useR! 2010 Sage-R Talk

Sage and R: Using R via the Sage notebook

useR! 2010 Focus: Interfaces

Speaker: Karl-Dieter Crisman, Gordon College (MA)

This talk has three purposes:

- Introduce users of R to Sage, another open source (GPL v2+) mathematics software program.
- Show the basics of how to use R through the "notebook interface" of Sage.
- Begin discussing the possibility of deeper interaction between the two communities.

Sage (www.sagemath.org) is a comprehensive mathematics software package, whose general goal is to create "*a viable free open source alternative to Magma, Maple, Mathematica and Matlab*." If this sounds ambitious, it is. Part of the key is broad but high-quality functionality (about which more in a moment).

Sage has just about all of the usual functionality you associate with the various mathematics programs starting with "M".

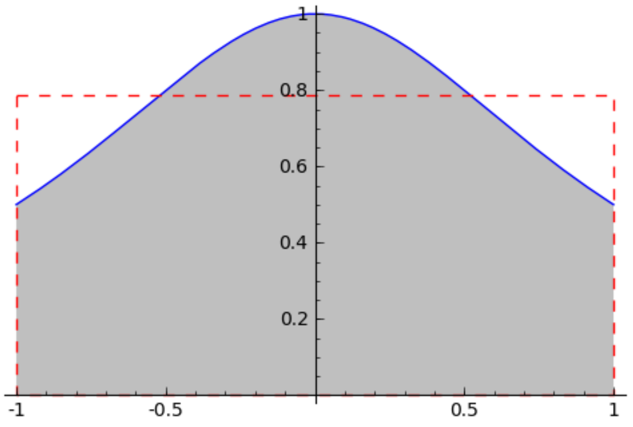
```
integrate(1/(1+x^2),x); show(integrate(1/(1+x^2),x,-1,1))
```

```
arctan(x)
```

$$\frac{1}{2}\pi$$

Maxima gives good symbolic calculus functionality, and the matplotlib Python library gives nice graphics - all wrapped in a unified, easy-to-use interface.

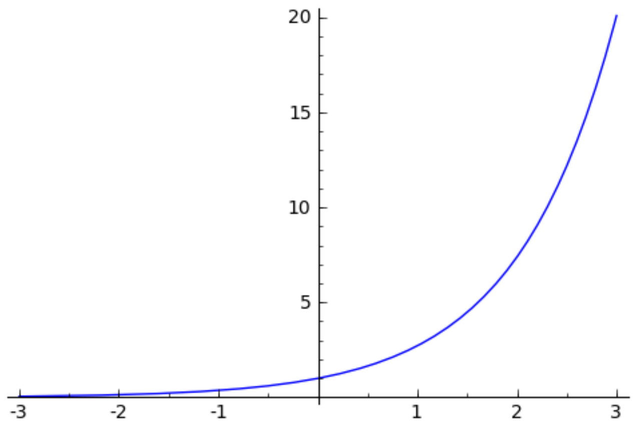
```
plot(1/(1+x^2),(x,-1,1),fill=True)+line([(-
1,pi/4),(1,pi/4),(1,0),(-1,0),(-1,pi/4)],color='red',linestyle="--
")
```



Sage is built on Python, so of course objects persist and we can access object-oriented methods.

```
y = function('y',x)
f = desolve(diff(y,x) - y, y, ics=[0,1])
```

```
plot(f,(x,-3,3))
```



```
show(f.taylor(x,0,7))
```

$$\frac{1}{5040}\,x^{7} + \frac{1}{720}\,x^{6} + \frac{1}{120}\,x^{5} + \frac{1}{24}\,x^{4} + \frac{1}{6}\,x^{3} + \frac{1}{2}\,x^{2} + x + 1$$

This function is, of course, $f(x) = e^x$. Notice that I can incorporate nearly arbitrary LaTeX in the notes, thanks to the jsmath Javascript package, and ask Sage to typeset the answers nicely (this can also be done with the 'Typeset' checkbox at the top).

