A Computational Reproducibility Benchmark

Fernando Chirigati[1] Matthias Troyer[2] Dennis Shasha[3] Juliana Freire[1,3]
[1]Department of Computer Science and Engineering, New York University
[2]Institute for Theoretical Physics, ETH Zürich
[3]Courant Institute of Mathematical Sciences, New York University
{fchirigati,juliana}@nyu.edu, troyer@phys.ethz.ch, shasha@courant.nyu.edu

Abstract

Creating and testing reproducible computational experiments is hard. Researchers must derive a compendium that encapsulates all the components needed to reproduce a result. Reviewers must unpack the encapsulated components, run them in an environment that could be different from the source environment, and verify the results. Although many tools support some aspect of reproducibility, there is no common benchmark against which single or multiple tools can be tested. This paper describes a benchmark that can be used to categorize and better understand existing systems. The benchmark will also serve as the basis for a competition whereby tool builders will demonstrate if and how their systems support end-to-end reproducibility.

1 Motivation

Ever since Francis Bacon, a hallmark of the scientific method has been that experiments should be described in enough detail so that they can be repeated and perhaps generalized. When Newton said that he could see farther because he stood on the shoulders of giants, he depended on the truth of his predecessors' observations and the correctness of their calculations. In computational terms, this implies the possibility of (i) repeating (or *replicating*) results on nominally equal configurations, and (ii) generalizing the results by replaying them on new data sets, verifying how they vary with different parameters, and re-using and extending the experiment.

In principle, reproducibility should be easier for computational experiments than for natural science experiments, because not only can computational processes be automated, but also computational systems do not suffer from the "biological variation" problem that plagues the life sciences. Unfortunately, the state of the art belies this apparent ease. Most computational experiments are specified only informally in papers, where experimental results are briefly described in figure captions; the code that produced the results is seldom available and may be tied to specific configurations; and configuration parameters change results in unforeseen ways.

The lack of reproducibility has serious implications and has led to a credibility crisis in computational science [5]. In the absence of reproducibility, it has become difficult and sometimes impossible to verify scientific results, sometimes leading to major mistakes that are corrected only long after they are published, if ever. Furthermore, scientific discoveries do not happen in isolation. Important advances are often the result of sequences of smaller steps. If results are not fully documented, reproducible, and generalizable, it becomes hard to re-use and extend them.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Recently, there has been great interest on reproducibility and in the publication of reproducible results [2, 6, 7, 11, 14, 15, 16, 19, 23]. A number of conferences and journals instituted a reproducibility review process [1, 13, 21, 24], and while some have started to encourage authors to submit the experiments together with their papers, others have this as a requirement [17, 20]. However, these efforts have had limited success. A major roadblock to the more widespread adoption of this practice is the fact that it is hard to derive a compendium that encapsulates all the components (e.g., data, code, parameter settings) needed to reproduce the results, publish and verify them. Indeed, many scientists do not make their data and experiment reproducible [22], because authors complain that the process is too laborious [3].

While there are tools that support reproducibility, these have often been developed in isolation and target specific communities. There is also confusion about what the end-to-end process to attain reproducibility entails. As a result, tools lack important features and cannot be easily integrated with other systems that support these features. Researchers in search of a reproducibility solution are thus left in a quandary, since it is both hard to identify the tools they need and understand the functionality they support.

As a starting point to address this problem, this paper makes two contributions. First, we characterize the key tasks involved in the lifecycle of reproducible results, as well as different modes for attaining reproducibility. We then propose a benchmark that exercises these tasks in one mode that we call *spontaneous*. We have selected scenarios that are common in a variety of scientific domains. We also describe set of criteria to evaluate the benchmark results. Besides serving as a tool to categorize and better understand reproducibility systems, this benchmark will be used in an upcoming competition, where tool builders will implement end-to-end solutions for reproducibility and these will be tested by judges. We hope that through this benchmark and competition, insights will be obtained as to how to build comprehensive and general solutions.

