

Reproducible Research in Computational Harmonic Analysis

Scientific computation is emerging as absolutely central to the scientific method, but the prevalence of very relaxed practices is leading to a credibility crisis. Reproducible computational research, in which all details of computations—code and data—are made conveniently available to others, is a necessary response to this crisis. The authors review their approach to reproducible research and describe how it has evolved over time, discussing the arguments for and against working reproducibly.

Massive computation is transforming science, as researchers from numerous fields launch ambitious projects involving large-scale computations. Emblems of our age include

- data mining for subtle patterns in vast databases; and
- massive simulations of a physical system's complete evolution repeated numerous times, as simulation parameters vary systematically.

The traditional image of the scientist as a solitary person working in a laboratory with beakers and test tubes is long obsolete. The more accurate image—not yet well recognized—depicts a computer jockey working at all hours to launch experiments on computer servers. In fact, today's

academic scientist likely has more in common with a large corporation's information technology manager than with a philosophy or English professor at the same university.

A rapid transition is now under way—visible particularly over the past two decades—that will finish with computation as absolutely central to scientific enterprise. However, the transition is very far from completion. In fact, we believe that the dominant mode of scientific computing has already brought us to a state of crisis. The prevalence of very relaxed attitudes about communicating experimental details and validating results is causing a large and growing credibility gap. It's impossible to verify most of the results that computational scientists present at conferences and in papers.

The Crisis

To understand our claim, and the necessary response, we must look at the scientific process more broadly. Originally, there were two scientific methodological branches—*deductive* (for example, mathematics) and *empirical* (for example, statistical data analysis of controlled experiments). Many scientists accept *computation* (for example, large-scale simulation) as the third branch—some believe this shift has already occurred, as one can see in grant proposals, keynote speeches, and newsletter editorials. However, while computation is already

1521-9615/09/\$25.00 © 2009 IEEE
COPUBLISHED BY THE IEEE CS AND THE AIP

DAVID L. DONOHO, ARIAN MALEKI, AND MORTEZA SHAHRAM
Stanford University
INAM UR RAHMAN
Apple Computer
VICTORIA STODDEN
Harvard Law School

indispensable, it does not yet deserve elevation to third-branch status because current computational science practice doesn't generate routinely verifiable knowledge. The scientific method's central motivation is the *ubiquity of error*—the awareness that mistakes and self-delusion can creep in absolutely anywhere and that the scientist's effort is primarily expended in recognizing and rooting out error. Before scientific computation can be accorded the status it aspires to, it must be practiced in a way that accepts the ubiquity of error, and work then to identify and root out error. Deduction and empiricism are powerful but error-prone. Even very disciplined and cultivated branches of science suffer notably from the problem of errors in final, published conclusions.¹ Vagueness, wandering attention, forgetfulness, and confusion can plague human reasoning. Data tend to be noisy, random samples contain misleading apparent patterns, and it's easy to mislabel data or misinterpret calculations. Beginning researchers in science face a lengthy struggle to discipline their thoughts and expressions and to identify and root out mistakes. They must also adopt disciplined habits of collection, treatment, and data processing and develop a profound understanding that measurement error causes mistakes in a substantial fraction of conclusions. Even established empirical scientists struggle when interpreting or processing a dataset; obscure misunderstandings could invalidate the analysis, or random noise could be confused with meaningful signal. Mature responses to the ubiquity of error have evolved: for deduction, formal logic, and mathematical proof; for empiricism, statistical hypothesis testing, and standardized reproducibility information (data, materials, and methods).

Like deduction and empiricism, computation is also highly error-prone. From the newcomer's struggle to make even the simplest computer program run to the seasoned professional's frustration when a server crashes in the middle of a large job, all is struggle against error. In the world of computing in general, not just scientific computing, the ubiquity of error has led to many responses: special programming languages, error-tracking systems, disciplined programming efforts, organized program testing schemes, and so on. The point we make is that the tendency to error is central to every application of computing.

In stark contrast to the sciences relying on deduction or empiricism, computational science is far less visibly concerned with the ubiquity of error. At conferences and in publications, it's now completely acceptable for a researcher to simply

say, "here is what I did, and here are my results." Presenters devote almost no time to explaining why the audience should believe that they found and corrected errors in their computations. The presentation's core isn't about the struggle to root out error—as it would be in mature fields—but is instead a sales pitch: an enthusiastic presentation of ideas and a breezy demo of an implementation. Computational science has nothing like the elaborate mechanisms of formal proof in mathematics or meta-analysis in empirical science. Many users of scientific computing aren't even trying to follow a systematic, rigorous discipline that would in principle allow others to verify the claims they make. How dare we imagine that computational science, as routinely practiced, is reliable!

A Practical Response

Jon Claerbout saw this crisis coming more than 20 years ago. Working in exploration geophysics, which requires data analysis and computational algorithm development, he saw that the informa-

***We publish not just computational results
but also the complete software environment
and data that generated those results.***

tion computational scientists conveyed in their publications was seriously incomplete. Paraphrasing Claerbout you might say that "an article about computational science in a scientific publication is not the scholarship itself, it's merely scholarship advertisement. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."²

Claerbout, together with his Stanford Exploration Project team, developed a system for document preparation that attempted to encapsulate a body of work's full scholarly content. When reading an article using the special document viewer his team created, you find yourself inside the complete computational environment (code and data) that generated the results that the article presents. His group hyperlinked the code underlying the figures in the papers to the document so that readers could study and even rerun the code to reproduce the results from scratch. Readers could even play "what if" games, modifying the original version's parameters, to see how the author's results would change.

