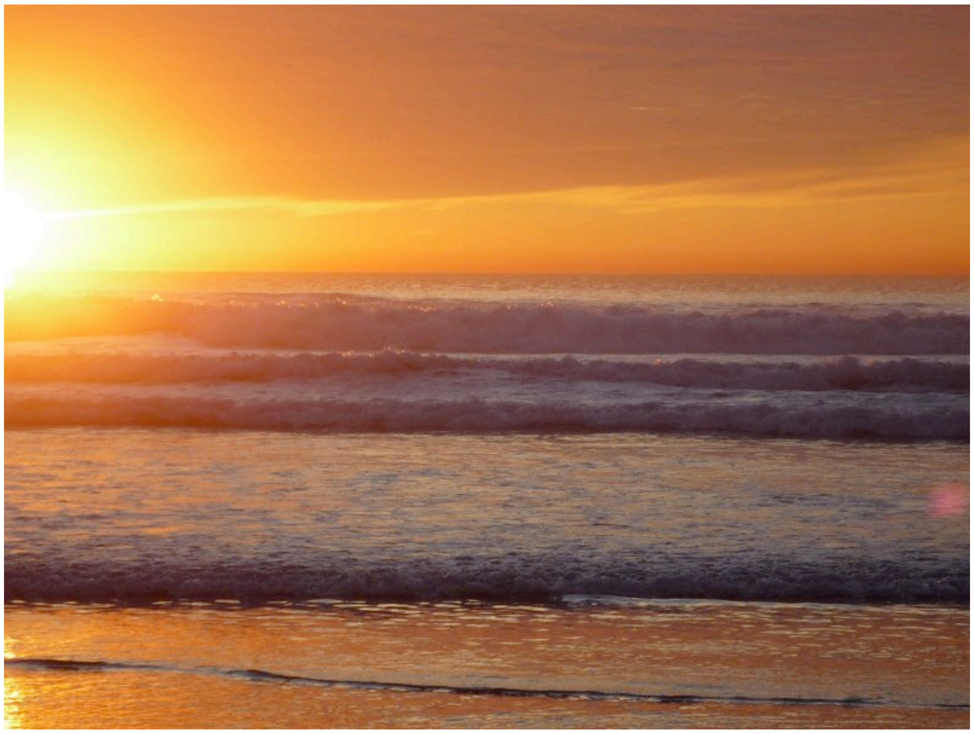
June 11: What is on the Horizon

Sage: What is on the Horizon?

William Stein

June 2009



Outline of Talk

1. **Background**: about Sage
2. **Ports**: to Solaris, Windows, OS X 64-bit; also SPD
3. **The Sage Notebook**: rewrite core to better support embedding in web pages, test framework, better architecture, and LaTeX integration
4. **Symbolic Calculus**: switch to Pynac; implement symbolic algorithms in Pynac/Sage
5. **2d and 3d Graphics:** add more features and support
6. **Statistics**: create a natural, clean, native interface to stats functionality
7. **Arithmetic**: many optimizations across the spectrum
8. **Number Theory:** quaternion algebras, quadratic forms, modular forms, descent on elliptic curves, Heegner points, L-functions
9. **Funding:** status report

Background

about Sage

Mission Statement: *Create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab.*

... Sage is to the Ma*'s somewhat like Linux is to MS Windows or Firefox is to IE or Opera.

- I started the Sage project in January 2005.
- Over 140 contributors since then.
- Structure of Sage: large distribution of math software + large new ***Python*** library to tie it

all together
- About **5000** downloads per month.
- Support mailing list has 1220 subscribers.
- Sage is 100% open source -- you can see or change anything you want.
- All code that goes into Sage's Python library is thoroughly **peer reviewed**.

Quick Demo -- Basic arithmetic

```
2 + 3
```

```
5
```

```
2^3
```

```
8
```

```
2/3
```

```
2/3
```

Magma interface

```
E = magma('EllipticCurve([1..5])'); E
```

```
Elliptic Curve defined by y^2 + x*y + 3*y = x^3 + 2*x^2 + 4*x + 5
over Rational Field
```

```
E.Rank()
```

```
1
```

Same calculation completely in Sage (doesn't use Magma or anything else not in Sage!)

