15 Years of Reproducible Research in Computational Harmonic Analysis

David Donoho[1,4] Arian Maleki[1,5] Inam Rahman[2,6]
Morteza Shahram[1,7] Victoria Stodden[3,8]

April 15, 2008
Revised Aug 2008

Abstract

Scientific Computation is emerging as absolutely central to the scientific method. Unfortunately, it is error-prone and currently immature: traditional scientific publication is incapable of finding and rooting out errors in scientific computation; this must be recognized as a crisis. Reproducible computational research, in which the full computational environment that produces a result is published along with the article, is an important recent development, and a necessary response to this crisis.

We have been practicing reproducible computational research for 15 years and integrated it with our scientific research, and with doctoral and postdoctoral education. In this article, we review our approach, how the approach has spread over time, and then discuss some of the arguments for and against working reproducibly.

[1] Department of Statistics, Stanford University, Stanford, CA 94305
[2] Apple Computer, Cupertino, CA
[3] Berkman Center for Internet and Society, Harvard Law School, Cambridge, MA 02138
[4]donoho@stanford.edu, [5] arianm@stanford.edu, [6]inam@apple.com,
[7]mshahram@stanford.edu, [8]vcs@stanford.edu

Index Terms: Reproducible Research, Scientific Computing, Computational Science, Scientific Publication

1 The Crisis

Massive computation is transforming science. In field after field, ambitious projects involving large-scale computations are being launched daily. Emblems of our age include:

- data mining for subtle patterns in vast databases; and
- massive simulations of the complete evolution of a physical system – repeated many times over, as parameters of the simulation are varied systematically.

The traditional popular picture of the scientist as a solitary person working in a laboratory with beakers and test-tubes or as a hunched figure scribbling with pencil and paper in the back aisles of some library is long obsolete; the more representative picture – not yet really well-recognized – would be a computer jockey working at all hours to launch experiments on compute servers. Today's academic scientist will have more in common with the information technology manager

of a large corporation than with a Philosophy or English professor at the same university, as one can tell from conversations at town-and-gown cocktail parties or waiting lounges at airports.

The rapid transition now underway – taking place particularly over the last two decades, and nearing completion – will finish with computation as absolutely central to the scientific enterprise. However, the transition will not be smooth; in fact we believe that it has already brought us to a state of crisis. The vast body of results being generated by current computational science practice suffer a large and growing credibility gap: *it is impossible to verify most of the computational results shown in conferences and papers.*

To understand our claim, and the necessary response, we need look at the scientific process more broadly. The methodological branches of science classically were two in number – *deductive* (eg. mathematics) and *empirical* (eg. statistical data analysis of controlled experiments). Many believe that computation (eg large-scale simulation) will soon be accepted as a third branch of the scientific method [1]. Although this will undoubtedly be the case someday, claims that this has already happened are mistaken: *current computational science practice does not generate routinely verifiable knowledge.*

The central motivation for the scientific method as practiced in the two classical methodological branches is the *ubiquity of error* – the phenomenon that mistakes and self delusion can creep in absolutely anywhere, and that the work of the scientist is primarily about recognizing and rooting out error.

Deductive science, while powerful, is error-prone. Human reasoning is plagued by vagueness, wandering attention, forgetfulness and confusion. The young mathematician faces a lengthy struggle over many years to discipline thoughts and expressions, to identify and root out mistakes; the popular notion of the theorist with pencil and paper throwing out ream after ream of failed calculations is, while humorous to cartoonists, also very true. Even the mature mathematician at a typical moment will be struggling with fixing some mistake in a calculation or deduction. The formal notion of mathematical proof is a mature response to the ubiquity of errors in mathematical theorizing.

Empirical science, also very powerful, is also error-prone. Data tend to be noisy and random samples contain misleading apparent patterns. It is easy to mislabel data or misinterpret the results of calculations. The young data analyst faces a lengthy struggle over many years to develop disciplined habits of collection, treatment, and processing of data, and a profound understanding that measurement error inevitably causes mistakes in a substantial fraction of conclusions. Even the mature empirical scientist, at any given moment, will be struggling with some obscure mistake in the interpretation or processing of a dataset, or with the fact that the data are corrupted, or simply that the random noise has sent a confusing signal. The formal notion of statistical hypothesis testing is a mature response to the fact that noisy data sow confusion; the standard format of empirical research papers – data, materials, methods – is a mature response to the fact that every single detail counts – every aspect of data collection and analysis may be a source of mistaken conclusions.

The would-be third branch of scientific process – computation – is again 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, disciplined programming efforts, organized program testing schemes, and so on. The point we make is that tendency to error is central to every application of computing.

[1] Many believe that this acceptance has already occurred; witness numerous grant proposals, keynote speeches and newsletter editorials.

In stark contrast to the other two branches of scientific methodology, computational science is far less mature and far less concerned about the ubiquity of error. At conferences and publications in computational science fields, it is now completely acceptable for someone to simply say: "here is what I did, and here are my results". Almost no time is devoted to explaining to the audience why one should believe that errors have been found and eliminated. The core of the presentation is not about the struggle to root out error – *as it would be in mature fields* – it is rather a *sales pitch*: an enthusiastic presentation of ideas and a breezy demo of an implementation.