Although most scientists today can more easily

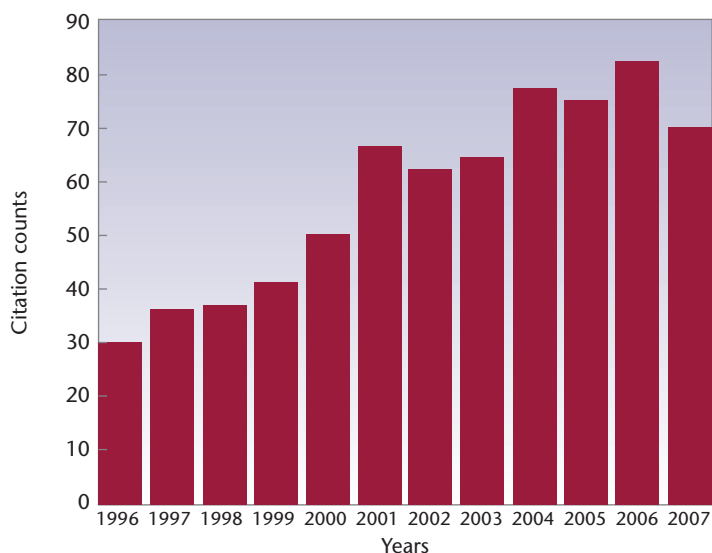


Figure 1. Citation counts for Wavelab by year. We found 690 citations in Google Scholar, with citations continuing from the mid 1990s until today.

understand these concepts than 20 years ago, and the technology is much easier to use today than it was then, scientists still don't widely practice or routinely demand this approach.

Our Own Efforts

In the early 1990s, David Donoho learned of Claerbout's ideas and began to practice them in his own research. Roughly speaking, Donoho didn't trust himself to turn out good work unless his results were subject to the open criticism of other researchers. He also learned from experience not to believe that graduate students were doing the work they believed they were doing unless he studied and ran the code himself. Cross-platform, high-level quantitative programming environments, such as Matlab, were becoming standard tools.

Donoho's approach was to make sure that the details underlying the datasets, simulations, figures, and tables were all expressed uniformly in Matlab's standard language and computing environment and made available on the Internet—which was, at that time, becoming a standard tool—so that interested parties could reproduce the calculations underlying a particular paper. He also decided to impose that discipline on students. At the time, he was working in computational harmonic analysis (wavelets, wavelet packets, and time-frequency methods), and few computational tools were available. After he and

his collaborators had written several papers on wavelets, Donoho was able to combine all the tools into a single package, Wavelab, which contained a unified set of wavelet and time-frequency tools, and reproduced all the calculations in those papers.

Wavelab is based on the Matlab quantitative computing environment, which has a programming language for manipulating matrices and vectors as well as useful plotting tools. Matlab runs on numerous operating systems, so it offers a programming language that's free of machine dependencies. Wavelab is installed as a Matlab toolbox and has been available online in some version since 1993. The original version had more than 700 files, including M-files, C-code, data, and documentation—it has since grown. For more information on Wavelab, see Jonathan Buckheit and Donoho's paper, "Wavelab and Reproducible Research."²

Although many Matlab toolboxes were available before Wavelab's release, it seems that Wavelab was the first effort to provide a toolbox specifically designed to reproduce results in a series of computational science papers. Research ethos at the time (and mostly since) wasn't to publish a complete reproducible package. Instead, researchers either did nothing to release algorithms and data to the outside world, or they published fragments of their environment, ignoring the notion of true reproducibility. The purpose of Wavelab was never to provide a complete "wavelet toolbox" for general-purpose use. However, because of the broad range of papers that were reproduced in the original Wavelab release, the package itself contained a fairly complete offering of wavelet transforms and time-frequency-analysis tools. Teachers had also been using it and had developed numerous pedagogical examples. As a result, many users viewed it as free software that competed with commercial packages. In fact, the central motivation in the authors' minds was reproducibility; the papers in Wavelab became a kind of benchmark because the papers fully documented signal-processing techniques' performances in certain datasets. When developing other new techniques, researchers often used the Wavelab datasets and performance measures as a way to compare methods. Partly as a result, Wavelab became highly cited. We found that from 1996 to 2007, Wavelab received 690 citations from Google Scholar, and we studied 100 randomly selected citations in preparing this article (Figure 1 shows the citation count by year, and Table 1 categorizes usage patterns).

Why Do This?

Why did we feel it was important to proceed in this way? Several reasons are mentioned in “Wavelab and Reproducible Research,”² the thrust being that if everyone on a research team knows that everything they do is going to someday be published for reproducibility, they’ll behave differently from day one. Striving for reproducibility imposes a discipline that leads to better work.

In pursuing full reproducibility, we fundamentally produce code for strangers to use. Although it might seem unnatural to help strangers—and reluctance to do so certainly impedes the spread of reproducible research—our term *stranger* really means anyone who doesn’t possess our current short-term memory and experiences. In the heat of a computational project, we store many things in short-term memory that we need at that moment to use the code productively. Developing for a stranger means avoiding reliance on this soft, transient knowledge and, more specifically, codifying that knowledge objectively and reproducibly. Is it then really necessary to help strangers? We think that it is. Only if we work as if a stranger must use the code without complaint will we produce code that’s truly reproducible.

