

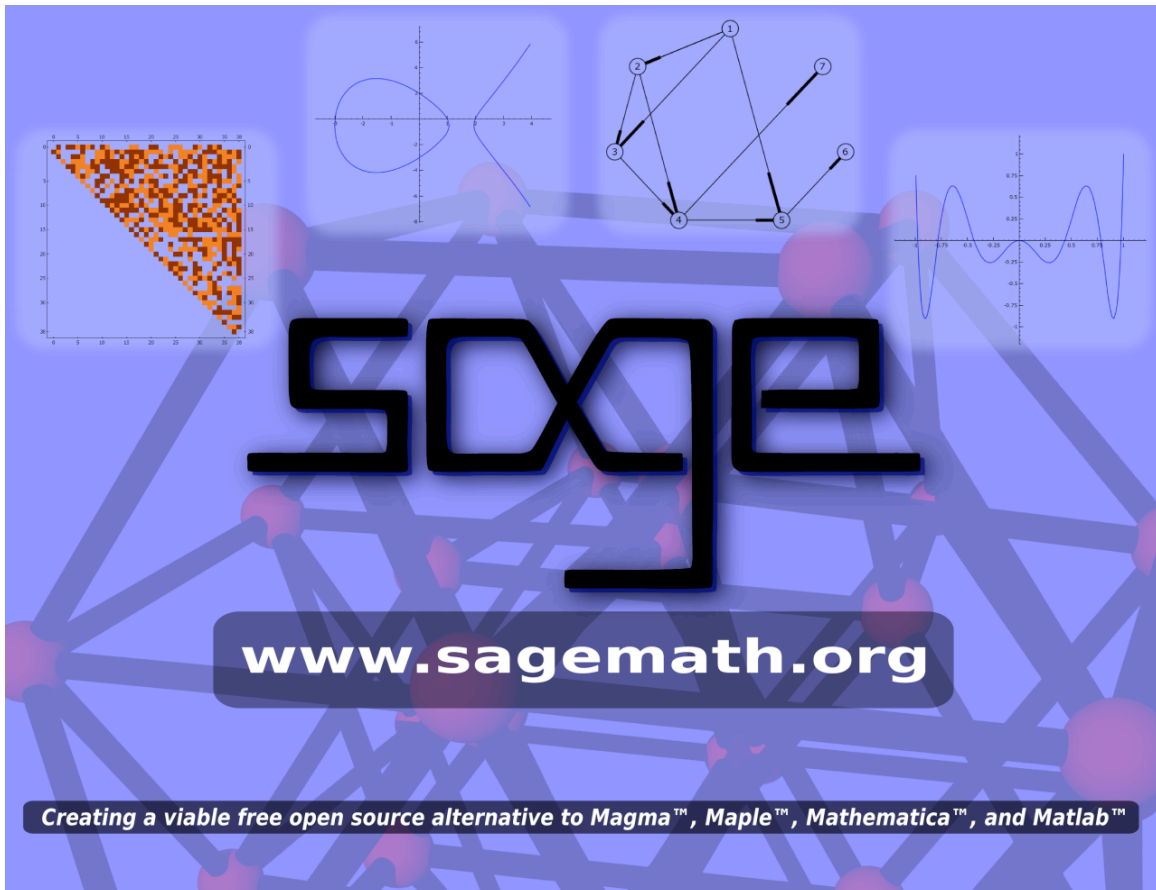
MEGA talk

---

**Sage: Unifying Mathematical Software**

MEGA 2009

*William Stein, Associate Professor, University of Washington*



---

# Part 1: What is Sage?

Part 2: Useful Features of Sage

Part 3: Tour of some functionality you may care about

---

---

## The Sage Project's Goal

Create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab.

---

Firefox <--> Internet Explorer, Opera

Open Office, Latex <--> Microsoft Office

Linux <--> Microsoft Windows

PostgreSQL, MySQL <--> Oracle, Microsoft SQLserver

GIMP <--> Photoshop

---



---

## Motivation: Linus quote

"I think, fundamentally, open source does tend to be more stable software. It's the right way to do things. I compare it to science versus witchcraft. In science, the whole system builds on people looking at other people's results and building on top of them. In witchcraft, somebody had a small secret and guarded it -- but never allowed others to really understand it and build on it.

**Traditional software is like witchcraft.** In history, witchcraft just died out. The same will happen in software. When problems get serious enough, you can't have one person or one company guarding their secrets. You have to have everybody share in knowledge."

-- Linus Torvalds



---

## Motivation: Neubuser quote

"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?"



Neubuser and Huppert

---

---

---

## History

- *I started Sage at Harvard in January 2005.*

- Sage-1.0 released **February 2006** at Sage Days 1 (UC San Diego).
- **19 Sage Days Workshops (!)** at UCLA, UW, Cambridge, Bristol, Austin, France, San Diego, Seattle, MSRI, ..., Barcelona (*next week* at UPC!)
- Sage **won first prize** in Trophees du Libre (November 2007)
- Funding from **Microsoft, Univ of Washington, UC San Diego, NSF, DoD, Google, Sun**, private donations, etc.



---

Part 1: What is Sage?

## Part 2: Useful Features of Sage

Part 3: Tour of some functionality you may care about

---

## Sage Provides a Notebook interface to software you use

(the Sage Notebook in Singular mode)

Primary Decomp - Singular (Sage)

http://localhost:8000/home/admin/185/

Most Visited home gmail arxiv nyt \$\$ sage needs review sagenb mathscinet irc face schedule Catalyst tootleto pandora Pynac

Active Worksheets | Sage Noteb... | Primary Decomp - Singular (Sa... | Primary Decomposition

**SAGE Notebook**  
Version 4.0.1

admin | [Toggle](#) | [Home](#) | [Published](#) | [Log](#) | [Settings](#) | [Report a Problem](#) | [Help](#) | [Sign out](#)

**Primary Decomp - Singular**  
last edited on June 18, 2009 12:13 AM by admin

Save Save & quit Discard & quit

File... Action... Data... singular Typeset Print Worksheet Edit Text Undo Share Publish

```
LIB "primdec.lib";

ring r=0,(x,y,z),dp;
ideal i=y^2*z^2-x^2*y^3-x*z^3+x^3*y*z,y^2*z-x*z^2;
primdecGTZ(i);

// ** redefining r **
ideal i=y^2*z^2-x^2*y^3-x*z^3+x^3*y*z,y^2*z-x*z^2;
[1]:
  [1]:
    _[1]=-y2+xz
  [2]:
    _[1]=-y2+xz
[2]:
  [1]:
    _[1]=z2
    _[2]=y
  [2]:
    _[1]=z
    _[2]=y
[3]:
  [1]:
    _[1]=z
    _[2]=x2
  [2]:
    _[1]=z
    _[2]=x
```

