R's Foreign Function Interface (FFI)

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2013

Outline

1 General

2 Review of Return Concept

3 .C and .Call

- .C
- .Call

Outline

1 General

2 Review of Return Concept

3 .C and .Call

- .C
- .Call

Why bother with code written in C, C++, Fortran, or Java, or ...

- One of the most thorough discussions of the foreign function interfaces is found in *R Programming for Bioinformatics* (2009), by one of the R origininators, Robert Gentleman.
- Gentleman discusses 2 reasons for using compiled code through R.
  - Programs in C (C++, Fortran) may be faster
  - Programs (really, libraries, algorithms, etc) exist in C (C++, Fortran) and can be put to use from R.

About Speed

- Most authors emphasize the speed of calculations in C or Fortran
- Counter-arguments
  - Gentleman's opinion (2009): The sheer speed (reduction in run time) not usually a compelling reason to use foreign functions.
  - Claims:
    - Well written R code can be fast
    - Much faster to write an R program that works than a really fast C program that's complicated

About Speed

- Suggestions:
  - Write it in R, at least for a prototype (Knuth "premature optimization is the root of all evil").
  - Profile the code, find out where the slowdown might be, look for algorithmic accelerations within R
  - If necessary, can re-write to push some calculations to C, C++, etc.
- Several well known programmers have expressed this same view to me directly (John Nash (author R optim and *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*, 2ed, 1990).

Don't Get Carried Away, though...

- I'd still rather have a program written entirely in C (or C++, Objective-C, or Fortran), if it works dependably, than a program written in R. I feel certain it will be faster
- But
  - That's a really BIG IF, and
  - The time required to write a program in R will be 1/2 or 1/3 of time to write a program in C (for me, at least)
  - My co-authors don't know much C, but do know R.
- For example, Martyn Plummer's JAGS program is written in C++, not R.

If Not For Speed, then Why the FFI?

- Use existing programming libraries, which are written in C, C++, Fortran, ...
- 100s of optimizations and 1000s of tests have been applied against Famous C libraries like
  - Atlas
  - GotoBLAS2

What this Lecture is NOT About

- Some R packages masquerade as usages of the foreign function interface.
- They follow this approach:
  - Write a text file of program syntax
  - Use system commands to call a compiler on that syntax
  - Run the program in a shell, write results on disk
  - Use R to harvest the results from the disk file
- Examples: OpenBUGS (BUGS code), SabreR (Fortran), MPlusAutomation

What this Lecture is About

- The Foreign Function Interface
- Shared library approaches that allow R to use functions written in other languages
- Exemplified in the R functions .C, .Call, .Fortran

Outline

1 General

2 Review of Return Concept

3 .C and .Call

- .C
- .Call

Return by Value versus Return by Reference

- Return by Value: do calculations on copies of input variables, don't allow changes in those input variables, return results to user as "new" thing
- Return by Reference: input variables are pointers, allow the function to dereference values and change them at the memory location.
- Recall R: heavy preference for "return by value".
  - Arguments into an R function are "local copies". They cannot be altered.
  - R design strongly prefers we return results as new objects that are created in the last line of each function.

C Allows both

- Elementary C is taught with "return by value"

```
int myFunction();
```

  means the value coming out will be one integer

```
double myFunction();
```

  or one double with real number
- Return by value recommended for any C function that returns one thing
- Understanding of "return by reference" requires
  - conceptual understanding of pointers
  - caution!

Return by Value in C

```
int myFunc (double x, double y){
   // local copies of x and y are created
   // calculations using x and y, may change them
   return z;
}
int myRes1; double g1, k1;
myRes1 = myFunc(g1, k1);
```

The only change observed is the value of myRes1.

g1 and k1 “go into” myFunc, but they are not affected by it.

Return by Reference in C

```
void myFunc (double x, double y, double *z){
   // x and y are still local. But *z is a memory pointer
   // *z "dereferences" the value pointed at by z, and changes it
   *z = x + y;
}
double g1, k1; double * m1;
myFunc(g1, k1, m1);
```

- There's no formal return
- g1 and k1 "go into" myFunc, but they are not affected by it.
- The value pointed to by m1 IS changed by myFunc

Almost All Famous C Programs use Return by Reference

- BLAS “Basic Linear Algebra Subprograms” . Interfaces & implementations in Fortran, C, etc. The interface (http://www.netlib.org/blas/blast-forum)
- Use return by value where possible.
  - calculate an inner “dot” product of vectors pointed to by X and Y.

```
double cblas_ddot(const int N, const double *X, const int incX,
    const double *Y, const int incY);
```

Explain: N: number of elements in both *X and *Y. incX and incY are set to 1, almost always.