The crisis of credibility that computational science faces arises from the fact that it should learn from the mature branches of scientific endeavour, but has not. *Like* those branches, computation suffers from ubiquity of error as a central phenomenon; *unlike* those fields, computational science today does not insist that the value of a piece of research is *identical* to the degree of rigorous effort deployed in systematically identifying and rooting out error and documenting the process of rooting out error.

Mature branches of science, despite all their efforts, suffer severely from the problem of errors in final, published conclusions [8]. Computational science has nothing like the elaborate mechanisms of meta-analysis and large multicenter clinical trials. How dare we imagine that computational science, as routinely practiced, is reliable! Many researchers using scientific computing *are not even trying* to follow a systematic, rigorous discipline that would in principle allow others to verify the claims they make.

2 A Practical Response

Jon Claerbout saw this crisis coming, more than 20 years ago. Working in exploration geophysics, which requires data analysis and the development of new computational algorithms, he saw that the information being conveyed in publications in computational science was seriously incomplete. His insight, as paraphrased in [2]: "an article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."[1]

Claerbout, together with his Stanford Exploration Project team, developed a system for document preparation which attempted to encapsulate the full scholarly content of a body of work. When reading an article using the special document viewer created by his team, one was actually inside the complete computational environment (code and data) that generated the results presented in the article. The code underlying the figures seen in the papers was hyperlinked to the document and this code could be studied and even rerun, producing the results anew from scratch. Moreover, parameters could be modified from their original version, thus allowing the reader to interact with the author's calculations.

While most scientists can more easily understand these concepts today than twenty years ago, and the technology is much more easy to build today than twenty years ago, the approach is still not widely practiced and is certainly not rigorously required.

3 Our Own Efforts

In the early 1990's, Donoho became aware of Claerbout's ideas and began to practice them in his own research. Roughly speaking, he didn't trust himself to do good work unless his results were

[1] http://sepwww.stanford.edu/research/redoc/

subject to open criticism by others. He also learned from experience not to believe that graduate students were doing the work they believed themselves to be doing unless he studied and ran the code himself. The internet was at that time becoming a standard tool. Cross-platfrom high-level quantitative programming environments like 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 the standard language and computing environment Matlab, and made available on the internet, so that interested parties could reproduce the calculations underlying that paper. He also decided to impose that discipline on students. At the time he was working in computational harmonic analysis (wavelets, wavelet packets, time frequency methods) and very little in the way of computational tools was available. After several papers on the subject of wavelets had been written, it became possible 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 any of several 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 is available for use on numerous operating systems and hardware and functions as a programming language free of machine dependencies. WaveLab is installed as a Matlab toolbox, and has been available on-line in some version since 1993. The original version had over 700 files, including .m files, C-code, data, and documentation. It has since grown. For a fuller description, see [2].

While many Matlab toolboxes were available for a wide range of purposes even before WaveLab's release, it seems that WaveLab was the first effort to provide a toolbox explicitly for the purpose of reproducing results in a series of computational science papers. It seems that the psychology of researchers at the time (and mostly since) would not have been to publish a complete environment to allow reproducibility of a series of papers. Instead, either one did not release the algorithms and data to the outside world, or else, one made a sketchy effort at algorithm publication, not suitable for complete reproducibility.

Because several papers were being reproduced by the original WaveLab release, the package itself contained a fairly complete offering of wavelet transforms and time-frequency analysis tools. It had also been used in teaching and contained numerous pedagogical worked examples. As a result, although the central motivation in the authors' minds was reproducibility, users often looked at it as a general piece of free software competitive with commercial packages.

Reproducibility had the effect of making the papers in WaveLab a kind of benchmark, in the following sense. The papers documented the performance of certain signal processing techniques on certain datasets. Other researchers developing new techniques often used those same datasets and the specific performance measures used in those papers as a way to document the performance of their own new procedures and to compare with the methods in WaveLab. Partly as a result, WaveLab became highly cited. An analysis of citation counts shows that during the period 1996-2007, WaveLab received 690 citations; the citation count by year is shown in Figure 1. Analysis of use patterns is given in Table 1.

4 Why Do This?

Why was it deemed important to proceed in this way? Several reasons are mentioned in [2], the thrust being that *if everyone in a research team knows that everything they do is going to someday be published for reproducibility, they will behave differently and will do better work.* Thus, striving for reproducibility imposes a discipline that leads to better work.

It is a fundamental fact that in striving for reproducibility, we are producing code for the use of strangers. While helping strangers may seem unnatural to many, and certainly impedes

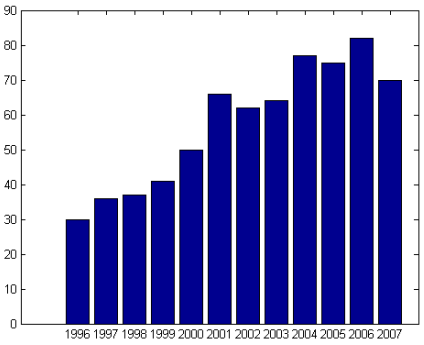

Figure 3.1: Citation counts for WaveLab by year. Source: analysis of Google Scholar search results.