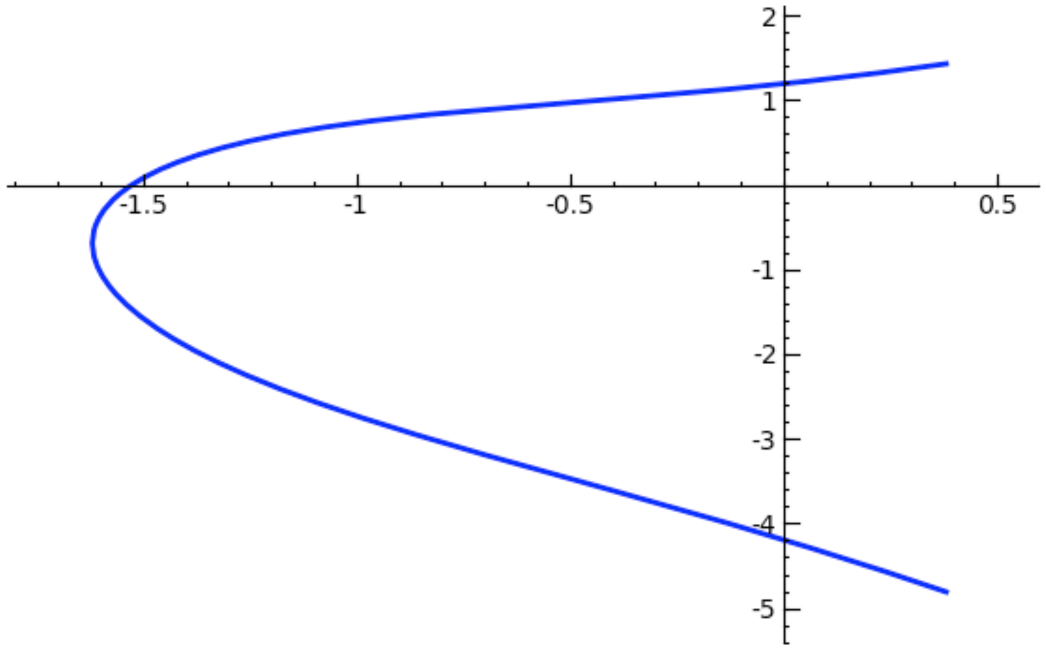
```
E = EllipticCurve([1..5]); E
```

```
Elliptic Curve defined by y^2 + x*y + 3*y = x^3 + 2*x^2 + 4*x + 5
over Rational Field
```

```
E.rank()
```

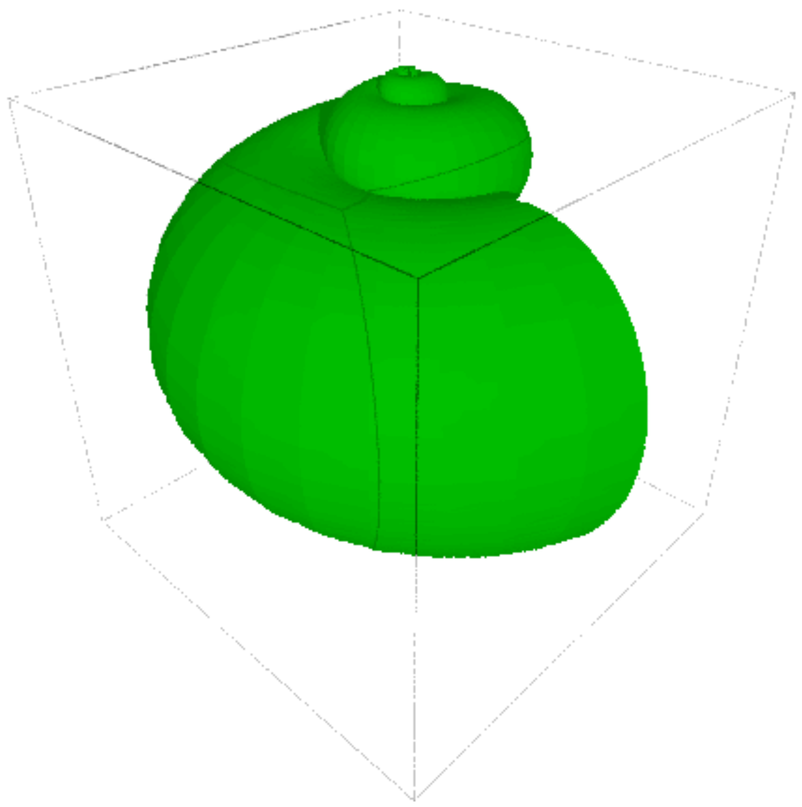
```
1
```

```
plot(E,plot_points=500,thickness=2)
```



A 3d Plot

```
u, v = var('u,v')
k = 1.2; k_2 = 1.2; a = 1.5
f = (k^u*(1+cos(v))*cos(u), k^u*(1+cos(v))*sin(u), k^u*sin(v)-a*k_2^u)
parametric_plot3d(f, (u,0,6*pi), (v,0,2*pi), plot_points=[100,100],
texture=(0,0.5,0), viewer='tachyon')
```



Ports

to Solaris, Windows, OS X 64-bit; also SPD

Solaris: Various people have worked off and on very hard to port Sage to Solaris since **late 2005!** Finally, after years of work (especially by Michael Abshoff), the 32-bit port of Sage to Solaris 10 (both Sparc and x86) is nearly done! This involved everything from replacing CLISP by ECL to fixing bugs all over the place. Sun Microsystems also expressed strong interest in this work, and just donating a high-end sparc machine to the project.

