Reproducibility, Correctness, and Buildability: the Three Principles for Ethical Public Dissemination of Computer Science and Engineering Research

Kristin Yvonne Rozier
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
Email: Kristin.Y.Rozier@nasa.gov

Eric W. D. Rozier
Electrical and Computer Engineering
University of Miami
Coral Gables, FL 33146
Email: erozier2@ieee.org

***Abstract*—We propose a system of three principles of public dissemination, which we call reproducibility, correctness, and buildability, and make the argument that consideration of these principles is a necessary step when publicly disseminating results in any evidence-based scientific or engineering endeavor. We examine how these principles apply to the release and disclosure of the four elements associated with computer science research: theory, algorithms, code, and data. Reproducibility refers to the capability to reproduce fundamental results from released details. Correctness refers to the ability of an independent reviewer to verify and validate the results of a paper. We introduce the new term buildability to indicate the ability of other researchers to use the published research as a foundation for their own new work. This is more broad than extensibility, as it requires that the published results have reached a level of completeness that the research can be used for its stated purpose, and has progressed beyond the level of a preliminary idea. We argue that these three principles are not being sufficiently met by current publications and proposals in computer science and engineering, and represent a goal for which publishing should continue to aim. We introduce standards for the evaluation of reproducibility, correctness, and buildability in relation to the varied elements of computer science research and discuss how they apply to proposals, workshops, conferences, and journal publications, making arguments for appropriate standards of each principle in these settings. We address modern issues including big data, data confidentiality, privacy, security, and privilege. Our examination raises questions for discussion in the community on the appropriateness of publishing works that fail to meet one, some, or all of the stated principles.**

I. Introduction

Recently, many prominent researchers have expressed concerns about shortfalls in the current methods and standards for public dissemination of computer science and engineering research. In *Communications of the ACM*, Vardi expressed concern that "our system has compromised one of the cornerstones of scientific publication – peer review" [1] and Crowcroft, Keshav, and McKeown pointed out flaws in the current review process that have lead to declining paper quality [2]. Birman and Schneider called for "an informed debate and a community response" lest our behavior "stunt the impact of our work and retard evolution of the scientific enterprise" [3]. Vitek and Kalibera summed up the problem [4]:

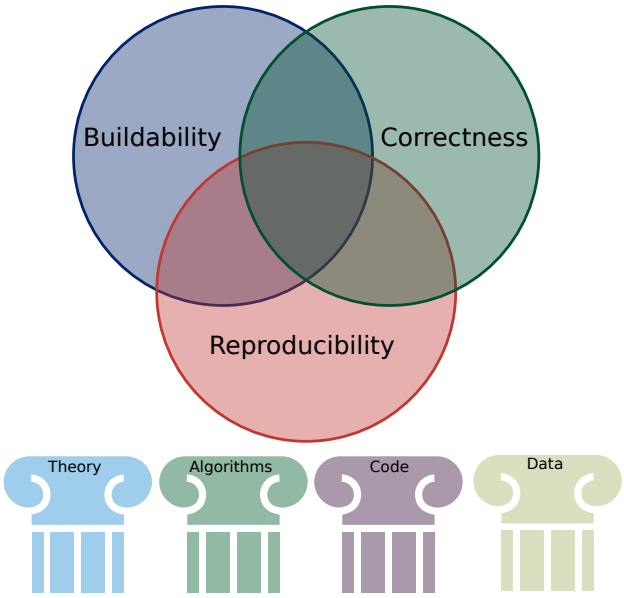

Fig. 1. The three overlapping principles for ethical public dissemination are upheld over the four elements of computer science and engineering research: theory, algorithms, code, and data.

> Evaluating a non-trivial idea is beyond the time budget of any single paper . . . Often a careful comparison to the state of the art means implementing competing solutions. The result of this state of affairs is that papers presenting potentially useful novel ideas regularly appear without a comparison to the state of the art, without appropriate benchmarks, without any mention of limitations, and without sufficient detail to reproduce the experiments. This hampers scientific progress and perpetuates the cycle.

The review process for papers and proposals certainly deserves a debate. It is well-documented that reviewers are overwhelmed by rapidly growing numbers of submissions [3], [2]; with the pressure to publish and inconsistent acceptance

standards there is incentive for researchers not to police themselves. Often authors are made to prove that their work is novel or interesting, rather than making them prove that their findings are true [5], [6]. The current review process, combined with mandated low acceptance rates, can allow some good papers to be rejected while some flawed ones are accepted or even receive awards before the community discovers them to be inaccurate [7]. We aim to mitigate these problems by providing a roadmap to more easily evaluate essential elements of different types of research and provide a common language for more fair evaluation. By enumerating ways to uphold the principles of reproducibility, correctness, and buildability, even under release restrictions and space constraints, we can find a better balance. Hopefully this will serve to increase the efficiency of reviewing and help with the problem of bringing up the average quality of papers and proposals. We can do better without hindering the rate of scientific progress [4].

We propose a system of three principles of public dissemination:

1) Reproducibility: Can other researchers reasonably recreate the work? Is there enough information for them to do this? Reproducibility refers to the capability to reproduce fundamental results from released details. Results are considered reproducible if there is some sufficient combination of theory, data, algorithms, libraries, or executable code such that it is reasonable to believe the fundamental results could be reproduced independently by a third party.

2) Correctness: Can the correctness of the work be checked? Can readers reasonably know it's correct and why it's correct? Correctness refers to the ability of an independent reviewer to verify and validate the results. While it is generally accepted that any new theoretical material requires accompanying proofs with enough details for independent verification, the question of establishing correctness in a practical way is much more complex for other research elements. Results may demonstrate correctness if there is some sufficient combination of evidence including formal proofs, informal correctness arguments, comparisons with previous work, experimental evaluations, or data. For new code or algorithms, correctness may mean establishing consistency of results versus existing implementations, standard benchmarks, or sanity checks via statistically significant experimental results, with any inconsistencies explained and validated.

3) Buildability: Can others build on this work? Can others build on this work without having to re-discover parts of the work? Does it advance state of the art in such a way that others can proceed without reinventing any part? We introduce the new term buildability to indicate the ability of other researchers to use the published research as a foundation for their own new work. This is more broad than extensibility, as it requires that the published results have reached a level of completeness so that the research can be used for its stated purpose, and has progressed beyond the level of a preliminary idea. To meet the standard of buildability, it is important that the work provide the necessary information for others to build on it, extend it, or utilize it (even as a black box) to produce new work suitable for an archive-quality research publication.

We recognize that there are many reasons to publish scientific work. For example, Meyer suggests four such reasons: as publicity, as examination, as business, and as a ritual, but principally states, "Publication is about helping the advancement of humankind" [7]. Significantly, the history of scientific publication illustrates this purpose. The great advances of scientists such as Hooke and Newton motivated governments and wealthy patrons to subsidize science so that both the public and scientific community could benefit from shared discoveries, transforming a culture of scientific secrecy to one where publication and the sharing of research brought prestige and rewards [8].

In order for research to help advance humankind, we argue that the consideration of the three principles of reproducibility, correctness, and buildability is a necessary step when publicly disseminating results in any evidence-based scientific or engineering endeavor. Reproducibility is necessary because it is important to first repeat previous experiments before conducting original work [9]. Correctness, the most universal of the three, is necessary due to the role of publication as exam [7] and sanction [10]. Buildability is necessary because, to advance humankind, we make our discoveries by building upon those previously revealed. The often cited adage, whether attributed to Newton [11] or Torvalds [12], "If I have seen further it is by standing on the shoulders of giants," is still as true today about our scientific endeavors as it always has been. We believe these ideals and standards apply to papers submitted for publication at all types of venues, including workshops, symposia, conferences, and journals, as well as proposals and reviews of final deliverables from grant proposals, though not all papers or venues require focus on all three principles.