2 Creating and Reviewing Reproducible Papers

In reproducible papers, the results reported, including data, plots and visualizations are linked to the experiments and inputs. Having access to these, reviewers and readers can examine the results, then repeat or modify an execution. Figure 1 shows one reproducible paper. In this paper, all plots have *deep captions* consisting of the workflow used to derive the plot, the underlying libraries invoked by the workflow, and the input data. This information allows the plots to be reproduced. In what follows, we describe tasks required to create reproducible experiments which can be published and shared.

Reproducibility Tasks. The first task is to (i) *create a description of the experiment*. A reproducible experiment must contain the description of the data used, the specification of the experiment – the steps followed, the underlying code needed to execute the specification, and the description of the environment where the experiment was executed. With this description, it should be possible for the author of the experiment to reproduce the experiment at a later time. However, to publish or share the experiment, the author must also (ii) *package all the components of the experiment* so that it can be executed by others in different computational environments. Finally, when creating a reproducible paper, the author needs to (iii) *connect the published results to the experiments*. Once a reproducible paper is submitted (or published), a reviewer should be able to unpack and run the experiments so that she can (iv) *reproduce and validate the results*.

Axes of Reproducibility. We have identified three distinct criteria to characterize experiments with respect to the level of reproducibility [9]: transparency, portability, and coverage. The *transparency* indicates whether partial or complete data and code are available. There are many possibilities for each step of an experiment pipeline, including: (a) partial data, e.g., a set of figures presented in a manuscript; (b) all data; (c) all data data plus the executable scripts; (d) the software system as a white box (source, configuration files, build environment) or black box (executable) on which the pipeline step is performed.

A second criterion is *portability*. An experiment can potentially be reproduced (a) on the original environment (basically, the author of the experiment can replay it on his or her machine); (b) on a similar environment (i.e., same OS but different machine), or (c) on a different environment (i.e., on a different OS and machine).

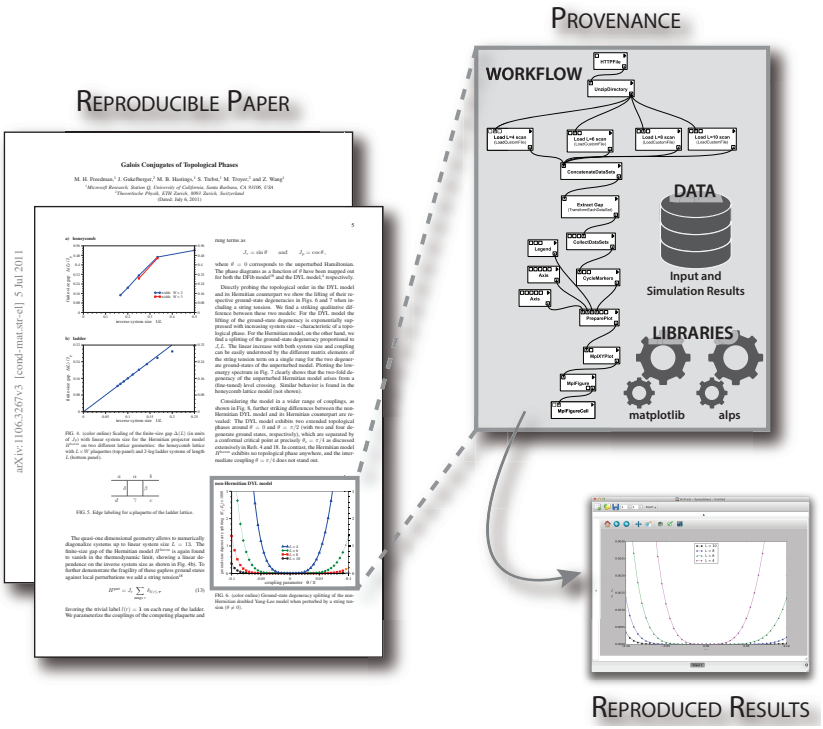

Figure 1: A reproducible paper. This paper by Freedman et al. [8] contains provenance-rich figures that have been created using the VisTrails system. Clicking on a figure downloads the workflow instance and associated provenance needed to derive the figure. This information can be examined and executed in VisTrails, reproducing the plot shown in the figure.