In fact, we needn’t think that we’re doing something unnecessarily altruistic by developing code for mere strangers. There are some very important strangers in our lives:

- coauthors,
- current and future graduate students,
- new postdoctoral fellows,
- anonymous referees of our papers and grant proposals, and
- future employers.

We call them strangers because they wouldn’t share all the short-term information we’ve recently accumulated in developing our current computational results. In fact, years from now, we ourselves will be such strangers, not remembering the myriad small details that accumulated in our minds during this project. It’s not uncommon for a researcher who doesn’t follow reproducibility as a personal discipline to forget how the software in some long-ago project was used, what its limitations were, or even how to generate an example of its application. In contrast, a researcher who has worked reproducibly can actually go to the Internet, find his own long-ago work there, download it, and use it in the same way that anyone else could. Thus, working reproducibly helps everyone, including our

Table 1. Usage pattern analysis of Wavelab citations.*

Use pattern	%
Used algorithms in Wavelab to process data	74
Used both algorithms and datasets in Wavelab	11
Used only datasets from Wavelab	3
Compared performance of algorithms with algorithms in Wavelab	12

*Selected citations in Google Scholar classified by the way the citing article used Wavelab.

future selves. It seems that striving for truly reliable computational results is inseparable from dedicating our efforts to producing a package that anyone can run from scratch to reproduce our results.

Our Framework Broadens

We’ve now worked within the reproducibility paradigm for 15 years, during which time we’ve tried to get our collaborators, students, and postdocs to participate. To help make this happen, we’ve created a family of toolboxes following the Wavelab framework:

- *Atomizer*, for sparse representation of signals by ℓ_1 minimization;
- *Beamlab*, for multiscale geometric analysis;
- *Sparselab*, for sparsity-seeking decomposition and reconstruction;
- *Symmmlab*, for multiscale analysis of manifold-valued data; and
- *Spherelab*, for multiscale decomposition of data on the sphere.

All these toolboxes share a common arrangement of files and a common scheme for documenting and organizing the information. For more information on our research and tools, see the extended version of this article at www-stat.stanford.edu/~wavelab/Reproducibility.htm. An author working on a novel project can easily adopt the Wavelab framework: simply download one of these packages, study its file structure, and treat that structure as a model. Then, clone the general-purpose files in the package, rename them, and edit them to create an entirely new package in a different topic area.

Over time, the packages in question—particularly Wavelab—have grown through additions. Although many of these additions have come from doctoral students or postdocs in our group, we’ve also had contributions from *outsiders*; see the Web address in the preceding paragraph for URL links to several such examples, including those that fol-

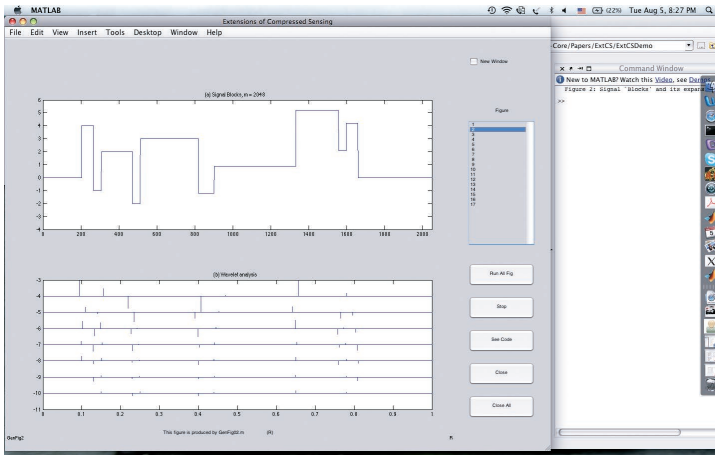


Figure 2. A recomputed and reproduced figure from the paper “Extensions of Compressed Sensing.”⁹ The box in the middle lists all the figure numbers in the paper; we reproduce here Figure 2 from that paper, but we could reproduce any other figure by clicking elsewhere in the list. Note the button in the lower right labeled *See Code*; clicking here reveals the underlying source code that generates this figure.

low. The framework’s modular design is such that it’s fairly easy to simply drop a new folder into a toolbox and create a new distributable archive. We’ve apparently inspired former students to develop their own frameworks.

- Emmanuel Candès at Cal Tech has worked with doctoral students Laurent Demanet, Lixing Ying, and Justin Romberg to produce two reproducibility-inspired packages—Curvelab for curvelet analysis and ℓ_1 Magic for ℓ_1 minimization,
- Xiaoming Huo in the School of Industrial Systems and Engineering at Georgia Tech has released the package CTDLab, which solves “connect the dots” problems.³

Researchers elsewhere have built packages that build on our framework (for Web addresses, see this article’s extended version, cited earlier):

- Jalal Fadili, in the Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen, has developed an open software package called MCALab.
- Stephane Mallat of Ecole Polytechnique has written a major book using our approach,⁴ in which he uses Wavelab to generate all the figures and examples—this code is included in recent releases of Wavelab.

Books reach a major milestone when published in

a foreign language. The analogy for a software package: translation to another computer language. Astronomer Amara Graps developed the “IDL Wavelet Workbench,”⁵ based partially on the Wavelab package. IDL is a popular computing environment for astronomical data analysis. We next briefly describe three recent toolboxes developed in our framework.