In this paper, we examine reproducibility, correctness, and buildability, detailing how they apply to the release and disclosure of each of the four elements associated with computer science research: theory, algorithms, code, and data. These are like a Venn Diagram with three intersecting circles: the *best* papers published at a top-rated conference should be in the middle; see Figure 1. However, a workshop or a solicitation for preliminary research may call for work falling into other areas within the diagram. For example, a workshop aimed at discussing ideas in progress may choose to ask for only reproducibility and partial correctness, e.g., correctness of the work done so far, leaving full correctness, e.g., working out corner cases, for a fruitful discussion at the workshop and buildability for future publication when the work is more mature. We ask the question: why publish any work that falls outside of all three? While, for example, conferences legitimately have rules enforcing other requirements, like that work is sufficiently novel or relating to a 'hot' topic area, there is still benefit in publicly disseminating work that is correct, reproducible, and buildable but not novel or interesting. However, we argue that not only is there no point to publicly disseminating research that does not adhere to any of these three principles; it is unethical.

Given that it is possible for all publicly disseminated research to uphold these three principles and deviation from them is determined by factors that restrict public dissemination, such as intellectual property, secure/classified information, and time, it is unethical to publicly disseminate work that does not adhere to at least one of the principles of reproducibility, correctness, and buildability.

While these principles may seem obvious, we argue that

they are not being sufficiently met by current publications in computer science and engineering, and represent a goal for which publishing should continue to aim. In this paper, we address data confidentiality and how it relates to these principles, discussing techniques for data sanitization, and dissemination of confidential results in ways that preserve privacy, security, and privilege while still addressing reproducibility, correctness, and buildability. We discuss ethical questions inherent in public dissemination of research along with ideas for adhering to the three principles under publishing constraints. We question the appropriateness of publishing works based on data, code, and algorithms that can not be released in any fashion.

We divide our suggestions into these categories for ease of reference in any part of a given paper, e.g., our discussion of code is intended to apply to any paper created using any code, not only papers about code. So, we consider a theory paper containing a graph to contain code, specifically the code used to produce the graph. We expect many computer science and engineering papers will then contain all four types of research: theory, algorithms, code, and data.

While it is often implied that these standards should be met, and we believe many authors who do meet these standards simply do not explicitly say so in their papers due to space or other restrictions, we argue that implication is not enough. Both a research call and the final, publicly disseminated work, need to call them out directly. The Evaluate Collaboratory recently wrote an open letter to PC chairs advocating reproducibility be explicitly required in calls for papers.[1] Perhaps conferences could include on their submission pages a place, i.e., alongside standard author data, for up to three bullet points stating how the paper demonstrates each of reproducibility, correctness, and buildability. This would aid the strained conference review system both by helping reviewers more efficiently judge these most essential standards and by helping them to check if they are clear in the paper's text. This requirement would also help authors communicate clearly by establishing a common framework to use when describing these elements of their results.

Finding space to include these principles is a significant problem. Conferences are the top publication venues, with conference publications considered superior to journal publications [13], [1], [14] but conference publications must adhere to stricter page limits. We agree with recent arguments [3] that publications should be evaluated more in terms of impact. Essentially, conferences are tasked with communicating higher-impact work in less space. We hope to mitigate this problem by helping organize the question of how to fit it all in. We believe that cases for the required principles (one, two, or all three) named by any publication venue should be covered explicitly in any published paper's main text. We argue that in the modern digital era it is reasonable for a publication venue with a strict page limit to require each published paper to have a website containing any additional information required to fully support the case(s) contained in the paper's main text.

There is also the increasing problem of overly critical reviews; we aim to mitigate that problem, not to fuel it. By providing a list of options to uphold the three principles, we hope to make focusing on the big picture easier and more standardized, not provide a tool for nitpicking or rejecting more papers and proposals. It is likely impossible to implement all of the options we list in one paper; many apply only in limited contexts. We make no claims that this list is complete, only that it is a starting point.

This paper is organized as follows. In Sections II, III, and IV we address the three principles of public dissemination in turn: reproducibility, correctness, and buildability, respectively. In each of these three sections, we discuss options for achieving that principle addressing the four types of contributions publicly disseminated in computer science and engineering research. Specifically, we refer to the public dissemination of research whose contributions take the form of theory, algorithms, code, and data. Section V concludes and offers a perspective for the future.

II. Reproducibility

The essentiality of reproducibility in publication has been well documented [15], [16], [17], [18], [6], [4], [19]; see the Evaluate Collaboratory for more references.[2] Perhaps most directly, "For a field to qualify as a science, it is important first and foremost that published work be reproducible by others" [15]. Ensuring that independent scientists can reproduce published results can take many forms. For example, Thompson and Burnett proposed Three Concepts of Reproducible Research [17]. These principles include use of collaborative environments and computing tools that facilitate the performance, preservation, and transmission of computational methods; preserving the computations used to support academic work products; and writing *active papers* that allow the reader to both read about and actively perform tasks. Claerbout gives four rules for producing reproducible research, ensuring that old versions are cleaned out and the published version can be reliably automatically reconstructed [18]. Many separate *repetitions*, or the ability to re-run a method and obtain the same or very similar result, from *reproducibility*, which is carried out based on a publication and using information provided by the authors such as data sets or code [4].

Reproducibility also involves the use of clearly defined approaches for the inclusion of information for public dissemination [17]. Indeed, authors making available all information required to reproduce their published results has famously led to many important corrections, greatly influenced policy-makers, and benefited the progression of science and engineering [20], [5], [21], [22]. Perhaps by more explicitly asking authors to show reproducibility in their paper submissions we could catch more mistakes earlier, or even before publication. A common complaint is the culture of valuing novel and interesting work over reproducible, and demonstrably correct work [5]. A recent study found that the vast majority of published results in computer science and engineering are positive; therefore we propose more explicit support for publishing more impactful negative, but reproducible results [23].

It is important to remember that not all work is reproducible. We use our own work as an example. We published a paper [24] about an extensive system analysis in which, in addition to our quantitative methods of validation that were posted online for others to re-run, we took the extra step of a

[1] http://evaluate.inf.usi.ch/letter-to-pc-chairs

[2] http://evaluate.inf.usi.ch/issues/reproducibility

small qualitative sanity check. We asked the system designers whether they thought we accurately captured their system. (Their affirmative reply is evidenced in the system design changes made in response to our analysis.) One reviewer claimed it was not ethical to publish this paper because all results are not reproducible: other people may not have access to that same set of system designers. We think that particular aside in the paper is not reproducible for other reasons; for example, while these particular system designers were quite consistent, others may have different opinions on different days. Nevertheless, we advocate a standard of reasonableness: checking for reproducibility overall, based on the major contributions of the paper or proposal and whether or not reproducibility is possible. For example, a lengthy journal paper that includes a brief mention of an unreproducible sanity check would not violate our reproducibility principle based solely on this inclusion.

Reproducibility is an old idea worth continuously revisiting in this age of big data and increasingly complex systems, and intellectual property legalities. For example, there are cases where it may not be possible to distribute the complete data set. Or a combination of complex computational analysis and highly specialized or proprietary analysis toolkits may make reproducibility difficult. There may be a need to take scaling into account. Claerbout states, "When a project is truly large, we actually do not test the reproducibility, but we do test the presence of the makefile commands to reproduce the result" [18]. This applies to the review process in particular: spending the time and resources to fully check a paper for publication may require a very high level of effort deemed unreasonable for producing a timely review. We argue that checking simply for the presence of artifacts needed for reproducibility should be required however.

