Sage: Creating a Viable Free Open Source Alternative to Magma, Maple, Mathematica, and MATLAB*

William Stein†

August 28, 2011

Abstract

Sage (`http://www.sagemath.org`) is a large free open source software package aimed at all areas of mathematical computation. Hundreds of people have contributed to the project, which has steadily grown in popularity since 2005. This paper describes the motivation for starting Sage and the history of the project.

1 Introduction

The goal of the Sage project is to create a viable free open source alternative to Magma, Maple™, Mathematica®, and MATLAB®, which are the most popular non-free closed source mathematical software systems.[1] Magma is (by far) the most advanced non-free system for structured abstract algebraic computation, Mathematica and Maple are popular and highly developed systems that shine at symbolic manipulation, and MATLAB is the most popular system for applied numerical mathematics. Together there are over 3,000 employees working at the companies that produce the Ma's, which take in over a hundred million dollars of revenue annually.

A viable alternative to the Ma's will have the important mathematical features of each Ma, with comparable speed. It will have 2d and 3d graphics, an interactive notebook-based graphical user interface, and documentation, including books, papers, school and college curriculum materials, etc.

A viable alternative need not be a drop-in replacement for any of the Ma's; in particular, it need not run programs written in the custom languages of those systems. Thus Sage is nothing like Octave[2]. Development focuses on implementing function that users demand, rather than systematically trying to implement every single function of the Ma's. The culture, architecture, and general feel of Sage is very different than that of any of the Ma's.

*This article will appear in the proceedings of FoCM 2011.

†The work was supported by NSF grants...

[1]Maple is a trademark of Waterloo Maple Inc. Mathematica is a registered trademark of Wolfram Research Incorporated. MATLAB is a registered trademark of MathWorks. I will refer to the four systems together as "Ma" in the rest of this article.

[2]Octave is an open source alternative to MATLAB, that can run many MATLAB programs.

In Section 2 we explain some of the motivation for starting the Sage project. In Section 3 we describe the basic architecture of Sage. Finally, Section 4 sketches some aspects of the history of the project.

2 Motivation for Starting Sage

Each of the Ma's cost substantial money, and is hence expensive for me, my collaborators, and students. The Ma's are not *owned by the community* like Sage is, or Wikipedia is, for that matter.

The Ma's are closed, which means that the implementation of some algorithms are secret, in which case you are not allowed to modify or extend them.

> "Most of the documentation provided for Mathematica is concerned with explaining what Mathematica does, not how it does it. You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose. Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant. Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. [...] But most often the analyses will not be worthwhile. For the internals of Mathematica are quite complicated, and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances."
>
> – Mathematica Documentation, `http://reference.wolfram.com/mathematica/tutorial/WhyYouDoNotUsuallyNeedToKnowAboutInternals.html`

Suffice to say, the philosophy espoused in Sage, and indeed by the vast open source software community, is exactly the opposite.

```
sage: crt(2, 1, 3, 5)   # Chinese Remainder Theorem
11
sage: crt?              # ? = documentation and examples
Returns a solution to a Chinese Remainder Theorem problem.
...
sage: crt??             # ?? = source code
def crt(...):
...
    g, alpha, beta = XGCD(m, n)
    q, r = (b - a).quo_rem(g)
    if r != 0:
        raise ValueError("No solution to crt problem ...")
    return (a + q*alpha*m) % lcm(m, n)
```

Moreover, by browsing `http://hg.sagemath.org/sage-main/`, you can see exactly who wrote or modified any particular line of code in the Sage library, when they did it, and why. Everything in Sage is open source, and always will be that way.

"I see open source as Science. If you don't spread your ideas in the open, if you don't allow other people to look at how your ideas work and verify that they work, you are not doing Science, you are doing Witchcraft. Traditional software development models, where you keep things inside a company and hide what you are doing, are basically Witchcraft. Open source is all about the fact that it is open; people can actually look at what you are doing, and they can improve it, and they can build on top of it. [...] One of my favorite quotes from history is Newton: 'If I had seen further, it has been by standing on the shoulders of giants.'"

– Linus Torvalds, 2011. Listen to it at `http://www.youtube.com/watch?v=bt_Y4pSdsHw`

The design decisions of the Ma's are not made openly by the community. In contrast, important decisions about Sage development are made via open public discussions and voting that is archived on mailing lists.