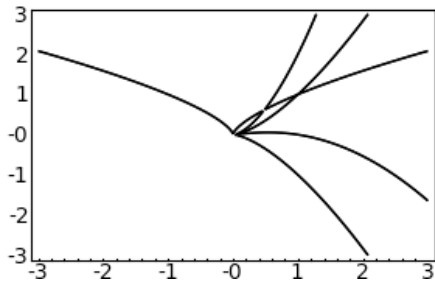
evaluate jsMath

Done

## Sage combines the software you use

Create a highly singular curve in Sage, plot it using matplotlib, then convert it to Singular:

```
R.<x,y> = QQ[]
f1 = (y^2 - x^3)^2 - 4*x^5*y - x^7; f2 = y^2 - x^3; f3 = y^3 - x^2
g = singular(f1*f2*f3); print g
implicit_plot(f1*f2*f3,(x,-3,3),(y,-3,3),plot_points=300).show(figsize=3)
x^10*y^3-x^12-x^9*y^3+4*x^8*y^4-x^7*y^5+x^11-4*x^10*y+x^9*y^2+3*x^6*\
y^5-4*x^5*y^6-3*x^8*y^2+4*x^7*y^3-3*x^3*y^7+3*x^5*y^4+y^9-x^2*y^6
```



Use Singular to compute the the *incidence matrix*  $M$  of the Enriques diagram:

```
singular.load('alexpoly.lib'); M = g.proximitymatrix()[3]; M
```

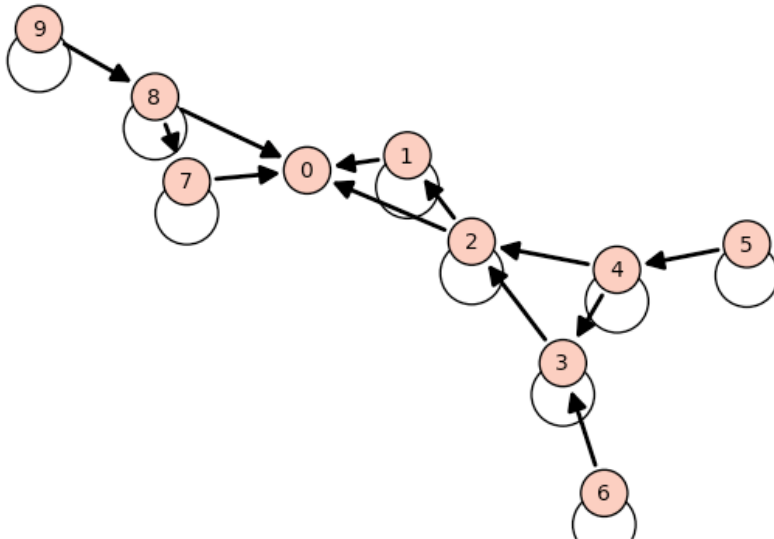
```

0 0 0 0 0 0 0 0 0 0
-1 1 0 0 0 0 0 0 0 0
-1 -1 2 0 0 0 0 0 0 0
0 0 -1 3 0 0 0 0 0 0
0 0 -1 -1 4 0 0 0 0 0
0 0 0 0 -1 5 0 0 0 0
0 0 0 -1 0 0 4 0 0 0
-1 0 0 0 0 0 0 1 0 0
-1 0 0 0 0 0 0 0 -1 2 0
0 0 0 0 0 0 0 0 -1 3

```

Move matrix back to Sage and plot the Enriques diagram:

```
G = DiGraph(M.sage(),incidence_matrix=True); G.plot(scaling_term=0.15)
```




---



---



---

# Sage Includes Extensive and Beautiful Documentation

1. **3600 pages of documentation:** [pdf docs](#) [html docs](#)
2. **Context sensitive help:** introspection and tab completion

To illustrate interactive documentation, we create an ideal.

```
R.<x,y> = QQ[]
I = ideal(x^2-y^2, x^3 + y^3)
show(I)
```

*Unknown control sequence '\Bold'*

Sage objects know all the methods that you can call on them:

```
I. # put cursor after . and press the tab key
```

Syntax Error:

```
I. # put cursor after . and press the tab key
```

```
I.complete_primary_decomposition? # use ? to view help (press tab with cursor
after ?)
```

Syntax Error:

```
I.complete_primary_decomposition? # use ? to view help
```

```
I.complete_primary_decomposition?? # use ?? to view source code (press tab with
cursor after ??)
```

Syntax Error:

```
I.complete_primary_decomposition?? # use ?? to view source
code.
```

---

## Sage's @interact -- for exploring and presenting your work

Put @interact before a function to make the inputs into interactive controls.

```
@interact
def f(n=(2..10), R=[ZZ,QQ,CDF,RIF,Integers(5)]):
    print "A random n x n matrix over R"
    show(random_matrix(R,n))
```

n

R

Integer Ring	Rational Field	Complex Double Field
Real Interval Field with 53 bits of precision	Ring of integers modulo 5	

