Package ‘R2admb’

November 7, 2022

Title 'ADMB' to R Interface Functions

Version 0.7.16.3

Description A series of functions to call 'AD Model Builder' (i.e., compile and run models) from within R, read the results back into R as 'admb' objects, and provide standard accessors (i.e. coef(), vcov(), etc.)

Depends R (>= 3.0.1)

Imports coda, lattice

Suggests bbmle, lme4, ggplot2 (>= 3.4.0), testthat, knitr

SystemRequirements AD Model Builder <http://admb-project.org>

VignetteBuilder knitr

License GPL

LazyLoad yes

Repository CRAN

NeedsCompilation no

RoxygenNote 6.0.1

Author Ben Bolker [aut, cre],
Hans Skaug [aut],
Jeff Laake [aut]

Maintainer Ben Bolker <`bolker@mcmaster.ca`>

Date/Publication 2022-11-07 16:40:06 UTC

R topics documented:

R2admb-package *ADMB to R interface functions*

Description

A series of functions to call AD Model Builder (i.e., compile and run models) from within R, read the results back into R as "admb" objects, and provide standard accessors (i.e. coef(), vcov(), etc.)

Details

Package:	R2admb
Type:	Package
Version:	0.5
Date:	2009-11-11
License:	GPL
LazyLoad:	yes

More here!

Author(s)

Ben Bolker

Maintainer: Ben Bolker <bolker@ufl.edu>

References

http://www.admb-project.org

See Also

PBSadmb package, glmmADMB package, ADMB2R

admb_version — *Query ADMB version*

Description

Report on the version of ADMB being used.

Usage

```
admb_version()
```

Value

Prints the version string from a compiled ADMB file, and returns the value (invisibly) as a character vector; returns NA if ADMB is not installed

Author(s)

Ben Bolker

Examples

```
admb_version()
```

AIC.admb — *Standard accessor functions for ADMB model fits*

Description

Extract standard information such as log-likelihood, AIC, coefficients, etc. from ADMB model fits

Usage

```
## S3 method for class 'admb'
AIC(object, ..., k = 2)

## S3 method for class 'admb'
confint(object, parm, level = 0.95, method = "default",
  type = "fixed", ...)

## S3 method for class 'admb'
print(x, verbose = FALSE, ...)

## S3 method for class 'admb'
summary(object, correlation = FALSE, symbolic.cor = FALSE,
  ...)
```

```
## S3 method for class 'summary.admb'
print(x, digits = max(3, getOption("digits") - 3),
  symbolic.cor = x$symbolic.cor,
  signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'admb'
logLik(object, ...)

## S3 method for class 'admb'
coef(object, type = "fixed", ...)

## S3 method for class 'admb'
vcov(object, type = "fixed", ...)

stdEr(object, ...)

## S3 method for class 'admb'
stdEr(object, type = "fixed", ...)

## S3 method for class 'admb'
deviance(object, ...)
```

Arguments

`object`	an ADMB model fit (of class "admb")
`...`	other parameters (for S3 generic compatibility)
`k`	penalty value for AIC fits
`parm`	(currently ignored: FIXME) select parameters
`level`	alpha level for confidence interval
`method`	(character): "default" or "quad", quadratic (Wald) intervals based on approximate standard errors; "profile", profile CIs (if profile was computed); "quantile", CIs based on quantiles of the MCMC-generated posterior density (if MCMC was computed); "HPDinterval", CIs based on highest posterior density (ditto)
`type`	which type of parameters to report. Character vector, including one or more of "fixed" or "par" (standard, fixed-effect parameters); "random" (random effect parameters); "rep" (report variables); "sdrpt" (sdreport variables); "extra" (report and sdreport); "all" (all of the above).
`x`	an ADMB model fit (of class "admb")
`verbose`	show messages
`correlation`	currently unused parameter
`symbolic.cor`	currently unused parameter
`digits`	number of digits to display
`signif.stars`	show significance stars?

Value

Extracts appropriate values: numeric (scalar) for AIC, type logLik for logLik, numeric vector of coefficients, numeric variance-covariance matrix of parameter estimates

Examples

```
admbex <- system.file("doc","Reedfrog_runs.RData",package="R2admb")
load(admbex)
m1
coef(m1)
summary(m1)
coef(summary(m1)) ## returns just z-table
AIC(m1)
vcov(m1)
logLik(m1)
deviance(m1)
stdEr(m1)
```

---

compile_admb *Compile ADMB files, run, read output*

Description

With various tests, calls the admb script to compile from a TPL file to an executable, or runs the resulting executable

Usage

```
compile_admb(fn,safe=FALSE,re=FALSE,
verbose=FALSE,
admb_errors=c("stop","warn","ignore"))

 run_admb(fn,verbose=FALSE,mcmc=FALSE,
mcmc.opts=mcmc.control(),profile=FALSE,
extra.args="",admb_errors=c("stop","warn","ignore"))

read_admb(fn,verbose=FALSE,profile=FALSE,
mcmc=FALSE,mcmc.opts=NULL,admbOut=NULL,checkterm=TRUE)
```

Arguments

fn	(character) name of TPL file, without extension
safe	(logical) Compile in safe mode?
re	(logical) Compile in random effects (ADMB-RE) mode?
verbose	(logical) Verbose output?