Pct.	Use Pattern
74%	used algorithms in WaveLab to process their own data
11%	used both algorithms and datasets of WaveLab
3%	used only datasets from WaveLab
12%	compared performance of their own algorithms with algorithms in WaveLab
100%	total

Table 1: Usage pattern analysis of WaveLab citations. 100 randomly-chosen citations were studied out of 690 visible in Google Scholar.

the spread of reproducible research, note that "stranger" really means anyone not in possession of our current short-term memory and experiences. In the heat of a computational project we have many things stored in short-term memory that are needed at that moment to use the code productively. Developing for a stranger means avoiding reliance on this soft, transient knowledge; more specifically, codifying that knowledge objectively and reproducibly.

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

- Our co-authors.
- Our current graduate students.
- Our future graduate students.
- Our new postdoctoral fellow.
- Anonymous referees of our papers and grant proposals.
- Future Employers.

All of these 'strangers' would not share all the soft, transient knowledge we have recently accumulated in developing our own computational result. In fact, we ourselves, years from now, will be such strangers, not remembering the myriad of small details that accumulated in our mind during this project.

For example, it is not uncommon for a researcher who does *not* follow reproducibility as a personal discipline to forget how the software in some long-ago project is used, or what its

limitations are, 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 as anyone else could.

Thus in working for reproducibility, we may be helping ourselves and our friends, even if it seems we are also helping strangers. Is it then really necessary to help strangers? We think that it *is*. Only if we work *as if* a stranger has to use the code without complaint will we produce code that is truly reproducible. Only if we know *that the code really is going to be used by strangers* will we really work *as if* the code is going to be used by strangers. QED.

In short, striving for truly reliable computational results seems, to us, inseparable from dedicating our efforts to producing a package that can be run from scratch by an anonymous stranger and produce our results.

5 Spread of Our Framework

We have now worked within the reproducibility paradigm for 15 years and during that period have tried to get our collaborators, students, and postdocs to participate. Not all of them have agreed to do so, a point to which we return later. In the extended version of this paper, we give dozens of citations of published work and PhD theses, supporting these efforts. At the URL `http://www-stat.stanford.edu/~wavelab/ReproducibilitySpread.pdf` thisinformation in hyperlinked way. Information about our work can be pursued by consulting that URL.

We have created a family of toolboxes, in some way related to wavelab and complementing it in some way:

- *Atomizer.* Toolbox for overcomplete representation of signals by ℓ_1 minimization;
- *BeamLab.* Toolbox for multiscale geometric analysis;
- SymmLab. Toolbox for multiscale analysis of manifold-valued data;.
- *SparseLab.* Toolbox for sparsity-seeking decomposition and reconstruction; and
- *SphereLab.* Toolbox for multiscale decomposition of data on the sphere.

All the toolboxes share a common arrangement of files, and a common scheme for documentation and organizing the information. This arrangement, described in the extended version of this article and references given there, helps new graduate students and postdocs to make their code reproducible. Essentially, an author working on a novel project an download one of these packages, study its file structure and treat that structure as a model. Next the researcher can clone the files in the package, rename the files and then edit them, creating an entirely new package in a different topic area.

Over time the packages in question – in particular WaveLab – have grown through additions, and even the most recent packages have had such additions. While many of these additions have come from Ph.D. students or postdocs in our group, we also have had contributions from 'outsiders', as mentioned above. The modular design of the framework is such that it is fairly easy to simply 'drop in' a new folder into a toolbox and create a new distributable archive.

We have also apparently inspired former students to develop their own frameworks

- Professor Emmanuel Candès in Applied and Computational Mathematics has worked with Ph.D. students L. Demanet, L. Ying, and J. Romberg to produce several reproducibility-inspired packages, including

  - CurveLab for curvelet analysis and
  - l_1 Magic for ℓ_1 minimization. See *http://www.acm.caltech.edu/l1magic/*.

- Professor Xiaoming Huo in ISYE at Georgia Tech has released the package CTDLab which solves "Connect-the-Dots" problems [1].

Researchers elsewhere have also been inspired to build packages at least partly building on our example: Jalal Fadili, Professor of Image and Signal Processing in the Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen, has also developed the open software package MCALab.

One major book has been written using our approach – *A Wavelet Tour of Signal Processing* [9] by Stephane Mallat of Ecole Polytechnique – this uses WaveLab to generate all the figures and examples in his book, the code is published in recent releases of WaveLab. Professor Brani Vidakovic of Georgia Tech is developing a second edition of the book *Statistical Modeling by Wavelets* [14] converting the software from another language to Matlab/WaveLab.

In book publishing a major milestone is achieved with the translation to a foreign language. Astronomer Amara Graps developed the IDL Wavelet Workbench [7] based partially on the WaveLab package; IDL is a popular computing environment for astronomical data analysis reminiscent of Matlab.

6 Example: SparseLab

A key achievement of modern computational harmonic analysis has been to show that many kinds of signal content are highly compressible in the right representation. Thus, images are compressible in the wavelets basis or curvelets frame; acoustic signals may be compressible in local cosine bases, and so on. This is a central phenomenon of the modern information era; we exploit it every time we download movies and music from the web or use a cellphone to view multimedia content.

