

JPL09 -- Sage: Unifying Mathematical Software

Sage: Unifying Mathematical Software

JPL 09

William Stein, Associate Professor, University of Washington

The image features a blue background with a network of grey lines and red nodes. Overlaid on this are several mathematical plots: a heatmap in the top left, a coordinate plane with a blue curve in the top center, a graph with 7 numbered nodes in the top right, and a sine wave in the middle right. The word "SAGE" is written in large, bold, black letters with a blue outline in the center. Below it, the URL "www.sagemath.org" is displayed in white text on a dark blue rounded rectangle. At the bottom, a dark blue banner contains the text "Creating a viable free open source alternative to Magma™, Maple™, Mathematica™, and M" in white.

SAGE

www.sagemath.org

Creating a viable free open source alternative to Magma™, Maple™, Mathematica™, and M

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 Primary Goal

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

Firefox <--> Internet Explorer, Opera

Open Office, Latex <--> Microsoft
Office

Linux, OS X <--> Microsoft Windows

PostgreSQL, MySQL <--> Oracle,
Microsoft SQLserver

GIMP <--> Photoshop

Sage <--> Magma, Maple,
Mathematica, Matlab



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



Neubuser and Huppert

colleagues in
Moldava several
years of their salary
for a computer
algebra system?"

- Neubuser who
founded GAP in
1985.

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: Eric S. Raymond quote

"No closed-source developer can match the pool of talent the Linux community can bring to bear on a problem. Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software "hoarding" is



morally wrong
(assuming you
believe the latter,
which neither
Linus nor I do),
but simply
because the
closed-source
world cannot win
an evolutionary
arms race with
open-source
communities that
can put orders of
magnitude more
skilled time into a
problem."

-- Eric S.
Raymond

Brief History of the Sage Project

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

- Sage-1.0 released **February 2006** at Sage Days 1 (UC San Diego).
- **20 Sage Days Workshops (!)** at UCLA, UW, Cambridge, Bristol, Austin, France, San Diego, Seattle, MSRI, ..., Barcelona (*next week* at UPC!), Lopez Island
- Sage **won first prize** in the 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)

The screenshot shows a web browser window titled "Pari in Sage Notebook (Sage)" with the address bar showing "http://localhost:8000/home/admin/191/". The browser's "Most Visited" list includes "home", "gmail", "arxiv", "nyt", "\$\$", "sage", "needs review", "sagenb", "mathscinet", "irc", "face", "schedule", "Catalyst", and "tooldo".

The Sage Notebook interface displays the "Sage Notebook" logo and "Version 4.0.1". The user "admin" is logged in, with navigation links for "Toggle", "Home", "Published", "Log", "Settings", and "Rep". The current session is titled "Pari in Sage Notebook", last edited on June 21, 2009 06:03 PM by admin. A "Save" button is visible in the top right.

The interface includes a toolbar with "File...", "Action...", "Data...", and "gp" dropdown menus, a "Typeset" checkbox, a "Print" icon, and buttons for "Worksheet", "Edit", and "Text".

The main content area shows the following code and output:

```
e = ellinit([1,2,3,4,5])
```

```
[1, 2, 3, 4, 5, 9, 11, 29, 35, -183, -3429, -10351, 6128487/10351,
[-1.618909932267371342378000940, -0.3155450338663143288109995302 -
2.092547096911958607981689447*I, -0.3155450338663143288109995302 +
2.092547096911958607981689447*I]~, 2.780740013766729771063197627,
-1.390370006883364885531598814 + 1.068749776356193066159263547*I,
-1.554824121162190164275074561 + 3.415713103 E-29*I,
0.7774120605810950821375372806 - 1.727349756386839866714149879*I,
2.971915267817909670771647951]
```

```
ellan(e,20)
```

```
[1, 1, 0, -1, -3, 0, -1, -3, -3, -3, -1, 0, 1, -1, 0, -1, 5, -3, 4
```

```
ellglobalred(e)
```

```
[10351, [1, -1, 0, -1], 1]
```

Below the output, there is a blue "evaluate" link and three empty rectangular boxes for further input.