Sparselab

Modern research in computational harmonic analysis has shown that many kinds of signal content are highly compressible in an appropriate transform domain. Thus, images are compressible in the wavelets domain, acoustic signals can be compressible in the local cosine domain, and so on. This is the modern information era’s central phenomenon—we exploit it every time we download movies and music from the Web or use a cell phone to view multimedia content. Compressibility can be explained simply. The transformed signal is *sparse*: many values are essentially zero and can be ignored.

Sparsity has fundamental advantages beyond mere data compression. Amazingly, an underdetermined system of equations might well have many solutions but only one highly sparse solution.^{6,7} This is the foundation of *compressed sensing*, a rapidly developing field driven by the discovery that when a signal has a highly sparse transform, the required number of measurements is proportional not to the size of the uncompressed signal but instead to the size of the compressed signal.⁸

Sparse representation is a rapidly developing field. The ISI Thompson index classified the topic of “sparse solution of underdetermined systems of linear equations” as a new research front in May 2005. The journal *IEEE Signal Processing* named a core paper reproduced by Sparselab, “Extensions of Compressed Sensing,” as its most-cited paper in 2006.⁹ By our count, in 2008, roughly 50 papers with some variant of the words *compressed sensing* appeared in *Magnetic Resonance in Medicine*, and the March 2008 issue of *IEEE Signal Processing* magazine (circulation 20,000) was devoted to this topic. Besides compressed sensing, Sparselab covers subtopics such as statistical model selection. Ten researchers have contributed software, and as many as three people contributed their time to support and maintain the package. One of this article’s coauthors, Victoria Stodden, managed the creation of this reproducible research software package as a component of her doctoral thesis.¹⁰

Sparselab is both a vehicle for authors in the field to make their papers fully reproducible and a source of established problems and algorithms for

sparse representation. We released Sparselab in 2006 at <http://sparselab.stanford.edu>, and the latest version, Sparselab 2.1, has been downloaded more than 7,000 times in 2008 alone. A casual search turned up more than a dozen papers that have used Sparselab to generate their results, and the tool is still evolving through volunteer contributions.

The package has a user interface that goes beyond merely reproducing published results. It has a convenient graphical interface that lets users easily make variations on experiments in the published papers; see Figure 2 for an example of this user interface.

The core package has 663 files and folders, mostly associated with articles. Sparselab also includes some code for general algorithms, such as Michael Saunders' primal-dual method for convex optimization.

Reproducibility can accelerate progress in a hot field and implicitly creates a source of challenging problems and datasets. Many researchers are interested in comparing their own novel algorithms with the algorithms first published in the Sparselab package. This, in turn, leads those researchers to cite reproducible papers in Sparselab. Some of those early researchers later contributed their own algorithms to the package, thus making it even more authoritative as a source for compressed sensing and related applications of sparsity-seeking methods.

Symmlab

Traditionally, computational harmonic analysis works with arrays of numbers. Many new technological and scientific datasets don't contain traditional numbers; instead, measurements pinpoint locations on curved surfaces and manifolds. Human motion data provides a simple example. The measurements quantify a configuration of the principal joint angles in the human body (elbows, wrists, and so on). Such data are not an arbitrary collection of numbers, because the joint angles must be mutually consistent. If we view them as specifying points in a manifold, all the consistency conditions are naturally in force. The configuration manifold in question is a product of numerous copies of the Lie groups $SO(2)$ and $SO(3)$, one copy for each joint. Here, $SO(k)$ refers to the collection of orientations of a k -dimensional orthogonal coordinate set in R^k . The choice of $SO(2)$ versus $SO(3)$ depends on the type of flexibility available in a specific joint. Aircraft orientation data provide another example, in which the data express the aircraft's orientation as a function of time. You can convert traditional aircraft orien-

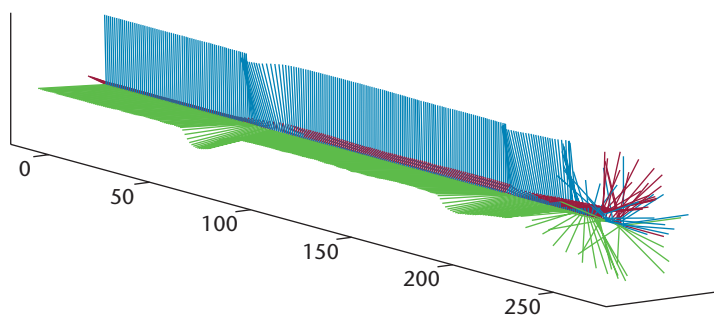


Figure 3. $SO(3)$ -valued data: aircraft orientation as function of time. A tripod with blue, red, and green vectors indicates each instant's orientation. The wild variation in orientation near the series' end indicates that the airplane lost control and crashed.