Finally, *coverage* indicates how much of the experiment pipeline is provided: (a) partial pipeline, i.e., only a subset of the experimental pipeline from raw data to final figures can be reproduced, or (b) full pipeline, i.e., the entire pipeline, from raw data all the way up to document, can be reproduced. As an example, experiments that rely on data derived by third-party Web services, special hardware, or lab experiments may not be fully reproduced, because repeating all the steps may be impossible. At that point, coverage would be reduced to intermediate data alone. Note that transparency indicates whether for a step of the pipeline data and/or code are available, whereas coverage has to do with how many steps of the pipeline are available.

Reproducibility Methodology. There are two main modes of creating reproducible experiments: (i) one can *plan for reproducibility* while doing research, or (ii) one can make the experiments reproducible *after the fact*, i.e., after they have been developed on a source machine. For the former, there are both best practices (e.g., the use of version control systems) as well as specialized tools that systematically capture provenance as the experiments are carried out. One advantage of this mode is that not only it ensures that the derived results are reproducible, but also this provenance captures the different trails followed, and can serve as a guide to researchers who want to understand the different choices that were examined [10], e.g., data input and parameter value combinations, or different algorithms used. However, when appropriate tools are not available, or they are simply not adopted, it should still be possible to make experiments reproducible, which is the goal of the second mode, called *unplanned*, or spontaneous. For a given set of results, this entails the creation of an executable description of how the results were derived.

3 The Reproducibility Benchmark

Desiderata. We had a number of goals while designing this benchmark. Besides verifying the ability of tools to perform the different reproducibility tasks, we would also like to understand how well these are supported and under which conditions. Thus, to serve as the basis of the benchmark, it is critical to use an experiment that is representative and covers many of the steps that are common in scientific exploration. Furthermore, the experiment must be expandable, in the sense that it should be possible to tweak it to match different reproducibility methodologies, scenarios (e.g., single and multiple machines), and levels of reproducibility. Last, but not least, to be inclusive, the experiment should also run on different multiple OS platforms.

The Experiment. We selected a computational experiment that includes both the execution of a simulation and the analysis of its results: Monte Carlo simulation of the Ising model, described in detail in [18]. The experiment pipeline consists of three main steps:

1. *Simulation Phase.* First, large-scale simulations are prepared and executed, resulting in the raw simulation output. This phase is commonly time-consuming and not easily reproducible by a reader of the paper. Thus, the output is often archived, as well as all steps to reproduce this data.
2. *Evaluation Phase.* Next, the data is analyzed and evaluated. As an example, the data is plotted in figures that allows readers to judge the quality of the derived results.
3. *Publishing Phase.* Finally, both the figures and other results are included in a manuscript.

Such workflows are common in many scientific domains and exercise important requirements for the different reproducibility tasks. The experiment can run on Windows, Linux and OS X, and it can also be tuned to use different computational environments (e.g., single machine and cluster).

Reproducibility Modes and Evaluation Criteria. To consider the two reproducibility methodologies and the tools that handle them, the benchmark has two categories: *unplanned*, which tests solutions for making an existing experiment reproducible, even though the experiment was never designed with reproducibility in mind; and *planned*, which simulates the creation of an explicitly reproducible experiment from scratch. In both cases, the goal is to evaluate the completeness of the tool with respect to the reproducibility tasks, as well as the transparency of the derived experiment. Additional features that are tested include usability, the ability to run on multiple platforms and on a remote cluster, and the inclusion and automatic update of results in the paper. For the planned mode, the benchmark includes binary and source code, and a textual description of the experiment. Users will utilize this information to simulate the creation of the experiment from scratch. The spontaneous benchmark provides binary code which runs on a particular linux environment. Tools will need to wrap these binaries after one execution of this binary code in the source environment. The wrapped package must then be able to run in a target environment and create an executable paper. Here, we focus on the spontaneous mode.

There will be two kinds of users who deal with this benchmark. *Authors* will create a reproducible experiment using tools, and *reviewers/judges* will unpack the experiment and look at results for different parameter combinations and input files.