Every one of the Ma's uses a special mathematics-oriented interpreted programming language, which locks you into their product, makes writing code outside mathematics more difficult, and impacts the number of software engineers that are experts at programming in that language. In contrast, Sage uses the mainstream free open source language Python `http://python.org`, which is one of the world's most popular interpreted programming languages.

The bug tracking done for three of four of the Ma's is secret[3]; thus there is no published accounting of all known bugs, the status of work on them, and how bugs are resolved. But the Ma's do have many bugs; see the release notes of each new version, which lists bugs that were fixed[4]. Sage also has numerous bugs, but at least they are all publicly tracked at `http://trac.sagemath.org`, and there is a well-funded sequence of "Bug Days" workshops devoted entirely to fixing bugs in Sage. Moreover, all discussion about resolving a given bug, including peer review of solutions, is publicly archived. We note that sadly even some prize winning[5] free open source systems, such as GAP `http://www.gap-system.org/`, do not have an open bug tracking system, resulting in people reporting the same bugs over and over again.

None of the Ma's has an optimizing compiler that converts programs written in their language to a faster compiled form.[6] In contrast, Sage is tightly integrated with Cython[7] `http://www.cython.org`, which is a Python-to-C/C++ compiler that speeds up code execution and has support for statically declaring data types (for potentially enormous speedups). For example, enter the following in a cell of the Sage notebook (e.g., `http://sagenb.org`):

```
def python_sum2(n):
    s = int(0)
    for i in xrange(1, n+1):
        s += i*i
    return s
```

And, enter the following in another cell:

[3]We commend MATLAB for having an open bug tracker, though it requires free registration to view.

[4]See also `http://cybertester.com/` and `http://maple.bug-list.org/`.

[5]Jenks Prize, 2008

[6]MATLAB has a compiler, but "the source code is still interpreted at run-time, and performance of code should be the same whether run in standalone mode or in MATLAB."

[7]The Cython project was co-started by Sage developers, but is now generally popular in the world of Python-based scientific computing.

```
%cython
def cython_sum2(long n):
    cdef long i, s = 0
    for i in range(1, n+1):
        s += i*i
    return s
```

The second implementation, despite looking nearly identical, is nearly a hundred times faster than the first one (timings will vary when you try this).

```
sage: timeit('python_sum2(2*10^6)')
5 loops, best of 3: 154 ms per loop
sage: timeit('cython_sum2(2*10^6)')
125 loops, best of 3: 1.76 ms per loop
sage: 154/1.76
87.5
```

WARNING: For big input the second implementation will overflow and give the wrong answer.

Of course, it is better to choose a different algorithm. In case you can't remember a closed form expression for the sum of the first n squares, Sage can deduce it:

```
sage: var('k, n')
sage: factor(sum(k^2, k, 1, n))
1/6*(n + 1)*(2*n + 1)*n
```

And now our simpler fast implementation is:

```
def sum2(n):
    return n*(2*n+1)*(n+1)/6
```

Just as above, we can also use the Cython compiler:

```
%cython
def c_sum2(long n):
    return n*(2*n+1)*(n+1)/6
```

Comparing times, we see that again Cython is faster:

```
sage: n = 2*10^6
sage: timeit('sum2(n)')
625 loops, best of 3: 1.41 microseconds per loop
sage: timeit('c_sum2(n)')
625 loops, best of 3: 0.145 microseconds per loop
sage: 1.41/.145
9.72413793103448
```

In this case, the enhanced speed comes at a cost, in that the answer is *wrong* when the input is long enough to cause an overflow:

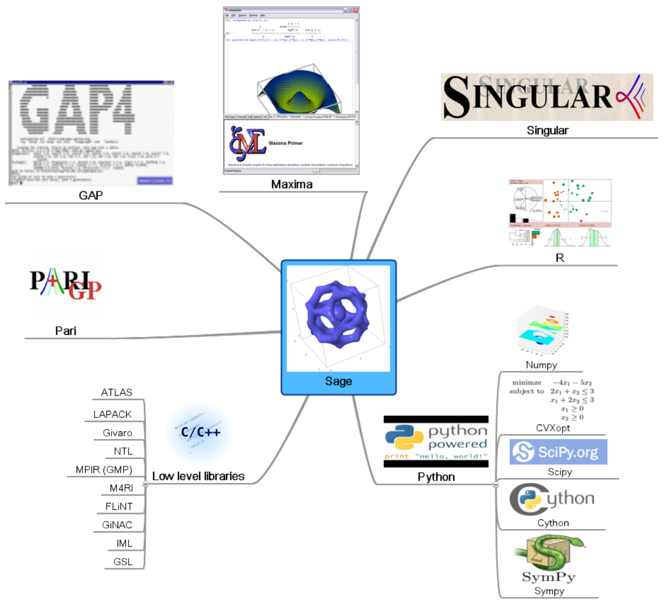
```
sage: c_sum2(2*10^6)    # WARNING: overflow
-4077886789512586O3
```

Cython is very powerful, but to fully benefit from it, one must understand machine level arithmetic data types, such as long, int, float, etc. At least with Sage you have that option.

