REALLY MINIMAL .C()

JAN DE LEEUW

The function

```
double hypot (double x, double y)
```

is part of the standard C library. On your Mac OS X system this is the file /usr/lib/libSystem.dylib. The function hypot() takes doubles x and y as arguments and returns a double equal to $\sqrt{x^2 + y^2}$. If you want to know more, go to the terminal and say

```
man hypot
```

We now show how to make hypot() a part of R, using the .C() interface. The .C() interface only handles C functions that do not explicitly return a result ("return a void") and take only pointers as arguments ("pass by reference"). Thus we need a little C wrapper that replaces hypot() by a functions satisfying these requirements.

```
#include <math.h>

void
hypotC (double *x, double *y, double *z) {
    *z = hypot (*x, *y);
}
```

Let's put this code in a file hypot.c.

Date: Friday 13th April, 2012 — 12h 18min — Typeset in UTOPIA.

We now go back to the terminal, move to the directory where `hypot.c` sits, and say

```
R CMD SHLIB hypot.c
```

The terminal responds

```
/usr/bin/gcc -std=gnu99 -I/usr/local/R/lib/R/include -DNDEBUG  -I/usr/local/include    -fPIC  -m64 -O3 -Wall -fopenmp -mtune=native  -c hypot.c -o hypot.o
/usr/bin/gcc -std=gnu99 -dynamiclib -Wl,-headerpad_max_install_names -undefined dynamic_lookup -single_module -multiply_defined suppress -lgomp -m64 -o hypot.so hypot.o -L/usr/local/R/lib/R/lib -lR -Wl,-framework -Wl,CoreFoundation
```

This assumes you have a `C` compiler installed. It is also very likely your output looks different, because your OS, compiler, and setup will be different from mine, but if you do not see errors and a file `hypot.so` is created in your directory, you are probably OK.

The file `hypot.so` is a shared library that contains the compiled code of `hypot.c`, as well as the information from the `C` library and runtime that is needed. To see what is linked into `hypot.so` you can say

```
otool -L hypot.so
```

which will tell you something like

```
hypot.so:
    hypot.so (compatibility version 0.0.0, current version 0.0.0)
    libR.dylib (compatibility version 2.16.0, current version 2.16.0)
```

```
	/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation (compatibility version 150.0.0, current version 635.21.0)
	/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 159.1.0)
```

To see which symbols are defined in `hypot.so` say

```
nm hypot.so
```

which results in

```
                 U _hypot
0000000000001f20 T _hypotC
                 U dyld_stub_binder
```

You see, from the `T` in the second column, that `hypot.so` actually contains the code for `hypotC`. It does not contain the code for `hypot`, but it knows where to find it in the library `/usr/lib/libSystem.B.dylib`.

We write a second wrapper, now in `R`, and put it in `hypot.R`.

```
hypot <- function (x, y) {
    return (.C ("hypotC", as.double (x), as.double (y), as.double (0))[[3]])
}
```

Now start R, which in my case runs in the terminal, and say

```
> dyn.load("hypot.so")
> source("hypot.R")
```

The first command makes the compiled code part of R, the second part makes the code in `hypot.R` available, and thus defines the R function `hypot`. We can now tell R to use it in the same way as any R function is used

```
> hypot(3,4)
[1] 5
> hypot(100,1)
[1] 100.005
> hypot(1,1,1)
Error in hypot(1, 1, 1) : unused argument(s) (1)
> hypot(1)
Error in hypot(1) : argument "y" is missing, with no default
```

DEPARTMENT OF STATISTICS, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90095-1554

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

URL, Jan de Leeuw: http://gifi.stat.ucla.edu