Solaris 64-bit: The port of Sage to 64-bit Solaris has not been attempted yet.

OS X 64-bit: The port of Sage to 64-bit OS X is now also almost done.

Advantages of 64-bit: you can use far more RAM; many basic arithmetic operations are ***over twice as*** fast, which impacts high level algorithms. GMP on OS X 64-bit has much better assembly level optimizations than on 32-bit:

32-bit OS X:

```
sage: time n = factorial(10^6)
CPU times: user 2.02 s, sys: 0.14 s, total: 2.16 s

age: time f = ZZ['x'].random_element(degree=100000)^2
CPU times: user 0.38 s, sys: 0.10 s, total: 0.48 s

sage: time set_random_seed(0); a=random_matrix(ZZ,300).det()
CPU times: user 0.74 s, sys: 0.08 s, total: 0.82 s
```

64-bit OS X (same computer):

```
sage: time n = factorial(10^6)
CPU times: user 0.79 s, sys: 0.10 s, total: 0.89 s

sage: time f = ZZ['x'].random_element(degree=100000)^2
CPU times: user 0.16 s, sys: 0.04 s, total: 0.19 s

sage: time set_random_seed(0); a=random_matrix(ZZ,300).det()
CPU times: user 0.62 s, sys: 0.04 s, total: 0.66 s
```

Microsoft Windows:

- Windows users currently use VMware or http://sagenb.org.
- Overall Sage download stats since Feb 15 (thanks to Harald Schilly):

```
        1.      microsoft_windows  5,661     <--- =vmware, by
*far* the most popular !!
```

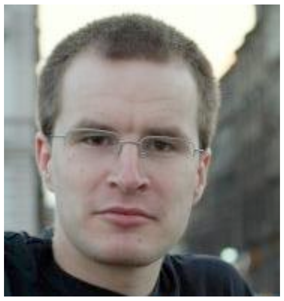
```
         2.      linux/32bit       3,077
         3.      src               1,964
         4.      apple_osx/intel   1,404
         5.      linux/64bit       1,070
         6.      apple_osx/powerpc   372
         7.      linux/atom          122
         8.      apple_osx/ppc        98
         9.      solaris              15
        10.      linux/itanium        14

Total............................. 13797 (or just under
5000/month)
```

I started a **"fully native"** port of Sage to Windows in mid-February 2009. It is a new project and is growing organically ***just like*** Sage-for-UNIX did. See http://windows.sagemath.org and my talk on Tuesday.

- **Currently includes**: bzip2, cython, docutils, freetype, ipython, lapack, libpng, matplotlib, mercurial, m4ri, moin, mpir, networkx, ntl (as WinNTL), numpy, scipy, pycrypto, pyreadline, python, pywin32, scons, setuptools, sphinx (with dependencies: jinja2, pygments), sqlalchemy, sqlite, sympy, twisted, wexpect, zlib, zodb (with dependencies: zope.interface, zope.proxy, zope.testing, zconfig, and zdaemon).
- **MS Visual Studio 2008**: Everything builds from scratch on Windows XP or Windows Vista using MSVC 2008 (and mingw's fortran).
- **Contributors so far**: Dan Shumow, Blair Sutton, and me. Some help from Tom Boothby, Chris Gorecki, William Cauchois.
- Microsoft Research funding this.
- **Next hard step:** get the notebook with preparser ***ported***, which will provide a viable alternative to all the other existing Windows scientific Python distros (Enthought Tool Suite, Python(x,y), etc.).
- **REvolution**: they released a full native (64-bit) version of R for Windows, which might be helpful

SPD = Source Python Distribution



- Project that ***Ondrej Certik*** started that is basically a "lightweight Sage distribution for numerical computation".
- http://code.google.com/p/spdproject/
- Superb way to make the work we've done with Sage available to a wider range of users.
- Will provide "for free" numerous fixes back to Sage.
- I'm excited about this project.

The Sage Notebook

rewrite to better support embedding in web pages, test framework, better architecture, and LaTeX integration

- **Sage lite:** Mike Hansen and I recently split the notebook off from Sage, but we need to get this patch into Sage (ASAP): See trac #5789. It is currently bit rotting.
- **Another major rewrite:** Mike Hansen has been experimenting with rewriting some of the core of the notebook to use a general web framework.
- **Testing:** We need a better automatic testing system.
- **LaTeX**: Users love the LaTeX support provided by the Sage notebook. It just needs more polish (John Palmierri)
- **Embeded interacts**: ability to easily and robustly embed interacts in any random web page. I need this for the modular forms and L-functions database project. (Mike Hansen working on this.)