Sage Includes Extensive and Beautiful Documentation

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

To illustrate interactive documentation, we create an elliptic curve.

```
E = EllipticCurve('389a')
```

Sage objects know all the methods that you can call on them (below, put cursor after . and

press the tab key):

```
E. #
  2
```

Use ? to view help (press tab with cursor after ?):

```
E.rank? #
  3
```

Use ?? to view source code (press tab with cursor after ??):

```
E.rank?? #
Syntax Error:
E.rank?? #
```

The Sage Notebook -- live demos of *your* code in *your language*

The Sage notebook is useful for doing live demos of code in almost any language.

Compute a Sylow subgroup using GAP (no license fees!):

```
%gap
s8 := Group( (1,2), (1,2,3,4,5,6,7,8) );
Size(s8);
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
40320
```

```
%gap
G := SylowSubgroup(s8, 2);
Size(G);
Group([ (4,6), (1,5)(4,6), (1,5)(2,8)(4,6), (3,7), (2,3)(7,8),
(1,4)(2,3)(5,6)(7,8), (1,2)(3,4)(5,8)(6,7) ])
128
```

Sage's @interact -- image compression example

Use singular value decomposition to compress the courtyard outside.

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'magic.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)
    html("(Magic square and lovers at Sagrada Familia in
Barcelona.)")
```


Sage's Fast C-Library Interfaces to Singular, PARI, NTL, etc.

Many Sage developers (including me, Martin Albrecht, Craig Citro, Carl Witty, Gonzalo Tornaria) spent several months writing highly optimized Python interfaces to *Singular*, *PARI*, and *NTL*. (GAP may be next on the hit list!)

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 (which takes 1.1 seconds):

```
%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 **
1120 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_lj
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: 0.16 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: 0.370'
```

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

```
f.factor()
(-1) * (-y^2 + x*z) * (-x^2*y + z^2)
```

```
var('x,y,z')
h=1; implicit_plot3d(sin(x)-cos(y)+x*z, (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 (a native Windows port of Sage is in progress).

