Sage: Introduction and Status Report

Sage Days 11, Austin, TX

Craig Citro

November 7, 2008

Outline

Outline

What is Sage?

- Sage is open source math software that aims to be a viable, high-quality, free and open source alternative to Magma, Maple, Mathematica, and Matlab.
- Sage is about "building the car instead of reinventing the wheel." This means that as much as possible, Sage uses existing open source libraries and packages instead of spending time repeating existing efforts.
- Sage is built on Python. This means that anything built by the (massive) Python community can be used from within Sage. (I think that choosing Python for Sage was probably the single best decision William Stein made in the whole process.)

What is Sage?

Sage consists of four major "pieces":

- A distribution of a large number of open source math software packages, currently numbering around 70 packages.
- A library of new code, currently over 200,000 lines, providing new functionality.
- Interfaces to lots of other existing math software, both free (e.g. Pari/GP, Singular) and non-free (e.g. Magma, Mathematica, Maple, Matlab).
- A friendly and open community of users and developers.

Why? A Call to Arms ...

J. Neubüser, Creator of GAP

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but ... for many computer algebra systems license fees have to be paid regularly for the total time of their use. ... You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research ... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Why? A frightening quote ...

Mathematica Tutorial:

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. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

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.

Sage – What's inside?

Sage comes standard with over 70 packages, including:

Arithmetic	**GMP, MPFR, Givaro, MPFI**
Commutative Algebra	**PolyBoRi, SINGULAR** (libSINGULAR)
Linear Algebra	**LinBox, M4RI, IML, fpLLL**
Cryptosystems	**GnuTLS, PyCrypto**
Integer Factorization	**FlintQS, ECM**
Group Theory	**GAP**
Combinatorics	**Symmetrica, sage-combinat**
Graph Theory	**NetworkX**
Number Theory	**PARI, NTL, Flint, mwrank, eclib**
Numerical Computation	**GSL, Numpy, Scipy, ATLAS**
Calculus, Symbolic Comp.	**Maxima, Sympy, Pynac**
Statistics	**R**
User Interface	Sage Notebook, **jsmath, Moin wiki, IPython**
Graphics	**Matplotlib, Tachyon, libgd, JMol**
Networking	**Twisted**
Databases	**ZODB, SQLite, SQLAlchemy**, Python pickle
Programming Language	**Python, Cython** (compiled)

Sage's many uses:

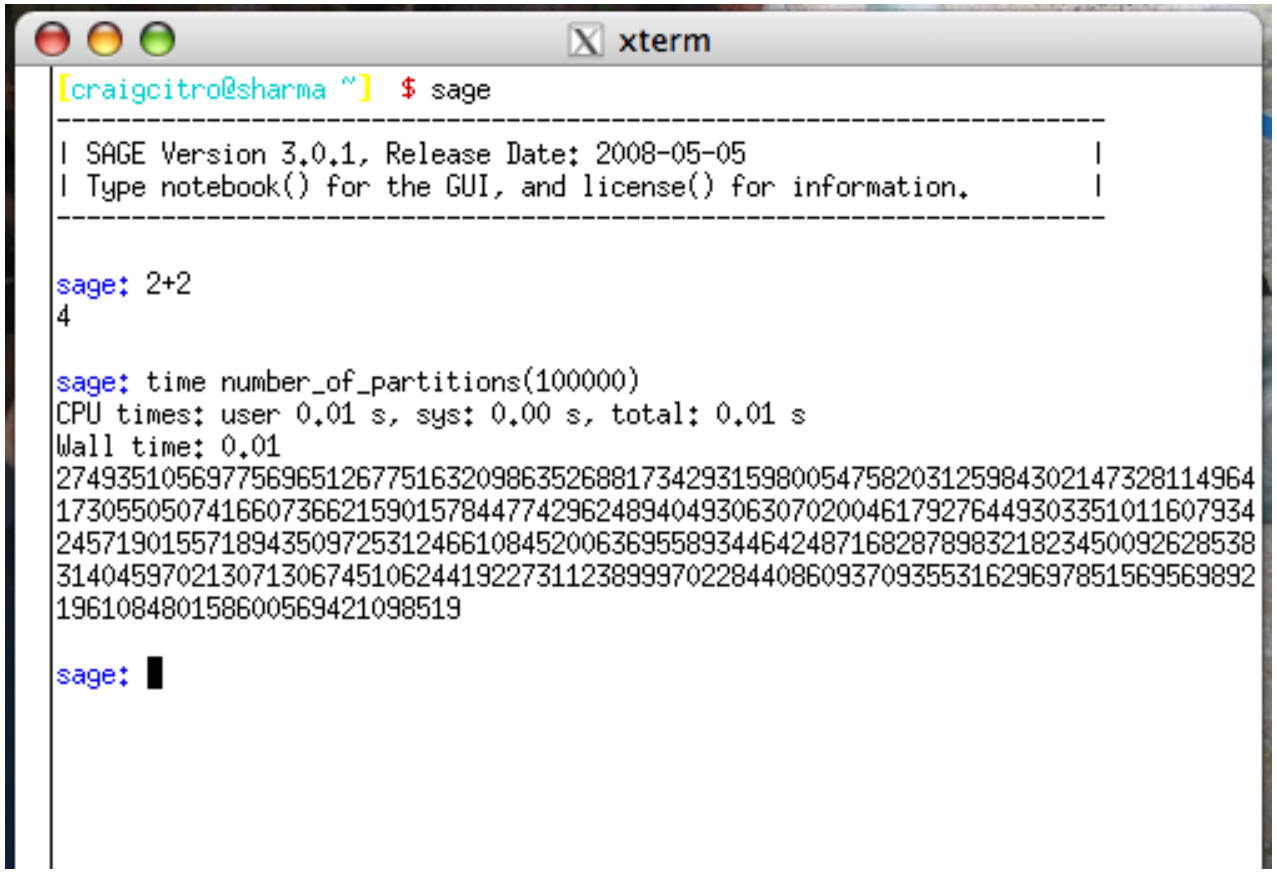
Neil Sloane

```
From: N. J. A. Sloane <njas@research.att.com>
Date: 8 Nov 2007 06:28
Subject: Re: dumb question about installing pari-gp with fink

I would like to thank everyone who responded to
my question about installing PARI on an iMAC.

The consensus was that it would be simplest to install sage,
which includes PARI and many other things.

I tried this and it worked!

Thanks!

Neil

(It is such a shock when things actually work!!)
```

