User's Guide to the R Package PBSadmb
by Jon Schnute and Rowan Haigh
Pacific Biological Station, Nanaimo, BC, Canada

What is PBSadmb?

PBSadmb provides an R interface for developing models with the open version of a software package called AD Model Builder (ADMB). An R function handles every command normally associated with ADMB, including all steps required to make executable files. Consequently, the package enables an R user to handle all aspects of ADMB development with R script files. It includes a facility for ensuring that variable names match between the source file for ADMB (called a *template*) and a corresponding source file for R.

PBSadmb depends heavily on another R package PBSmodelling. We use this to implement a Graphical User Interface (GUI) that greatly facilitates ADMB model development. A user can edit code, test it rapidly, and inspect the results of analyses, including Markov chain Monte Carlo (MCMC) simulations. A new user can learn key aspects of ADMB template code with the examples provided.

In short, PBSadmb extends R to provide full support for ADMB. R users (affectionately called useRs) can regard ADMB as just another R application. Currently, PBSadmb exists only in Microsoft Windows; however, because R supports many common operating systems, we anticipate an R package that will eventually provide a uniform interface across all supported platforms.

You can obtain PBSadmb from the web site: http://code.google.com/p/pbs-software/.

What is PBS?

The initials PBS refer to the Pacific Biological Station, a major fisheries laboratory operated by Fisheries and Oceans Canada on the Pacific coast in Nanaimo, British Columbia, Canada. For more information, see: http://www.pac.dfo-mpo.gc.ca/sci/pbs/.

We have developed a number of packages for R, each starting with the acronym PBS. Three of these (PBSmapping, PBSmodelling, and PBSddesolve) are available on the Comprehensive R Archive Network (CRAN, http://cran.r-project.org/). Because PBSadmb currently includes binary libraries and executable code, we are distributing it from the Google Code site mentioned above rather than from CRAN.

What is ADMB?

The remarkable software package ADMB offers powerful tools for estimating parameters and their uncertainty from complex statistical models. It uses automatic differentiation (sometimes called algorithmic differentiation) to compute function gradients needed for efficient estimation. It includes robust algorithms for modal estimation and MCMC sampling from Bayesian posterior distributions. Other common inference methods, such as likelihood profiles are also supported.

ADMB allows you to examine your data with any statistical model that has a properly defined likelihood function or Bayesian posterior. The model can have hundreds or even thousands of unknown parameters that require estimation.

Originally, ADMB was developed commercially by its principal author David Fournier and the company Otter Research Ltd. (http://www.otter-rsch.com/). It quickly gained wide use in fishery data analyses, although it has potential value in many scientific fields. In 2008, the ADMB Project (http://admb-project.org/) acquired rights to the software and placed it in the public domain. A number of people worked hard to make this possible, and we thank all of them for their efforts. In particular, John Sibert plays a key role in developing and maintaining the Project web site.

How do ADMB and R tie together?

As useRs know, the R software environment easily accommodates external programs. R packages (such as PBSddesolve) often include C/C++ code directly, and the packaging system automatically compiles the code for all supported operating systems. ADMB is a bit unusual in this context because it necessarily involves a C++ environment that cannot be entirely masked by R. The automatic differentiation algorithms, implemented with C++ classes, require a user to express the posterior or likelihood in C++. Fournier had the ingenious idea of making this process as easy as possible with a *template* that handles most of the annoying bookkeeping, so that a user need only write code (very similar to R code) that expresses the model analytically. Program development involves three distinct steps: (1) converting the template to true C++ code, (2) compiling the C++ code, and (3) linking the resulting object module to ADMB libraries. The complete cycle makes an executable file that recognizes a variety of command line arguments.

PBSadmb implements these three steps with the R commands `convAD` (convert to C++), `compAD` (compile C++), and `linkAD` (link to libraries). A composite command `makeAD` performs all three steps sequentially. The ADMB libraries come as part of the PBSadmb installation, so the required paths to them are automatic. However, a user *must* declare a path to the relevant C++ compiler and the text editor used to develop source code. A user can also override the default ADMB path within PBSadmb, but we recommend against it. This version of PBSadmb is designed to work explicitly with the GNU C compiler used when building R packages.

The most convenient interface between ADMB and R comes from the GUI shown below in Figure 1. This allows a user to explore all aspects of ADMB model development. The interface emphasizes four distinct phases:
- **Initialize** the package with appropriate paths , check that they make sense, and save them in a file normally called `Adopts.txt`.
- **Make** the executable file for a chosen prefix, with options between "Safe" and "Optimized" compilation and a choice to have random effects or not.
- **Run** the executable code with suitable command line arguments, where the "All args" button shows all available arguments. The interface gives particular support for generating MCMC samples and likelihood profiles. The "Custom" button supports arbitrary "AD args".