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

```
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 versior
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
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

Many people use and talk about Sage...

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

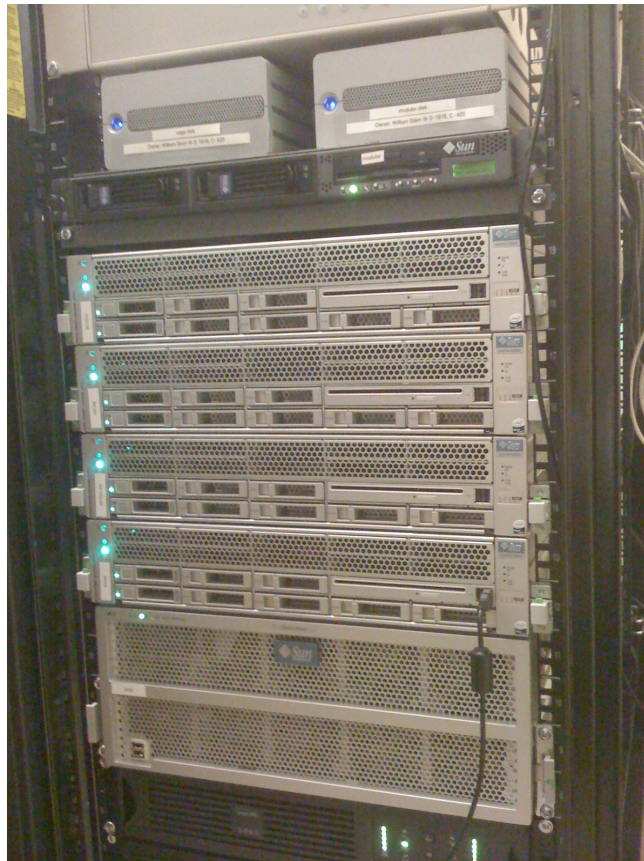
3. **IRC:** #sage-devel on irc.freenode.net (very active channel).

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 (funded by the NSF)

(access is a perk of working on Sage)

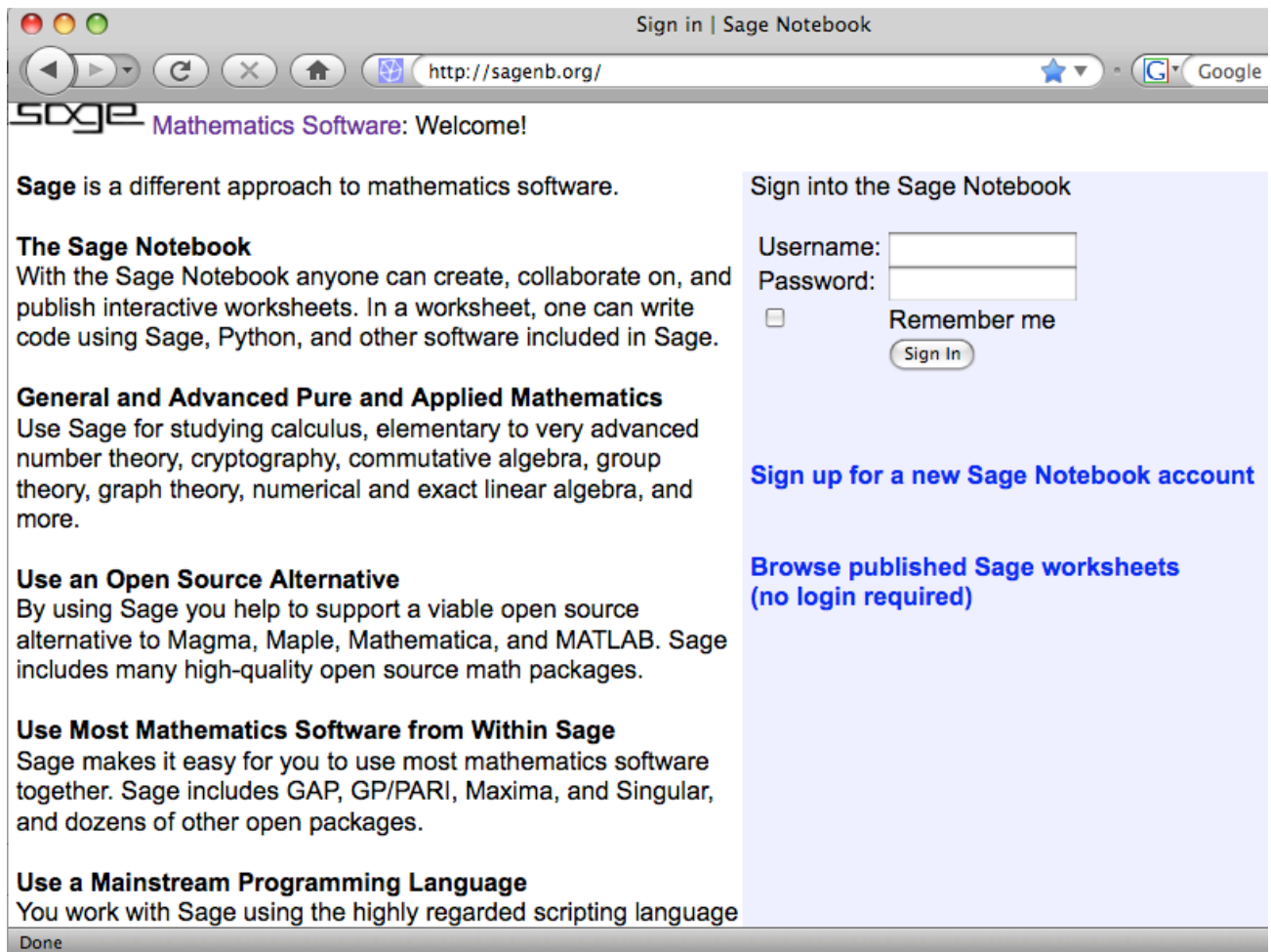


- Four 24-core Sun X4450's with **128GB RAM each**
- one 8-core Sun X4540 with 24TB disk
- one 16-core Sun Sparc T5440



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 displaying "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 describing Sage's capabilities and features. On the right side, there is a light blue sidebar containing a sign-in form and two links: "Sign up for a new Sage Notebook account" and "Browse published Sage worksheets (no login required)".

Sign in | Sage Notebook

Mathematics Software: 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

Sign into the Sage Notebook

Username:

Password:

Remember me

[Sign up for a new Sage Notebook account](#)

[Browse published Sage worksheets \(no login required\)](#)

Done

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 Share now (published)	2 days ago by wstein
