Fomel, S et al 2013 Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments. *Journal of Open Research Software* 1:e8, DOI: http://dx.doi.org/10.5334/jors.ag

SOFTWARE METAPAPER

# Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments

Sergey Fomel,[1] Paul Sava,[2] Ioan Vlad,[3] Yang Liu,[4] Vladimir Bashkardin,[5]

[1] Jackson School of Geosciences, The University of Texas at Austin, Austin, Texas, USA

[2] Center for Wave Phenomena, Colorado School of Mines, Golden, Colorado, USA

[3] TGS, Houston, Texas, USA

[4] College of Geo-exploration Science and Technology, Jilin University, Changchun, Jilin, China

[5] BP, Houston, Texas, USA

The Madagascar software package is designed for analysis of large-scale multidimensional data, such as those occurring in exploration geophysics. Madagascar provides a framework for reproducible research. By "reproducible research" we refer to the discipline of attaching software codes and data to computational results reported in publications. The package contains a collection of (a) computational modules, (b) data-processing scripts, and (c) research papers. Madagascar is distributed on SourceForge under a GPL v2 license https://sourceforge.net/projects/rsf/. By October 2013, more than 70 people from different organizations around the world have contributed to the project, with increasing year-to-year activity. The Madagascar website is http://www.ahay.org/.

**Keywords:** reproducibility, data analysis, geophysics, seismology, Python

## (1) Overview

### Introduction

Reproducible research, as defined by Jon Claerbout [1], refers to the discipline of attaching software code and data to scientific publications, in order to enable independent verification and replication of computational experiments. The so-called "Claerbout's principle" [2,3] states that "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." The Madagascar software package implements a computational environment that is designed both for conducting computational experiments in the area of large-scale geophysical data analysis and for attaching links to software code and data in scientific publications in order to enable reproducible research. As of October 2013, Madagascar includes more than 120 scientific papers and book chapters complete with software codes necessary for independent verification and replication of computational results (see http://www.ahay.org/wiki/Reproducible_Documents).

The work on the Madagascar project started in 2003, and the beta version of the package was publicly released in June 2006. Since then, many people have joined the project and contributed to the code. The 1.0 version was released in 2010 and tested by an open community. The community stays in touch using mailing lists, social networks, and annual meetings.

Although the main applications have focused so far on applied geophysics and exploration seismology in particular, the core package is suitable for other scientific fields that require reproducible analysis of large-scale multidimensional data.

### Implementation/architecture

The design of Madagascar follows the Unix principle: "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface." [4] Analysis of complex multidimensional data, such as those occurring in exploration seismology requires multiple steps. In addition, the data size can be too large for storing data objects in memory (a typical modern seismic survey generates terabytes of data). We break the data analysis chain into multiple steps by writing short programs that implement individual steps ("do one thing and do it well") with control parameters specified on the Unix command line. The programs act as filters ("work together") by taking input from a file on disk or from a Unix pipe and writing either to disk or to another pipe. We adopt a universal data format, called RSF (regularly sampled file). The RSF format is based on a text description ("because that is a universal interface") that points to the raw binary data stored in a separate file. Conceptually, an RSF file represents a regularly sampled multi-dimensional hypercube, while the corresponding binary data are stored (or passed through a Unix pipe) in simple contiguous arrays for optimally efficient input/output operations [5].

To assemble data analysis workflows from individual programs, we have adopted SCons, a Python-based make-like utility [6]. SCons configuration files (SConstruct scripts) are written in Python and specify the database of dependencies between input files, programs, and target files. SCons supports other useful features, such as multithreaded execution. In our extension of SCons, we define four specific commands for establishing data-processing dependencies [7]:

- "Fetch" describes a rule for downloading data files from a remote data server or a local data directory.
- "Flow" describes a rule (command or Unix pipeline) for generating one or more target files from one or more (or none) source files.
- "Plot" is similar to "Flow" but the target file is a figure.
- "Result" is similar to "Plot" but the target file is a final "result" figure for inclusion in a publication.