A random n x n matrix over R



$$\begin{pmatrix} 5 & 20 & -26 \\ -3 & -2 & 5 \\ 1 & -4 & 0 \end{pmatrix}$$

---

## Sage's @interact -- image compression example

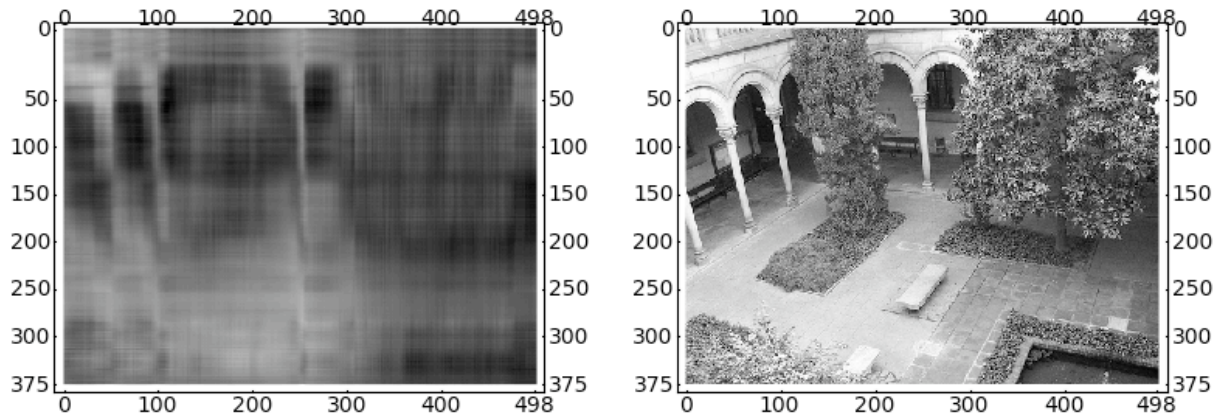
Use singular value decomposition to compress the courtyard outside.

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'ub.png'), 2)
@interact
def svd_image(i = ("Eigenvalues (quality)", (20, (1..100))),
              display_axes = ("Display Axes", True)):
    u,s,v = pylab.linalg.svd(A_image)
    A      = sum(s[j]*pylab.outer(u[0:,j], v[j,0:]) for j in range(i))
    g = graphics_array([matrix_plot(A),matrix_plot(A_image)])
    show(g, axes=display_axes, figsize=(8,3))
    html('<h2>Image compressed using %s eigenvalues</h2>%i')
```

Eigenvalues (quality)

Display Axes

**Image compressed using 4 eigenvalues**



---

## The Sage Notebook -- good for live demos of your code

The Sage notebook is useful for doing live demos of code. You can step through, go back to any previous step, change inputs, annotate with mathematics between cells, etc.

Create  $P = \mathbb{Q}[a, b, c]$  with lexicographic term order.

```
P.<a,b,c> = PolynomialRing(QQ,3, order='lex'); P
Multivariate Polynomial Ring in a, b, c over Rational Field
```

Create the ideal Katsura 3 in  $P$ .

```
I = sage.rings.ideal.Katsura(P,3); I
Ideal (a + 2*b + 2*c - 1, a^2 - a + 2*b^2 + 2*c^2, 2*a*b + 2*b*c -
b) of Multivariate Polynomial Ring in a, b, c over Rational Field
```

Compute a Groebner basis for  $I$  with respect to the lex term order.

```
I.groebner_basis()
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c,
c^4 - 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

Create the quotient ring  $Q = P/I$ .

```
Q.<aa,bb,cc> = P.quotient(I); Q
Quotient of Multivariate Polynomial Ring in a, b, c over Rational
Field by the ideal (a + 2*b + 2*c - 1, a^2 - a + 2*b^2 + 2*c^2,
2*a*b + 2*b*c - b)
```

We do arithmetic in the quotient ring (note that elements are displayed in reduced form):

```
aa - 60*cc^3 + 158/7*cc^2 + 8/7*cc - 1
0
```

---

## Sage has Fast C-Library Interfaces to Singular, PARI, etc.

Many Sage developers (including me, Martin Albrecht, Craig Citro, Carl Witty, Gonzalo Tornaria) spent several months writing highly optimized Python interfaces to Singular and PARI.

To illustrate the Singular interface, we do a simple benchmark of polynomial multiplication in the Singular interpreter and in the Sage (=Python) interpreter.

First we square a polynomial  $10^5$  times in Singular:

```
%singular
int t = timer; ring r = 0, (x,y,z), dp; def f = y^2*z^2-x^2*y^3-x*z^3+x^3*y*z;
int j; def g=f; for (j=1; j <= 10^5; j++ ) { g=f*f; }
(timer-t), system("--ticks-per-sec");
// ** redefining t **
// ** redefining r **
// ** redefining j **
1070 1000
```

Next we create exactly the same polynomial in Sage, which uses libSingular (by Martin Albrecht) to directly in memory create a Singular polynomial.

```
R.<x,y,z> = QQ[]; f = y^2*z^2-x^2*y^3-x*z^3+x^3*y*z; type(f)
<type
'sage.rings.polynomial.multi_polynomial_libsingular.MPolynomial_libs\
ingular'>
```

Then we do the same squaring as above. (The timing in Sage is about 7 times faster! This is because Python is a faster interpreter than Singular's own interpreter.)

```
%time
for j in range(10^5): g = f*f
CPU time: 0.15 s, Wall time: 0.16 s
```

---

## Continuing the above example (poly multiplication)...

**Benchmarking Note:** The Sage interfaces make it easy to keep track of relative speeds of various software. E.g., Sage is twice as fast as Magma at this benchmark (on my OS X laptop).

```
R.<x,y,z> = QQ[]; f = y^2*z^2-x^2*y^3-x*z^3+x^3*y*z
```

```
ff = magma(f)
magma.eval('time for j in [1..10^5] do g := %s*s; end for;'%(ff.name(),ff.name()))
'Time: 0.360'
```

We can also **plot** the zero locus of  $f$ :

```
h=1; implicit_plot3d(f, (x,-h,h), (y,-h,h), (z,-h,h), plot_points=50, opacity=0.7,
color='green')
```

---

## Sage can plot Yoda too, of course (50,000 triangles)

```
from scipy import io
x = io.loadmat(DATA + 'yodapose.mat')
from sage.plot.plot3d.index_face_set import IndexFaceSet
```

```
V = x['V']; F3 = x['F3']-1; F4 = x['F4']-1
Y = (IndexFaceSet(F3, V, color = Color('#00aa00')) +
     IndexFaceSet(F4, V, color = Color('#00aa00')))
Y = Y.rotateX(-1)
Y.show(aspect_ratio = [1,1,1], frame = False, figsize = 4)
```

---

## Sage -- way to get binaries of GAP, Singular, Maxima, Pari, R, etc.,

For OS X and Linux (and almost Solaris). And a virtual machine for Windows right now.