Note that we are not advocating that paper reviewers become even more overburdened by being required to reproduce the submitted work, or part thereof, which may constitute as much work as writing the paper in the first place. Rather we are suggesting simple checks that materials that would be needed for someone to do this in the future are present. This benefits both reviewers and authors as, "Experience shows the principle beneficiary of reproducible research is you the author yourself" [18].

A. Theory

Theory is one of the two legs of science; while it was once largely carried out by hand, "'doing' theory today requires highly sophisticated computational-science techniques carried out on cutting-edge high-performance computers" [25]. Reproducibility for theory increasingly means providing the details of these computations and computational models. In our anecdotal experience, theory is most often published in reproducible ways in line with the following suggestions:

- All theorems upon which the paper is building should be explicitly stated, in the paper's main text.
- Every previously published theorem should be accompanied by a citation indicating where the proof originally appeared inside the theorem statement for easy reference.
- Every new theorem should be accompanied by a full version of the proof, without any exercises left to the reader. Proofs may not fit in the main text; they may be included elsewhere such as in an appendix or online version.
- We have also seen 'proof arguments' that either give helpful intuition behind a longer proof, which may or may not itself be included elsewhere in the paper's supplemental materials. Proof arguments for previously published theorems can also state how the previously published proof may be adapted for the context of the current paper, thus enabling readers to reproduce the proof but in a more straightforward way in relation to the current work.
- For all types of theory contributions, authors should make every effort to clearly enumerate all assumptions. Details of context are also helpful here and usually contribute to demonstrating novelty and interest in addition to reproducibility.

Errors can and will certainly continue to occur in every step of the scientific process, starting with the underlying theory. Reproducibility increases the chance they are caught earlier. This in turn decreases the chance they unduly influence derivative work or otherwise cause others to try to build on erroneous results. Previous studies [17] have shown that these errors are often made by the omission of small details. Therefore we do not believe it is unreasonable to equate full, detailed proofs, absent of any 'hand-waving', with greater reproducibility.

B. Algorithms

While one way of providing reproducibility for an algorithm is to provide the code of the authors' implementation of that algorithm, we argue it is also useful to consider reproducibility of an algorithm independently. Separating the algorithm from its implementation aids in both reproducibility and correctness. We recommend the following options for reproducibility of algorithms:

- List the algorithm *in its entirety*, including small details like the types of the inputs. We presume the use of some formal semantics like pseudocode.
    - If the algorithm is short enough, this can be done in the papers' main text.
    - If not, important snippets of the algorithm can appear in the main text and the full algorithm can be provided in an attached appendix, online appendix, paper's website, or expanded version e.g. in the form of an accompanying technical report if necessary.
- Include an English-prose description of the algorithm alongside the formal listing/algorithm snippets. This dual-presentation better enables understanding and greatly increases the readers' ability to both reproduce and build upon the algorithm.
- Comment the algorithm, every line if possible. While algorithms are not code, helpful labels greatly aid in reproducibility. For example, it is important to designate essential, unchangeable bits (such as a search that

must be performed in a very specific order) from more standardized operations that might easily be optimized or otherwise altered for the users' purposes (such as a search that need only find the correct element). This is even more important when components of different operations, such as a search, are interleaved with more complex operations.

- Provide the code of the authors' implementation of the algorithm if it was implemented. This aids in both reproducibility and correctness.
- Post a video walking through the algorithm on the paper's website or a video sharing site like YouTube. Multimedia presentation may be the most effective way to thoroughly explain and demonstrate the algorithm [8].

In addition to considering reproducibility of an algorithm that is itself the result of the paper, we argue that it is important to include reproducibility of the algorithms utilized by authors to conduct statistical analysis, create graphs or tables, and draw conclusions. Therefore, we argue for including, in an appendix, online version, or paper website, algorithms for reproducing important figures and statistics, such as a graph, including links or citations for any data sets, data filters, support software, scripts/makefiles, or other tools used in its creation. This may be as simple as providing the command-line call to a standard tool. Specific parameters used to create a figure or table should be included in legends, column labels, and captions. Details such as software version numbers, command-line flags, start values/seeds, and input parameters should be required content in the paper's main text. They do not require much room yet provide an essential foundation for reproducibility.

C. Code

Code may comprise the main contribution of a paper or only a small element, such as a script used to make a graph for a theory paper. While it is not important for others to be able to recreate the same shade of blue used in a figure, any significant code contributions and quantitative results stemming from that code need to be reproducible. Vitek and Kalibera ask, "Is a paper for which the author feels that it is not worth making the code available worth reviewing?" [4]. Boettiger states, "I feel that means reviewing not just the submitted manuscript but the software itself. If we can agree on nothing else, we as a community should at least be able to say: my review of a software paper is a review of the software" [26]. Many [27], [28], [4] suggest a checklist for reproducibility and extensibility of code publications that includes ensuring that a reviewer, during the review period, can *efficiently* build/deploy/install/run the software, and identify whether the software is operating as expected. This should not further overload the reviewer but instead ease the ability to check for reproducibility.

Actions supporting full code reproducibility include:

- Link the full code listing online or print it in an appendix (or both).
- Store the code in an online database or code repository.
- Give the full algorithm or pseudocode so the code can be reproduced from that.
- Provide documentation explicitly specifying
  - why the code is useful;
  - precisely what the code does;
  - what operating systems;
  - full, step-by-step instructions for running one illustrative example;
  - plain text explanations of functions, how to call them, and their arguments;
  - any constraints on code use;
  - any suggestions for reuse including optimizations, applications, targets for parallelization, extensions proposed by "Future Work" sections.
- Accompany the code with precompiled binaries to save others from compilation.

When the results of running the code are also reported, the following should also be detailed:

- the operating system of the experimental machine, including precise version and patch information where relevant
- the hardware specifications of the experimental machine, including any details that may have affected timing results reported, such as processor, memory, speed, etc.
- any relevant details of the compiler or other support software used
- if timing and/or memory results are reported, information about the run environment that might have affected the results, such as having exclusive access to a compute node
- optionally a link to a paper website with documentation, additional examples, or sample input and output data

These items do not take up much room and should be provided in the main text of the paper. The paper's main text needs to cite all dependencies and other software packages used, including version numbers, URLs for web downloads, etc. We recommend adding a footnote with the version number and URL upon first mention of an external software package used in the paper.

It is becoming more popular for research coders to share their work via an online SVN repository, GitHub account[3], SourceForge[4], or other software distribution center, such as the Software Sustainability Institute[5]. Perhaps one option is that a paper could have a link to such a location storing any code described in the text or needed to reproduce the results.

Of course there are times when it is not possible to release the full source of code but the paper can still release sufficient details to advance the state of the art and thus should still be

[3] https://github.com/
[4] http://sourceforge.net/
[5] www.software.ac.uk

published. In that case, we strongly recommend including at least some, if not all, of the following:

- binary executables for the code, compiled for several different environments
- a link to download scripts or other material in tandem with the binary executable that can reproduce test cases from the publication and demonstrate the reported functionality
- code snippets of the most essential parts, included in the paper
- a thorough description of the code, including sufficient details so that someone could reasonably produce an imitation of the novel code that is the paper's subject
- equivalent public-domain code [29]

D. Data

A key element of reproducibility for any empirical study is the availability of data. Confirming the results obtained by a study is an important step, not just in extending or utilizing previous results, but also for the purpose of building trust in the results of a study, such as during peer review. The availability of data is thus of paramount importance if results from a scholarly publication are ever to be reproduced. The pitfall of this position is that often times important results are obtained through sensitive data that cannot be released, either due to it being proprietary data, or private information subject to disclosure limits and regulations on distribution. When dealing with proprietary data the key is identifying transformations that satisfy privacy constraints, while still maintaining the utility of the original data sets [30].

We understand that many data sets, particularly those from industry that deal with customers or private records, are sensitive and cannot be published. However, we also believe that the advancement of de-identification, statistical characterization, and synthetic data generation technologies has matured to the point that avenues for the release of some data are now possible.

