NOAA Technical Memorandum NMFS-SEFSC-547

User's Guide to C2R: A Set of C Language Output Routines Compatible with the R Statistics Language

Andi Stephens
Michael H. Prager
Jennifer L. Martin

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Marine Fisheries Service
Southeast Fisheries Science Center
75 Virginia Beach Drive
Miami, Florida 33149

October, 2006
Revised April, 2007
Software version 1.01

NOAA Technical Memorandum NMFS-SEFSC-547

User's Guide to C2R: A Set of C Language Output Routines Compatible with the R Statistics Language

Andi Stephens
Michael H. Prager
Southeast Fisheries Science Center
Beaufort, North Carolina

Jennifer L. Martin
Northeast Fisheries Science Center
Woods Hole, Massachusetts

U. S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Oceanic and Atmospheric Administration
Conrad C. Lautenbacher, Jr., Under Secretary for Oceans and Atmosphere

National Marine Fisheries Service
William T. Hogarth, Assistant Administrator for Fisheries

Revised April, 2007
Software version 1.01

Notice of nonendorsement

The National Marine Fisheries Service (NMFS) does not approve, recommend or endorse any proprietary product or material mentioned in this publication. No reference shall be made to NMFS, or to this publication furnished by NMFS, in any advertising or sales promotion which would imply that NMFS approves, recommends, or endorses any proprietary product or proprietary material mentioned herein which has as its purpose any intent to cause directly or indirectly the advertised product to be used or purchased because of this NMFS publication.

Suggested citation

Stephens, A., M. H. Prager, and J. L. Martin. 2006. User's guide to C2R: A set of C language output routines compatible with the R statistics language. U.S. Department of Commerce, NOAA Technical Memorandum NMFS–SEFSC–547, iv + 21 p.

Contacting the authors

The authors can be contacted by email at Mike.Prager@noaa.gov and Jennifer.Martin@noaa.gov and will endeavor to provide support to users.

Revision history

January, 2006	Distributed for internal SEFSC use
October, 2006	Released as NOAA Technical Memorandum

Obtaining this document and software

Primary distribution of this document and software will be online. The document can be obtained from the publications entry of the Southeast Fisheries Science Center Web site,

http://www.sefsc.noaa.gov/

The document *and software* will be submitted to the R Comprehensive Archive Network (CRAN) for distribution,

http://cran.r-project.org/

Paper copies are best obtained by dowloading and printing the PDF file. Copies will also be available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161, telephone (703) 605-6000.

Contents

1 Introduction

1.1 Overview of C2R

C2R is a collection of C routines for saving complex data structures into a file that can be read in the R statistics environment with a single command.[1] C2R provides both the means to transfer data structures significantly more complex than simple tables, and an archive mechanism to store data for future reference.

We developed this software because we write and run computationally intensive numerical models in Fortran, C++, and AD Model Builder. We then analyse results with R. We desired to automate data transfer to speed diagnostics during working-group meetings.

We thus developed the C2R interface to write an R data object (of type `list`) to a plain-text file. The master list can contain any number of matrices, values, dataframes, vectors or lists, all of which can be read into R with a single call to the `dget` function. This allows easy transfer of structured data from compiled models to R.

Having the capacity to transfer model data, metadata, and results has sharply reduced the time spent on diagnostics, and at the same time, our diagnostic capabilities have improved tremendously. The simplicity of this interface and the capabilities of R have enabled us to automate graph and table creation for formal reports. Finally, the persistent storage in files makes it easier to treat model results in analyses or meta-analyses devised months—or even years—later.

We offer C2R to others in the hope that they will find it useful.

Please note that C2R is considered an experimental product by NOAA and is released to the scientific community for testing and research purposes. Neither the U.S. government nor the authors make any warranty of correct operation.

The X2R interface is available in three forms: for Fortran as For2R, for C/C++ as C2R, and for AD Model Builder as ADMB2R. This guide specifically describes the C2R interface. The other packages, including documentation, are available from the authors or from CRAN.