---

```
laptop:~ sage -singular
                SINGULAR                               /  Development
A Computer Algebra System for Polynomial Computations /  version 3-0-4
                by: G.-M. Greuel, G. Pfister, H. Schoenemann 0<
FB Mathematik der Universitaet, D-67653 Kaiserslautern  \  Nov 2007
>
```

---

```
laptop:~ sage -gap

Information at: http://www.gap-system.org
Try '?help' for help. See also '?copyright' and '?authors'

Loading the library. Please be patient, this may take a while.
GAP4, Version: 4.4.10 of 02-Oct-2007, i686-apple-darwin9.7.0-gcc
gap>
```

---

```
laptop:~ sage -maxima
Maxima 5.16.3 http://maxima.sourceforge.net
Using Lisp ECL 9.4.1
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1)
```

---

```
laptop:~ sage -gp

                GP/PARI CALCULATOR Version 2.3.3 (released)
i386 running darwin (ix86/GMP-4.2.1 kernel) 32-bit version
compiled: May  2 2009, gcc-4.0.1 (Apple Inc. build 5488)
                (readline v5.2 enabled, extended help available)
                Copyright (C) 2000-2006 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and comes WITHOUT ANY WARRANTY WHATSOEVER.
Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.
parisize = 4000000, primelimit = 500000
?
```

---

```
laptop:~ sage -R
R version 2.6.1 (2007-11-26)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
...
>
```

---

## Sage -- A Worldwide Community

A lot of people are using and talking about Sage...

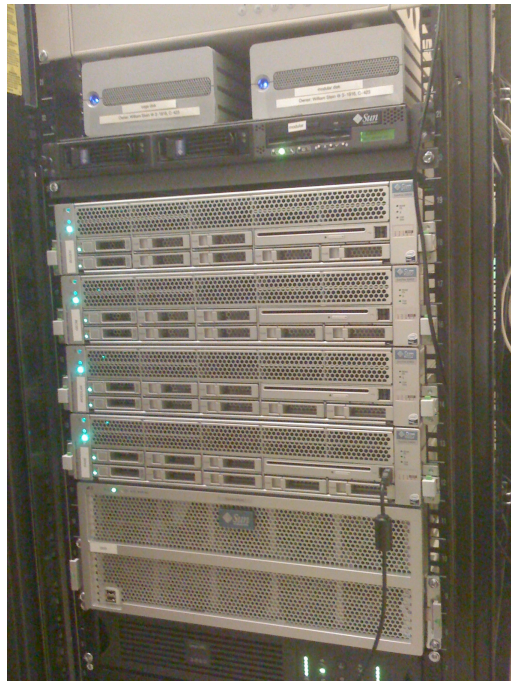
1. **Sage Downloads:** Over 150 downloads of Sage everyday.
2. **Sage Mailing lists:** Over 1,200 subscribers; average of about ***60 messages per day***.

A big plus of Sage is that there is a *lot* of public discussion about everything: <http://groups.google.com/group/sage-support/about>

---

## Sage -- Powerful Development Hardware

(access is a perk of working on Sage)

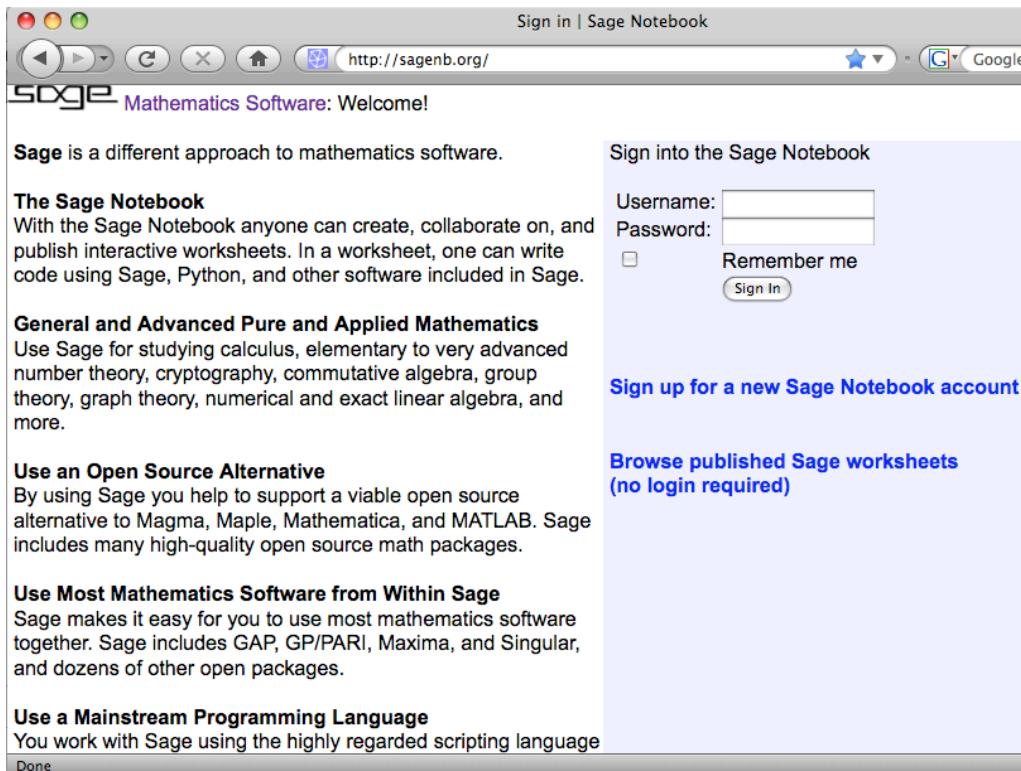


- Four 24-core Sun X4450's with **128GB RAM each**
- one 8-core Sun X4540 with 24TB disk
- one 16-core Sun Sparc T5440

---

## Sage's Free Online Notebook Server

Sign up for free account on <http://sagenb.org> or <http://demo.sagenb.org> in seconds.



The screenshot shows a web browser window with the address bar set to <http://sagenb.org/>. The page title is "Sign in | Sage Notebook". The main content area features the Sage logo and the text "Mathematics Software: Welcome!". Below this, there are several sections of text:

- 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

On the right side of the page, there is a "Sign into the Sage Notebook" section with a login form containing fields for "Username:" and "Password:", a "Remember me" checkbox, and a "Sign In" button. Below the login form are two links: "Sign up for a new Sage Notebook account" and "Browse published Sage worksheets (no login required)".

