Sage as a research tool

http://www.sagemath.org/

John Cremona
University of Warwick

Sage Days 35 / Sage-Flint Days / MIR@W Day
Warwick, 19 December 2011

Introduction

- Who am I? A pure mathematician, working in number theory, with a special interest in "explicit methods" to solve problems in number theory (closely related to, but distinct from, computational number theory).

Introduction

- Who am I? A pure mathematician, working in number theory, with a special interest in "explicit methods" to solve problems in number theory (closely related to, but distinct from, computational number theory).
- What is this talk about? This MIR@W day's theme is "the use of mathematical software in research" and it is part of a five day workshop on Sage. I will talk about some ways in which Sage has can be used as a research tool for mathematicians and people in related areas.

Introduction

- Who am I? A pure mathematician, working in number theory, with a special interest in "explicit methods" to solve problems in number theory (closely related to, but distinct from, computational number theory).
- What is this talk about? This MIR@W day's theme is "the use of mathematical software in research" and it is part of a five day workshop on Sage. I will talk about some ways in which Sage has can be used as a research tool for mathematicians and people in related areas.
- What is Sage? Since there are people here who are not familiar with Sage (yet!) I will start by telling you what it is.

What is Sage

- Over half a million lines of Python and Cython source code, not counting comments and whitespace;
- A distribution of mathematical software (nearly 100 third-party packages); builds from source without dependency (over 5 million lines of code)
- Exact and numerical linear algebra, optimization (numpy, scipy, R, and gsl all included)
- Group theory (includes GAP), number theory (includes Pari, eclib/mwrank), combinatorics, graph theory
- Symbolic calculus (includes Maxima)
- Coding theory, cryptography and cryptanalysis
- 2d and 3d plotting (includes matplotlib, jmol and more)
- Statistics (includes R)
- Overall range of functionality rivals that of Maple, Matlab, and Mathematica and is growing very rapidly
- Sage is huge! (The reference manual is over 7000 pages.)

Where did Sage come from, and who is behind Sage?

- William Stein (U. Washington, Seattle) started Sage at Harvard in January 2005.
- He reckoned that no existing mathematical software (free or commercial) was good enough for his research needs.
- Sage-1.0 released February 2006 at Sage Days 1 (San Diego).
- Sage Days Workshops 1, 2, ..., 35, at many locations, including SD6 in Bristol (2007) and SD35 here and now.
- Over 640 developer accounts on http://trac.sagemath.org/sage_trac
- The last release (4.7.2 on 2011-10-29) had contributions from 100 people (for 19 of whom this was their first contribution).
- supported by UW, NSF, DoD, Microsoft, Google, Sun, etc.

The Sage community

- sage-support@googlegroups.com: 2148 members, several hundred messages per month. Users usually get a response very quickly.
- AskSage http://ask.sagemath.org, modelled on MathOverFlow, is another forum for users.
- sage-devel@googlegroups.com for Sage developers: 1389 members, 500–1000 messages per month. Where most design decisions are made (democratically!).
- 2 IRC channels at irc.freenode.net (#sage-devel for development issues, #sage-support for support questions)
- trac server http://trac.sagemath.org/sage_trac for tracking bugs, suggested enhancements etc. **All new code is peer-reviewed before acceptance**.
- Over 80% of Sage library code functions have both full documentation (from which the reference manual is automatically created) and examples ("doctests") which are tested on many platforms before each release.

Sage worldwide

File Edit View History Bookmarks Tools Help

Sage developers around the world

v4.7.2 (2011-10-29) · Like 1k +1 302 Language

RSS · Blog · Trac · Report Bugs · Wiki · Ask · Feedback · Search:

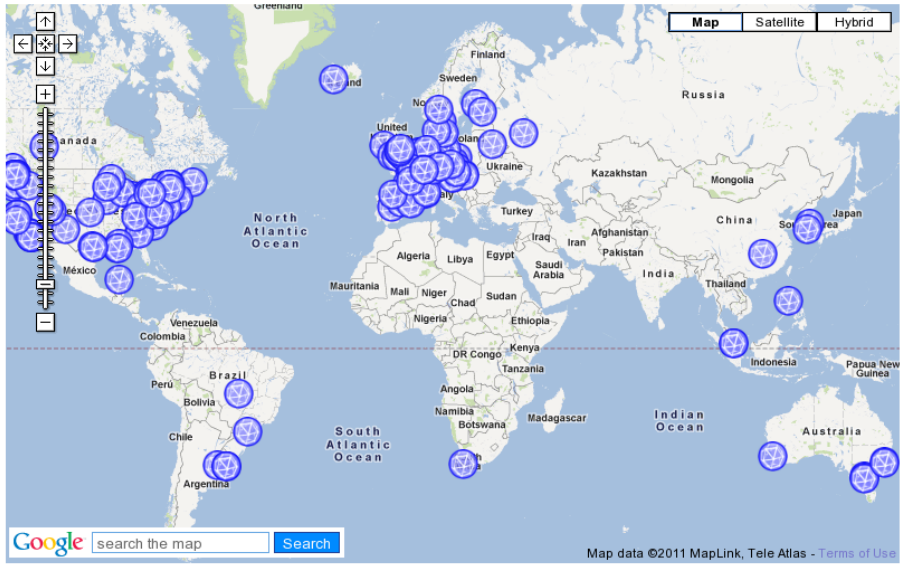
open source mathematics software · Try Online: sagenb / KAIST or Download

