NOAA Technical Memorandum NMFS-SEFSC-546

User's Guide to ADMB2R: A Set of AD Model Builder Output Routines Compatible with the R Statistics Language

Jennifer L. Martin
Michael H. Prager
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
Revised October, 2009
Software version 1.15

NOAA Technical Memorandum NMFS-SEFSC-546

User's Guide to ADMB2R: A Set of AD Model Builder Output Routines Compatible with the R Statistics Language

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

Michael H. Prager
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

Revised October, 2009
Software version 1.15

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

Martin, J. L., M. H. Prager, and A. Stephens. 2006. User's guide to ADMB2R: A set of AD Model Builder output routines compatible with the R statistics language. U.S. Department of Commerce, NOAA Technical Memorandum NMFS–SEFSC–546. iv + 24 p.

Contacting the authors

Please refer all question to Mike Prager: Mike.Prager@noaa.gov. In coordination with the other authors, he will endeavor to provide support to users.

Revision summary

August, 2005	Distributed for internal SEFSC use
October, 2006	Released as NOAA Technical Memorandum
October, 2007	Revision coincident with improved compatibility.

More complete information is found in the change log file distributed with the software.

Obtaining this document and software

It is the authors' intent to provide each new version of this document, along with corresponding software, to the Comprehensive R Archive Network (CRAN) for distribution,

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

We will also supply the software and documentation to the ADMB Project,

http://admb-project.org/

Paper copies with latest software revisions can be obtained by printing this PDF file from the software distribution. Copies of the original version will remain 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 ADMB2R

ADMB2R is a collection of AD Model Builder routines for saving complex data structures into a file that can be read in the R statistics environment with a single command.[1] ADMB2R 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, and then analyze results with R. We wanted to automate data transfer to speed diagnostics during working-group meetings. The ADMB2R interface writes an R data object (of type `list`) to a plain-text file. The master list can contain matrices, values, dataframes, vectors or lists, all of which can be read into R with a single call to the `dget` function.

Having the capacity to transfer model data, metadata, and results has sharply reduced time we spend on diagnostics, while increasing our diagnostic capabilities. The simplicity of the interface and the capabilities of R have let us to automate graph and table creation for reports. Finally, persistent storage in files makes it simple to treat model results in analyses devised months or years later. We hope that others will find ADMB2R similarly useful.

The format used by the R `dput` and `dget` functions (and thus by ADMB2R) is human-readable code compatible with the R parser. We think it highly unlikely that the R parser will become incompatible with files written by ADMB2R, which uses only R basic features. However, should that occur, the files can be modified with a text editor.

Please note that ADMB2R 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 ADMB2R interface. The other packages, including documentation, are available from the authors or from CRAN.

1.2 The ADMB2R distribution

ADMB2R is intended for use with AD Model Builder (here, ADMB), an open-source modeling package that is a C++ code generator.[1,2] Therefore, the ADMB2R user must have a working copy of ADMB and a suitable C++ compiler.

The ADMB2R distribution contains the following:

- This guide, file `admb2r-guide.pdf`.

[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.

[2] ADMB was created by Otter Research Ltd. and is currently maintained by the ADMB Foundation and ADMB Project http://admb-project.org/

- The ADMB2R source, file admb2r.cpp.
- A copy of the NASA Open-Source Agreement under which ADMB2R is distributed, in file ADMB2R-license.pdf.
- Brief installation instructions, in file ADMB2R-INSTALL.txt.
- **A simple example** of a fishery catch–age model, supplied in the Example1 subdirectory of the ADMB distribution.
  - The sample template file, testadmb2r.tpl, is set up for ADMB2R and calls ADMB2R routines. (It is a modified version of file catage.tpl supplied by Otter Research and is used by permission.)
  - The data file, testadmb2r.dat, is needed to run the sample template.
  - The sample ADMB2R calls are in file make-Rfile.cxx. This is incorporated into the sample template file by an #include preprocessor directive.
  - A copy of the R-compatible output produced by the sample calls is also included. It can be found in file testadmb2r.rdat.original, so named to avoid being overwritten when the user runs the example, whose output will be in testadmb2r.rdat.
- **A more complex example** of a fishery catch–age model is provided in the Example-GM subdirectory.
  - This example also includes an ADMB template, data file, and file of ADMB2R calls.
  - The R-compatible output from this example works with our group of automated R graphics functions, FishGraph. The directory also includes an R script (go.r) that reads the generated file, sources the FishGraph functions, and generates several hundred graphs. (For the latter to work, FishGraph must be installed separately.)[3]

1.3 Installing ADMB2R

To begin installation, unpack the supplied archive (x2r.zip or x2r.tar.gz) to your hard disk in a location of your choice. Then, just copy the file admb2r.cpp from the ADMB subdirectory of that location to a location where ADMB can find it.

- A simple approach is to copy admb2r.cpp to the same directory as your model template. The disadvantage of this approach is that you can wind up with multiple copies of admb2r.cpp.
- The recommended approach is to copy admb2r.cpp into the ADMB installation tree. The ADMB installation routine sets environment variable ADMB_HOME. Simply copy admb2r.cpp into the include subdirectory of the directory indicated by ADMB_HOME.

☛ Windows users can find the location indicated by ADMB_HOME by using a Run... dialog to run the command "cmd /k set a" (without the quotes). That will open a command window showing the values of all environment variables that begin with "a". Environment variable names on Windows are not case-sensitive.

After installation, we suggest that you try compiling the examples provided.

[3] As of September, 2009, FishGraph is available from CRAN as a Windows installer. Work is underway to eliminate platform dependencies and make it available as a true R package.

1.4 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).

1.5 Reporting Problems in ADMB2R

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 ADMB2R 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.