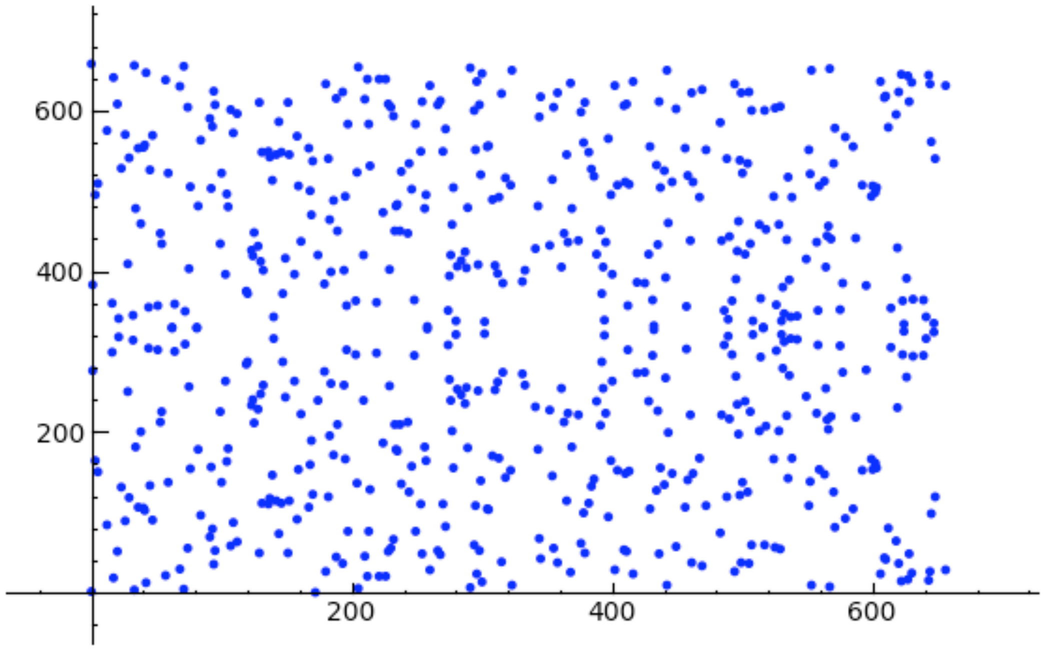
```
@interact
def f(a_invariants=input_box([1..5]), p=slider(prime_range(1000),
default=389)):
   try:
      E = EllipticCurve(GF(p), a_invariants)
   except:
      print "Invalid curve"; return
   show(E)
   print "p = %s"%p
   show(E.change_ring(GF(p)).plot(),xmin=0,ymin=0)
```

a_invariants [1, 2, 3, 4, 5]

p

$$y^2 = x^3 + 389x + 1$$

```
p = 659
```



Symbolic Calculus

switched to Pynac; implement symbolic algorithms in Pynac/Sage

In any technical field, ***not understanding and controlling*** ones core tools has the danger of leading to *second-rate* results. That is a concern with Sage, due to wrapping Maxima.

1. The **pure Python/Maxima hybrid implementation** in Sage < 4.0 was too slow and inflexible.
2. **Burcin** suggested somehow using Ginac, and with a few weeks very hard work, I found a way to completely strip Ginac's CLN (all basic arithmetic types) dependency and replace it by Python objects.
3. **Ginac** is a mature, healthy, lively project, with regular releases, a native Windows version, etc. The code is easy to read and adapt. It is a solid foundation on which to build.
4. **Burcin, me, Mike Hansen, Robert Bradshaw, and Nick Alexander** -- all worked to switch Sage to use Pynac. We finished this for Sage-4.0.

ginac
is not a CAS

(but it is a fine library on which to build a CAS)

```
var('x,theta')
f = sin(x)*cos(2*x) + log(x) - theta*x^3
f
```

```
    -theta*x^3 + sin(x)*cos(2*x) + log(x)
```

```
show(f.taylor(x,1,2))
```

$$-\frac{1}{2}(x-1)^2(5\sin(1)\cos(2)+4\sin(2)\cos(1)+6\,\theta+1)-(x-1)(2\sin(1)\sin(2)-\cos(1)\cos($$

```
show(f.integrate(x))
```

$$-\frac{1}{4}\theta x^4+x\ln(x)-x-\frac{1}{6}\cos(3\,x)+\frac{1}{2}\cos(x)$$

```
expand(f^3)
```

```
    -theta^3*x^9 + 3*theta^2*x^6*sin(x)*cos(2*x) + 3*theta^2*x^6*log(x)
    - 3*theta*x^3*sin(x)^2*cos(2*x)^2 -
```

```
6*theta*x^3*log(x)*sin(x)*cos(2*x) - 3*theta*x^3*log(x)^2 +
sin(x)^3*cos(2*x)^3 + 3*log(x)*sin(x)^2*cos(2*x)^2 +
3*log(x)^2*sin(x)*cos(2*x) + log(x)^3
```

Where To?

