SAGE: Software for Algebra and Geometry Experimentation

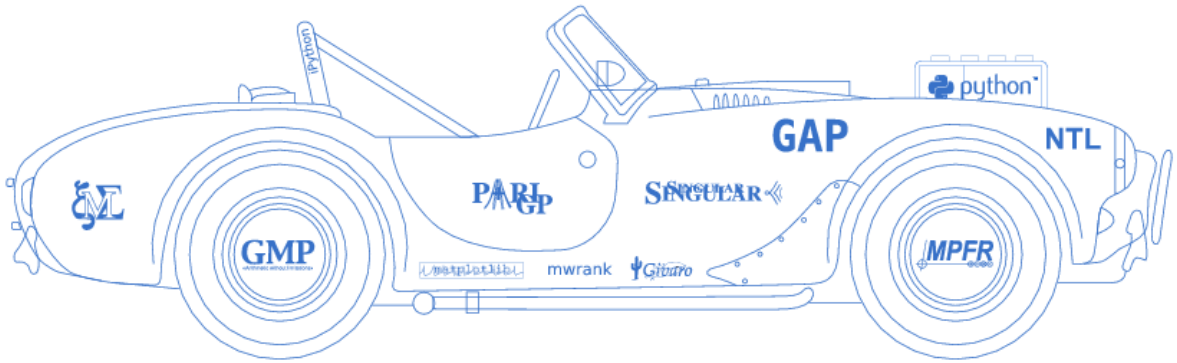
William Stein

October 23, 2006, IMA Algebraic Geometry Software Workshop

http://modular.math.washington.edu/sage

SAGE

Building »The Car«



»Every free computer algebra system I've tried has reinvented many times the wheel without being able to build the car.«

Despite any rules to the contrary, please **do type** and **ask questions** at any time during my talk!

Does Open Source Matter for Math Research?

"You can read Sylow's Theorem and its proof in Huppert's book in the library [...] 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 [...]. You 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?"

– J. Neubüser in **1993** (he started GAP in 1986).

Problem: Create unifying **free open source** mathematics software that is powerful and easy to use.

SAGE addresses this problem.

The development model of SAGE and the technology itself is distinguished by a strong emphasis on **openness, community, cooperation, and collaboration**:

SAGE is building the car, not reinventing the wheel.

Who is Writing SAGE?

I am a **number theorist**. SAGE has a **wide** target area: **algebra, geometry, number theory, algebraic geometry, numerical analysis, statistics, etc.**

Contributors Include: Martin Albrecht, Tom Boothby, Robert Bradshaw, Iftikhar Burhanuddin, Craig Citro, Alex Clemesha, John Cremona, Didier Deshommes, David Harvey, Naqi Jaffery, David Joyner, Josh Kantor, Kiran Kedlaya, David Kirkby, Emily Kirkman, David Kohel, Jon Hanke, Robert Miller, Bobby Moretti, Gregg Musiker, Bill Page, Fernando Perez, Yi Qiang, David Roe, Nathan Ryan, Kyle Schalm, Steven Sivek, Jaap Spies, Gonzalo Tornaria, Justin Walker, Mark Watkins, Joe Weening, Joe Wetherell, ...

- **7 Undergraduates**: have many **extremely interesting** ideas; superb at researching available free software.
- **Many graduate students**: excellent at implementing optimized code and finding fast algorithms.
- **Faculty and computer professionals**: general direction, great writing, and quality control.
- **A snapshot of my inbox...**

SAGE Days 2: Coding Sprints...

Bobby Moretti (UW undergrad), Robert Miller (UW grad), David Harvey (Harvard grad), Joel Mohler (grad), David Joyner (USNA), Bill page (Axiom).