The ADMB2R distribution includes an example of an ADMB template (program) that writes a fairly complex R list structured to work with FishGraph. We recommend examining that example if you would like to use any of the automatic graphing provided by FishGraph.

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.

1.7 Data Structures in ADMB2R

Output from ADMB2R 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 ADMB2R 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 ADMB2R calls, along with corresponding object names, metadata, and data structures specified by the user. Much of the usefulness of ADMB2R 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 ADMB2R 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.

  The **info list** subroutines can also be used to easily write an R list whose components are single numeric values, such as a collection of parameter estimates.
- **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.
- **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). ADMB2R 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. ADMB2R supports writing missing values to its output file in R-compatible form.

2 Usage considerations

2.1 Compatibility with C++ compilers

ADMB2R was written to the ANSI/ISO C++ standard. It should function properly on all compilers and operating systems supported by ADMB. We have tested it with the following combinations of ADMB and compilers:

- With ADMB 6.03: Borland C++ compiler, version 5.5.1;[4] and the MinGW release of gcc, version 2.95.
- With ADMB 7.7.1: Borland C++ compiler, version 5.5.1; and the MinGW release of gcc, version 3.4.2.

The authors are collaborating with the ADMB Project to ensure compatibility with all compilers supported by ADMB. As of October, 2009, no such problems are known. Problem reports should be sent directly to the ADMB Project.

2.2 Writing Objects

Data objects are written by calling ADMB2R routines from the user's ADMB program. Objects may require one, two, or more calls for complete writing. Most objects require a call to initialize the object, additional call(s) to write data, and a call to finish writing the object.

2.3 Precision

Values are output by ADMB2R using the compiler's default precision. (This is likely to be about 6 digits.) Scientific notation is used when needed. When opening the output file, the user can specify that more or less precision be used in data transfer (§5.1).

2.4 Error checking

Basic error checking is provided by C++ compilers. ADMB2R checks for some additional errors. It generates a log file, `admb2r.log`, that contains any errors encountered at runtime.

Many types of error are unchecked by ADMB2R. For example, it is possible to call ADMB2R routines to create a file that cannot be parsed correctly by R. This might happen if the ADMB call to close an object is missing or if matrix or vector indices are referenced incorrectly.

[4] As of the date of this writing, the Borland compiler was available for free download at http://www.codegear.com/tabid/139/Default.aspx

2.5 Object names

The ADMB2R system does not enforce correct naming of R data objects and object components—that is up to the user! Legal R names vary by context,[5] but in general, a syntactic name in S should begin with a letter and may contain only upper- and lower-case letters, digits, and the period (dot) character. Names so formed are handled correctly by ADMB2R.

In a few cases, ADMB2R is more restrictive than R; this avoids problems when the R parser reads the file generated by ADMB2R.

- The R language allows the underscore character in names, but we advise against it, and we have not tested ADMB2R with such names.[6]
- Some names beginning with a dot are allowed by R, but they may be handled incorrectly by ADMB2R. Therefore, names such as `".height"` should be avoided.
- The R system allows a nonstandard name to be used in some contexts if the name is enclosed in quotation marks. Such usage is not compatible with ADMB2R, so names such as `"2y"` or `"abc&2"` should be avoided.

Dimension names of arrays (`dimnames`) are not considered object or component names; therefore, they may be given numeric names in ADMB2R.[7] However, a data frame is a form of R list, not an R array. Columns of the data frame are list components and must have standard syntactic names. In contrast, row names of data frames may be composed solely of digits.

2.6 Missing values

Missing values are supported by ADMB2R in several ways. When opening the output file (§5.1), the user can supply a number that represents missing data (e.g., −99999). Data that match that number will be written as missing values (represented NA in R). In writing a vector object element by element, the user can explicitly set a datum missing by omitting its value (§5.4). In writing a data frame object, missing values are inserted automatically when a vector doesn't span the full length of the data frame. Alternatively, a boolean vector can indicate missing values (§5.7). In writing a matrix, a boolean matrix can indicate missing values (§5.6) as well. Details are given with the call specifications, in the sections indicated.

[5] See section 7.14 in the R 2.6.2 FAQ.
[6] In other versions of the S language, the underscore can be an assignment operator.
[7] More precisely, they may be given names that are strings composed solely of digits.

3 Using ADMB2R with ADMB

3.1 Including the ADMB2R source

To make the ADMB2R functions available, the user must incorporate the ADMB2R source code with an `#include` preprocessor directive in `GLOBALS_SECTION` of the model. The file `admodel.h`, distributed with ADMB, must be referenced in the same way. It may be necessary to refer to the files by full pathnames unless they are in the same directory as the ADMB program. Here is an example of the added statements under Windows:

```
//*****************************************************
GLOBALS_SECTION
   #include "c:\admb\include\admodel.h"
   #include "c:\admb2r\admb2r.cpp"
//*****************************************************
```

3.2 Including calls to ADMB2R functions

Calls to ADMB2R functions are executed during the reporting phase of ADMB execution. It is recommended that users write their ADMB2R code in a separate file (we use files with a `.cxx` extension) and incorporate them with an `#include` preprocessor directive at the bottom of the ADMB template file's `REPORT_SECTION`. Alternatively, the ADMB2R function calls can be written directly into the `REPORT_SECTION`. The following section of a template file shows two lines of conventional ADMB output, followed by ADMB2R output:

```
//*****************************************************
REPORT_SECTION
   ....
   report << "Bmsy " << B_msy_out << endl;
   report << "F/Fmsy " << fullF/F_msy_out << endl;
   ....
#include "make_Robject.cxx"   // ADMB2R code
//*****************************************************
```

4 Typical sequence of calls

The following sequence of calls could be used to write a typical (if brief) data object. The example writes a master list containing an info list, a vector with two elements, a matrix, and a list containing a matrix and a vector. For simplicity, calls are shown here without arguments and are indented structurally.