- **Low level symbolic arithmetic**: All basic symbolic manipulation will switch to using the ***Pynac C++ library*** (=Ginac with Python data types) as soon as possible. This is potentially ***dramatically*** (100s of times) faster than what we get from Maxima, and is far easier for us to improve and extend.
- **Reduce (or eliminate) the Maxima depedency:** All other symbolic calculus functionality will be implemented via Cython/C++/Sympy/Sage with the eventual goal to *completely eliminate the dependency on Maxima*: In each case, the goal will be to rethink the implementations using modern ideas, tools, libraries, etc. This will lead to interesting new results, speedups, etc.:
    - Equality testing (using interval arithmetic -- mostly done by Robert Bradshaw)
    - Solving equations for a variable
    - Taylor expansion (sympy does a lot already)
    - Symbolic integration (implement numerous heuristics, plus a heuristic Riesch)
    - Factorization (via Singular, etc.)
    - Formal infinite sums (Burcin's Ph.d. thesis)

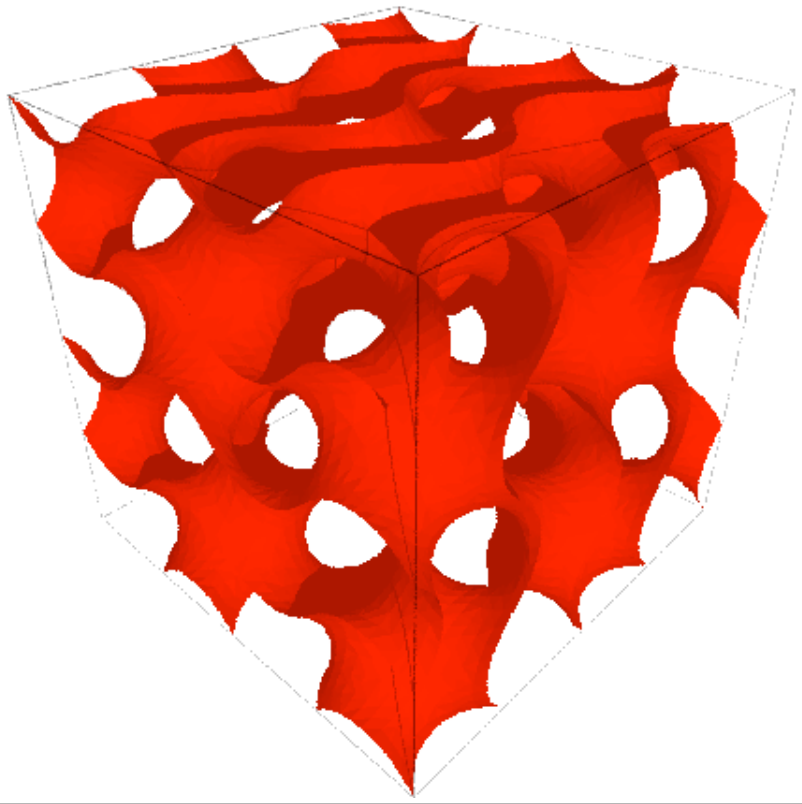
2d and 3d Graphics

add more features and support

3d Plotting

- **implicit_plot3d:** Carl Witty and Bill Cauchois recently implemented this, thus filling in a glaring gap in Sage's 3d plotting functionality (compared to Maple and Mathematica).
- **doctests:** Bill Cauchois greatly improved doctest coverage, though much still remains to do here
- **Mathematica's** 3d plotting still has numerous options than Sage's doesn't -- this needs to be fixed via hard work
- **Jmol --** isn't as rock solid as we wish it were (Bill Cauchois is doing a summer project to write a new java applet for doing 3d software rendering).
- **Resize** -- given jQuery, being able to drag and resize a 3d plot would be easy to implement, but nobody has got around to it.
- **Tick marks** -- currently no control over axes labels or tick marks in 3d plots
- **Interaction** -- make it easy for people to ***manipulate*** elements of 3d plots, select points, etc., and have this interact with Sage.

```
var('x,y,z')
implicit_plot3d(cos(x)*sin(y) + cos(y)*sin(z) + cos(z)*sin(x),
                (x,-2*pi,2*pi),(y,-2*pi,2*pi),(z,-2*pi,2*pi),
color='red', viewer='tachyon')
```

