

# A Brief Introduction to SAGE

Justin C. Walker

February 22, 2006



Power Corrupts; PowerPoint Corrupts Absolutely

# SAGE is . . .

- ▶ Software for **A**lgebra and **G**eometry **E**xperimentation.
- ▶ Free software
- ▶ A language for writing programs
- ▶ A high-level calculator, using your terminology



## SAGE is ...

- ▶ A **distribution** of free open source mathematics software.
  - ▶ <http://modular.ucsd.edu/sage/>
- ▶ A **new computer algebra system** (very structural like Gap and Magma; object-oriented; user extensible; do things right).
- ▶ A **better way** to use all your favorite (**commercial** or free) mathematics software *together*.
- ▶ Why this rather than **Your Favorite CAS?**

## You can use SAGE in several ways:

- ▶ Experiment
- ▶ Accumulate data to inform intuition
- ▶ Prove a theorem!

## A Plug (MSRI/SAGE)

- ▶ Summer Graduate Workshop in Computational Number Theory (MSRI, 7/31-8/11, 2006)
- ▶ Details on the website
  - ▶ [http://www.msri.org/calendar/sgw/WorkshopInfo/392/show\\_sgw](http://www.msri.org/calendar/sgw/WorkshopInfo/392/show_sgw)
- ▶ Check out the book
  - ▶ [http://modular.ucsd.edu/edu/fall05/168/notes/modular\\_forms\\_book](http://modular.ucsd.edu/edu/fall05/168/notes/modular_forms_book)

## SAGE's History

- ▶ William Stein: Seven years of computation
  - ▶ Hecke: Open Source
  - ▶ Magma: Closed Source
- ▶ Jan 2005: Stein and D. Joyner talk at the winter AMS meeting; SAGE is born
- ▶ **One year** of work with many people:

David Kohel, David Joyner, Iftikhar Burhanuddin, John Cremona, Martin Albrecht, Wilson Cheung, Alex Clemesha, Neal Harris, Naqi Jaffery, Kiran Kedlaya, David Kirkby, Jon Hanke, Gregg Musiker, Kyle Schalm, Steven Sivek, Justin Walker, Mark Watkins, Joe Wetherell, Karim Belebas, John Tate, and many others...

# Computer Algebra Systems

## 1. Closed/Commercial(\$)

- ▶ Mathematica

- ▶ <http://www.wolfram.com>

- ▶ Maple

- ▶ <http://www.maplesoft.com>

- ▶ Magma

- ▶ <http://magma.maths.usyd.edu.au/magma/>

- ▶ Matlab

- ▶ <http://www.mathworks.com>

## 2. Closed/Free

- ▶ Kash/Kant - Algebraic Number Theory
  - ▶ <http://www.math.tu-berlin.de/kant/>
- ▶ CoCoA - Commutative Algebra/Algebraic Geometry
  - ▶ <http://cocoa.dima.unige.it>

### 3. Open (Freely available, with source)

- ▶ **GAP** - Group Theory, Discrete Math

- ▶ <http://www.gap-system.org>

- ▶ **Macaulay2** - Commutative Algebra, Algebraic Geometry.

- ▶ First cut at Macaulay2 support (Kiran Kedlaya) is in 1.0.x.

- ▶ <http://www.math.uiuc.edu/Macaulay2/>

- ▶ **Maxima** – Symbolic Manipulation (Macsyma's Child)

- ▶ <http://maxima.sourceforge.net>

### 3. Open (Freely available, with source) Continued...

- ▶ **NTL** - Number Theory library (C++)
  - ▶ <http://www.shoup.net>
- ▶ **Octave** - Numerical computations, Matlab-like
  - ▶ <http://www.octave.org>
- ▶ **Pari/GP** - Number Theory
  - ▶ <http://pari.math.u-bordeaux.fr>
- ▶ **Singular** - Commutative Algebra, Algebraic Geometry
  - ▶ <http://www.singular.uni-kl.de>

## 4. Other programs (Open) - these do one job

- ▶ **mwrnk** - The famous 'mwrnk'; see John Cremona's home page
  - ▶ <http://www.maths.nott.ac.uk/personal/jec/>
- ▶ ec, simon, sea - Elliptic Curves
- ▶ NZMATH - Python-based Number Theory package
  - ▶ <http://tnt.math.metro-u.ac.jp/nzmath/>

## Not Included With SAGE

1. **Gnuplot** – screwy license (e.g., we wanted to change C source so paths not hard coded, but this is not allowed!)
2. **KASH** – closed source (but **FREE** and very powerful)
3. **Magma** – expensive and closed source (**the dominant system** in arithmetic geometry)
4. **Mathematica / Maple** – expensive and closed source