Active Worksheets | Sage Notebook

http://sagenb.org/home/wstein/

**SAGE Notebook** Version 4.0.1

wstein | Home | Published | Log | Help | Settings | Sign out | Create Shortcut

New Worksheet Upload  Search Worksheets

Archive Delete Stop Current Folder: Active Archived Trash

<input type="checkbox"/>	Active Worksheets	Owner / Collaborators	Last Edited
<input type="checkbox"/>	File Rank 0 curves	wstein <a href="#">Share now</a> (published)	2 days ago by wstein
<input type="checkbox"/>	File Benchmarking	wstein <a href="#">Share now</a>	5 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein <a href="#">Share now</a>	11 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein <a href="#">Share now</a>	23 days ago by wstein
<input type="checkbox"/>	File Calculus in Sage -- a tour (Sage Da ...	wstein <a href="#">Share now</a> (published)	32 days ago by wstein
<input type="checkbox"/>	File Sage Talk	wstein <a href="#">Share now</a>	33 days ago by wstein
<input type="checkbox"/>	File Sage Talk	wstein <a href="#">Share now</a>	33 days ago by wstein
<input type="checkbox"/>	File darmon weight 3	wstein <a href="#">Share now</a> (published)	34 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein <a href="#">Share now</a>	39 days ago by wstein
<input type="checkbox"/>	File publish an object	wstein <a href="#">Share now</a> (published)	47 days ago by wstein
<input type="checkbox"/>	File squares	wstein <a href="#">Share now</a>	51 days ago by wstein

http://sagenb.org/home/wstein/51

Benchmarking (Sage)

http://sagenb.org/home/wstein/50/

**SAGE Notebook** Version 4.0.1

wstein | Toggle | Home | Published | Log | Settings | Report a Problem | Help | Sign out

**Benchmarking**  
last edited on June 12, 2009 03:49 AM by wstein

Save Save & quit Discard & quit

File... Action... Data... sage Typeset [Print](#) Worksheet Edit Text Undo Share Publish

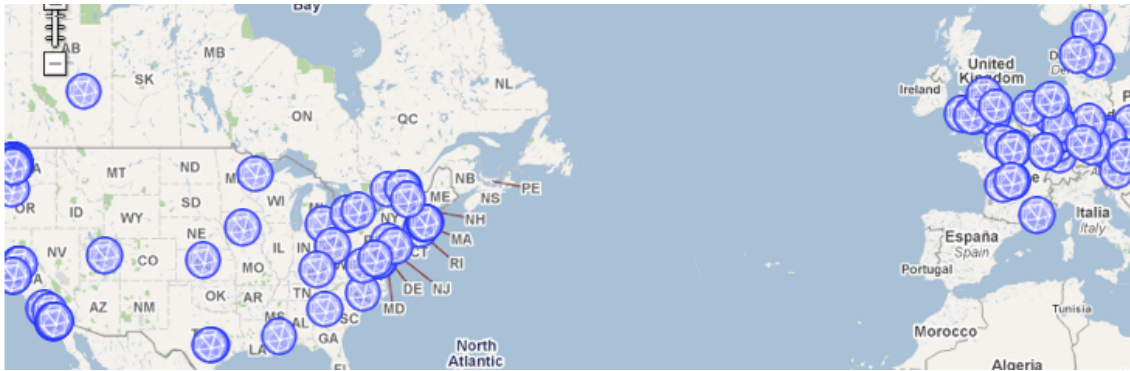
## Benchmarking

### June 2009, William Stein

This talk is about capabilities that both Sage and Magma have, but for which Sage is fast. Here **fast** is defined as follows:  
*faster than Magma on eno.*

---

## Sage -- An Open & International Development Effort



1. Over 150 contributors total -- see [the developer map](#).
2. Copious credit given to every developer contributions in every release
3. New stable release every 2-3 weeks
4. Rotating group of release managers
5. All bugs etc. publicly tracked at <http://trac.sagemath.org>

---

## Sage Uses Python -- A Mainstream Programming Language

1. Sage code is written in Python; Sage = Python + a big Python library
2. Python -- one of top 5 most used programming languages, with millions of users.
3. Python -- Tens of thousands of third party packages are immediately available to you.
4. Sage may be the *first* successful math software system to not invent its own new language just for mathematics.





**"Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code." -- from Python.org**

---

## Sage is Free!

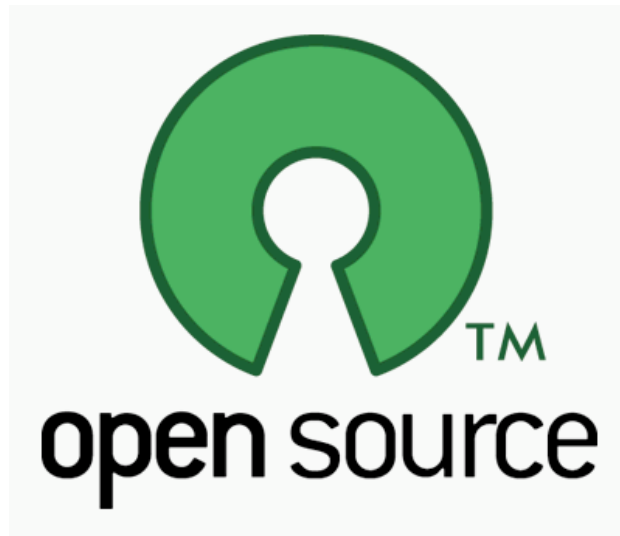
1. Sage is free software.
2. You can legally serve all its functionality over the web (unlike Magma, Maple, Mathematica, and Matlab).
3. You can make unrestricted copies
4. You can run Sage on supercomputers without having to buy expensive licenses

---

## Sage is Open Source

1. Everything in Sage is 100% GPL-compatible (except jsmath, which is Apache licensed and runs in browser).
2. A lot of work has went into "clarifying" licenses on existing math software (tell the Singular/oMalloc story).
3. Sometimes we reimplement major algorithms from the ground up because of license problems (tell the Nauty/NICE story).

4. Sage will always remain free: unlike MuPAD (mention MATLAB story); unlike Maple (mention M. Monogan dinner conversation)
5. Because Sage comes as a complete distribution with dependencies, you can change absolutely anything in Sage or any of its dependencies and definitely rebuild or publicly redistribute the result. This can be very useful for putting tracing code in to understand algorithms.