1.2 The C2R package

The C2R package includes the following files—

- This guide, file `c2r-guide.pdf`
- The C2R source code, in C header file `C2R.h`
- A driver program,`test-c2r.c` demonstrating use of C2R routines
- The output file, `test-c2r.rdat`, generated by compiling and running the driver program

Because the C++ programming language is a superset of C, the C2R routines can be called from C++ as well as C.

[1] Mention of commercial or noncommercial products does not imply endorsement by NOAA, US Department of Commerce, or any other government agency. No such endorsement is made or implied.

1.3 R in brief

The R statistics environment (R Development Core Team 2004) is a free, open-source programmable statistics system implemented as a dialect of the S language. R offers modern statistics and excellent graphics, which are controlled from a command line, by programming, or from one of several graphical interfaces. R can be downloaded from the Comprehensive R Archive Network (CRAN) or its mirrors, e. g., from

http://cran.r-project.org/

All CRAN mirrors contain links to the R Project home page and to R documentation (much available for free download) at several levels of complexity. Among commercially available books, an introductory text is provided by Dalgaard (2002). More extensive treatments, still at an introductory level, are given by Maindonald and Braun (2003) and Verzani (2005). Two widely used reference books are Venables and Ripley (2003) and Venables and Ripley (2000).

1.4 Reporting problems

The authors have tested C2R with three compilers (Fujitsu C, version 3.0; gnu gcc, version 3.4.2; and Borland bcc32, version 5.5.1). We found no problem under any of those compilers. Still, it is impossible to test with every compiler.

The authors will greatly appreciate receiving reports of any bugs found, so that they can be corrected. We will also attempt to include user improvements or extensions. Such information can be sent to Mike.Prager@noaa.gov.

1.5 C2R and FishGraph

The authors have developed a series of R graphics functions that produce typical graphs of fisheries stock-assessment model output. We have used them for several years in our assessment work. The FishGraph functions are currently available in beta-test versions from M.H. Prager's Web site, http://www.sefsc.noaa.gov/mprager/. We anticipate making them available on the CRAN archive at some later date.

Each FishGraph function takes an argument that is an R list, making FishGraph highly compatible and easy to use with X2R. The required structure of that list, described in the FishGraph manual, allows for extensive user expansion or customization.

By using X2R to save model results and FishGraph to generate graphs, it is possible to produce hundreds of diagnostic plots in seconds. The plots are saved automatically as files for use in reports.

FishGraph is not a formal R package, but rather a series of R functions that we use in our work. We offer it to colleagues to use as is or to modify for their own needs.

2 Data structures in C2R

2.1 Data storage and types

Output from C2R is stored as an R list object in a structured ASCII file readable by R with a `dget` function call. An R list is a container object that holds other data items. Each component of a list is named, and subcomponents (e. g., the rows and columns of a contained matrix) may be named as well.

If output from C2R is stored in (for example) a file named `test.rdat`, it can be read into R as a list named `results` with the single R function call

```
results = dget("test.rdat")
```

Then the resulting R list object will contain the data saved by the C2R calls, along with corresponding object names, metadata, and data structures specified by the user. Much of the usefulness of C2R is that the files it creates may contain complicated data structures, and yet are read with a single statement.

The following data types may be components of the master list:

- **Comment.** A subroutine is provided for writing R comments to the output file.
- **Info list.** An info list in C2R is a list of character strings in name–value pairs, intended for storing metadata such as the analyst's name, units used in calculations, etc. A date–time stamp can be included automatically.
- **Vector** of real or integer numbers or character strings. An R vector may be used to store a series of name–value pairs, such as scalars from a model run, or an entire vector at once, such as output from or input to a modeling run.
- **Matrix.** A two-dimensional array of real or integer values.
- **Data frame.** The R data frame is like the "dataset" of some statistics packages: a set of samples (stored as rows) on different variables (stored as columns). C2R supports giving meaningful column names and, optionally, row names, to data frames.
- **List.** A list may contain any number of other data objects, such as vectors and lists.