But **using these** from **SAGE** is supported!

```
sage: (-2006).factor()
-1 * 2 * 17 * 59
sage: (-2006).factor(algorithm="kash")
-1 * 2 * 17 * 59
sage: gap(-2006).FactorsInt()
[ -2, 17, 59 ]
sage: pari(-2006).factor()
[-1, 1; 2, 1; 17, 1; 59, 1]
sage: maxima(-2006).factor()
-2*17*59
sage: kash(-2006).Factorization()
[ <2, 1>, <17, 1>, <59, 1> ], extended by: ext1 := -1
sage: magma(-2006).Factorization()
[ <2, 1>, <17, 1>, <59, 1> ]
sage: maple(-2006).ifactor()
-''(2)*''(17)*''(59)
sage: mathematica(-2006).FactorInteger()
{{-1, 1}, {2, 1}, {17, 1}, {59, 1}}
```

## Non-math SAGE Components

1. **IPython**: Wonderful Interactive Shell
2. **Python**: A **Mainstream** Programming Language (many books; numerous excellent tutorials; constantly being improved by hundreds of developers)
3. **Pyrex**: **Compiled** Python-Like Extension Language
4. Saving and Loading Objects (**ZODB** and **cPickle**)

## Saving and Loading Objects

Almost any individual object in [SAGE](#) can easily be loaded and saved in a compressed format, as can sessions. This requires almost **no programmer support**, even for very complicated objects.

```
sage: E = EllipticCurve([1,2,3,4,5])
sage: time v = E.anlist(10^5)
CPU times: user 1.03 s, sys: 0.22 s, total: 1.25 s
Wall time: 1.59
sage: E.save('E')
sage: quit
Exiting SAGE (CPU time 0m1.45s, Wall time 0m25.36s).
was@form:~/sd/talk$ sage
sage: F = load('E')
sage: time v = F.anlist(10^5)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 0.00
```

## Good Documentation is a Goal

- ▶ Language supports it
- ▶ Documentation on the website
  - ▶ <http://modular.ucsd.edu/sage/documentation.html>
  - ▶ Tutorial
  - ▶ Reference Manual (800+ pages)
  - ▶ Programming Guide
  - ▶ Examples

## Help System

1. `function?` gives documentation about function (extracted from source code)
2. `function??` gives the **source code** of function
3. Because Python is so readable, `function??` is incredibly useful and users frequently use it.
4. `help(module or object)` gives man-page like docs
5. **TO DO:** full text search

# TEX-friendly

You can do the following:

- ▶ Embed **SAGE** code in your document
- ▶ Execute this code (semi-)automatically, to produce up-to-date results
- ▶ Gonzalo Tornaria, Joe Wetherell, and others designed and implemented this at the SAGE Days coding sprint

Here's an Example...

# T<sub>E</sub>X'ing with SAGE

- ▶ Create *foo.tex*
- ▶ Get *sagetex.sty*, *sagetex.py*
- ▶ Create your masterpiece:

Now we evaluate the following block:

```
\begin{sageblock}
# do some stuff
E = EllipticCurve("37a")
# more stuff
\end{sageblock}
```

Now the elliptic curve  $E$  given by  $\text{\sage{E}}$  has discriminant  $\text{\sage{E.discriminant()}}$ .

# T<sub>E</sub>X'ing with SAGE (continued)

- ▶ Run “pdflatex foo.tex”
- ▶ Run “sage foo.sage”
- ▶ Run “pdflatex foo.tex” (again, as with, e.g., *bibtex*)
- ▶ Et voila:

Now we evaluate the following block:

```
# do some stuff
E = EllipticCurve("37a")
# more stuff
```

Now the elliptic curve  $E$  given by  $y^2 + y = x^3 - x$  has discriminant 37.

## Some Real-world Examples

- ▶ Some Computations related to the Birch & Swinnerton-Dyer Conjecture
- ▶ Finding Elliptic Curves (searching the Cremona Database)
- ▶ The Congruent Number Problem

These are due to William Stein and others; see the [SAGE](#) documentation website for details.

## Example: From Stein's work on Birch & Swinnerton-Dyer

See Stein et. alia, "Computational Verification of the Birch and Swinnerton-Dyer Conjecture for Individual Elliptic Curves", Math. Comp, ....

```

sage: E = EllipticCurve('225a'); E
---> Elliptic Curve defined by  $y^2 + y = x^3 + 1$  over Rationals
sage: f = E.division_polynomial(5); f
--->  $5*x^{12} + 475*x^9 - 375*x^6 - 3125*x^3 - 625$ 
sage: F = f.factor(); F
---> (5) * (x^4 + 5*x^3 - 10*x - 5) * (x^8 - 5*x^7 + 25*x^6
---> - 20*x^5 + 55*x^4 - 50*x^3 + 100*x^2 - 50*x + 25)
sage: F.unit()
---> 5
sage: h = F[1][0]; h
--->  $x^8 - 5*x^7 + 25*x^6 - 20*x^5 + 55*x^4 - 50*x^3 + 100*x^2 - 50*x + 25$ 
sage: G = h.galois_group(); G
---> Transitive group number 2 of degree 8
sage: G.gens()
---> ((1,2,3,8)(4,5,6,7), (1,5)(2,6)(3,7)(4,8))
sage: G.order()
---> 8

```