencies. Before you continue, we ask you to run the above commands to check `Git`, `Make`, and `Docker`—both to become familiar with the functionality of the `check_*()` functions and to make sure your system is prepared for the remainder of this tutorial.

After you have installed the necessary software, we suggest that you set up a secure connection to GitHub:

```
repro::check_github()
```

```
## v You and GitHub are on good terms, don't worry.
```

If you know what Secure Shell (SSH) is and want to use it, you may alternatively use:

```
# only an alternative: DO NOT USE if you are unsure what SSH means
repro::check_github(auth_method = "ssh")
```

```
## v You and GitHub are on good terms, don't worry.
```

If necessary, follow any instructions presented until all checks are passed.

3.1. Creating an RStudio Project

We start by creating a project folder with RStudio by clicking the menu item:

File → New Project... → New Directory → Example Repro Template

This creates a project with a sample analysis. This sample analysis consists of a single R Markdown document and a single data file. The only special thing about the R Markdown document is the `repro` metadata that we will learn about later. However, you may turn any other template or existing R project into a reproducible research project by adding those `repro` metadata there.

3.2. Implementing Version Control

Now that your project is set up, we will introduce you to version control with `Git`. `Git` does not automatically track all files in your project folder; rather, you must manually add files to the `Git` repository. To make sure you do not accidentally add files that you do not wish to share (e.g., privacy-sensitive data), you can list specific files that you do not want to track in the `.gitignore` file. You can also block specific filetypes; for example, to prevent accidentally sharing raw data. You can add something to the `.gitignore` file directly or with this command:

```
usethis::use_git_ignore("private.md")
```

Now the file `private.md` will not be added to the `Git` repository, and hence also not be made public if you push the repository to a remote service like GitHub. Please also consider carefully whether you can include data in the repository without violating privacy rights. If you are not allowed to share your data publicly, add the data file(s) to the `.gitignore` file and only share them on request.

New users are advised to explicitly exclude any sensitive files before proceeding. When you are ready, you can begin tracking your remaining files using `Git` by running:

```
usethis::use_git()
```

For `Git` to recognize changes to a given file, you have to stage and then commit these changes (this is the basic save action for a project snapshot). One way to do this is through the visual user interface in the RStudio `Git` pane (see Figure 2). Click on the empty box next to the file you want to stage. A checkmark then indicates that the file is staged. After you have staged all of the files you want, click on the commit button, explain in the commit message why you made those changes, and then click on commit. This stores a snapshot of the current state of the project.

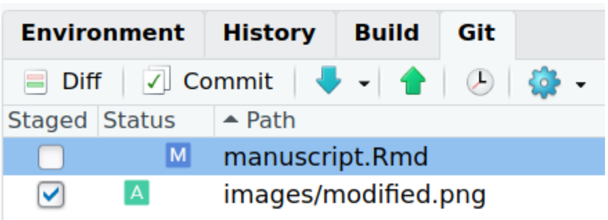


Figure 2. The Git pane in R Studio, showing `manuscript.Rmd` modified but unstaged and `modified.png` newly added and staged.

The files you created and the changes you made have not yet left your computer. All snapshots are stored in a local repository in your project folder. To back up and/or share the files online, you can push your local repository to a remote repository. While you can choose any `Git` service (like GitLab or BitBucket), we will use GitHub in this tutorial. Before you upload your project to GitHub, you need to decide whether you would like the project to be publicly accessible (viewable by anyone, editable by selected collaborators) or if you want to keep it private (only viewable and editable by selected collaborators). To upload the project publicly to GitHub use:

```
usethis::use_github()
```

To upload it privately:

```
usethis::use_github(private = TRUE)
```

Depending on your computer's configuration, it may ask you to set up a secure connection to GitHub. In this case, first, follow the suggestions shown on the R console.

3.3. Using Dynamic Document Generation

Now that you have created a version-controlled project, we will proceed with dynamic document generation. A dynamic document has three elements:

1. Text (prose; e.g., a scientific paper or presentation)
2. Executable code (e.g., analyses)
3. Metadata (e.g., title, authors, document format)

`R Markdown` is a type of dynamic document well-suited to the RStudio user interface. The text of an `R Markdown` is formatted by `Markdown` (see [27] for technical details and [17] for practical guidance). The code mostly consists of R code (although other programming languages are supported, like Python, C++, Fortran, etc). The following example serves to illustrate the Markdown syntax. It shows how to create a heading, a word in bold font, a citation, and a list of several items in Markdown:

```
<!--this is a Markdown file -->
# Heading (level 1)

Normal text.
Important **word** in bold.
A citation: @einstein1935 did important research on this topic.

## Subheading (level 2)

To do list:

* Do research
* Do more research
* Spend time with family and friends
```

One advantage of this type of markup for formatting is that it can be rendered to many different output formats—both in terms of file types, like .docx, .html, .pdf, and in terms of style, e.g., specific journal requirements. For social scientists, the papaja package [28] may be relevant, as it produces manuscripts that follow the American Psychological Association formatting requirements [29]. R Markdown files are plain text, which is more suitable for version control using Git than binary files generated by some word processors. Some users might find it easier to activate the "Visual Editor" of RStudio ([Ctrl] + [Shift] + [F4] or click on the icon that resembles drawing materials or a compass in the upper right corner of the R Markdown document), which features more graphical elements like a traditional word processor but still creates an R Markdown underneath with all of its flexibility. The visual editor has some additional benefits, such as promoting best practices (for example, each sentence should be written on a new line, which makes it easier to track changes across versions) and improving the generation of citations and references to tables and figures.

Now that you are familiar with Markdown formatting basics, we turn our attention to including code and its results in the text. Code is separated by three backticks (and the programming language in curly brackets) like this:

````
This is normal text, written in Markdown.

```{r}
# this is R code
1 + 1
```
````

The hotkey [Control] + [Alt] + [i] inserts a block of code in the file. The results of code enclosed in such backticks will be dynamically inserted into the document (depending on specific settings). This means that whenever you render the R Markdown to its intended output format, the code will be executed and the results updated. The resulting output document will be static, e.g., a pdf document, and can be shared wherever you like, e.g., on a preprint server.

Once the R Markdown file has been rendered to a static document (the output, e.g., PDF), the resulting file is decoupled from the R Markdown and the code that created it. This introduces a risk that multiple versions of the static document are disseminated, each with slightly different results. To avoid ambiguity, we, therefore, recommend referencing the identifier of the Git commit at the time of rendering in the static document. Simply put, a static document should link to the version of the code that was used to create it. The `repro` package comes with the function `repro::current_hash()` for this purpose. This document was created from the commit with the hash `a8bb0d4` (view on GitHub).

Now that you know how to write text and R code in an `R Markdown`, you need to know about metadata (also called: YAML front matter). These metadata contain information about the document, like the title and the output format. Metadata are placed at the beginning of the document and are separated from the document body by three dashes. The following example is a full markdown document where the metadata (the "YAML front matter") are in lines 1–6. Some metadata fields are self-explanatory (like the author field), and exist across all output formats (like the title field). Others are specific to certain output formats or R packages.

```
1  ---
2  title: "A Tutorial on how to Do the Same Thing More Than Once"
3  author: Aaron Peikert, Caspar J. van Lissa, and Andreas M. Brandmaier
4  abstract: A hitchhiker's guide to reproducible research in R
5  output: html_document
6  ---
7  
8  # Introduction
9  
10 Important for reproducibility:
11 
12 1. *Version control*
13 2. *Dynamic document creation*
14 3. *Dependency tracking*
15 4. *Software management*
16 
17 ```{r}
18 # this is R code
19 t.test(extra ~ group, data = sleep)
20 ```
```

3.4. Manage Software and File Dependencies

The `repro` package adds fields to the metadata to list all dependencies of the research project. This includes R scripts, data files, and external packages. The format is as follows (see everything below the line *repro:*):

```
---
title: "A tutorial on how to do the same thing more than once"
author: Aaron Peikert, Caspar J. Van Lissa and Andreas M. Brandmaier
output: html_document
repro:
  scripts:
    - R/load.R
  data:
    - data/mtcars.csv
  packages:
    - tidyverse
    - usethis
    - gert
---
```

This information clarifies what dependencies (in the form of files and R packages) a project relies on. `repro` uses this information to construct a `Makefile` for the dependencies on other files and a `Dockerfile` that includes all required packages. Together, these two files form the basis for consistency within a research project and consistency across different systems. The function `repro::automate()` converts the metadata from all `R Markdown` files in the project (all files with the ending `.Rmd`) to a `Makefile` and a `Dockerfile`. These files allow users (including your future self) to reproduce every step in the analysis automatically. Please run `repro::automate()` in your project:

```
repro::automate()
```

It is important to re-run `repro::automate()` whenever you change the `repro` metadata, change the output format, or add a new `R Markdown` file to the project to keep the `Makefile` and `Dockerfile` up to date. There is no harm in running it too often. Other than the `Makefile` and the `Dockerfile`, which are created in the document root path, `repro` generates a few more files in the `.repro` directory (which we will explain in detail later), all of which you should add and commit to `Git`.

3.5. Reproducing a Project

If someone (including you) wants to reproduce your project, they first have to install the required software, that is `Make`, and `Docker`. Remember, you can use the `check_*`-functions to test if these are installed:

```
repro::check_make()
```

```
## v Make is installed, don't worry.
```

```
repro::check_docker()
```

```
## v Docker is installed, don't worry.
```

When these are set up, they can ask `repro` to explain how they should use `Make` and `Docker` to reproduce the project (or you could explain it to them):

```
repro::reproduce()
```

```
## * To reproduce this project, run the following code in a terminal:

##    make docker &&
##    make -B DOCKER=TRUE
```

If you feel uncomfortable using the terminal directly, you can send the command to the terminal from within R:

```
system(repro::reproduce())
```

The only "hard" software requirement for reproducing a project is `Docker`, assuming users know how to build a `Docker` image and run `Make` within the container. However, if they have installed `Make` in addition to `Docker`, they do not even need to know how to use `Docker` and can simply rely on the two `Make` commands "`make docker`" and "`make -B DOCKER=TRUE`".

3.6. Summary

1. Install the `repro` package:

```
options(
repos = c(aaronpeikert = 'https://aaronpeikert.r-universe.dev',
CRAN = 'https://cloud.r-project.org')
)
install.packages('repro')
```

2. Check the required software:

```r
repro::check_git()
```

```
## v Git is installed, don't worry.
```

```r
repro::check_github()
```

```
## v You and GitHub are on good terms, don't worry.
```

```r
repro::check_make()
```

```
## v Make is installed, don't worry.
```

```r
repro::check_docker()
```

```
## v You are inside a Docker container!
```

3. Create an R project or use an existing one. Do not forget to add `repro` metadata (i.e., packages, scripts, data).

```yaml
repro:
  scripts:
    - R/load.R
  data:
    - data/mtcars.csv
  packages:
    - tidyverse
```

The sample `repro` project already has theese metadata:

```r
repro::use_repro_template("/some/folder")
```

4. Let `repro` generate `Docker`- and `Makefile`:

```r
repro::automate()
```

5. Enjoy automated reproducibility:

```r
repro::reproduce()
```

```
## * To reproduce this project, run the following code in a terminal:

##   make docker &&
##   make -B DOCKER=TRUE
```

4. Advanced Features

This section is for advanced users who want to overcome some limitations of `repro`. If you read this paper the first time, you will probably want to skip this section and continue reading from the section "Preregistration as Code." As explained above, `repro` is merely a simplified interface to the tools that enable reproducibility. This simplified interface imposes two restrictions. Users who ask themselves either, "How can I install software dependencies outside of R in the `Docker` image?" or "How can I express complex dependencies between files (e.g., hundreds of data files are preprocessed and combined)?" need to be aware of these restrictions and require a deeper understanding of the inner workings of `repro`. Other users may safely skip this section or return to it if they encounter such challenges.

The first restriction is that users must rely on software that is either already provided by the base `Dockerimage` "rocker/verse" or the R packages they list in the metadata. The metadata the `repro::automate()` function relies on can only express R packages as dependencies for the `Dockerfile` and only trivial dependencies (in the form of "file must exist") for the `Makefile`. Other software that users might need, like other programming languages, not yet installed LaTeX packages, etc., must be added manually. We plan to add support for commonly used ways to install software beyond R packages via the metadata and `repro::automate()`, for example, for system libraries (via `apt` the Ubuntu package manager), LaTeX packages (via `tlmgr` the Tex Live package manager), Python packages (via `pip` the python package manager). The second limitation is related to dependencies. `Make` can represent complex dependencies, for example: A depends on B, which in turn depends on C and D. If B is missing in this example, `Make` would know how to recreate it from C and D. These dependencies, and how they should be resolved, are difficult to represent in the metadata. Users, therefore, have to either "flatten" the dependency structure by simply stating that A depends on B, C, and D, thereby leaving out important information or express the dependencies directly within the `Makefile`.

The following section explains how to overcome these limitations despite reliance on the automation afforded by `repro`. Lifting these restrictions requires the user to interact more directly with `Make` or `Docker`. Users need to understand how repro utilizes `Make` and `Docker` internally to satisfy more complicated requirements.

Let us have a closer look at the command for reproducing a repro project: `make docker && make -B DOCKER=TRUE`; which consists of two processing steps. First, it recreates the virtual software environment (`Docker`), and then it executes computational recipes in the virtual software environment (`Make`). The first step is done by the command `make docker`. The command `make docker` will trigger `Make` to build the target called `docker`. The recipe for this target builds an image from the `Dockerfile` in the repository. The `&&` concatenates both commands and only runs the second command if the first is successful. Therefore, the computational steps are only executed when the software environment is set up. The second step executes the actual reproduction and is again a call to `Make` in the form of `make -B DOCKER=TRUE` with three noteworthy parts. First, a call to `make` without any explicit target will build the `Make` target `all`. Second, the flag `-B` means that `Make` will consider all dependencies as outdated and will hence rebuild everything. Third, repro constructs `Make` targets so that if you supply `DOCKER=TRUE` they are executed within the `Docker` image of the project.

The interplay between `Docker` and `Make` resembles a chicken or egg problem. We have computational steps (`Make`) that depend on the software environment (`Docker`) for which we again have computational steps that create it. Users only require a deeper understanding of this interdependence when they either want to have more complex computational recipes than rendering an `R Markdown` or require software other than R packages.

Users can have full control over the software installed within the image of the project. `repro` creates three `Dockerfiles` inside the `.repro` directory. Two `Dockerfiles` are automatically generated. The first is `.repro/Dockerfile_base`. It contains information about the base image on which all the remaining software is installed. By default we rely on the "verse" images provided by the Rocker project [19]. These contain (among other software) the packages `tidyverse`, `rmarkdown`, and a complete LaTeX installation, which makes these images ideal for the creation of scientific manuscripts. Users can choose which R version they want to have inside the container by changing the version number in line 1 to the desired R version number. By default, the R version corresponds to the locally installed version on which `repro::automate()` was called the first time. The build date is used to install packages in the version that was available on the Comprehensive R Archive Network on this specific date and can also be changed. By default, this date is set to the date on which `repro::automate()` was called the first time. This way, the call to the automate function virtually freezes the R environment to the state it was called the first time inside the container. Below, you see the Docker base file we used to create this manuscript:

```
FROM rocker/verse:4.0.4
ARG BUILD_DATE=2021-05-06
WORKDIR /home/rstudio
```

The second automatically generated Dockerfile is .repro/Dockerfile_packages. Whenever repro::automate() is called, repro gathers all R packages from all .Rmd files and determines whether they should be installed from CRAN or GitHub fixed to the date specified in Dockerfile_base. Finally, there is one manually edited Dockerfile: .repro/Dockerfile_manual. It is blank by default and can be used to add further dependencies outside of R, like system libraries or external software. Using Docker may require you to install software on an operating system that may not be familiar to you. The images supplied by [19], for example, are based on the Ubuntu operating system. The most convenient way to install software on Ubuntu is through its package manager apt. If the following snippet is added to .repro/Dockerfile_manual, the Docker image will have, for example, Python installed. Other software is installed identically, only the software name is exchanged.

```
RUN apt-get update && apt-get install -y python3
```

Docker eventually requires a single Dockerfile to run, so repro::automate() simply concatenates the three Dockerfiles and saves the result into the main Dockerfile at the top level of the R project. With this approach, users of repro can build complex software environments and implement complex file dependencies. The standard repro metadata only make sure that all dependencies are available but does not allow you to specify custom recipes for them in the metadata. If you can formulate the creation of dependencies in terms of computational steps, e.g., the file data/clean.csv is created from data/raw.csv by script R/preprocess.R, you should include these in the Makefile. The Makefile that repro creates is only a template, and you are free to change it. However, make sure you never remove the following two lines:

```
include .repro/Makefile_Rmds
include .repro/Makefile_Docker
```

The file .repro/Makefile_Rmds contains the automatically generated targets from repro::automate() for the R Markdown files. This file should not be altered manually. If you are not satisfied with the automatically generated target, simply provide an alternative target in the main Makefile. Targets in the main Makefile take precedent.