Like most statistics software, R supports the concept of missing (unobserved) values in data objects. C2R supports writing missing values to its output file in R-compatible form.

2.2 Writing data in C2R

The following C code fragment opens a file, and writes a master R list object containing (1) the special info object, (2) a vector, (3) a matrix, (4) a list of two matrices, and (5) a dataframe object. The final call terminates the master list and closes the file.

For clarity, calls here are shown without arguments. Details of calls, including required and optional arguments, are given in following sections by object type.

```
#include "C2R.h"
open_r_file(...);
   open_r_info_list(...);
      wrt_r_item(...);
      ...
   close_r_info_list(...);
   open_r_vector(...);
      wrt_r_item(...);
      ...
   wrt_r_item(...,"LAST",...);
   wrt_r_matrix(...);
   open_r_list(...);
      wrt_r_matrix(...);
      wrt_r_matrix(...);
   close_r_list(...);
   open_r_df(...);
      wrt_r_df_col(...);
      ...
   wrt_r_df_col(...,"LAST",...);
close_r_file();
```

Note that files and info and list objects are opened and closed by corresponding open and close routines. In contrast, closure of a vector or data frame object is signalled by the LAST flag. A matrix is opened, written, and closed with a single call.

3 Usage considerations

3.1 Precision

To allow maximum flexibility in use of C2R, real numbers are formatted to the user's specified level of precision — the number of digits after the decimal — up to the maximum allowed by the compiler.

3.2 Object names

The C character array used to store the names of R objects can accept names up to 128 characters long. This can be increased by editing the source. No tests are made to ensure that names written are useable in R. Legal R names may contain upper- and lower-case letters, digits, and the period (dot) character. Names should not begin with digits. R also allows the underscore in names, but we advise against it; in some versions of the S language, the underscore is an assignment operator. In summary, the user is responsible for choosing suitable names for data objects.

3.3 Data types

At present, support for data types is limited. There is no support yet for complex numbers, and character data are supported in only some places.

3.4 Missing values

The missing value flag in R is the string NA. To indicate the position of missing values in the data array or matrix being written, C2R uses a logical array the same size and shape as the data. The value TRUE indicates that C2R should write NA rather than the value from the data array to the output file.

C2R defines the logical values TRUE and FALSE for clarity:

```
#ifndef TRUE
   #define TRUE (1==1)
   #define FALSE (1==0)
#endif
```

EXAMPLE

Data array:	`[1, 0, 0]`
Missing value array:	`[FALSE, TRUE, FALSE]`
Output produced:	`[1, NA, 0]`

3.5 Error checking

Basic error checking, for example of argument type, is provided by the compiler.

Other errors are possible, and C2R itself checks for some of them. However, many unchecked errors are possible in writing code. For example, if the C2R routines are called out of order, it is possible to create a file that cannot be parsed correctly by R. This will also occur when the LAST flag is not used to signal the final element in a sequence.

4 Specifications

4.1 Open and close output file

```
open_r_file( char *filename, int maxlevel, int maxcomp, int maxdigits );
close_r_file();
```

The output file is opened and the master `list` object it contains is initialized with subroutine `open_r_file`. Only one file may be open at a time.

The object is finalized and the open file closed with subroutine `close_r_file`. These two subroutines must be the first and last calls in the sequence that writes an R object.

4.1.1 Arguments

Argument	Default	Required	Description
`filename`	none	Required	Name of file to be written
`maxlevel`	128	Optional	Maximum number of nesting levels of objects in the master `list`
`maxcomp`	128	Optional	Maximum number of subcomponents an object may contain
`maxdigits`	6	Optional	Precision of floating-point number—digits after the decimal.

4.1.2 Example

```
open_r_file("test.rdat", 0, 0, 0);
close_r_file();
```

This sequence opens the file `test.rdat` and accepts the defaults for `maxlevel`, `maxcomp`, and `maxdigits`. It then closes the file, which contains an empty master `list` object.