Home Tour Support Library Download Development Links

Sage developers around the world

This is a map of all contributors to the Sage project. There are currently 244 contributors in 165 different places from all around the world.

Map Zoom: Earth - USA (UW, West, East) - Europe - Asia - S. America - Australia



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. Volker Braun. 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.

What is Sage for, and who uses it?

- Mission statement: "Creating a viable free open source alternative to Magma, Maple, Mathematica, and Matlab."
- Mathematical research: Sage's founders are researchers in number theory, who wanted a tool they could use, and extend. Sage is now used by many researchers in mathematics (not just number theory: combinatorics, algebra, graph theory, and more).
- Algorithm development: Sage is a good platform for developing new algorithms. Once tested and peer-reviewed, they can (easily) be incorporated into Sage and used by others.
- Teaching: many people have used sage as an adjunct to traditional courses, or as the basis of an entire course, in subject ranging from calculus and linear algebra to cryptography and number theory.

What is in Sage?

- Sage is built out of nearly 100 open-source packages and features a unified interface.
- Sage can be used to study elementary and advanced, pure and applied mathematics.
- This includes a huge range of mathematics, including basic algebra, calculus, elementary to very advanced number theory, cryptography, numerical computation, commutative algebra, group theory, combinatorics, graph theory, exact linear algebra and much more.
- Sage combines various software packages and seamlessly integrates their functionality into a common experience. (Detailed example later).

Components of Sage

- ATLAS: Automatically Tuned Linear Algebra Software
- BLAS: Basic Fortran 77 linear algebra routines
- Bzip2: High-quality data compressor
- Cddlib: Double Description Method of Motzkin
- Common Lisp: Multiparadigm and general-purpose programming language
- CVXOPT: Convex optimization, linear programming, least squares, etc.
- Cython: C-Extensions for Python
- Docutils: an open-source text processing system for processing plaintext documentation into useful formats, such as HTML or LaTeX. It includes reStructuredText, the easy to read, easy to use, what-you-see-is-what-you-get plaintext markup language.
- mwrank: mwrank is a program for computing Mordell-Weil groups of elliptic curves over Q via 2-descent. Since November 2007 mwrank has formed part of the eclib package which is included in Sage.
- F2c: Converts Fortran 77 to C code
- FLINT: Fast Library for Number Theory
- flintqs: William Hart's highly optimized multi-polynomial quadratic sieve for integer factorization
- FpLLL: Euclidean lattice reduction
- FreeType: A Free, High-Quality, and Portable Font Engine
- G95: Open source Fortran 95 compiler
- GAP: Groups, Algorithms, Programming
- GD: Dynamic graphics generation tool
- Genus2reduction: Curve data computation
- Gfan: Grbner fans and tropical varieties
- Givaro: C++ library for arithmetic and algebra
- GMP-ECM: Elliptic Curve Method for Integer Factorization
- GNU TLS: Secure networking
- GSL: Gnu Scientific Library
- Jinja: state of the art, general purpose template engine
- JsMath: JavaScript implementation of LaTeX
- IML: Integer Matrix Library
- IPython: Interactive Python shell

Components of Sage (continued)