The file .repro/Makefile_Docker does again contain a rather complicated template that you could, but should usually not modify. This Makefile coordinates the interplay between Make and Docker and contains targets for building (with make docker) and saving (with make save-docker) the Docker image. Additionally, it provides facilities to execute commands within the container. If you write a computational recipe for a target, it will be evaluated using the locally installed software by default. To evaluate commands inside the Docker image instead, you should wrap them in $(RUN1) command $(RUN2), as done in this example, which is identical to the first Make example we gave above except for the addition of $(RUN1) and $(RUN2) in l. 2:

```
1 simulated_data.csv: R/simulate.R
2 $(RUN1) Rscript -e 'source("R/simulate.R")' $(RUN2)
```

If users execute this in the terminal:

```
make data/simulation_results.csv
```

It behaves exactly as in the first Make example, the script R/simulate.R is run using the locally installed R. Because this translates simply to:

```
Rscript -e 'source("R/simulation.R")'
```

But if users use

```
make DOCKER=TRUE data/simulation_results.csv
```

It is evaluated within the `Docker` container using the software within it and not the locally installed R version:

```
docker run --rm --user 1000 -v "/home/rstudio":"/home/rstudio/"
reprotutorial Rscript -e 'source("R/simulate.R")'
```

To summarize, `repro` automates dependency tracking (in the form of `Make`) and software management (using `Docker`) without the necessity to learn both tools, but users with advanced requirements can still customize all aspects of both programs.

5. Preregistration as Code

Preregistration refers to the practice of defining research questions and planning data analysis before observing the research outcomes [5]. It serves to separate a-priori planned and theory-driven (confirmatory) analyses from unplanned and post-hoc (exploratory) analyses. Researchers are faced with a myriad of choices in designing, executing, and analyzing a study, often called researchers degrees of freedom. Undisclosed researcher degrees may be used to modify planned analyses until a key finding reaches statistical significance or to inflate effect size estimates, a phenomenon referred to as *opportunistic bias* [30]. Preregistration increases transparency by clarifying when and how researchers employ their degrees of freedom. It expressly does not restrict what researchers may do to gather or analyze their data.

There are still several shortcomings to preregistration. One is that written study plans are often interpretable in multiple ways. Empirical research has shown that, even when several researchers describe their analysis with the same terms, use the same data, and investigate the same hypothesis, their results vary considerably [31]. The current best practice to ensure comprehensive and specific preregistration is to impose structure by following preregistration templates [32,33]. However, such templates cannot ensure full transparency because it is impossible to verbally describe every detail of an analysis for any but the most straightforward analysis. This ambiguity causes a second problem, namely, comparing the initial plan and the resulting publication to decide if and how researchers deviated from the preregistration. This task is difficult because it is impossible to decide without additional information weather the analysis was actually carried out differently or just described differently. Even if researchers were faithful to the preregistration, readers may reach opposite conclusions because they have to compare two different text that may be worded differently or describe the same thing in varying levels of detail. A third limitation is that preregistrations are susceptible to non-reproducibility, just like primary research. To illustrate, a review of 210 preregistrations found that, even though 174 (67%) included a formal power analysis, only 34 (20%) of these could be reproduced [34]. Even when researchers have gone to great lengths in preregistering an analysis script, they sometimes inexplicably fail to reproduce their own results. For example, Steegen et al. [35] realized after publication that part of their preregistered code resulted in different test statistics than they reported initially (see their Footnote 7). A final limitation is that written plans may turn out to be unfeasible once data are obtained and analyzed. For example, a verbal description of a statistical model may be unidentified, e.g., if it includes reciprocal paths between variables or more parameters than observed data. Conversely, a model may be misspecified in a major way; for example, by omitting direct effects when the research question is about mediation, thus leading to a model with an unacceptable fit. Many researchers would only realize that such a model cannot be estimated once the data are obtained, thus necessitating a deviation from the preregistered plans.

The workflow described in this paper facilitates a rigorous solution to this problem: Instead of describing the analysis in prose, researchers include the code required to conduct the analysis in the preregistration. We term this approach of writing and publishing code at the preregistration stage *Preregistration as Code (PAC)*. PAC has the potential to eliminate undisclosed researchers degrees of freedom to a much greater extent than, e.g., preregistration templates. Moreover, it reduces overhead by removing the need to write a separate preregistration and manuscript. For PAC, researchers can write a reproducible, dynamically generated draft of their intended manuscript at the preregistration stage. This already includes most of the typical sections, such as introduction, methods, and results. These results are initially based on simulated data with the same structure as the data the authors expect to obtain from their experiments. For guidance on how to simulate data, see Morris et al. [36], Paxton et al. [37], and Skrondal [38], as well as the R packages `simstudy` [39] and `psych` [40].

Once the preregistration is submitted and real data have been collected or made available, the document can be reproduced with a single command, thus updating the Results section to the final version. Reproducibility is of utmost importance at this stage since the preregistration must produce valid results at two points in time, once before data collection and once after data collection. As outlined before, reproducibility builds upon four pillars (version control, dynamic document generation, dependency tracking, and software management). To use PAC, the dangers to reproducibility we described must be eliminated.

The idea of submitting code as part of a preregistration is not new (e.g., [41]). A prominent preregistration platform, The Open Science Framework, suggests submitting scripts alongside the preregistration of methods. In an informal literature search (we skimmed the first 300 results of Google Scholar with the keywords `("pre registration"|"pre-registration"|preregistration)&(code|script|matlab| python|"R")`) we only found close to a dozen published articles that did include some form of script as part of their preregistration. Though the notion of preregistering code has been around for a while (cf. [35]), it has not gained much traction—perhaps because, to date, this has constituted an extra non-standard step in the research process. This tutorial integrates the preregistration of code into the reproducible research workflow by encouraging researchers to preregister the whole manuscript as a dynamic document.

5.1. Advantages of PAC over Traditional Preregistration

We believe that pairing PAC with the workflow presented above offers five advantages over classical preregistration. First, PAC is merely an intermediate stage of the final manuscript, thus sparing authors from writing, and editors and reviewers from evaluating, two separate documents. Relatedly, writing the preregistration in the form of a research article has the advantage that researchers are usually familiar with this format. By contrast, a preregistration template is a novelty for many. Second, PAC is a tool for study planning. A study can be carried out more efficiently if all steps are documented clearly than when every step is planned ad hoc. Third, PAC removes ambiguity regarding the translation of verbal analysis plans into code. PAC is more comprehensive by design because its completeness can be empirically checked with simulated data. Evaluating the intended analysis code on simulated data will help identify missing steps or ambiguous decisions. PAC, therefore, minimizes undisclosed researchers degrees of freedom more effectively than standard preregistration does [33,41]. Fourth, despite its rigor, PAC accommodates data-dependent decisions if these can be formulated as code. Researchers can, for example, formulate conditions (e.g., in the form of if-else-blocks) under which they prefer one analysis type over the other. For example, if distributional assumptions are not met, the code may branch out to employ robust methods; or an analysis may perform automated variable selection mechanisms before running the final model. Another example of data-dependent decisions are more explorative analyses, i.e., explorative factor analysis or machine learning. Decisions that do not lend themselves to formulation in code, e.g., visual inspection, must still be described verbally or be treated as noted in the next section. Fifth, deviations from the preregistration are clearly documented because they are reflected in changes to the code, which are managed and tracked with version control.

5.2. Deviating from the Preregistration and Exploration

We would like to note that PAC allows explicit comparison of the preregistration and the final publication. Authors should retrospectively summarize and justify any changes made to the preregistered plan, e.g., in the discussion of the final manuscript (In section [Preregistration as Code—a Tutorial] we conducted an actual PAC and summarize the changes we make to the preregistered code in the discussion). During the analysis process, authors can additionally maintain a running changelog to explain changes in detail as they arise. Each entry in the changelog should explain the reasoning behind the changes and link to the commit id that applied the changes. This enables readers and reviewers to inspect individual changes and make an informed judgment about their validity and implications.

Deviations from the preregistration are sometimes maligned, as if encountering unexpected challenges invalidates a carefully crafted study [42]. However, we share the common view that deviation from a preregistration is not a problem [43]; rather, a failure to disclose such deviations is a problem. In fact, it is expected that most PACs will require some modification after empirical data becomes available. Often, deviations provide an opportunity to learn from the unexpected.

For example, imagine that authors preregistered their intention to include both "working memory" and "fluid intelligence" as covariates in an experimental study, examining the effect of task novelty on reaction time. When evaluating the planned analyses on the empirical data, these two covariates reveal high collinearity, thus compromising statistical inference. The authors decide to use PCA to extract common variance related to "intelligence", and include this component as a covariate instead. This change pertains to an auxiliary assumption (that working memory and fluid intelligence are distinct constructs), but does not undermine the core theory (that task novelty affects reaction time). Now imagine that a different researcher is interested in the structure of intelligence. This change to the preregistration directly relates to their theory of intelligence. That researcher might thus interpret the same result as an explorative finding, suggesting that these aspects of intelligence are unidimensional. A deviation from preregistration thus requires a judgement about what changes affect the test of the theory to what extend [44]. Only transparent reporting enables such judgment.

Another common misunderstanding is that preregistration, including PAC, precludes exploratory analyses. We differentiate between two kinds of exploration, neither of which is limited by PAC. The first, more traditional kind of exploration involves ad hoc statistical decisions and post hoc explanations of the results. Such traditional exploratory findings should be explicitly declared in the manuscript to distinguish them from confirmatory findings [5,43]. The second kind of exploration is through procedurally well defined exploration with exploratory statistical models that are standard in machine learning [45]. These models often involve dozens, if not hundreds of predictors, which makes it difficult to describe them verbally. With PAC, such models can be preregistered clearly and in comprehensive detail, and the researcher can precisely define a priori how much they want to explore. We specifically recommend PAC for such exploratory statistical models. The merit of preregistration in these cases is to communicate precisely how much exploration was done; a piece of information that is crucial to assess e.g., whether the results might be overfit ([46], p. 220f.).

5.3. Planned Analyses as Functions

Although researchers may use any form to preregister their planned analyses (e.g., scripts), we suggest writing three functions for each planned hypothesis: one to conduct the planned analysis, one to simulate the expected data, and one to report the results. Using functions makes the analysis more portable (i.e., it can easily be used for other datasets), and facilitates repeated evaluation, as is the case in a simulation study. The functions shown here do not contain executable code, but the interested reader can find working functions in the online Supplementary Materials (https://github.com/aaronpeikert/repro-tutorial/blob/main/R/simulation_funs.R accessed on 11 October 2021) that power the example below.

It is difficult to write analysis code when it is not clear what the expected data will look like. We therefore recommend first simulating a dataset that resembles the expected structure of the empirical data that will be used for the final analysis. Dedicated packages to simulate data for specific analyses exist .

The general format of a simulation function might be as follows:

```
simulate_data <- function(n, effect_size){
# 1. warn users that the results are "fake"
# 2. draw `n` samples with `effect_size`
# 3. format and return in expected data format
}
```

For linear models, simulating data is extremely simple:

```
simulate_data <- function(n, effect_size){
warning("This manuscript contains mock results based on simulated data.")
# Draw n samples from a normal distribution for predictor X
  x <- rnorm(n)
# Calculate dependent variable Y..
#.. as a function of population effect size and residual error
  y <- effect_size * x + rnorm(n)
# Return a data.frame
data.frame(x = x, y = y)
}
```

Next, write a function to conduct the planned analysis. This function should receive the data and compute all relevant results from it. The general format of an analysis function might be:

```
planned_analyis <- function(data){
# 1. preprocess e.g. with `rowMeans(data)`
# 2. conduct analysis e.g. with `t.test()`
# 3. `return(results)`
}
```

In the simplest case, an analysis function might already exist in R. For the linear model above, the analysis function might be:

```
planned_analyis <- function(data){
lm(y ~ x, data = data)
}
```

As soon as we have written `planned_anaylsis()` and `simulate_data()` we can iteratively improve both functions, e.g., until `planned_analysis()` runs without error and recovers the correct parameters from `simulate_data()`. The goal is to ensure that the output of `simulate_data()` works as input to the function `planned_analysis()`.

When the researchers are satisfied with the function `planned_analysis()`, they can think about the way the would they would like to report the analysis results via tables, plots, and text. The implementation of this reporting should be in the function `report_analysis()`.

```r
report_analysis <- function(results){
# 1. create markdown tables from results
# 2. conditionally interpret results e.g. if(p < .025)"Result is significant."
# (optional) visualize results
# 3. return results section formatted in markdown
}
```

This function should again accept the output of `planned_analysis()` as input. The output of this function should be formatted in Markdown. The idea is to automatically generate the full results section from the analysis. This way, the preregistration not only specifies the computation but also how the its results are reported. Various packages automatically generate well-formatted Markdown outputs of statistical reports or even entire tables of estimates or figures directly from R goal to help with this objective. Packages like `pander` [47], `stargazer` [48], `apaTables` [49] and `papaja` [28] help you to create dynamically generated professional looking results. The package `report` [50] is particularly noteworthy because it not only generates tables but also a straightforward interpretation of the effects as actual prose (e.g., it verbally quantifies the size of an effect).

Ideally, these three functions can be composed to create a "fake" results section, e.g., when composed to `report_analysis(planned_analysis(simulate_data()))` or `simulate_data() %>% planned_analysis() %>% report_analysis()` outputs a results section.

Turning a Dynamic Document into a Preregistration

After researchers are satisfied with their draft preregistration, they should archive a time-stamped and uneditable version of the project that serves as the preregistration. zenodo.org [51] is a publicly funded service provider that archives digital artefacts for research and provides digital object identifiers (DOI) for these archives. While the service is independent of GitHub—in terms of storage facilities and financing—you can link GitHub and zenodo.org. Please note that you can only link public GitHub repositories to zenodo.org. You may log into zenodo.org through your GitHub account. To log in with your GitHub account:

Navigate to https://zenodo.org/login/ → Log in with GitHub

To link zenodo.org and GitHub

Log into zenodo.org → Account → GitHub (https://zenodo.org/account/settings/github/)

Or:

Navigate to https://zenodo.org/account/settings/github/

After you have linked a GitHub repository, you trigger the archival by creating a GitHub release. To create GitHub release, navigate to GitHub:

```r
usethis::browse_github()
```

Then click on Releases → Draft a new release. Here you can add all relevant binary files but at least a rendered version of the manuscript and the Docker image.

To summarize, researchers need to write three functions, `planned_analysis()`, `simulate_data()`, and `report_analysis()` and embed these into a manuscript that serves as a preregistration in an uneditable online repository. After they gathered the actual data, they can replace the simulated data, render the dynamic manuscript (therefore run `planned_analysis()` on the actual data), and write the discussion.