Level	Call	Action
0	`open_r_file`	Open the master data list
1	`  open_r_info_list`	Open the info object
2	`    wrt_r_item`	Write an info element
1	`  close_r_info_list`	Close the info object
1	`  open_r_vector`	Open a vector
2	`    wrt_r_item`	Write a vector element
2	`    wrt_r_item`	Write a vector element
1	`  close_r_vector`	Close the vector
1	`  open_r_matrix`	Open a matrix
2	`    wrt_r_matrix`	Write the matrix
1	`  close_r_matrix`	Close the matrix
1	`  open_r_list`	Open a list
2	`    open_r_matrix`	Open a matrix as part of the list
3	`      wrt_r_matrix`	Write the matrix
2	`    close_r_matrix`	Close the matrix
2	`    open_r_vector`	Open a vector as part of the list
3	`      wrt_r_item`	Write a vector element
2	`    close_r_vector`	Close the vector
1	`  close_r_list`	Close the list
0	`close_r_file`	Close the master data list

A more complete example is given in Appendix A on page 25.

5 Specifications

The following subsections include calling specifications of all functions in ADMB2R, grouped by type of object written. Objects may require several calls for complete writing. Most require a function call to initialize the object, call(s) to write data, and a call to close the object.

A table of arguments is given for each set of functions. Each line in the table includes the argument name; its data type (real, integer, character or logical); whether the argument is required or optional; and its meaning. When calling routines, arguments must be given in the order specified.

Caution: The interpretation of actual arguments in C++ and ADMB2R is based on their sequence in the call. A function call that includes a value for *any* optional argument *must also include* all optional arguments that precede it in the call specification.

5.1 Open and close output file

Subroutine `open_r_file` opens the output file and writes initialization information to it. This also opens the enclosing `R` list in the file. Subroutine `close_r_file` finalizes the output data, closes the master list, and closes the file. We recommend writing ADMB2R output files with a consistent file extension, such as `.rdat`.

```
open_r_file(fname, numdigits, missingval)
   ...
close_r_file()
```

5.1.1 Table of arguments

Argument	Type	Required	Description
`fname`	Character	Required	Name of file to open for writing.
`numdigits`	Integer	Optional	Digits after the decimal point in transfer of real values. If not given, output uses the compiler's default precision, typically about 6 digits.
`missingval`	Real	Optional	Missing value flag. Any datum matching this value (see §5.1.2) will be output as the `R` missing-value indicator, `NA`.

☛ See caution box on optional arguments, p. 9.

5.1.2 MISSING-VALUE COMPARISONS

Because comparison of floating-point values for equality is unreliable, ADMB2R uses a fuzzy test to decide whether a datum matches the missing-value flag. Let d be the datum being written and m the missing-value flag. Then NA is written instead of the datum when $|m - d| < \varepsilon$. The ADMB2R code declares $\varepsilon = 10^{-6}$. The user can change the value of ε by modifying its declaration in the ADMB2R code.

5.1.3 EXAMPLE 1

```
open_r_file("run22.rdat");
   ...
close_r_file();
```

Example 1 creates an R-compatible output file, `run22.rdat`. All data are written using default precision. No missing-value flag is specified.

5.1.4 EXAMPLE 2

```
open_r_file("run23.rdat", 5, -99999);
   ...
close_r_file();
```

Example 2 creates output file `run23.rdat`. Data are written with five digits after the decimal point. Any datum equal to −99999 will be replaced with NA (but see §5.1.2 for details). Because the optional argument for missing-value flag is given, the optional argument for precision is *required* here.

5.1.5 EXAMPLE 3

```
open_r_file(adprogram_name + ".rdat", 12);
   ...
close_r_file();
```

This code uses the built-in ADMB variable `adprogram_name` to construct an output file name like that of the ADMB template file, but with `.rdat` extension. Data values are written with twelve digits after the decimal point. No missing-value flag is specified.

5.1.6 NOTES

These two functions must be the first and last calls in the sequence that writes an R object. ADMB2R allows only one file holding an R object to be written at a time. It must be completed and closed before another is written.

5.2 Info list

The info list functions were designed for storing a list of metadata as a series of name–value pairs (Example 1, §5.2.2). The series is written to an R list object, to which a date stamp can be written as the first list component. At least one additional component must be written.

The same functions can also be used to write any R list whose components are single values. That might be, for example, the list of parameter estimates from a model (Example 2, §5.2.3).

To write an R list whose components are matrices or vectors of length > 1, the list-writing functions described in §5.8 should be used. They are slightly more complex but offer more power and flexibility.

```
open_r_info_list(name, writestamp)
   wrt_r_item(name, value)
close_r_info_list()
```

5.2.1 Table of arguments

Argument	Type	Required	Description
open_r_info_list()			
name	Character	Required	Name of list written
writestamp	Boolean	Optional	Flag to write date/time stamp to info list. Default: true
wrt_r_item()			
name	Character	Required	Name of info item
value	Character, boolean, integer or double	Optional	Value of info item. If not specified, NA is written.

5.2.2 Example 1—List of metadata

```
open_r_info_list("info");
   wrt_r_item("units.length", "mm");
   wrt_r_item("units.mass", "mt");
   wrt_r_item("run.title", title);
   wrt_r_item("finaldraft", false);
close_r_info_list();
```

The example demonstrates writing an info object to store metadata on an analysis. In the fourth line, the actual argument title is assumed to be a C++ character variable.

