A Tour Of Sage

Release 9.2

The Sage Development Team

Oct 25, 2020
1  Sage as a Calculator 3
2  Power Computing with Sage 5
3  Accessing Algorithms in Sage 9
This is a tour of Sage that closely follows the tour of Mathematica that is at the beginning of the Mathematica Book.
The Sage command line has a `sage:` prompt; you do not have to add it. If you use the Sage notebook, then put everything after the `sage:` prompt in an input cell, and press shift-enter to compute the corresponding output.

\[
\text{**sage:** } 3 + 5 \\
8
\]

The caret symbol means “raise to a power”.

\[
\text{**sage:** } 57.1^100 \\
4.60904368661396\times10^{175}
\]

We compute the inverse of a $2 \times 2$ matrix in Sage.

\[
\text{**sage:** } \text{matrix}([[1,2], [3,4]])^{-1} \\
\begin{bmatrix}
-2 & 1 \\
3/2 & -1/2
\end{bmatrix}
\]

Here we integrate a simple function.

\[
\text{**sage:** } x = \text{var}('x') \quad \# \text{ create a symbolic variable} \\
\text{**sage:** } \text{integrate}(\sqrt{x}\sqrt{1+x}, x) \\
\frac{1}{4}(x + 1)^{3/2}/x^{3/2} + \sqrt{x + 1}/\sqrt{x}/((x + 1)^2/x^2 - 2*(x + 1)/x + 1) - \\
-1/8\log(\sqrt{x + 1}/\sqrt{x} + 1) + 1/8\log(\sqrt{x + 1}/\sqrt{x} - 1)
\]

This asks Sage to solve a quadratic equation. The symbol == represents equality in Sage.

\[
\text{**sage:** } a = \text{var}('a') \\
\text{**sage:** } S = \text{solve}(x^2 + x == a, x); S \\
x == -1/2*\sqrt{4*a + 1} - 1/2, x == 1/2*\sqrt{4*a + 1} - 1/2
\]

The result is a list of equalities.

\[
\text{**sage:** } S[0].\text{rhs}() \\
-1/2*\sqrt{4*a + 1} - 1/2
\]

Naturally, Sage can plot various useful functions.

\[
\text{**sage:** } \text{show}(\text{plot}\left(\sin(x) + \sin(1.6*x), 0, 40\right))
\]
A Tour Of Sage, Release 9.2

Chapter 1. Sage as a Calculator
First we create a $500 \times 500$ matrix of random numbers.

```
sage: m = random_matrix(RDF,500)
```

It takes Sage a few seconds to compute the eigenvalues of the matrix and plot them.

```
sage: e = m.eigenvalues()  #about 2 seconds
sage: w = [(i, abs(e[i])) for i in range(len(e))]
sage: show(points(w))
```
Thanks to the GNU Multiprecision Library (GMP), Sage can handle very large numbers, even numbers with millions or billions of digits.

\[
sage: \text{factorial(100)}
9332621544394415268169923888562667004907159682643816214685929638952175999322991560894146397615651828125
\]

\[
sage: n = \text{factorial(1000000)} \quad \#\text{about 2.5 seconds}
\]

This computes at least 100 digits of \(\pi\).

\[
sage: \text{N(pi, digits=100)}
3.\ldots141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825392117067
\]

This asks Sage to factor a polynomial in two variables.

\[
sage: R.<x,y> = \text{QQ[]}
sage: F = \text{factor(x^99 + y^99)}
sage: F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6) * 
(x^10 - x^8*y + x^8*y^2 - x^7*y^3 + x^6*y^4 - x^5*y^5 + 
x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 + y^10) * 
(x^20 + x^19*y - x^17*y^3 - x^16*y^4 + x^14*y^6 + x^13*y^7 -
\]

(continues on next page)
Sage takes just under 5 seconds to compute the numbers of ways to partition one hundred million as a sum of positive integers.

```python
sage: z = Partitions(10^8).cardinality() #about 4.5 seconds
sage: str(z)[:40]
'1760517045946249141360373894679135204009'
```
ACCESSING ALGORITHMS IN SAGE

Whenever you use Sage you are accessing one of the world’s largest collections of open source computational algorithms.