The phenomenon of compressibility discovered by harmonic analysts is of a particularly useful form: the coefficients in the representation are *sparse* – many are essentially zero, and can be ignored. Sparsity has fundamental advantages extending far beyond the obvious ones of data compression. In recent years, a great deal of research activity has been inspired by the fact that sparsity allows for the correct solution of underdetermined systems of linear equations. That is, an underdetermined system of equations may well have many solutions, but only one highly sparse solution; see [6, 4]. Implications include the notion of *Compressed Sensing*, which holds that, when the underlying object has a highly sparse representation, it is not necessary to take as many samples as pixels, but instead only to take as many samples as the underlying number of degrees of freedom [3].

The field of sparse representation is a very rapidly developing one. The topic of 'sparse solution of underdetermined systems of linear equations' was classified by ISI Thompson as a New Research Front in May 2005. One of the core papers reproduced by Sparselab, 'Extensions of Compressed Sensing', [5], was named the most-cited paper to appear in the journal *Signal Processing* in 2006. The article 'Compressed Sensing' appeared in 2006; by 2008 about 50 papers with some variant of the words 'Compressed Sensing' appeared in the Magnetic Resonance in Medicine Meeting, and the March 2008 issue of IEEE Signal Processing magazine (circulation 20,000) was devoted to this topic.

SparseLab is a collaborative effort designed to provide access to code and data for sparse solution of underdetermined systems. It is both a vehicle for authors in the field to make their

papers fully reproducible and it provides researchers with established problems and established algorithms for sparse representation by convex optimization and by heuristic methods.

SparseLab was released in 2006 at `http://sparselab.stanford.edu` and the latest version, SparseLab 2.1, has been downloaded over 7000 times in 2008 alone. A casual search turned up more than a dozen papers that have used SparseLab to generate their results. SparseLab is still growing through volunteer contributions.

The package has a user interface that goes beyond merely reproducing published results, allowing other researchers to interactively modify the assumptions and obtain new figures and results based on variations of the published papers. To do this, all code and data used to generate the figures is included in SparseLab, along with graphical demos that allow an interested reader to adjust parameters and investigate each figure in each paper (see Figure 6.2).

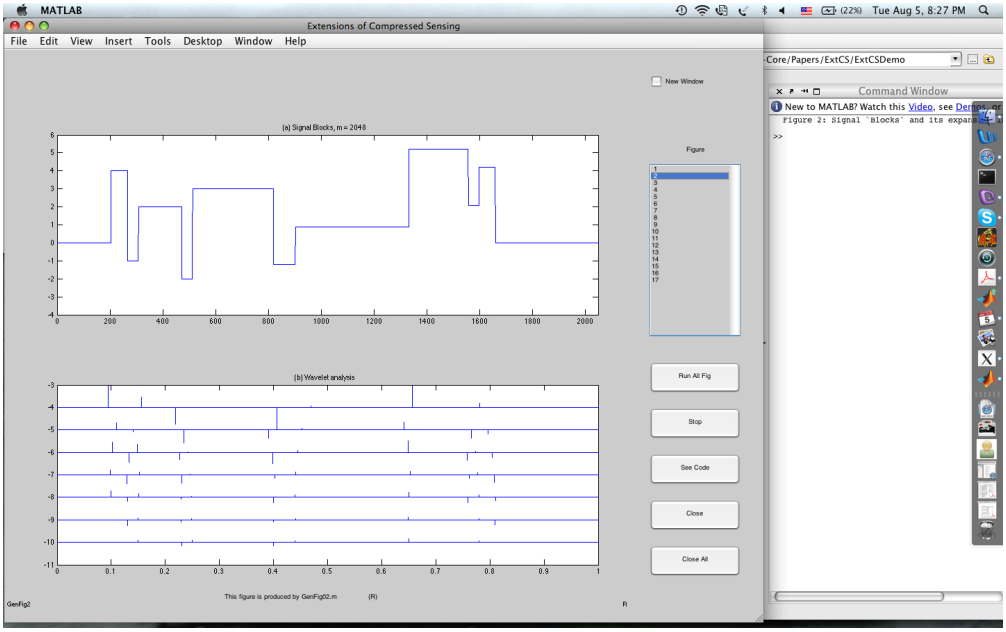

Figure 6.2: A figure from the paper 'Extensions of Compressed Sensing', recomputed and reproduced by SparseLab. Note that the box on the middle right lists all figure numbers in that paper; we are reproducing here Figure 2 in that paper. We could reproduce any other figure by clicking elsewhere in the list. Note the button in the lower right labelled 'See Code'; this reveals the underlying source code used to generate the figure.

SparseLab covers subtopics such as model selection, compressed sensing, shrinkage, Stagewise Orthogonal Matching Pursuit, and Basis Pursuit. There have been ten contributors of code and as many as three people contributing their time to support the effort. Victoria Stodden managed the creation of this reproducible research software package as a component of her Ph. D. thesis [13] .

There are 663 files and folders in the core package, most associated with papers. Each author has contributed the code that produced the results in his or her paper, and SparseLab also includes some general sparse solvers not associated with a particular paper, such as Michael Saunders' Primal-Dual Method for Convex Optimization.