Almost All Famous C Programs use Return by Reference ...

- BLAS: multiply matrices:
  I worked my heart out, lets examine my SVN-guides repository in folder:
  `c-programming/Examples/cblas-examples`
- Note in examples that the interface to those BLAS functions is quite complicated.
- GSL abstracts that somewhat. Offers a Matrix "struct" to avoid some details.
- GSL: return matrix results, pass by reference:
  - GSL: subtract 2 matrices: a - b

```
int gsl_matrix_sub (gsl_matrix * a, const gsl_matrix * b)
```

Almost All Famous C Programs use Return by Reference ...

- Matrix subtraction $a - b$, the result is put into a, b remains unchanged.
- The "int" return indicates success or failure of the calculation.

Outline

1 General

2 Review of Return Concept

3 .C and .Call
- .C
- .Call

.C and .Call

Why 2 interfaces?

Why do some people claim .C is discouraged?

Why do the experts recommend we use .Call?

.C can work with a pre-existing C function "as is"

- Programs that R can access through .C do not have a "rich interaction" between R and C.
- The C function must have
  - (void) return type
  - Arguments must be pointer variables
- R passes C some pointers, C writes results there.
- The .C call it returns an R list with "copies" of the variables.

.C Examples

- In my SVN-guides, look in the folders:
  `c-programming/Examples/FromR-dotC-1`
  And
  `c-programming/Examples/FromR-dotC-2`
- Note how we have to use as.integer() and as.double() to prepare R variables to be passed as pointers to C.
- The results come back as a list of “revised arguments” variables. We better step through the examples to see...

Translating Variables

Gentleman(2009) p. 187

R	C
logical	int *
integer	int *
double	double *
single	single *
complex	Rcomplex *
character	char **
raw	char*
list	SEXP
other	SEXP

- C provides built in types int, double, char
- Typedefs for Rcomplex and SEXP found in Rinternals.h

.C Sales Pitch

- If we pass integers, doubles, and characters, we don't need to revise the C code much, if at all.
- As long as the function can create a shared library, its all easy.

Use GSL Routines in C, via R

In my SVN-guides, look in the folder:
`c-programming/Examples/FromR-CallGSL-dotC`

.C Usage Examples

R packages in CRAN: MNP (case study below)

James Lindsey R packages (supporting books such as Models for Repeated Measurements.

`http://www.commanster.eu/rcode.html`. Consider the R package "repeated", for example:
`http://www.commanster.eu/rcode/repeated.tgz`

Why Some Folks discourage .C

- No easy "error checking"
- C code doesn't use R idioms or structures
- Dangers discussed in .C help page on duplication
- Missing and other non-numeric variables.

Why MNP

- It is code you might actually understand: very clear coding, no nonsense naming etc
- The fitted model is relevant
- We see the strengths and weaknesses of C as a way of life. This one creates a vector storage structure and random number generation from scratch

When I install that, What Do I See?

```
> install.packages("MNP", repos="http://rweb.quant.ku.edu/cran")
Installing package into '/home/pauljohn/R/x86_64-pc-linux-gnu-library
    /3.0'
(as 'lib' is unspecified)
trying URL 'http://rweb.quant.ku.edu/cran/src/contrib/
    MNP_2.6-4.tar.gz'
Content type 'application/x-gzip' length 974626 bytes (951 Kb)
opened URL
==================================================
downloaded 951 Kb

* installing *source* package 'MNP' ...
** package 'MNP' successfully unpacked and MD5 sums checked
** libs
gcc -std=gnu99 -I/usr/share/R/include -DNDEBUG      -fpic  -O3 -pipe
    -g  -c MNP.c -o MNP.o
gcc -std=gnu99 -I/usr/share/R/include -DNDEBUG      -fpic  -O3 -pipe
    -g  -c rand.c -o rand.o
gcc -std=gnu99 -I/usr/share/R/include -DNDEBUG      -fpic  -O3 -pipe
    -g  -c subroutines.c -o subroutines.o
gcc -std=gnu99 -I/usr/share/R/include -DNDEBUG      -fpic  -O3 -pipe
    -g  -c vector.c -o vector.o
gcc -std=gnu99 -shared -o MNP.so MNP.o rand.o subroutines.o vector.o
    -llapack -lblas -lgfortran -lm -lquadmath -L/usr/lib/R/lib -lR
installing to /home/pauljohn/R/x86_64-pc-linux-gnu-library/3.0/MNP/
    libs
** R
```

When I install that, What Do I See? ...

```
** data
*** moving datasets to lazyload DB
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (MNP)
```

Check that in the package insall directory

In the install directory for the package, I have...

```
$ pwd
/home/pauljohn/R/x86_64-pc-linux-gnu-library/3.0
$ ls MNP/libs/
MNP.so
```

