Foreign Language Interfaces
Self-Study Exercises

Valerie Obenchain

Fred Hutchinson Cancer Research Center

17-18 February, 2011

1 Introduction

These excercises will take you though the steps necessary to include compiled C code in your course package. We will be using a C function that computes linkage disequilibrium. For this course we are using simulated snp data and do not have the haplotype phase. Because of this our C function computes a 'composite' linkage disequlibrium which is a statistical measure of association across loci when haplotype phase is not known. See Weir and Cockerham, 1989; Weir 1996 pp. 94, 125.

We will compiling the C code and loading it into an *R* session for testing. Next the package NAMESPACE file will be modified so the dynamic library will be automatically loaded when the package is loaded. The final step is registering the C method with the *R* code.

Exercise 1
In this excercise we compile the C code into a shared object. The shared object is then loaded into an R sesesion for testing.

Question 1

- *Move the C function composite_linkage_disequilibrium.c into the src directory of your package. Compile the C function using R CMD SHLIB*

Solution:

```
R CMD SHLIB composite_linkage_disequlibrium.c
```

Question 2

- *Before loading the shared object in the package, you may want to perform some basic testing. Any shared object can be loaded into R by using the* `dyn.load` *command. All functions in the shared object are now made availble to R. Start an Rsession and load the shared library with* `dyn.load`*.*

- *Call the composite linkage disequlibrium function with .C*

Solution:

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

snps <- matrix(sample((0:2), replace=TRUE), nrow=100, ncol=4)
nsnp <- ncol(snps)
nsub <- nrow(snps)
width <- 3
delta <- rep.int(0, (nsnp-width)*width)
out <- .C("composite_linkage_disequilibrium",
          snp = as.integer(snp),
          n_ind = as.integer(nind),
          n_snp = as.integer(nsnp),
          width = as.integer(width),
          delta = as.double(delta))
```

Question 3
Create a more convienent interface to the linkage disequlibrium function by writing an R wrapper function. Call the function `cld.R` and put it in the /R directory of your package.

Solution:

```
> cld <- function(data, width = 5)
+ {
+     nsnp <- ncol(data)
+     nind <- nrow(data)
+     delta <- rep.int(0, nsnp*width)
+     if (width < nsnp)
+         stop("Width must be less than the number of snps.")
+     .C("composite_linkage_disequilibrium",
+        snp = as.integer(snp),
+        n_ind = as.integer(nind),
+        n_snp = as.integer(nsnp),
+        width = as.integer(width),
+        delta = as.double(delta))
+ }
```

Exercise 2
In this excercise we modify the package NAMESPACE file to automatically load the shared library when the package is loaded.

Question 4
- *Modify the NAMESPACE to load the dynamic library*

Solution:

```
useDynLib{StudentGWAS}
```

Exercise 3
The purpose of this excercise is to register the C function with R. We will create an initialization fuction and register the linkage disequlibrium function there.

When a DLL is loaded, *R* looks for a routine within that DLL named R_init_lib where lib is the name of the DLL. If such a routine is present, *R* will invoke it. This is a convienent way of executing some code automatically when an object/DLL is loaded or unloaded. We use this function to register native routines with *R*'s dynamic symbol mechanism.

When .C, .Call or .Fortran is used, *R* must locate the specified native routine by looking in the shared obejct/DLL. Registering a native routine with *R* allows the use of a platform-independent mechanism for finding the routines in the DLL instead of an operating system-specific method to lookup the routine. The registration mechanism can also be used to make the routine available to *R* programmers under a different name.

To register routines with *R* we use the C routine R_registerRoutines. It takes 5 arguments, the first is the DLL information followed by arrays describing the routines for each of the 4 different interfaces: .C, .Call, .Fortran and .External. Each argument is NULL-terminated array of the element types given in the following table :

```
.C     R_CMethodDef
.Call     R_CallMethodDef
.Fortran     R_FortranMethodDef
```

Question 5

- *Create a file called R_init_StudentGWAS.c and put it in /inst/src*
- *Define the linkage disequlibrium using R_CMethodDef*
- *Register the function using R_registerMethod*

Solution:

```
 /* R_init_StudentGASE.c file */

 #include <R.h>
 #include <Rinternals.h>
 #include <R\_ext/Rdynload.h>

/* Create the R\_CMethodDef array */
```

```
 R_CMethodDef CMethods[]  = {
 {"composite_linkage_disequlibrium", (DL_FUNC) &composite_linkage_disequibrium, 5,
    {INTSXP, INTSXP, INTSXP, INTSXP, REALSXP},
    {NULL, NULL, 0}
 };

/* Register the routine */

 void R_init_StudentGWAS(DllInfo *info){

   R_registerRoutines(info, CMethods, NULL, NULL, NULL);
   return;
 }
```