1) k-anonymity: The de-identification and anonymization of data attempts to achieve this via the replacement of explicitly identifying or sensitive information, often through the use of randomized placeholders or tokens. This approach is not always sufficient, as released data can contain other information that, when linked with other data sets, can narrow down information on individuals or entities [31], [32], [33], [34] allowing the re-identification of the data set, for example by linking medical data with voter lists [34]. Privacy levels can thus be characterized as k-anonymity, i.e., if every combination of values occurring in the identifying data cannot be matched in fewer than k rows of the database, we say the data is k-anonymous [35]. In order to ensure that identity or attribute disclosures do not occur [30], [36] we must work to develop accurate models that do not rely on access to precise information in the exact data records.

Data that is k-anonymous has the feature that the probability of ever identifying a user is no more than $1/k$ through linking related data. K-anonymization suppresses data, often via the deletion of cell values and tuples, and generalizes data by substituting specific values with general ones. All records within a k-anonymized data set remain truthful, and hopefully this transformation has little impact on correctness. While the problem of optimal k-anonymity is NP-hard [37] other methods, such as greedy algorithms [31], [33] or incomplete stochastic search methods [30], [38] can be employed, but do not provide guarantees of quality and can be vulnerable to attack [30].

2) Data Integrity Preserving Transformations: Any transformation that is applied to data for the purposes of anonymization or de-identification should maintain *data integrity*, i.e. the data should have the same properties it had prior to the transformation. For analysis techniques that do not require individual records, just distributions, perturbation of values is a possibility [39], [40]. User data can be modified using a perturbing random distribution. This perturbing distribution can be used later to reconstruct the underlying aggregate distribution, without revealing individual data records.

Another method for preserving privacy while maintaining data integrity is the notion of data condensation [41]. Data condensation is a form of privacy preserving data mining. Data condensation works for streams of data while maintaining correlations between data dimensions without assuming any a priori knowledge of the entire data set. Data from incoming streams is condensed into multiple groups of predefined size. For each group some given level of statistical information is maintained, typically the mean and correlations across various dimensions. Within a group it should not be possible to distinguish individual records. Groups have a some minimum size k, the indistinguishability level, similar to the idea of k-anonymity. The greater the level, the greater the privacy, however as the indistinguishability level increases, some information is necessarily lost.

Data can also be transformed through the use of scrambling and swapping techniques [31], [42]. These techniques attempt to mask quantitative data through the use of additive-noise and the swapping of key values to provide some level of confidentiality. Any noise added should have the same correlation structure as the original data to preserve integrity. A drawback of these methods is that unusual combinations of data, or outliers, may still be re-identified. Adding noise to data [39] is a popular transformation method. Adding noise allows for value distortion to protect privacy. The key hypothesis is that introduced noise helps prevent the reconstruction of original records, but should still allow for the reconstruction of the distributions underlying those records.

A final method for transforming data while maintaining data integrity is that of association rule mining [43], [44]. Similar to the previously described techniques, association rule mining attempts to identify correlations of interest between attributes in a database of records. Noise can be added to allow for the probabilistic distortion of data using a predefined distribution function. Distorted information is then supplied to the user, along with information on the distortion procedure. These methods are typically used to preserve so called *association rules*, which represent implications with the support of statistical significance and confidence. In theory, more interesting rules are significant with high confidence and will thus be preserved by the method.

3) Synthetic Data: In lieu of transforming and releasing potentially private data, researchers can also ensure reproducible studies through the use of synthetic data [45], [46], [47]. When dealing with sparse data, patient data, or other data that requires a very high level of privacy, anonymization may be difficult or impractical [45]. The use of synthetic data revolves around the idea of building a statistical model from real data and then sampling points from this model to generate a new data set. While different from the original data the new data set resembles it in key ways, maintaining the same moments, statistical properties, or relationships. These sample points can be released instead of the original data, or even the model derived from the data, which might be very similar to data that was de-identified with the aforementioned methods. The privacy of the original data and/or model can be further increased by the introduction of noise to the data from two primary sources:

1) Bias introduced during model creation,
2) Random sampling from the model.

Synthetic data has been used for a number of studies, and in a wide range of domain problem such as in the successful detection of fraud [46]. Synthetic data works through the identification of important statistical properties, and the preservation of parameters that are important for the application at hand. Care is required to make sure the model used to generate synthetic data also contains those parameters and statistical properties. Synthetic data will necessarily result in the loss of information, and this loss can be subtle. The advantage, however, is the ability to release data specifically for the study, without direct privacy concerns [47]. Another feature of using synthetic data is that it can be used to create many sets of data from the same underlying data, adding additional evidence of reproducibility of the methods presented.

III. Correctness

Correctness is perhaps the most obvious principle for public dissemination, yet it is also the most ignored. Many times correctness is implied but not explicitly stated. Anecdotally, we often have heard when serving on program committees, 'of course the authors proved correctness; they would not have submitted for publication if not.' While we believe this is frequently true, there is a lot to be gained from explicitly requiring a demonstration of correctness.

If research is publicly disseminated, it will almost certainly be presumed to be correct. Given the prevailing culture of research that promotes trusting previous results, researchers may spend valuable time and resources building upon them instead of reproducing the original results first as a check, particularly when doing so would be particularly time-consuming or expensive [9]. Since 2000, there has been a disproportionate increase in the retraction of papers due to incorrect results. This increase has come about for a variety of reasons, and has been complicated due to the difficulty of publishing corrections in a climate that values the novelty of the topic above all. All of these trends makes the *explicit* demonstration of correctness in publications even more vital [5]. Good documentation is a necessary requirement for explicit correctness. It enables understanding of why something works and it enables reimplementation of discoveries or prior mistakes long after the actual hardware or software infrastructures needed for reproducibility become unavailable [4]. Feynman aptly described this as "a specific, extra type of integrity that is not lying, but bending over backwards to show how you're maybe wrong, that you ought to do when acting as a scientist" [9].

Proving correctness may not always be possible. Just because correctness cannot be total, does not mean it is acceptable to be absent. It should be at least *mentioned*. There needs to be some reason to believe that the results presented are correct. Correctness may sometimes be qualitative: for a case study, if the system designers declare that the work done on/for their system is correct/meaningful then one might reasonably consider it valid. Similarly, formulating the obligatory Threats to Validity section in empirical software engineering papers has developed into an art form [7]. Best attempts at correctness may range from convincing arguments or proof to collections of circumstantial evidence. We believe most researchers do check that their work is correct before public dissemination so requiring these checks be included in any publication should not be burdensome.

A. Theory

For theoretical contributions, correctness is nearly indistinguishable from reproducibility. It is our anecdotal experience that theory in computer science and engineering publications is usually quite rigorous, though there is some evidence that occurrences of measuring the wrong thing or meaninglessly measuring the right thing are on the rise [4]. We include here a check-list of helpful elements.

- All new theorems require accompanying proofs. These may appear in the paper's main text or in an appendix if there is not enough room, or on a website for the paper, accompanying tech report, or other extended format if appendices are not allowed. There is always room somewhere for proofs for all theorems; these are required content.
- Automated checks, such as completing proofs using an automated theorem prover, provide a huge gain for demonstrating correctness. Of course such checks are not always possible; for large and complicated proofs this check may constitute an entire research paper by itself.
- Any means of stating the theory in multiple ways is helpful in demonstrating correctness. Accompanying new theory with examples, a code implementation, experimental results that uphold the correctness, or any other checks can be very insightful.
- All problem statements (i.e. what the paper is trying to solve) and solution strategies (i.e. what the paper actually does) should be stated straightforwardly and rigorously. They should be accompanied by a detailed methodology including a list of what will be measured and what proof the authors have that their solution solves the problem.
- Experimental methodologies should be clearly laid out, including justification for all benchmarks, evidence that the results presented are statistically significant, and steps taken to account for any possible

measurement bias. We do not find the typical small proof of concept studies contribute to correctness.