3 What is Sage?

The goal of Sage is to compete with the Ma's, and the intellectual property at our disposal is the complete range of GPL-compatibly licensed open source software.

Sage is a completely self-contained free open source Python-based *distribution* of about 100 open source software packages and libraries[8] that aims to address all computational areas of pure and applied mathematics. The download of Sage contains all dependencies, *including Python*, required for the normal functioning of Sage. Sage includes a substantial amount of code that provides a unified Python-based *interface* to all of these other packages. Sage also includes a large library of new code written in Python, Cython and C/C++, which implements a range of algorithms that are not available in any other open source library, and in some cases anywhere else at all.



An Incomplete Diagram Illustrating Sage and Some of its Components

Much of the work that the hundreds of Sage developers does goes into writing new code that is included in the core library. They also deal with the never ending task of updating the

[8]See the list at `http://sagemath.org/packages/standard/`.

constantly changing packages included in Sage and testing to what extent the new versions work well together. This involves reporting and sometimes fixing the bugs that result when these package inevitably do not work together. Moreover, as popular new operating systems versions are released, developers sometimes port Sage to run on them.

Google
search the map
Search
Map data ©2011 Geocentre Consulting, MapLink, Tele Atlas - Terms of Use

William Stein, Tim Abbott, Michael Abshoff, Antti Ajanki, Martin Albrecht, Nick Alexander, Bill Allombert, Ethan Van Andel, Ivan Andrus, Pablo Angulo, Benjamin Antieau, André Apitzsch, Maite Aranes, Oscar Gerardo Lazo Arjona, Eviatar Bach, Jennifer Balakrishnan, Jason Bandlow, Gregory Bard, Sébastien Barthélemy, Rob Beezer, Karim Belabas, Arnaud Bergeron, Luis Berlioz, Erin Beyerstedt, Francois Bissey, Jonathan Bober, Tom Boothby, Nicolas Borie, Johan Bosman, Robert Bradshaw, Michael Brickenstein, Nils Bruin, André-Patrick Bubel, Stanislav Bulygin, Dan Bump, Iftikhar Burhanuddin, Paul Butler, Oriol Castejón, Ondrej Certik, Wilson Cheung, Dan Christensen, Craig Citro, Anders Claesson, Francis Clarke, Timothy Clemans, Alex Clemesha, Nathann Cohen, Jenny Cooley, John Cremona, Karl-Dieter Crisman, Fidel Barrera Cruz, Doug Cutrell, Alyson Deines, Vincent Delecroix, Jeroen Demeyer, Tom Denton, Maarten Derickx, Didier Deshommes, Ryan Dingman, Dan Drake, Tom Draper, Alexander Dreyer, Tim Dumol, Nathan Dunfield, Gabriel Ebner, Ben Edwards, Dana Ernst, Burcin Erocal, Ron Evans, Richard J. Fateman, Lars Fischer, Jean-Pierre Flori, Evan Fosmark, Laurent Fousse, Gary Furnish, Alex Ghitza, Andrzej Giniewicz, Alain Giorgetti, Samuele Giraudo, Amy Glen, Daniel Gordon, Chris Gorecki, Jan Groenewald, Rob Gross, Jason Grout, Ryan Grout, Mathieu Guay-Paquet, Alexey U. Gudchenko, Harold Gutch, Jonathan Gutow, Jose Guzman, Anna Haensch, Carlo Hamalainen, Marshall Hampton, Jon Hanke, David Møllerller Hansen, Mike Hansen, Bill Hart, David Harvey, Leif Hille, Florent Hivert, Ryan Hinton, Neal Holtz, Golam Mortuza Hossain, Sean Howe, Alexander Hupfer, Wilfried Huss, Hamish Ivey-Law, Naqi Jaffery, Peter Jeremy, Peter Jipsen, Fredrik Johansson, Niles Johnson, Timo Jolivet, Benjamin Jones, David Joyner, Michael Kallweit, Josh Kantor, Kiran Kedlaya, Lloyd Kilford, Simon King, Keshav Kini, David Kirkby, Emily Kirkman, David Kohel, Ted Kosan, Ross Kyprianou, Sébastien Labbé, Yann Laigle-Chapuy, Kwankyu Lee, Julien Leroy, Richard Lindner, David Loeffler, Miguel Marco, Michael Mardaus, Robert Mařík, Jason Martin, Alexandre Blondin Massé, Peter McNamara, Gregory McWhirter, Jason Merrill, Matthias Meulien, Robert Miller, Kate Minola, Moritz Minzlaff, Joel Mohler, Thierry Monteil, Peter