---

## Sage uses Cython Extensively

1. **Cython** -- Python-to-C compiler **and** way to very efficiently use C/C++ constructions and libraries
2. Over a third of Sage core library written in Cython



To illustrate Cython, we create a function to compute  $\sum_{k=1}^N k$  in both pure Python and Cython. The Cython versions is much faster, because it avoids the overhead of Python object creation, deletion, memory management, etc.

```
def mysum(N):
```

```
s = int(0)
for k in range(1,N): s += k
return s
```

```
time mysum(10^7)
49999995000000L
Time: CPU 2.54 s, Wall: 2.97 s
```

On the next slide we create a Cython version of the above function...

## "Cythonizing" what took > 2 seconds in pure Python...

Using C long long to do arithmetic instead is vastly faster.

```
%cython
def mysum_cython(N):
    cdef int k
    cdef long long s = 0
    for k in range(N): s += k
    return s
```

[Users ws... code sage245 spyx.c](#) [Users ws...de sage245 spyx.html](#)

```
time mysum_cython(10^7)
49999995000000L
Time: CPU 0.01 s, Wall: 0.01 s
```

We can also use MPIR(=GMP) integers. This illustrates how you can directly work with C libraries and C datatypes via Cython.

```
%cython
from sage.rings.integer cimport Integer
def mysum_mpir(Integer N):
    cdef int k
    cdef mpz_t s
    mpz_init(s); mpz_set_si(s,0)
    for k in range(N):
        mpz_add(s, s, N.value)
    cdef Integer ans = Integer()
    mpz_set(ans.value, s)
    return ans
```

[Users ws... code sage280 spyx.c](#) [Users ws...de sage280 spyx.html](#)

```
time mysum_mpir(10^7)
100000000000000L
Time: CPU 0.24 s, Wall: 0.25 s
```

## Some of Sage's Current Development Directions

1. More powerful symbolics manipulation (go way

beyond Maxima)

2. Finish full native port of Sage and all dependencies to Windows
3. Fix bugs
4. Fill in gaps in functionality between Sage and Magma, e.g., fast F4, multivariate factorization, algebraic curves, 3-descent
5. Rewrite the GAP kernel as a library? --> libgap (like "libsingular")

---

---

Part 1: What is Sage?

Part 2: Useful Features of Sage

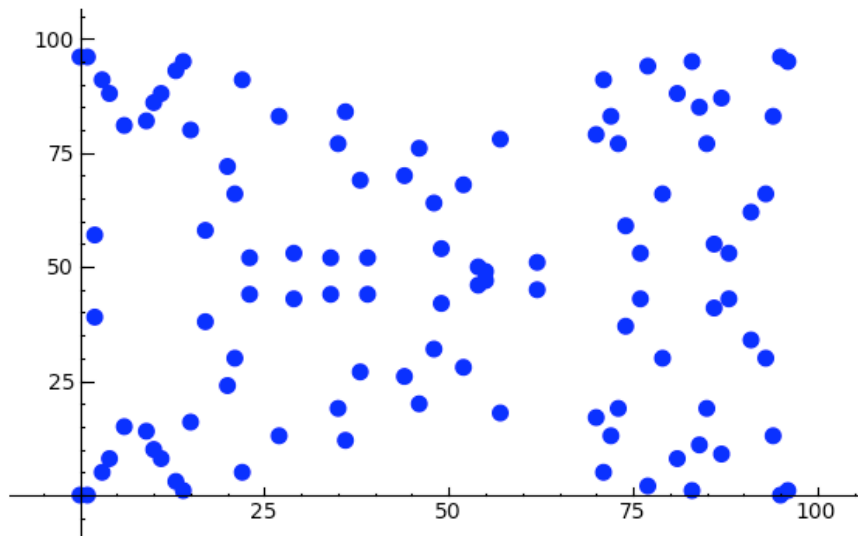
## Part 3: Tour of some functionality you may care about

---

### Elliptic curves

1. All standard algorithms
2.  $p$ -adic L-functions, complex L-functions
3. Heegner points
4. Euler system and Iwasawa-theoretic bounds on Shafarevich-Tate groups
5. Group structure over finite fields
6. Fast point counting modulo  $p$
7. Plotting pictures of elliptic curves

```
plot(EllipticCurve('389a').change_ring(GF(97)),pointsize=50)
```



## Commutative Algebra

1. Clean, structured, object-oriented multivariate polynomial rings and ideals
2. Uses singular as backend when possible for arithmetic speed and certain algorithms
3. Can also use Macaulay2 or Magma for Groebner Basis computations

```
n = 8; P = PolynomialRing(QQ,n,'x'); I = sage.rings.ideal.Katsura(P,n); I
Ideal (x0 + 2*x1 + 2*x2 + 2*x3 + 2*x4 + 2*x5 + 2*x6 + 2*x7 - 1, x0^2
+ 2*x1^2 + 2*x2^2 + 2*x3^2 + 2*x4^2 + 2*x5^2 + 2*x6^2 + 2*x7^2 - x0,
2*x0*x1 + 2*x1*x2 + 2*x2*x3 + 2*x3*x4 + 2*x4*x5 + 2*x5*x6 + 2*x6*x7
- x1, x1^2 + 2*x0*x2 + 2*x1*x3 + 2*x2*x4 + 2*x3*x5 + 2*x4*x6 +
2*x5*x7 - x2, 2*x1*x2 + 2*x0*x3 + 2*x1*x4 + 2*x2*x5 + 2*x3*x6 +
2*x4*x7 - x3, x2^2 + 2*x1*x3 + 2*x0*x4 + 2*x1*x5 + 2*x2*x6 + 2*x3*x7
- x4, 2*x2*x3 + 2*x1*x4 + 2*x0*x5 + 2*x1*x6 + 2*x2*x7 - x5, x3^2 +
2*x2*x4 + 2*x1*x5 + 2*x0*x6 + 2*x1*x7 - x6) of Multivariate
Polynomial Ring in x0, x1, x2, x3, x4, x5, x6, x7 over Rational
Field
```

```
time gb1 = sage.rings.ideal.Katsura(P,n).groebner_basis(algorithm='magma')
Time: CPU 0.82 s, Wall: 1.09 s
```

```
time gb2 = sage.rings.ideal.Katsura(P,n).groebner_basis()
Time: CPU 0.36 s, Wall: 10.27 s
```

```
time gb3 =
sage.rings.ideal.Katsura(P,n).groebner_basis(algorithm='libsingular:slimgb')
Time: CPU 13.00 s, Wall: 13.63 s
```

```
time gb4 = sage.rings.ideal.Katsura(P,n).groebner_basis(algorithm='macaulay2:gb')
```

