Sage, software matemático basado en python que podría interesarte

por Pablo Angulo Ardoy

Misión: Crear una alternativa viable, libre y gratuita, a Magma, Maple, Mathematica y Matlab

Esquema de la charla

- Introducción a Sage
  - ¿Qué es Sage?
  - ¿Para qué sirve?
- Algunos ejemplos
- Caso real: problema con cadenas de Markov
- Conclusiones
  - Comparando con R, ipython, scipy...
  - Comparando con Magma, Maple, Mathematica y Matlab
- Links a mansalva
- Preguntas...

¿Que es Sage?

- Un interprete de python, con un *preparser*
- Una buena coleccion de *librerias científicas y matemáticas*
- Otras cuantas *librerías útiles*
- Código propio, escrito en python y *cython*, de propósito matemático o práctico.
- Una *comunidad*, compuesta sobre todo (a día de hoy) por profesores y estudiantes de Universidad.

python, con un *preparser* [1]

No se modifica el intérprete de python, sólo se convierte el código Sage a código python.

- Convierte los literales `int` a `Integer`, `float` a `RealNumber` (se puede escapar con una `r`):

```
sage: 16.sqrt()
4
sage: preparse('16.sqrt()')
'Integer(16).sqrt()'
sage: a = 3.1415r
sage: a,type(a)
(3.1415, ⟨type⟩'float')
sage: 1/2, (1/2).parent()
(1/2, Rational Field)
```

- Crea variables y expresiones simbólicas:

```
sage: a = 5; f(x,y) = x*y*sqrt(a)
sage: f
(x, y)|--> sqrt(5)*x*y
```

- Otras utilidades para anillos de polinomios, etc

```
sage: preparse("R.⟨x,y⟩ = ZZ['x,y']")
"R = ZZ['x,y']; (x, y,) = R._first_ngens(2)"
```

Paquetes de Matemáticas incluídos en SAGE

Algebra	GAP, Maxima, Singular
Algebraic Geometry	Singular
Arbitrary Precision Arithmetic	GMP, MPFR, MPFI, NTL, MPIR
Arithmetic Geometry	PARI, NTL, mwrank, ecm
(Symbolic) Calculus	Pynac, Maxima, Sympy
Combinatorics	Symmetrica, Sage-Combinat
Linear Algebra	ATLAS, BLAS, Linbox, IML
Linear Programming	GLPK
Graph Theory	NetworkX,
Group Theory	GAP
Number Theory	FLINT
Numerical Linear Algebra	GSL, SciPy, NumPy
Numerical computation	GSL, mpmath, Pynac
Optimization	GLPK, cvxopt

y aún hay más [2]...

Otro software usado en SAGE

Command line	IPython
Compiler	gcc
Cryptography	pycrypto
Database	ZODB, Python Pickles, Sqlite
Documentation	Sphinx
Graphical Interface	Sage Notebook, Sage cell server (MathJax, jquery, codemirror, SockJs, Tornado, IPython web server)
Graphics	Matplotlib, Tachion3d, GD, Jmol, PIL
Interactive programming language	Python
Networking	Twisted (older), Flask (newer)
Version control	Mercurial

y aún hay más [2] [3]...

Código propio

- 600.000+ líneas de código, sobre todo python y cython
- **cython**: fork de pyrex permite compilar código python [4]
  - el código python suele compilar casi sin cambios, y se gana en velocidad casi sin esfuerzo
  - sólo con declarar los tipos de las variables, el aumento de velocidad es importante
  - si tienes tiempo, puedes afinar como harías con un programa en C
- El proceso de desarrollo es *abierto, público y peer-reviewed* [5]
- El código está bastante documentado, y los cambios deben superar toda la batería de *tests automatizados* [6] [7]
- No todo el código es matemático, también cython, sagenb, sage cell server...