Sage also has extensive research-level tools, especially in algebraic geometry and number theory. Michael Rubinstein's L-function calculator, for example, is state-of-the-art in such calculations for elliptic curves.

```
lcalc.zeros_in_interval(100,110,0.01)
```

```
    [(101.317851, 0.290553812), (103.725538, 0.115642480), (105.446623
    0.0363238092), (107.168611, -0.190935080)]
```

Notice that Sage has some nice interactive help, just like R, in case one isn't sure what exactly this module does.

```
lcalc?
```

File: /Users/karl-dietercrisman/Desktop/sage-4.4.4/local/lib/python2.6/site-packages/sage/lfunctions/lcalc.py

Type: <class 'sage.lfunctions.lcalc.LCalc'>

Definition: lcalc(args)

Docstring:

Rubinstein's L-functions Calculator

Type `lcalc.[tab]` for a list of useful commands that are implemented using the command line interface, but return objects that make sense in Sage. For each command the possible inputs for the L-function are:

- `''` - (default) the Riemann zeta function
- `'tau'` - the L function of the Ramanujan delta function
- elliptic curve E - where E is an elliptic curve over $\mathbb{Q}$; defines $L(E, s)$

You can also use the complete command-line interface of Rubinstein's L-functions calculations program via this class. Type `lcalc.help()` for a list of commands and how to call them.

This is in addition to many, many lines of new code for a wide variety of purposes.

```
S = SimplicialComplex([1, 3, 7], [[1], [3, 7]]); S
```

```
    Simplicial complex with vertex set (1, 3, 7) and facets {(3, 7),
    (1,)}
```

Pressing [tab] after the dot gives all applicable methods, as one might expect in an object-oriented context

```
s.
```

For much more, including introductions to Sage and its capabilities at various levels, see www.sagemath.org.

When it comes to state-of-the-art open source answers for statistics, R is in a class of its own, so Sage includes R as a standard package!

It is possible to access R directly from a Sage installation through from a shell via the command `sage -R`, or inside a Sage command line instance via `r_console()`. There is also a library-level interface called "rpy" (now "rpy2") many of you may be familiar with, which is included in Sage.

However, we will focus here on using R (directly or indirectly) from the wonderful "notebook" interface this talk is in. Most of this talk can be run from any web browser anywhere in the world, as long as you can find a public Sage server to do it from (such as www.sagenb.org), though for a few things we will need optional R packages to be installed. This actual talk is located at http://www.sagenb.org/home/pub/2270, and you can create an account and edit a copy as we speak!

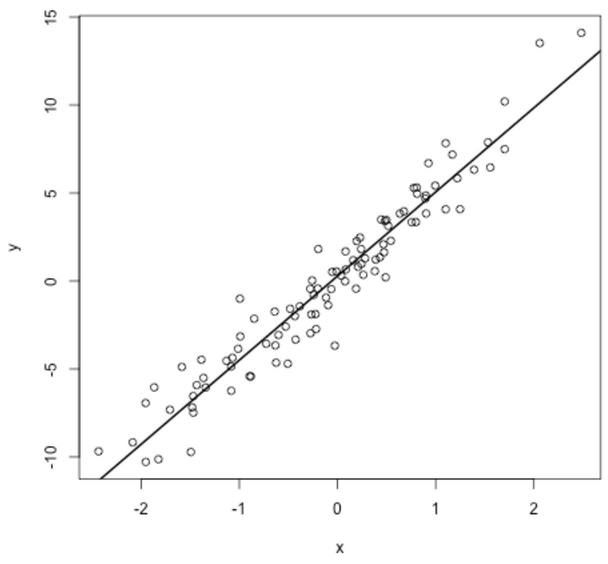
There are two main ways to access R in the notebook. The more obvious is to put a "percent directive" in a cell and just use normal R commands. This first example is reminiscent of examples in John Verzani's introductory statistics with R book.

```
%r
library("MASS")
data(Cars93)
mean(Cars93$MPG.city)
xtabs( ~ Origin + MPG.city, data=Cars93)
```

```
[1] 22.36559
         MPG.city
Origin    15 16 17 18 19 20 21 22 23 24 25 26 28 29 30 31 32 33 39
42 46
  USA      2  3  4  5  8  3  3  4  7  2  2  0  1  2  0  2  0  0  0
0  0
  non-USA  0  0  4  7  2  5  3  3  1  3  4  2  1  4  1  0  1  1  1
1  1
```

This example is due to Andrzej Giniewicz.