`admb_errors`	(character) how to handle compilation/linking errors?
`profile`	(logical) Run likelihood profiles?
`extra.args`	(character) extra arguments for ADMB run
`mcmc`	(logical) run post-hoc MCMC?
`mcmc.opts`	options for MCMC run (see `mcmc.control`)
`admbOut`	(character) ADMB run output for inclusion in `admb` object (for internal use)
`checkterm`	(logical) compute termination criteria (ratio of min/max eigenvalue) and include it in the saved object?

Value

- `compile_admb` returns nothing (it has the side effect of creating an executable)
- `run_admb` invisibly returns the output produced by the ADMB run; it also produces output files on disk as a side effect
- `read_admb` returns an object of class `admb`, containing as much information as possible gleaned from the output files (parameter estimates, standard errors, variance-covariance matrix, profiles, MCMC output)

Note

Compiling also sets executable mode.

Author(s)

Ben Bolker

`do_admb` *Compile and/or run an ADMB model, collect output*

Description

Compile an ADMB model, run it, collect output

Usage

```
do_admb(fn, data = NULL, params = NULL, bounds = NULL, phase = NULL,
  re = NULL, data_type = NULL, safe = TRUE, profile = NULL,
  profile.opts = NULL, mcmc = NULL, mcmc.opts = mcmc.control(),
  impsamp = FALSE, verbose = FALSE, run.opts = run.control(),
  objfunname = "f", workdir = getwd(), admb_errors = c("stop", "warn",
  "ignore"), extra.args)
```

Arguments

`fn`	(character) base name of a TPL function, located in the working directory
`data`	a named list of input data variables (order must match TPL file): each element of the list can either be a single value, or a list containing elements • valuethe value of the data • data_typecharacter: possible values as in `storage.mode`, typically "integer" or "numeric": this overrides R2admb's attempts to guess whether variables are supposed to be integers or floats (default NA)
`params`	a named list of starting parameter values (order must match TPL file): each element of the list can either be a single value, or a list containing elements **value** starting value of the parameter (default 0) **bounds** two-element vector of lower and upper bounds **phase** integer, specifying phase: not implemented yet
`bounds`	named list of 2-element vectors of lower and upper bounds for specified parameters
`phase`	named numeric vector of phases (not implemented yet)
`re`	a named list of the identities and dimensions of any random effects vectors or matrices used in the TPL file
`data_type`	a named vector specifying (optional) data types for parameters, in parname="storage mode" format (e.g. `c(x="integer",y="numeric")`)
`safe`	(logical) compile in safe mode?
`profile`	(logical) generate likelihood profiles? (untested!)
`profile.opts`	(list) list of options, including • parsvector of names of parameters to profile
`mcmc`	(logical) run MCMC around best fit?
`mcmc.opts`	options for MCMC (see `mcmc.control` for details)
`impsamp`	(logical) run importance sampling?
`verbose`	(logical) print details
`run.opts`	options for ADMB run (see `run.control` for details)
`objfunname`	(character) name for objective function in TPL file (only relevant if `checkparam` is set to "write")
`workdir`	temporary working directory (dat/pin/tpl files will be copied)
`admb_errors`	how to treat ADMB errors (in either compilation or run): use "ignore" option at your own risk!
`extra.args`	(character) extra argument string to pass to admb

Details

`do_admb` will attempt to do everything required to start from the model definition (TPL file) specified by `fn`, the data list, and the list of input parameters, compile and run (i.e. minimize the objective function of) the model in AD Model Builder, and read the results back into an object of class `admb` in R. If checkparam or checkdata are set to "write", it will attempt to construct a DATA section, and construct or (augment an existing) PARAMETER section (which may contain definitions of non-input parameters to be used in the model). It copies the input TPL file to a backup (.bak); on finishing, it restores the original TPL file and leaves the auto-generated TPL file in a file called [fn]_gen.tpl.

Value

An object of class admb.

Note

1. Mixed-case file names are ignored by ADMB; this function makes a temporary copy with the file name translated to lower case. 2. Parameter names containing periods/full stops will not work, because this violates C syntax (currently not checked). 3. There are many, many, implicit restrictions and assumptions: for example, all vectors and matrices are assumed to be indexed starting from 1.