Time: CPU 0.46 s, Wall: 7.28 s

```
gb1 == gb2
```

```
True
```

```
gb2 == gb3
```

```
True
```

```
gb3 == gb4
```

```
True
```

---

## Algebraic geometry

1. Varieties and Schemes
2. Genus 2 curves and their Jacobians (including fast  $p$ -adic point counting algorithms of Kedlaya and Harvey)
3. Implicit plotting of curves and surfaces

```
P.<x,y,z> = ProjectiveSpace(QQ,2)
X = P.subscheme([x*z^2, y^2*z, x*y^2]); X
```

```
Closed subscheme of Projective Space of dimension 2 over Rational
Field defined by:
  x*z^2
  y^2*z
  x*y^2
```

```
X.dimension()
```

```
0
```

```
X.irreducible_components()
```

```
[
  Closed subscheme of Projective Space of dimension 2 over Rational
  Field defined by:
    z
    y,
  Closed subscheme of Projective Space of dimension 2 over Rational
  Field defined by:
    z
    x,
  Closed subscheme of Projective Space of dimension 2 over Rational
  Field defined by:
    y
    x
]
```

Hyperelliptic curve Frobenius sing Monsky-Washnitzer cohomology:

```
from sage.schemes.hyperelliptic_curves.hypellfrob import hypellfrob
R.<x> = PolynomialRing(ZZ)
f = x^5 + 2*x^2 + x + 1; p = 97
time M = hypellfrob(p, 4, f); M
```

```

Time: CPU 0.03 s, Wall: 0.03 s
[80122582 + O(97^4) 73731349 + O(97^4) 48822670 + O(97^4) 81731002 +
O(97^4)]
[87978030 + O(97^4) 3237569 + O(97^4) 43055445 + O(97^4) 52926365 +
O(97^4)]
[65075166 + O(97^4) 82731009 + O(97^4) 34966498 + O(97^4) 7359568 +
O(97^4)]
[63660518 + O(97^4) 20102765 + O(97^4) 78303210 + O(97^4) 58731896 +
O(97^4)]

```

---



---

## Linear algebra

1. Sparse and dense linear algebra over many rings
2. Highly optimized in many cases
3. In some cases, possibly the fastest money can buy

Computing the determinant of a dense matrix over the integers is fast in Sage:

```

a = random_matrix(ZZ,200,x=-2^128,y=2^128)
time d = a.det()

```

```

Time: CPU 3.22 s, Wall: 3.41 s

```

```

b = magma(a)
magma.eval('time e := Determinant(%s);'%b.name())

```

```

'Time: 12.050'

```

```

d == magma('e')

```

```

True

```

```

a = random_matrix(ZZ,200,x=-2^64,y=2^64)
time h = a.hermite_form()

```

```

Time: CPU 8.12 s, Wall: 8.60 s

```

```

b = magma(a)
magma.eval('time e := HermiteForm(%s);'%b.name())

```

```

'Time: 14.640'

```

```

h == magma('e')

```

```

True

```

---

## Rings

1. **Algebraic rings:** All of the standard rings, such as  $\mathbf{Z}$ ,  $\mathbf{Q}$ , finite fields  $\mathbf{F}_{p^n}$ , and polynomial and power series rings over any other ring in Sage. Substantial code for number fields, and threes models of  $p$ -adic numbers: capped relative, capped absolute, fixed modulus. The algebraic closure of  $\mathbf{Q}$  and its maximal totally real subfield are also implemented, using intervals.
2. **Numerical:** Real and complex numbers of any fixed precision. Double precisions reals and complex (for speed). Rings that model  $\mathbf{R}$  and  $\mathbf{C}$  with intervals (interval arithmetic).

```
@interact
def _(number=(2..20)):
    html('<h2>%s Random Rings</h2>'%number)
    i=0
    for R in sage.rings.tests.random_rings(1):
        print R
        i += 1
        if i > number: break
```

number

## 9 Random Rings

```
Finite Field of size 79639062882915859283
Ring of integers modulo 42888
Univariate Polynomial Ring in x over Integer Ring
Integer Ring
Number Field in a with defining polynomial x^2 + 18284
Ring of integers modulo 22858
Number Field in a with defining polynomial x^2 - 34534
Number Field in a with defining polynomial x^5 - 81*x^4 - 82*x^3 +
50*x^2 - 16*x + 37
Rational Field
Finite Field of size 39841
```

---

## Number fields

1. Absolute, relative, arbitrary towers (built on Pari but offers much more flexibility)
2. Class groups, units, norm equations, maximal orders, reduction mod primes



### 3. Sage and Magma are the only options I know of that have *both* serious algebraic number theory and commutative algebra

```
K.<a> = NumberField(x^3 + 17*x + 3)
R = K.maximal_order(); R
```

```
Maximal Order in Number Field in a with defining polynomial x^3 +
17*x + 3
```

```
U = K.unit_group(); U
```

```
Unit group with structure C2 x Z of Number Field in a with defining
polynomial x^3 + 17*x + 3
```

```
U.gens()
```

```
[-1, 2*a^2 - 11*a - 2]
```

```
K.factor(13)
```

```
(Fractional ideal (13, a^2 - 2*a + 8)) * (Fractional ideal (13, a +
2))
```

```
P = K.factor(13)[0][0]; F = P.residue_field(); F
```

```
Residue field in abar of Fractional ideal (13, a^2 - 2*a + 8)
```

```
F(a^2 + 2/3*a - 5)
```

```
7*abar
```

```
F.lift(-F.0 + 5)
```

```
12*a + 5
```

---

## Algebraic topology

### 1. The Steenrod algebra

### 2. Simplicial complexes and their homology

```
X = simplicial_complexes.SurfaceOfGenus(2); X
```

```
Simplicial complex with 11 vertices and 26 facets
```

```
X.homology()
```

```
{0: 0, 1: Z x Z x Z x Z, 2: Z}
```

```
S = simplicial_complexes.Sphere(1)
```

```
torus = S.product(S); torus
```

```
Simplicial complex with 9 vertices and 18 facets
```

