Open-source math software, SageMath and Cython

Harald Schilly

2017-05-06

Linuxtage: PyDays Vienna, 2017

The big picture

The Big Picture

Software up to the end of 1979:

- **Fortran**: LINPACK (later LAPACK), BLAS, etc.
- **Macsyma** (later Maxima): symbolic computing
- **S** Programming language (later R)

Open-source until 2000:

- **R**: emerges as a serious statistics and data analysis platform
- **Maxima**: open-source computer algebra system
- **Python**: invented early 90s, based on ABC, very user-friendly

Mid-2000 until now:

- **Python**: growing usage in scientific computing, data analysis, machine learning, etc.
- **SageMath**: Python-based environment for mathematical computing
- **R**: de-facto standard for scientific publications in statistics
- ... and many more emerging tools and libraries like **Julia**

Shifting Paradigm: Open by Default

Open-source and Open-access

Scientific publications, databases, programming languages, libraries, file formats, etc.: all are shifting towards being *open and accessible*.

Shifting Paradigm: Open by Default

Open-source and Open-access

Scientific publications, databases, programming languages, libraries, file formats, etc.: all are shifting towards being *open and accessible*.

Networked Computing

The personal computing area brought rise to packaged software for users.
This model already shifted towards *Software as a Service* (SaaS).

Shifting Paradigm: Open by Default

Open-source and Open-access

Scientific publications, databases, programming languages, libraries, file formats, etc.: all are shifting towards being *open and accessible*.

Networked Computing

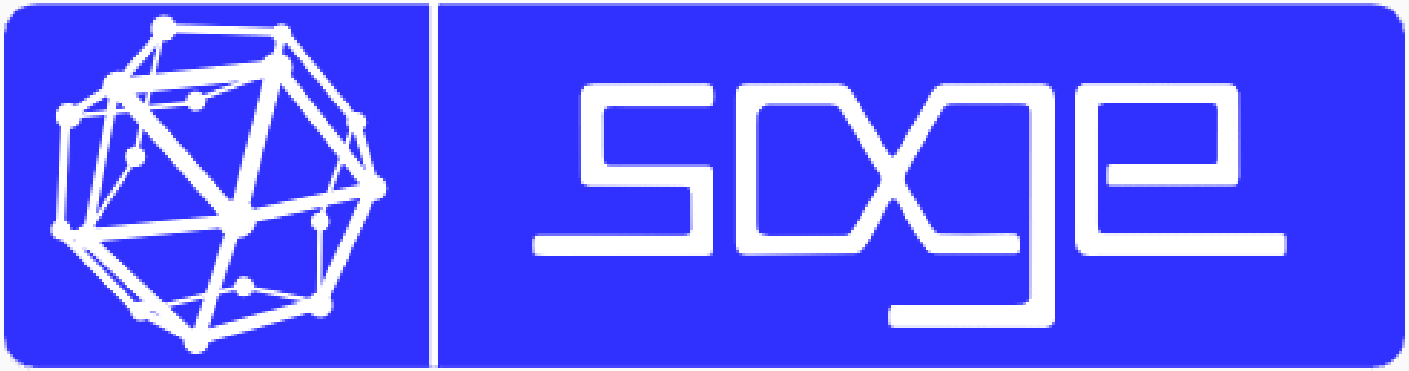
The personal computing area brought rise to packaged software for users.
This model already shifted towards *Software as a Service* (SaaS).

Collaboration

Software development happens publicly and worldwide (e.g. *GitHub*).
Research collaboration has no borders.
Proprietary software locks up users in walled gardens.
"Open Data" initiatives: *Zenodo*, *OpenAIRE*, ...
Reproducible Research.