4.2 Info list

```
open_r_info_list( char *name );
   wrt_r_info_item( char *name, char *value );
close_r_info_list( char *name, char *value );
```

The info list object constains a series of name–value pairs (list components) stored as character strings. The current date and time (with name `Date`) can be stored automatically as the first component. At least one additional component is required. The number of name–value pairs is limited by the `maxcomp` value with which the file was opened (§4.1 on p. 7).

The object is initialized with the routine `open_r_info_list`, and data elements are written to it with `wrt_r_info_item`. The final name–value pair is written and the info list closed with a call to `close_r_info_list`.

4.2.1 Arguments

Argument	Description
`name`	Name of info element
`value`	Associated value

4.2.2 Example

```
open_r_info_list("info");
   wrt_r_info_item("Analyst", "R. A. Fisher");
   wrt_r_info_item("units.length", "mm");
   wrt_r_info_item("units.mass", "kg");
close_r_info_list("run.title", title_string);
```

4.3 Vector object

```
open_r_vector( char *name );
wrt_r_item( char *name, char *spec, value);
```

In C2R, a vector object is an unordered collection of named numeric (real or integer) or character string values. An ordered collection of values is better represented as a column of a dataframe object.

There is no requirement that vectors be of uniform type; however, when real and integer values are written to a single vector, the integers will be converted to real numbers when R reads the file. Similarly, if a vector contains character strings as well as numeric values, R will interpret all values as character strings.

4.3.1 Arguments

Argument	Description
name	Name of vector, name of vector element.
spec	String indicating type of item and any output options. Options are— %s value is a string %d value is an integer %f value is a real number (float) NA Write the missing value indicator, NA. LAST Item is the final element of the vector.
value	Value, which must be of the type flagged. This argument may be omitted when NA flag is used.

The value of spec is formulated as follows: first, one of the formats string (%s, %d, or %f) or the indicator flag NA. This is optionally followed by one or more spaces and the indicator flag LAST. The LAST flag is *required* to indicate the final element of a vector.

4.3.2 Example

```
open_r_vector("Parameters");
   wrt_r_item("Sex", "%d", 1);
   wrt_r_item("Age", "%d", 37);
   wrt_r_item("Lung Capacity", "NA");
   wrt_r_item("Resting Pulse", "%d", 73);
wrt_r_item("Temperature", "%f LAST", 98.6);
```

When read by R, the data generated by this example becomes a vector of real numbers, because one element is noninteger (type %f) and no elements are character strings.

4.3.3 Note on Literal Subscripting in R

When using the resulting object in R, components can be selected using literal subscripting. For example, consider the vector written above and named `Parameters`. Once the master object containing this vector has been read into R and named (e. g.) `result`, the `Age` value in the vector can be assigned to a variable as follows :

```
age <- result$Parameters["Age"]
```

4.4 Data frame object

```
open_r_df( name );
wrt_r_df_col( char *name, char *flags, int length,
   value, <optional arguments> );
```

A data frame in R is a collection of data columns (vectors) of equal length. If the vectors in the user's program are not of equal length, the program must reshape the data so that all calls to `wrt_r_df_col` transmit data of equal length when a data frame is written. Columns of a data frame may contain different data types.

Each column of a data frame carries a name. If row names are given (they are optional), they refer to all columns of the data frame. Either character strings or integer numeric values may be used as names.

4.4.1 Arguments

Arguments to `wrt_r_df_col` must be supplied in the proper order:

(1) Name of the column
(2) String indicating type of array and any output options.
(3) Length of the value array
(4) Array of values of the type flagged
(5) NA array, if flagged
(6) Row labels, if flagged

The value array must be of type int*, double*, or char**, as specified in the flags string.

4.4.2 Option Flags

`NA`	A logical array signifying missing values (`TRUE = NA`) is provided. The NA array must be the same length as the value array.
`Ri` or `Rn`	Integer row labels (`Ri`) or strings (row names, `Rn`) are supplied. Allowed only when the LAST flag is present.
`LAST`	This column is the final one in the data frame.