Because the call to `open_r_info_list` in this example does not include optional argument `writestamp`, the default value `true` is used. Thus, the name `"date"` and a character representation of the current date and time form the first pair in the info list.

5.2.3 Example 2—List of parameter estimates

```
open_r_info_list("parms", false);
   wrt_r_item("vb.li", vonbert.linf);
   wrt_r_item("vb.k", vonbert.k);
   wrt_r_item("vb.t0", vonbert.k);
   wrt_r_item("msy", MSY);
close_r_info_list();
```

In this example, an R list is written that holds a set of parameter values from an analysis. Each estimate is stored as a separate list component. Because argument `writestamp` is set to `false`, a timestamp is not written to the list by ADMB2R.

Parameters and other values stored this way are compatible with our set of graphics functions, FishGraph.

5.3 Comment object

5.3.1 SUMMARY

For troubleshooting purposes, or whenever the ADMB2R 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.

```
wrt_r_comment(text)
```

5.3.2 EXAMPLES

```
wrt_r_comment("INFO object follows this comment.")
   ...
wrt_r_comment("This file written with revised source.")
```

5.3.3 NOTES

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

5.4 Vector object (elementwise)

5.4.1 Overview

In ADMB2R, a vector can be stored by two different methods. This section describes storing a vector element by element, with a name stored for each element. That might be useful, e. g., when recording several scalars from a model run. Data types (real, integer, logical, character) should not be mixed in the same vector, as an R vector is always of a single data type.

A vector can also be stored by ADMB2R all at once. That would be useful, e. g., when storing an existing vector from an AD Model Builder program. For storing a vector all at once, see §5.5. The design of ADMB2R also accommodates ordered data as matrices (§5.6) or as columns of a data frame (§5.7).

Three functions are used to store a vector elementwise:

```
open_r_vector(name)
   wrt_r_item(name, value)
   ...
close_r_vector()
```

5.4.2 Table of arguments

Argument	Type	Required	Description
`open_r_vector()`			
`name`	Character	Required	Name of vector
`wrt_r_item()`			
`name`	Character	Required	Name of vector element
`value`	Character, boolean, integer or double	Optional	Value of vector element. If not specified, `NA` is written.

5.4.3 Example

```
open_r_vector("parms");
   wrt_r_item("vb.li", linf);
   wrt_r_item("vb.k", vbk);
   wrt_r_item("vb.t0");
   wrt_r_item("MSY", msy);
close_r_vector();
```

5.4.4 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 ADMB variables; the third element is set to a missing value.

A single element from an ADMB vector or matrix can be given as the `value` argument using subscript notation; for example, `myvector(3)` or `mymatrix(2,5)`.

When using the resulting vector 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 as follows:

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

5.5 Vector object (written at once)

5.5.1 Overview

This function writes a vector of numeric values (real or integer) to the output file in a single function call. To store a vector one element at a time, see §5.4.

```
wrt_r_complete_vector(name, xvec, name_vec, isna, na_vector)
```

5.5.2 Table of arguments

Argument	Type	Required	Description
name	Character	Required	Name of vector
xvec	Real or integer	Required	ADMB vector of values
name_vec	Integer	Optional	Integer vector for element names
isna	Boolean	Optional	Signals presence of na_vector argument. Default: false.
na_vector	Boolean	Optional	Where true, corresponding element of xvec vector is written as missing (NA).

Argument name_vec is *completely* optional: even when a missing value vector is supplied, it is not necessary to supply name_vec. When names *are* supplied, they are accessible from R with the names function and will appear when the vector is printed.

If used, name_vec and na_vector each must be the same length as xvec. They need not have the same index values, but they must contain the same number of elements as xvec.

5.5.3 Examples

```
wrt_r_complete_vector(ht, heights);
wrt_r_complete_vector(ht.labeled, heights, year);
wrt_r_complete_vector(trial.results, P_values, true, invalid_trials);
```

5.5.4 Notes

The first line of the example writes a vector named ht from the AD Model Builder vector heights. Elements of ht are not labeled. No elements are set to NA.

The second line of the example writes a vector named ht.labeled from the (same) ADMB vector heights. The ADMB integer vector year is used to label elements of ht.labeled.

The third line writes an unlabeled vector named trial.results from the ADMB vector P_values. The third argument (true), indicates that the following argument, invalid_trials, is a Boolean vector indicating elements of P_values to be set to NA.

5.6 Matrix object

A matrix in R is a two-dimensional array, with every element having the same data type. This version of ADMB2R can write matrices of real or integer numbers. If the types are mixed, R will interpret the matrix as real.

Matrices may have row or column names, both, or neither. Names can be taken from the indices of the ADMB data matrix, or they can be provided explicitly by a call or calls to `wrt_r_namevector` before closing the matrix.

```
open_r_matrix(name)
   wrt_r_matrix(xx, rowoption, coloption, isna, na_matrix)
   wrt_r_namevector(rowvec, i_start, i_stop)  // Call type 1
   wrt_r_namevector(start, stop, inc)         // Call type 2
   ...
close_r_matrix()
```

5.6.1 Arguments of `open_r_matrix`, `wrt_r_matrix`

Argument	Type	Required	Description
`name`	Character	Required	Name of matrix.
`xx`	Real or integer	Required	ADMB matrix of values.
`rowoption`	Integer	Optional	Flag to write row names— 0: Omit names (default). 1: Row indices become names. 2: Names supplied in subsequent call to `wrt_r_namevector`.
`coloption`	Integer	Optional	Flag to write column names— 0: Omit names (default). 1: Column indices become names. 2: Names supplied in subsequent call to `wrt_r_namevector`.
`isna`	Boolean	Optional	Signals presence of `na_matrix` argument. Default: `false`.
`na_matrix`	Boolean	Optional	Where `true`, corresponding element of `xx` matrix is written as missing (`NA`).