Lesson Learned from SparseLab: Reproducibility can accelerate progress in a hot field. Reproducibility implicitly creates a source of challenge problems and challenge datasets; many

researchers are interested in comparing their own novel algorithms with the algorithms that were first published in the SparseLab package. This in turn leads those researchers to cite the reproducible papers in SparseLab. Some of those researchers have later contributed their own algorithms to SparseLab (about which, more below), thereby making the package even more authoritative as a source for compressed sensing and related applications of sparsity-seeking methods.

7 Example: SymmLab

Traditionally, computational harmonic analysis treats real-valued functions of time or space. Recently, many new technological and scientific problems are generating fascinating datasets which can be modelled as manifold-valued functions of time or space. A simple example is provided by human motion data, where a human body is represented in terms of all the joint angles (eg elbow wrist, shoulder, hip, knee, neck.) Such data may be viewed as taking values, not in the reals, but in a manifold; in this case a product of numerous copies of SO(2) and SO(3), one copy for each joint; the choice of SO(2) vs SO(3) depends on the type of joint flexibility for the given joint. Here SO(k) refers to the collection of orientations of a k-dimensional orthogonal coordinate set in R^k. Another example is provided by aircraft orientation data, where the aircraft's orientation is expressed as a function of time. Traditional coordinates for aircraft orientations – pitch, roll, and yaw – can be converted into SO(3). See Figure 7.3.

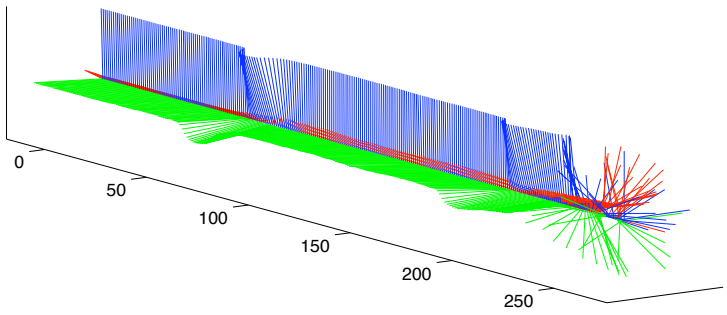

Figure 7.3: $SO(3)$-valued data: aircraft orientation as function of time. Each instant's orientation is indicated by a tripod with blue, red, and green vectors. The airplane lost control and crashed, as reflected in the wild variation in orientation near the end of the series.

It turns out that a very broad collection of manifold-valued data can be modelled as arising from a simple class of manifolds called Riemannian Symmetric Spaces. Such manifolds, in turn, allow for an elegant systematic notion of wavelet transform, as explained in the paper [11]. See Figure 7.4 for a presentation of the $SO(3)$-valued wavelet coefficients of the aircraft orientation data.

Symmlab is a matlab toolbox that plays for symmetric-space valued data roughly the same role that WaveLab plays for more traditional data. It reproduces the paper [11] and the thesis [10]. Symmlab may be the first available toolbox for multiscale analysis of *"Manifold-Valued Data"*.

SymmLab was initially started by Victoria Stodden, followed by major extensions from Inam-Ur-Rahman for his Ph.D. thesis [10];. It resembles *WaveLab* and successors in structure and

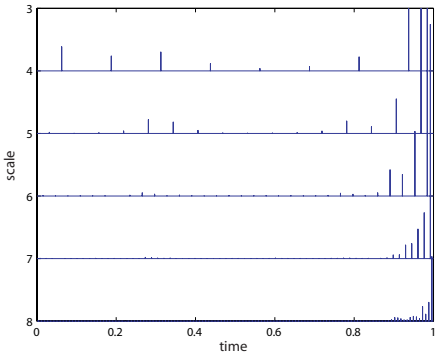

Figure 7.4: Wavelet transform of aircraft orientation as function of time. Transform is portrayed by scale (vertical axis) and time(horizontal axis). Each wavelet coefficient (not presented here) is an element of the Lie algebra $SO(2)$; the norm of each coefficient is presented. The large coefficients at coarse scales and at the end of the series at fine scales demonstrate that the aircraft's motion was smooth until control was lost, at which time motion became energetic at all scales

design philosophy, with valuable improvements related to developments in later packages such as *SparseLab*.

What differentiates SymmLab from its sister packages is the emphasis and effort devoted to acquiring and organizing real data sets. The fraction of examples running on actual data sets to synthetic/toy examples is much higher in *SymmLab* than in *WaveLab* and *SparseLab*, mainly because of the fact that *WaveLab* and *SparseLab* are based on well established algorithms and data formats, the value of which is well-established. Thus the philosophy behind these packages was to provide a platform for reproducible research rather than stressing the actual algorithms themselves. Instead, *SymmLab* aims to create an awareness about the availability of novel data itself and about the urgent and emerging need for tools to analyze this type of data. One may even view the main contribution of the paper [11] as the formalization of this class of datasets, and the main sense in which reproducibility is important here as the diffusion of the datasets rather than the results.