- Inspect the **Output** by "View"ing various reports or "Import"ing them into the R working environment. Again, we give special support to MCMC samples with plots that allow a user to inspect the sampled chain. A user can choose variables for plotting and thin the current chain.
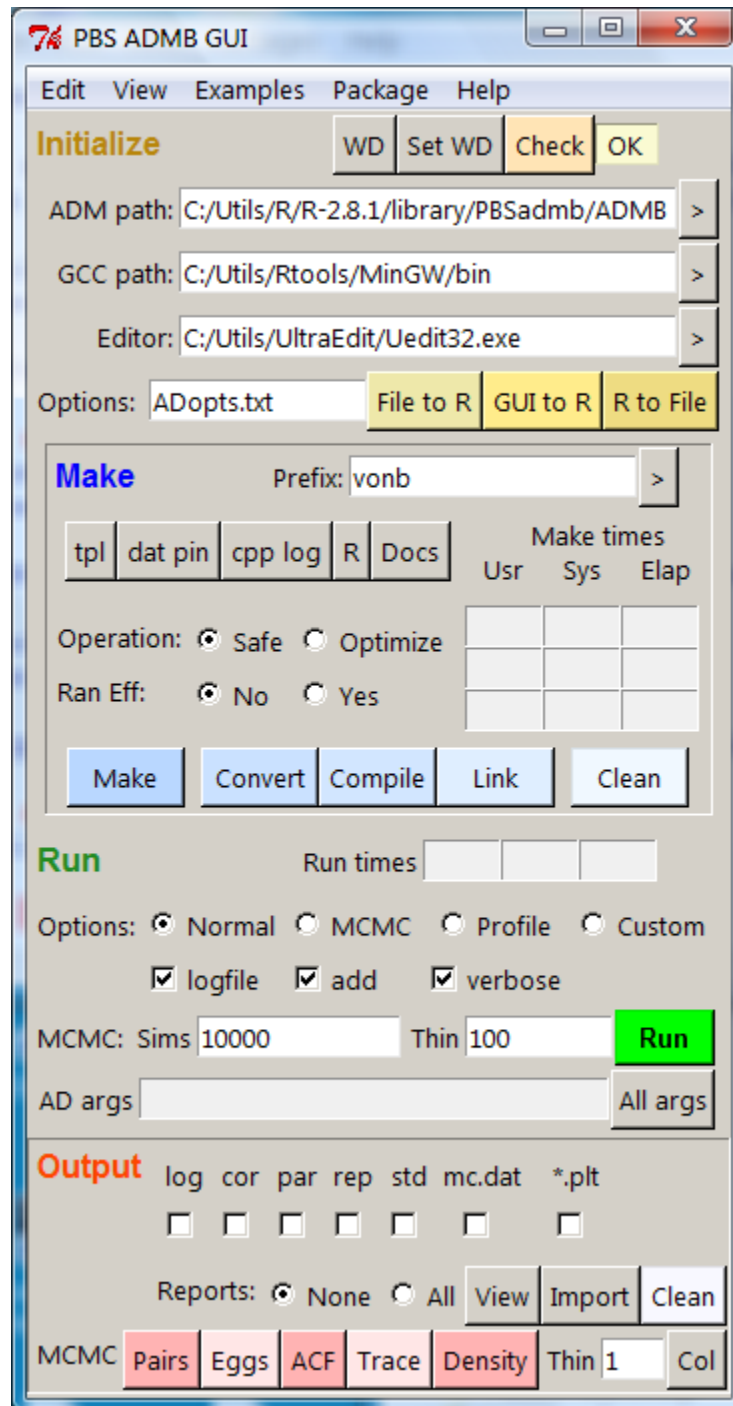

Figure 1. The graphical user interface (GUI) in PBSadmb, generated by the R command `admb()`.

Buttons ">" in the "Initialize" and "Make" sections allow a user to browse for available choices. Text boxes in the "Make" section show the times required for converting (row 1) compiling (row 2), and linking (row 3). The R function `proc.time` reports the 'user time' and 'system time', as well as the elapsed time, and these correspond to the three columns in the interface. Similarly, text boxes in the "Run" section show the run times.

Experienced ADMB users know that ADMB leaves many "footprints", i.e., files in the current working directory. The interface gives you "Clean" buttons to help clean them up. To make things easy, each "Clean" button activates a second GUI that displays potential files associated with the project prefix, as well as other debris files spawned by ADMB. The user can fine-tune the selection using the "Select" and "Deselect" buttons. When the "Clean" button is pressed, a final prompting GUI pops up to confirm deletion of the selected files. Once the files have been deleted, the Clean window remains and the user can choose another prefix (manually typing or pressing the selection button ">") AND hitting the "Refresh" button. This causes the GUI to rebuild itself with files having the newly selected prefix. If no additional files are apparent, the Clean window disappears. Files with suffixes `.tpl, .dat, .pin, .r,` and `.pdf` are never picked for potential deletion. Be careful when cleaning; for example don't delete an output file until you're sure you're ready to do so.

We encourage you to experiment with the GUI. You can quickly see the functionality available in the main menu items. <Edit> allows you to edit the main project files, and <View> displays the output files. <Examples> copies various examples (discussed below) into your working directory. <Package> shows the R code for this package and the Window description file used to create the GUI in Figure 1. <Help> points to manuals in the package, online resources, and this User's Guide.

How do I install PBSadmb?