Author(s)

Ben Bolker

Examples

```
## Not run:
setup_admb()
file.copy(system.file("tplfiles","ReedfrogSizepred0.tpl",package="R2admb"),"tadpole.tpl")
tadpoledat <-
 data.frame(TBL = rep(c(9,12,21,25,37),each=3),
            Kill = c(0,2,1,3,4,5,0,0,0,0,1,0,0,0,0L),
            nexposed=rep(10,15))
m1 <- do_admb("tadpole",
             data=c(list(nobs=15),tadpoledat),
             params=list(c=0.45,d=13,g=1),
             bounds=list(c=c(0,1),d=c(0,50),g=c(-1,25)),
             run.opts=run.control(checkparam="write",
               checkdata="write",clean="all"))
m2 <- do_admb("tadpole",
             data=c(list(nobs=15),tadpoledat),
             params=list(c=list(0.45,bounds=c(0,1)),
                         d=list(13,bounds=c(0,50)),
                         g=list(1,bounds=c(-1,25))),
             run.opts=run.control(checkparam="write",
               checkdata="write",clean="all"))
unlink("tadpole.tpl")

## End(Not run)
```

`extract_gradient` *Extract gradients*

Description

Extract gradient values from last iteration of screen output and return dataframe with variable names, values and gradient, sorted in order of ascending absolute value of the gradient.

Usage

```
extract_gradient(admbfile)
```

Arguments

`admbfile`	base name of admb project

Value

a dataframe with 3 columns var=variable name, value= final parameter value, gradient= gradient value

Author(s)

Jeff Laake

`find_large_cor` *Find large correlations*

Description

Find any correlations for which their absolute value exceeds a specified amount (rho). Returns a dataframe with row and column names and correlation from lower triangular matrix.

Usage

```
find_large_cor(x, rho = 0.9)
```

Arguments

`x`	correlation matrix
`rho`	abolute value for lower bound of correlation

Value

a dataframe with 3 columns var1=row name, var2= column name or number, Value of matrix element. Only contains rows in which matrix element satisfies logical expression.

Author(s)

Jeff Laake

`mcmc.control` *Control options for MCMC after ADMB fitting*

Description

Determines the options (number of steps, save interval, etc.) for running MCMC based on the estimated mode (maximum likelihood estimate) and parameter variance-covariance matrix

Usage

```
mcmc.control(mcmc = 1000, mcmc2 = 0, mcsave, mcnoscale = FALSE,
  mcgrope = FALSE, mcmult = 1, mcmcpars = NULL)
```

Arguments

`mcmc`	Total number of MCMC steps
`mcmc2`	MCMC2 steps (see ADMB-RE manual)
`mcsave`	Thinning interval for values saved in the PSV file. Default is `pmax(1,floor(mcmc/1000))`, i.e. aim to save 1000 steps
`mcnoscale`	don't rescale step size for mcmc depending on acceptance rate
`mcgrope`	(double) Use a candidate distribution that is a mixture of a multivariate normal and a fatter-tailed distribution with a proportion `mcmcgrope` of the fatter-tailed distribution; the ADMB manual suggests values of `mcgrope` between 0.05 and 0.1
`mcmult`	Multiplier for the MCMC candidate distribution
`mcmcpars`	(character) vector of parameters to track in MCMC run. *At least one must be specified.* ADMB produces two kinds of output for MCMC. For any `sdreport` parameters it will produce a `hst` file that contains a summary histogram; `mcmcpars` constructs appropriate `sdreport` parameters in the auto-generated TPL file. Step-by-step output for all parameters (regulated by `mcsave`) is saved in the `PSV` file.

Details

See the AD Model Builder reference manual. The `mcrb` option (reduce correlation of the Hessian when constructing the candidate distribution) and the `mcseed` options (seed for random number generator) are not yet implemented; `mcnoscale` above may not work properly

Value

Returns a list of options suitable for passing as the `mcmc.opts` argument to `do_admb`

Note

Some options (`mcmc2`, etc.) that can be used in AD Model Builder and ADMB-RE may not be available

Author(s)

Ben Bolker

Examples

```
mcmc.control(mcmc=2000)
```

`plot.admb_hist` *Plot MCMC histogram*

Description

Plot MCMC histogram

Usage

```
## S3 method for class 'admb_hist'
plot(x,type=c("lattice","ggplot"),dtype=c("hist","density"),pars,...)
```

Arguments

`x`	plotting data
`type`	only "lattice" at present
`dtype`	either "hist" or "density"
`pars`	passed to rhist
`...`	additional parameters for compatibility

Value

plot object

`read_pars` *Read in parameters from an AD Model Builder run*

Description

Reads coefficients, standard errors, log-likelihoods, maximum gradients, correlation and variance-covariance matrices from AD Model Builder output files