One can think of the Madagascar environment as existing on three different levels that correspond to three different stages of research activities of a computational scientist:

1. Implementing a new computational algorithm for data analysis. This level involves writing low-level programs (command-line modules).
2. Testing a new algorithm or a new workflow by applying them to data. This level involves assembling workflows from existing command-line modules and tuning their parameters through repeated computational experiments to achieve the desired result.
3. Publishing new results. Results from computational experiments (figures in our case) get referenced in papers and included in publications.

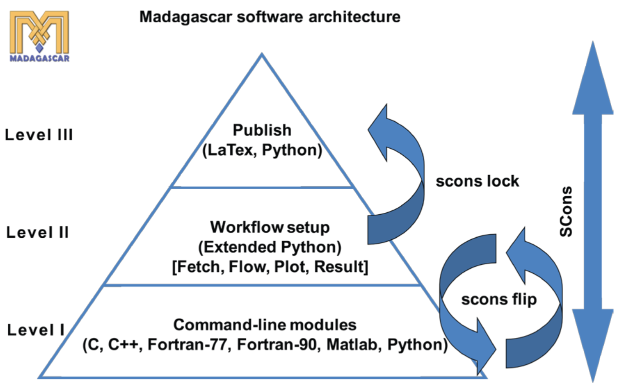

**Figure 1**: Madagascar Software Architecture.

We adopt SCons for the third level as well, to simplify creation of documents that include results from the second level. Customized SCons commands create documents from LaTeX sources with output either in PDF or HTML format. The HTML format is produced using LaTeX-2HTML [8]. In the HTML version, reproducible figures are followed by links to SConstruct scripts from level 2 and low-level programs from level 1 in order to let the reader verify the details of the computational experiment and reproduce it.

### Quality Control

Testing of scientific research codes is important not only for detecting software bugs but also for assuring computational reproducibility and enabling other researchers to expand on published research results [9,10].

The design of Madagascar turns every documented computational experiment into a regression test. The results of an experiment are figures in a custom Vplot format, which are saved in a Subversion repository. When the experiment is repeated, new figures are compared with the saved ones. Testing is simplified by implementing SCons commands "scons test" for testing all results or "scons <result>.test" for testing an individual result and "scons <result>.flip" for visual flipping between the new figure and the previous stored figure in the event that the test fails. The comparison (implemented with sfvplotdiff utility) distinguishes between changes in decoration elements and scientific-content elements and has a tolerance for possible floating-point differences from computational experiments on different architectures.

For providing stable releases, Madagascar installation is tested on a variety of Unix-compliant platforms: different versions of Lunux, Solaris, and MacOS X operating systems, and on Windows under the Cygwin environment.

## (2) Availability

The package is currently available in the source format.

### Operating system

Unix (including Linux, MacOS X, and Unix emulations on Windows such as Cygwin).

### Programming language

Most of the data-processing computational modules are currently written in C. Additional interfaces to the Madagascar library are provided for C++, Java, Python, Fortran-77, Fortran-90, and MATLAB.

Data-processing scripts are written in Python, using SCons, a Python-based make-like building utility [7].

Papers are written in LaTeX.

### Additional system requirements

Certain optional components of the package have additional requirements. For example, CUDA codes require GPU units, large-scale MPI programs require computer clusters, etc. Computations experiments using such resources are "conditionally reproducible" [11].

### Dependencies

The minimal dependency for installation is a C compiler and Python. Other, optional dependencies are configured during the installation process using SCons.

### List of contributors

The full list of contributors is in the AUTHORS.txt file http://sourceforge.net/p/rsf/code/HEAD/tree/trunk/AUTHORS.txt. As of October 2013, the list contains 57 names, not counting authors of additional software components included in the package. Ohloh counts 74 contributors, with the record activity of 18 contributors per month in April 2013 (see http://www.ohloh.net/p/m8r ).

Madagascar uses codes from Vplot, a graphics package developed at Stanford University in the 1980s. The Vplot authors include Jon Claerbout, Steve Cole, Dave Hale, Joe Dellinger, Chuck Karish, Stewart Levin, Dave Nichols, and Shuki Ronen.