- Experimental results should be accompanied by rigorous arguments establishing the quality of the benchmarks used.

Computer science and computer engineering are at the top of the Hierarchy of the Sciences, in terms of the rigor with which data is related to theory [23]. We argue that correctness requires both the theory and the respective data to be presented in a thorough enough manner to be able to rigorously demonstrate this relationship. We find it extremely disappointing when seemingly stellar theory papers go the extra mile by experimentally evaluating their theoretical results, but the experiment is limited to a trivial example such as the Dining Philosophers problem with four philosophers. We can do better to employ proper statistical evaluation methods with a little extra effort.

B. Algorithms

It is often quite difficult to determine if an algorithm is correct via careful review. Besides the challenge presented by algorithmic complexity, pseudocode can be opaque and space and formatting constraints can limit readability and therefore understanding. There are not always straightforward sanity checks and "contemporary computational methods often produce results which are not amenable to simple evaluations for accuracy" [17].

There are several options for demonstrating algorithm correctness, of varying levels of rigor:

- Construct a formal proof of correctness of the algorithm.
- Make a correctness argument, i.e., argue informally or based on a perturbation of a previous proof that the algorithm is correct.[6]
- Trace through the algorithm in the paper's text in detail, making a case for every path. Running examples and detailed case studies may also help serve this function.
- Implement the algorithm, validate the code implementation, and demonstrate correctness of that code.

Some algorithms may require still more creative descriptions to ensure correctness; [17] discusses the relatively subjective nature of algorithms for image and signal processing versus algorithms that produce comparable values that are easier to check. A case for correctness of the former may emphasize the link between input parameters and algorithm output.

C. Code

Code correctness is a prominent and active research area in its own right [49]. Here we provide tips for increasing the confidence that code published in other research areas is correct, with the caveat that all of these suggestions could be improved upon via applying the latest research from that area. It is often not possible, due to space, time, complexity, or other constraints, to conclusively prove correctness of code releases; not all code should be formally verified before publication. After all, formally verifying all software, algorithms, and theory in a paper could produce another (or several other) papers. However, this is not an excuse to ignore correctness entirely.

Good practices for demonstrating code correctness include:

- Run an automated verification tool on the code, such as a code-level model checker or static analyzer, or a debugger that checks for basic assertions.
- Compare the output of the code to output from other code that purports to do the same thing, at least for the given test cases. Good comparators include a simpler version of the code, a simpler algorithm implementation, known benchmarks, and the previous best tool.
- Release the source of the code. Note that this does not directly demonstrate correctness but it enables others to verify correctness for their own purposes.
- Post automated tests online that reviewers can run easily during the review process [26].
- For tools, include a link to a software development page for the code with a bug tracking system where readers can transparently see the latest version of the code, if a package is being actively maintained, and whether earlier versions suffered from any significant bugs [26].
- Include a 'quality control' section in the main text explaining how and why any code developed for the paper can be trusted [27], [28].

Easy sanity checks are often absent. While checking size or run time or other performance characteristics, we strongly encourage checking the following questions:

- **When comparing run times against another tool, did both tools return the same answer for each of those run times?**
- Does the tool always have the same high-level behavior? Does the output fall into the expected region of answer-space? Does it follow the same pattern? These basic sanity checks apply even if exact answers are not comparable.
- Does the new, more complicated code implementation always return the same basic behaviors as a simpler variant of the same system?[7]

For example [51] compares tools for encoding temporal logic formulas in the context of satisfiability checking. This is important, for the same formula different encodings may lead to different search orders that may lead to different specific answers as a formula may be satisfiable for multiple valuations of the variables. However, the high-level answer of whether or not a single formula is satisfiable or not should always

[6] See [48] for an example.

[7] Such a technique was used for code verification of the data-management module of the Curiosity Mars Science Lab (MSL) [50], [29].

be the same for every encoding of the formula. To evaluate correctness this paper implements a simple voting strategy: for every formula, check the encoding in every existing tool and compare the results of satisfiable or not. Shockingly, this simple sanity check uncovered errors in every single one of the nine encodings in the *explicit* category![8]

Experimental results sections should have a statistically significant number of experimental results. Including a small experiment like Dining Philosophers with four philosophers does not demonstrate correctness. Certainly it can be a useful example and may be valuable to aid reproducibility, but not correctness.

D. Data

The primary issue with correctness and data is providing data and example outputs so that users of the techniques can build confidence in their use of provided algorithms, tools, and methods. As such in addition to the data sets used to perform the original experiments, the results from these experiments, and the data used to generate them, should be given. This means that documentation of the data and methods is essential. Researchers should endeavor to document their process and data used just as they document their code, to allow reviewers and colleagues to easily build trust in the correctness of their results.

IV. Buildability

Buildability combines the many principles of utility, usability, extensibility, and having a foundational nature. Buildability provides the basis for future research and enables technology transfer, a required element for many government grants. Checks for buildability include:

- Can others use, build upon, or extend the work in the future without reinventing or rediscovering it in whole or in part?
- Has the work reached a minimum completeness level that it is usable in some way? Has the research developed past the idea phase?
- Is there a reasonable expectation that the results of the research can be used for the stated purpose? There is a level of trust inherent to buildability. Is there some reason to believe such trust is warranted?
- Is the research released in such a way that others can build upon it, extend it, or utilize it, either as a white box, black box, or gray box?

The original point of publishing was to share research so that others may build upon it for the benefit of society [8], [20]. Prior to that, scientific discoveries were encoded and hidden to advance the fame of the scientists at the expense of scientific progress. The use of forceps for the delivery babies, for example, was kept as a proprietary secret for over 150 years. In that time, countless women and babies died due to the lack of widespread availability of so simple a procedure [52]. We argue this makes buildability a fundamental principle of public dissemination that should be valued as highly in the publication process as the much more common checks that the research topic is novel and interesting.

Note that publishing negative results, as well as positive ones, is important for buildability. Negative results prevent others from wasting their time on paths that have been shown not to work and prevent future researchers from building on unstable foundations.

Though not by the name 'buildability' the concept is supported in literature. The "four conditions for ethical research" [17] include respect for resources, research sponsors, colleagues, and the integrity of the scientific community, which essentially constitute buildability. Prior work provides a foundation for others to build upon as well as information and inspiration for further research and development. Research or publication practices that provide an unstable foundation, misinforms scientific colleagues about essential factors, obscure or omit important details, or otherwise compromise the ability of others to work from the published results wastes time, disrespects colleagues and resources, and hinders scientific progress.

A. Theory

Vardi stated, "A scientific theory is an explanatory framework for a body of natural phenomena. A theory can be thought of as a model of reality at a certain level of abstraction. For a theory to be useful, it should explain existing observations as well as generate predictions, that is, suggest new observations. In the physical sciences, theories are typically mathematical in nature ..." [25]. As buildability implies both that the theory is useful and extensible, we argue that this principle extends a bit beyond simply showing the math:

- Theorems with proofs consisting of statements like 'The proof is obvious' or 'left as an exercise for the reader' do not contribute to buildability. An appropriate exercise for the reader would be to devise an alternative proof; it is wholly in appropriate to leave the burden of establishing there is a proof for future work. Subtle details in simple proofs are often important for the deeper level of understanding required to build upon new work. Any theorem enumerated in a publication is not so obvious that it is not worth stating; theorems worth stating by definition have some non-obvious proof aspects. Maybe the proof is not ground-breaking enough to warrant space in a page-limited publication but we argue that the proofs of *every* theorem need to appear somewhere: an appendix, an online version, the authors' websites, etc. if not in the paper's main text. We cannot ignore the fact that mistakes are also made in the simplest steps of a problem. In a similar vein, we argue posting expanded versions of especially terse proofs e.g. of the form 'the proof stems from the ten previously proved lemmas.'
- Proofs from automated theorem provers provide a huge gain for buildability. Such proofs are not only made useful but easily extensible by providing either the program or proof script required to recreate the

[8]It is important to note that since publication, one encoding was corrected and later correct encodings followed. We hope for similar success stories in other areas.

proof, printed in an appendix or linked online, or sufficient details to recreate the proof including the setup, i.e., the statement of the proof for an automated theorem prover, plus the exact version of the prover that was used, references for the required libraries, the proof strategy, etc.