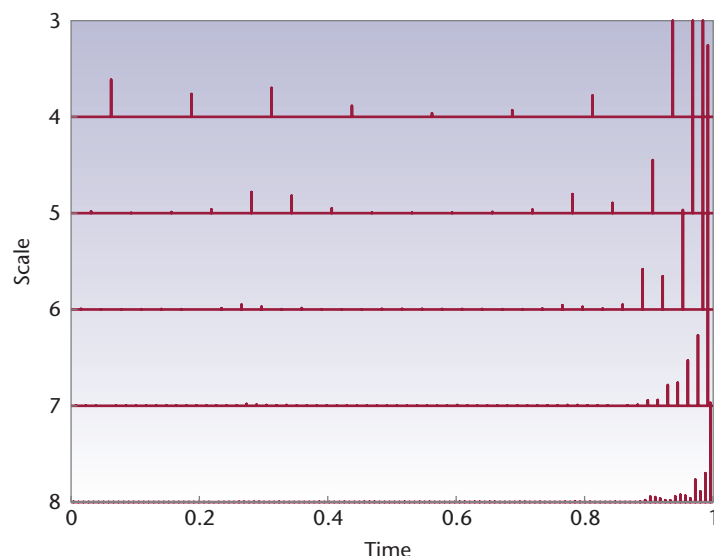


Figure 4. Aircraft orientation's wavelet transform, organized by scale (vertical axis) and time (horizontal axis). Each wavelet coefficient (not presented here) is a Lie algebra $SO(2)$ element; the display presents each coefficient's norm. The large coefficients at coarse scales and at the series' end at fine scales demonstrate that the aircraft's motion was smooth until control was lost, at which time motion became energetic at all scales.

tation coordinates—pitch, roll, and yaw—into an $SO(3)$ -valued representation (see Figure 3).

It turns out that you can model a very broad collection of manifold-valued data using an elegant class of manifolds called Riemannian symmetric spaces. Such manifolds, in turn, permit an elegant systematic notion of wavelet transform.¹¹ Figure 4 presents aircraft orientation data's $SO(3)$ -valued wavelet coefficients. Sym-

mlab, a Matlab toolbox, plays the same role for symmetric-space-valued data that Wavelab plays for more traditional data. It can reproduce “Multiscale Representations for Manifold-Valued Data,” a paper by Inam Ur Rahman and his colleagues¹¹ and Rahman’s thesis,¹² and it might be the first available toolbox for manifold-valued data’s multiscale analysis.

Symmmlab follows the Wavelab framework, with valuable improvements related to developments in later packages such as Sparselab. Compared to its sister packages, there is a great deal of emphasis on providing new and interesting datasets. Symmmlab raises awareness of manifold-valued data and makes available tools to analyze such data. You might even view the paper’s main contribution¹¹ as the formalization of this need. Here, reproducibility is important more for the diffusion of the datasets rather than the results.

The datasets available in the download come from diverse sources:

- weak-gravitational-lensing data from astronomy,
- motion-capture data in animation,
- aircraft-motion data from aviation,
- time-varying-deformation-tensor data from geophysics,
- diffusion-tensor data from medical imaging,
- subspace-tracking data from array signal processing,
- wind-direction and speed data from climatology, and
- synthetic-aperture-radar- (SAR-) interferometric data from remote sensing.

The Symmmlab package is available at www-stat.stanford.edu/~symmmlab. It contains more than 100 M-files, including core algorithms, utilities to generate synthetic data, and visualization tools. The Web site contains instructions and documentation for registering, downloading, installing, and using the package. Rahman and his colleagues’ paper describes technical details about the algorithms and can also be downloaded from this Web site.¹²

This toolbox demonstrates an important advantage of reproducible research: reproducibility is a way to rapidly disseminate new problems and illustrate new data types. In this case, the person downloading Symmmlab might be uninterested in wavelets per se but very interested in symmetric-space-valued data and other fascinating application areas. The interested reader can download, install, verify, see the raw data, and learn about the problems a reproducible paper addresses, rather

than the mathematical solutions it proposed. Exact reproducibility has side effects from publishing the underlying data and the algorithm, not just the intended result.

Spherelab

Traditional wavelet analysis concerns data that take real values and are organized by a Cartesian parameter such as time (for example, an equispaced sampling in time) or two-dimensional space (for example, a regular Cartesian grid in the plane). However, in the Earth sciences, aviation, atmospheric sciences, and astronomy, many important datasets measure properties of various locations on the sphere. Such settings require wavelet transforms adapted to spherically organized data.

There are many systems of coordinates on the sphere. The Healpix system is the basis for much recent work in observational cosmology (<http://Healpix.jpl.nasa.gov>). The major astronomical organizations—NASA and the European Southern Observatory—adopted the Healpix system to organize and process their data. They sponsored an open source software package performing Healpix manipulations; it includes Fortran 90, IDL, and C++ sources. Although relatively new and little known to outsiders, scientists and engineers in many other fields could profitably use Healpix.

The article “Multiscale Representation for Data on the Sphere and Applications to Geopotential Data”¹³ develops wavelet representations for Healpix-organized data. Spherelab is a Matlab toolbox that lets readers reproduce calculations from that paper. Morteza Shahram developed Spherelab as part of his postdoctoral fellowship at Stanford; his sponsor wanted him to develop a new kind of wavelet for spherical data that would cooperate with traditional spherical harmonics expansions. Astronomers developed the Healpix grid as a pointset on the sphere so that researchers could perform fast spherical harmonics expansion evaluation; this makes the Healpix grid particularly natural for wavelet development that cooperates with spherical harmonics.

Because of sponsor interest, Shahram specifically focused Spherelab on geopotential applications, where researchers must evaluate different geo-related quantities and improve current geopotential models as they collect new measurements. The main deliverable was a system for compressing and decompressing a geopotential field on the sphere much more rapidly than possible by traditional spherical harmonics expansions.

