The .C Interface I

- say we have the following C function which does “useful” stuff (it lives in file prog1.c):

```
#include <R.h>

/*
 * Lots of stuff for bulding the interface for the
 * typedef MonteCarloSpecs
 */

void
do_stuff (int *n_iters, double *time_in_secs, double *prop_burn_in)
{
        MonteCarloSpecs *mcs = NULL;

        mcs = mcs_new(*n_iters, *time_in_secs, *prop_burn_in);
        mcs_print(mcs);
        mcs_free(&mcs);
}
```

- say we want to use the functionality of this function from within R

The .C Interface II

- points to note about the `do_stuff` function:
  - `C` functions called by `R` must all return void, which means they need to return the results of the computation in their arguments
  - all arguments passed to the `C` function are passed by reference, which means we pass a pointer to a number or array and so be *very careful* while dereferencing
  - each file containing `C` code to be called by `R` should include the `R.h` header file: `#include <R.h>`
  - if you are using special functions (e.g. random number generation functions or distribution functions of `R`), you need to include the `Rmath.h` header file: `#include <Rmath.h>` (details on this coming later!)

The .C Interface III

- we do either of:
  - `R CMD SHLIB -o NameForFooLibrary.so foo.c`
  - `R CMD SHLIB foo.c`
- say we used the last option then we would get a gift from `R`: `prog1.so` (and a by-product `prog1.o`)
- now lets use this this “shared object” or the contents of `prog1.so`
- under `Windows` you will get `prog1.dll` instead of `prog1.so`

The .C Interface IV

- use the following to mingle C and R (this lives in a file called prog1.R):

```
dyn.load("prog1.so")

doStuff <-
    function (nIters, timeInSecs, propBurnIn)
{
    .C("do_stuff",
       as.integer(nIters),
       as.numeric(timeInSecs),
       as.numeric(propBurnIn))
}

doStuff(100, 10, 0.1)
```

- all the files prog1.c, prog1.so and prog1.R (preferably) should be in the same directory
- under Windows you will say dyn.load("prog1.dll") instead of dyn.load("prog1.so")

Code Files

```
prog1.c
prog1.R
```