- Online scientific notebooks also contribute to buildability; by posting the authors' working notes online it is easier to see how the theory was derived and develop a better understanding of both why it is correct and the trajectory in which the authors were working [8].

B. Algorithms

Algorithms are inherently great foundations to build upon. Their buildability can be maximized in the following ways:

- Ensure the algorithm listing is complete, including small details like the types of the inputs, recommended data structures, and comments. Explicitly include assumptions, label clever tricks. Make it clear which parts of the algorithm are sensitive to ordering, dependent on specific subroutine implementations, or otherwise have relationships that need to be preserved to maintain correctness.
- When possible, include information about what may be easily changed, such as labeling parts of the algorithm that may be optimized or parallelized.
- Write clear documentation.
    - Use descriptive variable, function, and subroutine names
    - Focus on a readable presentation and a clear, intuitive layout
    - Include comments in any pseudocode, in-text code snippets, or accompanying code
    - Include multiple forms of the algorithm i.e. show the algorithm in two different languages or formalisms. Combine some subset of pseudocode, in-text code snippets, English descriptions, and the full code. Place these resources online or in an appendix if they do not fit in the paper proper.
- Consider ways to present the algorithm intuitively, e.g., by stepping through the algorithm with an example scenario and discussing the outcomes.

C. Code

Certainly it is not reasonable to expect production-quality code; research prototypes featured in publications will likely not meet high performance standards for example, but it is still reasonable to expect that published code meet certain minimal standards for buildability necessary for public dissemination. Boettiger argues, "Sustainability of the software is important for the same reason archiving the literature is important — without this, we cannot build on existing work. Building on researcher/developer software (e.g. the kind of software typically covered in software papers) is currently an uncommon and risky [endeavor]" [28].

We consider the following requirements for code buildability:

- If the code is a software package or part of one, or if it's an extension to an existing tool (as opposed to a code snippet), list information needed to run it, including dependencies with version numbers, a license, documentation, and instructions for running automatic checks [26].
- Take steps to make the code maximally readable: comment code, create re-usable modules, use descriptive naming schemes and standardized coding style, structures, spacing, and labeling. Make it as easy as possible for a reader to understand what a function should do and test that it does what it says. Stable, clean, and complete return objects aid in buildability as do following buildable styles for including package dependencies [26].
- Consider 'literate programming' or combining code and complete documentation in the same core file that can also be used to automatically generate either part separately [17].
- Clearly label optional arguments that affect the running modes of your code or calls to preexisting libraries/tools. Consider calling these out through explicitly passing all arguments in function calls, including organization schemes for long argument lists for clarity, and avoiding relying on default values [26]. This will enable others to understand, modify, and build upon future versions more easily.

Buildability is limited when the code cannot be released at all. However, there are some actions that can still be taken to increase buildability in that case, including releasing a binary executable of the code that implements the following:

- The ability to separately execute each function or part of the code, e.g., code that implements `OpX` then `OpY` then `OpZ` should also be able to perform only `OpX` and allow the user to inspect the output then call `OpY` on that output and again inspect the intermediate results before executing `OpZ`. Note that this also allows future users to substitute newer implementations of any of these functions.
- A configurable, parameterized front end that enables easily switching on or off all componentizable features of the code independently.
- Good documentation, including a `--help` screen or equivalent.
- A set of meaningful use-case examples or test cases to aid in producing different inputs for the code.

Often license options are determined by the authors' institutions but if there is an option, opting for a more open license certainly aids buildability.

D. Data

Releasing the data used for a study, or documenting the means by which it was obtained from a third party data clearing

house, contributes to the principle of buildability. Open sources of data represents an ideal for buildability, given the lack of associated restrictions. Researchers should strive to make contributions to open data sets to ensure ease of access by those who would extend their work, or conduct new research using the results, and to use existing open data-sets when possible.

One of the biggest challenges facing the principle of buildability as it applies to data is finding long term solutions for storing and hosting the data. Maintaining perpetual availability of data is a difficult task that has been studied extensively by advocates of Open Access [53]. The same principles that were defined by the Budapest Open Access Initiative also apply to the ideals of access to data for buildability of research [54]:

> There are many degrees and kinds of wider and easier access to this literature. By *open access* to this literature, we mean its free availability on the public Internet, permitting any users to read, download, copy, distribute, print, search, or link to the full texts of these articles, crawl them for indexing, pass them as data to software, or use them for any other lawful purpose, without financial, legal, or technical barriers other than those inseparable from gaining access to the Internet itself. The only constraint on reproduction and distribution, and the only role for copyright in this domain, should be to give authors control over the integrity of their work and the right to be properly acknowledged and cited.

This sentiment was expanded upon by the Bethesda Statement on Open Access Publishing [55], and the Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities [56] which emphasize the need for others to have the ability to copy, use, and distribute information and works, as well as the ability to make derivative works, and distribute those as well. This is the very essence of the principle of buildability. Data should be stored reliably, so that it is tolerant to faults, and widely available. This factor also raises the concern of funding repositories. The success of the Free and Open Source Software movement provides good evidence that, while difficult, these challenges are not insurmountable.

V. Conclusion

As Feynman famously said, "The idea is to try to give *all* of the information to help others to judge the value of your contribution; not just the information that leads to judgment in one particular direction or another" [9]. Reproducibility, correctness, and buildability constitute a foundation for publicly disseminating all of the information necessary to judge a research contribution and work in tandem to uphold other values such as novelty. We hope future calls for papers or proposals will specifically mention which of the three principles, or which combination, they are expecting. We can use this vocabulary to make the qualitative judgment of papers for review a bit more predictable and objective and mitigate the increasing perception of PC unfairness [3], [2]. As computer science and engineering researchers we should be specifically targeting our publications at a point in the overlapping space of reproducibility, correctness, and buildability, displayed in Figure 1. The *best* papers would be in the three-way intersection. However, a fruitful workshop might ask for only reproducibility and partial correctness but not yet full correctness or buildability; after all, the audience may just need to understand what was done so far and be able to give ideas to work out correctness for corner-cases and complete the work-in-progress, which is not yet expected to be complete, much less proved correct or ready for others to build upon yet. A successful tool paper might focus instead on buildability and correctness, providing evidence that the tool functions as claimed, and the necessary basis to use the tool for future research.

Though all three principles may not be required for every venue for public dissemination of research results, having none of these three would seem to present no evidence that any work was performed at all. If the work being done is so secret that it is really impossible to meet even one of the three principles then this seems to bring up the question: is it ethical to publicly disseminate the work at all? Either the content of the research is so secret that it cannot be released to the public or the work was not completed to a sufficient state to enable even one of the three principles yet public dissemination would serve to imply a more mature status of the work than is the case. Both of these would seem to raise the question of whether pubic dissemination is ethical.

We make no claim that the methods presented in this paper are novel; in fact all have been previously proposed by others and some reviewers and conferences already enforce them, if only by implication. What is new is our organization of the three principles, collecting the methods for establishing them into one, easily referenceable place, and arguing for the explicit classification of publications and reviews by such a checklist.