5.4. Alternatives to Simulated Data

Simulating data may prove challenging to applied researchers. In the spirit of team science and collaboration, one feasible solution is to involve a statistical co-author. However, several easy alternatives exist. The downside of these alternatives is that they all rely indirectly on the use of real data. This introduces a risk that the planned analyses may be cross-contaminated by any exploratory findings. It is crucial to disclose any exposure to the data in preparation of the preregistration (PAC or otherwise). This exposure to the data may decrease trust in the objectivity of the preregistration. Moreover, researchers should take rigorous measures to prevent exposure to exploratory findings that may unintentionally influence their decision making.

The simplest method is to collect empirical data first, but set it aside and proceed with a copy of the data that is blinded by randomly shuffling the order of rows for each variable (independently of each other). Shuffling removes any associations between variables, while retaining information about the level of measurement and marginal distribution of each variable. If the hypotheses pertain to associations between variables, this treatment should thus be sufficient to prevent cross-contamination. The researcher can still access the information about means or proportions (e.g., the number of participants belonging to group "A" are in the dataset), but remain uninformed about relations between variables (e.g., members of group "A" have a greater mean in variable "Z"). Preregistration after data collection is common for secondary data analysis of data obtained by other research groups [52] but not so much within the same research project. We argue that it is still an eligible preregistration. Guidelines for clinical trials already recommend analysis of blinded data to test the feasibility of a preregistration [53].

Another alternative to simulated data is to conduct a pilot study [54] and use the pilot data to develop the preregistration. A pilot study has obvious advantages for study planning, since it lets the researcher evaluate the feasibility of many assumptions. However, we must warn our readers that while piloting is more traditional than our approach of blinding the data before preregistration, the data from the pilot study must not enter the analysis data set.

5.5. When Is PAC Applicable?

PAC is applicable to every study that can be preregistered and ultimately uses computer code for the statistical analysis. Two types of preregistrations are particularly amenable to PAC. First, pregistrations of clinical trials (called statistical analysis plans, International Council for Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use [53]) typically describe analyses in exhaustive detail and typically contain a detailed description of how results will be presented, including shells of tables and graphics [55]. PAC may significantly reduce the required workload while maintaining (and exceeding) the required standards for preregistering a clinical trial.

Second, preregistering exploratory statistical models (i.e., those with large numbers of competing models or those inspired by machine learning) is hardly feasible with standard preregistrations since they are too complex to describe and depend strongly on their software implementation. PAC, however, captures the precise algorithmic model, including its software implementation, and is ideal for preregistering these models [45].

5.6. Preregistration as Code: Tutorial

We have argued that PAC has several advantages over classic preregistration and have outlined its implementation. To illustrate how PAC works in practice and to help researchers implement PAC themselves, we provide a worked example. We will use an exemplary research question that was based on openly available data:

> "Is there a mean difference in the personality trait 'Machiavellism' between self-identified females and males?"

Again, we propose a preregistration format that closely resembles a classic journal article but uses simulated data and dynamic document generation to create a document that starts out as a preregistration and eventually becomes the final report. The complete preregistration source is available in the online Supplementary Material (https://github.com/aaronpeikert/repro-tutorial/blob/main/preregistration.Rmd accessed on

11 October 2021). In this section, we show code excerpts of this preregistration (formatted in monospace) and explain the rationale behind them.

As usual, the authors state why they are interested in their research question in the "Introduction" section and provide the necessary background information and literature to understand the context and purpose of the research question. This example is drastically shortened for illustration purposes:

```
# Theoretical Background

Machiavellianism describes a personality dimension characterized by a
cynical disregard of morals in the pursuit of one's own interest, e.g.
through manipulation [@christie1970]. There is extensive literature reporting
differences in the dark triad (narcissism, machiavellianism, and psychopathy)
between self-identified males and females [@muris2017] but only few studies
focus solely on machiavellianism. We aim to replicate the finding that males
tend to have higher machiavellianism scores [@muris2017].
```

After researchers have provided the research question, they typically proceed to explain how they want to study it. For simplicity, we will use already published data that we have not yet analyzed:

```
# Method

We report how we determined our sample size, all data exclusions (if any), all
manipulations, and all measures in the study [cf. @simmons2012]. We use data
available from [openpsychometrics.org](https://openpsychometrics.org/_rawdata/)
from the online version of the MACH-IV[@christie1970] and included participants
that have responded to at least one machiavellianism item and reported their
gender as either "male" or "female".
```

We choose the following statistical procedure because many researchers are familiar with it (The *t*-test and Mann-Whitney-Wilcoxon test are arguably the most often used hypothesis tests (according to [56,57] reports that 26% of all studies employed a *t*-test and 27% employed a rank-based alternative in the *New England Journal of Medicine* in 2005). The analytical strategy presented here is, in fact, suboptimal in several respects (the assumption of measurement invariance is untested [58], the effect size is underestimated in the presence of measurement error [59], the effect size is overestimated for highly skewed distributions [60]). The interested reader can use the provided code for the simulation (https://github.com/aaronpeikert/repro-tutorial/blob/main/R/simulation.R accessed on 4 December 2021) to verify that the *t*-test provides unbiased effect sizes but the Mann-Whitney-Wilcoxon overestimates effect sizes with increasing sample size and skewness):

```
We conduct a Student's t-test [@studentProbableErrorMean1908] with Welch's
correction [@welchGeneralizationStudentProblem1947] of the average of
machiavellianism items between the binary-coded gender groups. If the skew of
this average is greater than 1.0 we conduct a supposedly more robust Mann--
Whitney--Wilcoxon test [@Wilcoxon1945] instead.
```

The methods section is the translation of the following `planned_analysis()` function:

```r
planned_analysis <- function(data, use_rank = "skew", skew_cutoff = 1){
# average over all variable supplied, except gender
  machiavellianism <- rowMeans(data["gender" != names(data)], na.rm = TRUE)
# discard rows that only contain NAs
  data <- data[!is.na(machiavellianism),]
  machiavellianism <- machiavellianism[!is.na(machiavellianism)]
# assure gender is factor
  gender <- as.factor(data$gender)
# note skewness and decide t.test vs wilcox based on it
  skew <- moments::skewness(machiavellianism)
# skewness cutoff
if(use_rank == "skew")use_rank <- abs(skew) > skew_cutoff
if(use_rank){
# t.test + rank = wilcox test
    machiavellianism <- rank(machiavellianism)
  }
  test <- t.test(machiavellianism ~ gender)
# return a bunch of information
list(test = test, skew = skew, use_rank = use_rank, n = length(gender))
}
```

This function illustrates two advantages of PAC. First, a PAC can easily include data-dependent decisions by creating different analysis branches under different conditions. Second, it highlights how difficult it is to describe a statistical analysis precisely. The same verbal descriptions may be implemented differently by different persons depending on their statistical and programming knowledge and assumptions. One example would be using the function `wilcox.test` instead of the combinations of the functions `rank` and `t.test`. Either of them is a valid implementation of the Mann–Whitney–Wilcoxon test, but the first assumes equal variance. In contrast, the second applies Welch's correction by default and hence is robust even with unequal variances across groups [61]. Mentioning every such minute implementation detail is almost impossible and would result in overly verbose preregistrations. Still, these details can make a difference in the interpretation of statistical results and, thus, represent undisclosed researchers' degrees of freedom.

Together with the function `simulate_data()` (not shown here), the function `planned_analysis()` can be used to justify the planned sample size. To that end, `simulate_data()` is repeatedly called with increased sample sizes and the proportion of significant results (power) is recorded. The results for such a Monte Carlo simulation for this example are visualized in Figure 3. The code for this power analysis can be found in the online Supplementary Material (https://github.com/aaronpeikert/repro-tutorial/blob/main/R/simulation.R accessed on 11 October 2021). The next snippet shows how we integrated the results dynamically into the preregistration (the origin of the R-variables `minn`, `choosen_power`, and `choosen_d` is not shown).

```
A simulation we conducted indicated that with a sample size of `r minn` for
an alpha of .05 (two-sided) we achieve at least `r choosen_power*100`% power
assuming a standardized effect size of d=`r choosen_d`.
```

Monte Carlo simulations are, of course, not only applicable for this analysis method and also allow researchers to investigate further relevant properties of their analysis method beyond power [62–64].

We implemented a mechanism that only uses simulated data when the actual data are not yet available (in this example, if the file `data/data.csv` does not exist) for the results section. This mechanism also warns readers if these results are based on simulated data. The warning is colored red to avoid any confusion between mock and actual results. As soon as the actual data are available, the simulated data are no longer used, and the results represent the actual empirical results of the study.

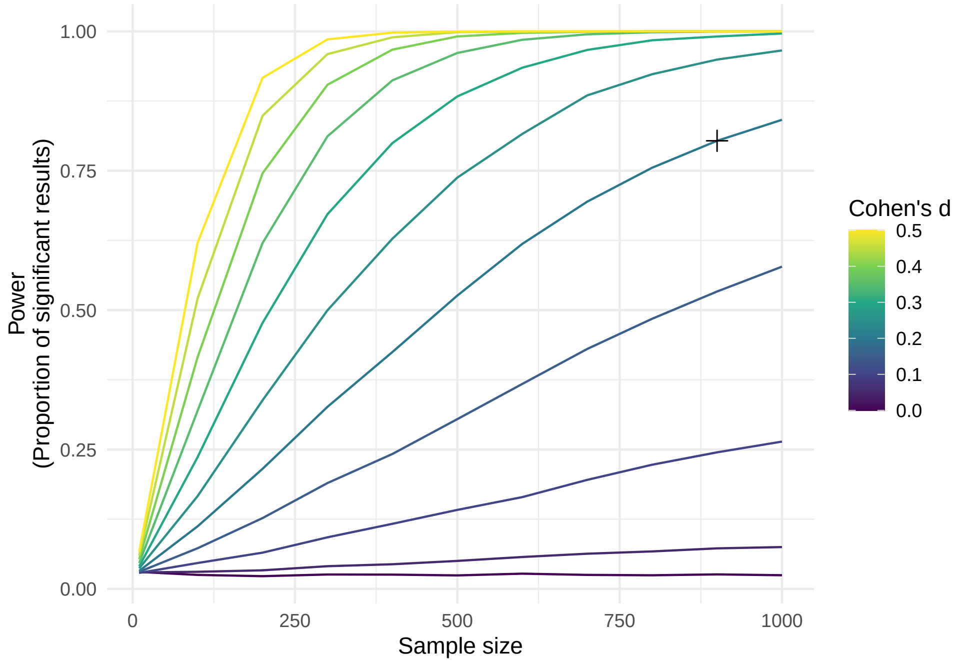


Figure 3. Results of simulation for the power analysis. The cross indicates the sample size that archives 80% assuming a Cohen's d of 0.2.

```
# Results

```{r, echo=FALSE, results='asis', warning=FALSE, message=FALSE}
real_data <- here::here("data", "data.csv")
simulated <- !fs::file_exists(real_data)
if(simulated){
  cat("\\textcolor{red}{The results are based on simulated data and must not be
      interpreted. They only serve to illustrate the result of the preregistered
      code.}")
  set.seed(1234)
  mach <- simulate_data(900, 8, 0.3, 10)
} else {
  mach <- readr::read_delim(real_data, delim = "\t", na = c("", "NA", "NULL"))
  # only keep MACH items + gender
  mach <- dplyr::select(mach, dplyr::matches("^Q\\d+A$"), gender)
  # code gender according to codebook (3 would be other)
  mach <-
    dplyr::mutate(mach, gender = factor(
      gender,
      levels = 1:2,
      labels = c("male", "female")
    ))
  # some items are reversed, see https://core.ac.uk/download/pdf/38810542.pdf
  reversed_nr <- c(1, 15, 2, 12, 4, 11, 14, 19)
  reversed <- stringr::str_c("Q", reversed_nr, "A")
  mach <- dplyr::mutate(mach, dplyr::across(one_of(reversed), ~ 6 - .x))
}
```
```

Following the recommendations outlined in this paper, we did not access the data when we initially wrote this code. We therefore did not know the exact format the data would have. This means that we did need to change our preregistration after accessing the data to include i.e., the recoding of gender (lines 17–22) and the items (lines 23–26). We invite the reader to evaluate the changes we made to the preregistered code. Either on GitHub (https://github.com/aaronpeikert/repro-tutorial/compare/v0.0.1.1-prereg..main accessed on 11 October 2021) → "Files changed" or directly in Git with `git diff v0.0.1.1-prereg preregistration.Rmd`.) This is our summary of what we changed:

```
# Discussion

This document only serves to illustrate Preregistration as Code. We, therefore,
do not discuss the results. After we have acquired the data, we realized that
we had to change the code for reading the data, including recoding gender,
missing values and reversed items (see commit [6556a93](https://github.com/
aaronpeikert/repro-tutorial/commit/6556a9395fcdd600b5b0c5358f92a2c6635ae360)
and commit [9f7ab21](https://github.com/aaronpeikert/repro-tutorial/commit/
9f7ab212dfaf84a0398752a4b80cf14c71000d00)). We do not believe that these changes
influence the results substantively.
```

Readers can inspect and judge the changes for themselves on GitHub (https://github.com/aaronpeikert/repro-tutorial/compare/v0.0.1.1-prereg..main#diff-e21a8fa2e44b297dfefef329a6ef56d283488d467c4b4ffe2a014111e52a170b accessed on 4 December 2021).

The last thing we need to preregister is the reporting of our results with the combination of the functions `planned_analysis()` and `report_analysis()`.

```
```{r, echo=FALSE, results='asis'}
report_analysis(planned_analysis(mach))
```
```

This is an example of how the results could be reported (based on simulated data):

```
report_analysis(planned_analysis(simulate_data(900, 8, 0.3, 10)))
```

```
The Welch Two Sample t-test testing the difference of machiavellianism by gender
(mean in group male = 0.96, mean in group female = 0.79) suggests that the
effect is - negative, statistically significant, and small (difference = -0.17,
95% CI [0.12, 0.22], t(887.46) = 6.38, p < .001; Cohen's d = 0.43, 95% CI [0.30,
0.56])
```

This example of a preregistration covers a single study with a single hypothesis. To organize studies with multiple hypotheses, we suggest multiple `planned_analysis()` and `report_analysis()` functions (possibly numbered in accordance with the hypotheses, e.g., 1.2, 2.3 etc.). Preregistrations that cover multiple distinct data sources may employ multiple `simulate_data()` functions. These are merely suggestions, and researchers are encouraged to find their own way of how to best organize their analysis code.

The example rendered as a PDF file with real data (https://github.com/aaronpeikert/repro-tutorial/files/7309455/preregistration.pdf accessed on 11 October 2021) is available in the online Supplementary Material (https://github.com/aaronpeikert/repro-tutorial/releases/tag/v0.0.3.1-results accessed on 11 October 2021). The changes we made since preregistering it can be inspected on this GitHub page (https://github.com/aaronpeikert/repro-tutorial/compare/v0.0.1.1-prereg..main#diff-e21a8fa2e44b297dfefef329a6ef56d283488d467c4b4ffe2a014111e52a170b accessed on 11 October 2021).

6. Discussion

Increased automation is increasingly recognized as a means to improve the research process [65], and therefore this workflow fits nicely together with other innovations that employ automation, like machine-readable hypothesis tests [66] or automated data documentation [67]. Automated research projects promise a wide range of applications, among them PAC ([68,69] potentially to be submitted as a registered report), direct replication [70], fully automated living metanalysis [71], executable research articles [72], and other innovations such as the live analysis of born open data [73,74].

Central to these innovations is a property we call "reusability", fully promoted by the present workflow. Reusable code can run on different inputs from a similar context and produce valid outputs. This property is based on reproducibility but requires the researcher to more carefully write the software [75] such that it is *built-for-reuse* [76]. The reproducible workflow we present here is heavily automated and hence promotes reusability. Furthermore, adhering to principles of reusability typically removes errors in the code and thus increases the likelihood that the statistical analysis is correct. Therefore reproducibility facilitates traditional good scientific practices and provides the foundation for promising innovations.