1. Author role. The author will make the experiment reproducible by *wrapping* to run on various platforms, after running it on a source platform (or a virtual machine or set of virtual machines representing the source platform). Multiple variations of the experiment are provided. One of them requires execution on a cluster of computers. The experiment can also be tuned to generate different numbers of results, allowing the scalability of the systems to be assessed.

2. Reviewer role. Judges/reviewers will determine usability: the tool-produced package should be easy to unpack, run the experiments, and explore parameter variations. In addition, since reviewers may have to copy the experiments, it is important to consider (and measure) the size of the package. For example, while virtual

machines provide great portability, they can be very large – much larger than packages derived from tools that include only the required dependencies [4, 12]. More important perhaps, virtual machines might capture information the code authors do not wish to reveal.

4 The Spontaneous Reproducibility Challenge

For the spontaneous challenge, there will be a training and a real experiment (binaries and data). The training set will be used by tool developers while preparing their tools for the challenge. The real set will be sent well in advance but encrypted. The decryption key will be sent the day the competition begins. Also, for each challenge, there will be both a tool-building portion and a blinded judgement portion. To create reproducible documents, there will have to be some integration with typesetting packages such as LaTeX or text editors such as Microsoft Word. Since there is no general tool that supports all the reproducibility tasks, we envision that tool builders will form alliances in order to win the benchmark competition.

Besides the evaluation criteria outlined in Section 3, another criterion for the tool-builder part is a timing test: how long does it take for the team to wrap the code and data for each target system? Scoring works as follows assuming N participants: the fastest team for a given target system receives N points, the second fastest receives N-1, ... Any team that does not succeed in wrapping the code for a particular target system receives zero points for that target.

The wrapped code will be uploaded to a judge site where it will be blinded by the chair of the competition. Each wrapped code will then be judged by three judges on the following criteria:

1. Given a configuration file in text format (using the same format for training and test), run through every combination of parameters and calculate the mean and the variance of some parameter (e.g., critical temperature in the case of the Ising model). In the configuration file the possible values of each parameters will be provided. If this works, then the team receives N/2 points. Otherwise zero points.

2. The size of the package that runs on the target machine. The smallest package of those that run receives N points, the second smallest N-1, etc. If the software does not run, then the team receives zero points.

3. How long does it take the judges to run the tool in wall clock time with new parameter settings, generate all the graphs and numerical values, and embed these into a new version of the paper? The fastest team receives N points, the second fastest receives N-1, ... If it doesn't work, then 0 points.

4. Can the judges interact with the paper and change parameters and/or input data and have the figures update without further special action? Any team that can do this receives N points. Otherwise 0.

5. The judges subjective score on a scale of easy (3) to difficult (0). The three judges' scores are summed and then the team with the highest score receives N points, the second highest N-1, ...

The team with the highest score will receive a monetary prize. If more than one team ties for the most points, the prize will be divided equally. All participants will have the option to write an article about their effort in the journal Information Systems.

Acknowledgments. This work was partially supported by the Sloan Foundation, and the National Science Foundation under grants CNS-1229185, IIS-1139832, IIS-1142013 DBI-0445666, DBI-0421604, and N2010 IOB-0519985. We thank M. Dolfi, J. Gukelberger, A. Hehn, J. Imriska, K. Pakrouski, T. F. Rnnow, M. Troyer, and I. Zintchenko for designing and implementing the experiment used in our benchmark.

References

[1] Biostatistics Journal – Information for Authors, 2013. http://www.oxfordjournals.org/our_journals/biosts/for_authors/msprep_submission.html.

[2] P. Bonnet, S. Manegold, M. Bjørling, W. Cao, J. Gonzalez, J. Granados, N. Hall, S. Idreos, M. Ivanova, R. Johnson, D. Koop, T. Kraska, R. Müller, D. Olteanu, P. Papotti, C. Reilly, D. Tsirogiannis, C. Yu, J. Freire, and D. Shasha. Repeatability and Workability Evaluation of SIGMOD 2011. *SIGMOD Rec.*, 40(2):45–48, Sept. 2011.