☛ See caution box on optional arguments, p. 9.

When using indices for row or column names, consider that ADMB matrices are not necessarily indexed from 1. For example, a matrix might have rows indexed from 1950 to 2005, designating calendar years of the study.

5.6.2 Arguments of `wrt_r_namevector`

Two types of call to `wrt_r_namevector` are supported. The first type generates dimension names from an ADMB data vector of integers. The second type generates names as a regular sequence of numbers.

First call type

```
wrt_r_namevector(rowvec, i_start, i_stop);
```

Argument	Type	Required	Description
`rowvec`	Integer	Required	Vector of values for row or column names.
`i_start`	Integer	Optional	First index of `rowvec` to use.
`i_stop`	Integer	Optional	Last index of `rowvec` to use.

☛ See caution box on optional arguments, p. 9.

Second call type

```
wrt_r_namevector(start, stop, inc);
```

Argument	Type	Required	Description
`start`	Integer	Required	First number in generated sequence.
`stop`	Integer	Required	Last number in generated sequence.
`inc`	Integer	Optional	Increment of generated sequence. Default: 1.

5.6.3 Example 1

```
open_r_matrix("n.at.age");
   wrt_r_matrix(naa, 1, 2);
   wrt_r_namevector(all_years, 5, 20);
close_r_matrix();
```

In Example 1, with 1 passed for `rowoption`, row names are taken from the row indices of the ADMB matrix `naa`. With 2 passed for `coloption`, column names are written by the call to `wrt_r_namevector`; they are the 16 elements of the vector `all_years` indexed 5 through 20.

5.6.4 Example 2

```
open_r_matrix("l.at.age");
   wrt_r_matrix(laa, 2);
   wrt_r_namevector(9, 27, 3);
close_r_matrix();
```

In this example, a matrix named `l.at.age` is written from the ADMB matrix `laa`. Row names are written as the series, $\{9, 12, 15, \ldots, 27\}$. No column names are written.

5.6.5 Example 3

```
open_r_matrix("l.at.age");
   wrt_r_matrix(laa, 1, 1, true, na_matrix);
close_r_matrix();
```

In this example, a matrix named `l.at.age` is written from the ADMB matrix `laa`. The actual argument `true` indicates that an additional, boolean matrix (here named `na_matrix`) is supplied, in which positions of `true` values correspond to missing values in `laa`. Row and column names for `l.at.age` are derived from the row and column indices of `laa`.

5.7 Data frame object

5.7.1 Overview

A data frame in R is a collection of columns (data vectors) of equal length; it forms a rectangular structure similar to a matrix. Unlike a matrix, columns may hold differing data types. A data frame typically contains a set of samples (stored as rows) on several variables (stored as columns). ADMB2R can write columns containing real numbers or integers. A data frame is opened with `open_r_df`, and each column is written in turn with `wrt_r_df_col`.

Function `wrt_r_df_col` can be called in two forms, as illustrated here. The first writes a column from an ADMB vector. The second writes a generated integer sequence.

```
open_r_df(name, start, stop, writerow)
   wrt_r_df_col(name, xx, shift, isna, na_vector)
   wrt_r_df_col(name, start, stop, inc, isna, na_vector)
   ...
   wrt_r_namevector(rowvec, i_start, i_stop)  // Call type 1
   wrt_r_namevector(start, stop, inc)         // Call type 2
close_r_df()
```

Row and column names. Each column of a data frame has a name. The rows may have names, which refer to rows across all columns. Row names are written by calling `wrt_r_namevector` (§5.6.2, §5.7.5) immediately before closing the data frame.

Missing values. Missing values can be written in three different ways. They can be added by the internal coordinate system (described next), by use of the global missing-value indicator (§5.1), or by providing boolean vectors to indicate missing values.

Internal coordinates. When ADMB2R opens a data frame, an internal coordinate system indexing the rows is set up from values passed for arguments `start` and `stop`. This coordinate system can be used to provide row names. More importantly, it is compared to the indices of each ADMB vector written as a column; those indices are expected to match the internal system or—if `shift` is passed—to match it when $(s - 1)$ is added to the vector's indices, where s is the value of `shift`. If the vector is shorter than the internal coordinates, ADMB2R pads it with missing values at the start, finish, or both to align its indices with the internal coordinates. When a data frame is closed by ADMB2R, its internal coordinate system ceases to exist.

In understanding the internal coordinate system, remember that ADMB vectors are not necessarily indexed from 1. For example, an analysis covering fifty years might include a variable indexed as `cost(1951)` through `cost(2000)`. In that case, a data frame might be written using ADMB2R with 1951 passed for `start` and 2000 passed for `stop`. A second variable, observed for only part of the study, might be indexed as `price(1962)` through `price(2000)`. When written as a dataframe column, `price` would be positioned by aligning its indices with the internal coordinate system, and missing values would be assigned at its beginning.

5.7.2 ARGUMENTS OF **open_r_df**

Argument	Type	Required	Description
name	Character	Required	Name of data frame
start	Integer	Required	First value of the data frame coordinate system (internal to ADMB2R)
stop	Integer	Required	Last value of the data frame coordinate system
writerow	Integer	Optional	Integer flag to write data frame row names— 0: No row names (default). 1: Generated sequence from start to stop by 1. 2: Names provided by subsequent call to wrt_r_namevector.

5.7.3 ARGUMENTS OF **wrt_r_df_col(name,xx,shift,isna,na_vector)**

Argument	Type	Required	Description
name	Character	Required	Name of column.
xx	Integer, real	Optional	ADMB data vector to be written.
shift	Integer	Optional	If given, (shift-1) is added to the vector's indices before aligning them with data frame's coordinate system. May be positive or negative.
isna	Boolean	Optional	Must be true if na_vector is supplied.
na_vector	Boolean	Optional	Vector of same length as xx. Where true, corresponding data element will be NA.

