Open-Source Mathematics: Sage

Martin Albrecht (M.R.Albrecht@rhul.ac.uk)

Information Security Group,
Royal Holloway, University of London

PhD Seminar, November 6th, 2008

Outline

Table of Contents

http://urbandictionary.com

Sage:

1. Word entered to the email field of a *chan (iichan, 4chan, 2chan, etc) message board so that the post is not bumped. It is Japanese for 'to lower.' It is the opposite of 'age' (ah-gay), which means to raise.
2. An antiquated word meaning a man of deep mind or wisdom, carrying the capacity to give advisement within more typically the management of life or spiritual and self-realizing matters.
3. A large massive blemish, boil or pimple; found on the top of one's nose. "holy crap, i got the biggest sage on my nose, and prom is tomorrow!"
4. Originating from the slang 'safe' and used in the same context.
   A: "Can I borrow your coat?"
   B: "Yeah"
   A: "Sage."
5. A scammer, thief, stealer. . . .

http://www.sagemath.org

Sage

is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface.

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

. . . I will stick to this definition.

Why? I

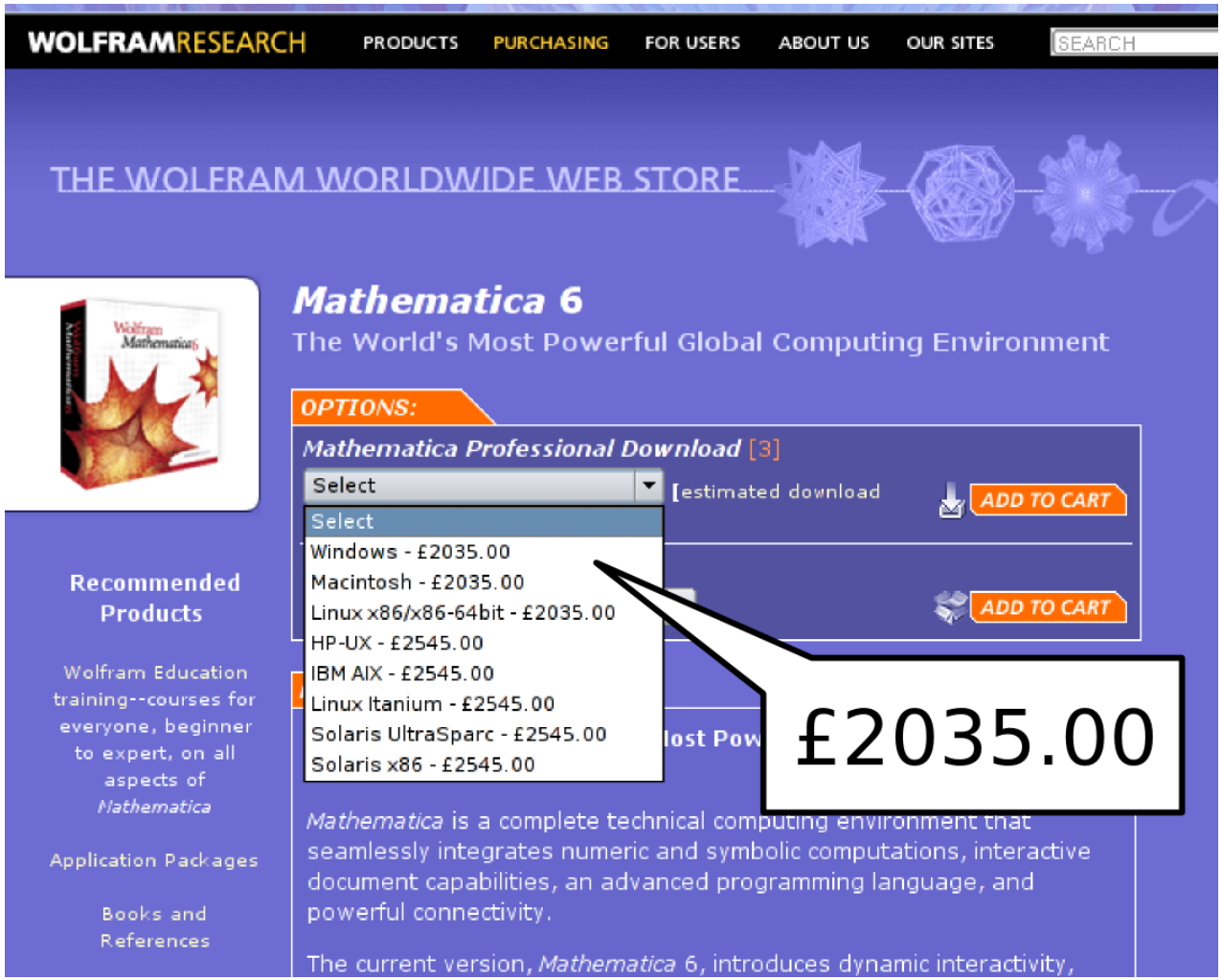
"Neubüser's Law"