SageMath

SageMath



http://sagemath.org/

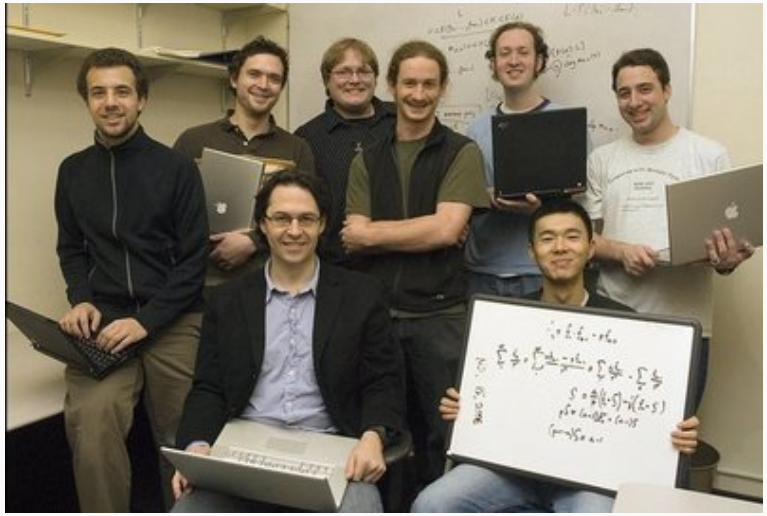
Quick Survey

- Who has ever heard of SageMath before?
- Who has used SageMath?
- Who has contributed to SageMath?

History

2004: **William Stein** started SageMath at Harvard.
Motivation: Frustrated with closed-source mathematics software and in particular with Magma.

2005: First version of Sage ever.
2006: After lots of hard work, small team at University of Washington formed around it.
2009: Traction grew, Sage notebook, ...
2015: Started work on "SageMathCloud" – an open source online service (SaaS) to overcome the inertia of running software on a local machine.



SageMathCloud

`http://cloud.sagemath.com/`

Motivation and Goals

Motivation

Frustration with the state of mathematical software: only commercial players and fragmented academic software.

Goals

Some of the general goals behind SageMath:

- Unify fragmented academic mathematical software.
- Easier installation/distribution of the software.
- Use type system to express mathematical knowledge.
- Allow for mixing instances of such types in calculations ("coercion"), e.g., multiplying a matrix over $\mathbb{Z}$ with an element in $\mathbf{F}_2$.
- Foster a mathematical research platform.

Solutions

Solutions

- Uses a common widely used programming language and use “types” to express mathematical objects in code.
- Package many open-source tools in a consistent manner.
- Stands on the shoulders of giants: uses existing software packages like Pari/GP, Python, Matplotlib, R, SymPy, Maxima, etc. In total, about 100 software packages.
- The core library uses these tools and implements its own algorithms;
- An extensive *test suite* ensures that the whole collection of functionality works well together.

Solutions

Solutions

- Uses a common widely used programming language and use "types" to express mathematical objects in code.
- Package many open-source tools in a consistent manner.
- Stands on the shoulders of giants: uses existing software packages like Pari/GP, Python, Matplotlib, R, SymPy, Maxima, etc. In total, about 100 software packages.
- The core library uses these tools and implements its own algorithms;
- An extensive *test suite* ensures that the whole collection of functionality works well together.

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

Python: core engine behind SageMath

Benefits of Python

- Easy to learn and teach: many ideas originate from the ABC language.
- Powerful and universal: mathematical objects are instances of "types" in Python.
- Widely used and supported by the industry: Google, Microsoft, etc.
- Spillover effect: learning SageMath means also learning Python.
- Since mid-2000s, thriving ecosystem in engineering, numerical mathematics, big data and machine learning.
- Many other Python libraries can be accessed from within SageMath; it's easy to interoperate!

Example: Mathematical Types in Knot Theory

First, define a Knot by its "oriented Gauss code".

```
K = Link([[[-1, 2, -4, 5], [1, -3, 4, -6], [-2, 3, -5, 6]],
[-1, 1, -1, 1, -1, 1]])
```

Orientation:

$\texttt{K.orientation()} = [-1, 1, -1, 1, -1, 1]$

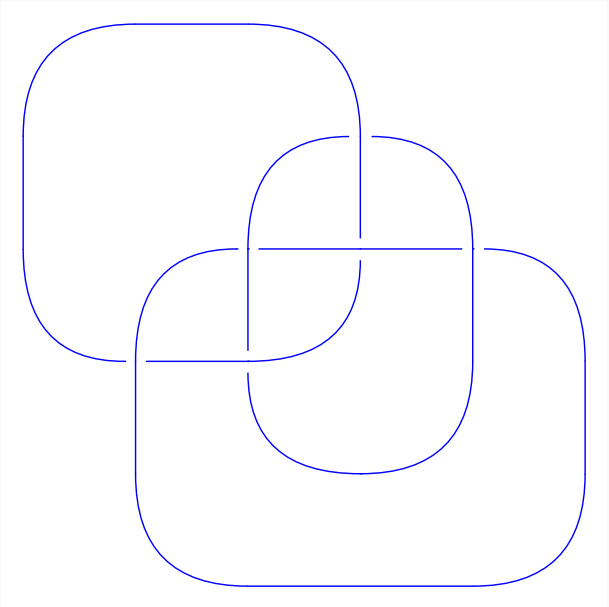
Number of components:

$\texttt{K.number\_of\_components()} = 3$

Alexander Polynomial:

$\texttt{K.alexander\_polynomial()} =$

$\frac{1}{t^2} + \frac{-4}{t} + 6 - 4t + t^2$



Example: Graph theory and Pandas /1

Create random graphs with varying number of nodes "N" an increasing probability of connecting nodes "p" and calculate their average distance:

```
data = []
for n in range(2, 20):
    print n,
    for p in srange(0.1, 1, .01, include_endpoint=True):
        for i in range(100):
            g = graphs.RandomGNP(n, p)
            d = g.diameter()
            di = np.nan if d == oo else int(d)
            data.append([int(n), float(p), di])
```

This generates about 200,000 graphs and data-points in a bit more than a minute!

Example: Graph theory and Pandas /1

Create random graphs with varying number of nodes "N" an increasing probability of connecting nodes "p" and calculate their average distance:

```
data = []
for n in range(2, 20):
    print n,
    for p in srange(0.1, 1, .01, include_endpoint=True):
        for i in range(100):
            g = graphs.RandomGNP(n, p)
            d = g.diameter()
            di = np.nan if d == oo else int(d)
            data.append([int(n), float(p), di])
```

This generates about 200,000 graphs and data-points in a bit more than a minute!

Hypothesis: **better connected graphs have a smaller distance**

Example: Graph theory and Pandas /2

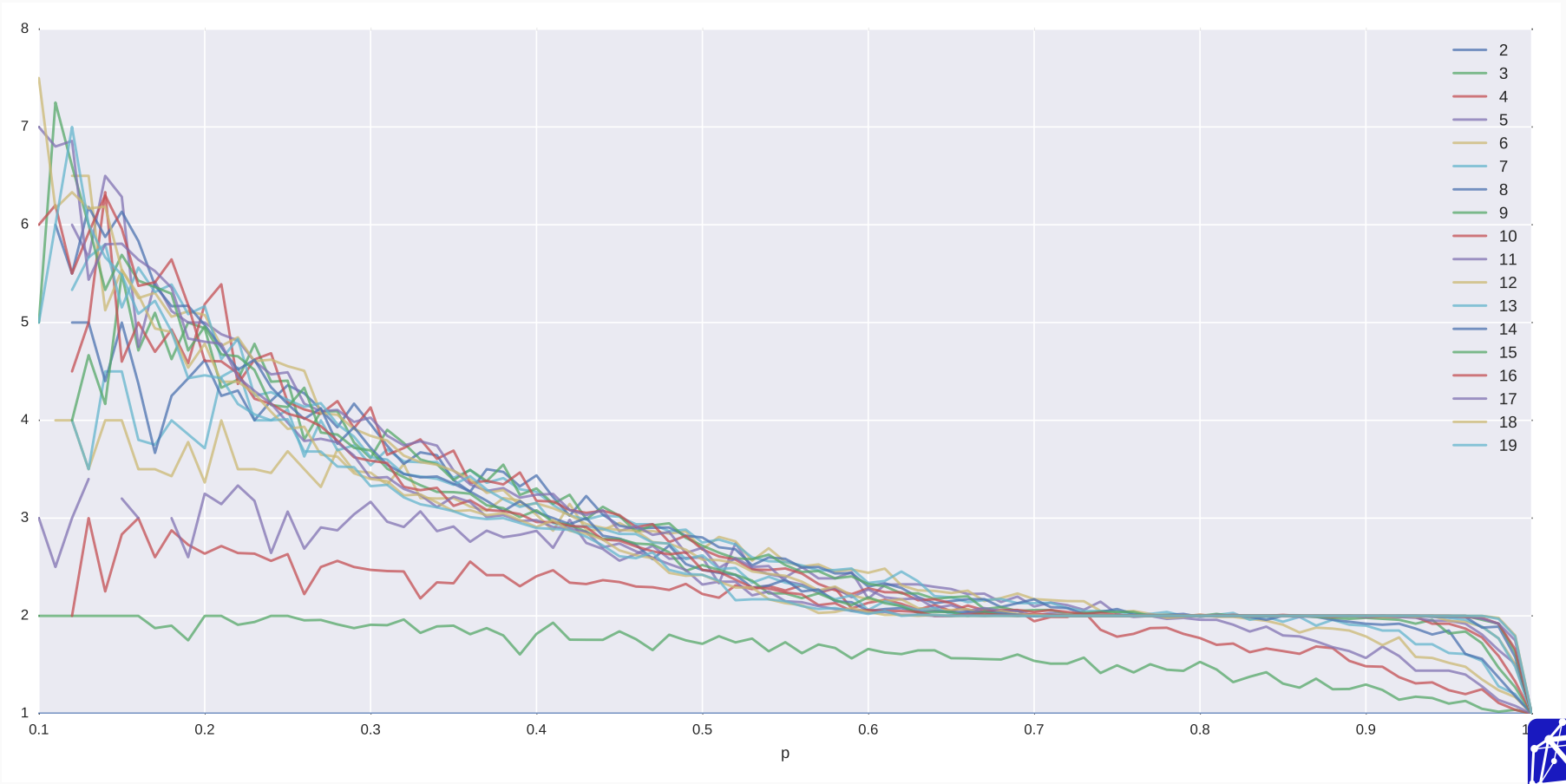
```
diameters = pd.DataFrame(data, columns=['N', 'p', 'diameter'])
```