Installation is easy, as it is for most R packages, although this one has a few extra twists. Essentially, you need to install R, PBSadmb, the R toolkit required for package development, and a text editor suitable for writing templates and viewing reports. Then you need to run R, load PBSadmb, and give it some configuration information. At this point, you have a working version of the interface in Figure 1. Here are the details.

Step 1. Install the current Windows version of R from the web site http://cran.r-project.org/. We assume that you have enough familiarity with R to do this without difficulty. In this example, we also assume that you've used the directory `C:\Utils\R\Rx.xx`, where `x.xx` is the current version number of R.

Step 2. Run R and install the current version of the package PBSmodelling (<Packages>, <Install package(s)...> in the R GUI). Enter the command `require(PBSmodelling)` and check that the R console reports version 2.01 or later.

Step 3. Go to the web site http://www.murdoch-sutherland.com/Rtools/, and download the file "Rtools28.exe". Run this executable file, and install the R tools in a directory of your choice. In this example, we assume you've used the directory `C:\Utils\Rtools`. Take a moment to

inspect the installed files. You should find a subdirectory `C:\Utils\Rtools\MinGW\bin` that contains the GNU compilers, including `g++.exe`. If you type

`C:\Utils\Rtools\MinGW\bin\g++ --version`

in a command window, you should see the result

`g++ (GCC) 4.2.1-sjlj (mingw32-2)`

You're using version 4.2.1, where the `sjlj` refers to "Short Jump/Long Jump". You also have all the tools required to build R packages like this one.

Step 4. Obtain a good text editor that you can use for code development. The Windows Notepad will work, but much better options are available. We happen to use a commercial program called UltraEdit (http://www.ultraedit.com/), but you may prefer to get something free. Our editor supports syntax highlighting and displays multiple files in a single window, with tabs to select among them.

Step 5. Go to the web site http://code.google.com/p/pbs-software/, and download `PBSadmb_y.yy.zip`, where `y.yy` is the most recent version available. Then install PBSadmb from this zip file. (In the R GUI, click <Packages> and select "Install package(s) from local zip files".)

Step 6. Run R in an empty working directory. Then type these two commands into the R console:

`> require(PBSadmb)`
`> admb()`

The GUI should appear, along with a warning message that you have no AD options file. You can use the GUI to set the necessary paths. The "ADM path" should be OK, consistent with the directory you chose in Step 1. On the "GCC path" line, click ">". This brings you into an interface where you can locate the `bin` directory (with `g++.exe`) selected in Step 3. Similarly, click ">" on the "Editor" line to select the executable file for the editor in Step 4. (By default, the GUI points to the Windows Notepad, but hopefully you have a better choice.)

Step 7. Next click "GUI to R" to create an R variable `.Adopts` that contains your specified options. As usual, you can inspect it in the R console by typing its name. Next click "R to File". This creates a file `Adopts.txt` in your current working directory that you can inspect with the text editor. Finally, click the "Check" button. If everything is OK, you should see "OK" in the adjacent text box. The message "Fix" means that something critical can't be found on the paths you've specified. Either you haven't installed something correctly, or one of the paths is wrong.

In the future, when you issue the R command `admb()` with this working directory, the file `Adopts.txt` will automatically determine the paths in the GUI. Furthermore, you can copy this file to any other directory from which you want to use PBSadmb. Conceivably, you might use different option files for projects in different directories

How can I see ADMB in action?

PBSadmb includes a number of examples that teach new users (and remind experienced users) how to write, test, and implement an ADMB template. To see them click <Examples> on the

interface menu. If you click one of them (the file prefix), the program will load all related files into the current working directory. Typically, these have the suffixes

- `.tpl` – the ADMB template file;
- `.dat` – the data used for this template;
- `.pin` – initial values for the parameter estimates;
- `.r` – R code that can be sourced to obtain an extended analysis using both ADMB and R;
- `.pdf` – documentation for this example.

We encourage new users to examine the files in the following order:

Simple, adapted from an example in the ADMB manual, codes the likelihood for regressing a vector `y` on a vector `x`. Take special note of how code is written for the four SECTIONs (DATA, PARAMETER, PROCEDURE, REPORT). Values initialized in the DATA_SECTION come from `simple.dat`, and values initialized in the PARAMETER_SECTION come from `simple.pin`.

SimpleMC, a variant of `simple`, can give a Bayesian posterior sample of the parameters. The GUI allows you to perform and "MCMC" run. You can then view results visually with plots generated from the "Output" section of the GUI.

