Linking R and Fortran

In statistical programming, sometimes using lower level subroutines in C or Fortran can provide better computational efficacy. However, the lack of random number generator could be troublesome for these users. As R provides a wide selection of random number generator, It's a good idea to adopt them in some of the C or Fortran code.

This note will focus on using R functions in your Fortran code and some useful source codes are provided. This note is Linux/Unix only, however, on Windows, you can do it in a similar way.

1. Calling Fortran functions from R

a. Write the Fortran Code.

I borrowed the Fortran code for calculating Fibonacci sequence on this website (http://rosettacode.org/wiki/Fibonacci_sequence - Iterative_15). I named this code as fib.f90.

```
INTEGER FUNCTION fib(n)
      integer n
      integer, parameter :: fib0 = 0, fib1 = 1
      integer back1, back2, i

      select case (n)
          case (:0);      fib = fib0
          case (1);       fib = fib1

          case default
              fib = fib1
              back1 = fib0
              do i = 2, n
                  back2 = back1
                  back1 = fib
                  fib   = back1 + back2
              end do
       end select
END FUNCTION
```

b. Write a wrapper for the Fortran Code.

As R can only call Fortran subroutines, sometimes writing a wrap is necessary. In the example, I wrote the following wrapper function:

```
SUBROUTINE fib_R_wrapper(n, answer)
       INTEGER n, answer, fib
       EXTERNAL fib
       answer = fib(n)
END SUBROUTINE
```

c. Compile the Fortran code for R as a shared library

You cannot use Fortran code directly in R. However, you can compile it as shared library. There are two ways to do this: the simplest way is to use *R CMD SHLIB* command.

```
R CMD SHLIB fib.f90
```

This command automatically does two things with properly complier: compile the source to the object file *.o; make the shared object *.so from the object file *.o. This command could be done manually (take intel compile on Linux as an example):

```
ifort -c -fPIC fib.f90
ifort -shared -o fib.so fib.o
```

Once we get the .so file. We can use that in R.

d. Call Fortran from R

```
n=10
.Fortran('fib_R_wrapper', n=as.integer(n), result=integer(1))$result
for(n in 1:10)
print(.Fortran('fib_R_wrapper', n=as.integer(n),
result=integer(1))$result)
```

Now, you can use .Fortran function in R to call the Fortran code:

As you see from the code, the Fortran function is returned by the name of the parameters.

We are able to call fortran subroutines in R now. But some R functions are useful in Fortran.

2. Calling R functions from Fortran

a. Write C wrapper for Fortran

A C wrapper for Fortran is required to cover platform-specific differences. An example is given for using the normal distribution and uniform distribution:

```
#include <R.h>

void F77_SUB(rndstart)(void) { GetRNGstate(); }
void F77_SUB(rndend)(void) { PutRNGstate(); }
double F77_SUB(normrnd)(void) { return norm_rand(); }
double F77_SUB(unifrnd)(void) { return runif(0, 1); }
```

b. Write Fortran code to call the functions

After setting up the C code, the Fortran code can call the functions:

```
subroutine test_random(x, y)
real*8 normrnd, unifrnd, x, y
      call rndstart()
      x = normrnd()
      y = unifrnd()
      call rndend()
      return
end
```

c. Compile the C and Fortran code

Compiling the code is critically:

```
R CMD SHLIB -o norm.so norm.c normf.f90
```

d. Use the Fortran code in R

```
###test normal random generator####
dyn.load('norm.so')
.Fortran("test_random", as.double(1), as.double(1))
dyn.unload('norm.so')
```

Reference:
http://stat.ethz.ch/R-manual/R-devel/library/utils/html/SHLIB.html
http://cran.r-project.org/doc/manuals/R-exts.html
http://math.acadiau.ca/ACMMaC/howtos/Fortran_R.html
http://rosettacode.org/wiki/Fibonacci_sequence