The space of scientific advancements in computer science and engineering is vast and we cannot hope to cover all special cases of research publications here. However, we believe that the majority of works fall into the categories we have listed. The special cases we have not covered still benefit from establishing a common vocabulary for reviews, even if their methods of establishing reproducibility, correctness, and buildability are not explicitly listed in this paper.

A. Future Outlook

In addition to the narrative critiquing the finer points of the submitted article, peer review forms standardly require numerical ratings of the overall evaluation and reviewer's confidence. For example, conferences conducting peer reviews through EasyChair [57] ask reviewers to select a rating from some subset of {strong accept, accept, weak accept, borderline, weak reject, reject, strong reject} with a confidence from some subset of {expert, high, medium, low, none}. Occasionally, conference chairs add their own additional numerical ratings for such qualities as originality, strength of contributions/technical significance, quality of writing/readability, originality, organization, technical consistency, relevance to the conference/level of interest in the topic/importance, coverage of related work, and completeness/thoroughness.

While all of these traits are important – indeed we would argue readability is essential to all three of our principles – we argue that reproducibility, correctness, and buildability should be included as standard numerical ratings on peer review forms, to the extent they are applicable to the venue. Often

these three principles are discussed in the narrative, sometimes explicitly and sometimes implicitly, but we argue this is not enough. We believe it would be pertinent for a conference that calls for reproducible results to include reproducibility explicitly as a numerical rating used in calculating paper rankings for deciding acceptance, and similarly for correctness and buildability. In order to build their scores in these categories, authors might start including sections of their paper explicitly named for these principles and pointing to how they are fulfilled. This may also help increase the quality, consistency, and fairness of the selection process. We envision similar adaptations for the process of reviewing research proposals and writing white papers/proposals for grants. Today, NASA research solicitations generally require all scientific data and resulting technical information to be committed into publicly accessible data archives for use by the scientific community and the public at no cost to them, but other funding agencies, such as the NSF, are less stringent [58].

Many of the options we present for demonstrating reproducibility, correctness, and buildability depend on disseminating information outside of the page limits of a conference publication. We argue this is a reasonable expectation in the Internet age, where electronic dissemination of papers alongside appendices or websites or other supplemental information sources is easy and usually free. We recall [20] that "the Internet was created to help scientists share their research. It seems overdue that scientists take full advantage of its original purpose." There are many avenues to ensuring the space needed: including requiring an accompanying technical report, creating an online version of the paper, requiring every paper to have a URL for a website with all necessary materials posted, or providing a common space online, perhaps even on the conference website, for these materials to be uploaded. Each conference could have an electronic extended journal version of each paper with sections for reproducibility, correctness, and buildability. A similar idea has been advocated by CRA [59] and is implemented in a limited fashion by arXiv.org, though their data repository was discontinued in 2013. The future value of publishers like Springer and Elsevier may be through providing paper code checks, along the lines of the recent Mozilla Science Lab experiment [60]. We can even envision a future where online versions of papers are more interactive, providing the ability to reproduce every figure with the click of a mouse, for example [16].

While large-scale failures of the review process are extremely rare [21], [5], [22], we argue that explicitly enumerating standards for our three principles will help to decrease the probability of such an occurrence while helping to lift the average overall quality of publications. It seems the Journal of Statistical Software has already moved in this direction. Papers submitted there are reviewed for "correctness and usefulness" (which sounds to us like buildability) and are required to be accompanied by commented, readable source code with instructions enabling reproducibility of all results from the manuscript [61]. The Journal of Open Research Software has a similar policy [27]. We argue that since code and data pervade all areas of computer science and engineering publications, similar review processes should be standard for conference publications.

We hope that this discussion helps to establish a common vocabulary for evaluating the ethical public dissemination of computer science and engineering research and provides tools for more explicit evaluation of the three principles of reproducibility, correctness, and buildability in publications and research proposals.

ACKNOWLEDGMENT

The authors would like to thank our proofreaders, Dorilyn Martz Ames, Linda Davis, Shravan Gaonkar, Tony Kaap, Moshe Vardi, and Saman Zonouz, for their input.

REFERENCES

[1] M. Y. Vardi, "Conferences vs. journals in computing research," *Commununications of the ACM*, vol. 52, no. 5, p. 5, 2009. [Online]. Available: http://cacm.acm.org/magazines/2009/5/24632-conferences-vs-journals-in-computing-research/fulltext

[2] J. Crowcroft, S. Keshav, and N. McKeown, "Viewpoint: Scaling the academic publication process to internet scale," *Communications of the ACM*, vol. 52, no. 1, pp. 27–30, January 2009. [Online]. Available: http://cacm.acm.org/magazines/2009/1/15664-viewpoint-scaling-the-academic-publication-process-to-internet-scale/fulltext

[3] K. Birman and F. B. Schneider, "Program committee overload in systems," *Communications of the ACM*, vol. 52, no. 5, pp. 34–37, May 2009. [Online]. Available: http://cacm.acm.org/magazines/2009/5/24644-program-committee-overload-in-systems/fulltext

[4] J. Vitek and T. Kalibera, "Repeatability, reproducibility, and rigor in systems research," in *Proceedings of the ninth ACM international conference on Embedded software*, ser. EMSOFT '11. New York, NY, USA: ACM, 2011, pp. 33–38. [Online]. Available: http://doi.acm.org/10.1145/2038642.2038650

[5] P. staff, "Fraud in the ivory tower," Dec. 2013. [Online]. Available: http://priceonomics.com/fraud-in-the-ivory-tower/

[6] E. Yong, "Replication studies: Bad copy," *Nature*, vol. 485, p. 298300, May 2012. [Online]. Available: http://www.nature.com/news/replication-studies-bad-copy-1.10634\#Graphic

[7] B. Meyer, "The modes and uses of scientific publication," *BLOG@CACM*, September 2011. [Online]. Available: http://bertrandmeyer.com/2011/11/22/the-modes-and-uses-of-scientific-publication/

[8] M. Nielsen. (2008, July) The future of science. Blog based on a keynote talk at the New Communication Channels for Biology workshop. [Online]. Available: http://michaelnielsen.org/blog/the-future-of-science-2/

[9] R. P. Feynman, "Cargo cult science," *Engineering and Science*, vol. 37, no. 7, pp. 10–13, 1974.

[10] B. Meyer, "Conferences: Publication, communication, sanction," *BLOGCACM*, January 2013. [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/159380-conferences-publication-communication-sanction/

[11] I. Newton, *The Correspondence of Isaac Newton: Edited by HW Turnbull,*. Grande Bretagne, University Press, 1959.

[12] S. Williams, *Free as in Freedom [Paperback]: Richard Stallman's Crusade for Free Software*. O'Reilly Media, Inc., 2011.

[13] H. F. Korth, P. A. Bernstein, M. Fernandez, L. Gruenwald, P. G. Kolaitis, K. McKinley, and T. Ozsu, "Paper and proposal reviews: is the process flawed?" *ACM SIGMOD Record*, vol. 37, no. 3, pp. 36–39, 2008.

[14] M. D. Ernst, "Re: conferences and journals in computer science," Open letter to the community, November 2010, Department of Computer Science & Engineering, University of Washington.

[15] J. B. Buckheit and D. L. Donoho, *Wavelab and reproducible research*. Springer, 1995.

[16] C. Laine, S. N. Goodman, M. E. Griswold, and H. C. Sox, "Reproducible research: moving toward research the public can really trust," *Annals of Internal Medicine*, vol. 146, no. 6, pp. 450–453, 2007.