- That's a dynamically loadable C library,

MNP Source code

- Get the source code (download.packages("MNP", type = "SOURCE", dest = "/tmp"). I grabbed MNP_2.6-4.tar.gz on 2013-12-02.
- Note the folders:

  R: The R code
  src: the C source code

Check the file onAttach.R

```
".onAttach" ← function(lib, pkg) {
  mylib ← dirname(system.file(package = pkg))
  title ← packageDescription(pkg, lib.loc = mylib)$Title
  ver ← packageDescription(pkg, lib.loc = mylib)$Version
  author ← packageDescription(pkg, lib.loc = mylib)$Author
  packageStartupMessage(pkg, ": ", title, "\nVersion: ", ver, "\
      nAuthors: ", author, "\n")
}
```

- When the user runs library(MNP) (or require(MNP)), the first thing it does is create 5 variables,
  - "mylib" is the value of the location where the package is installed.
  - It uses that to get the title & author information displayed in packageStartupMessage

Check the file onAttach.R ...

- Note "dirname(system.file(package = "MNP"))" is a way to ask your running R session where it is finding the MNP installed folder.

Check the package NAMESPACE file

- The first line is

```
useDynLib(MNP)
```

Check the R source for the function mnp()

- The function “mnp” is doing all of the heavy lifting.
- In the file mnp.R , find line 152:

```
param ← .C("cMNPgibbs", as.integer(n.dim),
  as.integer(n.cov), as.integer(n.obs), as.integer(n.draws),
  as.double(p.mean), as.double(p.prec), as.integer(p.df),
  as.double(p.scale*p.alpha0), as.double(X), as.integer(Y),
  as.double(coef.start), as.double(cov.start),
  as.integer(p.imp), as.integer(invcdf),
  as.integer(burnin), as.integer(keep), as.integer(trace),
  as.integer(verbose), as.integer(MoP), as.integer(latent),
  pdStore = double(n.par*floor((n.draws-burnin)/keep)),
  PACKAGE="MNP")$pdStore

param ← matrix(param, ncol = n.par,
  nrow = floor((n.draws-burnin)/keep), byrow=TRUE)
```

- Boom! There it is. A thing param is returned, and
- matrix() is used to grab the right rows and columns out if it.

C File Inventory

Makevars : A Makefile that controls how the C files are compiled. In this case, there is only a miniscule entry

- The C files:

```
$ ls
Makevars   rand.c   subroutines.c   vector.c
MNP.c      rand.h   subroutines.h   vector.h
```

C File Inventory

Makevars : A Makefile that controls how the C files are compiled. In this case, there is only a miniscule entry

```
$ cat Makevars
 PKG_LIBS = $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)
```

MNP.c : This is the orchestrating file, where the key functions are accessed from R. The functions are

C File Inventory ...

```
void cMNPgibbs(int *piNDim, int *piNCov, int *piNSamp, int *piNGen,
  double *b0,     /* prior mean for beta */
  double *pdA0, int *piNu0, double *pdS, double *pdX,
  int *y,         /* response variable: -1 for missing */
  double *pdbeta, double *pdSigma, int *piImp,
  int *invcdf,    /* use inverse cdf for TruncNorm? */
  int *piBurnin, /* the number of burnin */
  int *piKeep,
  int *itrace,
  int *verbose,   /* 1 if extra print is needed */
  int *piMoP,     /* 1 if Multinomial ordered Probit */
  int *latent,    /* 1 if W is stored */
  double *pdStore)

void predict(double *dX,      /* X matrix */
  int *nobs,        /* number of observations */
  double *dcoef,    /* coefficients */
  double *dSigma,   /* covariances */
  int *ndims,       /* number of dimensions */
  int *ncovs,       /* number of covariates */
  int *ndraws,      /* number of MCMC draws */
  int *moredraws,   /* number of extra draws */
  int *verbose,
  double *prob,     /* probability output */
  double *choice,   /* choice output */
  double *order   /* order output */)
```

C File Inventory ...

- note, both of these are "return by reference" approaches.

rand.[hc] : h is the header, c is the code. The header file declares 4 functions, there's nothing except for the function prototypes

```
double sTruncNorm(double bd, double mu, double var, int
     lower);
double TruncNorm(double lb, double ub, double mu,
     double var, int invcdf);
void rMVN(double *Sample, double *mean, double **
     inv_Var, int size);
void rWish(double **Sample, double **S, int df, int
     size);
```

- Note significance of **X, which is, basically, a pointer to one corner of a two-dimensional storage area

C File Inventory ...