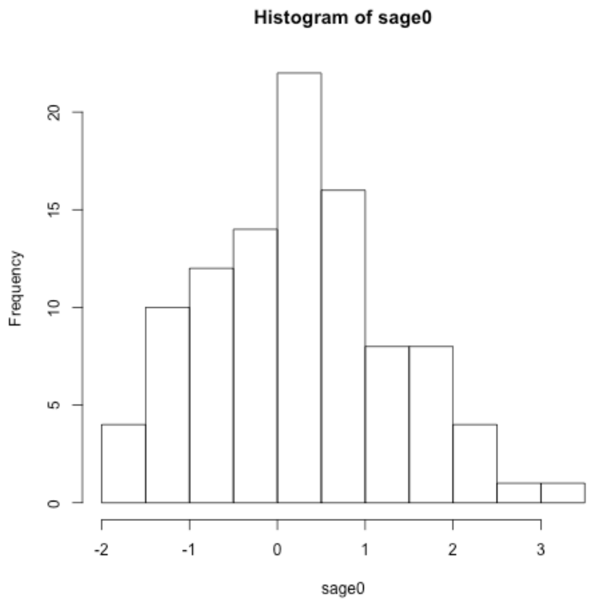
```
%r
x <- rnorm(100)
y <- rnorm(100, mean=5)*x+rnorm(100, mean=0.1)
png()
plot(y ~ x)
abline(lm(y ~ x), lwd=2)
.silent.me. <- dev.off()
```

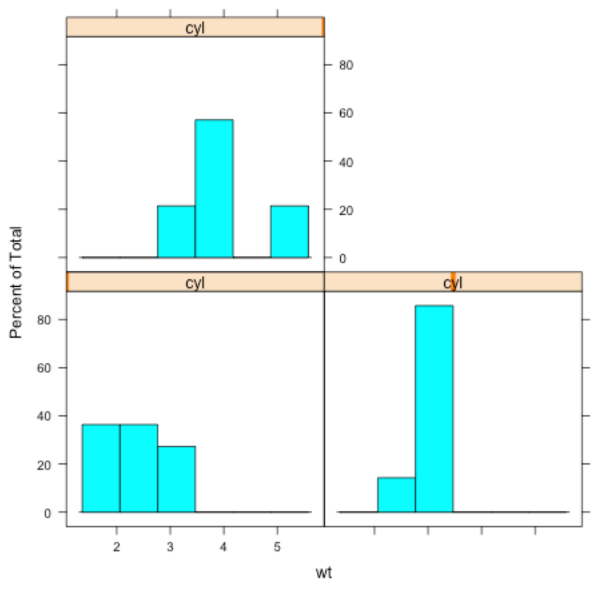


However, one can also access much of R through a regular Sage computation, using the R object and Pythonic dot-notation. Notice that the r.png() is NOT the usual R png function, but one that should use Quartz on Mac.

```
r.png()
x = r.rnorm(100)
r.hist(x)
r.library("lattice")
r("print(histogram(~wt | cyl, data=mtcars))") ## need print for
lattice graphics
r.histogram(x = "~ wt | cyl", data="mtcars") ## doesn't work, needs
to be "printed"
r.dev_off()
```

```
    null device 
              1 
```



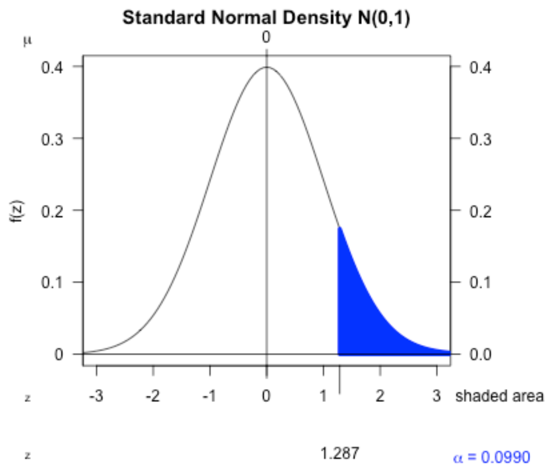


.Rplot002.png-0Vfo

Richard Heiberger helped me find the R command that creates this one.

```
# needs HH package installed
r.library('HH')
@interact
def _(alpha=slider(.001,.999,.001,.1)):
    r.png()
    r.normal_and_t_dist(alpha_right="%s"%alpha)
    r.dev_off()
```

alpha 0.0990000000000001



There are three things you can see with these example.

- In general, things work great!

    - I can even create an interactive 'Sagelet' quite easily.
    - On a pedagogical level, one could use these various interfaces to weave calculus and statistics together, or at the very least to have a way to minimize overhead for undergraduates.
    - Or, one could choose 'r' from a drop-down menu at the top (which currently reads 'sage') and never even know one was in the Sage notebook.
- Sage's R includes the recommended packages, but you'll have to download others on your own (though this is easy).
- Not everything works seamlessly yet; there is room for contribution! For example:
    - The random normal data set is just called 'sage0', the internal name of the variable, and right now this is annoying (though possible) to change.
    - It's difficult to get certain commands to run properly with options without just evaluating an R command directly.
        - (However, the 'print' issue is documented in the lattice graphics package and is an R issue, not Sage.)