[17] P. A. Thompson and A. Burnett, "Reproducible research," *CORE Issues in Professional and Research Ethics*, vol. 1, 2012, (Paper 6). [Online]. Available: https://nationalethicscenter.org/content/article/175

[18] J. Claerbout. (2011) Reproducible computational research: A history of hurdles, mostly overcome. [Online]. Available: http://sepwww.stanford.edu/sep/jon/reproducible.html

[19] B. Baumer, M. Cetinkaya-Rundel, A. Bray, L. Loi, and N. J. Horton, "R markdown: Integrating a reproducible analysis tool into introductory statistics," *Technology Innovations in Statistics Education*, vol. 8, no. 1, pp. 1–29, 2014.

[20] A. Mayyasi. (2013, May) Why is science behind a paywall? [Online]. Available: http://blog.priceonomics.com/post/50096804256/why-is-science-behind-a-paywall

[21] J. Timmer, "Anti-gmo crop paper to be forcibly retracted: Journal editor recognizes extensive flaws, says the paper shouldn't have run," *Scientific Method/Science & Exploration*, Dec 2013. [Online]. Available: http://arstechnica.com/science/2013/12/anti-gmo-crop-paper-to-be-forcibly-retracted/

[22] A. Gelman, "Science journalism and the art of expressing uncertainty," *Symposium Magazine*, Aug 2013. [Online]. Available: http://www.symposium-magazine.com/science-journalism-and-the-art-of-expressing-uncertainty/

[23] D. Fanelli, "Positive results increase down the hierarchy of the sciences," *PloS one*, vol. 5, no. 4, p. e10068, 2010.

[24] Y. Zhao and K. Y. Rozier, "Formal specification and verification of a coordination protocol for an automated air traffic control system," *Science of Computer Programming Journal*, p. To appear, 2014.

[25] M. Y. Vardi, "Science has only two legs," *Communications of the ACM*, vol. 53, no. 9, p. 5, September 2010. [Online]. Available: http://cacm.acm.org/magazines/2010/9/98038-science-has-only-two-legs/fulltext

[26] C. Boettiger, "What I look for in 'software papers' – pet peeves and faux pas," *Blog: Lab Notebook*, June 2013. [Online]. Available: http://carlboettiger.info/2013/06/13/what-I-look-for-in-software-papers.html

[27] S. M. (Ed.), "Journal of open research software," Online: Software Sustainability Institute, Ubiquity Press, 2014, http://openresearchsoftware.metajnl.com/.

[28] C. Boettiger, "Reviewing software revisited: Great ideas from the journal of open research software," *Blog: Lab Notebook*, Jul 2013. [Online]. Available: http://carlboettiger.info/2013/07/09/reviewing-software-revisited.html

[29] G. J. Holzmann, "Mars code," *Communications of the ACM*, vol. 57, no. 2, pp. 64–73, 2014.

[30] V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 279–288.

[31] A. Hundepool and L. Willenborg, "μ-and τ-argus: Software for statistical disclosure control," in *Third International Seminar on Statistical Confidentiality*, 1996.

[32] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," Technical report, SRI International, Tech. Rep., 1998.

[33] L. Sweeney, "Guaranteeing anonymity when sharing medical data, the datafly system." in *Proceedings of the AMIA Annual Fall Symposium*. American Medical Informatics Association, 1997, p. 51.

[34] P. Samarati, "Protecting respondents identities in microdata release," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 13, no. 6, pp. 1010–1027, 2001.

[35] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE, 2005, pp. 217–228.

[36] D. Lambert, "Measures of disclosure risk and harm," *Journal of Official Statistics-Stockholm*, vol. 9, pp. 313–313, 1993.

[37] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2004, pp. 223–228.

[38] W. Winkler, "Using simulated annealing for k-anonymity," Research Report 2002-07, US Census Bureau Statistical Research Division, Tech. Rep., 2002.

[39] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *ACM Sigmod Record*, vol. 29, no. 2, pp. 439–450, 2000.

[40] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2001, pp. 247–255.

[41] C. C. Aggarwal and S. Y. Philip, "A condensation approach to privacy preserving data mining," in *Advances in Database Technology-EDBT 2004*. Springer, 2004, pp. 183–199.

[42] J. J. Kim, W. E. Winkler *et al.*, "Masking microdata files," in *Proceedings of the Survey Research Methods Section, American Statistical Association*. Citeseer, 1995.

[43] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Information Systems*, vol. 29, no. 4, pp. 343–364, 2004.

[44] S. J. Rizvi and J. R. Haritsa, "Maintaining data privacy in association rule mining," in *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 682–693.

[45] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory meets practice on the map," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008, pp. 277–286.

[46] E. L. Barse, H. Kvarnstrom, and E. Jonsson, "Synthesizing test data for fraud detection systems," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 384–394.

[47] D. B. Rubin, "Statistical disclosure limitation," *Journal of Official Statistics*, vol. 9, no. 2, pp. 461–468, 1993.

[48] D. Giannakopoulou and F. Lerda, "From states to transitions: Improving translation of LTL formulae to Büchi automata," in *FORTE, Proc of 22 IFIP Int'l Conf*, Nov 2002. [Online]. Available: citeseer.ist.psu.edu/giannakopoulou02from.html

[49] V. D'silva, D. Kroening, and G. Weissenbacher, "A survey of automated techniques for formal software verification," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 7, pp. 1165–1178, 2008.

[50] G. J. Holzmann and R. Joshi, "A mini grand challenge: build a verifiable file-system," *position paper), Grand Challenge in Verified Software–Theories, Tools, Experiments, Zurich, Switzerland*, 2005.

[51] K. Rozier and M. Vardi, "LTL satisfiability checking," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 12, no. 2, pp. 123 – 137, March 2010. [Online]. Available: http://dx.doi.org/10.1007/s10009-010-0140-3

[52] W. Moore, "Keeping mum," *BMJ*, vol. 334, no. 7595, pp. 698–698, 3 2007.

[53] P. Suber, "Open access overview," *Retrieved from Peter Suber's website: http://www. earlham. edu/~ peters/fos/overview. htm*, 2007.

[54] L. Chan, D. Cuplinskas, M. Eisen, F. Friend, Y. Genova, J.-C. Guédon, M. Hagemann, S. Harnad, R. Johnson, R. Kupryte *et al.*, "Budapest open access initiative," 2002.

[55] P. Suber, P. O. Brown, D. Cabell, A. Chakravarti, B. Cohen, T. Delamothe, M. Eisen, L. Grivell, J.-C. Guédon, R. S. Hawley *et al.*, "Bethesda statement on open access publishing," 2003.

[56] B. Erklärung, "Berlin declaration on open access to knowledge in the sciences and humanities," *Zugriff am*, vol. 9, p. 2011, 2003.

[57] A. Voronkov. (2002) Easychair conference system. [Online]. Available: http://easychair.org/

[58] S. DeWitt and D. Brown, "Ensuring scientific integrity at the national aeronautics and space administration," http://www.nasa.gov/pdf/611201main_NASA_SI_Policy_12_15_11.pdf, December 2011.

[59] H. F. Korth, P. A. Bernstein, M. Fernandez, L. Gruenwald, P. G. Kolaitis, K. McKinley, and T. Ozsu, "Paper and proposal reviews: Is the process flawed?" *SIGMOD Rec.*, vol. 37, no. 3, pp. 36–39, Sep. 2008. [Online]. Available: http://doi.acm.org/10.1145/1462571.1462581

[60] E. C. Hayden, "Mozilla plan seeks to debug scientific code: Software experiment raises prospect of extra peer review," *Nature*, vol. 501, no. 7468, Sept 2013. [Online]. Available: http://www.nature.com/news/mozilla-plan-seeks-to-debug-scientific-code-1.13812

[61] J. de Leeuw, B. Grün, and A. Z. (Eds.), "Journal of statistical software instructions for authors," Online: American Statistical Association, 2014, http://www.jstatsoft.org/instructions.