"You can read [some] Theorem and its proof in [some] book in the library [...] then you can use [that] Theorem for the rest of your life free of charge, but for many computer algebra systems license fees have to be paid regularly [...]. You press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation **two of the most basic rules of conduct in mathematics are violated**: In mathematics **information is passed on free of charge** and **everything is laid open for checking**. Not applying these rules to computer algebra systems that are made for mathematical research [...] means **moving in a most undesirable direction**. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?" – J. Neubüser (1993) (he started GAP in 1986).

Why? II


Information Security Group


WOLFRAMRESEARCH
PRODUCTS
PURCHASING
FOR USERS
ABOUT US
OUR SITES
SEARCH
THE WOLFRAM WORLDWIDE WEB STORE
Mathematica 6
The World's Most Powerful Global Computing Environment
OPTIONS:
Mathematica Professional Download [3]
Select
[estimated download
ADD TO CART
ADD TO CART
Select
Windows - £2035.00
Macintosh - £2035.00
Linux x86/x86-64bit - £2035.00
HP-UX - £2545.00
IBM AIX - £2545.00
Linux Itanium - £2545.00
Solaris UltraSparc - £2545.00
Solaris x86 - £2545.00
£2035.00
Recommended Products
Wolfram Education training--courses for everyone, beginner to expert, on all aspects of Mathematica
Application Packages
Books and References
Mathematica is a complete technical computing environment that seamlessly integrates numeric and symbolic computations, interactive document capabilities, an advanced programming language, and powerful connectivity.
The current version, Mathematica 6, introduces dynamic interactivity,

Why? III

"Why You Do Not Usually Need to Know about Internals"

"Particularly in more advanced applications of **Mathematica**, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the **analyses will not be worthwhile**. For the internals of **Mathematica** are **quite complicated**, and even given a basic description of the algorithm used for a particular purpose, it is usually **extremely difficult** to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

– Mathematica Tutorial (2008)
`http://reference.wolfram.com`

Mission Statement

Mess with the Best

Provide an open source, high-quality, and free viable alternative to **Magma, Mathematica, Maple** and **MATLAB**.

In other words: create a unified mathematics software package for algebra, calculus, elementary to very advanced number theory, cryptography, numerical computation, commutative algebra, group theory, combinatorics, graph theory, exact linear algebra and more.

To achieve this do not reinvent the wheel but **reuse** as much **existing building blocks** as possible and make sure the result is **rigorously tested**, **easy to modify** by the end user and **very well documented**.

Also create a **helpful environment** for users to get help (mailinglists, irc-channel, meetings, coding sprints).

What is Sage?

Sage is a mathematics software package developed by a **worldwide** community of developers.

1. a **distribution** of the best free, open-source mathematics software available that is easy to compile or install from binaries,
2. an **interface** to most free and commercial mathematics software packages,
3. a **new library**, which uniformly covers the widest area of **functionality**, including several new implementations not yet found elsewhere.

```
| SAGE Version 3.1.3, Release Date: 2008-10-14
| Type notebook() for the GUI, and license() for information.
```

```
sage: 2 + 3
5

sage: K = Zp(5) # 5-adic Ring with capped relative precision 20
sage: K(2) + K(3)
5 + O(5^20)
```

Table of Contents

Python & Cython

Sage does not come with yet-another ad-hoc math language, it uses **Python** instead.

