NOAA Technical Memorandum NMFS-SEFSC-548

User's Guide to For2R: A Module of Fortran 95 Output Routines Compatible with the R Statistics Language

Michael H. Prager
Jennifer L. Martin
Andi Stephens

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
Software version 1.03

NOAA Technical Memorandum NMFS-SEFSC-548

User's Guide to For2R: A Module of Fortran 95 Output Routines Compatible with the R Statistics Language

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

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

Andi Stephens
Southeast Fisheries Science Center
Beaufort, North Carolina

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

October, 2006
Software version 1.03

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

Prager, M. H., J. L. Martin, and A. Stephens. 2006. User's guide to For2R: A Module of Fortran 95 output routines compatible with the R statistics language. U.S. Department of Commerce, NOAA Technical Memorandum NMFS–SEFSC–548. iv + 25 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

June, 2005	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 For2R

For2R is a collection of Fortran routines for saving complex data structures into a file that can be read in the R statistics environment with a single command.[1] For2R 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 For2R 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 For2R to others in the hope that they will find it useful.

Please note that For2R 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 For2R interface. The other packages, including documentation, are available from the authors or from CRAN.

Because Fortran 95 includes all of Fortran 77, programs written to the the earlier standard can be used with For2R, so long as compilation is done with a Fortran 95 compiler. Free compilers for Fortran 95 are now available; for example, the compiler g95 can be downloaded from

http://www.g95.org/

[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.

1.2 The For2R package

The For2R package contains the following files:

- This guide
- The Fortran module `For2R.f90`, which contains the output routines described here
- A sample driver program, `for2r-test.f90`, which demonstrates the use of the For2R routines
- A makefile, `makefile.mak`, for compiling the driver program. This is set up for Opus make and Lahey Fortran under Windows and will need modification for other systems.
- A sample output file, `for2r-test.rdat`, generated by the driver. This can be read into R to demonstrate the type of data structures possible with For2R.

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 For2R and FishGraph

The authors have developed a series of R graphics functions that produce typical graphs of fisheries stock-assessment model output. We anticipate making these FishGraph functions available on the CRAN archive in late 2006. Until FishGraph appears on CRAN, working copies are available from the authors upon email request.

The FishGraph functions take an argument that is an R list, making FishGraph highly compatible 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.

1.5 Reporting Problems

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.

1.6 Data Structures in For2R

Output from For2R 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 For2R 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 For2R calls, along with corresponding object names, metadata, and data structures specified by the user. Much of the usefulness of For2R 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.** This is a specialized list of character strings in name–value pairs. The current date and time are included automatically. The info list is intended for storing metadata such as the analyst's name, units used in calculations, etc.
- **Vector** of real or integer numbers or character strings. Here, a vector is a series of name–value pairs, intended to represent a collection of values, such as scalars from a model.
- **Complete vector** of numbers, written all at once from an array in the modeling source.
- **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). For2R 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. For2R supports writing missing values to its output file in R-compatible form.

2 Usage considerations

2.1 Precision

Real numbers are transferred in ASCII (formatted) form. Precision of written values is governed by the user's specification of the argument `digits` in the subroutine `open_r_file`. Internal data handling is done in double precision.

2.2 Object names

Names of R objects written by For2R may be up to 32 characters long. This limit can be increased by changing the For2R 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.

2.3 Error checking

Basic error checking is provided by Fortran compilers. In particular, the use of a Fortran module by For2R ensures that subroutine arguments are of correct type and rank.

Other errors are possible, and For2R checks for some of them. However, many unchecked errors are possible in writing code. For example, it is possible to call the For2R routines to create a file that cannot be parsed correctly by R. This will happen, for example, if the argument `last` is not used to signal the final element in an object, or if an incorrect sequence of calls leads to some other problem with object specification.

3 Sequence of calls

The following sequence of calls could be used to write a typical brief data object. For simplicity, calls are shown here without arguments. Details of calls, including required and optional arguments, are given in following sections by object type.

Dots (`...`) signify optional repetition of the preceding call.

Level	Call	Comment
0	`call open_r_file`	Open the master data list
1	`  call open_r_info_list`	Open the `info` list
2	`    call wrt_r_item`	Write a value to the `info` list
2	`    ...`	Write more values to `info` list
1	`  call open_r_vector`	Initialize a vector
2	`    call wrt_r_item`	Write a vector element
2	`    ...`	Write more vector elements
1	`  call wrt_r_matrix`	Write a matrix
1	`  call open_r_list`	Open a list
2	`    call wrt_r_matrix`	Write a matrix to the list
2	`    call wrt_r_matrix`	Write a matrix to the list
1	`  call close_r_list`	Close the list
0	`call close_r_file`	Close the file and master list

The preceding sequence writes a master object containing (1) the special `info` list (with date), (2) a vector, (3) a matrix, and (4) a list containing two matrices.