Outline

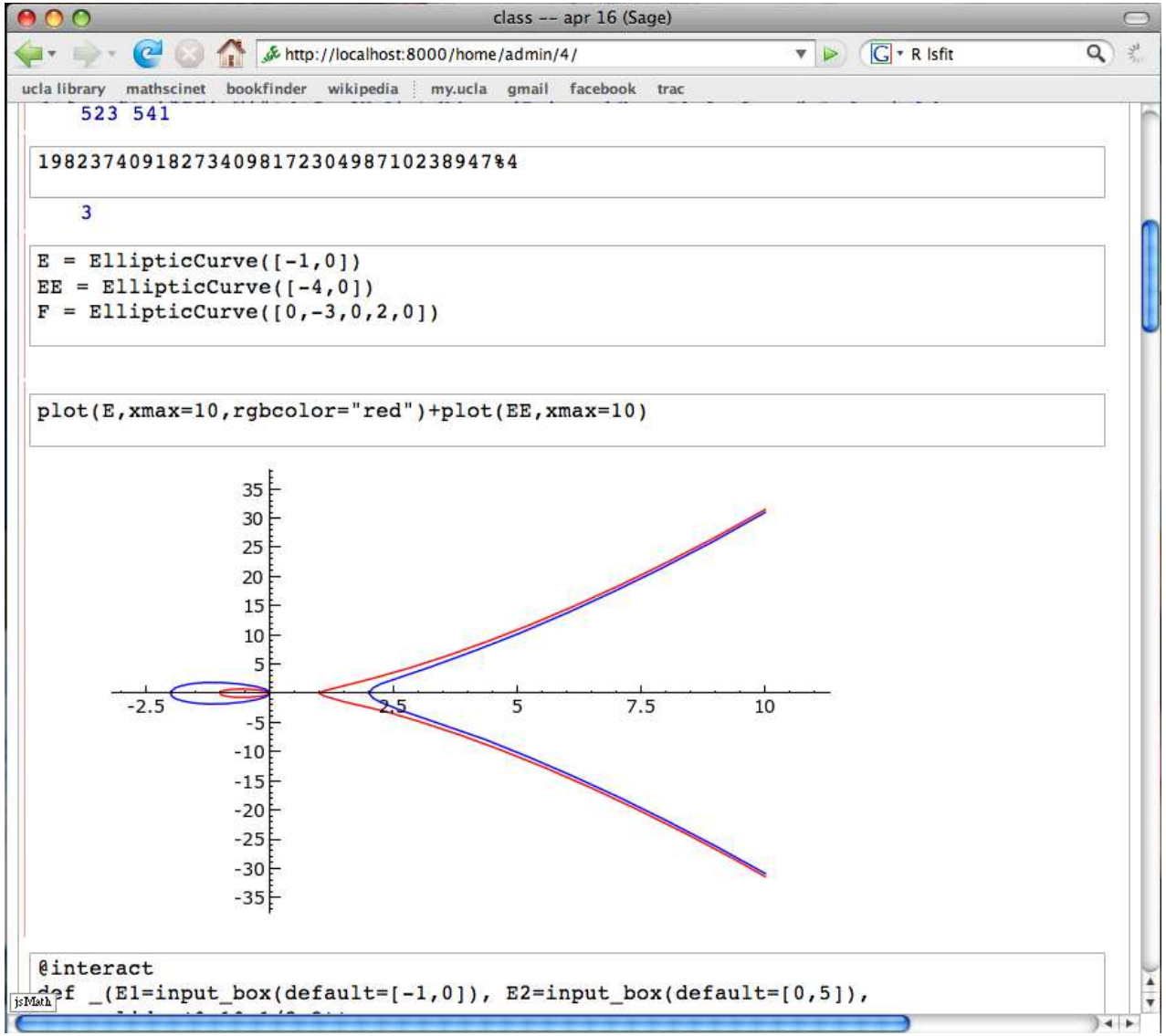
Sage via the Terminal

The first interface to Sage is exactly what everyone would expect, a simple command line interface. Here's a screenshot:



Sage via the web

Sage also has a snazzy web interface:



Some of Sage's Interface features:

- **Notebook**
  One can use the Sage notebook as a front-end for any of the systems that Sage has an interface to. It can also be used to work on your machine remotely with only a web browser, or to share work with others.
- **Tab Completion & Source Introspection**
  Sage has full tab completion and command history, even between sessions. Sage also can use ? and ?? to see documentation and source for any Sage function, right from the command line or Notebook. This helps lower the bar for moving from "user" to "developer."
- **2D and 3D Graphics**
  We have full support for 2D and 3D graphics, both from the command line and in the Notebook. For 2D graphics, we use several tools, especially Matplotlib. For 3D graphics, we have Jmol for interactive 3D, as well as the Tachyon3D ray tracer.

Let's try it!

Demo

Outline

python & Cython

- Cython is a fork of the Pyrex project by Greg Ewing. Cython is a Python-to-C compiler aimed at taking Python code and giving it the speed of pure C.
- Cython also allows you to mix Python with C/C++ code, giving a very fast and smooth interface between your Python code and existing libraries.
- Cython allows one to fully take advantage of the "90/10" rule in the context of Python.

Wind it up and watch it go . . .

Demo

The answer to `Manipulate`

- Interact was first developed by William Stein as the answer to Mathematica's `Manipulate` command. Interact is still not as full-featured, but is amazingly useful, both for teaching and research.