- one of the most widely used programming languages (Google, IML, YouTube, NASA),
- easy for you to define your own data types and methods on it (bitstreams, ciphers, rings, whatever),
- Very clean language that results in easy to read code,
- a **huge number of libraries**: statistics, networking, databases, bioinformatic, physics, video games, 3d graphics, numerical computation (scipy), and serious "pure" mathematics (via Sage)
- easy to use existing C/C++ libraries from Python (via **Cython**)

Python Example: Databases


```
On 11/1/07, Jack <XXXX.XXXXXXXXX@XXXXX.XXX> wrote:
> Is it possible to fetch information from a database (say, PostgreSQL)
> with sage?  If so, how is it done?  If not, is there  a workaround
> that people use?

Using Psycopg2 is the most popular way of using PostgreSQL from python,

see:
http://www.initd.org/tracker/psycopg/wiki/PsycopgTwo
...
```

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Python Example: Pen-Testing

Scapy is a powerful interactive packet manipulation program written in Python. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery (it can replace hping, 85% of nmap, arpspoof, arp-sk, arping, tcpdump, tethereal, p0f, etc.).

```
sage: from scapy import rdpcap, TCP, IP
sage: packets = [p[IP] for p in rdpcap("/home/malb/example.pcap") \
                 if p[TCP] and len(p[TCP]) > 32]