<input type="checkbox"/>	File Benchmarking	wstein Share now	5 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein Share now	11 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein Share now	23 days ago by wstein
<input type="checkbox"/>	File Calculus in Sage -- a tour (Sage Da ...	wstein Share now (published)	32 days ago by wstein
<input type="checkbox"/>	File Sage Talk	wstein Share now	33 days ago by wstein
<input type="checkbox"/>	File Sage Talk	wstein Share now	33 days ago by wstein
<input type="checkbox"/>	File darmon weight 3	wstein Share now (published)	34 days ago by wstein
<input type="checkbox"/>	File Untitled	wstein Share now	39 days ago by wstein
<input type="checkbox"/>	File publish an object	wstein Share now (published)	47 days ago by wstein
<input type="checkbox"/>	File squares	wstein Share now	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](#)

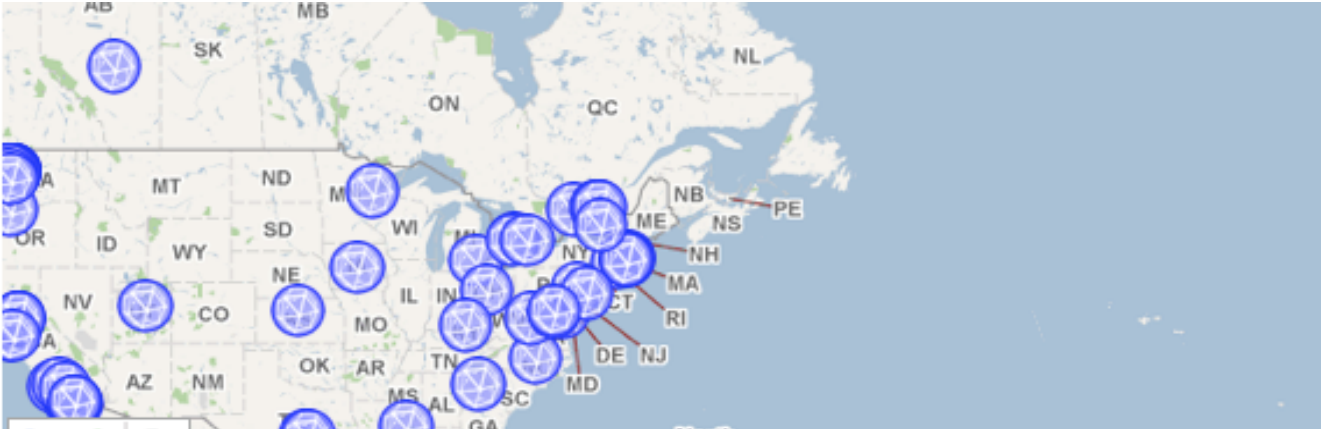
[File...](#) [Action...](#) [Data...](#) [sage](#) Typeset [Print](#) [Worksheet](#) [Edit](#) [Text](#) [Undo](#) [Share](#) [Publish](#)

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's 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

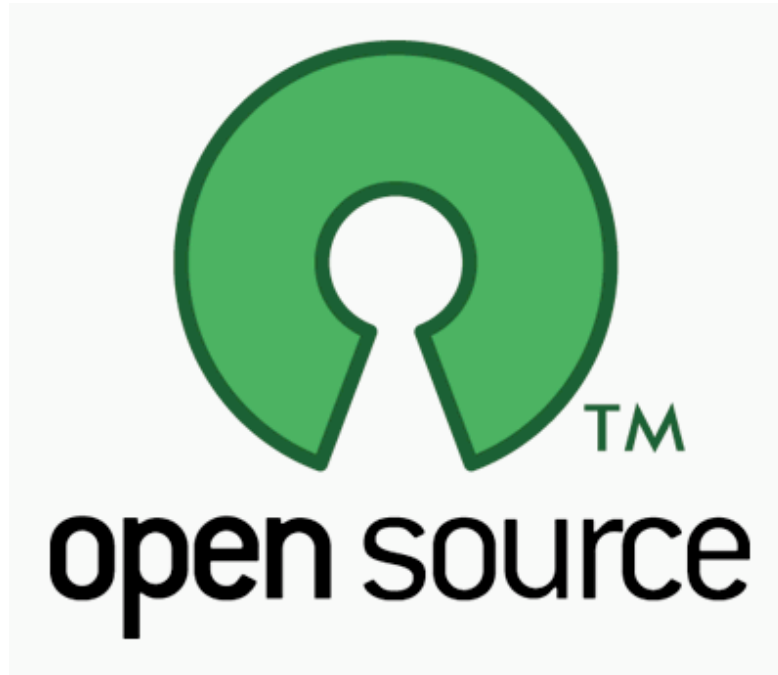
Sage is Free!

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 version 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 2.57 s, Wall: 2.63 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 sage138 spyx.c](#)

[Users ws...de sage138 spyx.l](#)