SimplePBS, a variant of `simpleMC`, has a REPORT_SECTION written explicitly for PBSadmb to ensure that variable names in R code match those from ADMB. In this case, the file `simplePBS.r` (1) makes `simplePBS.exe` from `simplePBS.tpl`, (2) runs `simplePBS.exe`, (3) loads the data from `simplePBS.rep` into R, preserving variable names, and (4) produces a standard regression plot for the data exported imported from `simplePBS.rep`.

vonb, similar to `simplePBS`, estimates parameters for a Bertalanffy growth curve. It can also generate a likelihood profile for the parameter `Linf`, renamed for this purpose as `VonBLinf`. In this case, ADMB generates a file named `VonBLinf.plt`, with the parameter name prefix, *not* the prefix `vonb`.

catage, taken from ADMB web sites, implements a more complex model designed for estimating biological parameters from fishery data on catch and age structure. In the case, the code allows a user to compute a likelihood profile for the predicted biomass `pred_B`.

pheno, also taken from ADMB web sites, implements a model with the "random effects" feature. The lines declaring a `random_effects_vector` play a role similar to `init_vector` in earlier examples, except that the estimation method for random effects variables works differently (and much more slowly). The file `pheno.pin` includes initial values for the two random effects vectors declared in `pheno.tpl`.

How do I write R code to run ADMB?

The examples `simplePBS` and `vonb` both contain R files (`.r`) that illustrate the process. We focus here on the `vonb` example. Display 1 shows the Report Section in the model template. It writes variable names (preceded by $) and variable values. Running the executable file produces the report file `vonb.rep` listed in Display 2. This file has "PBS format", defined in the package PBSmodelling. Think of it as an R list object with named components.

```
REPORT_SECTION
  report << "$Linf"   << endl;
  report << Linf      << endl;
  report << "$K"      << endl;
  report << K         << endl;
  report << "$t0"     << endl;
  report << t0        << endl;
  report << "$sigma"  << endl;
  report << sigma     << endl;
  report << "$fval"   << endl;
  report << fval      << endl;
  report << "$age"    << endl;
  report << age       << endl;
  report << "$size"   << endl;
  report << size      << endl;
  report << "$spred"  << endl;
  report << spred     << endl;
```

Display 1. The Report Section in `vonb.tpl`.

```
$Linf
57.2689
$K
0.164044
$t0
0.152865
$sigma
0.492146
$fval
-3.34367
$age
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
$size
 7.36 14.3 21.8 27.6 31.5 35.3 39 41.1 43.8 45.1 ...
$spred
 7.43029 14.9707 21.3702 26.8015 31.4111 35.3233 ...
```

Display 2. The report file `vonb.rep` produced by running `vonb.exe`. To fit in the space available on this page, the vectors `size` and `spred` are truncated. The file represents an R list with named components.

```
# Initialize
require(PBSmodelling); require(PBSadmb); initAD("Adopts.txt")

# Make and run the file
makeAD("vonb"); runAD("vonb");

# Use PBSmodelling functions to read and unpack the report;
vonb <- readList("vonb.rep"); unpackList(vonb);

# Plot the data, all from ADMB
plot(age,size); lines(age,spred,col="red",lwd=2);

# Check the calculations in R

spredR <- Linf*(1-exp(-K*(age-t0)));
nobs   <- length(age);
fvalR  <- nobs*log(sigma) + sum((spredR-size)^2)/(2.0*sigma^2)

cat("Functions values (ADMB & R):\n");
cat(fval,"   ",fvalR,"\n")

cat("Predictions (ADMB & R):\n");
cat(spred,"\n");
cat(spredR,"\n");
```

Display 3. The R source file `vonb.r`. In the R console, the command `source("vonb.r")` initializes PBSadmb from a file `Adopts.txt` (presumably available and correct), makes `vonb.exe` from `vonb.tpl`, generates the plot in Figure 2, and compares results computed independently by ADMB and R.

Once you understand the relationship between the Report Section in Display 1 and the report file in Display 2, examine the R code in Display 3. It produces the plot in Figure 2, based entirely on data exported from the ADMB model. The PBSmodelling functions `readList` and `unpackList` produce R variables with the same names as corresponding variables in the template file.