4.4.3 Example

```
open_r_df("Timeseries");
   wrt_r_df_col("year", "%d", 3, years);
   wrt_r_df_col("sector", "%s", 3, sector);
   wrt_r_df_col("gross", "%f NA", 3, gross, NA_vector);
wrt_r_df_col("pred.net", "%f LAST Ri", 3, net, quarter);
```

The example constructs a data frame with four columns. The first contains integer values; the second, character strings; the third and fourth, real (floating-point) numbers. Column three includes some missing values, which are represented in the output by NA. Row names are derived from integers.

The final call includes the LAST flag. This causes the data frame object to be closed properly and the row labels to be written.

4.5 Matrix object

```
wrt_r_matrix( char *name, char *flags, int nrows, int ncols,
   value, <optional arguments> );
```

C2R initializes, writes and closes a two-dimensional matrix in one call. Elements of the matrix may be floating-point or integer values.

In order for matrix indexing to work in a compiler-independent manner, a matrix object to be written by C2R must be constructed as a two-dimensional array of pointers. Code for creating such an array is shown in the example following the description of the call and arguments.

4.5.1 Arguments

Arguments to `wrt_r_matrix` must be supplied in order:

(1) Name of the matrix
(2) String indicating the type of matrix and any output optionals.
(3) Number of rows in the value matrix
(4) Number of columns in the value matrix
(5) Matrix of pointers to values of the type flagged
(6) NA matrix, if flagged
(7) Row labels, if flagged
(8) Column labels, if flagged

The matrix must be of type int**, or double**, as specified in the flags string.

4.5.2 Options Flags

`NA`	A logical matrix signifying missing values (`TRUE = NA`) is provided. The NA matrix must be the same size and shape as the value matrix.
`Ri` or `Rn`	Integer row labels (`Ri`) or strings (Row names, `Rn`) are supplied.
`Ci` or `Cn`	Integer column labels (`Ci`) or character strings (`Cn`) are supplied.

4.5.3 EXAMPLE

In this example, a 3×5 matrix of floating point values is written. Code is given immediately below to illustrate construction of such a matrix.

```
wrt_r_matrix("Three.by.five", "%f NA Rn Ci", 3, 5, fmatrix, na,
   rownames, colids);
```

Construction of matrix— Before using the above example, it is necessary to construct the data matrix and the corresponding missing-value matrix. First, a 3×5 matrix of pointers (here named `fmatrix`) is created to store floating-point values. In this example, it is initialized to zero, but in user code it can be set equal to the data values.

```
double **fmatrix[3];
for (i=0; i < 3; i++);{
   fmatrix[i] = (double *)malloc(5 * sizeof(double));
   for (j=0, j < 5; j++);{
      fmatrix[i][j] = 0.0;
   }
}
```

Then, a same-sized matrix of pointers is created to store logical values, initialized to `FALSE`. In this example, one value is changed to `TRUE` to indicate the location of the missing value in `fmatrix`:

```
int **na[3];
for (i=0; i < 3; i++);{
   na[i] = (int *)malloc(5 * sizeof(int));
   for (j=0, j < 5; j++);{
      na[i][j] = FALSE;
   }
}
na[2][4] = TRUE;
```

4.6 List object

```
open_r_list(name);
   ...
close_r_list();
```

A list in R is a collection of other objects, for example, of vectors, matrices, data frames, or other lists. Each item in a list must have a unique name.

Between initializing and closing a list, the user may initialize and write up to maxcomp other data objects as list components.

EXAMPLE

```
open_r_list("la.matrices");
   wrt_r_matrix("la.obsd", "%f", 3, 5, la_o);
   wrt_r_matrix("la.pred", "%f", 4, 6, la_p);
close_r_list();
```

The example shows the use of a list to store two matrices. Items within a list need not be of the same type, nor of the same shape or size.

4.7 Writing comments to the output file

For troubleshooting purposes, or whenever the C2R output file will be viewed directly, it can be useful to insert R comments. These are ignored by the R parser, just as they would be in an R script.