Getting Started with SAGE

1. **Free online** SAGE notebook: `http://sage.math.washington.edu:8100`
2. **Website:** `http://sage.math.washington.edu/sage`
3. **Documentation:** Tutorial, Install Guide, Programming Guide, Reference Manual, Constructions.
4. **Binaries:** For OS X, Windows, and Linux (and building from source is easy).
5. **Mailing lists:** sage-devel (very high volume), sage-announce, sage-forum, sage-support.
6. **Wiki:** like Wikipedea for SAGE.
7. **Trac:** Organizes development.
8. **IRC Chatroom:** `#sage-dev` on irc.freenode.net

SAGE Demo 1: A Groebner Fan

```
sage: R.<x,y,z> = PolynomialRing(QQ,3,order='lex')
sage: I = R.ideal([x^2*y - z, y^2*z - x, z^2*x - y])
sage: I.gro[tab key]
I.groebner_basis  I.groebner_fan
sage: I.groebner_basis()
[-1*z + z^15, -1*z^11 + y, -1*y^2*z + x]

sage: G = I.groebner_fan(); G
Groebner fan of the ideal:
Ideal (y^2*z - x, -1*y + x*z^2, -1*z + x^2*y) of
Polynomial Ring in x, y, z over Rational Field

sage: G.reduced_groebner_bases()
[[-1*z + z^15, -1*z^11 + y, -1*z^9 + x],
 [z^11 - y, -1*z + y*z^4, -1*z^8 + y^2, -1*z^9 + x],
 ...
```

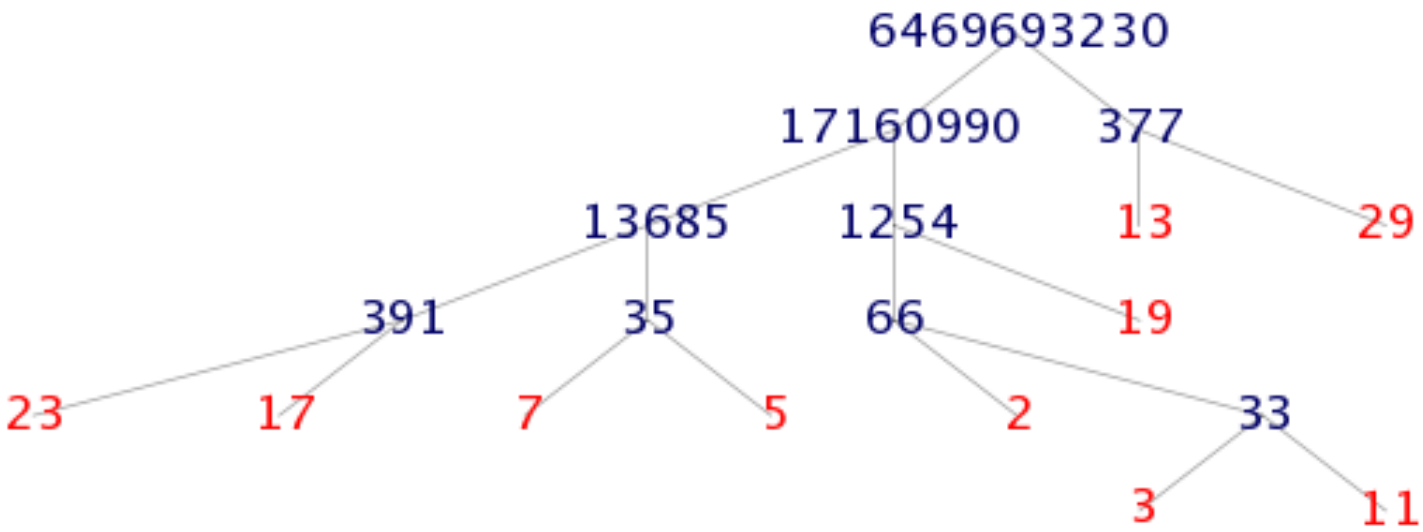
Background: From HECKE 0.1 to SAGE 1.4

- **1997–1999:** HECKE – my free C++ program for modular forms (I wrote an interpreter for it).
- **1999–2004:** I wrote much Magma code. Used it for my research. In computational **arithmetic geometry**, Magma **totally dominates the field**.
- **Feb 2005:** I got job offers with **tenure** – **SAGE 0.1**.
- **Feb 2006: SAGE Days 1** workshop – SAGE 1.0
- **June 2006: High school** workshop – Notebook
- **August 2006: MSRI Grad student** workshop
- **October 2006: SAGE Days 2** workshop.
- **Today:** SAGE 1.4 has a **wide range of functionality**.