Thus we put a lot of effort in collecting, organizing and documenting the data sets and examples using these data sets in order to make the user realize the ever growing importance of this new type of data and the need for tools to process data in this area. The data sets available in the download come from diverse sources:

- Weak gravitational lensing data from astronomy
- Motion capture data used in animation
- Aircraft motion data from aviation
- Geophysical time varying deformation tensor data
- Diffusion tensor data from medical imaging
- Subspace tracking data from array signal processing
- Wind direction and speed data

- SAR interferometric data

The package is available at `www-stat.stanford.edu/~symmlab`. It contains over 100 .m files which implement core algorithms themselves as well as utilities for generating synthetic data with user defined parameters, cleaning and loading real data sets and tools to visualize these novel data types. The website contains instructions and documentation for registering, downloading, installing and using the package. The paper [11] describing technical details of algorithm can also be downloaded from this website.

Lesson Learned from SymmLab: SymmLab demonstrates an important advantage of reproducible research; it shows that *reproducibility is a way to rapidly disseminate new problems and illustrate new data types.* In this case, the person downloading SymmLab may be not at all interested in wavelets per se, but only in symmetric-space valued data, and in many fascinating application areas. The interested reader can download, install and verify and see the raw data and learn about the problems being addressed in a reproducible paper, rather than the mathematical solutions being proposed. Exact reproducibility has side benefits, because the data and the problem get published, not only the intended solution.

8 Example: Wavelets on the Sphere

Traditional wavelet analysis concerns data which take real values and are indexed by a Euclidean parameter such as time (i.e. an equispaced sampling in time) or two-dimensional space (i.e. a regular cartesian grid in the plane). However in the earth sciences, in aviation, atmospheric sciences, and astronomy, one often deals with real-valued data indexed by the sphere. For such settings one needs wavelet transforms adapted to the spherically-organized data.

There are many ways to represent coordinates and coordinate sampling on the sphere. An attractive option is the HEALPix grid, the basis for much recent work in observational cosmology. The major astronomical organizations (NASA and ESO) now use HEALPix ; see the HEALPix webpage at http://HEALPix .jpl.nasa.gov for the HEALPix software (version 2.00), an open source package including Fortran 90, IDL and C++ sources. Though relatively new and little-known to outsiders, this could also be used in other fields.

The proceedings paper [12] develops wavelet representations for HEALPix -organized data; a full journal article is in preparation. SphereLab is a Matlab toolbox that allows reproducibility of the calculations announced in [12]; it will be released in 2008 when the journal paper is submitted.

The development of SphereLab was conducted as part of Morteza Shahram's postdoctoral fellowship at Stanford; the sponsor was interested in development of a new kind of wavelets for spherical data which would co-operate with traditional spherical harmonics expansions. The HEALPix grid was developed as a pointset on the sphere allowing fast evaluation of spherical harmonics expansions, this makes the HEALPix grid particularly natural for development of wavelets that co-operate with spherical harmonics.

Because of the sponsor's interest, SphereLab has been specially focused for geopotential applications, where we need to evaluate different geo-related quantities and also to improve current geopotential models as new measurements are collected. The main research deliverable was a demonstration of a system for compressing and decompressing a geopotential field on the sphere much more rapidly than traditional spherical harmonics expansions could do. The publication [12] described a set of results demonstrating that this milestone had been acheived, and the SphereLab package allows the sponsor (and others) to verify this claim.

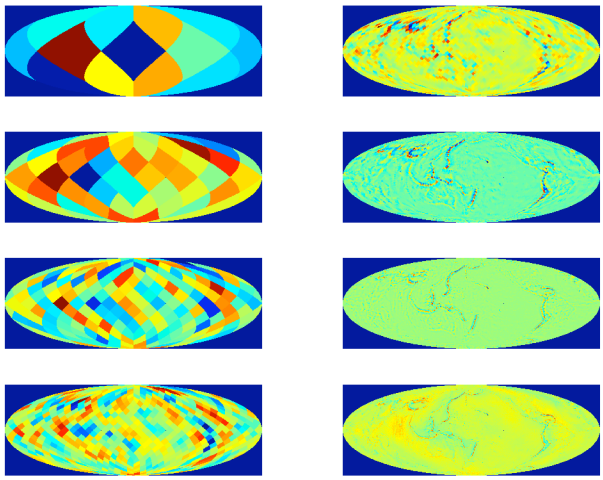

Figure 8.5: Multiscale decomposition of Global Geopotential using HEALPix Wavelets. The new high-resolution EGM-06 dataset became available in 2006; it is used in aircraft navigation and contains unprecedented data density. The National Geospatial Intelligence Agency has been interested in multiresolution representation of this massive dataset. Different panels represent different scales of the Healpix Wavelets decomposition. Fine scales features near mountain ranges and continental boundaries are evident.

Conceivably SphereLab will be an easily accessible and useful tool for non-geoptential applications such as 3-D object rendering in computer graphics. SphereLab include tools for conversions between HEALPix and HEALPix wavelets, refinement schemes on the sphere, statistical analysis, demos, test files etc. Overall this package has about 150 main files and about the same number of test files. A multiscale decomposition of the global geopotential is shown in Figure 8.5.

Lesson Learned from SphereLab: SphereLab demonstrates that *reproducibility is a way for research sponsors to ensure the deliverables assume tangible form.* In this case, the sponsor is not at all interested in the academic paper, but is interested in the demonstration of the principles discussed in the paper. The sponsor can download, install and verify the proper functioning of the code and see that its money has been well spent.

