Test (medium)

Alessandro De Bettin

March 22, 2016

My knowledge on C integration within R is limited; I know how to write, compile and call from R C functions, but I don't know how to integrate the procedure in an R package. I had planned to study the subject before finishing the tests, but the hard test took longer than expected. However, if I get accepted, I'll study all that is needed before the beginning of the programming period.

1 Example

Let's say a function for calculating the product of 2 vectors is needed. Being z and w the two equally n long vectors, the operation we wish to do is

$$z^T w.$$

A C function that does that is

```
#include<stdio.h>
#include<R.h>

void VecProd(double *ris, double *z, double *w, int *n){
  int i;
  *ris=0.0;
  for(i=0; i<*n; i++)
    *ris += z[i]*w[i];
}
```

The first two commands are needed to make the R/C interface possible. As for all of C functions that will be called form R, the output is void and the inputs are pointers.
After the compiling of the C code, an R wrapper function is needed.

```
dyn.load("VecProd.so")

VecProdR <- function(z,w){
  n <- length(z)
  out <- .C("VecProd",risR = double(1), zR = as.double(z),
   wR = as.double(w), nR = as.integer(n))
  out$risR
}
```

The first function loads the compiled C code in the R environment (the .so file must be in R's current working directory). The wrapper function calls the C function via the .C() command. In this way, we can use C code just by utilizing the wrapper function, making things easier. This procedure can make things way faster, in particular when applied to algorithms which use many for cycles, or many vector concatenations.