Shahram's article¹³ demonstrates that Spherelab achieved this milestone, and reproducing those results verifies this.

Spherelab should be an easily accessible and useful tool for non-geopotential applications, such as 3D object rendering in computer graphics. Spherelab includes tools for conversions between Healpix and Healpix wavelets, refinement schemes on the sphere, statistical analysis, demos, test files, and so on. Overall, this package has roughly 150 main files and about the same number of test files. Figure 5 shows a multiscale decomposition of the global geopotential.

Spherelab demonstrates that reproducibility is a way for research sponsors to ensure that funded research will assume tangible forms. In this case, the sponsor wasn't at all interested in the academic paper but rather in the software producing results in that paper. The sponsor could download, install, verify proper code function, and see that his or her money was well spent.

Reproducibility as Software Publishing

Full-blown reproducibility requires publishing code to regenerate computational results. Consequently, an adherent reproducibility becomes a software publisher. This could be a new experience for academics, and it surely adds a new set of issues to deal with—here are a few.

Tech Support

We hear many positive comments from our software users, but several times a year, we hear of problems in one of our packages. Typically, these are mundane issues, most frequently related to installation. Occasionally, more serious issues occur, often to do with the release of a new version of Matlab.

Tech support is both a burden and an opportunity. We often assign tech support tasks to graduate students who are just starting their doctoral research. This gives the student a chance to realize early on that strangers really will download and try to use published code and that mistakes in such code will lead to problem reports. This experience, we believe, makes students aware that it's important to do high-quality and lasting work in their software development efforts. They quickly realize that we really do mean it when we say all their computational work might someday be subject to public scrutiny.

Code Maintenance

Much of the original Wavelab package's Matlab code was written 15 years ago, but still runs cor-

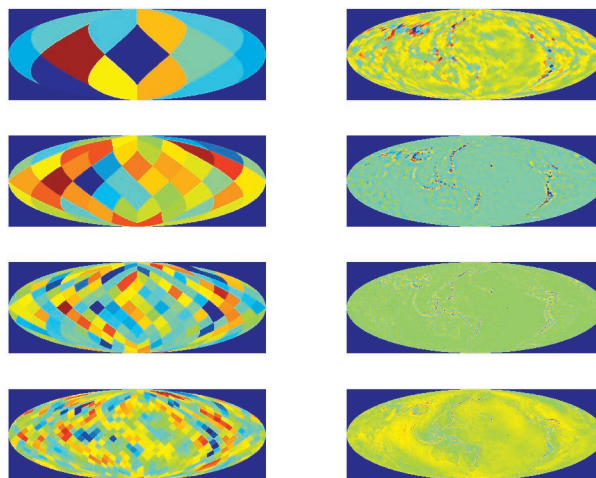


Figure 5. Multiscale decomposition of global geopotential using Healpix wavelets. The new high-resolution EGM-06 dataset became available in 2006; it's used in aircraft navigation and contains unprecedented data density. Different panels represent different scales of the Healpix wavelets decomposition; fine-scale features near mountain ranges and continental boundaries are evident.

rectly. We needed to change a few basic utility functions over the years, but otherwise, much of the code still runs without problems. This validates Matlab's role as a uniform computational environment for delivering reproducibility across hardware and operating system variations.

The bulk of the work involved in maintaining the code concerns problems that have arisen with major changes in Matlab or in the underlying operating system. This, again, isn't all bad: graduate students in the early stages of their doctoral careers assigned to maintenance tasks will learn that we intend to keep the code working for many years to come and will learn about potential problems involved in doing so. This awareness can prepare students to do a better job in their own future projects.

Maintaining a Web Presence

Of course, reproducibility means publication over the Internet. This requires setting up and maintaining a Web site, which again can be informative and educational but occasionally time-consuming. However, it seems to be getting easier all the time.

Reflections

Fifteen years of experience with reproducibility provoked numerous discussions with other researchers about the need for and effectiveness of this initiative. We offer a few take-away points.

Has It Worked?

Reproducible research has impact. Google Scholar displays thousands of citations of the core papers reproduced by Wavelab and Atomizer; there are already hundreds of citations of Sparselab's still very recent core papers. Many people have now used the software in these packages, which appears to perform as originally hoped. This is what reproducibility really delivers.

When Did It not Work?

Numerous examples of reproducible research emerged from our efforts over the past 15 years, but we can cite examples of reproducibility. Exceptions fall in three categories:

- *Postdocs*. By and large, postdocs are in a hurry to publish and would prefer to work the way in which they're accustomed rather than absorb some new way of working—and who can blame them?
- *Theorists*. For the most part, a theory paper might have a few diagrams in it and perhaps a numerical calculation, but the theory student probably hates computing anyway and sees no need to develop more disciplined ways of doing computational research.
- *One-off projects*. We did a few computational projects that might have benefited from reproducibility, but were too small-scale to justify building an entire toolbox that must then be maintained. Also, there was no prospect of any of the authors continuing their research in the given direction, so motivation for the required investment was very weak.

An advisor can't force reproducibility on collaborators. Researchers must believe that reproducibility is in their best interest. Postdocs, in particular, are generally panicked about their future prospects and tend to be short-sighted—although not always: Thomas Yu, Georgina Flesia, and Morteza Shahram prove that some postdoctoral scholars are willing to work reproducibly.