	N	p	diameter
100000	5	0.7	1.0
100001	5	0.7	2.0
100002	5	0.7	4.0
100003	5	0.7	2.0
100004	5	0.7	3.0
100005	5	0.7	2.0
100006	5	0.7	3.0
100007	5	0.7	2.0
100008	5	0.7	3.0
100009	5	0.7	2.0
100010	5	0.7	2.0
100011	5	0.7	2.0
100012	5	0.7	2.0
100013	5	0.7	2.0
100014	5	0.7	3.0
100015	5	0.7	2.0
100016	5	0.7	2.0
100017	5	0.7	2.0
100018	5	0.7	2.0
100019	5	0.7	2.0

Example: Graph theory and Pandas /3

Use pandas to process the data to calculate means for each group:

```
for idx, grp in diameters.groupby('N'):
    grp.groupby('p').mean().plot(y='diameter',alpha=.75, ax=ax, label=
```



Example: SageTeX

LaTeX with embedded calculations

SageTeX is a LaTeX package for running SageMath computations right inside a document. Results are directly embedded and cached between runs!

Examples

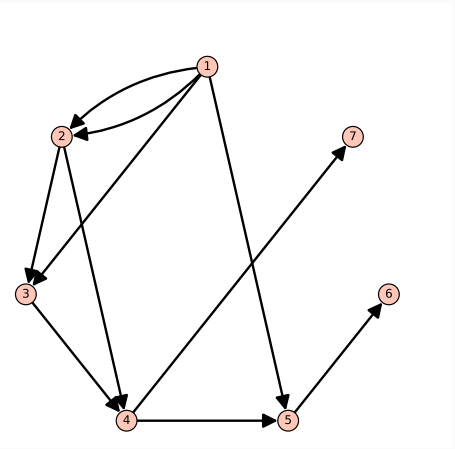
- Inline commands: `\sage{factor(2016)}` $= 2^5 \cdot 3^2 \cdot 7$
- `\sage{integrate(x^2*sin(x), x)}`:

$$\int x^2 sin(x)\,\mathrm{d}x = -\left(x^2 - 2\right)\cos\left(x\right) + 2\,x\sin\left(x\right)$$

- Define a graph G_4:

```
\begin{sageblock}
G4 = DiGraph({1:[2,2,3,5],      \
              2:[3,4], 3:[4],   \
              4:[5,7], 5:[6]}, \
             multiedges=True)
G4plot = G4.plot(layout='circular')
\end{sageblock}
```

Plot via `\sageplot{G4plot}`:



Cython

`http://cython.org/`

- Cython is an optimising static compiler for *Python*
  - not compatible with the whole Python language (subset)
  - adds a couple of new keywords (superset)
- Extended Python language for
  - writing fast Python extension modules
  - interfacing with C/C++ libraries