Comunidad

- Varias listas de correo [8]
  - **sage-devel** (~500 mensajes/mes)
  - **sage-support** (~200 mensajes/mes)
  - otras listas más específicas con menos tráfico
- **ask.sagemath.org** Sitio web de Q&A à la Stackoverflow: muy útil! (~800 con karma>10, ~90 con karma>100 (permiso para votar negativo)) [9]
- **Sage Days**: encuentros principalmente para desarrolladores (55 hasta el momento). También hay otras jornadas con otras orientaciones (Sage Edu Days, jornadas Sage/python...) [10]
- **trac.sagemath.org** [5] desarrollo público y peer reviewed: ~500 devs
- videotutoriales, libros, tutoriales temáticos, documentación oficial... desarrollados por la comunidad [11]

¿Para qué sirve?

- Una forma sencilla de instalar mucho software cientifico en muchos entornos [12]
  - Instalacion: descomprimir, **make**, y a esperar
  - Funciona en casi todas las distros de Linux, Mac OsX, Solaris
  - Sage para Windows va encapsulado dentro de una máquina virtual!
  - x86, x86_64, arm (casi), itanium
- Software científico desde el navegador
  - Instala en un servidor (si quieres), úsalo en cualquier parte
  - Comparte tus cálculos con tus compañeros
  - Usa el servicio **Sage Cloud** sin instalar nada [13]
  - Aunque ahora tb **ipython** y **rstudio** funcionan como servidores [14] [15]
- Embeber un pequeño cálculo en una web con el **Sage Cell Server** [16]
- Recuerda, es *una alternativa a Magma, Matlab, Maple y Mathematica*...

Algunos ejemplos...

Cambiamos al navegador...

[17]

Cadenas de Markov [18]

- Una cantidad finita de posibles *estados*
- Tiempo discreto
- En cada instante, el estado evoluciona del estado i al estado j con probabilidad p_{ij}

Ejemplos

- Los estados son páginas web, y pasamos de la página i a la página j con probabilidad

$$p_{ij} = \begin{cases} 1/N & \text{si } j \text{ es una de las } N \text{ páginas enlazadas desde } i \\ 0 & \text{si no hay link a } j \text{ en i} \end{cases}$$

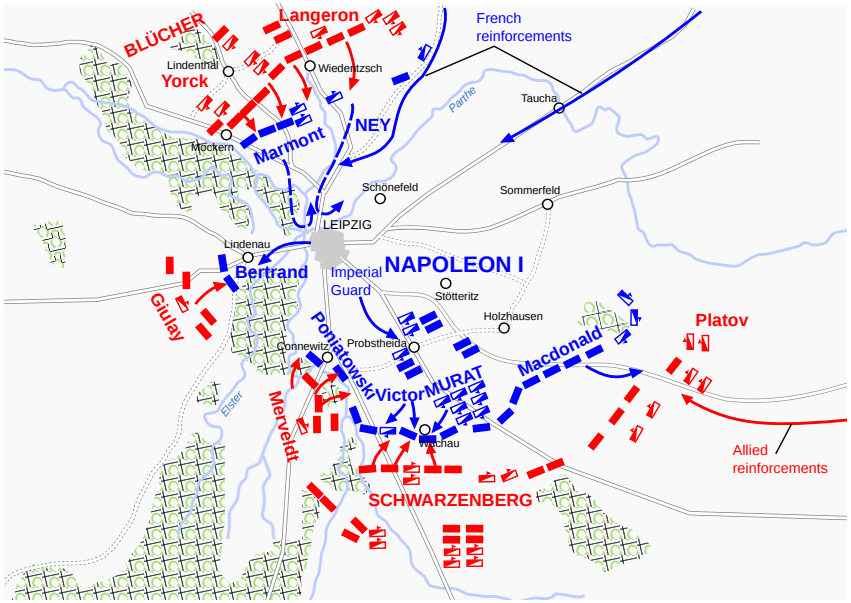
  Este es el modelo más clásico del **random surfer**.

- Los estados son las palabras de un texto. La probabilidad de pasar de una palabra w1 a otra w2 es proporcional al número de veces que w1 va seguida por w2 en el texto.