EXAMPLES

```
wrt_r_comment("Parameter values follow.");
wrt_r_comment("This run includes the revised October data.");
```

5 Acknowledgments

The authors are grateful for the support of the U.S. National Marine Fisheries Service, NOAA, during development of this software. In particular, support was provided by the NMFS Southeast Fisheries Science Center, NMFS Northeast Fisheries Science Center, and NMRS National Fisheries Toolbox program. K. W. Shertzer and E. H. Williams helped us test and refine ADMB2R; R. Cheshire and C. Krouse read drafts of the three X2R manuals; and D. Fournier helped us better understand his AD Model Builder software. Among other compilers, the GNU C++ compiler gcc and the open-source Fortran 95 compiler g95 (due to A. Vaught) were used in testing. We thank the authors of those tools for their many efforts. Finally, we thank R. Gentleman, R. Ihaka, and the R Core Team, for without them, R would not exist.

Bibliography

Dalgaard, P. 2002. *Introductory statistics with R.* Springer, New York. 288 p.

Maindonald, J., and J. Braun. 2003. *Data Analysis and Graphics Using R: An Example-Based Approach.* Cambridge University Press, New York. 362 pp.

R Development Core Team. 2004. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org. Last accessed: October 10, 2006.

Venables, W. N., and B. D. Ripley. 2000. *S Programming.* Springer, New York. 264 p.

Venables, W. N., and B. D. Ripley. 2003. *Modern Applied Statistics with S, fourth edition.* Springer, New York. 195 p.

Verzani, J. 2005. *Using R for Introductory Statistics.* Chapman & Hall/CRC, Boca Raton. 414 p.