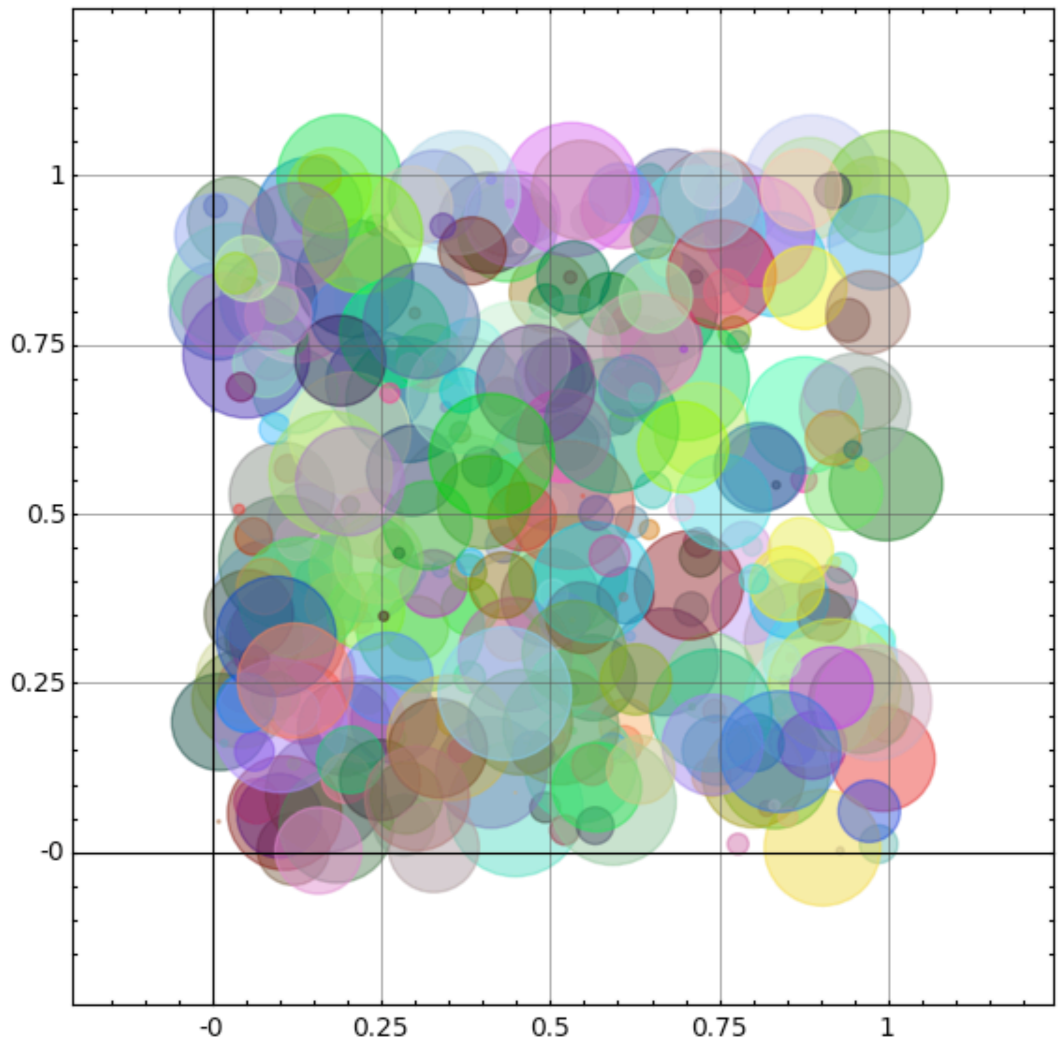


2d Plotting

- **refactoring** -- Mike Hansen has (and is still) refactoring the code to make it much cleaner and easier to work with
- **doctesting** -- Karl-Dieter Crisman recently greatly improved the doctest coverage of 2d plotting
- **tick marks** -- currently poor control over axes labels or tick marks in 2d plots
- **interactivity in the browser (!)** -- massive recent interest in FLOT, Canvas, Network Plotting in Javascript, etc., to make the 2d graphics in the Sage notebook far more interactive and flexible. This is the main next thrust for plotting.

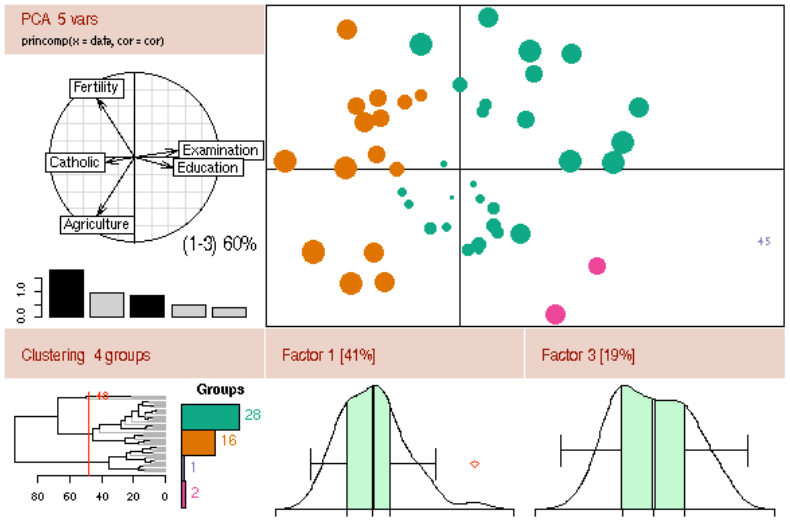
```
@interact
def _(n=(10..500)):
   r = random
   show(sum(circle((r(),r()),r()/10,fill=True,rgbcolor=
(r(),r(),r()),alpha=0.5) for _ in range(n)),
        aspect_ratio=1,frame=True,gridlines=True)
```

n



Statistics

create a natural, clean, native interface to stats functionality



Sage Includes Tons of Statistical Functionality:

- **R --** R is included in Sage, along with the **rpy** C library interface and a pexpect interface to **R** (this was a major Sage-3.0 goal a year ago).
- **scipy.stats --** Scipy's stats functionality is in Sage
- **finance.TimeSeries** -- highly **optimized** statistics (in the context of time series, but much more generally useful).