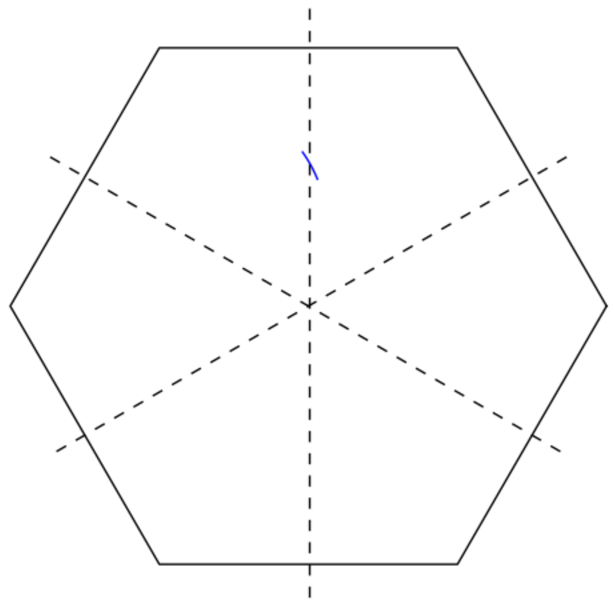
These are, of course, really just examples of using R through the Sage notebook, not really using Sage and R together. What about interactions?

The next example uses the only easily accessible open-source implementations of computing the so-called mediancentre of a data set - both of which are in R optional packages! - and then exploit Sage's graphics and interactive capabilities to visualize something not provided for in those packages (in this case, the outcome of hypothetical voting procedures).

```
@interact
def _(a=matrix([[100,100,300,0,0,200]]),iterations=(10,[1..100])):
    ShowStableMCBC(a.list(),iter=iterations)
```

a 100 100 300 0 0 200

iterations 10



The code for this is below. The important part isn't the code itself, but rather the following points about the integration of Sage and R. For many of you, this will be the main point of the talk!

- There is a tremendous amount of useful mathematics in R which I can use in very non-statistics settings.