4 Specifications

This section includes specifications of all subroutines in For2R, grouped by type of data object written. Most—but not all—objects require a call to initialize the object and additional call(s) to write data. **A table of arguments** for each set of subroutines indicates each argument's name, its Fortran data type, whether required or optional, and its purpose. Although we have given many examples, users unfamiliar with Fortran optional arguments may need to consult Fortran documentation.

When calling subroutines, optional arguments must be preceded by the argument name and an equals sign. (This is the *keyword form* of argument specification.) Required arguments may be specified in keyword form or by themselves. For example, the following two calls are equivalent, because argument `fname` is required and the default value for optional argument `mxlevel` is `6`—

```
call open_r_file(fname="myoutput.rdat", mxlevel=6)
```

```
call open_r_file("myoutput.dat")
```

4.1 Open and close R-compatible data file

4.1.1 Overview

Subroutine `open_r_file` finds a free Fortran unit number, opens the output file, and writes initialization information to it. Subroutine `close_r_file` writes final information to the output file and closes it.

4.1.2 Call specification

```
call open_r_file(fname, mxlevel=6, mxcomp=128, digits=7)
call close_r_file
```

☛ The values shown for `mxlevel`, `mxcomp`, and `digits` are used when values for those optional arguments are not given by the user.

4.1.3 Table of arguments

Argument	Type	Required	Description
`fname`	character	required	Name of file to open for writing
`mxlevel`	integer	optional	Maximum nesting level of object names
`mxcomp`	integer	optional	Maximum number of component names at any level
`digits`	integer	optional	Number of digits after the decimal point in transfer of real values

☛ Storage is allocated by For2R for object names based on the values of mxlevel and mxcomp. This involves the concept of nesting level (see §4.1.4).
☛ Each optional argument provided by the user should be preceded by argument name and equals sign.

4.1.4 Nesting levels

In For2R, the user specifies the name of each data object. However, that name is written only after the containing component is complete. For example, the name of a data frame column is written only after the entire data frame is complete. To accomplish this, For2R keeps track of names and the nesting level at which each name is defined. The user, when opening an output file, must specify the maximum nesting level possible in the application (mxlevel) and the number of names possible at each level (mxcomp).

Object names referring to direct components of the master object are by definition at nesting level 1. For example, a data frame within the master object has its name stored at level 1. Column names of that data frame are stored at level 2. Nesting level can increase markedly when lists are stored. Components of a list would have their names at level 2; subcomponents of those components at level 3, and so on.

The argument mxlevel sets the maximum nesting level allowed. If mxlevel is not given in the subroutine call, the default value of 6 is used.

The argument mxcomp sets the maximum number of names allowed at any nesting level. For example, this controls the maximum number of components at level 1 of the master object, the maximum number of column names in any data frame, or the maximum number of components in any list. Once a data item has been completely written, name storage is released for other objects. If mxcomp is not given in the subroutine call, the default value of 128 is used.

4.1.5 Example

```
call open_r_file("run22.rdat", digits=5)
   ...
call close_r_file
```

4.1.6 Notes

These two subroutines must be the first and last calls in the sequence that writes an R object. At present, only one file holding an R object can be open for writing at a time. It must be completed and closed before another is written.

4.2 Info list

4.2.1 Overview

The `info` list contains a series of name–value pairs. The date can be inserted automatically as the first such pair. At least one additional pair is required.

The list is initialized with the routine `open_r_info_list`, and data elements are written to it with subroutine `wrt_r_item`.

4.2.2 Call specification

```
call open_r_info_list(name, date=.true.)
call wrt_r_item(name, ax, last=.false.)
```

☛ The values shown for `date` and `last` are used if values for those optional arguments are not given by the user.

☛ There is no default value for `ax`, so a value must be passed in the call. Keyword form must be used when passing the value for `ax`. (This is because subroutine `wrt_r_item` has additional arguments, which are not used in this context.)

4.2.3 Table of arguments

Argument	Type	Required	Description
`open_r_info_list`			
`name`	character	required	Name for info list
`date`	logical	optional	Set `.false.` to disable automatic date insertion
`wrt_r_item`			
`name`	character	required	Name of info component
`ax`	character	required*	Value of info component
`last`	logical	optional	Set `.true.` to indicate final data component in `info` object

* `ax` is formally an optional argument, but it must be given (in keyword form) in this context.