SAGE Demo: Education...

(pull up the SAGE notebook)

```
sage: notebook()
----
F = factor_tree(prod(primes(30)))
F.show(xmin=-3,xmax=7,ymin=-5,ymax=0.5,
       figsize=[8,2],axes=False)
...
```



What is SAGE?

SAGE is:

1. **A Distribution** of free open source math software. 64MB source tarball that builds self-contained.
2. **New Readable Code** that fill in gaps in functionality; implement new algorithms.
3. **A Unified Mainstream Interface** to math software: to **Magma**, **Macaulay2**, Singular, **Maple**, MATLAB, Mathematica, Axiom, etc.

SAGE runs on **Linux, OS X, and Windows**.

1. A Distribution

Basic Arithmetic	**GMP, NTL, MPFR, PARI**
Command Line	**IPython**
Commutative algebra	**Singular** (libcf, libfactory)
Database	**ZODB**, Python Pickles
Graphical Interface	**jsmath, SAGE Notebook**
Graphics	**Matplotlib, Tachyon**
Group theory and combinatorics	**GAP**
Graph theory	**Networkx**
Interactive programming language	**Python** (mainstream !!!)
Networking	**Twisted**
Numerical computation	**GSL, Numeric, etc.**
Symbolic computation, calculus	**Maxima**

All core components are **free and open source**. You may **read the code** and **change anything** in SAGE or any of the core libraries it includes, and redistribute the result.

SAGE Demo: A Distribution

```
$ wget http://sage.math.washington.edu/sage/dist/
src/sage-1.4.1.2.tar
$ tar xvf sage-1.4.1.2.tar
...
$ cd sage-1.4.1.2
$ make      # completely automatic on OS X and Linux (!)
...
$ ./sage
-------------------------------------------------------
| SAGE Version 1.4.1.2, Build Date: 2006-10-19        |
| Distributed under the GNU General Public License V2. |
-------------------------------------------------------
sage: install_scripts('/home/was/bin/')
 ... (installs gap, gp, singular, etc. scripts).
$ /home/was/bin/gap
GAP4, Version: 4.4.8 of 18-Sep-2006, i686-apple-darwin8.7.1-gc
gap>
```

2. New Code

Python and Pyrex code — **designed to be readable**:

```
algebras     edu          lfunctions  monoids          sets
categories   ext          libs        plot             structure
coding       functions    matrix      quadratic_forms  tests
combinat     geometry     misc        rings
crypto       groups       modular     schemes
databases    interfaces   modules     server

UNIQUE Source Code Lines (including docstrings):
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
                */*/*.pyx */*/*.pyx |sort |uniq | wc -l
87513
UNIQUE Input Documentation Examples:
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
          */*/*.pyx */*/*.pyx |sort|uniq|grep "sage:" | wc -l
9759
```

SAGE Demo: New Code (interactive help)

```
sage: bernoulli?   # one ? for help
    Return the n-th Bernoulli number, as a rational number.
    INPUT:
        n -- an integer
        algorithm:
            'pari' -- (default) use the PARI C library, which
                      by *far* the fastest.
            'gap'  -- use GAP
            'gp'   -- use PARI/GP interpreter
            'magma' -- use MAGMA
            'python' -- use pure Python implementation

    EXAMPLES:
        sage: bernoulli(12)
        -691/2730
        sage: bernoulli(50)
        4950572052410796482124775 25/66
    ...
    AUTHORS: David Joyner and William Stein
```