Cython History

- about 2002: originates from **Pyrex** by Greg Erwing
- *SageX* fork for SageMath (Robert Bradshaw and William Stein)
- LXML fork by Stefan Behnel
- **July 28th, 2007:** merging both is Cython's official birthday
- Bradshaw and Behnel are the maintainers since then.
- 2008, Dag Seljebotn did pathbreaking work on directly interfacing with NumPy (sponsored by Google and Enthought) – some operations got a 1000x speedup!
- Today, heavily used inside SageMath (649,409 lines of Cython in release 7.6)
- Recently, got track with other Python libs (Pandas, scikit-learn, etc.)

Cython Core Features

- **Translation to C**: a mix of Python and Cython code is transspilled to pure C
- **Incremental:** enhance your Python code incrementally – instead of rewriting everything.
- **Module**: The compiled static object is loaded directly from Python – there is no intermediate layer!
- **Optional typing**:
  - `cdef double x`: floating point variable x
  - `cpdef int func(int i, int j)`: function $f(< int >, < int >) \rightarrow < int >$.
  - ... but you can keep "pure Python" statements
- Support for classes, includes wrapping C++ classes, explicit memory allocation, etc.
- Import functions from C header files – e. g. `cimport numpy`

Cython Testimonials

"You guys rock! In scikit-learn, we have decided early on to do Cython, rather than C or C++. That decision has been a clear win because the code is way more maintainable. We have had to convince new contributors that Cython was better for them, but the readability of the code, and the capacity to support multiple Python versions, was worth it."

— Gaël Varoquaux, scikit-learn

"The biggest surprise (and of course this is Cython's selling point) is how simple the interfacing between high level and low level code becomes, and the fact that it is all very robust."

— Fredrik Johansson, mpmath,

Cython Example /1

```
def f(x):
    return x**3 - 3 * x

def integrate_func(f, a, b, N):
    s = 0
    dx = float(b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx

>>> %timeit
>>> integrate_func(f, -2, 2, 10000)

5 loops, best of 3: 136 ms per loop
```

```
cpdef double cy_f(double x):
    return x**3 - 3 * x

cpdef double cy_integrate_func(f, \
                double a, double b, int N
    cdef double s = 0
    cdef double dx = f(b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx

>>> %timeit
>>> cy_integrate_func(cy_f, -2, 2, 100

125 loops, best of 3: 1.65 ms per loop
```

82.4x speedup

Cython Example /2

About 10 lines of Python code expand to about 2000 lines of commented C! Here a small excerpt:

```
 /* "my_funcs.pyx":16
* cpdef int fib(int i):
*     assert i >= 0
*     if i == 0 or i == 1:
*         return 1
*     else:
*         return fib(i-1) + fib(i-2)
*/
 switch (__pyx_v_i) {
   case 0:
   case 1:
   __pyx_r = 1;
   goto __pyx_L0;
   break;
   default:
   __pyx_r = (__pyx_f_8my_funcs_fib((__pyx_v_i - 1), 0) +
             __pyx_f_8my_funcs_fib((__pyx_v_i - 2), 0));
   goto __pyx_L0;
   break;
 }
```

Cython Pro-Tipp: Annotated HTML

For debugging where time is being spent, run `cython -a` to produce annotated HTML:

Generated by Cython 0.25.2

Yellow lines hint at Python interaction.
Click on a line that starts with a "+" to see the C code that Cython generated for it.

Raw output: my_funcs.c

```
 01: cdef extern from "math.h":
 02:     double log1p(double)
 03:
+04: cdef float f2(double x):
+05:     return log1p(x)
 06:
+07: cpdef float integrate(float a, float b, int N):
+08:     cdef float dx = (b - a) / N
+09:     cdef float s = 0
+10:     for i in range(N):
+11:         s += f2(a + i * (b - a) / N)
+12:     return s * dx
 13:
+14: cpdef int fib(int i):
+15:     assert i >= 0
+16:     if i == 0 or i == 1:
+17:         return 1
 18:     else:
+19:         return fib(i-1) + fib(i-2)
```

Thank You!

Harald Schilly – harald@schil.ly

http://sagemath.org/

http://cloud.sagemath.com/

http://cython.org/