4.2.4 Example

```
call open_r_info_list("info")
   call wrt_r_item("units.length", ax="mm")
   call wrt_r_item("units.mass", ax="mt")
call wrt_r_item("run.title", ax=title, last=.true.)
```

4.2.5 NOTES

The example shows the use of the info list to store metadata describing an analysis. In the final call, the example assumes that `title` is a Fortran character variable containing suitable data.

In this example, `open_r_info` was called without the optional argument `date`. Thus, the date and time will be stored in character format as the first component of the info list.

An alternative to using the info list is writing a vector of character values (§4.4). Such a vector displays more compactly in R, which in some cases is advantageous.

4.3 Comment object

4.3.1 Overview

For troubleshooting purposes, or whenever the For2R output file will be viewed directly, it can be useful to insert R comments. These are ignored by the R parser, just as they are when entered at the command line.

4.3.2 Call specification

```
call wrt_r_comment(text)
```

4.3.3 Examples

```
call wrt_r_comment("INFO list follows this comment.")
...
call wrt_r_comment(text="Vector Q follows this comment.")
```

4.3.4 Notes

Any trailing blanks in the comment text are removed when `text` is written to the output file.

Because the comments are ignored by the R parser, they are not part of the data object once it has been read into R.

4.4 Vector object (element by element)

4.4.1 Overview

In For2R, a vector may be stored in one of two ways. The routine described here writes an unordered collection of named real, integer, or character values to an R vector, one element at a time. This is useful, e. g., when storing a collection of scalars from a model run. In contrast, the routine in §4.5 writes a complete vector of values all at once.

When using the present routine, a separate vector should be written for each type of data (real, integer, or character) to maintain the correct data types when R reads the file.

4.4.2 Call specification

```
call open_r_vector(name)
call wrt_r_item(name, x, ix, ax, na=.false., last=.false.)
```

4.4.3 Table of arguments

Argument	Type	Required	Description
name	character	required	Name of vector or vector element
x	real	optional	Real value of data item
ix	integer	optional	Integer value of data item
ax	integer	optional	Character value of data item
na	logical	optional	Missing value indicator for data item
last	logical	optional	Set .true. to indicate final data item in vector

☛ If the optional argument last is not given by the user, the default value of .false. is used (as shown). The same applies to optional argument na.
☛ A missing but named element may be included in the vector by specifying na=.true. .
☛ One of x, ix, or ax should be given in the call. Omitting all three optional arguments is equivalent to specifying na=.true.

4.4.4 Example

```
call open_r_vector("parms")
   call wrt_r_item(name="vb.li", x=linf)
   call wrt_r_item("vb.k", x=vbk)
   call wrt_r_item("vb.t0", na=.true.)
call wrt_r_item("MSY", x=msy, last=.TRUE.)
```

4.4.5 NOTES

The example shows use of a vector to store four quantities from an analysis in fish population dynamics. The first, second, and fourth elements of the vector are stored from Fortran variables; the third element is set to a missing value.

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

```
localmsy <- result$parms["MSY"]
```

4.5 Vector object (complete)

4.5.1 Overview

This routine writes a vector of character or numeric values (real or integer) to the output file with a single subroutine call. To store a vector one element at a time, see §4.4.

4.5.2 Call specification

```
call wrt_r_complete_vector(name, x, ix, ax, na=.false., el_names, el_ids)
```

4.5.3 Table of arguments

Argument	Type	Required	Description
`name`	character	required	Name of vector
`x`	real	optional	Real values forming vector
`ix`	integer	optional	Integer values forming vector
`ax`	character	optional	character strings forming vector
`na`	logical	optional	Missing value mask for vector
`el_names`	character	optional	Element names as text
`el_ids`	integer	optional	Element names as integers

- All arguments except `name` are Fortran arrays of rank 1.
- One and only one data vector (`x`, `ix`, or `ax`) can be given. The data-matrix argument must be given in keyword form in the call.
- If names are desired, either `el_names` or `el_ids` may be given.
- If present, the missing-value vector `na` must be of the same size as the data vector. Elements of `na` set to `.true.` indicate missing elements of the data vector. All other elements of `na` should be set to `.false.`, since uninitialized elements may cause errors.