☛ See caution box on optional arguments, p. 9.

5.7.4 ARGUMENTS OF **wrt_r_df_col(name, start, stop, inc, isna, na_vector)**

Argument	Type	Required	Description
name	Character	Required	Name of column.
start	Integer	Optional	Starting value of generated sequence.*
stop	Integer	Optional	Ending value of generated sequence.
inc	Integer	Optional	Increment value of generated sequence.
isna	Boolean	Optional	Must be true if na_vector is supplied.
na_vector	Boolean	Optional	Vector of same length as xx. Where true, corresponding data element will be NA.

* When starting value is 0 (zero), include arguments through isna to avoid compilation errors.

☛ See caution box on optional arguments, p. 9.

5.7.5 Example of writing data frame

```
1   firstyear = 1951;
2   lastyear =  1990;
3   ...
4   open_r_df("timeseries", firstyear, lastyear, 2);
5      wrt_r_df_col("year", yr, firstyear);
6      wrt_r_df_col("yearx", firstyear, lastyear);
7      wrt_r_df_col("biomass", b, 0, true, na_vector);
8      wrt_r_df_col("cpue.obsd", u_obsd);
9      wrt_r_df_col("cpue.pred", u_pred);
10     wrt_r_namevector(rownums);
11  close_r_df();
```

In the example, we write a data frame named `timeseries`, whose full range of years is 1951–1990. The ADMB variables used are as follows—

- `firstyear` and `lastyear` are scalar variables.
- `yr` is an ADMB vector indexed from `1` to `40`.
- `b` is an ADMB vector indexed from `1951` to `1990`, with missing values for `b(1983)` through `b(1985)`.
- `na_vector` is a vector of the same size and indexed the same as `b`. All its values are `false`, except for the three years in which `b` has missing values, which are `true`.
- `u_obsd` is a vector of observed CPUE values, indexed from `1971` to `1990`.
- `u_pred` is a vector of predicted CPUE values, indexed from `1951` to `1990`.

Explanation of the example, by line number—

1–2. `firstyear` and `lastyear` are assigned values.

4. The data frame is opened, with internal coordinate system running from 1951 to 1990. An eventual call to `wrt_r_namevector` is indicated.

5. Vector `yr` is written as a column named `year`. It is shifted so that its first element aligns with the first row of the data frame.

6. Column `yearx` is written as a sequence of integers, `1951` to `1990`.

7. Vector `b` is written as column `biomass`. Boolean vector `na_vector` is passed to mark the missing values in `biomass`.

8. Vector `u_obsd` is written as column `cpue.obsd`. It is padded with 20 missing values at the start.

9. Vector `u_pred` is written as column `cpue.pred`.

10. Row names are written from ADMB integer vector `rownums`.

11. The data frame is closed.

5.8 List object

A list in R is a collection of other data objects, for example, of vectors, matrices, data frames, model results (not supported in ADMB2R), or other lists. Each component of a list has a unique name.

In this version of ADMB2R, a list may contain vectors, matrices, data frames, info objects, and other lists.

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

The single argument name is a quoted character string.

Between opening and closing a list, the user should open and write the components of (data objects contained by) the list.

5.8.1 Example

```
open_r_list("laa.matrices");
   open_r_matrix("laa.obsd");
      wrt_r_matrix(laa_o);
   close_r_matrix();
   open_r_matrix("laa.pred");
      wrt_r_matrix(laa_p);
   close_r_matrix();
close_r_list();
```

The example shows the use of a list to store two matrices. The matrices need not be of the same shape or size.