- Whereas *X is a pointer to the beginning of a one-dimensional storage area

vector.[hc] : Allocates storage for vectors and matrices!

```
#include <stdlib.h>
#include <assert.h>

int *intArray(int num);
void PintArray(int *ivector, int length);
int **intMatrix(int row, int col);
void PintMatrix(int **imatrix, int row, int col);

double *doubleArray(int num);
void PdoubleArray(double *dvector, int length);
double **doubleMatrix(int row, int col);
void PdoubleMatrix(double **dmatrix, int row, int col);

double ***doubleMatrix3D(int x, int y, int z);
void PdoubleMatrix3D(double ***dmatrix3D, int x, int y,
      int z);

long *longArray(int num);

void FreeMatrix(double **Matrix, int row);
void FreeintMatrix(int **Matrix, int row);
```

C File Inventory ...

```
void Free3DMatrix(double ***Matrix, int index, int row)
    ;
```

- Here we have VERY CLEARLY named functions, a style worth admiring.
- Functions to create and initialize {integer, double} arrays or matrices
  - The function doubleArray allocates a memory and returns a POINTER to the beginning of it.
  - If you want to “Print” that to the screen, use the PdoubleArray function.

C File Inventory ...

- Read the vector.c file and you notice that the print-to-screen work is being done by Rprintf, a function from the R C library. (hence the file includes the header R.h. THe top of vector.c has

```
#include <stdlib.h>
#include <assert.h>
#include <stdio.h>
#include <R_ext/Utils.h>
#include <R.h>
```

- Strictly speaking,
  - I think vector.h SHOULD be included here, I suppose the compiler might assume it. But all working C code I know of would include vector.h at the top of vector.c.

C File Inventory ...

  - "stdlib.h" and "assert.h" need not be included in vector.c since it was included in vector.h (assuming vector.h was included here).
- Note the Free functions to erase a vector or matrix when no longer needed. Vital to stop memory leaks.

.Call

Makevars : A Makefile that controls how the C files are compiled. In this case, there is only a miniscule entry

```
$ cat Makevars
PKG_LIBS = $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)
```

.Call

- .Call is used widely in the R source code
- Uses return by value, NOT return by reference
- Asks user to re-write C code, interacting more meaningfully with R objects using R-provided variable types and functions.

Benefits of using .Call, as opposed to .C

- C code is interacting with data structures in same way that R internal C code does
- Better checking of argument types
- For this purpose, R offers an elaborate collection of
  - C variable types
  - C functions

Garbage Collection: Why this is Dicey

- When you use .Call, your C code is running "inside" the R memory zone framework
- That means that the garbage collector might kill your constructed variables
- Note the PROTECT and UNPROTECT macros in this example from *Writing R Extensions*

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

SEXP convolve2(SEXP a, SEXP b)
{
    int na, nb, nab;
    double *xa, *xb, *xab;
    SEXP ab;

    a = PROTECT(coerceVector(a, REALSXP));
    b = PROTECT(coerceVector(b, REALSXP));
    na = length(a); nb = length(b); nab = na + nb - 1;
    ab = PROTECT(allocVector(REALSXP, nab));
```

Garbage Collection: Why this is Dicey ...

```
    xa = REAL(a); xb = REAL(b); xab = REAL(ab);
    for(int i = 0; i < nab; i++) xab[i] = 0.0;
    for(int i = 0; i < na; i++)
        for(int j = 0; j < nb; j++) xab[i + j] += xa[i] * xb[j];
    UNPROTECT(3);
    return ab;
}
```

- You'd compile that into a shared object, then inside R you could call like so

```
conv <- function(a, b) .Call("convolve2", a, b)
```

Rcpp as an Magic Bullett

- The difficulty of writing that special C motivated Dirk Eddelbuettel and Romain Francois to work very hard a developing a style of C++ coding that can be more-or-less automagically gobbled into R via the R package Rcpp.
- I'll write notes on that in the lecture folder ffi-3.

Background information 1: Use R library functions from a C Program

In my Guides repo (or http://pj.freefaculty.org/guides)

`c-programming/Examples/FromC-CallRmathlib`

Does not meaningfully interact with R, only uses some C functions that the R team has made available.

Background information 2: Using R embedded in a C Program

In my Guides repo (or http://pj.freefaculty.org/guides) FromC-RunRembedded

`c-programming/Examples/c-programming/Examples/FromC-RunRembedded`

Starts an R session AND actually interacts with it.

Can Generalize previous to use R functions in R.h

In the installed R, there should be a header folder that has R C function interfaces

Search for "R.h"

The same folder has subfolder "R_ext", which has more header files that are listed in R.h

R.h (headers in R_ext) provides R "replacements" for basic C functions

Example: printf can be replaced by Rprintf (See R_ext/Print.h)

.Call

.Call