4.5.4 Examples

```
call wrt_r_complete_vector("height", x=height(:))
call wrt_r_complete_vector("year", ix=yr)
call wrt_r_complete_vector("ht.labeled", x=height, el_ids=yr)
```

4.5.5 Notes

In the example, the first line stores a real vector under name `height`; the second, an integer vector under name `year`. The third line uses the values of the Fortran variable `yr` as labels for the elements of vector `ht.labeled`. This presumes that the two Fortran vectors have the same length.

4.6 Matrix object

4.6.1 Overview

A matrix object is a two-dimensional array. Only matrices of real or integer numbers are currently supported by `For2R`.

4.6.2 Call specification

```
call wrt_r_matrix(name, x, ix, na, rownames, colnames, rowids, colids)
```

4.6.3 Table of arguments

Argument	Type	Required	Description
`name`	character	required	Name of matrix
`x`	real	optional	Real data matrix (Fortran array of rank 2)
`ix`	integer	optional	Integer data matrix (Fortran array of rank 2)
`na`	logical	optional	Missing-value mask (Fortran array of rank 2)
`rownames`	character	optional	Row names as text (Fortran array of rank 1)
`rowids`	integer	optional	Row names as integers (Fortran array of rank 1)
`colnames`	character	optional	Column names as text (Fortran array of rank 1)
`colids`	integer	optional	Column names as integers (Fortran array of rank 1)

☛ *Rank* denotes the number of dimensions in an array.
☛ One and only one data matrix (`x` or `ix`) must be given. In either case, the data-matrix argument should be given in keyword form.
☛ Row names may be omitted or be given as character (`rownames=xxxx`) *or* integer (`rowids=xxxx`). The same is true of column names.
☛ If present, the missing-value matrix `na` must be of the same size and shape as the data matrix. Elements of `na` set to `.true.` indicate missing elements of the data matrix. All other elements of `na` should be set to `.false.`, as uninitialized elements may cause errors.

4.6.4 Examples

```
call wrt_r_matrix("n.at.age", x=naa)
call wrt_r_matrix("n.at.age", x=naa, rownames=years, colids=ages)
```

4.6.5 NOTES

The example shows two copies of the same real matrix stored into the output file. The first copy is stored without row or column names. The second copy is stored with row names from Fortran character variable `years` and column names from Fortran integer variable `ages`. Both `years` and `ages` are assumed to be arrays of rank one (vectors) in the calling Fortran program.

In choosing between integer and real storage, the user should bear in mind that the absolute magnitude of integers available is likely to be smaller than that of available real numbers.

An entire matrix is written by `For2R` in one call. For that reason, separate subroutines are not required for initializing and then writing the matrix.

4.7 Data frame object

4.7.1 Overview

A data frame in R is a collection of data columns (vectors) of equal length. The columns may be of different data types. The customary use of a data frame is to contain a set of observations (samples, stored as rows) on several variables (stored as columns). Columns of a data frame always carry names; rows may carry names. In this version of For2R, data frame columns may contain real numbers, integers, or character values.

4.7.2 Call specification

```
call open_r_df(name)
call wrt_r_df_col(name, x, ix, ax, na, last, rownames, rowids)
```

4.7.3 Table of arguments

Argument	Type	Required	Description
name	character	required	Name of data frame or column
x	real	optional	Real values for column (Fortran array of rank 1)
ix	integer	optional	Integer values for column (Fortran array of rank 1)
ax	character	optional	Character values for column (Fortran array of rank 1)
na	logical	optional	Missing value indicator (mask) for x, ix, or ax
last	logical	optional	Set .true. if last column in data frame
rownames	character	optional	Row names as character values. Used when last==.true.
rowids	integer	optional	Row names as integer values. Used when last==.true.

☛ One and only data vector should be given. Thus, the call should include any one of argument x, argument ix, or argument ax. The data-vector argument, like all optional arguments, should be named (i.e., keyword form should be used).

☛ The missing-value vector na must be of the same length and rank as the data vector. Elements of na set to .true. indicate missing elements of the data vector. Other elements of na should be set to .false., as elements of na that have not been assigned values may cause errors.

☛ In R data frames, row names refer to all columns of the data frame. If row names are desired in the output, they should be provided by the user with the data *for the final column written*. Either character or integer row names may be provided. If both are given, the character values will be used.