6 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 `make-Rfile.cxx`

The sample file, when incorporated into a suitable ADMB program, generates an output file readable with R. It demonstrates most of the features described in the guide.

```
//========================================================================
// Example ADMB code calling ADMB2R functions to create a file containing
// an R object as output from an AD Model Builder model.
//
// This source should be included in the ADMB REPORT_SECTION. See ADMB2R
// User's Guide.  This file by Andi Stephens, revised by Mike Prager.
// Address inquiries to
// Michael Prager <mike.prager@noaa.gov> or
// Jennifer Martin <jennifer.martin@noaa.gov>
//
// This file is part of the ADMB2R distribution.
//========================================================================
// Open the output file using the AD Model Builder template name, and
// specify 6 digits of precision

open_r_file(adprogram_name + ".rdat", 6, -999);


    // Example of metadata stored in an INFO object (R list)

    open_r_info_list("info", true);
        wrt_r_item("title", "Sample Catch at Age Model");
        wrt_r_item("species", "Flack Lake Trout");
        wrt_r_item("units.len", "mm");
    close_r_info_list();


    // Example of storing scalar parameter estimates into an R list

    open_r_info_list("parms", false);
        wrt_r_item("M", M);
        wrt_r_item("avg_F", avg_F);
        wrt_r_item("pred_B", pred_B);
        wrt_r_item("log_q", log_q);
        wrt_r_item("log_popscale", log_popscale);
        wrt_r_item("Obj.fcn", f);
    close_r_info_list();


   // Example of writing a COMPLETE VECTOR object
   // NOTE: vector "ages" was declared in the DATA_SECTION for use here as:
   //   ivector ages(1,nages);

   for (int i=1;i<=nages;i++) ages(i) = (i);    //populate the ages vector
   wrt_r_complete_vector("log.sel", log_sel, ages);


    // Example of a MATRIX object with no row or column names

    open_r_matrix("F.at.age");
        wrt_r_matrix(F);
    close_r_matrix();


    // Example of a MATRIX object with row names, but no column names;
    // rows specified by constructing a series since a vector of years
    // is not a variable

    open_r_matrix("N.at.age.1");
        wrt_r_matrix(N, 2);
        wrt_r_namevector(1968, 1979);
    close_r_matrix();


    // Example of the same matrix as above object with row and column names;
    // rows specified by constructing a series since the a vector of years
```

```
  // is not a variable; columns specified by constructing a series since
  // a vector of ages isn't supplied

  open_r_matrix("N.at.age.2");
      wrt_r_matrix(N, 2, 2);
      wrt_r_namevector(1968, 1979);
      wrt_r_namevector(3, 9);
  close_r_matrix();


  // Example of a DATA FRAME object composed of two items.
  // The data frame will span 1 to the number of ages. The vectors
  // span 2 to the number of ages, so NAs will be written for the first
  // data point in the two vectors. The row names will be included,
  // similar to the matrix object just before this one.

 open_r_df("aseries", 1, nages, 2);
      wrt_r_df_col("N.pred", predicted_N);
      wrt_r_df_col("N.ratio", ratio_N);
      wrt_r_namevector(3, 9);
  close_r_df();


  // Example of a LIST object

  open_r_list("C.at.age.mats");


      // List component #1: matrix with row and column names

      open_r_matrix("Est");
          wrt_r_matrix(C, 1, 1);
      close_r_matrix();


      // List component #2: another matrix with row and column names

      open_r_matrix("Obs");
          wrt_r_matrix(obs_catch_at_age, 1, 1);
      close_r_matrix();

  close_r_list();

  wrt_r_comment("Begin testing permutations of matrix calls unused above");
  wrt_r_comment("No rownames; col names from matrix");

  open_r_matrix("Obs_catch_at_age");
      wrt_r_matrix(obs_catch_at_age,0,1);
  close_r_matrix();


  wrt_r_comment("No names, NA matrix");

  for ( i = 1; i <= nyrs; i++) {
      for (int j = 1; j<= nages; j++){
          NA[i][j] = 0;
      }
  }
  NA[1][1] = 1;
  NA[2][2] = 1;
  NA[nyrs-1][nages-1] = 1;
  NA[nyrs][nages] = 1;

  open_r_matrix("Obs_catch_at_age");
      wrt_r_matrix(obs_catch_at_age,0,0,1,NA);
  close_r_matrix();

  testcol = column(obs_catch_at_age, 1);

  wrt_r_comment("DF column from series");
  wrt_r_comment("Then obs_catch 1,3:6 shifted by one.");
  open_r_df("Series", 1966, 1970, 2);
      wrt_r_df_col("First", 5,10);
```

```
        wrt_r_df_col("Second", testcol(6,10), 1967);
        wrt_r_namevector(1,5);
    close_r_df();


// close file

close_r_file();

// End of example
```

