MAT 305: Mathematical Computing

Introduction to Sage

John Perry

University of Southern Mississippi

Fall 2013

Outline

Outline

How to get Sage

- Download, install to your computer
  - latest version at `www.sagemath.org`
  - Windows? need LiveCD or VirtualBox player: `www.virtualbox.org/wiki/Downloads`
  - ask nicely, & I might give you a DVD with Sage for Windows, Mac, Linux
- Lab (SH 205)
  - often locked!
- Online
  - `http://www.sagenb.org/` (Not a USM site)
  - `https://sage.st.usm.edu:8000/`
  - create account
  - can share worksheets with me

First steps in Sage

- Start Sage
- Using web interface? create account, login
  - Don't forget which site or password!

Variables

- *Symbolic* computation: variables
- x pre-defined
  - Sage: unknown symbols give errors
  - some CAS's: unknown symbols $\longrightarrow$ variables
- Need more? use `var()`
  - `var('y')` defines y
  - `var('a b c d')` defines a, b, c, d
- Try to use an undefined variable?

```
sage:  x+y+z
...
NameError:  name 'z' is not defined
```

Arithmetic

operation	sage equivalent
add x, y	`x + y`
subtract y from x	`x - y`
multiply x, y	`x * y`
divide x by y	`x / y`
raise x to the yth power	`x ** y` or `x ^ y`

Arithmetic

operation	sage equivalent
add x, y	`x + y`
subtract y from x	`x - y`
multiply x, y	`x * y`
divide x by y	`x / y`
raise x to the yth power	`x ** y` or `x ^ y`

- Do not omit muliplication symbol
  - `2*x` $\longrightarrow 2x$
  - `2x` $\longrightarrow$ `SyntaxError: invalid syntax`
  - possible, but dangerous, to get around this using `implicit_multiplication(True)`
- Prefer `**` to `^` for various sordid reasons

Example

- Sage simplifies (of course)

```
sage:  5 + 3
8
sage:  (x + 3*x**2) - (2*x - x**2)
4x^2 - x
```

Transcendental numbers, functions

number	sage symbol
e	`e`
π	`pi`

operation	sage equivalent
e^x	`e**x`
$\ln x$	`ln(x)`
$\sin x$, $\cos x$, etc.	`sin(x)`, `cos(x)`, etc.

Transcendental numbers, functions

number	sage symbol
e	`e`
π	`pi`

operation	sage equivalent
e^x	`e**x`
$\ln x$	`ln(x)`
$\sin x$, $\cos x$, etc.	`sin(x)`, `cos(x)`, etc.

- Don't forget to use parentheses when necessary `e**(2*x)` and `e**2*x` are not the same
- `log(x)` $= \ln x \neq \log_{10} x$

Some useful operations

operation	sage equivalent
factor *expr*	`factor(`*expr*`)`
simplify *expr*	`simplify(`*expr*`)`
expand *expr*	`expand(`*expr*`)`
round *expr* to *n* decimal places	`round(`*expr*`, `*n*`)`

Examples

- Some algebraic expressions simplify automatically; others need a hint

```
sage:  (x**2 - 1) / (x - 1)
(x^2 - 1)/(x - 1)
sage:  (factor(x**2 - 1)) / (x - 1)
x + 1
```

- Expand the product $(x-1)\left(x^3+x^2+x+1\right)$

```
sage:  expand((x-1)*(x**3+x**2+x+1))
x^4 - 1
```

- Round e to 5 decimal places

```
sage:  round(e,5)
2.71828
```

LaTeX

- LaTeX?
  - system for laying out documents, created by Leslie Lamport
  - built on TeX, system for typesetting, originally by Donald Knuth

- Sage is LaTeX-friendly!

```
sage:  latex(factor(x^3-1))
{\left(x - 1\right)} {\left(x^{2} + x +
1\right)}
```

- This may look pointless, it makes graphs pretty

Getting help

- Online Sage documentation (tutorial, manual, etc.) at `http://www.sagemath.org/doc/`
- These notes: `www.math.usm.edu/perry/mat305fa13/`
- In-Sage help: command, question mark, `<Enter>`

```
sage:  round?
```

[output omitted]

- Email: `john.perry@usm.edu`

Outline

Expressions

- Use a computer's memory by defining *expressions* with the *assignment symbol* =

```
sage:  f = x**2 - 1
```

  Sage does not answer when you define an expression

- Expressions are remembered until you terminate Sage

```
sage:  f
x^2 - 1
```

Valid names

Names for expressions ("*identifiers*") can

- contain letters (A–Z), digits (0–9), or the underscore (_) *but*
- must begin with a letter or the underscore *and*
- may not contain other character (space, tab, !@#$%^, etc.)

Using expressions

- Manipulate expression in the same way as the mathematical object it represents

```
sage:  factor(f)
(x - 1)*(x + 1)
sage:  f - 3
x^2 - 4
```

- Avoid repeating computations: substitute!

```
sage:  f(x=3)
8
sage:  f(x=-1)
0
sage:  f(x=4)
15
```

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage:  f = x**2 + y**2
sage:  f(x=3)
9 + y^2
```

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage:  f = x**2 + y**2
sage:  f(x=3)
9 + y^2
sage:  z = x
sage:  f(z=3)
x^2 + y^2
```

Here we let z stand in place of x
We want to replace x by 3, but...

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage:  f = x**2 + y**2
sage:  f(x=3)
9 + y^2
sage:  z = x
sage:  f(z=3)
x^2 + y^2
sage:  f({x:3})
9 + y^2
```

Here we let z stand in place of x

We want to replace x by 3, but...

This also means replace x by 3 in f

Alternate method of substitution

Sometimes you should use the **dictionary** method of substitution. An example would be when an identifier stands for a variable.

```
sage:  f = x**2 + y**2
sage:  f(x=3)
9 + y^2
sage:  z = x
sage:  f(z=3)
x^2 + y^2
sage:  f({x:3})
9 + y^2
sage:  f({z:3})
9 + y^2
```

Here we let z stand in place of x
We want to replace x by 3, but...

This also means replace x by 3 in f

This works where `f(z=3)` did not

Expressions as functions

Define function using natural notation

```
sage:  f(x) = x**2
sage:  f(2)
4
```

Expressions as functions

Define function using natural notation

```
sage:  f(x) = x**2
sage:  f(2)
4
```

Automatically defines variables!

```
sage:  f(w,z) = 4*w**2-4*z**2
sage:  f(3,2)
20
sage:  f(1,z)/z
-4*(z**2 - 1)/z
sage:  f(3,2)/z
20/z
```

Expressions as functions

Define function using natural notation

```
sage:  f(x) = x**2
sage:  f(2)
4
```

Functions really expressions

```
sage:  factor(f)
4*(w - z)*(w + z)
sage:  type(f)
<type 'sage.symbolic.expression.Expression'>
```

Outline

Summary

- Basic, intuitive facilities for arithmetic
- Create variables to your heart's content
- Define expressions to avoid repeating computations