4.7.4 Example

```
call open_r_df("timeseries")
   call wrt_r_df_col(name="year", ix=year)
   call wrt_r_df_col("biomass", x=b, na=na_b)
   call wrt_r_df_col("cpue.obsd", x=u_obsd, na=na_cpue)
call wrt_r_df_col("cpue.pred", x=u_pred, last=.TRUE., rowids=year)
```

The preceding example shows the use of a data frame to store four time series from a fish stock analysis. In the calling program, the integer array `year` and real arrays `b`, `u_obsd`, and `u_pred` should all be of of the same size (say, n) and of rank 1. The two missing value arrays `na_b` and `na_cpue` and the row ID array `year` should also be of of size n and rank 1. In this example, the same numeric values are written as row names and as the first column of the data frame.

4.8 List object

4.8.1 Overview

A list in R is a collection of other objects, i. e., of vectors, matrices, data frames, model results (not supported in For2R), or other lists. Each component of a list must have a unique name.

In this version of For2R, a list may contain vectors, matrices, data frames, and other lists.

4.8.2 Call specification

```
call open_r_list(name)
   ...
call close_r_list
```

Between initializing and closing a list, the user should initialize and write the components contained by the list. The `name` argument is required.

4.8.3 Example

```
call open_r_list("laa.matrices")
   call wrt_r_matrix("laa.obsd", laa_o)
   call wrt_r_matrix("laa.pred", laa_p)
call close_r_list
```

4.8.4 Notes

The example shows the use of a list to store two matrices. The components of a list need not be of the same shape or size. The subroutine `wt_r_matrix` is described in §4.6.

Because of the nature of a list, a dedicated subroutine is used to close all lists.