BUT, there is no nice friendly natural ***Sage*** interface to statistics. E.g.,

```
sage: stat.mean([pi, e, sqrt(2)])
(pi + e + sqrt(2))/3

sage: stats.standard_deviation(...)
sage: d = stats.data([pi, e, sqrt(2)])
sage: d.mean()
...
```

It is important that something like this be designed and implemented soon, since not having such functionality is a ***major shortcoming*** of Sage. Somebody needs to:

1. **Math software stats survey:** Make a survey of the statistics functionality and user interface (API) for each of MATLAB, Maple, and Mathematica. In particular, what special features or ideas do these CAS's have about how statistics and powerful mathematics software (including computer algebra) can be combined?
2. **Stats software survey:** Similarly, survey the leading commercial statistics programs (e.g., SAS, Stata, etc.). What kind of data analysis features are most popular and important in those programs (e.g., which are the features their intro tutorials cover).
3. **Textbooks:** Look at standard textbooks in statistics and see what functionality is needed to do those problems.
4. **Sage Enhancement Proposal (SEP):** Design a rough interface for Sage (classes, functions, plotting capabilities, etc.), and post this for comment to Sage-devel. Always keep ***workflow*** in mind in the design (since I think that's how stats uses work -- they have data that goes in, pictures, analysis, then results that get plotted for papers, etc.)
5. **Prototype:** Do a quick proof-of-concept implementation in pure Python, possibly calling out to R/scipy.stats/etc. for anything at all difficult to implement, so this can get done quickly.
6. **Feedback/optimize:** Get the code from 4 into Sage, get feedback from users, and optimize. Optimization may include creating new Cython bindings to R, improving scipy.stats, etc.

Andrew Hou -- a UW undergraduate -- will likely work on this project with me next quarter, but it would be great to have something going before then.

```
import scipy.stats
```

```
scipy.stats.mean?
```

Arithmetic

many optimizations across the spectrum

- **Integer arithmetic -- eMPIRe:** Sage has switched to eMPIRe, which is a fork of the GMP project. I expect eMPIRe to soon be ***across the board*** superior to GMP.
- **Polynomial arithmtic in Z[x] -- FLINT**: Sage has switched from NTL to FLINT for polynomial arithmetic in **Z[*x*].** Unfortunately, number fields still use NTL -- so number field elements have to be totally rewritten to use FLINT. Using FLINT provides major speedups. It's also getting used more and more in code directly as a library, since it is very easy to use from Cython. (E.g., massively easier than PARI because FLINT is well designed, and easier than NTL, since FLINT is in C instead of C++.)
- **Multivariate polynomial arithmetic -- libSingular**:
  1. libSingular has been a little painful sometimes from a porting perspective, but it seems the bugs are ironed out now.
  2. They now support native arithmetic (and Groebner basis!) over **Z** and **Z/*n*Z** (not just **Q)**, and we'll get that into Sage ASAP.
  3. An exciting idea on the horizon is to make it so Singular can work with any Sage base ring (via PyObject*'s). This would remove the limitation on the size of the base field.
- **Quaternion algebras --** Jon Bober and I completely rewrote them from scratch so that now basic arithmetic is ***very fast***, over **Q** *and* number fields.

Number Theory

quaternion algebras, quadratic forms, modular forms, descent on elliptic curves, Heegner points, L-functions