- Interact is surprisingly useful in the following situation: when you have a few choices of input, and you want to repeatedly run a handful of commands in series with those inputs. Interact is basically an abstract tool that makes this incredibly smooth, with no work on the user's part.

Interacting with Interact

Demo

Modern programmers do it in parallel ...

Sage also includes $2\frac{1}{2}$ tools for taking advantage of multiple cores and machines:

- The `@interact` decorator, built using the `pyprocessing` Python extension. This gives a quick and easy way to take advantage of multiple cores.
- **DSage** (Distributed Sage) is a system for distributed computing with Sage. Sadly, DSage has languished since its primary developer (Yi Qiang) graduated from UW. Volunteers?
- IPython now includes an architecture for parallel computing (what was once known as the `ipython1` branch). No one has yet added code to Sage to take advantage of this.

All together now . . .

Demo

Outline

How's our coding? Call 1-800-www.sagemath.org

Sage is a thriving project. We have a new release roughly every three weeks, and an active community of over 100 developers. Almost every recent release has had several first-time contributors to the project. The `sage-devel` mailing list has 675 subscribers, and `sage-support` has 810. We also maintain an active IRC channel: `#sage-devel` on `irc.freenode.net`. Come and visit us!

Where's Antarctica?

Sage developers around the world

This is a map of all contributors to the Sage project. There are currently 111 contributors in 62 different places from all around the world.