The code in Display 3 illustrates the use of PBSadmb functions `initAD`, `makeAD`, and `runAD`. This guide ends with a complete list of functions currently available in the package.
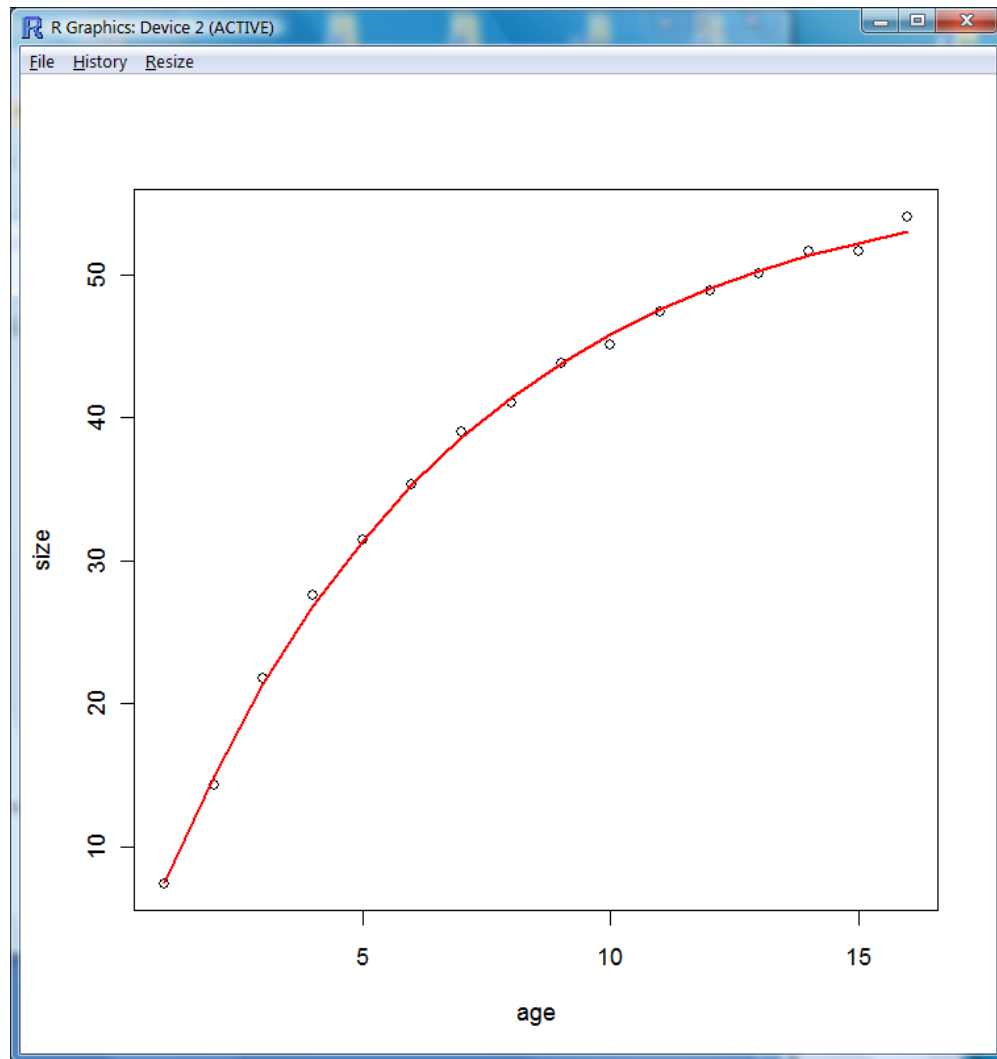
Figure 2. Plot of data points and a fitted von Bertalanffy growth curve, obtained by sourcing the R code in Display 3. The data portrayed here come entirely from the ADMB model. The source code compares these numbers with independent calculations in R.

This page intentionally left blank.

Package 'PBSadmb'

July 9, 2009

Version 0.43

Date 2009-07-09

Title PBSadmb 0.43

Author Jon T. Schnute, Rowan Haigh, Anisa Egeli

Maintainer Jon T. Schnute <Jon.Schnute@dfo-mpo.gc.ca>

Depends R (>= 2.7.0), PBSmodelling

Description R Support for ADMB (AD Model Builder)

License GPL (>=2)

R topics documented:

---

admb *Start the PBS ADMB GUI*

Description

Start up the PBS GUI for running ADMB.

Usage

```
admb(prefix="", pkg="PBSadmb", wdf="admbWin.txt",
     optfile="ADopts.txt", hnam=NULL)
```

Arguments

prefix	string name prefix of the ADMB project (e.g., `"vonb"`).
pkg	string name of package (redundant but may allow flexibility in future).
wdf	string name of the *window description file* that creates the GUI.
optfile	string name of options file (usually in user's working directory).
hnam	string name of history file (not currently used, no history widget).

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC, Canada

See Also

initAD, makeADopts

---

appendLog *Append Data to Log File*

Description

Append summary information or output to a previously created log file.

Usage

```
appendLog(prefix, lines)
```

Arguments

prefix	string name prefix of the ADMB project (e.g., `"vonb"`).
lines	data to append to `'prefix'.log`).

Value

No explicit value reurned. Appends data into a log file `'prefix'.log`.

Note

A wrapper function that can be called from a GUI exists as `.win.appendLog`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

startLog, editADfile

---

checkADopts   *Check ADMB Options for Link Integrity*

Description

Check that .ADopts has all required components and that links point to actual files on the hard drive.

Usage

```
checkADopts(opts=.ADopts, check=c("admpath","gccpath","editor"),
            warn=TRUE, popup=FALSE)
```

Arguments

opts	ADMB options hidden list object .ADopts.
check	components of .ADopts to check.
warn	logical: if TRUE, print the results of the check to the R console.
popup	logical: if TRUE, display program location problems in a popup GUI.

Value

Boolean value where TRUE indicates all programs were located in the specified directories and FALSE if at least one program cannot be found. The returned Boolean scalar has two attributes:
warn - named list of test results, and
message - named vector of test results.

Note

A wrapper function that can be called from a GUI exists as .win.checkADopts.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC, Canada

See Also

initAD, makeADopts, readADopts

---

cleanAD   *Clean ADMB-Generated Files from the Working Directory*

Description

Detects files in the working directory with the specified `prefix` and removes them all save those with the suffix `.tpl`, `.dat`, and `.pin`.

Usage

```
cleanAD(prefix)
```

Arguments

prefix   string name prefix of the ADMB project (e.g., `"vonb"`).