```
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 sage150 spyx.c](#)

[Users ws...de sage150 spyx.l](#)

```
time mysum_mpir(10^7)
```

```
1000000000000000
```

```
Time: CPU 0.24 s, Wall: 0.24 s
```

Part 1: What is Sage?

Part 2: Useful Features of Sage

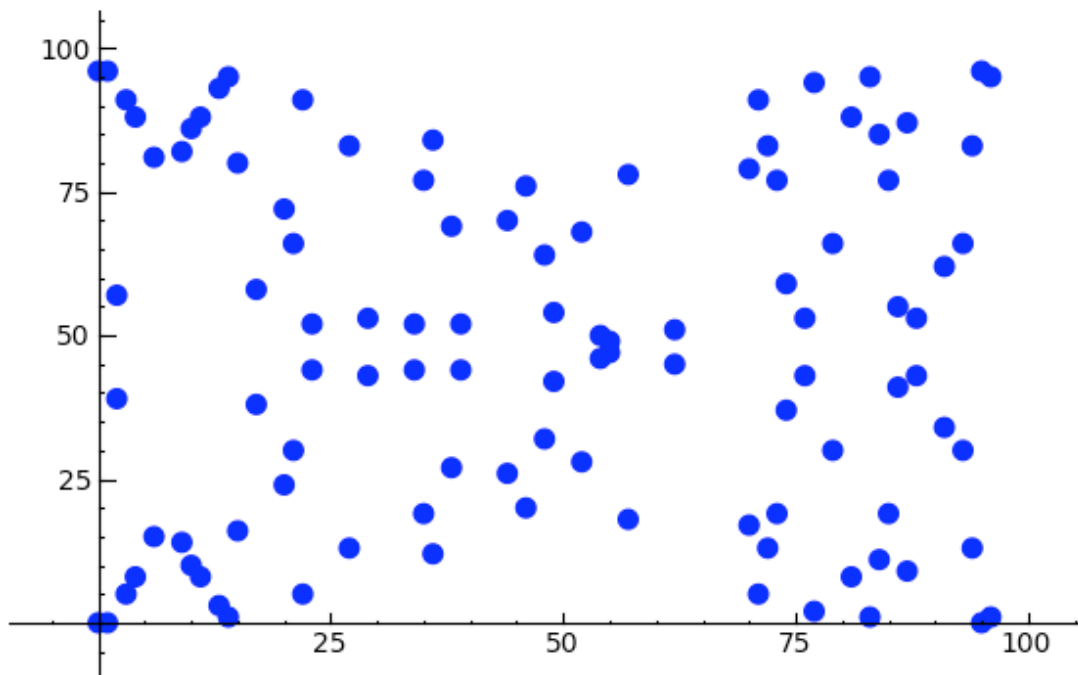
Part 3: Tour of some functionality you may care about

(I will pick 1 or 2 topics based on audience interest)

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*x1^2 + 2*x2^2 + 2*x3^2 + 2*x4^2 + 2*x5^2 + 2*x6^2 + 2*x7^2 - 1,
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*
- 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 0.77 s, Wall: 1.04 s
```

```
time gb2 = sage.rings.ideal.Katsura(P,n).groebner_basis()
Time: CPU 0.29 s, Wall: 9.03 s
```

```
time gb3 =
sage.rings.ideal.Katsura(P,n).groebner_basis(algorithm='libsingular:s
Time: CPU 8.40 s, Wall: 8.87 s
```

```
time gb4 =
sage.rings.ideal.Katsura(P,n).groebner_basis(algorithm='macaulay2:gb'
Time: CPU 0.38 s, Wall: 8.22 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.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 using 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: 0.04 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 3.97 s, Wall: 4.40 s
```

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