...
sage: def get_uint(pl, length):
...

sage: p_length, pl = get_uint(pl, 4)
sage: p, pl = get_uint(pl, p_length)
sage: g_length, pl = get_uint(pl, 4)
sage: g, pl = get_uint(pl, g_length)
sage: Zp = GF(p)
sage: g = Zp(g)
sage: e = g**ZZ.random_element(0,p)
sage: e.log(g) # yeah, right ;-)
```

Cython


```
from sage.misc.misc import cputime
cdef extern from "math.h":
  cdef float sinf_c "sinf"(float x)

def sinf(x):
  return sinf_c(x)

def timesinf(x):
  cdef int i
  cdef float _x = float(x)
  t = cputime()
  for i from 0 <= i < 1000000:
    _ = sinf_c(_x)
  return cputime(t)
```

```
sage: sinf(1.0), sin(1.0)
(0.841470984807897, 0.841471016407012 94)
sage: %timeit sinf(1.0)
625 loops, best of 3: 3.55 micro seconds per loop
sage: %timeit sin(1.0)
625 loops, best of 3: 12.3 micro seconds per loop
sage: timesinf(1.0)
0.007998999999998673
```

Symmetric Multiprocessing (SMP)

Often, parallel computing is hard. But it is quite easy for coarse grained embarrassingly (proudly) parallel problems:

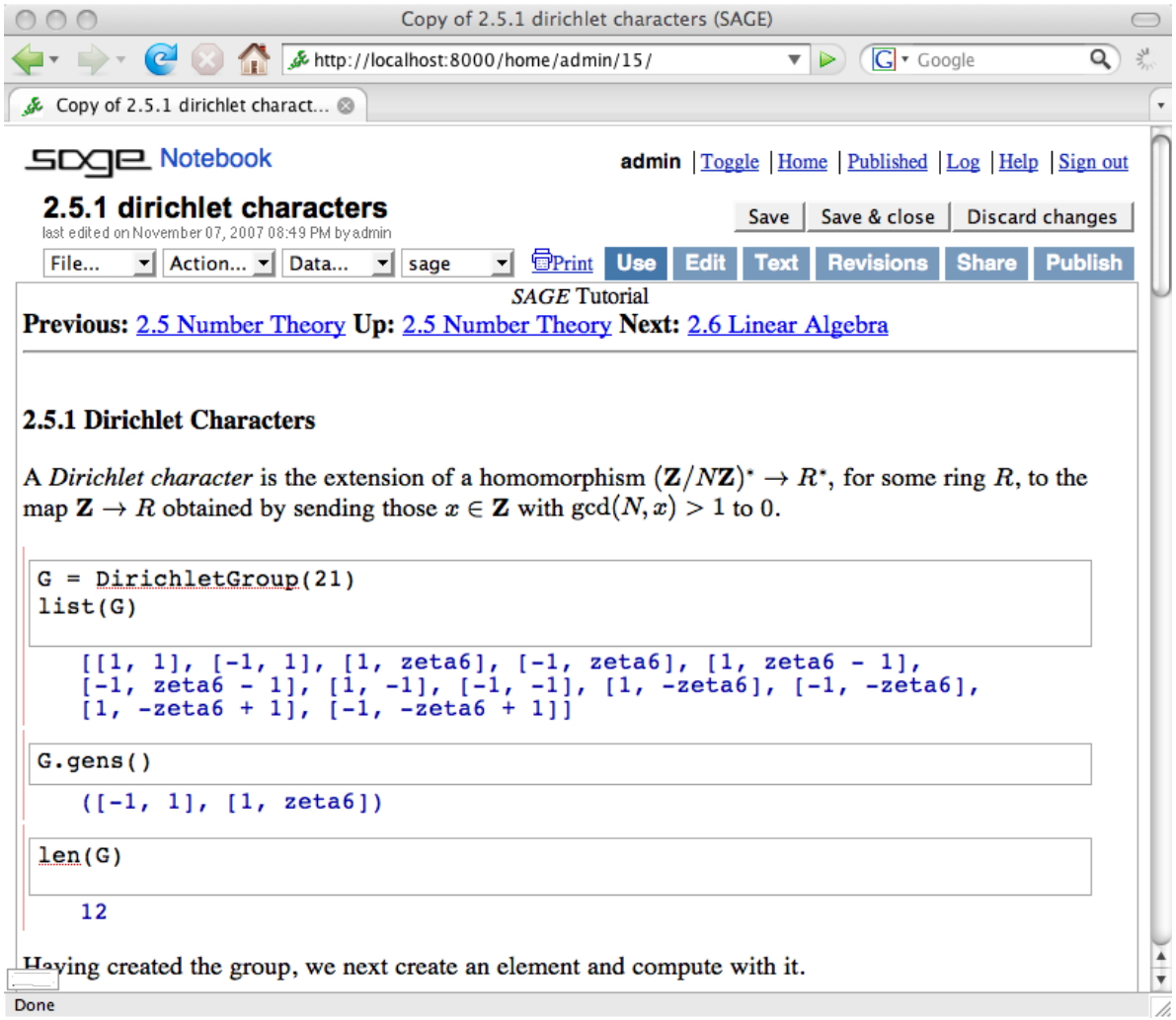
```
sage: @parallel(2)
sage: def f(n):
...       return factor(n)
sage: %time _ = [f(2^217-1), f(2^217-1)]
CPU time: 2.05 s,  Wall time: 2.08 s
sage: %time _ = list( f([2^217-1, 2^217-1]) )
CPU time: 0.01 s,  Wall time: 1.11 s
sage: 1.11/2.08
0.533653846153846
```