Details

Aside from potential garbage files with the specified `prefix`, other files associated with ADMB are detected. Also files `*.tmp` and `*.bak` are displayed. Calling `cleanAD` invokes the hidden function `.cleanUp`, which creates a GUI menu of the potential garbage files. The user can select whichever files s/he wishes for disposal.

Value

Returns nothing. Invokes a GUI menu of potential garbage files.

Note

A wrapper function that can be called from a GUI exists as `.win.cleanAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

[makeAD](), [runAD](), [readRep]()

---

compAD   *Compile C Code*

Description

Compile C++ code in `'prefix'.cpp` to create a binary object file `'prefix'.o`.

Usage

```
compAD(prefix, raneff=FALSE, safe=TRUE, logfile=TRUE, add=TRUE, verbose=TRUE)
```

Arguments

`prefix`	string name prefix of the ADMB project (e.g., `"vonb"`).
`raneff`	logical: use the random effects model, otherwise use the normal model (currently does not influence the compile stage, but the argument is preserved here for future development).
`safe`	logical: if `TRUE`, use safe mode with bounds checking on all array objects, otherwise use optimized mode for fastest execution.
`logfile`	logical: if `TRUE`, create a log file of the messages from the shell call.
`add`	logical: if `TRUE`, append shell call messages to an exsiting log file.
`verbose`	logical: if `TRUE`, report the shell call an its messages to the R console.

Details

This function uses the C++ comiler declared in `.ADopts`. If `logfile=TRUE`, any errors will appear in `'prefix'.log`. If `verbose=TRUE`, they will appear in the R console.

Value

Invisibly returns the shell call and its messages.

Note

A wrapper function that can be called from a GUI exists as `.win.compAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`convAD`, `linkAD`, `makeAD`

---

convAD	*Convert TPL Code to CPP Code*

Description

Convert code in `'prefix'.tpl` to C++ code in `'prefix'.cpp`.

Usage

```
convAD(prefix, raneff=FALSE, logfile=TRUE, add=FALSE, verbose=TRUE)
```

Arguments

`prefix`	string name prefix of the ADMB project (e.g., `"vonb"`).
`raneff`	logical: if `TRUE`, use the random effects model executable `tpl2rem.exe`, otherwise use the normal model executable `tpl2cpp.exe`.
`logfile`	logical: if `TRUE`, create a log file of the messages from the shell call.
`add`	logical: if `TRUE`, append shell call messages to an exsiting log file.
`verbose`	logical: if `TRUE`, report the shell call an its messages to the R console.

Details

This function invokes the ADMB command `tpl2cpp.exe` or `tpl2rem.exe`, if `raneff` is `FALSE` or `TRUE` respectively. If `logfile=TRUE`, any errors will appear in `'prefix'.log`. If `verbose=TRUE`, they will appear in R console.

Value

Invisibly returns the shell call and its messages.

Note

A wrapper function that can be called from a GUI exists as `.win.convAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

compAD, linkAD, makeAD

---

copyFiles	*Copy System Files*

Description

Copy files with specified prefixes and suffixes from one location to another.

Usage

```
copyFiles(prefix, suffix=NULL, dir0=getwd(), dir1=getwd(), ask=TRUE)
```

Arguments

`prefix`	string scalar/vector of potential file prefixes.
`suffix`	string scalar/vector of potential file suffixes.
`dir0`	source directory from which to copy files.
`dir1`	destination directory to copy files to.
`ask`	logical: if `TRUE`, popup boxes will prompt the user for every instance that a file will be overwritten.

Details

This function uses R's `list.files` and `file.copy` functions. The pattern recognition tends not to work when given the wildcard character `*`; however, the user may use this character, and the code will interpret it.

Value

Invisibly returns a Boolean vector with names of files that have been copied or not.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo, BC

See Also

editAD

---

editAD   *Edit ADMB Files*

Description

Edit files associated with specified prefix and suffixes.

Usage

```
editAD(prefix, suffix=c(".tpl",".cpp",".log"))
```

Arguments

prefix	string name prefix of the ADMB project (e.g., `"vonb"`).
suffix	string scalar/vector specifying one or more suffixes.

Value

Invisibly returns Boolean vector with elements TRUE if files exist, FALSE if they do not.

Note

A wrapper function that can be called from a GUI exists as `.win.editAD`.

This function explicitly uses the editor chosen for PBSadmb. PBSmodelling has another function `openFile` that uses Windows file associations or an application specified with `setPBSext`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

editADfile, makeADopts

---

editADfile   *Edit a File*

Description

Edit a file using the text editor specified in `.ADopts`.

Usage

```
editADfile(fname)
```

Arguments

fname	string name of file in current working directory (or elsewhere if path delimited by / or \).

Value

Returns Boolean: `TRUE` if file exists, `FALSE` if it does not.

Note

This function explicitly uses the editor chosen for PBSadmb. PBSmodelling has another function `openFile` that uses Windows file associations or an application specified with `setPBSext`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`editAD`, `makeADopts`

---

initAD   *Inititialize the ADMB Options List*

Description

Checks for te specified ADMB options file and if available reads it into R's memory.