6.1. Summary

This paper demonstrated how the R package `repro` supports researchers in creating reproducible research projects, including reproducible manuscripts. These are important building blocks for transparent and cumulative science because they enable others to reproduce statistical and computational results and reports later in time and on different computers. The workflow we present here rests on four software solutions, (1) version control, (2) dynamic document generation, (3) dependency tracking, and (4) software management to guarantee reproducibility. We first demonstrated how to create a reproducible research project. Then, we illustrated how such a project could be reproduced—either by the original author and/or collaborators or by a third party.

We finally presented an example of how the rigorous and automated reproducibility workflow introduced by `repro` may enable other innovations, such as Preregistration as Code (PAC). In PAC the entire reproducible manuscript, including planned analyses and results based on simulated data, is preregistered. This way, every use of a researchers' degree of freedom is disclosed. Once real data is gathered, the reproducible manuscript is (re-)created with the real data. PAC only becomes possible because reproducibility is ensured and leverages version control and dynamic document generation as key features of the workflow.

6.2. Limitations

We realize that the workflow outlined in this paper, and its application in PAC, remains challenging despite our efforts to simplify the procedure by means of the `repro` package. This paper should be considered as a starting point for those seeking to improve the reproducibility of their research. Two kinds of limitations can be distinguished. The first kind are limitations by design, which are unlikely to change. Our workflow inherits these from the software it relies on and the fundamental design principles these share with the workflow and `repro`. The second kind are limitations in `repro` and its dependencies that may be overcome by our future efforts and those of the open-source development community.

With regard to limitations by design, the workflow outlined in this paper is fundamentally incompatible with steps that cannot be automated. This principle may be at odds with some ingrained habits of researchers to mix and match manual and automated steps in data analysis. To allow for automation, many researchers will have to search for alternative software.

The automation-friendly software we present here has several technical but critical limitations. For example, `Git` can track any filetype, but tracked changes are only meaningful for text files (with endings like, `.txt`, `.csv`, `.R`, `.py`, or `.Rmd`), not for binary files (with endings like `.docx`, `.exe`, or `.zip`). Furthermore, tables and graphics dynamically generated from code are difficult to edit by hand. `Make` can automate any programmable software, but not software that is exclusively controlled through a point-and-click user interface. Finally, `Docker` can ship software that runs on Linux and can be automatically installed, which precludes much commercial or closed-source software.

This move away from software that has served researchers well for decades is understandably difficult and presents us with a conundrum. On the one hand, we firmly believe that automated reproducibility makes research more productive and collaboration easier. But, on the other hand, we expect researchers to invest considerable time in learning new tools and to persuade their collaborators to do the same. Three arguments reconcile this apparent paradox. First, this change will not happen all at once. Automated reproducibility is an ideal that we believe has many advantages, but it is not an all-or-nothing decision. Researchers can pick up one skill at a time and then help their fellow collaborators to do the same. Second, the upfront investment is required once (and efforts such as `repro` are underway to reduce it) and will pay dividends over many research projects. Third, the move towards open software for research offers several benefits beyond enabling automated reproducibility [77–80].

With regard to surmountable limitations, we acknowledge that the `repro` package is still in development. One limitation is that `repro` relies on several software dependencies, which represents a threat to long-term reproducibility in itself. For example, to benefit from automatic and convenient reproduction, researchers must use `Git`, `Make`, and `Docker`. However, `Git` and `Make` are themselves included in the `Docker` image created by `repro`. Researchers can therefore employ the `Docker` image manually to download the `Git` repository and execute `Make` for full reproduction. In other words, the only hard requirement for reproduction and therefore its Achilles' heel, is `Docker`. The `Docker` approach has two vulnerabilities. First, and more importantly, the `Docker` image for the project and the `Git` repository have to remain available. The `Dockerfile` (the plain text description to build a `Docker` image), as opposed to the image, is insufficient because it relies on too many service providers (e.g., Microsoft R Application Network, Ubuntu Package Archive). To overcome this limitation, we recommend archiving the `Git` repository and the `Docker` image with zenodo.org, a non-profit long-term storage for scientific data. The necessary steps for archival on zenodo.org are described at the end of Section [Preregistration as Code—a Tutorial].

The second vulnerability is that even if the existence of the `Docker` image and `Git` repository is guaranteed, future researchers still require software to run the image. To that end, they can either rely on `Docker` itself or `Docker`-compatible alternatives (e.g., `CoreOS rkt`, `Mesos Containerizer`, `Singularity`). The only way to remove the reliance on such external software is to turn the `Docker` image into a full operating system that subsequently can be installed and run on almost any modern computer. This process is technically possible and would guarantee reproducibility for decades without any software dependency, assuming hardware that conforms to the x86 instruction set architecture continues to be available. However, this process requires much technical knowledge and is currently not facilitated by `repro`. With regard to this vulnerability, it is worthwhile to note that the `R Markdown`, `Makefile`, and `Dockerfile` do provide information that allows researchers to trace the computational steps and recreate the computational environment manually. The `Makefile`, for example, is written in a way that researchers can manually trace the dependencies and execute commands in the right order, in case they are unable to run `Make` for some reason. Thus, hypothetically, even if `Docker` were to become unavailable one day, the `Dockerfile` still serves as unambiguous documentation of how the original system was set up, and may help future users to create a software environment that closely resembles the original.

6.3. Outlook

Open science practices are a continually evolving field where technical innovations foster changes in research practice. Open data are much more widespread today thanks to online storage facilities; preregistration is possible because there are preregistration platforms and so forth. Similarly, we hope that fully automatic reproduction, e.g., with `repro` as a technical innovation, will promote increased scientific rigor, efficiency, and productivity.

In practice, this ideal of a fully automatic reproduction of research projects can conflict with the wide range of demands for more user-friendly and powerful software. Some may find that `Make` is too complicated or that `Docker` requires too much storage space. Yet others may find that they require other programming languages or want to scale their computation across hundreds of computers, e.g., via high-performance computing clusters or cloud computing.

`repro` was designed modularly to meet many such demands. At the moment, `repro` only supports the combination of `R Markdown`, `Git`, `Make`, and `Docker`. However, there are alternatives for each of these elements that may fit better into an individual research project. `R Markdown` could be complemented or replaced by a dynamic Microsoft Word document with the help of `officer` [81] or `officedown` [82] to accommodate a wider range of journal submission standards. Instead of using formal version control with `Git`, `repro` could automatically save snapshots for increasing user-friendliness. `Make` could be replaced by the more R-centered alternative `targets` for more convenience. `Docker` could be combined with `renv` [22] to control the package versions precisely (our approach fixes the date, `renv` the exact package version). Alternatively, `Docker` could be replaced by the more lightweight `renv` if no dependencies outside of R are considered crucial. `Docker` does not satisfy the requirements of many HPC environments, but `Singularity` was designed to avoid this limitation while still being compatible with `Docker` images.

`repro`'s modular structure allows such alternative workflows, though they have not yet been implemented. Depending on the demand by users, we will implement some of them in `repro` and hope for broad adoption of computational reproducibility in the near future.

Supplementary Materials: All materials (i.e., the source code, all figures, and the data) that are neccesary for reproducing the submitted version of this article are availible at https://github.com/aaronpeikert/repro-tutorial and archived under https://doi.org/10.5281/zenodo.5724454 (accessed on 4 December 2021).

Author Contributions: A.P. took the lead in writing and provided the initial draft of the manuscript. A.M.B. and C.J.v.L. contributed further ideas, critical feedback, and revisions of the original manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: We would like to thank Maximilian Stefan Ernst for his contributions to the code for the simulation study. We are grateful to Julia Delius for her helpful assistance in language and style editing. The R package `repro` was developed as part of the first author's master thesis (https://aaronpeikert.github.io/repro-thesis/ accessed on 11 October 2021) at the Humboldt-Universität zu Berlin.

Conflicts of Interest: The authors declare no conflicts of interest. We have received no financial support for the research, authorship, and/or publication of this article.

Abbreviations

The following abbreviations are used in this manuscript:

PAC	Preregistration as Code
Gb	Gigabyte
Kb	Kilobyte
CRAN	Comprehensive R Archive Network
WORCS	Workflow for Open Reproducible Code in Science

References

1. Peikert, A.; Brandmaier, A.M. A Reproducible Data Analysis Workflow with R Markdown, Git, Make, and Docker. *Quant. Comput. Methods Behav. Sci.* **2021**, *1*, e3763. [CrossRef]
2. Popper, K.R. *The Logic of Scientific Discovery*; Routledge: London, UK, 2002.
3. Obels, P.; Lakens, D.; Coles, N.A.; Gottfried, J.; Green, S.A. Analysis of Open Data and Computational Reproducibility in Registered Reports in Psychology. *Adv. Methods Pract. Psychol. Sci.* **2020**, *3*, 229–237. [CrossRef]
4. van Lissa, C.J.; Brandmaier, A.M.; Brinkman, L.; Lamprecht, A.L.; Peikert, A.; Struiksma, M.E.; Vreede, B.M. WORCS: A Workflow for Open Reproducible Code in Science. *Data Sci.* **2021**, *4*, 29–49. [CrossRef]
5. Nosek, B.A.; Ebersole, C.R.; DeHaven, A.C.; Mellor, D.T. The Preregistration Revolution. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 2600–2606. [CrossRef] [PubMed]
6. Hardwicke, T.E.; Mathur, M.B.; MacDonald, K.; Nilsonne, G.; Banks, G.C.; Kidwell, M.C.; Hofelich Mohr, A.; Clayton, E.; Yoon, E.J.; Henry Tessler, M.; et al. Data Availability, Reusability, and Analytic Reproducibility: Evaluating the Impact of a Mandatory Open Data Policy at the Journal Cognition. *R. Soc. Open Sci.* **2018**, *5*, 180448. [CrossRef]
7. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2021.
8. Vuorre, M.; Curley, J.P. Curating Research Assets: A Tutorial on the Git Version Control System. *Adv. Methods Pract. Psychol. Sci.* **2018**, *1*, 219–236. [CrossRef]
9. Bryan, J. Excuse Me, Do You Have a Moment to Talk About Version Control? *Am. Stat.* **2018**, *72*, 20–27. [CrossRef]
10. Nuijten, M.B.; Hartgerink, C.H.J.; van Assen, M.A.L.M.; Epskamp, S.; Wicherts, J.M. The Prevalence of Statistical Reporting Errors in Psychology (1985–2013). *Behav. Res. Methods* **2016**, *48*, 1205–1226. [CrossRef]
11. Knuth, D.E.; Levy, S. *The CWEB System of Structured Documentation*; Addison-Wesley Longman: Boston, MA, USA, 1994.
12. Claerbout, J.F.; Karrenbach, M. Electronic Documents Give Reproducible Research a New Meaning. *SEG Tech. Program Expand. Abstr.* **1992**, 601–604. [CrossRef]
13. Lamport, L. *LATEX: A Document Preparation System: User's Guide and Reference Manual*, 2nd ed.; Addison-Wesley: Reading, MA, USA, 1994.
14. Allaire, J.; Xie, Y.; Foundation, R.; Wickham, H.; Journal of Statistical Software; Vaidyanathan, R.; Association for Computing Machinery; Boettiger, C.; Elsevier; Broman, K.; et al. *Articles: Article Formats for R Markdown*; R Package Version 0.19; 2021. Available online: https://pkgs.rstudio.com/rticles/ (accessed on 4 December 2021).
15. El Hattab, H.; Allaire, J. Revealjs: R Markdown Format for 'Reveal, Js' Presentations. 2017. Available online: https://bookdown.org/yihui/rmarkdown/revealjs.html (accessed on 4 December 2021).
16. O'Hara-Wild, M.; Hyndman, R. Vitae: Curriculum Vitae for r Markdown. 2021. Available online: https://cran.r-project.org/web/packages/vitae/vignettes/vitae.html (accessed on 4 December 2021).
17. Xie, Y.; Dervieux, C.; Riederer, E. *R Markdown Cookbook*, 1st ed.; The R Series; Taylor and Francis, CRC Press: Boca Raton, FL, USA, 2020.
18. Silver, A. Software Simplified. *Nature* **2017**, *546*, 173–174. [CrossRef]
19. Boettiger, C.; Eddelbuettel, D. An Introduction to Rocker: Docker Containers for R. *R J.* **2017**, *9*, 527. [CrossRef]
20. Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L.D.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; et al. Welcome to the tidyverse. *J. Open Source Softw.* **2019**, *4*, 1686. [CrossRef]
21. Wiebels, K.; Moreau, D. Leveraging Containers for Reproducible Psychological Research. Advances in Methods and Practices in Psychological. *Science* **2021**, *4*. [CrossRef]
22. Ushey, K. Renv: Project Environments. R Package Version 0.13.2. 2021. Available online: https://rstudio.github.io/renv/articles/renv.html (accessed on 4 December 2021).
23. Wickham, H.; Bryan, J. Usethis: Automate Package and Project Setup. 2021. Available online: https://usethis.r-lib.org (accessed on 4 December 2021).
24. Parasuraman, R.; Mouloua, M. *Automation and Human Performance: Theory and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2019.
25. RStudio Team. *RStudio: Integrated Development Environment for R. RStudio*; PBC: Boston, MA, USA, 2021.
26. Peikert, A.; Brandmaier, A.M.; van Lissa, C.J. Repro: Automated Setup of Reproducible Workflows and Their Dependencies. R Package Version 0.1.0. 2021. Available online: https://github.com/aaronpeikert/repro (accessed on 4 December 2021).
27. Xie, Y.; Allaire, J.J.; Grolemund, G. *R Markdown: The Definitive Guide*; CRC Press: Boca Raton, FL, USA, 2019.
28. Aust, F.; Barth, M. Papaja: Create APA Manuscripts with R Markdown. 2020. Available online: http://frederikaust.com/papaja_man/ (accessed on 4 December 2021).
29. Association, A.P. (Ed.) *Publication Manual of the American Psychological Association*, 7th ed.; American Psychological Association: Washington, DC, USA, 2019.
30. DeCoster, J.; Sparks, E.A.; Sparks, J.C.; Sparks, G.G.; Sparks, C.W. Opportunistic Biases: Their Origins, Effects, and an Integrated Solution. *Am. Psychol.* **2015**, *70*, 499–514. [CrossRef]
31. Silberzahn, R.; Uhlmann, E.L.; Martin, D.P.; Anselmi, P.; Aust, F.; Awtrey, E.; Bahnik, S.; Bai, F.; Bannard, C.; Bonnier, E.; et al. Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Adv. Methods Pract. Psychol. Sci.* **2018**, *1*, 337–356. [CrossRef]