Appendix A Listing of example file `test-c2r.c`

```
/*****************************************************************************************
/ C2R-Test.c
/  This is a driver for testing the output routines in C2R.h
/  Those routines write [parts of] an R- or S-language compatible
/  output object.
/  This program generates some dummy data and invokes those routines.
/  Code translated from Mike Prager's Fortran to C by Andi Stephens
*****************************************************************************************/

#include "C2R.h"

// Globals

#define nyr 6
#define nage 5

    int i, j, iyr[nyr], iage[nage];         // iyr and iage are integer vectors
    double vec1[nyr], vec2[nyr];            // v1 and v2 are real vectors
    double *matrix[nyr];                    // matrix is array of pointers to real arrays
    char *colnames[nage], *rownames[nyr];   // arrays of string pointers
    int *mv_mat[nyr];                       // missing-value matrix
    int mv_vec[nyr];                        // missing-value vector

int main() {

    // Fill the year and age vectors.
    for (i=0; i < nyr; i++) {
        iyr[i] = 1990 + i;
    }
    for (j=0; j < nage; j++) {
        iage[j] = j+1;
    }

    // Generate column names
    for (i=0; i < nage; i++) {
        // Allocate storage for the string
        colnames[i] = (char *)malloc(sizeof(r_name));
        sprintf(colnames[i], "Var%d", i+1);
    }

    // Generate row names
    for (j=0; j < nyr; j++) {
        // Allocate storage for the string
        rownames[j] = (char *)malloc(sizeof(r_name));
        sprintf(rownames[j], "Year%d", j+1);
    }

    // Initialize missing-value vector and matrix.
    for (i=0; i < nyr; i++) {
        mv_vec[i] = FALSE;
        // Allocate storage space for arrays of logical
        // values.  Matrix is nyr x nage in size.
        mv_mat[i] = (int *)malloc(nage * sizeof(int));
        for (j=0; j < nage; j++) {
            mv_mat[i][j] = FALSE;
        }
    }

    // Assign NAs in a few places.
    mv_mat[1][2] = TRUE;
    mv_mat[3][4] = TRUE;
    mv_vec[5] = TRUE;

    // Pre-fill the real arrays with random numbers
//    srand(time(NULL));
//    for (i=0; i < nyr; i++) {
//        // Allocate storage space for arrays of real
//        // values.  Matrix is nyr x nage in size.
//        matrix[i] = (double *)malloc(nage * sizeof(double));
//        vec1[i] = rand();
```

```
//        vec2[i] = rand();
//        for (j=0; j < nage; j++) {
//            matrix[i][j] = rand();
//        }
//    }

    // Pre-fill the real arrays identifiable (real) values
    for (i=0; i < nyr; i++) {
        // Allocate storage space for arrays of real
        // values.  Matrix is nyr x nage in size.
        matrix[i] = (double *)malloc(nage * sizeof(double));
        vec1[i] = i + 0.3;
        vec2[i] = i + 1.3;
        for (j=0; j < nage; j++) {
            matrix[i][j] = j + i + 0.3;
        }
    }

    // Now write an R object that can be read with dget("c2r-test.rdat")
    // Open the output file.  Args are max level, max components,
    // and floating-point precision.  If 0, accept default value.
    open_r_file("test-c2r.rdat", 0,0,2);

    // Write the INFO object : date is automatically inserted.
    open_r_info_list("info");
        wrt_r_info_item("Author", "Andi Stephens");
        wrt_r_info_item("Species", "sarcastic fringehead");
        wrt_r_info_item("Model", "CAA.v2.3");
    close_r_info_list("Units.Len", "mm");

    // Write a floating-point VECTOR
    open_r_vector("Vector_1");
        for (i=0; i < nage; i++) {
            wrt_r_item(rownames[i], "%f", vec1[i]);
        }
    wrt_r_item(rownames[i], "%f LAST", vec1[i]);

    // Write a MATRIX object with character row names and column names
    wrt_r_matrix("Random.matrix", "%f Rn Cn", nyr, nage, matrix, rownames, colnames);

    // Write a DATA FRAME object.  Mix column types: integer,
    // floating-point, character-string.  This illustrates the
    // difference between matrices and dataframes.
    open_r_df("Timeseries");
        wrt_r_df_col("Integers", "%d", nyr, iyr);
        wrt_r_df_col("Float1", "%f", nyr, vec1);
        wrt_r_df_col("Strings", "%s", nyr, rownames);
    wrt_r_df_col("Float2", "%f NA Rn LAST", nyr, vec2, mv_vec, rownames); //LAST flag

    // Write a LIST object containing a matrix and a vector.
    open_r_list("Two.Things");
        wrt_r_matrix("Random.matrix", "%f NA Ri Ci", nyr, nage, matrix, mv_mat, iyr, iage);
        // Write a VECTOR object
        open_r_vector("Int.vector");
            wrt_r_item("Int1", "%d", 100);
            wrt_r_item("Int2", "%d", 12);
            wrt_r_item("Int3", "NA"); // Missing value flag
        wrt_r_item("Int4", "%d LAST", -27); // LAST flag
   close_r_list();

    // Close the file
    wrt_r_comment("Calling close_r_file -- last call in program.");
    close_r_file();
    return;

}// END main
```

Appendix B Listing of resulting R-compatible file `test-c2r.rdat`

The following file was created by running the C program given in Appendix A.