Mora, Bobby Moretti, Rich Morin, Guillaume Moroz, Gregg Musiker, Tobias Nagel, Brett Nakashima, Pablo De Nápoli, Johan Sebastian Rosenkilde Nielsen, Minh Van Nguyen, Andrey Novoseltsev, Christopher Olah, Johan Oudinet, Bill Page, Ronan Paixão, Willem Jan Palenstijn, John Palmieri, Dmitrii Pasechnik, Javier López Peña, Paulo César Pereira de Andrade, David Perkinson, Clement Pernet, John Perry, Pearu Peterson, David Poetzsch-Heffter, Viviane Pons, Bill Purvis, Julien Puydt, Yi Qiang, Jordi Quer, Gustavo Rama, Jens Rasch, Martin Raum, Dorian Raymer, Stefan Reiterer, R. Rishikesh, David Roe, Bjarke Hammersholt Roune, Gordon Royle, Serge A. Salamanka, Franco Saliola, Leonardo Sampaio, Kyle Schalm, Ed Scheinerman, Anne Schilling, Harald Schilly, Jack Schmidt, Michael Schneider, Christopher Schwan, Dag Sverre Seljebotn, Dan Shumow, Denis Simone, Steven Sivek, Nils-Peter Skoruppa, Jaap Spies, Jonathan Spreer, Armin Straub, Marco Streng, Kevin Stueve, Christian Stump, Blair Sutton, Chris Swierczewski, Luis Felipe Tabera Alonso, Glenn Tarbox, Philippe Theveny, Nicolas Thiery, Griffen Thoma, Emmanuel Thomé, John Thurber, Igor Tolkov, Gonzalo Tornaria, Kiminori Tsukazaki, Charlie Turner, Michel Vandenbergh, Joris Vankerschaver, Soledad Villar, John Voight, Felipe Voloch, Steve Vonn, Justin Walker, Mark Watkins, Georg S. Weber, Eric Webster, Ralf-Philipp Weinmann, Joe Wetherell, Carl Witty, Cristian Wuthrich, Soroosh Yazdani, Dal S. Yu, Gary Zablackis, Mike Zabrocki, Bin Zhang, Paul Zimmermann, Mao Ziyang

There are Hundreds of Sage Developers all Over the World

4 History

I made the first release of Sage in February 2005, and at the time called it "**S**oftware for **A**rithmetic **G**eometry **E**xperimentation." I was a serious user of, and contributor to, Magma at the time, and was motivated to start Sage for many of the reasons discussed above. In particular, I was personally frustrated with the top-down closed development model of Magma, the fact that *several million lines* of the source code of Magma are closed source, and the fees that my colleagues had to pay in order to use the substantial amount of code that I contributed to Magma. Despite my early naive hope that Magma would be open sourced, this never happened. So I started Sage with the hope that someday the single most important item of software I use on a daily basis would be free and open. David Joyner, David Kohel and Joe Wetherell were also involved in the development of Sage during the first year.

In February 2006, the National Science Foundation funded a 2-day workshop called "Sage Days 2006" at UC San Diego, which had about 40 participants and speakers from several open and closed source mathematical software projects. After a year of work on Sage, I was very surprised by the positive reception to Sage by members of the mathematical research community.

What Sage promised was something many mathematicians wanted. Whether or not Sage would someday deliver on that promise was (and for many still is) an open question.

I had decided when I started Sage that I would make it powerful enough for my research, with or without the help of anybody else, and was pleasantly surprised at this workshop to find that other people were interested in helping, and understood the significant shortcomings of existing open source software, such as GAP and PARI, and the longterm need to move beyond Magma[9]. Encouraged, six months later, I ran another Sage Days workshop, which resulted in numerous talented young graduate students, including David Harvey, David Roe, Robert Bradshaw, and Robert Miller, getting involved in Sage development. I also learned that there was much broader interest in such a system, and stopped referring to Sage as being exclusively for "arithmetic geometry"; instead, Sage became "**S**oftware for **A**lgebra and **G**eometry **E**xperimentation." Today the acronym is deprecated.