For Grid computing

. . . try DSage or iPython.

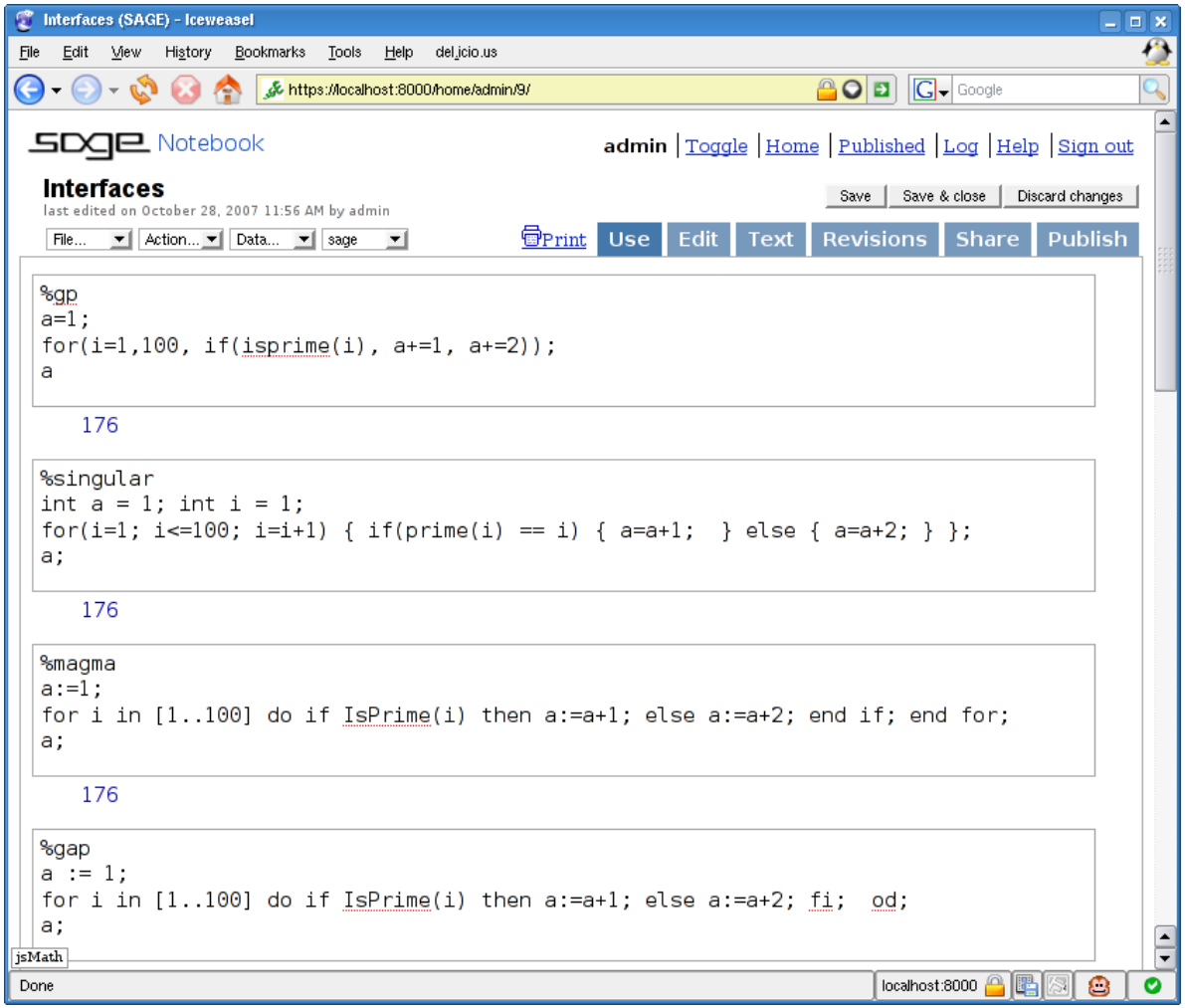
Web-based Notebook Interface

public notebooks available at `http://www.sagenb.org`



- graphical user interface
- 2D plotting
- interactive 3D plotting
- LaTeX typesetting
- web service (AJAX, SSL) inspired by Google Docs
- worksheet sharing
- worksheet up-/download

GUI for Many Mathematics Packages



Examples:

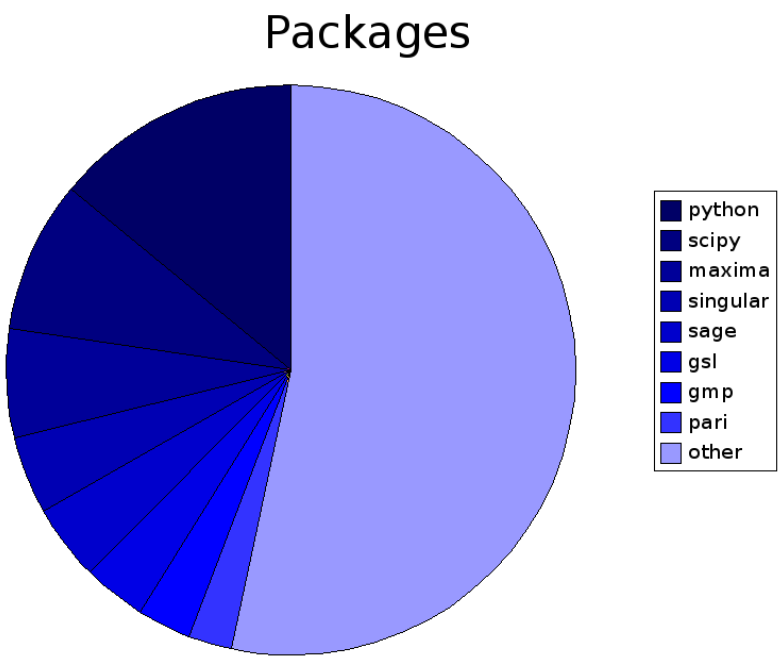
- Pari
- Maxima
- Singular
- Gap
- Mathematica
- Maple
- . . .