  Este es el modelo más sencillo de **generación de textos aleatorios**.

Problema de los cuatro generales (pueden ser dos) [19]

- Cuatro generales necesitan coordinar un ataque contra un enemigo común: el objetivo es atacar en la misma fecha
- Sólo pueden enviar mensajes por un canal inseguro: el mensaje llegará con probabilidad p
- ¿Hay algún algoritmo que les permita coordinar su ataque con certeza? La respuesta es **no**.
- Tiene aplicaciones al mundo real... (https://en.wikipedia.org/wiki/Two_Generals%27_Problem)



Estudiamos un algoritmo de coordinacion

- Al principio, cada general ha decidido una fecha de ataque
- Cada noche, un general puede decidir enviar un mensajero a otro general
- Si el mensajero alcanza su destino (con probabilidad p), el general que recibe su mensaje adopta la fecha que le propone el general que lo envía
- Cuando llega el primer mensaje, pasamos de cuatro generales descordinados (1,1,1,1) a un grupo de dos generales coordinados y dos descoordinados (2,1,1)
- El siguiente mensaje podría llegar de un general del grupo grande a uno de los generales descordinados (y pasamos a (3,1)), pero tb de un general descordinado al otro (y pasamos a (2,2)), o de un general descordinado a uno de los que está coordinado (y volvemos a (2,1,1))
- Poniendo probabilidades a estos sucesos, tenemos una cadena de Markov: los estados son (1,1,1,1), (2,1,1), (3,1), (2,2) y (4)
- Pero ¿cuánto tiempo tardarán en ponerse de acuerdo? (en media, con probabilidad mayor que 50%, mayor que 95%....)

...volvemos a Sage [20]

Conclusiones

- Sage permite (a menudo) cambiar de librería científica y de área de las matemáticas sin darte cuenta
- Cuando queremos superar una limitación, a veces hay que investigar qué librería es la responsable (hay herramientas para ésto)

Comparando con R, ipython, scipy...

Desde que ipython y rstudio sacaron sus versiones web, son más fáciles de comparar:

- Sage: hace de todo, pero es fuerte en Combinatoria, Grafos, Álgebra, Cálculo simbólico, Geometría, Ecuaciones Diferenciales.
- ipython+scipy: se pueden añadir paquetes para otras cosas, pero es más fuerte en Cálculo Numérico, Ecuaciones Diferenciales, Ecuaciones en Derivadas Parciales
- Rstudio: el más fuerte (el estándar) en estadística

Sage es interesante si tienes que combinar matemáticas distintas. Todo lo dicho aquí son *opiniones muy personales*

Comparando con Magma, Maple, Mathematica y Matlab

Siendo **software libre** [21], Sage puede ofrecer varias características que M* no puede:

- *Gratis* para tí hoy, gratis para tus alumnos, gratis mañana
- *Modular*, si haces un cálculo interesante, puedes coger las piezas que has usado y crear un producto más ligero e independiente de Sage.
- Uso desde la red, *instalando un servidor*, o usando **Sage Cloud**, y compartir, publicar...
- Puedes *embeber un cálculo en cualquier web*, sin limitaciones

Al usar **python** y otros paquetes de propósito general, tb puede ofrecer [22]:

- Errores ilustrativos (una de las ventajas más repetidas de Sage más repetidas en foros)
  Errors should never pass silently. Unless explicitly silenced.
- Puedes combinarlo con software para cualquier otra cosa (veanse las otras charlas)
- **Cython**: la optimización progresiva se adapta muy bien al cálculo científico
- decoradores, introspección, ast, parsers, debuggers, profilers, conversores a javascript...

Dicho ésto:

- *¿Cuál es más rápido*? Depende, Sage es más rápido a menudo, aunque a veces hay que tunearlo. En los cálculos rutinarios, son iguales (todos usan LAPACK, por ej) [23][24]
- ¿Cuál es más completo? Depende, cada uno hace algo mejor que los demás.