[3] P. Bonnet, S. Manegold, M. Bjørling, W. Cao, J. Gonzalez, J. Granados, N. Hall, S. Idreos, M. Ivanova, R. Johnson, D. Koop, T. Kraska, R. Müller, D. Olteanu, P. Papotti, C. Reilly, D. Tsirogiannis, C. Yu, J. Freire, and D. Shasha. Repeatability and Workability Evaluation of SIGMOD 2011. *SIGMOD Rec.*, 40(2):45–48, Sept. 2011.

[4] F. S. Chirigati, D. Shasha, and J. Freire. Packing experiments for sharing and publication. In *SIGMOD Conference*, pages 977–980, 2013.

[5] D. Donoho, A. Maleki, I. Rahman, M. Shahram, and V. Stodden. Reproducible Research in Computational Harmonic Analysis. *Computing in Science & Engineering*, 11(1):8–18, Jan.-Feb. 2009.

[6] Elsevier's Executable Paper Grand Challenge – 2011. http://www.executablepapers.com/.

[7] S. Fomel and J. F. Claerbout. Guest Editors' Introduction: Reproducible Research. *Computing in Science and Engineering*, 11(1):5–7, 2009.

[8] M. H. Freedman, J. Gukelberger, M. B. Hastings, S. Trebst, M. Troyer, and Z. Wang. Galois Conjugates of Topological Phases. *Phys. Rev. B*, 85:045414, Jan 2012.

[9] J. Freire, P. Bonnet, and D. Shasha. Computational Reproducibility: State-of-the-Art, Challenges, and Database Research Opportunities. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 593–596, New York, NY, USA, 2012. ACM.

[10] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[11] J. Freire and C. Silva. Towards Enabling Social Analysis of Scientific Data. In *ACM CHI Social Data Analysis Workshop*, 2008.

[12] P. Guo. CDE: A Tool for Creating Portable Experimental Software Packages. *Computing in Science and Engineering*, 14(4):32–35, 2012.

[13] IEEE Transactions on Signal Processing – Reproducible Research, 2013. http://www.signalprocessingsociety.org/publications/periodicals/tsp/.

[14] S. Manegold, I. Manolescu, L. Afanasiev, J. Feng, G. Gou, M. Hadjieleftheriou, S. Harizopoulos, P. Kalnis, K. Karanasos, D. Laurent, M. Lupu, N. Onose, C. Ré, V. Sans, P. Senellart, T. Wu, and D. Shasha. Repeatability & Workability Evaluation of SIGMOD 2009. *SIGMOD Rec.*, 38(3):40–43, Dec. 2010.

[15] I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, S. Zoupanos, and D. Shasha. The Repeatability Experiment of SIGMOD 2008. *SIGMOD Rec.*, 37(1):39–45, Mar. 2008.

[16] J. P. Mesirov. Accessible Reproducible Research. *Science*, 327(5964):415–416, 2010.

[17] Nature Magazine Policy – Reporting Checklist for Life Sciences Articles, 2013. http://www.nature.com/authors/policies/checklist.pdf.

[18] The Reproducibility Challenge. http://www.reproduciblescience.org/index.php/Reproducibility_Challenge.

[19] E. Santos, J. Freire, and C. Silva. Information Sharing in Science 2.0: Challenges and Opportunities. In *CHI Workshop on The Changing Face of Digital Science: New Practices in Scientific Collaborations*, 2009.

[20] Science Magazine – Submission of Supplementary Materials, 2013. http://www.sciencemag.org/site/feature/contribinfo/prep/prep_online.xhtml.

[21] SIGMOD Experimental Repeatability, 2012. http://www.sigmod.org/2012/reproducibility.shtml.

[22] C. Tenopir, S. Allard, K. Douglass, A. U. Aydinoglu, L. Wu, E. Read, M. Manoff, and M. Frame. Data Sharing by Scientists: Practices and Perceptions. *PLoS ONE*, 6, 06/2011 2011.

[23] J. Tohline and E. Santos. Visualizing a Journal that Serves the Computational Sciences Community. *Computing in Science Engineering*, 12(3):78–81, 2010.

[24] VLDB Experimental Reproducibility, 2013. http://www.vldb.org/2013/experimental_reproducibility.html.