The year 2007 was a major turning point for Sage. Far more people got involved with development, we had four Sage Days workshops, and prompted by Craig Citro, we instituted a requirement that all new code must have tests for 100% of the functions touched by that code, and every modification to Sage must be peer reviewed. Our peer review process is even more open than in mathematics, in that everything that happens is publicly archived at `http://trac.sagemath.org`. During 2007, I also secured some funding for Sage development from Microsoft Research, Google, and NSF. Also, another graduate student in cryptography, Martin Albrecht (from Germany), started contributing to Sage, and presented Sage at the Trophées du Libre competition in France. Sage won first place in "Scientific Software", which led to a huge amount of publicity, including articles in many languages around the world and appearances[10] on the front page of `http://slashdot.org`.

In 2008, I organized 7 Sage Days workshops, and for the first time, several people besides me made releases of Sage. In 2009, we had 8 more Sage Days workshops, and the underlying foundations of Sage improved, including development of a powerful coercion architecture. This *coercion model* systematically determines what happens when performing operations such as `a + b`, when `a` and `b` are elements of potentially different rings (or groups, or modules, etc.).

```
    sage: R.<x> = PolynomialRing(ZZ)
    sage: f = x + 1/2; f
    x + 1/2
    sage: parent(f)
    Univariate Polynomial Ring in x over Rational Field
```

We compare this with Magma (V2.17-4), which has a more ad hoc coercion system:

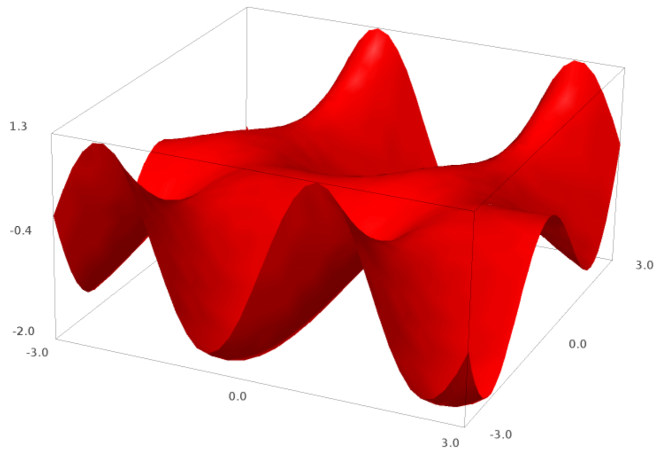
```
> R<x> := PolynomialRing(IntegerRing());
> x + 1/2
    ^
Runtime error in '+': Bad argument types
Argument types given: RngUPolElt[RngInt], FldRatElt
```

Robert Bradshaw and I also added support for beautiful browser-based 3D graphics to Sage,

[9]This was a community of researchers in number theory, for which the Ma's besides Magma are far behind.
[10]For example, `http://science.slashdot.org/story/07/12/08/1350258/Open-Source-Sage-Takes-Aim-at-High-End-Math-Software`

which involved writing a 3D graphics library, and adapting the free open source JMOL Java library (see `http://jmol.sourceforge.net/`) for rendering molecules to instead plot mathematical objects.

```
sage: f(x,y) = sin(x - y) * y * cos(x)
sage: plot3d(f, (x,-3,3), (y,-3,3), opacity=.9, color='red')
```



In 2009, development of algebraic combinatorics in Sage picked up substantial momentum, with the switch of the MuPAD-combinat group to Sage (forming sage-combinat `http://wiki.sagemath.org/combinat`), only months before the formerly free system MuPAD®[11] was bought out by Mathworks (makers of MATLAB).

In 2010, there were 13 Sage Days workshops in many parts of the world, and grant funding for Sage significantly improved again, including new NSF funding for undergraduate curriculum development. I also spent much of my programming time during 2010–2011 developing a highly technical number theory library called psage `http://code.google.com/p/purplesage/`, which is currently not included in Sage, but can be easily installed.

For many mathematicians and students, Sage is today the solid, open, and free foundation on which they can build their research program. It is a community-owned foundation for computational mathematics.

[11]MuPAD is a registered trademark of SciFace Software GmbH & Co.