32. Bowman, S.; DeHaven, A.; Errington, T.; Hardwicke, T.E.; Mellor, D.T.; Nosek, B.A.; Soderberg, C.K. OSF Prereg Template 2020. *MetaArXiv* **2020**. [CrossRef]
33. Bakker, M.; Veldkamp, C.L.S.; van Assen, M.A.L.M.; Crompvoets, E.A.V.; Ong, H.H.; Nosek, B.A.; Soderberg, C.K.; Mellor, D.; Wicherts, J.M. Ensuring the Quality and Specificity of Preregistrations. *PLoS Biol.* **2020**, *18*, e3000937. [CrossRef]
34. Bakker, M.; Veldkamp, C.L.S.; van den Akker, O.R.; van Assen, M.A.L.M.; Crompvoets, E.; Ong, H.H.; Wicherts, J.M. Recommendations in Pre-Registrations and Internal Review Board Proposals Promote Formal Power Analyses but Do Not Increase Sample Size. *PLoS ONE* **2020**, *15*, e0236079. [CrossRef]
35. Steegen, S.; Dewitte, L.; Tuerlinckx, F.; Vanpaemel, W. Measuring the Crowd within Again: A Pre-Registered Replication Study. *Front. Psychol.* **2014**, *5*. [CrossRef]
36. Morris, T.P.; White, I.R.; Crowther, M.J. Using Simulation Studies to Evaluate Statistical Methods. *Stat. Med.* **2019**, *38*, 2074–2102. [CrossRef]
37. Paxton, P.; Curran, P.J.; Bollen, K.A.; Kirby, J.; Chen, F. Monte Carlo Experiments: Design and Implementation. *Struct. Equ. Model. Multidiscip. J.* **2001**, *8*, 287–312. [CrossRef]
38. Skrondal, A. Design and Analysis of Monte Carlo Experiments: Attacking the Conventional Wisdom. *Multivar. Behav. Res.* **2000**, *35*, 137–167. [CrossRef] [PubMed]
39. Goldfeld, K.; Wujciak-Jens, J. Simstudy: Illuminating Research Methods through Data Generation. *J. Open Source Softw.* **2020**, *5*, 2763. [CrossRef]
40. Revelle, W. *Psych: Procedures for Psychological, Psychometric, and Personality Research*; Northwestern University: Evanston, IL, USA, 2021.
41. Wicherts, J.M.; Veldkamp, C.L.S.; Augusteijn, H.E.M.; Bakker, M.; van Aert, R.C.M.; van Assen, M.A.L.M. Degrees of Freedom in Planning, Running, Analyzing, and Reporting Psychological Studies: A Checklist to Avoid p-Hacking. *Front. Psychol.* **2016**, *7*, 1832. [CrossRef] [PubMed]
42. Szollosi, A.; Kellen, D.; Navarro, D.J.; Shiffrin, R.; van Rooij, I.; Van Zandt, T.; Donkin, C. Is Preregistration Worthwhile? *Trends Cogn. Sci.* **2020**, *24*, 94–95. [CrossRef] [PubMed]
43. Nosek, B.A.; Beck, E.D.; Campbell, L.; Flake, J.K.; Hardwicke, T.E.; Mellor, D.T.; van 't Veer, A.E.; Vazire, S. Preregistration Is Hard, And Worthwhile. *Trends Cogn. Sci.* **2019**, *23*, 815–818. [CrossRef]
44. Meehl, P.E. Theoretical Risks and Tabular Asterisks: Sir Karl, Sir Ronald, and the Slow Progress of Soft Psychology. *J. Consult. Clin. Psychol.* **1978**, *46*, 806–834. [CrossRef]
45. Brandmaier, A.M.; Jacobucci, R. Machine-Learning Approaches to Structural Equation Modeling. In *Handbook of Structural Equation Modeling*, 2nd ed.; Hoyle, R.H., Ed.; Guilford Press: New York, NY, USA, in press.
46. Hastie, T.; Tibshirani, R.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, Corrected at 12th Printing 2017 ed.*; Springer Series in Statisticsl; Springer: New York, NY, USA, 2017.
47. Daróczi, G.; Tsegelskyi, R. Pander: An R 'pandoc' Writer. 2021. Available online: https://www.r-project.org/nosvn/pandoc/pander.html (accessed on 4 December 2021).
48. Hlavac, M. *Stargazer: Well-Formatted Regression and Summary Statistics Tables*; Central European Labour Studies Institute (CELSI): Bratislava, Slovakia, 2018.
49. Stanley, D. apaTables: Create American Psychological Association (APA) Style Tables, 2021. Available online: https://dstanley4.github.io/apaTables/articles/apaTables.html (accessed on 4 December 2021).
50. Makowski, D.; Ben-Shachar, M.S.; Patil, I.; Lüdecke, D. Automated Results Reporting as a Practical Tool to Improve Reproducibility and Methodological Best Practices Adoption. CRAN 2021. Available online: https://easystats.github.io/report/ (accessed on 4 December 2021).
51. European Organization For Nuclear Research. *OpenAIRE*; Zenodo: online, 2013. [CrossRef]
52. Weston, S.J.; Ritchie, S.J.; Rohrer, J.M.; Przybylski, A.K. Recommendations for Increasing the Transparency of Analysis of Preexisting Data Sets. *Adv. Methods Pract. Psychol. Sci.* **2019**, *2*, 214–227. [CrossRef]
53. International Council for Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use. E 9 Statistical Principles for Clinical Trials. 1998. Available online: https://www.ema.europa.eu/en/ich-e9-statistical-principles-clinical-trials (accessed on 4 December 2021).
54. Thabane, L.; Ma, J.; Chu, R.; Cheng, J.; Ismaila, A.; Rios, L.P.; Robson, R.; Thabane, M.; Giangregorio, L.; Goldsmith, C.H. A Tutorial on Pilot Studies: The What, Why and How. *BMC Med. Res. Methodol.* **2010**, *10*, 1. [CrossRef] [PubMed]
55. Yuan, I.; Topjian, A.A.; Kurth, C.D.; Kirschen, M.P.; Ward, C.G.; Zhang, B.; Mensinger, J.L. Guide to the Statistical Analysis Plan. *Pediatric Anesth.* **2019**, *29*, 237–242. [CrossRef]
56. Fagerland, M.W. T-Tests, Non-Parametric Tests, and Large Studies—A Paradox of Statistical Practice? *BMC Med Res. Methodol.* **2012**, *12*, 78. [CrossRef]
57. Horton, N.J.; Switzer, S.S. Statistical Methods in the Journal. *New Engl. J. Med.* **2005**, *353*, 1977–1979. [CrossRef] [PubMed]
58. Putnick, D.L.; Bornstein, M.H. Measurement Invariance Conventions and Reporting: The State of the Art and Future Directions for Psychological Research. *Dev. Rev.* **2016**, *41*, 71–90. [CrossRef] [PubMed]
59. Frost, C.; Thompson, S.G. Correcting for Regression Dilution Bias: Comparison of Methods for a Single Predictor Variable. *J. R. Stat. Soc. Ser. A* **2000**, *163*, 173–189. [CrossRef]

60. Stonehouse, J.M.; Forrester, G.J. Robustness of the *t* and U Tests under Combined Assumption Violations. *J. Appl. Stat.* **1998**, *25*, 63–74. [CrossRef]
61. Zimmerman, D.W.; Zumbo, B.D. Rank Transformations and the Power of the Student *t* Test and Welch *t* Test for Non-Normal Populations with Unequal Variances. *Can. J. Exp. Psychol./Rev. Can. De Psychol. Exp.* **1993**, *47*, 523–539. [CrossRef]
62. Brandmaier, A.M.; von Oertzen, T.; Ghisletta, P.; Lindenberger, U.; Hertzog, C. Precision, Reliability, and Effect Size of Slope Variance in Latent Growth Curve Models: Implications for Statistical Power Analysis. *Front. Psychol.* **2018**, *9*, 294. [CrossRef]
63. Harrison, R.L. Introduction to Monte Carlo Simulation. *AIP Conf. Proc.* **2010**, *1204*, 17–21. [CrossRef]
64. Raychaudhuri, S. Introduction to Monte Carlo Simulation. In Proceedings of the 2008 Winter Simulation Conference, Miami, FL, USA, 7–10 December 2008; pp. 91–100. [CrossRef]
65. Rouder, J.N.; Haaf, J.M.; Snyder, H.K. Minimizing Mistakes in Psychological Science. *Adv. Methods Pract. Psychol. Sci.* **2019**, *2*, 3–11. [CrossRef]
66. Lakens, D.; DeBruine, L.M. Improving Transparency, Falsifiability, and Rigor by Making Hypothesis Tests Machine-Readable. *Adv. Methods Pract. Psychol. Sci.* **2021**, *4*. [CrossRef]
67. Arslan, R.C. How to Automatically Document Data With the Codebook Package to Facilitate Data Reuse. Advances in Methods and Practices in Psychological. *Science* **2019**, *2*, 169–187. [CrossRef]
68. Nosek, B.A.; Lakens, D. Registered Reports. *Soc. Psychol.* **2014**, *45*, 137–141. [CrossRef]
69. Chambers, C. What's next for Registered Reports? *Nature* **2019**, *573*, 187–189. [CrossRef] [PubMed]
70. Simons, D.J. The Value of Direct Replication. *Perspect. Psychol. Sci.* **2014**, *9*, 76–80. [CrossRef] [PubMed]
71. Elliott, J.H.; Turner, T.; Clavisi, O.; Thomas, J.; Higgins, J.P.T.; Mavergames, C.; Gruen, R.L. Living Systematic Reviews: An Emerging Opportunity to Narrow the Evidence-Practice Gap. *PLoS Med.* **2014**, *11*, e1001603. [CrossRef]
72. eLife Sciences Publications. eLife Launches Executable Research Articles for Publishing Computationally Reproducible Results. 2020. Available online: https://elifesciences.org/for-the-press/eb096af1/elife-launches-executable-research-articles-for-publishing-computationally-reproducible-results (accessed on 4 December 2021).
73. Rouder, J.N. The What, Why, and How of Born-Open Data. *Behav. Res. Methods* **2016**, *48*, 1062–1069. [CrossRef] [PubMed]
74. Kekecs, Z.; Aczel, B.; Palfi, B.; Szaszi, B.; Szecsi, P.; Zrubka, M.; Kovacs, M.; Bakos, B.E.; Cousineau, D.; Tressoldi, P.; et al. Raising the Value of Research Studies in Psychological Science by Increasing the Credibility of Research Reports: The Transparent Psi Project—Preprint. *PsyArXiv* **2020**. [CrossRef]
75. Lanergan, R.G.; Grasso, C.A. Software Engineering with Reusable Designs and Code. In *Software Reusability: Vol. 2, Applications and Experience*; Association for Computing Machinery: New York, NY, USA, 1989; pp. 187–195.
76. Al-Badareen, A.B.; Selamat, M.H.; Jabar, M.A.; Din, J.; Turaev, S. Reusable Software Components Framework. In Proceedings of the European Conference of Systems, and European Conference of Circuits Technology and Devices, and European Conference of Communications, and European Conference on Computer Science, Kuantan, Pahang, Malaysia, 27–29 June 2011; World Scientific and Engineering Academy and Society (WSEAS): Stevens Point, WI, USA, 2010; ECS'10/ECCTD'10/ECCOM'10/ECCS'10, pp. 126–130.
77. Schaffner, A.C. The Future of Scientific Journals: Lessons from the Past. *Inf. Technol. Libr.* **1994**, *13*, 239–247.
78. Fitzgerald, B. The Transformation of Open Source Software. *MIS Q.* **2006**, *30*, 587–598. [CrossRef]
79. Chaldecott, J.A. A History of Scientific and Technical Periodicals: The Origins and Development of the Scientific and Technological Press. *Br. J. Hist. Sci.* **1965**, *2*, 360–361. [CrossRef]
80. Sonnenburg, S.; Braun, M.L.; Ong, C.S.; Bengio, S.; Bottou, L.; Holmes, G.; LeCun, Y.; Müller, K.R.; Pereira, F.; Rasmussen, C.E.; et al. The Need for Open Source Software in Machine Learning. *J. Mach. Learn. Res.* **2007**, *8*, 2443–2466.
81. Gohel, D. Officer: Manipulation of Microsoft Word and PowerPoint Documents. 2021. Available online: https://davidgohel.github.io/officer/ (accessed on 4 December 2021).
82. Gohel, D.; Ross, N. Officedown: Enhanced 'R Markdown' Format for 'Word' and 'PowerPoint'. 2021. Available online: https://davidgohel.github.io/officedown/ (accessed on 4 December 2021).

Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization

The following article is prepared to be submitted to *Philosophical Psychology*. The following preprint is reprinted from the following source under the conditions of the Creative Commons Public Domain Dedication (CC0 1.0).

Peikert, A., & Brandmaier, A. M. (2023). *Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization.* PsyArXiv. https://doi.org/10.31234/osf.io/cs8wb

Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization

Aaron Peikert[1,2,3] & Andreas M. Brandmaier[1, 2, 4]

[1] Max Planck Institute for Human Development

[2] Max Planck UCL Centre for Computational Psychiatry and Ageing Research

[3] Humboldt-Universität zu Berlin

[4] MSB Medical School Berlin

The materials for this article are available on GitHub (Peikert & Brandmaier, 2023a). This version was created from git commit `1278017`. The manuscript is available as preprint (Peikert & Brandmaier, 2023b) and was submitted to Psychological Methods but has not been peer reviewed.

Author Note

Correspondence concerning this article should be addressed to Aaron Peikert, Center for Lifespan Psychology, Max Planck Institute for Human Development, Lentzeallee 94, 14195 Berlin, Germany. E-mail: peikert@mpib-berlin.mpg.de

Abstract

The replication crisis has led many researchers to preregister their hypotheses and data analysis plans before collecting data. A widely held view is that preregistration is supposed to limit the extent to which data may influence the hypotheses to be tested. Only if data have no influence an analysis is considered confirmatory. Consequently, many researchers believe that preregistration is only applicable in confirmatory paradigms. In practice, researchers may struggle to preregister their hypotheses because of vague theories that necessitate data-dependent decisions (aka exploration). We argue that preregistration benefits any study on the continuum between confirmatory and exploratory research. To that end, we formalize a general objective of preregistration and demonstrate that exploratory studies also benefit from preregistration. Drawing on Bayesian philosophy of science, we argue that preregistration should primarily aim to reduce uncertainty about the inferential procedure used to derive results. This approach provides a principled justification of preregistration, separating the procedure from the goal of ensuring strictly confirmatory research. We acknowledge that knowing the extent to which a study is exploratory is central, but certainty about the inferential procedure is a prerequisite for persuasive evidence. Finally, we discuss the implications of these insights for the practice of preregistration.

Keywords: preregistration; confirmation; exploration; hypothesis testing; Bayesian; Open Science

Word count: 7000

Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization

The scientific community has long pondered the vital distinction between exploration and confirmation, discovery and justification, hypothesis generation and hypothesis testing, or prediction and postdiction (Hoyningen-Huene, 2006; Nosek et al., 2018; Shmueli, 2010). Despite the different names, it is fundamentally the same dichotomy that is at stake here. There is a broad consensus that both approaches are necessary for science to progress; exploration, to make new discoveries and confirmation, to expose these discoveries to potential falsification, and assess empirical support for the theory. However, mistaking exploratory findings for empirically confirmed results is dangerous. It inflates the likelihood of believing that there is evidence supporting a given hypothesis, even if it is false. A variety of problems, such as researchers' degrees of freedom together with researchers' hindsight bias or naive p-hacking have led to such mistakes becoming commonplace yet unnoticed for a long time. Recognizing them has led to a crisis of confidence in the empirical sciences (Ioannidis, 2005), and psychology in particular (Open Science Collaboration, 2015). As a response to the crisis, evermore researchers preregister their hypotheses and their data collection and analysis plans in advance of their studies (Nosek et al., 2018). They do so to stress the predictive nature of their registered statistical analyses, often with the hopes of obtaining a label that marks the study as "confirmatory". Indeed, rigorous application of preregistration prevents researchers from reporting a set of results produced by an arduous process of trial and error as a simple confirmatory story (Wagenmakers et al., 2012) while keeping low false-positive rates. This promise of a clear distinction between confirmation and exploration has obvious appeal to many who have already accepted the practice. Still, the majority of empirical researchers do not routinely preregister their studies. One reason may be that some do not find that the theoretical advantages outweigh the practical hurdles, such as specifying every aspect of a theory and the corresponding analysis in advance. We believe that we can reach a greater acceptance of preregistration by explicating a more general objective of preregistration that benefits all kinds of studies, even those that allow data-dependent decisions.

One goal of preregistration that has received widespread attention is to clearly distinguish confirmatory from exploratory research (Bakker et al., 2020; Mellor & Nosek, 2018; Nosek et al., 2018; Simmons et al., 2021; Wagenmakers et al., 2012). In such a narrative, preregistration is justified by a confirmatory research agenda. However, two problems become apparent under closer inspection. First, many researchers do not subscribe to a purely confirmatory research agenda. Second, there is strict mapping of the categories preregistered vs. non-preregistered onto the categories confirmatory vs. exploratory research.