Map Zoom: Earth - USA (UW, West, East) - Europe

William Stein Tim Abbott Michael Abshoff Martin Albrecht Nick Alexander Jennifer Balakrishnan Jason Bandlow Francois Bissey Jonathan Bober Tom Boothby Robert Bradshaw Michael Brickenstein Dan Bump Iftikhar Burhanuddin Ondrej Certik Wilson Cheung Craig Citro Francis Clarke Timothy Clemans Alex Clemesha John Cremona Karl-Dieter Crisman Doug Cutrell Didier Deshommes Dan Drake Alexander Dreyer Gabriel Ebner Burcin Erocal Gary Furnish Alex Ghitza Andrzej Giniewicz Chris Gorecki Jason Grout Jon Hanke Carlo Hamalainen Marshall Hampton Mike Hansen Bill Hart David Harvey Neal Holtz Sean Howe Naqi Jaffery Peter Jipsen David Joyner Michael Kallweit Josh Kantor Kiran Kedlaya Emily Kirkman David Kohel Ted Kosan Jason Martin Robert Miller Kate Minola Joel Mohler Bobby Moretti Gregg Musiker Pablo De Nápoli Andrey Novoseltsev Willem Jan Palenstijn John Palmieri Clement Pernet Yi Qiang Dorian Raymer R. Rishikesh David Roe Bjarke Hammersholt Roune Franco Saliola Kyle Schalm Anne Schilling Harald Schilly Jack Schmidt Steven Sivek Jaap Spies Chris Swierczewski Nicolas Thiery Gonzalo Tornaria John Voight Justin Walker Mark Watkins Joe Wetherell Carl Witty Cristian Wuthrich Dal S. Yu Mike Zabrocki Paul Zimmermann

But do **you** trust it?

Before every single release, we:

- build Sage on dozens of different combinations of CPU and operating system (the "build farm"),
- run the doctest suite on **every one of these machines**, currently 78085 doctests,
- ask volunteers on `sage-devel` to do the same, and
- report all issues, and wait to release until these are fixed.

We also keep track of all known bugs on our bug tracker, `http://trac.sagemath.org`.

Sage: Immediate Goals

So here are what **I** think the current focus of development should be, in order:

- doctest timing and regression testing
- the pickle jar
- doctest coverage (currently: `62.9%`)

The first two are already underway, and the doctest coverage should be 100% by ... February?

Sage: Next Big Goals

In the longer term, i.e. in the next year, I think that our biggest goals should be:

- **DOCUMENTATION!!!**
- **DOCUMENTATION!!!**
- **DOCUMENTATION!!!**

In particular, I'm talking about **high level** documentation, and **introductory** documentation. This is already underway, but I think that we need a serious effort from the core Sage developers on this front.

How are you doing this?

When most people hear about Sage, they explain all the reasons that Sage must fail. **Sage is already a success.** Two important factors drive Sage development:

- a huge amount of extremely hard, mostly volunteer work, and
- refusal to acknowledge that Sage is impossible.

We're working hard, but we need **your** help.

Who pays for all this?

Sage has been lucky enough to receive generous funding from all kinds of sources. One of the most important factors in getting funding: **we apply for lots of funding**. Our generous sponsors include all of the following:

THANK YOU UT AUSTIN!

A few words from our sponsors . . .

PIMS

CLAY MATHEMATICS INSTITUTE

Institute for Pure & Applied Mathematics
IPAM
University of California, Los Angeles

MSRI
Mathematical Sciences Research Institute

NSF

UNIVERSITY OF WASHINGTON 1861 LVX SIT

DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA

Microsoft

Google

Upcoming Sage Days

The number of Sage Days conferences has been staggering. Even if we ignore the "underground" Sage Days, here are the numbers for the last few years:

- 2006: 2
- 2007: 4
- 2008: 8

William Stein, Aug 23 2007:

At this rate, soon every day will be a Sage Day.

Given that the average Sage Workshop lasts 5 days, this means that by the 2010-2011 school year, William Stein will have to quit his job at UW and continuously travel between Sage Days conferences.

Here are some upcoming Sage Days:

Sage Days: Austin

Sage Days: San Diego, January 2009

Sage Days: Athens, February 2009


Sacramento
San Jose
Los Angeles
San Diego

Sage Days: Seattle, May 2009

Sage Days: Oklahoma ... unlikely!

Outline

Yeah, about that ...

I'm sure I'm out of time by now. However, William just gave three very nice talks on the state of number theory and modular forms functionality in Sage, at a conference in Bordeaux. You can find his talk here:

`http://www.wstein.org/papers/2008-bordeaux/`

Or, if you want to know if Sage can do something, just come ask me. If I don't know, I'll know who to ask.

Any questions?

Thanks for listening!