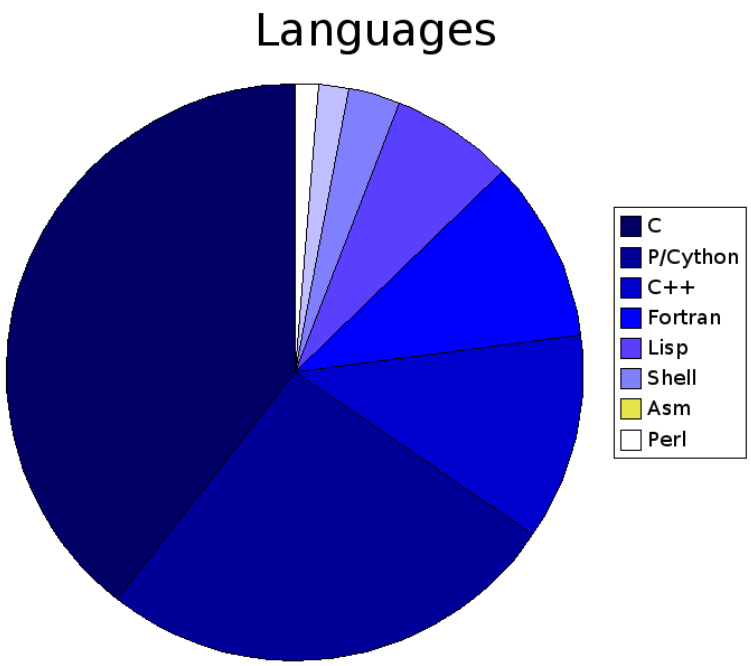
A Distribution I

Arithmetic	**GMP, MPFR, Givaro, MPFI**
Commutative Algebra	**PolyBoRi, SINGULAR** (libSINGULAR)
Linear Algebra	**LinBox, M4RI, IML, fpLLL**
Cryptosystems	**GnuTLS, PyCrypto**
Integer Factorization	**FlintQS, ECM**
Group Theory	**GAP**
Combinatorics	**Symmetrica, sage-combinat**
Graph Theory	**NetworkX**
Number Theory	**PARI, NTL, Flint, mwrank, eclib**
Numerical Computation	**GSL, Numpy, Scipy, ATLAS**
Calculus, Symbolic Comp.	**Maxima, Sympy, Pynac**
Statistics	**R**
User Interface	Sage Notebook, **jsmath, Moin wiki, IPython**
Graphics	**Matplotlib, Tachyon, libgd, JMol**
Networking	**Twisted**
Databases	**ZODB, SQLite, SQLAlchemy**, Python pickle
Programming Language	**Python, Cython** (compiled)

Overall more than 70 packages are included and shipped with **Sage**.

A Distribution II





Overall **very** roughly 4.5 million lines of source code and estimated several hundred person-years.

A Unified Interface

- **Sage** interfaces to: Axiom, GAP, GP/PARI, Kash, Macaulay2, Magma, Maple, Mathematica, MATLAB, Maxima, MuPad, Octave, Singular, ...
- This gives **Sage** a wide range of functionality.

Example:

```
sage: x = gp('9+6') # the GP/PARI math software
```

This fires up one copy of GP/PARI and sends the line `'sage[1] = 9+6'` to it

```
sage: x, x.name()
15, 'sage[1]'
sage: x.factor()
[3, 1; 5, 1]
```

Another Example:

```
sage: I.groebner_basis() # Singular
```

Table of Contents

Live Demo

Table of Contents

Getting Started With Sage

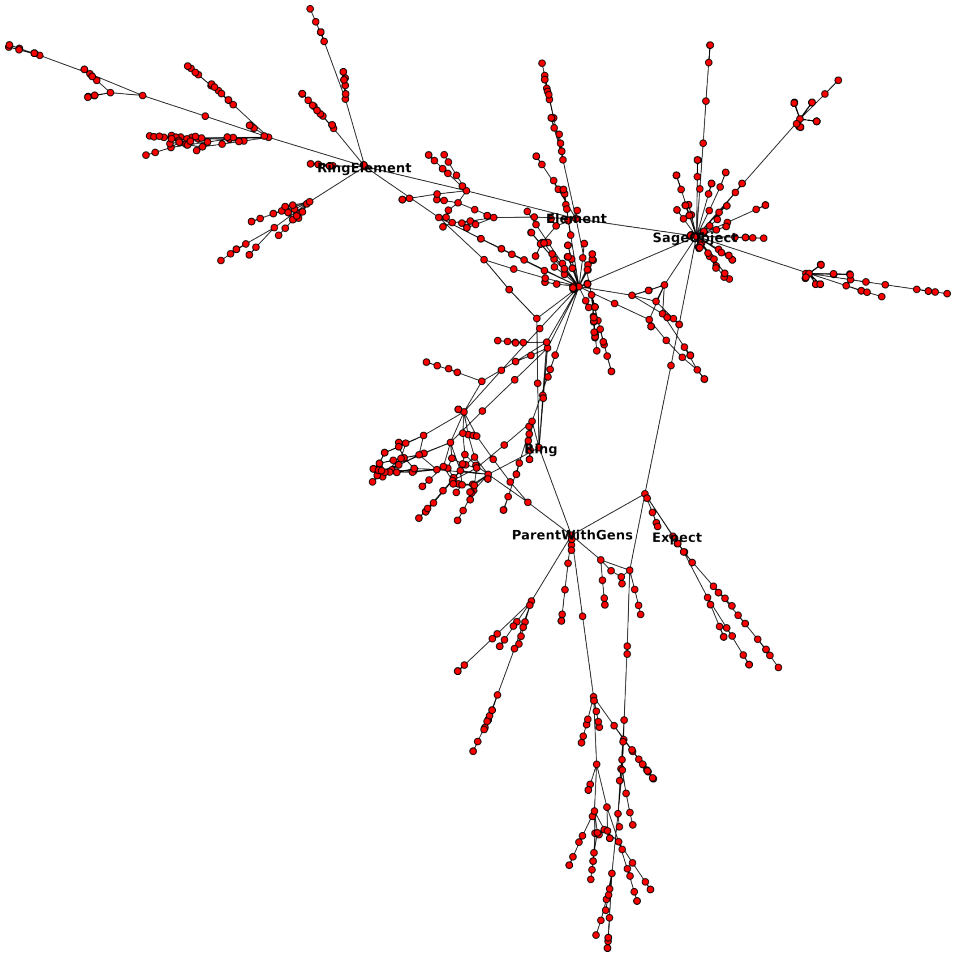
Ways to use Sage:

- download source; type 'make', wait, run (Linux, OSX, Solaris (soon))
- download binaries for a wide range of platforms
- download VMWare image (e.g. for Windows)
- try it online: `http://www.sagenb.org`

Documentation:

- **Sage** installation guide
- interactive tutorial
- comprehensive reference manual
- "How Do I Construct . . ." manual
- **Sage** developer's guide
- `http://wiki.sagemath.org`
- sage-support mailing list

Questions?



Thank You!

Table of Contents

Developers

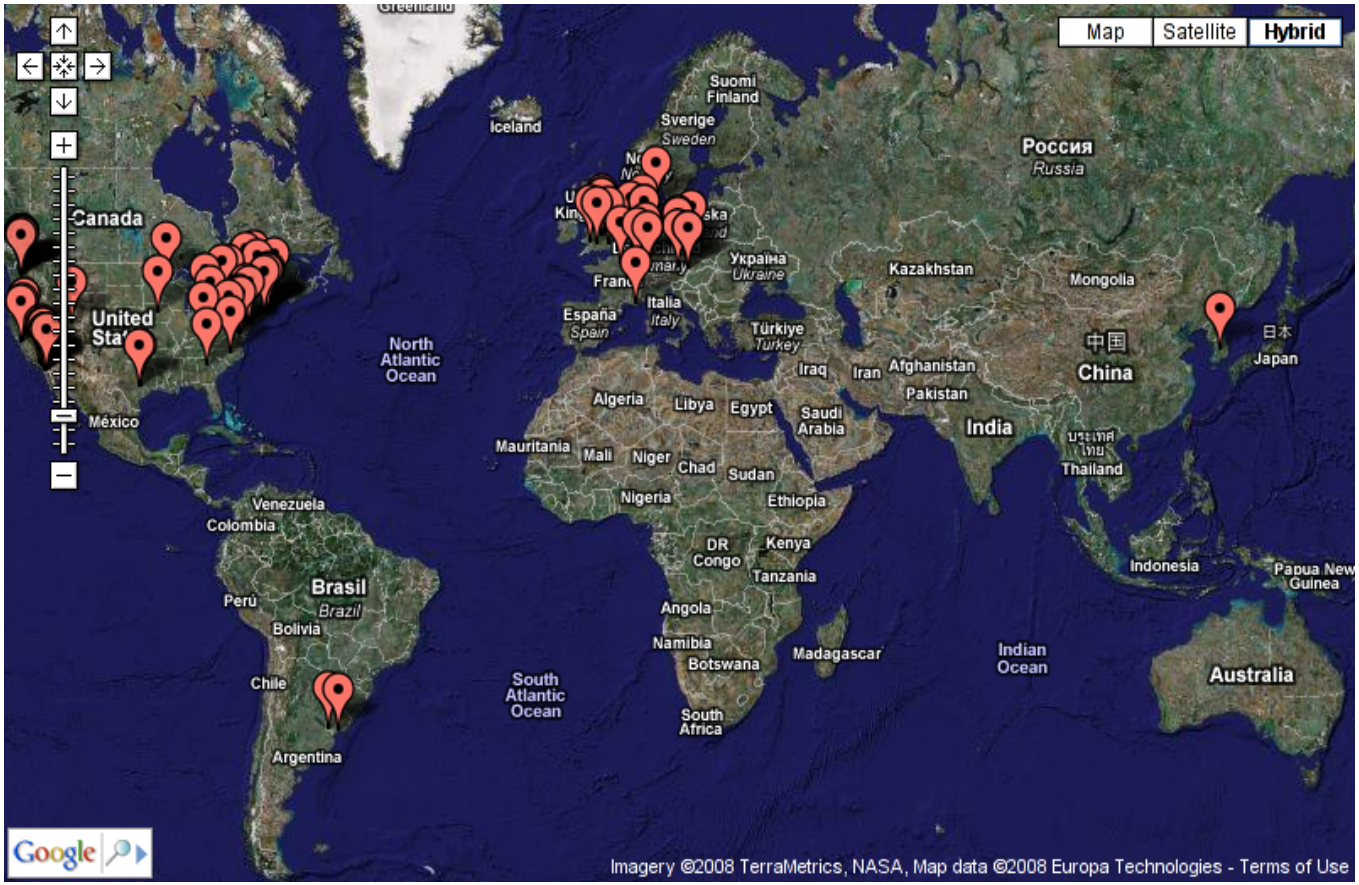


Figure: roughly 200 people have directly contributed to **Sage** so far.

Users

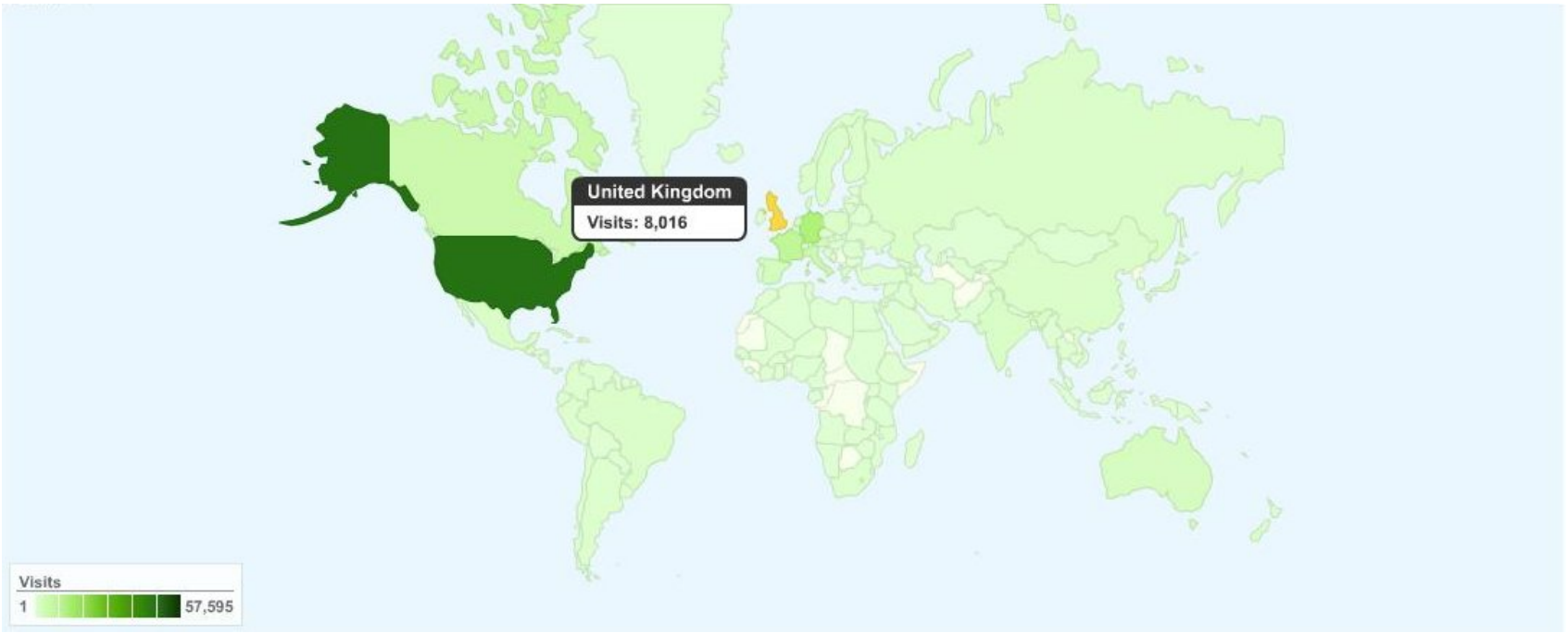


Figure: `http://www.sagemath.org`