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 `for2r-test.f90`

```
! ============================================================
! FOR2R-TEST.F90     M.H.Prager   May 2005 - Aug 2006
! ============================================================
   ! This is a driver program for testing for2r.f90.
   ! Those routines write [parts of] an R-compatible output object.
   ! This program generates some dummy data & calls those routines.
! ============================================================
! Use subroutines to test any scoping issues  6/16/05
! For2R Routines renamed for similarity to C2R, A. Stephens, 1/2006
! Changed name of INFO object, M. Prager, 8/2006
! Changed KINDs to use default kinds, M. Prager, 8/2006
! ============================================================
   program for2rtest
      implicit none
      integer, parameter            :: r4 = kind(1.0)    !--real single precision
      integer, parameter            :: r8 = kind(1.0d0)  !--real double precision
      integer, parameter            :: nyr=6, nage=5
      integer                       :: i, iyr(nyr), iage(nage)
      real(r8), dimension(nyr)      :: v1, v2, v3
      real(r8), dimension(nyr,nage) :: m1, m2
      character(len=16)             :: colnames(nage), rownames(nyr)
      character(len=4)              :: adum
      logical, dimension(nyr,nage)  :: mv_mat
      logical, dimension(nyr)       :: mv_vec


      !========  The following are initialization tasks to fill Fortran
      !========  arrays with numbers, etc., to prepare for testing the
      !========  export functions to R.
      ! Note: variables m1, m2 are matrices
      ! Note: variables v1, v2, etc. are real vectors
      ! Note: variables iyr and iage are integer vectors

      ! Initialize missing-value matrix and vector:
      mv_mat(:,:) = .false.
      mv_mat(1,2) = .true.
      mv_mat(4,3) = .true.
      !
      mv_vec(:)   = .false.
      mv_vec(2)   = .true.
      mv_vec(3)   = .true.

      ! Pre-fill the real arrays with random numbers:
      call random_number(m1)
      m2 = 1.0e12 * m1
      call random_number(v1)
      call random_number(v2)
      call random_number(v3)

      ! Fill the YEAR array with integers:
      do i = 1, nyr
         iyr(i) = 1990 + i
      end do

      ! Generate column names:
      do i = 1, nage
         write(unit=adum, fmt="(i2.2)") i
         colnames(i) = "var" // adum
         iage(i) = i - 1
      end do

      ! Generate row names:
      do i = 1, nyr
         write(unit=adum, fmt="(i4.4)") iyr(i)
         rownames(i) = "yr" // trim(adum)
      end do

      !===== Now write a complicated R object that can be read with dget("filename")

call sub1()
```

```
call sub2(m1, m2, mv_mat, rownames, colnames, iyr, nyr, nage)
call sub3(v1, v2, v3, mv_vec,iyr, iage, m1, m2, mv_mat, rownames, nyr, nage)

end program for2rtest

subroutine sub1
      use for2r
      implicit none
      integer, parameter      :: r4 = kind(1.0)    !--real single precision
      integer, parameter      :: r8 = kind(1.0d0)  !--real double precision

      ! Open the file
      call open_r_file("for2r-test.rdat", digits=6)
      ! Write the INFO object:
      call open_r_info_list("info")
         call wrt_r_item("author",ax="Michael H. Prager")
         call wrt_r_item("run",ax="32b")
         call wrt_r_item("species",ax="yellow grouper")
         call wrt_r_item("model",ax="Statistical CAA")
         call wrt_r_item("units.len",ax="mm")
         call wrt_r_item("units.L",ax="mt", last=.TRUE.)
      ! Write an ELEMENTWISE VECTOR object:
      call open_r_vector("real.parms")
         call wrt_r_item("vb.k", x=0.2_r8)
         call wrt_r_item("vb.li", x=122.2_r8)
         call wrt_r_item("vb.k", x=0.2_r8)
         call wrt_r_item("vb.t0", na=.true., last=.TRUE.)
      return
end subroutine sub1

subroutine sub2(mat1, mat2, mvmat, rnames, cnames, id0, nr, nc)
      use for2r
      implicit none
      integer, parameter      :: r4 = kind(1.0)    !--real single precision
      integer, parameter      :: r8 = kind(1.0d0)  !--real double precision
      integer, intent(IN)     :: nr, nc
      real(r8), intent(IN)    :: mat1(nr,nc), mat2(nr,nc)
      logical, intent(IN)     :: mvmat(nr,nc)
      character(len=*), intent(IN) :: rnames(nr), cnames(nc)
      integer, intent(IN)     :: id0(nr)
      !
      ! Write a complete vector:
      call wrt_r_complete_vector("vector3", ax=cnames)
      ! Write a MATRIX object with character rownames and column names:
      call wrt_r_matrix("ran.mat",x=mat1,rownames=rnames,colnames=cnames)
      ! Write another MATRIX object with integer row names, no column names, and NAs
      call wrt_r_matrix("ran2.mat",x=mat2, na=mvmat, rowids=id0)
      ! Write a MATRIX object with no row or column names:
      call wrt_r_matrix("ran3.mat",x=mat2)
      return
end subroutine sub2

subroutine sub3(vec1, vec2, vec3, mvvec, id1, id2, mat1, mat2, mvmat, rnames, nr, nc)
      use for2r
      implicit none
      integer, parameter      :: r4 = kind(1.0)    !--real single precision
      integer, parameter      :: r8 = kind(1.0d0)  !--real double precision
      integer, intent(IN)     :: nr, nc
      real(r8), intent(IN)    :: mat1(nr,nc), mat2(nr,nc), vec1(nr), vec2(nr), vec3(nr)
      logical, intent(IN)     :: mvmat(nr,nc), mvvec(nr)
      integer, intent(IN)     :: id1(nr), id2(nc)
      character(len=*), intent(IN) :: rnames(nr)


      ! Write a DATA FRAME object:
      call open_r_df("tseries")
         call wrt_r_df_col("id",ix=id1)
         call wrt_r_df_col("var1", x=vec1)
         call wrt_r_df_col("var2", x=vec2, na=mvvec)
         call wrt_r_df_col("var3", x=vec3, last=.TRUE., rownames=rnames)
      ! Write some complete vectors, with and without names
      call wrt_r_complete_vector("vector1", x=vec3)
      ! Write a list object containing two matrices and a complete vector:
```

```
      call open_r_list("lcomp.mats")
         call wrt_r_matrix("lcomp.a", x=mat1, na=mvmat, rowids=id1, colids=id2)
         call wrt_r_matrix("lcomp.b", x=mat2, rowids=id1, colids=id2)
         call wrt_r_complete_vector("vector2", x=vec3, el_names=rnames)
      call close_r_list
      !
      ! Close the file
      call wrt_r_comment("Calling close_r_file -- last call in program.")
      call close_r_file
      return
end subroutine sub3
! ============================================================
```