- LAPACK: Fortran 77 linear algebra library
- Lcalc: L-functions calculator
- Libgcrypt: General purpose cryptographic library
- Libgpg-error: Common error values for GnuPG components
- libpng: Bitmap image support
- Linbox: C++ linear algebra library
- M4RI: Linear Algebra over GF(2)
- Matplotlib: Python plotting library
- Maxima: computer algebra system
- Mercurial: Revision control system
- MoinMoin Wiki
- MPFI: Multiple Precision Floating-point Interval library
- MPFR: C library for multiple-precision floating-point computations with correct rounding
- MPIR: Multiple Precision Integers and Rationals
- ECLib:Cremona's Programs for Elliptic curves
- NetworkX: Graph theory
- NTL: Number theory C++ library
- Numpy: Numerical linear algebra
- OpenCDK: Open Crypto Development Kit
- OpenOpt: Integrates solvers for numerical optimization into a single common Python-based framework.
- PALP: A Package for Analyzing Lattice Polytopes
- PARI/GP: Number theory calculator
- Pexpect: Pseudo-tty control for Python
- PolyBoRi: Polynomials Over Boolean Rings
- PyCrypto: Python Cryptography Toolkit
- Python: Interpreted language
- Pynac: Symbolic manipulation with Python objects (based on GiNaC)
- Qd: Quad-double/Double-double Computation Package
- R: Statistical Computing

Components of Sage (continued)

- Readline: Line-editing
- Rpy: Python interface to R
- Scipy: Python library for scientific computation
- Singular: fast commutative and noncommutative algebra
- Scons: Software construction tool
- Sphinx: Python Documentation Generator
- SQLAlchemy: The Python SQL Toolkit and Object Relational Mapper
- SQLite: Relation database
- Sympow: L-function calculator
- Symmetrica: Representation theory
- Sympy: Python library for symbolic computation o mpmath: Mpmath is a pure-Python library for multiprecision floating-point arithmetic.
- Tachyon: lightweight 3d ray tracer
- Termcap: Simplifies the process of writing portable text mode applications
- Twisted: Python networking library
- Weave: Tools for including C/C++ code within Python
- Zlib: Data compression library
- ZODB: Object-oriented database

Command line and notebook interfaces

Sage can be used in several ways. First, there is a command-line interface:

```
jec@fermat%sage
----------------------------------------------------------------------
| Sage Version 4.7.2, Release Date: 2011-10-29
| Type notebook() for the GUI, and license() for information.
----------------------------------------------------------------------
sage: 2+2
4
sage: (1+factorial(30)).factor()
31 * 12421 * 82561 * 1080941 * 771906831992755 1
sage: F = FiniteField(next_prime(2^100)); F
Finite Field of size 1267650600228229401496703205653
sage: time EllipticCurve(F,[123,456]).cardinality()
1267650600228229939829009573820
Time: CPU 0.26 s, Wall: 0.28 s
```

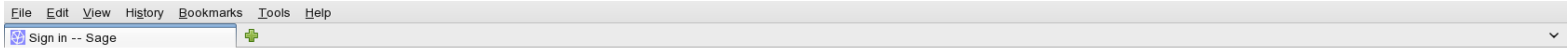
Notebook (via built-in web server)

Sage has a built-in web-server which enables users to connect to it either on a local machine (as on my laptop here) or on a remote server (such as the one at http://www.sagenb.org/).
Departments can run their own Sage servers for their students, or students can download and run their own copy of Sage.

Notebook (via built-in web server)

Sage has a built-in web-server which enables users to connect to it either on a local machine (as on my laptop here) or on a remote server (such as the one at http://www.sagenb.org/). Departments can run their own Sage servers for their students, or students can download and run their own copy of Sage.

(Screenshot on next page, hopefully followed by a demo. The demo is at https://selmer.warwick.ac.uk:8000/home/pub/11, accessible from this campus network only.)



sage The Sage Notebook

Version 4.7.2

Welcome!

Sage is a different approach to mathematics software.

The Sage Notebook

With the Sage Notebook anyone can create, collaborate on, and publish interactive worksheets. In a worksheet, one can write code using Sage, Python, and other software included in Sage.

General and Advanced Pure and Applied Mathematics

Use Sage for studying calculus, elementary to very advanced number theory, cryptography, commutative algebra, group theory, graph theory, numerical and exact linear algebra, and more.

Use an Open Source Alternative

By using Sage you help to support a viable open source alternative to Magma, Maple, Mathematica, and MATLAB. Sage includes many high-quality open source math packages.

Use Most Mathematics Software from Within Sage

Sage makes it easy for you to use most mathematics software together. Sage includes GAP, GP/PARI, Maxima, and Singular, and dozens of other open packages.

Use a Mainstream Programming Language

You work with Sage using the highly regarded scripting language Python. You can write programs that combine serious mathematics with anything else.

Sign into the Sage Notebook v4.7.2

Username

Password

Remember me

Sign in

Sign up for a new Sage Notebook account

Browse published Sage worksheets (no login required)

Forgot password

A case study

Here is an example of how Sage has been useful in my own research.

A case study

Here is an example of how Sage has been useful in my own research.

I have a long-term project to make a database of elliptic curves (if you do not know what these are, it does not matter). It currently has more than 1345219 curves in it.