Links

1. pre-parser: http://www.sagemath.org/doc/tutorial/afterword.html?highlight=interpreter#the-pre-parser-differences-between-sage-and-python

2. Paquetes de Matemáticas incluídos en SAGE: http://sagemath.org/packages/standard/

3. Software usado por el Sage Cell Server: https://sagecell.sagemath.org/static/about.html#technology

4. cython: http://cython.org/

5. servidor trac de Sage: http://trac.sagemath.org

6. código fuente de Sage: http://hg.sagemath.org/

7. Sage developer's guide: automated testing: http://www.sagemath.org/doc/developer/conventions.html#automated-testing

8. listas de correo: http://sagemath.org/development.html#mailingList

9. ask.sagemath.org: http://ask.sagemath.org

10. Sage Days y otras jornadas: http://wiki.sagemath.org/Workshops

11. Sage standard documentation: http://www.sagemath.org/help.html#SageStandardDoc

12. Supported Platforms: http://wiki.sagemath.org/SupportedPlatforms

13. Sage Cloud: https://cloud.sagemath.com/

14. ipython notebook: http://ipython.org/ipython-doc/dev/interactive/notebook.html

15. rstudio server: http://www.rstudio.com/ide/download/server

16. Sage cell server: https://sagecell.sagemath.org/static/about.html

17. Ejemplos en malabares.cancamusa.net: https://malabares.cancamusa.net/home/pub/14/

18. Cadenas de Markov: https://en.wikipedia.org/wiki/Markov_chain

19. Problema de los dos generales: https://en.wikipedia.org/wiki/Two_Generals%27_Problem

20. Markov en malabares.cancamusa.net: https://malabares.cancamusa.net/home/pub/12/

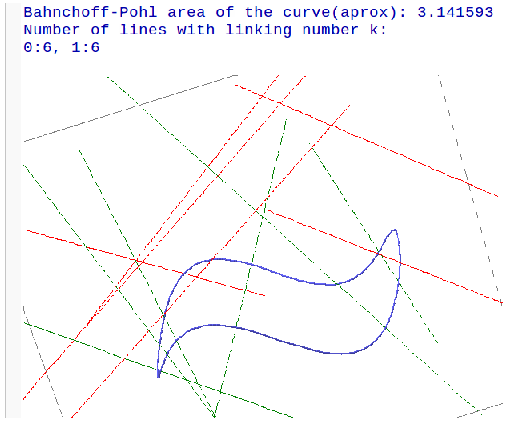
21. Longterm Goals for Sage: http://www.sagemath.org/doc/tutorial/introduction.html#longterm-goals-for-sage

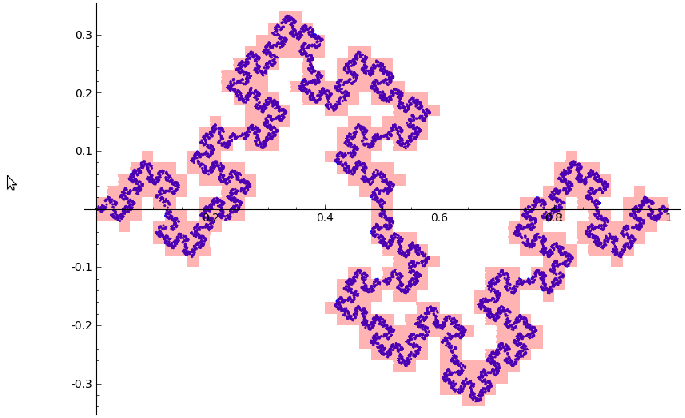
22. why python?: http://www.sagemath.org/doc/tutorial/afterword.html?highlight=interpreter#why-python

23. Sage benchmarks: http://www.sagemath.org/tour-benchmarks.html

24. Performance Python+numpy: http://lbolla.info/blog/2007/04/11/numerical-computing-matlab-vs-pythonnumpyweave

Preguntas por favor...





Texto y ejemplos de Pablo Angulo (ejemplo de cadenas de Markov con Javier Sanz-Cruzado, ejemplo de geodésicas con Antonio Valdés)