- **Quaternion Algebras --** Sage now has an algorithm for computing the right ideal classes in an order in a quaternion algebra over **Q**. The code is ***much much faster*** than Magma (or anything else available -- which is nothing).
    1. Critical to algorithms for **computing with modular forms** and modular abelian varieties.
    2. Mainly work of me, Gonzalo Tornaria, Alia Hamieh, and Jon Bober.
    3. Sage does not have the analogue of this over **number fields** (L. Dembele's interest), which Magma has (due mainly to John Voight), but we need to implement it and make our implementation much faster than the one in Magma. Needed for Hilbert modular forms...
- **Quadratic forms:** Jon Hanke's extensive quadratic forms code is now in Sage, and it provides a huge boost in functionality. This was years in the making. Jon is also rapidly improving the quality of this code.
- **Modular forms**: Craig Citro, **David Loeffler**, John Cremona, and I have all been putting a lot of effort into improving all the modular forms functionality in Sage. E.g., David Loeffler has recently filled in gaps all over the place, pushed functionality in new directions, found very subtle bugs, etc. There is still much left to do, e.g., computing weight 1 forms, half integral weight forms, forms on other groups, emuch better code for working with congruence subgroups (Kurth's Farey symbols package).
- **2-descent**: Robert Miller just implemented 2-descent for rational elliptic curves with an isogeny. A huge surprise is that his code -- which combines all the latest idea and algorithms in ***novel ways*** seems to be blazingly fast, totally blowing away everything (e.g., both mwrank and Magma) by orders of magnitude. This will appear in Sage soon.
- **3-descent**: Robert is also working on implementing at least computation of 3-Selmer groups.

```
B = BrandtModule(19,5)
B.right_ideals()
```

```
(Fractional ideal (2 + 10*j, 2*i + 12*j + 2*k, 20*j, 4*k),
Fractional ideal (2 + 10*j + 4*k, 2*i + 12*j + 6*k, 40*j, 8*k),
Fractional ideal (2 + 30*j + 4*k, 2*i + 12*j + 2*k, 40*j, 8*k),
Fractional ideal (2 + 30*j + 4*k, 2*i + 52*j + 2*k, 80*j, 16*k),
Fractional ideal (2 + 30*j + 20*k, 2*i + 52*j + 2*k, 160*j, 32*k),
Fractional ideal (2 + 50*j + 4*k, 2*i + 52*j + 14*k, 80*j, 16*k),
Fractional ideal (2 + 50*j + 12*k, 2*i + 12*j + 14*k, 80*j, 16*k),
Fractional ideal (2 + 130*j + 20*k, 2*i + 52*j + 30*k, 160*j, 32*k),
Fractional ideal (2 + 2*i + 2*j + 2*k, 4*i + 4*j, 20*j + 4*k, 8*k),
Fractional ideal (2 + 2*i + 22*j + 14*k, 4*i + 4*j + 8*k, 40*j +
8*k, 16*k))
```

```
B.quaternion_algebra()
```

```
Quaternion Algebra (-1, -19) with base ring Rational Field
```

```
ModularForms(Gamma1(13),2,prec=12).basis()
```

```
[
q - 4*q^3 - q^4 + 3*q^5 + 6*q^6 - 3*q^8 + q^9 - 6*q^10 + O(q^12),
q^2 - 2*q^3 - q^4 + 2*q^5 + 2*q^6 - 2*q^8 + q^9 - 3*q^10 + O(q^12),
1 + 21060/19*q^11 + O(q^12),
q + 11709/19*q^11 + O(q^12),
q^2 + 262*q^11 + O(q^12),
q^3 + 918/19*q^11 + O(q^12),
q^4 - 882/19*q^11 + O(q^12),
q^5 - 1287/19*q^11 + O(q^12),
q^6 - 1080/19*q^11 + O(q^12),
q^7 - 675/19*q^11 + O(q^12),
q^8 - 360/19*q^11 + O(q^12),
q^9 - 153/19*q^11 + O(q^12),
q^10 - 54/19*q^11 + O(q^12)
]
```

Funding

status report

Sage has received funding from: NSF, DoD, Microsoft, Google, HIMR, UW, Clay, IPAM, PIMS, and private donations

Current funding: Microsoft (5K), NSF (the Sage postdoc, my FRG grant), UW (startup, RRF)

MAGMA PROJECT: I estimate Sage currently reimplements about **80% of Magma's functionality**. Much of the remaining 20% could be filled in by the same people who wrote the code in Magma with additional funding. Example: quaternion algebras over number fields -- John Voight would love to reimplement his algorithms in Sage, if he had funding. Also, Robert Bradshaw could implement a parser for the Magma language.

PEOPLE: Carl Witty, Jason Moxham, and Mike Hansen: three absolutely brilliantly highly qualified programers, who are passionate about working on Sage. They would all love to work fulltime on Sage (for a very reasonable price), but mostly can't because of lack of funding.

Example -- Carl Witty: For example, Carl Witty is Chief Science Officer of Newton Labs in Renton, WA, but he finds his day job boring in comparison to Sage. He would like to finish writing a superb implementation of Cylindrical Algebraic Decomposition for Sage. Cylindrical algebraic decomposition is an algorithm for finding descriptions of semi-algebraic sets, i.e., the real solution sets of systems of multivariate polynomial equalities and inequalities. Mathematica uses CAD for exactly solving many problems for polynomials or systems of polynomials over the reals, including finding solution sets, finding maxima or minima, quantifier elimination, etc. For example, all of Minimize, Maximize, Reduce, FullSimplify, and SparseArray indirectly use CAD when operating on systems of real polynomial equations (or certain trigonometric equations that can be automatically transformed into polynomials) and no simpler algorithm is available.

Conclusion: Sage has huge momentum, and there is no question Sage is going to continue its dramatic growth.

- Funding to hire Carl Witty, Jason Moxham, and Mike Hansen on a fulltime basis would make a tremendous difference in how quickly Sage improves.
- Funding to support mathematicians to reimplement the other 20% of missing functionality from Magma would make Sage a fully viable alternative to Magma.