A case study

Here is an example of how Sage has been useful in my own research.

I have a long-term project to make a database of elliptic curves (if you do not know what these are, it does not matter). It currently has more than 1345219 curves in it.

I do not use Sage to construct these! I have a well-established C++ program to do that, which I run on Warwick's large cluster.

So where does Sage come in?

A case study

Here is an example of how Sage has been useful in my own research.
I have a long-term project to make a database of elliptic curves (if you do not know what these are, it does not matter). It currently has more than 1345219 curves in it.
I do not use Sage to construct these! I have a well-established C++ program to do that, which I run on Warwick's large cluster.
So where does Sage come in?
There is a large amount of processing to be done to the curves once they have been constructed, and this involves using a lot of different pieces of software written in different languages by different people:

- Computing isogenies: uses Sage (python) code written by me and students, now in the Sage library.
- Finding generators: uses (1) my C++ 2-descent code (mwrank); (2) Denis Simon's PARI/GP script; (3) Magma's HeegnerPoint function by Mark Watkins; (4) my C++ code (eclib) to saturate generators.
- Modular degrees: uses a C library (sympow) by Mark Watkins.
- Integral points: uses Sage library

- Computing isogenies: uses Sage (python) code written by me and students, now in the Sage library.
- Finding generators: uses (1) my C++ 2-descent code (mwrank); (2) Denis Simon's PARI/GP script; (3) Magma's HeegnerPoint function by Mark Watkins; (4) my C++ code (eclib) to saturate generators.
- Modular degrees: uses a C library (sympow) by Mark Watkins.
- Integral points: uses Sage library

Before Sage, managing all these was a nightmare of shell scripts and file transfers.

- Computing isogenies: uses Sage (python) code written by me and students, now in the Sage library.
- Finding generators: uses (1) my C++ 2-descent code (mwrank); (2) Denis Simon's PARI/GP script; (3) Magma's HeegnerPoint function by Mark Watkins; (4) my C++ code (eclib) to saturate generators.
- Modular degrees: uses a C library (sympow) by Mark Watkins.
- Integral points: uses Sage library

Before Sage, managing all these was a nightmare of shell scripts and file transfers.
Now a (fairly) simple Sage function does everything, with Python's good string and file handling tools replacing scripts (bash, awk, sed, grep, and so on).

- Computing isogenies: uses Sage (python) code written by me and students, now in the Sage library.
- Finding generators: uses (1) my C++ 2-descent code (mwrank); (2) Denis Simon's PARI/GP script; (3) Magma's HeegnerPoint function by Mark Watkins; (4) my C++ code (eclib) to saturate generators.
- Modular degrees: uses a C library (sympow) by Mark Watkins.
- Integral points: uses Sage library

Before Sage, managing all these was a nightmare of shell scripts and file transfers.
Now a (fairly) simple Sage function does everything, with Python's good string and file handling tools replacing scripts (bash, awk, sed, grep, and so on).
I also use pymongo, the python front-end to put the data into the MongoDB database.

A manifesto: Reproducible Research (and more)

After discussing the content of this talk recently with William Stein, he made a blog from an email message to me which is here: http://sagemath.blogspot.com/2011/12/when-using-sage-to-support-research.html

A manifesto: Reproducible Research (and more)

After discussing the content of this talk recently with William Stein, he made a blog from an email message to me which is here:
http://sagemath.blogspot.com/2011/12/when-using-sage-to-support-research.html
Stein says:

> *"the most important point to make is to strongly encourage people to do the extra work to turn their 'scruffy research code' into a patch that can be peer reviewed and included in Sage. They will have to 100% doctest it, and the quality of their code may improve dramatically as a result. Including code in Sage means that the code will continue to work as Sage is updated. Also, the code is peer reviewed and has to have examples and documentation for every function. That's a* **much higher bar than just reproducible research**."

. . . getting code up to snuff to include in Sage will often also **reveal mistakes** *that will avoid embarrassment later. I'm fixing some issues related to a soon-to-be-done paper right now that I found when doing just this. . .*

This final step of turning snippets of research code into a peer-reviewed contribution to Sage is: (1) a surprisingly **huge amount of very important useful work**, *(2) something that is emphasized as an option for Sage more than with Magma or Mathematica or Pari (say), (3) something whose* **value people have to be sold on, since they get no real extra academic credit for it, at present,** *usually, and journal* **referees often don't care either way** *(I do, but I'm probably in the minority there), and (4) something that a* **lot of research mathematicians do not do**. *As an example of (4), in the last two months I've seen a ton of (separate!) bodies of code which is all sort of secret research code in various Dropbox repos. . .*