Usage

```
initAD(optfile="ADopts.txt")
```

Arguments

optfile   string name of an options file in the user's working directory.

Details

..This function also creates the variable `.ADopts` in the global environment. Use `checkADopts` to ensure that the paths in `.ADopts` make sense.

Value

Returns a Boolean value, `TRUE` if the file was available and read into R's memory.

Note

A wrapper function that can be called from a GUI exists as `.win.initAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`readADopts`, `makeADopts`, `writeADopts`, `checkADopts`

---

linkAD   *Link Object Files to Make an Executable*

Description

Links the binary object file `'prefix'.o` to the ADMB libraries and produces the executable file `'prefix'.exe`.

Usage

```
linkAD(prefix, raneff=FALSE, safe=TRUE, logfile=TRUE, add=TRUE, verbose=TRUE)
```

Arguments

`prefix`	string name prefix of the ADMB project (e.g., `"vonb"`).
`raneff`	logical: use the random effects model, otherwise use the normal model.
`safe`	logical: if `TRUE`, use safe mode with bounds checking on all array objects, otherwise use optimized mode for fastest execution.
`logfile`	logical: if `TRUE`, create a log file of the messages from the shell call.
`add`	logical: if `TRUE`, append shell call messages to an exsiting log file.
`verbose`	logical: if `TRUE`, report the shell call an its messages to the R console.

Details

This function uses the C++ comiler declared in `.ADopts`. If `logfile=TRUE`, any errors will appear in `'prefix'.log`. If `verbose=TRUE`, they will appear in the R console.

Value

Invisibly returns the shell call and its messages.

Note

A wrapper function that can be called from a GUI exists as `.win.linkAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`convAD`, `compAD`, `makeAD`

---

makeAD   *Make an Executable Binary File from a C File*

Description

Essentially a wrapper function that calls in sequence: `convAD`, `compAD`, and `linkAD`.

Usage

```
makeAD(prefix, raneff=FALSE, safe=TRUE, logfile=TRUE, verbose=TRUE)
```

Arguments

prefix	string name prefix of the ADMB project (e.g., `"vonb"`).
raneff	logical: use the random effects model, otherwise use the normal model.
safe	logical: if `TRUE`, use safe mode with bounds checking on all array objects, otherwise use optimized mode for fastest execution.
logfile	logical: if `TRUE`, create a log file of the messages from the shell call.
verbose	logical: if `TRUE`, report the shell call an its messages to the R console.

Details

This function uses the C++ comiler declared in `.ADopts`. If `logfile=TRUE`, any errors will appear in `'prefix'.log`. If `verbose=TRUE`, they will appear in the R console.

Value

Returns nothing. The three functions called by `makeAD` each return the shell call and its messages.

Note

A wrapper function that can be called from a GUI exists as `.win.makeAD`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`convAD`, `compAD`, `linkAD`, `cleanAD`

---

makeADopts   *Creates the ADMB Options List*

Description

Creates a global list object detailing the pathways to the ADMB directory, the GCC bin, and the user's preferred text editor.

Usage

```
makeADopts(admpath, gccpath, editor, ver="gcc421")
```

Arguments

admpath   explicit path to the user's ADMB directory.

gccpath   explicit path to the user's GCC bin (C-compiler) directory.

editor   explicit path and program to use for editing text.

ver   optional C-compiler version number (not used at present).

Value

Creates a global, hidden list object called .ADopts.

Note

A wrapper function that can be called from a GUI exists as .win.makeADopts.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

initAD, makeADopts, writeADopts

---

plotMC   *Plot Results of MCMC Simulation*

Description

Plot results of an ADMB MCMC simulation using various plot methods.

Usage

```
plotMC(prefix, act="pairs", pthin=1, useCols=NULL)
```

Arguments

prefix	string name prefix of the ADMB project (e.g., `"vonb"`).
act	string scalar: action describing plot type (current choices: `"pairs"`, `"eggs"`, `"acf"`, `"trace"`, and `"dens"`).
pthin	numeric scalar indicating interval at which to collect records from the `.mc.dat` file for plotting.
useCols	logical vector indicating which columns of `.mc.dat` to plot.

Note

A wrapper function that can be called from a GUI exists as `.win.plotMC`. Use the PBSadmb GUI to explore these plots easily.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC, Canada

See Also

[runMC](#), [showADargs](#)

---

readADopts	*Reads an ADMB Options List into Memory From a File*

Description

Reads ADMB options into a global, hidden list object called `.ADopts` from an ASCII text file using `PBSmodelling::readList`).

Usage

```
readADopts(optfile="ADopts.txt")
```

Arguments

optfile	string name of an ASCII text file containing ADMB options information.

Value

No values returned. Reads the ADMB options into the list object `.ADopts`.

Note

A wrapper function that can be called from a GUI exists as `.win.readADopts`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

[initAD](#), [makeADopts](#),[writeADopts](#)

readRep	*Read an ADMB Report into R Memory*

Description

Import ADMB-generated report files into R's memory using the names of the report files to name the R-objects.