Appendix B Listing of resulting R-compatible file `for2rtest.rdat`

This listing shows the file created by running the Fortran program in Appendix A. The symbol ➔ indicates a line that has been broken for printing but is continuous in the source.

```
### This file written with For2R version 1.03.
### Read this file into R or S with x=dget('for2r-test.rdat').
### For2R written by Mike.Prager@noaa.gov. Please credit author and report bugs/➔
    improvements.

structure(list(

info = structure(list
(date ="Monday, 11 Sep 2006 at 15:38:35"
,"Michael H. Prager","32b","yellow grouper","Statistical CAA","mm","mt"),
.Names = c("date","author","run","species","model","units.len","units.L"))

,real.parms=structure(
c( 2.000000E-01, 1.222000E+02, 2.000000E-01,NA),
.Names = c("vb.k","vb.li","vb.k","vb.t0"))

,vector3=structure(c(
"var01","var02","var03","var04","var05"),
.Names=NULL)

,ran.mat=structure(c(
 4.343074E-01, 4.543782E-01, 9.143141E-01, 4.826358E-01, 7.198956E-01, 7.313963E-01,
 8.962970E-01, 4.702251E-01, 9.891793E-02, 3.266718E-01, 7.035185E-01, 2.288867E-01,
 6.176318E-01, 7.967285E-02, 6.718546E-01, 2.578035E-01, 7.432624E-01, 7.414805E-01,
 9.543727E-01, 7.298225E-01, 3.383379E-01, 4.264307E-01, 1.169208E-01, 3.524058E-01,
 5.314035E-01, 5.779926E-01, 8.660569E-01, 1.344878E-02, 8.788866E-01, 1.122137E-01),
.Dim = c(6,5),
.Dimnames = list(
c("yr1991","yr1992","yr1993","yr1994","yr1995","yr1996"),
c("var01","var02","var03","var04","var05")))

,ran2.mat=structure(c(
 4.343074E+11, 4.543782E+11, 9.143141E+11, 4.826358E+11, 7.198956E+11, 7.313963E+11,
NA, 4.702251E+11, 9.891793E+10, 3.266718E+11, 7.035185E+11, 2.288867E+11,
 6.176318E+11, 7.967285E+10, 6.718546E+11,NA, 7.432624E+11, 7.414805E+11,
 9.543727E+11, 7.298225E+11, 3.383379E+11, 4.264307E+11, 1.169208E+11, 3.524058E+11,
 5.314035E+11, 5.779926E+11, 8.660569E+11, 1.344878E+10, 8.788866E+11, 1.122137E+11),
.Dim = c(6,5),
.Dimnames = list(
c("1991","1992","1993","1994","1995","1996"),
NULL))

,ran3.mat=structure(c(
 4.343074E+11, 4.543782E+11, 9.143141E+11, 4.826358E+11, 7.198956E+11, 7.313963E+11,
 8.962970E+11, 4.702251E+11, 9.891793E+10, 3.266718E+11, 7.035185E+11, 2.288867E+11,
 6.176318E+11, 7.967285E+10, 6.718546E+11, 2.578035E+11, 7.432624E+11, 7.414805E+11,
 9.543727E+11, 7.298225E+11, 3.383379E+11, 4.264307E+11, 1.169208E+11, 3.524058E+11,
 5.314035E+11, 5.779926E+11, 8.660569E+11, 1.344878E+10, 8.788866E+11, 1.122137E+11),
.Dim = c(6,5),
.Dimnames = list(
NULL,
NULL))

,tseries=structure(list
(id=c(1991,1992,1993,1994,1995,1996)
,var1=c( 4.388587E-01, 8.772471E-01, 6.230486E-01, 2.340929E-01, 5.005605E-01, 2.091781E➔
    -01)
,var2=c( 3.236074E-01,NA,NA, 1.334418E-01, 6.757775E-01, 6.904809E-01)
,var3=c( 7.323978E-02, 7.664885E-01, 9.251272E-01, 4.576593E-01, 3.932098E-01, 7.195412E➔
    -01)
),
.Names = c(
"id","var1","var2","var3"),
row.names = c("yr1991","yr1992","yr1993","yr1994","yr1995","yr1996"),
```

```
class = "data.frame")

,vector1=structure(c(
 7.323978E-02, 7.664885E-01, 9.251272E-01, 4.576593E-01, 3.932098E-01, 7.195412E-01),
.Names=NULL)

,lcomp.mats=structure(list(

,lcomp.a=structure(c(
 4.343074E-01, 4.543782E-01, 9.143141E-01, 4.826358E-01, 7.198956E-01, 7.313963E-01,
NA, 4.702251E-01, 9.891793E-02, 3.266718E-01, 7.035185E-01, 2.288867E-01,
 6.176318E-01, 7.967285E-02, 6.718546E-01,NA, 7.432624E-01, 7.414805E-01,
 9.543727E-01, 7.298225E-01, 3.383379E-01, 4.264307E-01, 1.169208E-01, 3.524058E-01,
 5.314035E-01, 5.779926E-01, 8.660569E-01, 1.344878E-02, 8.788866E-01, 1.122137E-01),
.Dim = c(6,5),
.Dimnames = list(
c("1991","1992","1993","1994","1995","1996"),
c("0","1","2","3","4")))

,lcomp.b=structure(c(
 4.343074E+11, 4.543782E+11, 9.143141E+11, 4.826358E+11, 7.198956E+11, 7.313963E+11,
 8.962970E+11, 4.702251E+11, 9.891793E+10, 3.266718E+11, 7.035185E+11, 2.288867E+11,
 6.176318E+11, 7.967285E+10, 6.718546E+11, 2.578035E+11, 7.432624E+11, 7.414805E+11,
 9.543727E+11, 7.298225E+11, 3.383379E+11, 4.264307E+11, 1.169208E+11, 3.524058E+11,
 5.314035E+11, 5.779926E+11, 8.660569E+11, 1.344878E+10, 8.788866E+11, 1.122137E+11),
.Dim = c(6,5),
.Dimnames = list(
c("1991","1992","1993","1994","1995","1996"),
c("0","1","2","3","4")))

,vector2=structure(c(
 7.323978E-02, 7.664885E-01, 9.251272E-01, 4.576593E-01, 3.932098E-01, 7.195412E-01),
.Names = c(
"yr1991","yr1992","yr1993","yr1994","yr1995","yr1996"))
)
,.Names = c("lcomp.a","lcomp.b","vector2"))
### Calling close_r_file -- last call in program.
),

 .Names=c("info","real.parms","vector3","ran.mat","ran2.mat","ran3.mat","tseries","vector1"➔
     ,"lcomp.mats"))
```