Obviously, researchers can conduct confirmatory research without preregistration — though it might be difficult to convince other researchers of the confirmatory nature of their research, that is, that they were free of cognitive biases, made no data-dependent decisions, and so forth. The opposite, that is, preregistered but not strictly confirmatory studies, are also becoming more commonplace (Chan et al., 2004; Dwan et al., 2008; Silagy et al., 2002).

This is the result of researchers applying one of two strategies to evade the self-imposed restrictions of preregistrations: writing a loose preregistration, to begin with (Stefan & Schönbrodt, 2023) or deviating from the preregistration afterward. Both strategies may be used for sensible scientific reasons or with the self-serving intent of generating desirable results. Thus, insisting on equating preregistration and confirmation has led to the criticism that, all things considered, preregistration is actually harmful and neither sufficient nor necessary for doing good science (Pham & Oh, 2021; Szollosi et al., 2020).

We argue that such criticism is not directed against preregistration itself but against a justification through a confirmatory research agenda (Wagenmakers et al., 2012). When researchers criticize preregistration as being too inflexible to fit their research question, they often simply acknowledge that their research goals are not strictly confirmatory. Forcing researchers into adopting a strictly confirmatory research agenda does not only imply changing *how* they investigate a phenomenon but also *what* research questions they pose. However reasonable such a move is, changing the core beliefs of a large community is much harder than convincing them that a method is well justified. We, therefore, attempt to disentangle the *methodological* goals of preregistration from the *ideological* goals of confirmatory science. It might well be the case that psychology needs more confirmatory studies to progress as a science. However, independently of such a goal, preregistration can be useful for any kind of study on the continuum between strictly confirmatory and fully exploratory.

To form such an objective for preregistration, we first introduce some tools of Bayesian philosophy of science and map the exploration/confirmation distinction onto a dimensional quantity we call "theoretical risk" (a term borrowed from Meehl, 1978, but formalized as the probability of proving a hypothesis wrong if it does not hold), which is inversely related to the type-I error rate in null hypothesis testing.

Further, we outline two interpretations of preregistration. The first one corresponds to the traditional application of preregistration to research paradigms that focus on confirmation by maximizing the theoretical risk or, equivalently, by limiting type-I error (when dichotomous decisions about theories are an inferential goal). We argue that this view on the utility of preregistration can be interpreted as maximizing theoretical risk, which is reduced by researchers' degrees of freedom, p-hacking, and suchlike. The second interpretation is our main contribution: We argue that contrary to the classic view, the objective of preregistration is *not* the maximization of theoretical risk but rather the minimization of uncertainty about the theoretical risk. This interpretation leads to a broad applicability of preregistration to both exploratory and confirmatory studies.

To arrive at this interpretation, we rely on three arguments. The first is that theoretical risk is vital for judging evidential support for theories. The second argument is that the theoretical risk for a given study is generally uncertain. The third and last argument is that this uncertainty is reduced by applying preregistration. We conclude that because preregistration decreases uncertainty about the theoretical risk, which in turn increases the amount of knowledge we gain from a particular study, preregistration is potentially useful for any kind of study, no matter how exploratory.

Epistemic value and the Bayesian rationale

Let us start by defining what we call expected epistemic value. If researchers plan to conduct a study, they usually hope that it will change their assessment of some theory's verisimilitude (Niiniluoto, 1998). In other words, they hope to learn something from conducting the study. The amount of knowledge researchers gain from a particular study concerning the verisimilitude of a specific theory is what we call epistemic value. Researchers cannot know what exactly they will learn from a study before they run it. However, they can develop an expectation that helps them decide about the specifics of a planned study. This expectation is what we term expected epistemic value. To make our three arguments, we must assume three things about what an ideal estimation process entails and how it relates to what studies (preregistered vs not preregistered) to conduct.

1. Researchers judge the evidence for or against a hypothesis rationally.
2. They expect other researchers to apply a similar rational process.
3. Researchers try to maximize the expected epistemic value for other researchers.

The assumption of rationality can be connected to Bayesian reasoning and leads to our adoption of the framework. Our rationale is as follows. Researchers who decide to conduct a certain study are actually choosing a study to bet on. They have to “place the bet" by conducting the study by investing resources and stand to gain epistemic value with some probability. This conceptualization of choosing a study as a betting problem allows us to apply a "Dutch book" argument (Christensen, 1991). This argument states that any better must follow the axioms of probability to avoid being "irrational," i.e., accepting bets that lead to sure losses. Fully developing a Dutch book argument for this problem requires careful consideration of what kind of studies to include as possible bets, defining a conversion rate from the stakes to the reward, and modeling what liberties researchers have in what studies to conduct. Without deliberating these concepts further, we find it persuasive that researchers should not violate the axioms of probability if they have some expectation about what they stand to gain with some likelihood from conducting a study. The axioms of probability are sufficient to derive the Bayes formula, on which we will heavily rely for our further arguments. The argument is not sufficient, however, to warrant conceptualizing the kind of epistemic value we reason about in terms of posterior probability; that remains a leap of faith. However, the argument applies to any reward function that satisfies the "statistical relevancy condition" (Fetzer, 1974; Salmon, 1970). That is, evidence only increases epistemic value for a theory if the evidence is more likely to be observed under the theory than under the alternative.

Please note that our decision to adopt this aspect of the Bayesian philosophy of science does not imply anything about the statistical methods researchers use. In fact, this conceptualization is intentionally as minimal as possible to be compatible with a wide range of philosophies of science and statistical methods researchers might subscribe to.

Epistemic value and theoretical risk

Our first argument is that theoretical risk is crucial for judging evidential support for theories. Put simply, risky predictions create persuasive evidence if they turn out to be correct. This point is crucial because we attribute much of the appeal of a confirmatory research agenda to this notion.

Let us make some simplifying assumptions and define our notation. To keep the notation simple, we restrict ourselves to evidence of a binary nature (either it was observed or not). We denote the probability of a hypothesis before observing evidence as $P(H)$ and its complement as $P(\neg H) = 1 - P(H)$. The probability of observing evidence under some hypothesis is $P(E|H)$. We can calculate the probability of the hypothesis after observing the evidence with the help of the Bayes formula:

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)} \tag{1}$$

The posterior probability $P(H|E)$ is of great relevance since it is often used directly or indirectly as a measure of confirmation of a hypothesis. In the tradition of Carnap, in its direct use, it is called confirmation as firmness; in its relation to the a priori probability $P(H)$, it is called increase in firmness (Carnap, 1950, preface to the 1962 edition). As noted before, we concentrate on posterior probability as a measure of epistemic value since no measure shows universally better properties than others. However, it is reasonable that any measure of confirmation increases monotonically with an increase in posterior probability $P(H|E)$, and our argument applies to those measures as well.

In short, we want to increase posterior probability $P(H|E)$. Increases in posterior probability $P(H|E)$ are associated with increased epistemic value, of which we want to maximize the expectation. So how can we increase posterior probability? The Bayes formula yields three components that influence confirmation, namely $P(H)$, $P(E|H)$ and $P(E)$. The first option leads us to the unsurprising conclusion that higher a priori probability $P(H)$ leads to higher posterior probability $P(H|E)$. If a hypothesis is more probable to begin with, observing evidence in its favor will result in a hypothesis that is more strongly confirmed, all else being equal. However, the prior probability of a hypothesis is nothing our study design can change. The second option is similarly commonsensical; that is, an increase in $P(E|H)$ leads to a higher posterior probability $P(H|E)$. $P(E|H)$ is the probability of obtaining evidence for a hypothesis when it holds. We call this probability of detecting evidence, given that the hypothesis holds "detectability." Consequently, researchers should ensure that their study design allows them to find evidence for their hypothesis, in case it is true. When applied strictly within the bounds of null hypothesis testing, detectability is equivalent to power (or the complement of type-II error rate). However, while detectability is of great importance for study design, it is not directly relevant to the objective of preregistration. Thus, $P(E)$ remains to be considered. Since $P(E)$ is the denominator, decreasing it can increase the posterior probability. In other words, high risk, high reward.

If we equate riskiness with a low probability of obtaining evidence (when the hypothesis is false), the Bayesian rationale perfectly aligns with the observation that risky predictions lead to persuasive evidence. This tension between high risk leading to high gain is central to our consideration of preregistration. A high-risk, high-gain strategy is bound to result in many losses that are eventually absorbed by the high gains. Sustaining many "failed" studies is not exactly aligned with the incentive structure under which many, if not most, researchers operate. Consequently, researchers are incentivized to appear to take more risks than they actually do, which misleads their readers to give their claims more credence than they deserve. It is at this juncture that the practice and mispractice of preregistration comes into play. We argue that the main function of preregistration is to enable proper judgment of the riskiness of a study.

To better understand how preregistrations can achieve that, let us take a closer look at the factors contributing to $P(E)$. Using the law of total probability, we can split $P(E)$ into two terms:

$$P(E) = P(H)P(E|H) + P(\neg H)P(E|\neg H) \tag{2}$$

We have already noted that there is not much to be done about prior probability ($P(H)$, and hence its counter probability $P(\neg H)$), and that it is common sense to increase detectability $P(E|H)$. The real lever to pull is therefore $P(E|\neg H)$. This probability tells us how likely it is that we find evidence in favor of the theory when in fact, the theory is not true. Its counter probability $P(\neg E|\neg H) = 1 - P(E|\neg H)$ is what we call "theoretical risk", because it is the risk a theory takes on in predicting the occurrence of particular evidence in its favor. We "borrow" the term from Meehl (1978), though he has not assigned it to the probability $P(\neg E|\neg H)$. Kukla (1990) argued that the core arguments in Meehl (1990) can be reconstructed in a purely Bayesian framework. However, while he did not mention $P(\neg E|\neg H)$ he suggested that Meehl (1978) used the term "very strange coincidence" for a small $P(E|\neg H)$ which would imply, that $P(\neg E|\neg H)$ can be related to or even equated to theoretical risk.

Let us note some interesting properties of theoretical risk $P(\neg E|\neg H)$. First, increasing theoretical risk leads to higher posterior probability $P(H|E)$, our objective. Second, if the theoretical risk is smaller than detectability $P(E|H)$ it follows that the posterior probability must decrease when observing the evidence. If detectability exceeds theoretical risk, the evidence is less likely under the theory than it is when the theory does not hold. Third, if the theoretical risk equals zero, then posterior probability is at best equal to prior probability but only if detectability is perfect ($P(H|E) = 1$). In other words, observing a sure fact does not lend credence to a hypothesis.

The last statement sounds like a truism but is directly related to Popper's seminal criterion of demarcation. He stated that if it is impossible to prove that a hypothesis is false ($P(\neg E|\neg H) = 0$, theoretical risk is zero), it cannot be considered a scientific hypothesis (Popper, 2002, p. 18). We note these relations to underline that the Bayesian rationale we apply here is able to reconstruct many commonly held views on riskiness and epistemic value.

Both theoretical risk $P(\neg E|\neg H)$ and detectability $P(E|H)$ aggregate countless influences; otherwise, they could not model the process of evidential support for theories. To illustrate the concepts we have introduced here, consider the following example of a single theory and three experiments that may test it. The experiments were created to illustrate how they may differ in their theoretical risk and detectability. Suppose the primary theory is about the cognitive phenomenon of "insight." For the purpose of illustration, we define it, with quite some hand-waving, as a cognitive abstraction that allows agents to consistently solve a well-defined class of problems. We present the hypothesis that the following problem belongs to such a class of insight problems:

> Use five matches (IIIII) to form the number eight.

We propose three experiments that differ in theoretical risk and detectability. All experiments take a sample of ten psychology students. We present the students with the problem for a brief span of time. After that, the three experiments differ as follows:

1. The experimenter gives a hint that the problem is easy to solve when using Roman numerals; if all students come up with the solution, she records it as evidence for the hypothesis.
2. The experimenter shows the solution "VIII" and explains it; if all students come up with the solution, she records it as evidence for the hypothesis.
3. The experimenter does nothing; if all students come up with the solution, she records it as evidence for the hypothesis.

We argue that experiment 1 has high theoretical risk $P(\neg E_1|\neg H)$ and high detectability $P(E_1|H)$. If "insight" has nothing to do with solving the problem ($\neg H$), then presenting the insight that Roman numerals can be used should not lead to all students solving the problem ($\neg E_1$); the experiment, therefore, has high theoretical risk $P(\neg E_1|\neg H)$. Conversely, if insight is required to solve the problem (H), then it is likely to help all students to solve the problem (E_1), the experiment, therefore, has high detectability $P(E_1|H)$. The second experiment, on the other hand, has low theoretical risk $P(\neg E_2|\neg H)$. Even if "insight" has nothing to do with solving the problem ($\neg H$), there are other plausible reasons for observing the evidence (E_2), because the students could simply copy the solution without having any insight. With regard to detectability, experiments 1 and 2 differ in no obvious way. Experiment 3, however, also has low detectability. It is unlikely that all students will come up with the correct solution in a short time (E_3), even if insight is required (H); experiment 3 therefore has low detectability $P(E_3|H)$. The theoretical risk, however, is also low in absolute terms, but high compared to the detectability (statistical relevancy condition is satisfied). In the unlikely event that all 10 students place their matches to form the Roman numeral VIII (E_3), it is probably due to insight (H) and not by chance $P(\neg E_3|\neg H)$). Of course, in practice, we would allow the evidence to be probabilistic, e.g., relax the requirement of "all students" to nine out of ten students, more than eight, and so forth.

As mentioned earlier, the we restrict ourselves to binary evidence, to keep the mathematical notation as simple as possible. We discuss the relation between statistical methods and theoretical risk in the Statistical Methods section.

Preregistration as a means to increase theoretical risk?

Having discussed that increasing the theoretical risk will increase the epistemic value, it is intuitive to task preregistration with maximizing theoretical risk, i.e., a confirmatory research agenda. Indeed, limiting the type-I error rate is commonly stated as *the* central goal of preregistration (Nosek et al., 2018; Oberauer, 2019; Rubin, 2020). We argue that while such a conclusion is plausible, we must first consider at least two constraints that place an upper bound on the theoretical risk.

First, the theory itself limits theoretical risk: Some theories simply do not make risky predictions, and preregistration will not change that. Consider the case of a researcher contemplating the relation between two sets of variables. Suppose each set is separately well studied, and strong theories tell the researcher how the variables within the set relate. However, our imaginary researcher now considers the relation between these two sets. For lack of a better theory, they assume that some relation between any variables of the two sets exists. This is not a risky prediction to make in psychology (Orben & Lakens, 2020). However, we would consider it a success if the researcher would use the evidence from this rather exploratory study to develop a more precise (and therefore risky) theory, e.g., by using the results to specify which variables from one set relate to which variables from the other set, to what extent, in which direction, with which functional shape, etc., to be able to make riskier predictions in the future. We will later show that preregistration increases the degree of belief in the further specified theory, though it remains low till being substantiated by testing the theory again. This is because preregistration increases the expected epistemic value regardless of the theory being tested, as we will show.

Second, available resources limit theoretical risk. Increasing theoretical risk $P(\neg E|\neg H)$ will usually decrease detectability $P(E|H)$ unless more resources are invested. In other words, one cannot increase power while maintaining the same type-I error rate without increasing the invested resources. Tasking preregistration with an increase in theoretical risk makes it difficult to balance this trade-off. Mindlessly maximizing theoretical risk would either never produce evidence or require huge amounts of resources.

Uncertainty about theoretical risk

We have established that higher theoretical risk leads to more persuasive evidence. In other words, we have reconstructed the interpretation that preregistrations supposedly work by restricting the researchers, which in turn increases the theoretical risk (or equivalently limits the type-I error rate) and thereby creates more compelling evidence. Nevertheless, there are trade-offs for increasing theoretical risk. Employing a mathematical framework allows us to navigate the trade-offs more effectively and move towards a second, more favorable interpretation. To that end, we incorporate uncertainty about theoretical risk into our framework.