- I could use an easy-to-learn, mainstream programming language (Python) to prototype and run this code.
    - If I wanted to speed it up, it would be relatively easy to convert this to Cython, which creates pure C code; I didn't need this.
    - It is not hard to use your own C++ or C libraries and so on as well, perhaps in a way analogous to Rcpp.
- It was easy to use Sage to combine various combinatorics and normal graphics I needed to do this.
- I can share this worksheet very easily with a colleague or student so they can run it too and expand on it.
    - It is also possible to integrate my Sage code into a LaTeX document using `sagetex`, which is (as far as I can tell) *very* similar in purpose to Sweave for R.

```
%auto
# This is a piece of a module for using Sage and R to compute and
graph mediancentre Borda Count and
# related things.  This is a prerelease version, not suitable for
distribution but suitable for
# computation.  It needs more documentation and probably some
optimization/cleaning up.

##########################################################################

#  Copyright (C) 2010 Karl-Dieter Crisman <karl.crisman@gordon.edu>
##########################################################################


# One must first install the optional R packages 'depth' and
'orloca' to use this module. This can
# be done from the Sage command line with
# sage: r.install_package('depth')
# sage: r.install_package('orloca')
# It can be useful to install a custom 'depth' package for using
algorithm='depth'; see R
# documentation for how to install from your own local computer.
The reason is that
# 'depth' has some weird hangs for certain input data if we don't
change source slightly for
# depth.f in the R package 'depth', to add a few things for another
iterator, namely changing things
# so that
#      integer
l(n),m,n,ifault,i,nn,j,k,nt,j1,j2,itdone,itdon2,maxit
# and
#   12 continue
#      itdon2=itdon2+1
#      if (itdon2.gt.maxit) then
#         ifault=-2
#         return
#      endif
#      if (nn .gt. 0) goto 3

# TODO: Lots more documentation, in ReST and Python docstring
format!

# Here we install the packages
r.library('orloca')
r.library('depth')
npi = n(pi) # or math.pi is probably even better, can optimize all
this a lot

# The following three functions get the mediancentre picture.

def MCBC(prof,algorithm='w'): # the function that gets the
mediancentres from a profile
    # typical profile=[0,0,0,0,0,0] - they will (for now) have six
entries
    if prof.count(0)==4: # Remove '2-split' profiles and
arbitrarily make them complete tie
        return (RR(0),RR(0))
    for k in range(6): # Check for mediancentre at a vertex as in
Footnote 14
        S = sqrt(3)/2*prof[mod(k+1,6)]+1/2*prof[mod(k+2,6)]-
1/2*prof[mod(k+4,6)]-
sqrt(3)/2*prof[mod(k+5,6)]+I*(1/2*(prof[mod(k+1,6)]+prof[mod(k+5,6)])+prof[mod(k+3,6)]+sqrt(3)/2*(prof[mod(k+2,6)]+pr
# add unit vectors in directions of other vertices
        if abs(S).n()<=prof[k]: # in this case, the point IS the
mediancentre
            return (RR(cos(k*npi/3)),RR(sin(k*npi/3)))
    exes=[]
    whys=[]
    if algorithm in ['w','g']: # Weiszfeldt or gradient methods
        for k in range(6):
            exes += [cos(k*npi/3)]
            whys += [sin(k*npi/3)]
        # Z = r.loca_p(exes,whys,prof) # third is weights
        # median = Z.zsummin()._sage_()
        Z =
r.eval('zsummin(loca.p(c(%s),c(%s),c(%s)),max.iter=400,algorithm="\%s")'%(str(exes)[1:-
1],str(whys)[1:-1],str(prof)[1:-1],algorithm))
        Z = Z.split()
        return (RR(Z[1]),RR(Z[2]))
    if algorithm in ['depth','depth2']:
        for k in range(6):
            exes += prof[k]*[cos(k*npi/3)]
            whys += prof[k]*[sin(k*npi/3)]
        # Z = r.data_frame(r(exes),r(whys))
        # median = r.med(Z,method="Spatial") # for some reason this
won't work!
        if algorithm=='depth':
            Z =
r.eval('med(matrix(c(%s),ncol=2),maxit=400,method="Spatial")'%str(exes+whys)[1:-
1]) # Note optional arguments maxit=200 and eps=1e-8 can be changed
        if algorithm=='depth2':
            r.eval('spamed=function(x){list(median =
.Fortran("medctr78", as.numeric(x), med = numeric(length(x[1, ])),
as.integer(length(x[, 1])), as.integer(length(x[1, ])), integer(1),
err = integer(1), PACKAGE = "depth")$med)}')
            Z =
r.eval('spamed(matrix(c(%s),ncol=2))'%str(exes+whys)[1:-1])
        Z = Z.splitlines()[1].split() # this returns only the [1] 1
0 part of the string as a list of strings
        return (RR(Z[1]),RR(Z[2])) # do something like this to
account for floating point error, though certainly this is one of
the most naive things one could do

# The following function is *very* preliminary for investigating a
"stable" MCBC.

def ShowStableMCBC(prof, algorithm='w',iter=100):
    G = Graphics()
    hex = line([(cos(k*pi/3),sin(k*pi/3)) for k in
range(7)],color='black')+sum([line([(cos(k*pi/3+pi/6),sin(k*pi/3+pi/6)),(cos(k*pi/3+7*pi/6),sin(k*pi/3+7*pi/6))],colo
-') for k in range(3)])
    G+=hex
    points_list=[]
    points_list.append(MCBC(prof,algorithm))
    prof = [item+1 for item in prof]
    points_list.append(MCBC(prof,algorithm))
    #while points_list[-1]!=points_list[-2]:
    for i in range(iter):
        prof = [item+1 for item in prof]
        points_list.append(MCBC(prof,algorithm))
    G+=line(points_list)
    G.show(aspect_ratio=1,axes=False)
```

The utility of R to Sage, and the immense benefit of being able to use advanced statistics packages along with robust calculus, number theory, and so forth within a nice GUI with interactive capabilities should be clear.

However, in what ways can the main Sage project be useful to the R project?

- In some sense, R doesn't 'need' Sage:
    - It has a large and stable user and developer base.
    - It is far more connected to those who use mathematics, as opposed to mathematicians.
    - It has a HUGE number of optional packages to do a lot of stuff.
- But Sage can be a resource to R too, I think. Some naive ideas:
    - One way to resolve the periodic requests on R-help for things like symbolic integration and the like.
    - Simplifying pedagogical issues.
    - The notebook - a way to use R without downloading anything. (This doesn't mean it's bad if R develops its own such interface!)
    - Providing easy access to complicated combinatorics or group theory as needed (such as in Diaconis' book!).

```
SymmetricGroupRepresentation([3,1,1],"orthogonal")
```

```
Orthogonal representation of the symmetric group corresponding to [3, 1, 1]
```

Thank you for listening; I look forward to the interaction time in a bit. Special thanks to:

- The organizers for accepting my abstract
- John Verzani for encouraging me to submit an abstract
- NSF and NIST for making it feasible to attend
- The Sage community for being excited about improving R functionality in Sage

This talk is available on the Internet at http://www.sagenb.org/home/pub/2270.