9 Reproducibility as Software Publishing

Reproducibility requires publishing for strangers a code base that reproduces some computational results. In consequence, one is inevitably working in software publishing. This may be a new experience for academics, and it surely adds a new dimension beyond the journal publishing experience.

9.1 Tech Support

We hear many positive comments from users of our software – but we also hear problem reports. Several times a year we hear of problems in one of our packages. Often these are mundane issues,

most frequently an installation problem. Occasionally, there are more serious issues, often to do with the release of a new version of MATLAB.

Tech support is both burden and an opportunity. We have often assigned tech support tasks to a graduate student who is just starting work on Ph.D. 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 of the importance of doing a high quality, lasting job in their software development efforts. They realize we really mean it when we say all their computational work may some day be subject to public scrutiny.

9.2 Code Maintenance

Much of the Matlab code we wrote for the original WaveLab package 15 years ago still runs correctly. A small number of basic utilities functions have had to be changed over the years, but otherwise, much of the code runs without problems. This validates the role of Matlab as a uniform computational environment for delivering reproducibility across hardware and operating system variations over 15 years.

The bulk of the work involved in maintaining the code has concerned problems arising when Matlab changed. This is again not totally a bad thing. A graduate student early in the Ph.D career who is assigned a maintenance task will learn that we intend to keep the code working for years in the future and will learn about likely problems involved in doing so. This awareness may prepare the student to do a better job in his/her own future work.

9.3 Maintaining a Web Presence

Of course, reproducibility means publication over the internet. This requires setting up a website, and maintaining it. There is work involved, which again can be informative or educational, but can occasionally be time consuming. It seems to be getting easier all the time.

10 Reflections

10.1 Has it Worked?

It seems that reproducible research gets noticed. As measured by tools like Google Scholar, the core papers being reproduced by WaveLab have received thousands of citations and the core paper of Atomizer, more than a thousand. Even the core papers of Sparselab, relatively recent package, receive in the hundreds of citations.

Although errors have cropped up from time to time, it seems that the code in these packages is largely correct. Many people have now used the code, and it seems to perform as originally hoped. This is what reproducibility really delivers.

10.2 When did we *not* work reproducibly?

Numerous examples of reproducible research emerged from our efforts over the last 15 years. There are also numerous examples of non-reproducibility. Three main reasons cover almost all our examples of failure to work within our framework.

- *Postdocs.* By and large postdocs are young men in a hurry who don't want to absorb some new way of working, but just to use their existing way of working to get some quick publications; and who can blame them?

- *Theorists.* By and large a theory paper might have a few diagrams in it, and perhaps a numerical calculation; the theorist 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 projects of a computational nature which might have benefited from reproducibility, except that they were too small scale to justify building an entire toolbox which would then have to be maintained. Also there was no prospect of any of the authors continuing research in the given direction, so motivation for the required investment was very weak.

An advisor cannot *force* reproducibility on collaborators. Researchers have to believe that reproducibility is in their interests. Postdocs, in particular are generally panicked about their future prospects and short-sighted – although not always. Thomas Yu, Georgina Flesia and Morteza Shahram are particularly to be cited as Postdoctoral Scholars willing to work reproducibly.

It seems to us that reproducibility can be a very successful part of graduate education; it is less likely to take hold at the postdoctoral stage.

10.3 Snappy Answers to Knee-Jerk Objections

Many researchers exhibit an instinctive aversion to working reproducibly; when they vocalize their instincts, there are often easy 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 worry about also publishing correct and polished code. Actually, however, our goal is to educate future leaders; this takes time and requires effort. Moreover, if we view our job in graduate education as actually reviewing and correcting the work of students, this is dramatically easier to do if they work reproducibly, so in many cases there is actually less total time and effort than otherwise.

- *No one else does it. I won't get any credit for it.* Response: Partly true. Few people today work reproducibly. However, *exactly because of this*, if one's code is available, one's work will get noticed and actually used, and because it will get used, it will eventually become a reliable trusted tool. Would you prefer for your work to not be trusted, for it to be ignored, or for someone else's implementation of *your idea* to be better known simply because they make it available?

- *Strangers will use your code to compete with you.* Response: True. But competition means 'strangers will read your papers and try to learn from 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 is a winning argument if you must work on a 10,000 node cluster using idiosyncratic hardware. But who can believe that your obscure computing environment is doing what you say it does? Also, how can you believe it? Shouldn't you test your computations on a standard environment, also test it in the idiosyncratic one, and verify that identical results obtain? Isn't reproducibility more important in this case, not less?

10.4 Sober Answers to Thoughtful Objections

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

- *Reproducibility Undermines the Creation of Intellectual Capital.*

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

  Response: this argument applies after the Ph.D. – to a postdoc – but not before. It also does not apply to senior researchers. For a senior researcher running a lab where Ph.D. students come for a time and then leave, actually *the only way to ensure the creation of valuable capital* is to require the students 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, hadn't you best learn *now* how to organize substantial computational projects in such a way that you and colleagues can benefit from them down the road? Won't potential employers be more impressed by your skills if you are able to show them that your work is reproducible and transparent to them? Won't potential senior collaborators view you as more valuable as a partner if they believe they can work transparently with code that you develop? Isn't the problem of postdocs the one of getting noticed? Isn't having other people using your code the same thing as getting noticed?