Quick Answers to Knee-Jerk Objections

Many researchers exhibit an instinctive aversion to working reproducibly; when they vocalize their instincts, we give them the following winning responses.

Reproducibility takes time and effort. Response: Undeniably true. If our only goal were getting papers published, we would get them published sooner if we didn't have to also worry about pub-

lishing correct and polished code. Actually, our goal is to educate future researchers; this takes time and requires effort. Moreover, if we view our job in graduate education as actually reviewing and correcting students' work, this is dramatically easier to do if they work reproducibly—so, in many cases, there's actually less total time and effort than otherwise.

No one else does it, so I won't get any credit for it.

Response: Partly true. Few people today work reproducibly. However, for exactly this reason, if your code is available, your work will get noticed and actually used, and because it'll get used, it'll eventually become a reliable tool. Would you prefer that other researchers not trust your work or ignore it or that someone else's implementation of your idea become better known simply because they make it available?

Strangers will use your code to compete with you.

Response: True. But competition means that strangers will read your papers, try to learn from them, cite them, and try to do even better. If you prefer obscurity, why are you publishing?

My computing environment is too complicated to publish like this.

Response: This seems a winning argument if you must work on a 10,000-node cluster using idiosyncratic hardware. But if so, should anyone believe that your obscure computing environment is doing what you say it does? Also, how can you believe it? Shouldn't you test a subset of your computations in a standard environment, instead of in an idiosyncratic one, and then verify that you obtained identical results? Isn't reproducibility more important in this case, not less?

Sober Answers to Thoughtful Objections

Some researchers have been able to vocalize thoughtful objections to reproducibility; here are a few such objections, together with what we view as winning arguments.

Reproducibility undermines the creation of intellectual capital.

Instead of building up a comprehensive set of tools in a lab over years, you give the tools away for free, before they ripen. This prevents a researcher from developing a toolkit in the course of a career.

Response: A postdoc could argue this with some justification, but not a senior researcher. For a senior researcher running a lab in which doctoral students come for a time and then leave, the only way to actually ensure the creation of valuable

capital is to require students to work reproducibly, so that something of value can be created under the scrutiny and emendation of others—including the senior researcher. For postdocs specifically: if you intend to someday be a senior researcher, shouldn't you learn today how to best organize substantial computational projects in a way that you and your colleagues can benefit from them in the future? Won't your skills impress potential employers more if you can show them that your work is reproducible and transparent to them? Won't potential senior collaborators view you as a more valuable partner if they believe they can work transparently with code that you develop? Isn't getting noticed a problem for postdocs? Wouldn't other people using your code be the same as getting noticed?

Reproducibility destroys time-honored motivations for collaboration. Traditionally, science is developed through collaboration in which scientists visit each others' laboratories and share tools and ideas in quid pro quo arrangements. This promotes the spread of ideas through close human contact, which is both more humane and more likely to lead to educational and intellectual progress. If people don't need to collaborate because they can just download tools, science's social fabric is weakened. As science is primarily a social entity, this is quite serious.

Response: This is the intellectual capital argument in disguise. Reproducibility is important within one laboratory or a team of investigators or even for future use of the same code by the same investigator. If you were working in a secret establishment, air-gapped to the outside world, working reproducibly would still make sense. The issue is thus not reproducibility, but widespread publication. You always have the option to work reproducibly but then modify or skip the last step—namely, giving away your code. You can even provide remote, partial, controlled access to the computing environment underlying a research project. The traditional approach is to publish only the binary code and not your calculation's source code. A more modern approach is to set up a Web server that lets outsiders access proprietary algorithms in client-server fashion. Outsiders would not see source code of key algorithms but would instead upload their own datasets to the server that those algorithms would process. Upon completion, outsiders would then download results from the server. This server architecture also solves the problem of keeping data private while letting others reproduce

calculations. The server could apply algorithms to preselected private datasets that reside on the server only. The server architecture lets others check an implementation's accuracy and timing results while not giving away proprietary algorithms and data.

Reproducibility floods journals with marginal science. People will just download reproducible papers, make one tweak, and then modify the original paper in a few lines and resubmit a new paper.

Response: This phenomenon is probably real (our conclusion from anecdotal evidence), but trivial mechanisms exist for this problem, such as routine rejection of papers embodying tiny variations.


True reproducibility means reproducibility from first principles. It proves nothing if I point and click and see the numbers I expect to see. It only proves something if I start from scratch and build my version of your system and get your results with my implementation.

Response: If you exactly reproduce my results from scratch, that's quite an achievement! But it proves nothing if your implementation fails to give my results because we won't know why it fails. The only way we'd ever get to the bottom of such a discrepancy is if we both worked reproducibly and studied detailed differences between code and data.

Claerbout's original idea envisioned people working from a hyperdocument interface where, as they read an article, they could click on a figure, see the underlying code, and study or even rerun it. Our approach falls far short of this Utopian vision. We give the sophisticated Matlab user the ability to repeat our computations, access our M-files, and access our datasets.

Many people who find reproducibility interesting do so for reasons we wouldn't have predicted—that is, they want to understand one small detail or look at one aspect of a dataset. A fancier user interface might make the package less useful to the people really interested in reproducing computational science and really capable of understanding what's going on.