Note: The symbol ➔ indicates that a line has been broken in printing, but is continuous in the actual source.

```
### This file written with C2R version 1.02
### Read this file into R or S with x=dget("test-c2r.rdat")
### C2R originated by Mike.Prager@noaa.gov.  Please credit author and report bugs/➔
    improvements.

structure(list(

info=structure(list
(date="Friday, 05 Oct 2007 at 17:46:27"
,Author="Andi Stephens"
,Species="sarcastic fringehead"
,Model="CAA.v2.3"
,Units.Len="mm"),
.Names = c("Date","Author","Species","Model","Units.Len"))

,Vector_1=structure(c
(3.00e-01,1.30e+00,2.30e+00,3.30e+00,4.30e+00,5.30e+00),
.Names = c("Year1","Year2","Year3","Year4","Year5","Year6"))

,Random.matrix=structure(c(
3.00e-01,1.30e+00,2.30e+00,3.30e+00,4.30e+00,1.30e+00,2.30e+00,3.30e+00,4.30e+00,5.30e➔
    +00,2.30e+00,3.30e+00,4.30e+00,5.30e+00,6.30e+00,3.30e+00,4.30e+00,5.30e+00,6.30e➔
    +00,7.30e+00,4.30e+00,5.30e+00,6.30e+00,7.30e+00,8.30e+00,5.30e+00,6.30e+00,7.30e➔
    +00,8.30e+00,9.30e+00),
.Dim = c(6,5),.Dimnames = list(c("Year1","Year2","Year3","Year4","Year5","Year6"),c("Var1",➔
    "Var2","Var3","Var4","Var5")))

,Timeseries=structure(list
(Integers=c(1990,1991,1992,1993,1994,1995)
,Float1=c(3.00e-01,1.30e+00,2.30e+00,3.30e+00,4.30e+00,5.30e+00)
,Strings=c("Year1","Year2","Year3","Year4","Year5","Year6")
,Float2=c(1.30e+00,2.30e+00,3.30e+00,4.30e+00,5.30e+00,NA)),
.Names = c("Integers","Float1","Strings","Float2"),
row.names = c("Year1","Year2","Year3","Year4","Year5","Year6"),
class = "data.frame")

,Two.Things=structure(list(

Random.matrix=structure(c(
3.00e-01,1.30e+00,2.30e+00,3.30e+00,4.30e+00,1.30e+00,2.30e+00,NA,4.30e+00,5.30e+00,2.30e➔
    +00,3.30e+00,4.30e+00,5.30e+00,6.30e+00,3.30e+00,4.30e+00,5.30e+00,6.30e+00,NA,4.30e➔
    +00,5.30e+00,6.30e+00,7.30e+00,8.30e+00,5.30e+00,6.30e+00,7.30e+00,8.30e+00,9.30e+00),
.Dim = c(6,5),.Dimnames = list(c("1990","1991","1992","1993","1994","1995"),c("1","2","3","➔
    4","5")))

,Int.vector=structure(c
(100,12,NA,-27),
.Names = c("Int1","Int2","Int3","Int4"))

),.Names = c("Random.matrix","Int.vector"))

### Calling close_r_file -- last call in program.
),.Names=c("info","Vector_1","Random.matrix","Timeseries","Two.Things"))
```

Appendix C Reading `test-c2r.rdat` into R

The following text is a transcript of an R session.

```
> results=dget("test-c2r.rdat")
> results

$info
$info$Date
[1] "Monday, 11 Sep 2006 at 13:05:27"

$info$Author
[1] "Andi Stephens"

$info$Species
[1] "sarcastic fringehead"

$info$Model
[1] "CAA.v2.3"

$info$Units.Len
[1] "mm"


$Vector_1
Year1 Year2 Year3 Year4 Year5 Year6
  0.3   1.3   2.3   3.3   4.3   5.3

$Random.matrix
      Var1 Var2 Var3 Var4 Var5
Year1  0.3  2.3  4.3  6.3  8.3
Year2  1.3  3.3  5.3  7.3  5.3
Year3  2.3  4.3  6.3  4.3  6.3
Year4  3.3  5.3  3.3  5.3  7.3
Year5  4.3  2.3  4.3  6.3  8.3
Year6  1.3  3.3  5.3  7.3  9.3

$Timeseries
      Integers Float1 Strings Float2
Year1     1990    0.3   Year1    1.3
Year2     1991    1.3   Year2    2.3
Year3     1992    2.3   Year3    3.3
Year4     1993    3.3   Year4    4.3
Year5     1994    4.3   Year5    5.3
Year6     1995    5.3   Year6     NA

$Two.Things
$Two.Things$Random.matrix
       1   2   3   4   5
1990 0.3 2.3 4.3 6.3 8.3
1991 1.3  NA 5.3  NA 5.3
1992 2.3 4.3 6.3 4.3 6.3
1993 3.3 5.3 3.3 5.3 7.3
1994 4.3 2.3 4.3 6.3 8.3
1995 1.3 3.3 5.3 7.3 9.3

$Two.Things$Int.vector
Int1 Int2 Int3 Int4
 100   12   NA  -27
```