Statistical methods

One widely known factor is the contribution of statistical methods to theoretical risk. Theoretical risk $P(\neg E|\neg H)$ is deeply connected with statistical methods, because it is related to the type-I error rate in statistical hypothesis testing $P(E|\neg H)$ by $P(\neg E|\neg H) = 1 - P(E|\neg H)$, if you consider the overly simplistic case where the research hypothesis is equal to the statistical alternative-hypothesis because then the nill-hypothesis is $\neg H$. Because many researchers are familiar with the type-I error rate, it can be helpful to remember this connection to theoretical risk. Researchers who choose a smaller type-I error rate can be more sure of their results, if significant, because the theoretical risk is higher. However, this connection should not be overinterpreted for two reasons. First, according to most interpretations of null hypothesis testing, the absence of a significant result should not generally be interpreted as evidence against the hypothesis (Mayo, 2018, p. 5.3). Second, the research hypothesis seldomly equals the statistical alternative-hypothesis. We argue that theoretical risk (and hence its complement, $P(E|\neg H)$) also encompasses factors outside the statistical realm, most notably the study design and broader analytical strategies.

Statistical methods stand out among these factors because we have a large and well-understood toolbox for assessing and controlling their contribution to theoretical risk. Examples of our ability to exert this control are the choice of type-I error rate, adjustments for multiple testing, the use of corrected fit measures (i.e., adjusted R^2), information criteria, or cross-validation in machine learning. These tools help us account for biases in statistical methods that influence theoretical risk (and hence, $P(E|\neg H)$.

The point is that the contribution of statistical methods to theoretical risk can be formally assessed. For many statistical models it can be analytically computed under some assumptions. For those models or assumptions where this is impossible, one can employ Monte Carlo simulation to estimate the contribution to theoretical risk. The precision with which statisticians can discuss contributions to theoretical risk has lured the community concerned with research methods into ignoring other factors that are much more uncertain. We cannot hope to resolve this uncertainty; but we have to be aware of its implications. These are presented in the following.

Sources of Uncertainty

As we have noted, it is possible to quantify how statistical models affect the theoretical risk based on mathematical considerations and simulation. However, other factors in the broader context of a study are much harder to quantify. If one chooses to focus only on the contribution of statistical methods to theoretical risk, one is bound to overestimate it. Take, for example, a t-test of mean differences in two samples. Under ideal circumstances (assumption of independence, normality of residuals, equal variance), it stays true to its type-I error rate. However, researchers may do many very reasonable things in the broader context of the study that affect theoretical risk: They might exclude outliers, choose to drop an item before computing a sum score, broaden their definition of the population to be sampled, translate their questionnaires into a different language, impute missing values, switch between different estimators of the pooled variance, or any number of other things. All of these decisions carry a small risk that they will increase the likelihood of obtaining evidence despite the underlying research hypothesis being false. Even if the t-test itself perfectly maintains its type I error rate, these factors influence $P(E|\neg H)$. While, in theory, these factors may leave $P(E|\neg H)$ unaffected or even decrease it, we argue that this is not the case in practice. Whether researchers want to or not, they continuously process information about how the study is going, except under strict blinding. While one can hope that processing this information does not affect their decision-making either way, this cannot be ascertained. Therefore, we conclude that statistical properties only guarantee a lower bound for theoretical risk. The only thing we can conclude with some certainty is that theoretical risk is not higher than what the statistical model guarantees without knowledge about the other factors at play.

The effects of uncertainty

Before we ask how preregistration influences this uncertainty, we must consider the implications of being uncertain about the theoretical risk. Within the Bayesian framework, this is both straightforward and insightful. Let us assume a researcher is reading a study from another lab and tries to decide whether and how much the presented results confirm the hypothesis. As the researcher did not conduct the study (and the study is not preregistered), they can not be certain about the various factors influencing theoretical risk (researcher degrees of freedom). We therefore express this uncertainty about the theoretical risk as a probability distribution Q of $P(E|\neg H)$ (remember that $P(E|\neg H)$ is related to theoretical risk by $P(E|\neg H) = 1 - P(\neg E|\neg H)$, so it does not matter whether we consider the distribution of theoretical risk or $P(E|\neg H)$). To get the expected value of $P(H|E)$ that follows from the researchers' uncertainty about the theoretical risk, we can compute the expectation using Bayes theorem:

$$\mathbb{E}_Q[P(H|E)] = \mathbb{E}_Q\left[\frac{P(H)P(E|H)}{P(H)P(E|H) + P(\neg H)P(E|\neg H)}\right] \tag{3}$$

Of course, the assigned probabilities and the distribution Q vary from study to study and researcher to researcher, but we can illustrate the effect of uncertainty with an example. Assuming $P(E|H) = 0.8$ (relective of the typically strived for power of 80%). Let us further assume that the tested hypothesis is considered unlikely to be true by the research community before the study is conducted ($P(H) = 0.1$) and assign a uniform distribution for $P(E|\neg H) \sim U([1-\tau, 1])$ where τ is set to $1-\alpha$, reflecting our assumption that this term gives an upper bound for theoretical risk $P(\neg E|\neg H)$. We chose this uniform distribution as it is the maximum entropy distribution with support $[1-\tau, 1]$ and hence conforms to our Bayesian framework (Giffin & Caticha, 2007).

With this, we derive the expected value of $P(H|E)$ as

$$\mathbb{E}_Q[P(H|E)] = \mathbb{E}_Q\left[\frac{P(H)P(E|H)}{P(H)P(E|H) + P(\neg H)P(E|\neg H)}\right] \tag{4}$$

$$= \int_{[1-\tau,1]} \tau^{-1} \frac{P(H)P(E|H)}{P(H)P(E|H) + P(\neg H)P(E|\neg H)}\, \mathrm{d}P(E|\neg H) \tag{5}$$

$$= \frac{P(H)P(E|H)}{P(\neg H)\tau} \ln\left(\frac{P(H)P(E|H) + P(\neg H)}{P(H)P(E|H) + P(\neg H)(1-\tau)}\right) \tag{6}$$

Figure 1 shows exemplary the effect of theoretical risk (x-axis) on the posterior probability (y-axis) being certain (solid line) or uncertain (dashed line) about the theoretical risk of a study. Our expectation of the gained epistemic value varies considerably depending on how uncertain we are about the theoretical risk a study took on. Mathematically, uncertainty about theoretical risk is expressed through the variance (or rather entropy) of the distribution. The increase in uncertainty (expressed as more entropic distributions) leads to a decreased expected epistemic value.

The argument for a confirmatory research agenda is that by increasing theoretical risk we increase expected epistemic value, i.e., moving to the right on the x-axis in Figure 1 increases posterior probability (on the y-axis). However, if a hypothesis in a certain study has low theoretical risk, there is not much researchers can do about it. However, studies do not only differ by how high the theoretical risk is but also by how certain the recipient is about the theoretical risk. A study that has a very high theoretical risk (e.g., 1.00% chance that if the hypothesis is wrong, evidence in its favor will be observed,) but has also maximum uncertainty will result in a posterior probability of 22%, while the same study with maximum certainty will result in 90% posterior probability. The other factors (detectability, prior beliefs, measure of epistemic value) and, therefore, the extent of the benefit varies, of course, with the specifics of the study. Crucially, even studies with some exploratory aspects benefit from preregistration, e.g., in this scenario with a $\tau = 0.80$ (false positive rate of 0.20) moving from uncertain to certain increases the posterior from 0.15 to

0.31.

Preregistration as a means to decrease uncertainty about the theoretical risk

We hope to have persuaded the reader to accept two arguments: First, the theoretical risk is important for judging evidential support for theories. Second, the theoretical risk is inherently uncertain, and the degree of uncertainty diminishes the persuasiveness of the gathered evidence. The third and last argument is that preregistrations reduce this uncertainty. Following the last argument, a preregistered study is represented by the solid line (certainty about theoretical risk), and a study that was not preregistered is more similar to the dashed line (maximally uncertain about theoretical risk) in Figure 1 and Figure 2.

Let us recall our three assumptions:

1. Researchers judge the evidence for or against a hypothesis rationally.
2. They expect other researchers to apply a similar rational process.
3. Researchers try to maximize the expected epistemic value for other researchers.

The point we make with these assumptions is that researchers aim to persuade other researchers, for example, the readers of their articles. Not only the original authors are concerned with the process of weighing evidence for or against a theory but really the whole scientific community the study authors hope to persuade. Unfortunately, readers of a scientific article (or, more generally, any consumer of a research product) will likely lack insight into the various factors that influence theoretical risk. While the authors themselves may have a clear picture of what they did and how it might have influenced the theoretical risk they took, their readers have much greater uncertainty about these factors. In particular, they never know which relevant factors the authors of a given article failed to disclose, be it intentionally or not. From the perspective of the ultimate skeptic, they may claim maximum uncertainty.

Communicating clearly how authors of a scientific report collected their data and consequently analyzed it to arrive at the evidence they present is crucial for judging the theoretical risk they took. Preregistrations are ideal for communicating just that because any description after the fact is prone to be incomplete. For instance, the authors could have opted for selective reporting, that is, they decided to exclude a number of analytic strategies they tried out. That is not to say that every study that was not-preregistered was subjected to practices of questionable research practices. The point is that we cannot exclude it with certainty. This uncertainty is drastically reduced if the researchers have described what they intended to do beforehand and then report that they did exactly that. In that case, readers can be certain they received a complete account of the situation. They still might be uncertain about the actual theoretical risk the authors took, but to a much smaller extent than if the study would not have been preregistered. The remaining sources of uncertainty might be unfamiliarity with statistical methods or experimental paradigms used, the probability of an implementation error in the statistical analyses, a bug in the software used for analyses, etc. In any case, a well-written preregistration should aim to reduce the uncertainty about the theoretical risk and hence increase the persuasiveness of evidence. Therefore, a study that perfectly adhered to its preregistration will resemble the solid line in Figure 1/2. Crucially, perfect means here that the theoretical risk can be judged with low uncertainty, not that the theoretical risk is necessarily high.

Discussion

To summarize, we showed that both higher theoretical risk and lower uncertainty about theoretical risk lead to higher expected epistemic value across a variety of measures. The former result that increasing theoretical risk leads to higher expected epistemic value reconstructs the appeal and central goal of preregistration of confirmatory research agendas. However, theoretical risk is something researchers have only limited control over. For example, theories are often vague and ill-defined, resources are limited, and increasing theoretical risk usually decreases detectability of a hypothesized effect (a special instance of this trade-off is the well-known tension between type-I error and statistical power). While we believe that preregistration is always beneficial, it might be counterproductive to pursue high theoretical risk if the research context is inappropriate for strictly confirmatory research. Specifically, appropriateness here entails the development of precise theories and the availability of necessary resources (often, large enough sample size, but also see Brandmaier et al. (2015)) to adequately balance detectability against theoretical risk.

In terms of preparing the conditions for confirmatory research, preregistration may at most help to invest some time into developing more specific, hence riskier, implications of a theory. But for a confirmatory science, it will not be enough to preregister all studies. This undertaking requires action from the whole research community (Lishner, 2015). Incentive structures must be created to evaluate not the outcomes of a study but the rigor with which it was conducted (Cagan, 2013; Schönbrodt et al., 2022). Journal editors could encourage theoretical developments that allow for precise predictions that will be tested by other researchers and be willing to accept registered reports (Fried, 2020a, 2020b; van Rooij & Baggio, 2021, 2020). Funding agencies should demand an explicit statement about theoretical risk in relation to detectability and must be willing to provide the necessary resources to reach adequate levels of both (Koole & Lakens, 2012).

Our latter result, on the importance of preregistration for minimizing uncertainty, has two important implications. The first is, that even if all imaginable actions regarding promoting higher theoretical risk are taken, confirmatory research should be preregistered. Otherwise, the uncertainty about the theoretical risk will diminish the advantage of confirmatory research. Second, even under less-than-ideal circumstances for confirmatory research, preregistration is beneficial. Preregistering exploratory studies increases the expected epistemic value by virtue of reducing uncertainty about theoretical risk. Nevertheless, exploratory studies will have a lower expected epistemic value than a more confirmatory study if both are preregistered and have equal detectability.

Focusing on uncertainty reduction also explains two common practices of preregistration that do not align with a confirmatory research agenda. First, researchers seldomly predict precise numerical outcomes, instead they use preregistrations to describe the process that generates the results. Precise predictions would have very high theoretical risk (they are likely incorrect if the theory is wrong). A statistical procedure may have high or low theoretical risk depending on the specifics of the model used. Specifying the process, therefore, is in line with the rationale we propose here, but is less reasonable when the goal of preregistration is supposed to be a strictly confirmatory research agenda.

Second, researchers often have to deviate from the preregistration and make data-dependent decisions after the preregistration. If the only goal of preregistration is to ensure confirmatory research, such changes are not justifiable. However, under our rational, some changes may be justified. Any change increases the uncertainty about the theoretical risk and may even decrease the theoretical risk. The changes still may be worthwhile if the negative outcomes may be offset by an increase in detectability due to the change. Consider a preregistration that failed to specify how to handle missing values, and researchers subsequently encountering missing values. In such case, detectability becomes zero because the data cannot be analyzed without a post-hoc decision about how to handle the missing data. Any such decision would constitute a deviation from the preregistration, which is possible under our proposed objective. Note that a reader cannot rule out that the researchers leveraged the decision to decrease theoretical risk, i.e., picking among all options the one that delivers the most beneficial results for the theory (in the previous example, chosing between various options of handling missing values). Whatever decision they make, increased uncertainty about the theoretical risk is inevitable and the expected epistemic value is decreased compared to a world where they anticipated the need to deal with missing data. However, it is still justified to deviate. After all they have not anticipated the case and are left with a detectablilty of zero. Any decision will increase detectability to a non-zero value offsetting the increase in uncertainty. The researchers also may do their best to argue that the deviation was not motivated by increasing theoretical risk, thereby, decreasing the uncertainty. Ideally, there is a default decision that fits well with the theory or with the study design. Or, if there is no obvious candidate, the researchers could conduct a multiverse analysis of the available options to deal with missings to show the influence of the decision (Steegen et al., 2016).

As explained above, reduction in uncertainty as the objective for preregistration does not only explain some existing practice, that does not align with confirmation as a goal, it also allows to form recommendations to improve the practice of preregistration. Importantly, we now have a theoretical measure to gauge the functionality of preregistrations, which can only help increase its utility. In particular, a preregistration should be specific about the procedure that is intended to generate evidence for a theory. Such a procedure may accommodate a wide range of possible data, i.e., it may be exploratory. The theoretical risk, however low, must be communicated clearly. Parts of the process left unspecified imply uncertainty, which preregistration should reduce. However, specifying procedures that can be expected to fail will lead to deviation and, subsequently, to larger uncertainty.

We have proposed a workflow for preregistration called preregistration as code (PAC) elsewhere (Peikert et al., 2021). In a PAC, researchers use computer code for the planned analysis as well as a verbal description of theory and methods for the preregistration. This combination is facilitated by dynamic document generation, where the results of the code, such as numbers, figures, and tables, are inserted automatically into the document. The idea is that the preregistration already contains “mock results” based on simulated or pilot data, which are replaced after the actual study data becomes available. Such an approach dissolves the distinction between the preregistration document and the final scientific report. Instead of separate documents, preregistration, and final report are different versions of the same underlying dynamic document. Deviations from the preregistration can therefore be clearly (and if necessary, automatically) isolated, highlighted, and inspected using version control. Crucially, because the preregistration contains code, it may accommodate many different data patterns, i.e., it may be exploratory. However, while a PAC does not limit the extent of exploration, it is very specific about the probability to generate evidence even when the theory does not hold (theoretical risk). Please note that while PAC is ideally suited to reduce uncertainty about theoretical risk, other more traditional forms of preregistration are also able to advance this goal.

Contrary to what is widely assumed about preregistration, a preregistration is not necessarily a seal of confirmatory research. Confirmatory research would almost always be less persuasive without preregistration, but in our view, preregistration primarily communicates the extent of confirmation, i.e., theoretical risk, of a study. Clearly communicating theoretical risk is important because it reduces the uncertainty and hence increases expected epistemic value.