Usage

```
readRep(prefix, suffix=c(".cor",".rep",".std",".mc.dat"), global=FALSE)
```

Arguments

prefix   string name prefix of the ADMB project (e.g., `"vonb"`).

suffix   string scalar/vector specifying one or more suffixes.

global   logical: if `TRUE`, save the imported reports as objects to global environment using the same names as the report files.

Details

If the report object is one of `c(".cor", ".std", ".mc.dat")`, the report object is a data frame, otherwise it is a string vector. Multiple report objects are returned as a list of objects. A single report object is returned as the object itself.

This function attempts to detect the file format from a number of possibilities. For example, if the file has the special format recognized by PBSmodelling, then the function returns a list with named components. The example `vonb` included with this package shows how to write the template to get consistent variable names between ADMB and R. See the User's Guide for complete details.

Value

Invisibly returns the list of report objects. If only one report is imported, a single report object is returned.

Note

A wrapper function that can be called from a GUI exists as `.win.readRep`.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC, Canada

See Also

[editADfile](), .win.viewRep

---

runAD   *Run an Executable Binary File*

Description

Run the executable binary file 'prefix'.exe that was created by makeAD.

Usage

```
runAD(prefix, argvec="", logfile=TRUE, add=TRUE, verbose=TRUE)
```

Arguments

prefix	string name prefix of the ADMB project (e.g., "vonb").
argvec	string scalar/vector of arguments appropriate for the executable 'prefix'.exe.
logfile	logical: if TRUE, create a log file of the messages from the shell call.
add	logical: if TRUE, append shell call messages to an exsiting log file.
verbose	logical: if TRUE, report the shell call an its messages to the R console.

Details

This function typically reads the two files 'prefix'.dat and 'prefix'.pin, although in same cases one or both of these files may not be necessary.

If logfile=TRUE, output (including error messages, if any) will appear in 'prefix'.log. If verbose=TRUE, it will appear in the R console.

Value

Invisibly returns the results of the shell call.

Note

A wrapper function that can be called from a GUI exists as .win.runAD.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

runMC, makeAD, cleanAD

runMC	*Run an Executable Binary File in MCMC Mode*

Description

Run the executable binary file `'prefix'.exe`, created by `makeAD`, to generate MCMC simulations.

Usage

```
runMC(prefix, nsims=2000, nthin=20, outsuff=".mc.dat",
      logfile=FALSE, add=TRUE, verbose=TRUE)
```

Arguments

`prefix`	string name prefix of the ADMB project (e.g., `"vonb"`).
`nsims`	numeric scalar indicating number of MCMC simulations to perform.
`nthin`	numeric scalar indicating the sampling rate or thinning of the `nsims` MCMC simulations to report.
`outsuff`	string name suffix of the MCMC output data file.
`logfile`	logical: if `TRUE`, create a log file of the messages from the shell call.
`add`	logical: if `TRUE`, append shell call messages to an exsiting log file.
`verbose`	logical: if `TRUE`, report the shell call an its messages to the R console.

Details

This function runs `'prefix'.exe` twice, first with the arguments `-mcmc 'nsims' -mcsave 'nthin'` and second with the argument `-mceval`. By default, output goes to the file `'prefix'.mc.dat`, although a user can specify a different output suffix.

To see this function in action, use the PBSadmb GUI with the example `vonb` or `simpleMC`.

Value

Invisibly returns the results of the shell call.

Note

A wrapper function that can be called from a GUI exists as `.win.runMC`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

[runAD](), [makeAD](), [cleanAD]()

---

showADargs   *Show All Arguments for an ADMB Executable*

Description

Show all arguments available for an ADMB executable in the default text editor.

Usage

```
showADargs(prefix, ed=TRUE)
```

Arguments

prefix   string name prefix of the ADMB project (e.g., `"vonb"`).

ed   logical: if `TRUE`, write the ADMB arguments to a file and view them with the text editor, else
                display the arguments on the R console.

Value

Invisibly returns the argument list.

Note

A wrapper function that can be called from a GUI exists as `.win.showADargs`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

editADfile, runAD

---

startLog   *Start a Log File*

Description

Start a log file by removing any previous version and appending header information.

Usage

```
startLog(prefix)
```

Arguments

prefix   string name prefix of the ADMB project (e.g., `"vonb"`).

Value

No explicit value reurned. Writes header lines into a log file `'prefix'.log`.

Note

A wrapper function that can be called from a GUI exists as `.win.startLog`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`appendLog`, `editADfile`

`writeADopts`	*Writes the ADMB Options List from Memory to a File*

Description

Writes the global ADMB options list to a file in 'PBS' format (see `PBSmodelling::writeList`).

Usage

```
writeADopts(opts=.ADopts, optfile="ADopts.txt")
```

Arguments

`opts`	the global options list object `.ADopts`.
`optfile`	string name of the intended output file.

Value

Returns `opts` invisibly. Writes the options list object to an ASCII file.

Note

A wrapper function that can be called from a GUI exists as `.win.writeADopts`.

Author(s)

Jon T. Schnute, Pacific Biological Station, Nanaimo BC, Canada

See Also

`initAD`, `makeADopts`, `readADopts`

Index