Nevertheless, we can see many ways to modernize our framework. A principal development of the past five years is the spread of social networking, Wikipedia, SourceForge, and related phenomena. Deploying such ideas in connection with reproducible research might let anonymous users post

improvements to packages on their own, without real involvement from our own group. Instead, we now require that users who want to add an improvement to one of our packages contact us and work with us. At the moment, we don't see any reason to abandon our old-fashioned approach—but we could be admittedly behind the times. 

Acknowledgments

The US National Science Foundation partially supported our work through its statistics and probability program (David Donoho and Iain Johnstone, co-PIs; DMS 92-09130, DMS 95-05150, DMS 00-772661, and DMS 05-05303), its optimization program (Stephen Boyd, PI), its focused research group (Multiscale Geometric Analysis, FRG DMS-0140698, David Donoho, PI), an information technology research project, a knowledge diffusion initiative project (Amos Ron, PI; KDI Towards Ideal Data Representation; and ITR Multiresolution Analysis of The Global Internet), and its signal-processing program. In addition, the US Air Force Office of Scientific Research partially supported our work through a multi-university research initiative project (David Castanon, PI); as did the Office of Naval Research (Comotion: Claire Tomlin, PI). Finally, DARPA partially supported our work through three projects: Efficient Mathematical Algorithms for Signal Processing, 1998–2000; Efficient Multiscale Methods for Automatic Target Recognition, 2001–2002; and GeoStar: Efficient Multiscale Representation of Geopotential Data, 2005–2007. Thanks to the program officers at these agencies for guidance and interest; we only mention a few: Mary Ellen Bock, Doug Cochran, John Cozzens, Dennis Healy, Reza Malek-Madani, Wen Masters, Jong-Shi Pang, Carey Schwartz, Jon Sjogren, Gabor Szekely, Grace Yang, and Yazhen Wang.

References

1. J.P.A. Ioannidis, "Why Most Published Research Findings are False," *PLoS Medicine*, vol. 2, no. 8, 2005, pp. 696–701.
2. J. Buckheit and D.L. Donoho, "Wavelab and Reproducible Research," *Wavelets and Statistics*, A. Antoniadis, ed., Springer-Verlag, 1995, pp. 55–81.
3. E. Arias-Castro et al., "Connect the Dots: How Many Random Points Can a Regular Curve Pass Through?," *Advances in Applied Probability*, vol. 37, no. 3, 2005, pp. 571–603.
4. S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed., Academic Press, 1999.
5. A. Graps, "An Introduction to Wavelets," *IEEE Computational Sciences & Eng.*, vol. 2, no. 2, 1995, pp. 50–61.
6. D.L. Donoho and X. Huo, "Uncertainty Principles and Ideal Atomic Decomposition," *IEEE Trans. Information Theory*, vol. 47, no. 7, 2001, pp. 2845–2862.
7. D.L. Donoho, "For Most Large Underdetermined Systems of Linear Equations the Minimal ℓ_1 -Norm Solution Is also the Sparsest Solution," *Comm. Pure and Applied Mathematics*, vol. 59, no. 6, 2006, pp. 797–829.

8. D.L. Donoho, "Compressed Sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, 2006, pp. 1289–1306.
9. D.L. Donoho and Y. Tsaig, "Extensions of Compressed Sensing," *Signal Processing*, vol. 86, no. 3, 2006, pp. 549–571.
10. V. Stodden, *Model Selection When the Number of Variables Exceeds the Number of Observations*, doctoral dissertation, Dept. Statistics, Stanford Univ., 2006.
11. I. Rahman et al., "Multiscale Representations for Manifold-Valued Data," *SIAM Multiscale Modelling and Simulation*, vol. 4, no. 4, 2005, pp. 1201–1232.
12. I. Rahman, *Multiscale Decomposition of Manifold-Valued Data*, doctoral dissertation, Scientific Computing and Computational Math Program, Stanford Univ., 2006.
13. M. Shahram, D.L. Donoho, and J.L. Stark, "Multiscale Representation for Data on the Sphere and Applications to Geopotential Data," *Proc. SPIE Ann. Meeting*, SPIE Press, 2007, pp. 67010a1–67010a11.

David L. Donoho is a professor at the University of California, Berkeley, and at Stanford University. His research interests include computational harmonic analysis, high-dimensional geometry, and mathematical statistics. Donoho has a PhD in statistics from Harvard University. He is a member of the American Academy of Arts and Sciences and the US National Academy of Sciences. Contact him at donoho@stat.stanford.edu.

Arian Maleki is pursuing an MS in statistics and a PhD in electrical engineering at Stanford University. His research interests include signal processing, applied mathematics, and machine learning. Maleki has an MSc in electrical engineering with honors from the Sharif University of Technology. Contact him at arianm@stanford.edu.

Inam Ur Rahman works at Apple Computer. He has a PhD from Stanford University in scientific computing and computational mathematics. Contact him at inam@apple.com.

Morteza Shahram is a postdoctoral research fellow at Stanford University. His research interests are in statistical signal and image processing and computational mathematics. Shahram has a PhD in electrical engineering at the University of California, Santa Cruz. Contact him at mshahram@stanford.edu.

Victoria Stodden is a research fellow at the Berkman Center for Internet and Society at Harvard University. Her current research includes understanding how new technologies and open source standards affect societal decision-making and welfare. Stodden has a PhD in statistics at Stanford University and an MLS from Stanford Law School. Contact her at vcs@stanford.edu; <http://blog.stodden.net>.