- *Reproducibility Destroys Time-Honored Motivations for Collaboration*

  Argument: traditionally, science develops by collaborations in which scientists visit each others' laboratories and in quid-pro-quo arrangements share tools and ideas. 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, the social fabric of science is weakened. As science is primarily a social entity, this is quite serious.

  Response: This is the 'intellectual capital argument' in disguise.

  Reproducibility is even important within one laboratory or within a team of investigators, or even for future use of the same code by the same investigator. Even if one were working in a secret establishment, air gapped to the outside world, working reproducibly would make sense

  The issue is thus not reproducibility, but widespread publication. One always has the option to work reproducibly but then modify or skip the last step, namely giving away one's code. One can even provide remote, partial, controlled access to the computing environment underlying a research project.

  - A traditional approach: publish only the binary code and not the source code of one's calculations.
  - A more modern approach: set up a web server that allows outsiders to access proprietary algorithms in client-server fashion. Outsiders would upload datasets to the server to be processed by those algorithms and could be offered the option to select among the actual datasets used in some article which is being reproduced. This allows others to check the accuracy and timing results of an implementation while not giving it away.

- *Reproducibility Floods Journals with Marginal Science.*

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

- *True Reproducibility Means Reproducibility from First Principles.*

  Argument: It proves nothing if I point and click and see a bunch of numbers as expected. It only proves something if I start from scratch and build your system and in my implementation I get your results.

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

10.5 Does Our Framework still Apply?

Jon Claerbout's original idea envisioned people working from a hyperdocument interface where, as they were reading 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 of the people who find reproducibility interesting do so for reasons we wouldn't have predicted – i.e. 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 is going on.

Nevertheless, we can imagine many, many ways our framework could be modernized. A principal development of the last five years is the spread of 'Social Networking', 'Wikipedia' , 'SourceForge' and related phenomena. Deploying such ideas in connection with reproducible research might allow anonymous users to post improvements to packages on their own, without real involvement from our own group. Instead, we require that users wanting to add an improvement to one of our packages contact us and work with us. At the moment, we don't have evidence of a reason to abandon our old-fashioned approach – but we could be admittedly out of touch.

11 Conclusion

The formalization of reproducible computational science is a lengthy process. Its completion will mark an important milestone in the transition of computational science to a mature third branch of the scientific method, to stand alongside the empirical and deductive branches as major contributors to the progress of knowledge. Fifteen years of experience using a simple Matlab-based framework shows that even with today's tools, reproducibility in computational science is possible. Working reproducibly leads to downloads and citations; it is also an important new component of graduate education.

Funding agencies could promote reproducible computational research; this would match their own stated objectives and increase the impact of the research they are sponsoring.

12 Funding Acknowledgements

The 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, DMS 05-05303) its optimization program (Stephen Boyd, PI), an FRG (Multiscale Geometric Analysis, FRG DMS-0140698, David Donoho, PI), an ITR project and a KDI project (Amos Ron, PI; KDI Towards Ideal Data Representation; ITR Multiresolution Analysis of The Global Internet) and its signal processing program. In addition, the Air Force Office of Scientific Research partially supported our work through a MURI project (David Castanon, PI); and the Office of Naval Research through a MURI project (Comotion: Claire Tomlin, PI). Finally DARPA partially partially supported our work through three projects: Efficient Mathematical Algorithms for Signal Processing 1998-2000 and 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, Yazhen Wang.

References

[1] E. Arias-Castro, D. L. Donoho, X. Huo, and C. A. Tovey. Connect the dots: how many random points can a regular curve pass through? *Advances in Applied Probability*, 37(3):571–603, 2005.

[2] J. Buckheit and D. L. Donoho. *Wavelets and Statistics*, chapter Wavelab and reproducible research. Springer-Verlag, Berlin, New York, 1995.

[3] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, april 2006.

[4] D. L. Donoho. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[5] D. L. Donoho and Y. Tsaig. Extensions of compressed sensing. *Signal Processing*, 86(3):549–571, March 2006.

[6] D.L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47:2845–2862, Nov 2001.

[7] A. Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2:50–61, 1995.

[8] J. P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8):e124, 2005.

[9] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, 2 edition, 1999.

[10] I. Rahman. *Multiscale Decomposition of Manifold-Valued Data*. PhD thesis, Program in Scientific Computing and Computational Math, Stanford University, Stanford, CA, July 2006.

[11] I. Rahman, I. Drori, V. Stodden, D. L. Donoho, and P. Schroder. Multiscale representations for manifold-valued data. *SIAM Multiscale Modelling and Simulation*, 4(4):1201–1232, 2005.

[12] M. Shahram, D. L. Donoho, and J.L. Stark. Multiscale representation for data on the sphere and applications to geopotential data. *Proceedings of SPIE Annual Meeting*, 2007.

[13] V. Stodden. *Model selection when the number of variables exceeds the number of observations*. PhD thesis, Department of Statistics, Stanford University, Stanford, CA, 2006.

[14] B. Vidakovic. *Statistical Modeling by Wavelets*. J. Wiley, New York, 1 edition, 1999.