Appendix B Listing of resulting R-compatible file `test-admb2r.rdat`

This example shows the data transferred by running the ADMB2R calls in Appendix A and reading the resulting file into R.

```
> x <- dget("test-admb2r.rdat")
> # Let's see what's in x:
> print(x)
$info
$info$date
[1] "Friday, 05 Oct 2007 at 14:25:22"

$info$title
[1] "Sample Catch at Age Model"

$info$species
[1] "Flack Lake Trout"

$info$units.len
[1] "mm"


$parms
$parms$M
[1] 0.3

$parms$avg_F
[1] 0.7084428

$parms$pred_B
[1] 1674.736

$parms$log_q
[1] -3.583497

$parms$log_popscale
[1] 7.543647

$parms$Obj.fcn
[1] 330.4038


$log.sel
         1          2          3          4          5          6          7
-3.7432470 -1.0202060  0.6132520  1.7387920  1.6700060  0.7414037  0.7414037

$F.at.age
             [,1]       [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
 [1,] 0.004118935 0.06271720 0.3212096 0.9899298 0.9241253 0.3651271 0.3651271
 [2,] 0.005342071 0.08134136 0.4165943 1.2838940 1.1985490 0.4735533 0.4735533
 [3,] 0.004816444 0.07333788 0.3756040 1.1575670 1.0806190 0.4269586 0.4269586
 [4,] 0.006786159 0.10332980 0.5292095 1.6309610 1.5225440 0.6015659 0.6015659
 [5,] 0.005721330 0.08711618 0.4461703 1.3750440 1.2836390 0.5071731 0.5071731
 [6,] 0.008840350 0.13460810 0.6894029 2.1246580 1.9834230 0.7836618 0.7836618
 [7,] 0.006944907 0.10574700 0.5415893 1.6691140 1.5581610 0.6156383 0.6156383
 [8,] 0.005173629 0.07877657 0.4034586 1.2434110 1.1607570 0.4586216 0.4586216
 [9,] 0.006181739 0.09412662 0.4820747 1.4856970 1.3869370 0.5479865 0.5479865
[10,] 0.006863532 0.10450800 0.5352434 1.6495560 1.5399040 0.6084248 0.6084248
[11,] 0.007660283 0.11663980 0.5973770 1.8410450 1.7186630 0.6790536 0.6790536
[12,] 0.012383660 0.18856050 0.9657229 2.9762430 2.7784010 1.0977620 1.0977620

$N.at.age.1
          [,1]     [,2]     [,3]      [,4]      [,5]      [,6]      [,7]
1968 3925.3690 2303.208 2803.705 1634.4120 251.84320  87.93243 18.881240
1969 8319.1780 2896.032 1602.533 1506.4140 449.93750  74.04560 45.215570
1970 6943.2130 6130.163 1977.830  782.6979 309.07790 100.54040 34.162480
1971 3044.9250 5118.944 4220.204 1006.4170 182.21370  77.70927 48.598930
1972 3013.3460 2240.480 3419.924 1841.6720 145.93930  29.44828 31.544790
1973 4165.2520 2219.606 1521.313 1621.6560 344.94520  29.95066 13.137390
1974 5339.9670 3058.536 1437.236  565.6222 143.53020  35.16186 10.133930
```

```
1975 7091.6370 3928.566 2038.449  619.4844  78.94984  22.38485 14.073920
1976 6172.0740 5226.504 2689.884 1008.7690 132.35350  18.32114 10.483080
1977 4654.6430 4544.207 3524.068 1230.5030 169.15060  24.49673  7.846513
1978 3831.7460 3424.658 3032.372 1528.6310 175.14640  26.86664  9.876071
1979  586.7187 2816.966 2257.735 1236.1090 179.66300  23.26520 10.092890

$N.at.age.2
            3        4        5         6         7         8         9
1968 3925.3690 2303.208 2803.705 1634.4120 251.84320  87.93243 18.881240
1969 8319.1780 2896.032 1602.533 1506.4140 449.93750  74.04560 45.215570
1970 6943.2130 6130.163 1977.830  782.6979 309.07790 100.54040 34.162480
1971 3044.9250 5118.944 4220.204 1006.4170 182.21370  77.70927 48.598930
1972 3013.3460 2240.480 3419.924 1841.6720 145.93930  29.44828 31.544790
1973 4165.2520 2219.606 1521.313 1621.6560 344.94520  29.95066 13.137390
1974 5339.9670 3058.536 1437.236  565.6222 143.53020  35.16186 10.133930
1975 7091.6370 3928.566 2038.449  619.4844  78.94984  22.38485 14.073920
1976 6172.0740 5226.504 2689.884 1008.7690 132.35350  18.32114 10.483080
1977 4654.6430 4544.207 3524.068 1230.5030 169.15060  24.49673  7.846513
1978 3831.7460 3424.658 3032.372 1528.6310 175.14640  26.86664  9.876071
1979  586.7187 2816.966 2257.735 1236.1090 179.66300  23.26520 10.092890

$aseries
       N.pred   N.ratio
3          NA        NA
4  429.302500 0.1863933
5 1728.234000 0.6164107
6  636.761000 0.3895963
7   46.687690 0.1853840
8    8.270396 0.0940540
9    5.749983 0.3045342

$C.at.age.mats
$C.at.age.mats$Est
           1        2         3         4         5         6         7
1  13.941170 121.1532  670.7917  909.0004 134.22420 23.449780  5.035240
2  38.297610 195.8555  476.6139  970.5528 279.44990 24.415620 14.909280
3  28.825510 375.1892  540.0599  476.8911 181.09360 30.506380 10.365710
4  17.794450 435.2752 1518.0020  726.7909 127.61940 30.803040 19.264040
5  14.854210 161.8405 1075.2690 1228.6640  94.01607 10.248660 10.978300
6  31.679000 242.3181  665.9130 1295.2390 269.08350 14.330680  6.285930
7  31.934220 265.8581  526.2487  412.5265 101.58650 14.178680  4.086410
8  31.619730 257.6167  590.5583  392.4451  48.17720  7.195165  4.523783
9  32.866270 406.5814  899.5692  698.5611  88.67582  6.768924  3.873076
10 27.510730 390.5950 1278.7220  892.9574 119.08430  9.792302  3.136559
11 25.266520 326.6844 1195.7590 1159.9530 129.30960 11.633960  4.276598
12  6.240365 420.1974 1236.7730 1080.5080 154.68990 13.755950  5.967596

$C.at.age.mats$Obs
    1   2    3    4   5  6  7
1  13 129  646  954  99 19  4
2  19 169  416 1031 243 47 18
3  40 354  606  479 152 18  7
4  32 606 1424  644 157 23 17
5   0 226 1178 1156 116 16  5
6   2 165  593  982 428 22 11
7  53 209  560  410  30  0  4
8   0 105  674  446  16  2  2
9  46 422  838  726  70  4  4
10  3 310 1224 1068  65  0  0
11 14 354 1264 1172  69  0  6
12  6 429 1222 1067 192  0  0


$Obs_catch_at_age
      1   2    3    4   5  6  7
 [1,] 13 129  646  954  99 19  4
 [2,] 19 169  416 1031 243 47 18
 [3,] 40 354  606  479 152 18  7
 [4,] 32 606 1424  644 157 23 17
 [5,]  0 226 1178 1156 116 16  5
 [6,]  2 165  593  982 428 22 11
 [7,] 53 209  560  410  30  0  4
 [8,]  0 105  674  446  16  2  2
```

```
 [9,] 46 422  838  726  70  4  4
[10,]  3 310 1224 1068  65  0  0
[11,] 14 354 1264 1172  69  0  6
[12,]  6 429 1222 1067 192  0  0

$Obs_catch_at_age
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]   NA  129  646  954   99   19    4
 [2,]   19   NA  416 1031  243   47   18
 [3,]   40  354  606  479  152   18    7
 [4,]   32  606 1424  644  157   23   17
 [5,]    0  226 1178 1156  116   16    5
 [6,]    2  165  593  982  428   22   11
 [7,]   53  209  560  410   30    0    4
 [8,]    0  105  674  446   16    2    2
 [9,]   46  422  838  726   70    4    4
[10,]    3  310 1224 1068   65    0    0
[11,]   14  354 1264 1172   69   NA    6
[12,]    6  429 1222 1067  192    0   NA

$Series
  First Second
1     5     NA
2     6      2
3     7     53
4     8      0
5     9     46

>
```