```
'Time: 13.990'
```

```
d == magma('e')
```

```
True
```

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

```
Time: CPU 9.59 s, Wall: 10.13 s
```

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

```
'Time: 14.830'
```

```
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 three 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 precision 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 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

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

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

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

Unit group with structure $C2 \times Z$ of Number Field in a with defining polynomial $x^3 + 17x + 3$

```
U.gens()
```

$[-1, 2a^2 - 11a - 2]$

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

$(\text{Fractional ideal } (13, a^2 - 2a + 8)) * (\text{Fractional ideal } (13, a^2))$

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

Residue field in a bar of Fractional ideal $(13, a^2 - 2a + 8)$

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

7abar

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

$12a + 5$

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()
```

$\{0: 0, 1: Z \times Z \times Z \times Z, 2: Z\}$

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



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

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

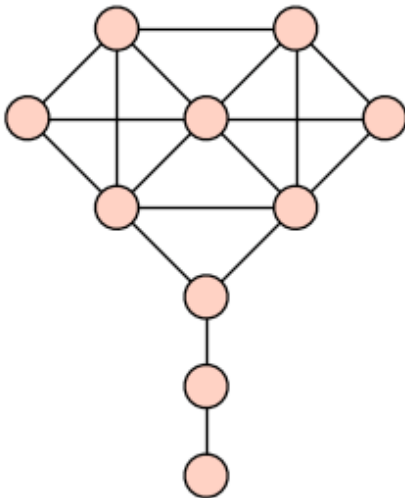
```
torus.homology()
```

```
{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).show(figsize=3)
```



```
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 4.75 s, Wall: 4.95 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 0.50 s, Wall: 0.44 s
```

```
time s = a.SVD()
Time: CPU 5.25 s, Wall: 5.54 s
```

```
time lu = a.LU()
Time: CPU 0.41 s, Wall: 0.39 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
```

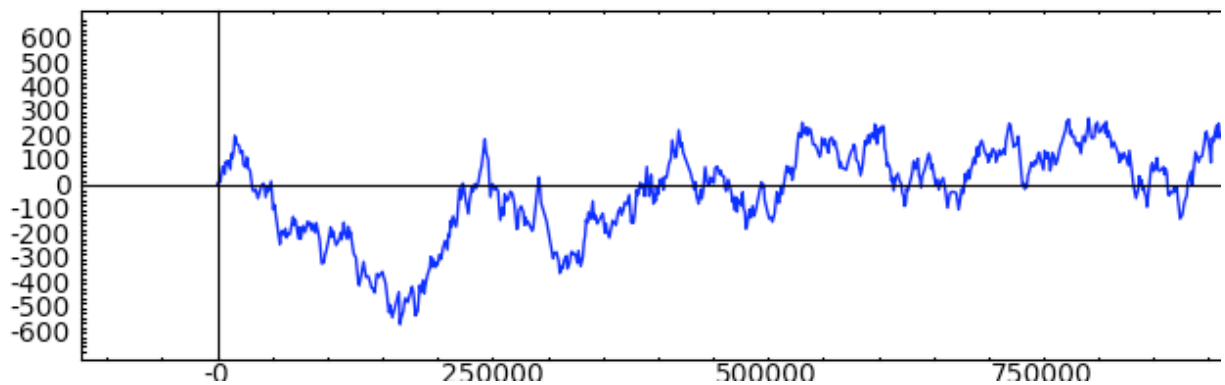
```
array([-0.58850098])
```

Statistics

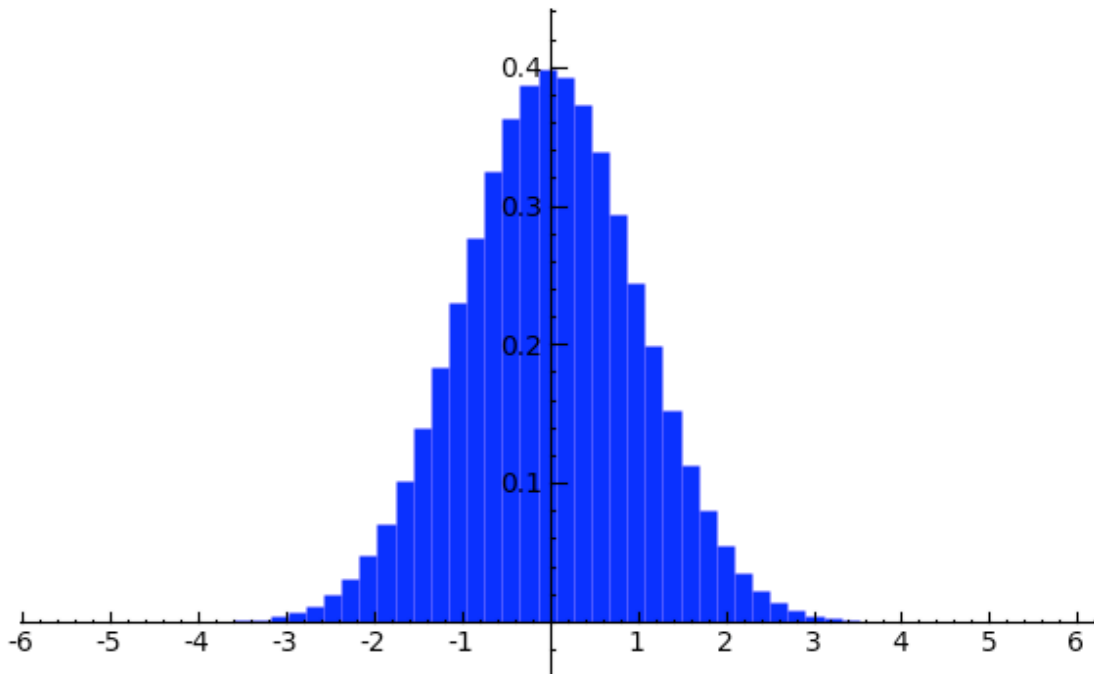
1. Sage includes R and scipy.stats

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