SAGE Demo: New Code (interactive code)

```
sage: bernoulli??          # two question marks for source code
File: ... python2.5/site-packages/sage/rings/arith.py
...
    if algorithm == 'pari':
        x = pari(n).bernfrac()     # Use the PARI C library
        return Rational(x)
    elif algorithm == 'gap':
        x = sage.interfaces.gap.gap('Bernoulli(%s)'%n)
        return Rational(x)
    elif algorithm == 'magma':
        x = sage.interfaces.magma.magma('Bernoulli(%s)'%n)
        return Rational(x)
    elif algorithm == 'gp':
        x = sage.interfaces.gp.gp('bernfrac(%s)'%n)
        return Rational(x)
    elif algorithm == 'python':
        return sage.rings.bernoulli.bernoulli_python(n)
    else:
        raise ValueError, "invalid choice of algorithm"
```

3. A Unified Interface

- SAGE **interfaces to**: Axiom, GAP, GP/PARI, Kash, Macaulay2, Magma, Maple, Mathematica, MATLAB, Maxima, Octave, Singular, etc.
- Wide range of **functionality**.
- Unified **command completion and help**.

SAGE Demo: Interfaces

IDEA: Use buffered psuedo-tty's and Python objects that wrap native objects. This makes it possible to wrap **all** computer algebra systems with a command line interfaces using very similar code.

```
sage: R = macaulay2('ZZ/5[x,y,z]')
```

This fires up one copy of Macaulay2 (if it wasn't already started) and sends the line `sage1=ZZ/5[x,y,z]` to M2. It also creates a Python class R with a field set to `"sage1"`.

```
sage: !ps ax |grep M2
 2527  pc  Ss+    0:00.53 M2 --no-debug --no-readline
sage: type(R)
<class 'sage.interfaces.macaulay2.Macaulay2Element'>
sage: R
ZZ/5 [x, y, z]
sage: R.name()
'sage1'
```

Then `n=R.char()` would send `sage2=char(sage1)` to Macaulay2.

The Overall Structure of SAGE

- **Custom package management system** – 42 standard packages, and 28 optional ones. Automated upgrades.
- **Interactive command-line** interface – IPython.
- **Graphical user interface** – via your web browser.
- **Fast underlying arithmetic** – built on mature robust C libraries (GMP, NTL, PARI, GSL). New code is written in Pyrex and Python.
- **Interfaces with other software** use buffered **psuedo-tty**'s.
- **Special purpose components** – e.g., **Gfan** (for computing with Groebner fans and **tropical varieties**), **polymake** for polytopes, **GMP-ECM** (for integer factorization), etc.
- **Mercurial revision control system** – included standard; encourages users to be developers.

The SAGE Notebook: GUI For Mathematics Software

1. The SAGE Notebook – an **"AJAX application"** like Google maps or Gmail.
2. **Written from scratch** by me, Alex C. and Tom B.
3. Uses Python's built-in **BaseHTTPServer** web server (we may soon switch to Twisted for a more robust security model).
4. Works well with Firefox, Safari, and Opera.
5. Client/server model which works **over network** or locally.
6. Current version is **stable and usable**. There is a big idea list that will influence the next version. Wiki!
7. Try it: `http://sage.math.washington.edu:8100`
8. **DEMO: SAGE, Magma, and Macaulay 2 Notebooks**

Goals for SAGE 2.0

All planning is **done in the open**:

`http://sage.math.washington.edu/trac`

Main Goals for SAGE 2.0 (January 2007):

1. Optimize **basic arithmetic**, e.g., finite fields, exact linear algebra, etc.; this involves moving classes from interpreted Python to compiled code.
2. Make the **SAGE Notebook** more wiki like, e.g., easier to edit, better security and robustness.
3. Improve **introspection**: viewing of source code, hyperlinks to files, etc.

SAGE in 2007: Parallelism

1. Support for **parallelism** and use of it for algorithms, e.g., multimodular matrix multiply over $\mathbb{Q}$.

2. I'm co-organizing a **workshop January 29–Feb 2, 2007** at MSRI on parallel computation, which will help get the ball rolling.

Questions?