Appendix C Sample R Session

This listing illustrates an R session that reads the data file created in Appendix A.

```
R : Copyright 2006, The R Foundation for Statistical Computing
Version 2.3.1 (2006-06-01)
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
>
> ##### Read the data #####
> fordata <- dget("for2r-test.rdat")
>
> ##### Get some information about fordata #####
> str(fordata, max.level=1)
List of 9
 $ info      :List of 7
 $ real.parms: Named num [1:4]   0.2 122.2   0.2    NA
  ..- attr(*, "names")= chr [1:4] "vb.k" "vb.li" "vb.k" "vb.t0"
 $ vector3   : chr [1:5] "var01" "var02" "var03" "var04" ...
 $ ran.mat   : num [1:6, 1:5] 0.434 0.454 0.914 0.483 0.720 ...
  ..- attr(*, "dimnames")=List of 2
 $ ran2.mat  : num [1:6, 1:5] 4.34e+11 4.54e+11 9.14e+11 4.83e+11 7.20e+11 ...
  ..- attr(*, "dimnames")=List of 2
 $ ran3.mat  : num [1:6, 1:5] 4.34e+11 4.54e+11 9.14e+11 4.83e+11 7.20e+11 ...
  ..- attr(*, "dimnames")=List of 2
 $ tseries   :'data.frame':     6 obs. of  4 variables:
 $ vector1   : num [1:6] 0.0732 0.7665 0.9251 0.4577 0.3932 ...
 $ lcomp.mats:List of 3
>
> ##### Names and contents of info list #####
> names(fordata$info)
[1] "date"      "author"    "run"       "species"   "model"     "units.len" "units.L"
>
> fordata$info$date
[1] "Monday, 11 Sep 2006 at 15:38:35"
>
> fordata$info$run
[1] "32b"
>
> ##### Print the included data frame #####
> fordata$tseries
         id      var1      var2       var3
yr1991 1991 0.4388587 0.3236074 0.07323978
yr1992 1992 0.8772471        NA 0.76648850
yr1993 1993 0.6230486        NA 0.92512720
yr1994 1994 0.2340929 0.1334418 0.45765930
yr1995 1995 0.5005605 0.6757775 0.39320980
yr1996 1996 0.2091781 0.6904809 0.71954120
>
> ##### Run simple statistics on the data frame #####
> summary(fordata$tseries)
       id            var1             var2             var3
 Min.   :1991   Min.   :0.2092   Min.   :0.1334   Min.   :0.07324
 1st Qu.:1992   1st Qu.:0.2853   1st Qu.:0.2761   1st Qu.:0.40932
 Median :1994   Median :0.4697   Median :0.4997   Median :0.58860
 Mean   :1994   Mean   :0.4805   Mean   :0.4558   Mean   :0.55588
 3rd Qu.:1995   3rd Qu.:0.5924   3rd Qu.:0.6795   3rd Qu.:0.75475
 Max.   :1996   Max.   :0.8772   Max.   :0.6905   Max.   :0.92513
                                 NA's   :2.0000
>
```