Acknowledgement

We thank Leo Richter, Caspar van Lissa, Felix Schönbrodt, the discussants at the DGPS2022 conference and Open Science Center Munich, and many more for the insightful discussions about disentangling preregistration and confirmation. We are grateful to Julia Delius for her helpful assistance in language and style editing.

References

Bakker, M., Veldkamp, C. L. S., Assen, M. A. L. M. van, Crompvoets, E. A. V., Ong, H. H., Nosek, B. A., Soderberg, C. K., Mellor, D., & Wicherts, J. M. (2020). Ensuring the quality and specificity of preregistrations. *PLOS Biology*, *18*(12), e3000937. https://doi.org/10.1371/journal.pbio.3000937

Brandmaier, A. M., Oertzen, T. von, Ghisletta, P., Hertzog, C., & Lindenberger, U. (2015). LIFESPAN: A tool for the computer-aided design of longitudinal studies. *Frontiers in Psychology*, *6*, 272.

Cagan, R. (2013). San Francisco Declaration on Research Assessment. *Disease Models & Mechanisms*, dmm.012955. https://doi.org/10.1242/dmm.012955

Carnap, R. (1950). *Logical Foundations of Probability*. Chicago, IL, USA: Chicago University of Chicago Press.

Chan, A.-W., Hróbjartsson, A., Haahr, M. T., Gøtzsche, P. C., & Altman, D. G. (2004). Empirical Evidence for Selective Reporting of Outcomes in Randomized TrialsComparison of Protocols to Published Articles. *JAMA*, *291*(20), 2457–2465. https://doi.org/10.1001/jama.291.20.2457

Christensen, D. (1991). Clever Bookies and Coherent Beliefs. *The Philosophical Review*, *100*(2), 229–247. https://doi.org/10.2307/2185301

Dwan, K., Altman, D. G., Arnaiz, J. A., Bloom, J., Chan, A.-W., Cronin, E., Decullier, E., Easterbrook, P. J., Elm, E. V., Gamble, C., Ghersi, D., Ioannidis, J. P. A., Simes, J., & Williamson, P. R. (2008). Systematic Review of the Empirical Evidence of Study Publication Bias and Outcome Reporting Bias. *PLOS ONE*, *3*(8), e3081. https://doi.org/10.1371/journal.pone.0003081

Fetzer, J. H. (1974). Statistical Explanations. In K. F. Schaffner & R. S. Cohen (Eds.), *PSA 1972: Proceedings of the 1972 Biennial Meeting of the Philosophy of Science Association* (pp. 337–347). Springer Netherlands. https://doi.org/10.1007/978-94-010-2140-1_23

Fried, E. I. (2020a). Lack of Theory Building and Testing Impedes Progress in The Factor and Network Literature. *Psychological Inquiry*, *31*(4), 271–288. https://doi.org/10.1080/1047840X.2020.1853461

Fried, E. I. (2020b). Theories and Models: What They Are, What They Are for, and What They Are About. *Psychological Inquiry*, *31*(4), 336–344. https://doi.org/10.1080/1047840X.2020.1854011

Giffin, A., & Caticha, A. (2007). Updating Probabilities with Data and Moments. *AIP Conference Proceedings*, *954*, 74–84. https://doi.org/10.1063/1.2821302

Hoyningen-Huene, P. (2006). Context of Discovery Versus Context of Justification and Thomas Kuhn. In J. Schickore & F. Steinle (Eds.), *Revisiting Discovery and Justification: Historical and philosophical perspectives on the context distinction* (pp. 119–131). Springer Netherlands. https://doi.org/10.1007/1-4020-4251-5_8

Ioannidis, J. P. A. (2005). Why Most Published Research Findings Are False. *PLOS Medicine*, *2*(8), e124. https://doi.org/10.1371/journal.pmed.0020124

Koole, S. L., & Lakens, D. (2012). Rewarding Replications: A Sure and Simple Way to Improve Psychological Science. *Perspectives on Psychological Science*, *7*(6), 608–614. https://doi.org/10.1177/1745691612462586

Kukla, A. (1990). Clinical Versus Statistical Theory Appraisal. *Psychological Inquiry*, *1*(2), 160–161. https://doi.org/10.1207/s15327965pli0102_9

Lishner, D. A. (2015). A Concise Set of Core Recommendations to Improve the Dependability of Psychological Research. *Review of General Psychology*, *19*(1), 52–68. https://doi.org/10.1037/gpr0000028

Mayo, D. G. (2018). *Statistical Inference as Severe Testing: How to Get Beyond the Statistics Wars* (First). Cambridge University Press. https://doi.org/10.1017/9781107286184

Meehl, P. E. (1990). Appraising and Amending Theories: The Strategy of Lakatosian Defense and Two Principles that Warrant It. *Psychological Inquiry*, *1*(2), 108–141.

https://doi.org/10.1207/s15327965pli0102_1

Meehl, P. E. (1978). Theoretical risks and tabular asterisks: Sir Karl, Sir Ronald, and the slow progress of soft psychology. *Journal of Consulting and Clinical Psychology*, *46*(4), 806–834. https://doi.org/10.1037/0022-006X.46.4.806

Mellor, D. T., & Nosek, B. A. (2018). Easy preregistration will benefit any research. *Nature Human Behaviour*, *2*(2), 98–98. https://doi.org/10.1038/s41562-018-0294-7

Niiniluoto, I. (1998). Verisimilitude: The Third Period. *The British Journal for the Philosophy of Science*, *49*(1), 1–29. https://doi.org/10.1093/bjps/49.1.1

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, *115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Oberauer, K. (2019). Preregistration of a forking path – What does it add to the garden of evidence? In *Psychonomic Society Featured Content*.

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716. https://doi.org/10.1126/science.aac4716

Orben, A., & Lakens, D. (2020). Crud (Re)Defined. *Advances in Methods and Practices in Psychological Science*, *3*(2), 238–247. https://doi.org/10.1177/2515245920917961

Peikert, A., & Brandmaier, A. M. (2023a). *Supplemental materials for preprint: Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization.* Zenodo. https://doi.org/10.5281/zenodo.7648471

Peikert, A., & Brandmaier, A. M. (2023b). *Why does preregistration increase the persuasiveness of evidence? A Bayesian rationalization.* PsyArXiv; PsyArXiv. https://doi.org/10.31234/osf.io/cs8wb

Peikert, A., van Lissa, C. J., & Brandmaier, A. M. (2021). Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once. *Psych*, *3*(4), 836–867. https://doi.org/10.3390/psych3040053

Pham, M. T., & Oh, T. T. (2021). Preregistration Is Neither Sufficient nor Necessary for

Good Science. *Journal of Consumer Psychology*, *31*(1), 163–176. https://doi.org/10.1002/jcpy.1209

Popper, K. R. (2002). *The logic of scientific discovery*. Routledge.

Rubin, M. (2020). Does preregistration improve the credibility of research findings? *The Quantitative Methods for Psychology*, *16*(4), 376–390. https://doi.org/10.20982/tqmp.16.4.p376

Salmon, W. C. (1970). Statistical Explanation. In *The Nature & function of scientific theories: Essays in contemporary science and philosophy* (pp. 173–232). University of Pittsburgh Press.

Schönbrodt, F., Gärtner, A., Frank, M., Gollwitzer, M., Ihle, M., Mischkowski, D., Phan, L. V., Schmitt, M., Scheel, A. M., Schubert, A.-L., Steinberg, U., & Leising, D. (2022). *Responsible Research Assessment I: Implementing DORA for hiring and promotion in psychology*. PsyArXiv. https://doi.org/10.31234/osf.io/rgh5b

Shmueli, G. (2010). To Explain or to Predict? *Statistical Science*, *25*(3), 289–310. https://doi.org/10.1214/10-STS330

Silagy, C. A., Middleton, P., & Hopewell, S. (2002). Publishing Protocols of Systematic Reviews Comparing What Was Done to What Was Planned. *JAMA*, *287*(21), 2831–2834. https://doi.org/10.1001/jama.287.21.2831

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2021). Pre-registration: Why and How. *Journal of Consumer Psychology*, *31*(1), 151–162. https://doi.org/10.1002/jcpy.1208

Steegen, S., Tuerlinckx, F., Gelman, A., & Vanpaemel, W. (2016). Increasing Transparency Through a Multiverse Analysis. *Perspectives on Psychological Science*, *11*(5), 702–712. https://doi.org/10.1177/1745691616658637

Stefan, A. M., & Schönbrodt, F. D. (2023). Big little lies: A compendium and simulation of p-hacking strategies. *Royal Society Open Science*, *10*(2). https://doi.org/10.1098/rsos.220346

Szollosi, A., Kellen, D., Navarro, D. J., Shiffrin, R., Rooij, I. van, Zandt, T. V., & Donkin,

C. (2020). Is Preregistration Worthwhile? *Trends in Cognitive Sciences*, *24*(2), 94–95. https://doi.org/10.1016/j.tics.2019.11.009

van Rooij, I., & Baggio, G. (2021). Theory Before the Test: How to Build High-Verisimilitude Explanatory Theories in Psychological Science. *Perspectives on Psychological Science*, *16*(4), 682–697. https://doi.org/10.1177/1745691620970604

van Rooij, I., & Baggio, G. (2020). Theory Development Requires an Epistemological Sea Change. *Psychological Inquiry*, *31*(4), 321–325. https://doi.org/10.1080/1047840X.2020.1853477

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., van der Maas, H. L. J., & Kievit, R. A. (2012). An Agenda for Purely Confirmatory Research. *Perspectives on Psychological Science*, *7*(6), 632–638. https://doi.org/10.1177/1745691612463078

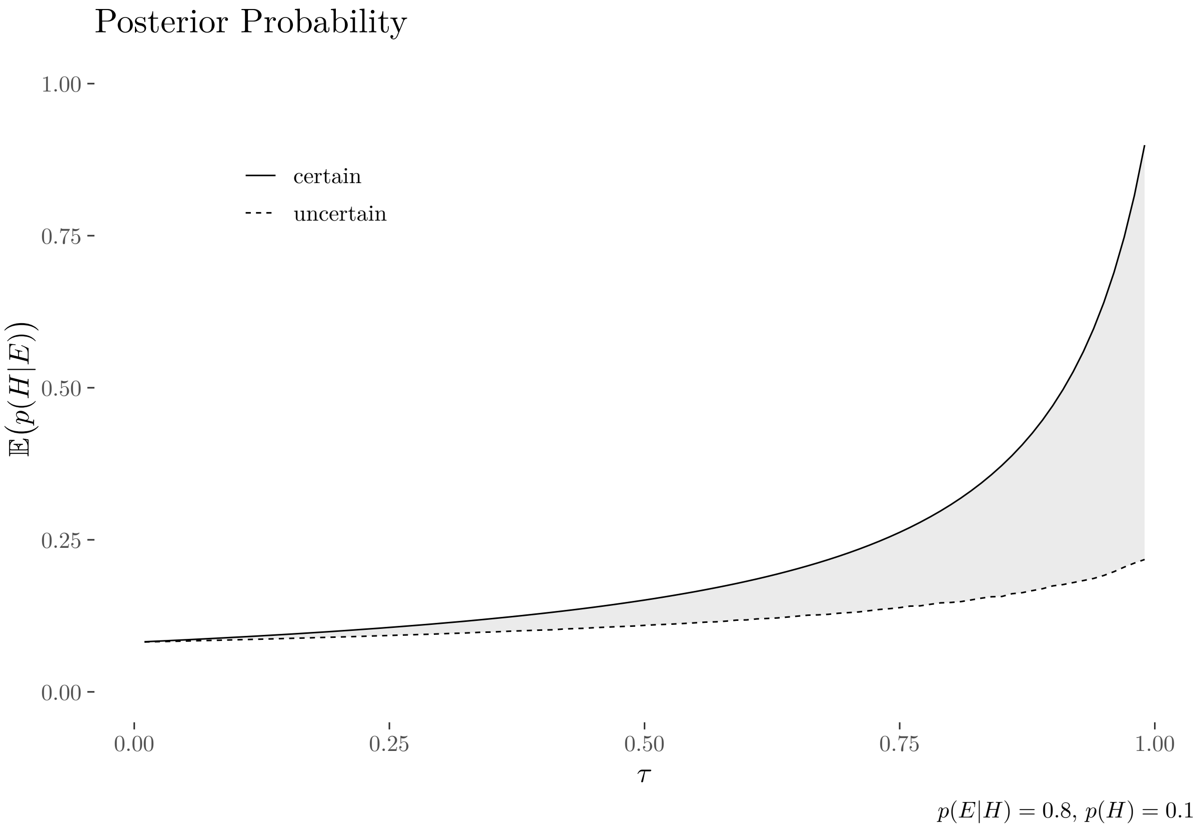


Figure 1

Posterior probability (confirmation as firmness) as a function of theoretical risk τ, where τ is either certain (solid line) or maximally uncertain (dotted line).

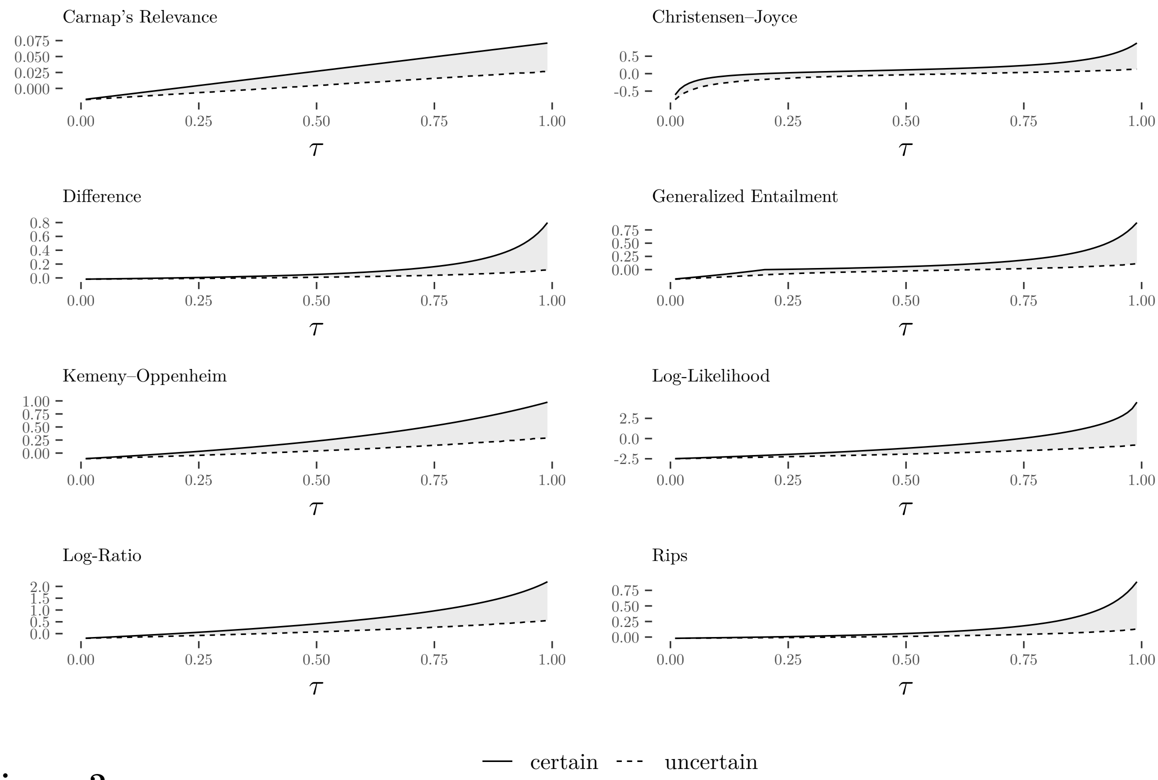


Figure 2

Several measures for confirmation as an increase in firmness as a function of τ, where τ is either certain (solid line) or maximally uncertain (dotted line).