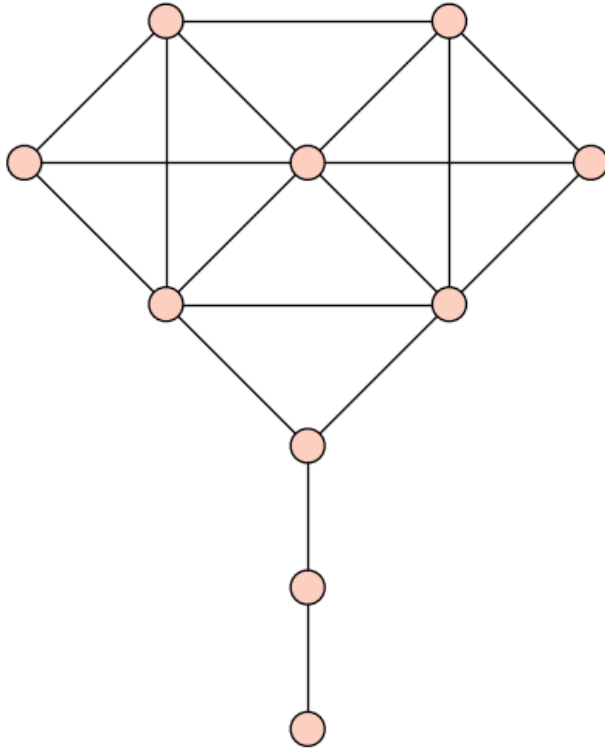
```
torus.homology()
```

```
{0: 0, 1: Z x Z, 2: Z}
```

# Graph theory

1. Sage may overall be the best graph theory software money can buy...

```
g = graphs.KrackhardtKiteGraph(); g.plot(vertex_labels=False)
```



```
g.automorphism_group()
Permutation Group with generators [(1,10)(2,4)(5,6)]
```

---

## Combinatorics

1. **Nicolas Thiery:** Mupad-combinat --> Sage-combinat
2. Symmetric functions, partitions, Lie algebras and root systems, enumeration, crystals, species, etc.

```
time n = number_of_partitions(10^8)
Time: CPU 3.95 s, Wall: 4.16 s
```

---

## Numerical computation

1. Sage also taking on MATLAB
2. Sage includes scipy, numpy, and GSL

We create a random double precision 1000 x 1000 matrix, and quickly do multiplication, and compute SVD and LU decompositions.

```
a = random_matrix(RDF,1000); a  
1000 x 1000 dense matrix over Real Double Field
```

```
time b = a*a  
Time: CPU 0.31 s, Wall: 0.21 s
```

```
time s = a.SVD()  
Time: CPU 3.67 s, Wall: 5.16 s
```

```
time lu = a.LU()  
Time: CPU 0.33 s, Wall: 1.16 s
```

We use scipy.optimize to optimize a function.

```
import scipy.optimize; scipy.optimize.fmin(lambda x: (math.exp(x)-1)-math.cos(x),  
-1.5)  
Optimization terminated successfully.  
Current function value: -1.276615  
Iterations: 17  
Function evaluations: 34  
array([-0.58850098])
```

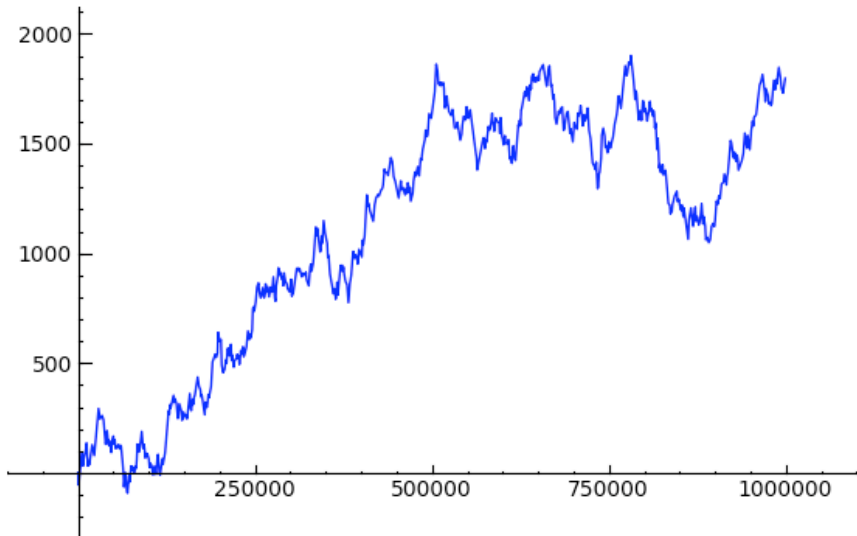
---

## Statistics

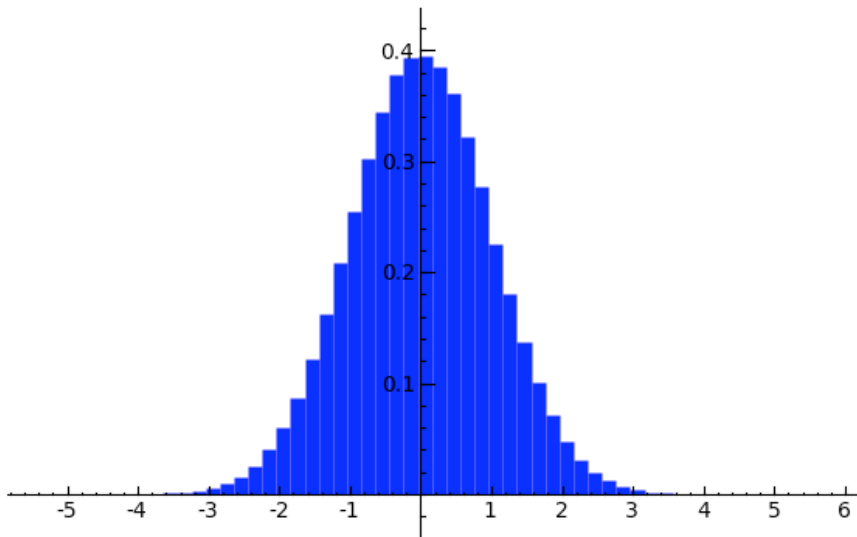
1. Sage includes R and scipy.stats

```
t = finance.TimeSeries(10^6).randomize('normal')
```

```
plot(t.sums())
```



```
t.plot_histogram()
```



---

**Thank You. Questions?**