```
plot(t.sums()).show(figsize=[8,2],frame=True)
```



```
t.plot_histogram()
```



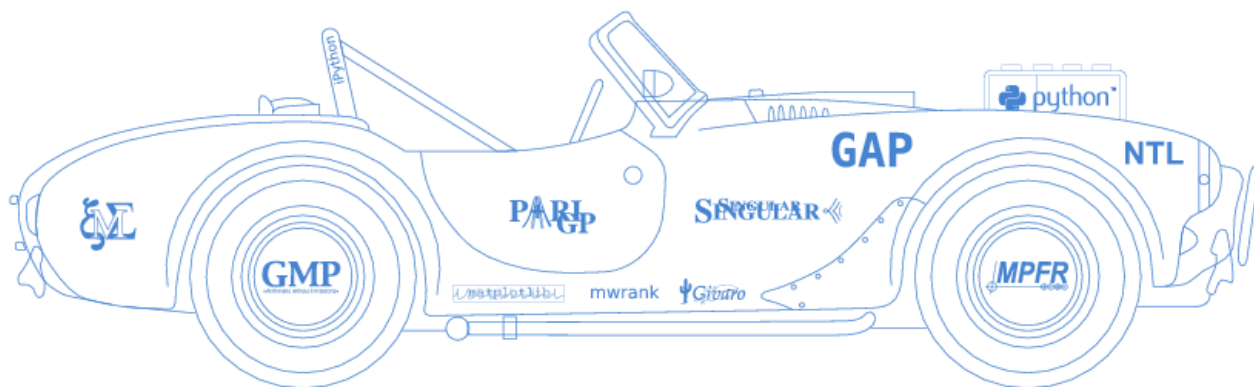
Summary:

Sage is about building the car instead of reinventing the wheel

1. Sage uses a *popular mainstream programming language* instead of inventing a custom mathematics language
2. Use straightforward method to link programs $x^n + y^n$ together -- *C library and pseudotty's*, instead of XML servers/OpenMath. **We** implement all conversion routines, instead of expecting upstream to do it: we ***make them*** communicate with Sage, whether they want to or not. Resistance is futile.
3. *Give copious credit to contributors* and be very developer friendly

(easily build from source).

4. Reuse, improve, and *contribute to existing libraries and projects* (e.g., Singular, Linbox, NTL, Pari, GAP, Maxima), instead of starting over and competing with them.
5. Make the GUI using a web browser: the world of java and javascript plugin is immediately available and Sage *integrates with the web*.



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