### Archive

***Name***
SourceForge

***Persistent identifier***
https://sourceforge.net/projects/rsf/files/madagascar/

***License***
GPL version 2

***Publisher***
Sergey Fomel

***Date published***
09/06/2006

### Code Repository

***Name***
SourceForge

***Identifier***
https://sourceforge.net/p/rsf/code/HEAD/tree/trunk/

***License***
GPL version 2

***Date published***
17/03/2006

***Language***
Subversion repository; Python/SCons scripts for configuration, compilation, and workflow recipes; C libraries, with interfaces to C++, Java, Python, Fortran-77, Fortran-90, and MATLAB.

## (3) Reuse potential

The main goal of community-maintained computational reproducibility in the Madagascar project is to enable other researchers to reuse "computational recipes" from previous numerical experiments and to build new research results on top of previous ones. In the 10-year history of the Madagascar project, there have indeed been several examples of such expansion: one paper generating new results by building on top of results from another paper, sometimes involving scientific collaboration across different continents. We expect more examples of such collaboration in the future.

Although GPL license is more restrictive than BSD-style attribution-only licenses, we find it to be adequate for our needs, because it allows us to protect the integrity of the package and to integrate Madagascar with other GPL-licensed scientific software packages, such as FFTW [12].

### Acknowledgements

We thank all the contributors for making Madagascar a viable open-source research-software project. In addition, we thank Jon Claerbout for promoting the idea of reproducible research and its initial implementation at Stanford, which served both as an inspiration for the Madagascar project and as the source of some of the codes and papers included in the package. Robert Clapp, Martin Karrenbach, Dave Nichols, Matthias Schwab, and other Ph.D. students of Claerbout's participated in the early reproducibility efforts and made invaluable contributions.

### References

1. **Fomel, S** and **Claerbout, J F** 2009 Guest Editors' Introduction: Reproducible Research. *Computing in Science & Engineering* 11(1): 5-7. DOI: http://dx.doi.org/10.1109/MCSE.2009.14
2. **Buckheit, J B** and **Donoho, D L** 1995 WaveLab and reproducible research. In: Antoniadis, A and Oppenheim, G *Wavelets in Statistics.* New York: Springer. pp. 55-81.
3. **de Leeuw** 2001 Reproducible research: the bottom line. *Technical Report 2001031101*. Los Angeles: Department of Statistics Papers, University of California. Available at: http://repositories.cdlib.org/uclastat/papers/2001031101
4. **Salus, P H** 1994 A quarter century of UNIX. Reading, Massachusetts: Addison-Wesley.
5. Guide to RSF file format. Available at http://www.ahay.org/wiki/Guide_to_RSF_file_format [Last accessed on 7 October 2013].
6. **Knight, S** 2005 Building software with SCons. *Computing in Science & Engineering* 7(1): 79-88. DOI: http://dx.doi.org/10.1109/MCSE.2005.11(410)%207
7. **Fomel, S** and **Hennenfent, G** 2007 Reproducible computational experiments using SCons. In: Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing 4: iv-1257. DOI: http://dx.doi.org/10.1109/ICASSP.2007.367305
8. **Drakos, N** 1994 From text to hypertext: a post-hoc rationalisation of latex2html. *Computer networks and ISDN systems* 27(2): 215-224. DOI: http://dx.doi.org/10.1016/0169-7552(94)90135-X
9. **Donoho, D L, Maleki, A, Rahman, I U, Shahram, M,** and **Stodden, V** 2009 Reproducible research in computational harmonic analysis. *Computing in Science & Engineering* 11(1): 8–18. DOI: http://dx.doi.org/10.1109/MCSE.2009.15
10. **LeVeque, R J, Mitchell, I M,** and **Stodden, V** 2009 Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing*

*in Science & Engineering* 14(4):13. DOI: http://dx.doi.org/10.1109/MCSE.2012.38

11. **Schwab, M, Karrenbach, M** and **Claerbout, J** 2009 Making scientific computations reproducible. *Computing in Science & Engineering* 2(6): 61–67. DOI: http://dx.doi.org/10.1109/5992.881708
12. **Frigo, M** and **Johnson, S G** 1998 FFTW: An adaptive software architecture for the FFT. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing 3: 1381–1384. DOI: http://dx.doi.org/10.1109/ICASSP.1998.681704

**How to cite this article:** Fomel, S et al 2013 Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments. *Journal of Open Research Software* 1:e8, DOI: http://dx.doi.org/10.5334/jors.ag

**Published:** 21 November 2013