Usage

```
read_pars(fn, drop_phase = TRUE, covfn = "admodel.cov",
  warn_nonstd_rep = TRUE)

read_psv(fn, names = NULL)

read_rep(fn, names = NULL, warn_nonstd_rep = TRUE)
```

Arguments

`fn`	(character) Base name of AD Model Builder
`drop_phase`	(logical) drop negative-phase (fixed) parameters from results?
`covfn`	(character) file name for covariance matrix information
`warn_nonstd_rep`	warn if report file is in nonstandard format?
`names`	(character) Names of variables

Details

Given the output from an ADMB run on FOO.tpl, `read_pars` reads the files FOO.par (parameters, log-likelihood, max gradient); FOO.std (standard deviations); FOO.cor (correlations); FOO.rep (report variables); admodel.hes for hessian; and admodel.cov for covariance matrix. `read_psv` reads the output of MCMC runs.

`read_rep` (called by `read_admb` to read the `.rep` file) first checks if the report file is in a standard format: first line starts with a comment character (#); thereafter, each block starts with a single commented line containing the name of the parameter (possibly ending with a colon), followed by a block of all-numeric lines, which are read as a single vector. If the report file is in a standard format, the values are added to the end of the coefficients list. Otherwise, the numeric values from the report file are included in the results as a single, concantenated numeric vector.

Value

List containing the following elements

- coefficientsparameter estimates
- coeflistparameter estimates in list format, with proper shape (vectors, matrices, etc.)

- seestimated standard errors of coefficients
- logliklog-likelihood
- maxgradmaximum gradient of log-likelihood surface
- corcorrelation matrix
- vcovvariance-covariance matrix
- nparnumber of parameters
- heshessian matrix (only if no vcov matrix)
- reportvalues from report file (if non-standard report file)

Warnings

- The `coeflist` component is untested for data structures more complicated than scalars, vectors or matrices (i.e. higher-dimensional or ragged arrays)
- Because ADMB hard-codes the file name for covariance matrix information (`admodel.cov`), care is necessary when running different models in the same directory; users may want to rename this file by hand and use the `covfn` argument

See Also

`write_pin`, `write_dat`

`read_plt` *Read in ADMB profile file*

Description

Read in the output from ADMB likelihood profiling stored in a `.plt` file

Usage

```
read_plt(varname)
```

Arguments

`varname`	(character) Name of profiled variable (base name of `.plot` file)

Value

List containing the following elements:

`prof`	likelihood profile: a two-column matrix containing the parameter value and the corresponding likelihood (*not* the log-likelihood or negative log-likelihood), scaled to integrate to 1.0
`ci`	matrix of upper and lower confidence intervals at the 0.9, 0.95, and 0.975 levels
`prof_norm`	likelihood profile, based on a normal approximation
`cinorm`	confidence interval matrix, based on normal approximation

run.control *set run options for running ADMB via R*

Description

A helper function

Usage

```
run.control(check_tpl = TRUE, write_files = TRUE, checkparam = c("stop",
  "warn", "write", "ignore"), checkdata = c("stop", "warn", "write",
  "ignore"), compile = TRUE, run = TRUE, read_files = TRUE,
  clean_files = "all")
```

Arguments

`check_tpl`	Check the specified TPL file for problems?
`write_files`	Write out data and initialization files?
`checkparam`	How to check `PARAMETERS` section of the TPL file: `stop`=stop if there are problems; `warn`=give a warning if there are problems, but try to proceed; `write`=modify TPL file, writing appropriate sections; `ignore`=assume TPL file is OK, proceed
`checkdata`	as with `checkparam`: how/whether to check/generate the `DATA` section of the TPL file
`compile`	compile the TPL file (via ADMB) into an executable?
`run`	run the executable file with the specified data/initial values?
`read_files`	read the results of an ADMB run into R?
`clean_files`	Delete working files after completion of the run? Options are `"all"`, `"sys"`, `"output"`, `"none"`; `TRUE` is equivalent to `"all"` and `FALSE` is equivalent to `"none"`

Value

A list with appropriate default values inserted for passing to `do_admb`

Author(s)

Ben Bolker

setup_admb — *Set up AD Model Builder environment variables*

Description

Attempts to set environment variables so that AD Model Builder will "just work" when run from inside R

Usage

```
setup_admb(admb_home)

       clean_admb(fn,which=c("sys","output"))
```

Arguments

`admb_home`	(character) directory containing AD Model Builder binary files
`fn`	(character) base name of ADMB model files
`which`	what to remove: any combination of "sys" (system), "input", "output", or "all" or "none"

Details

(1) If the environment variable ADMB_HOME is not already set and admb_home is not specified, this function will try to set it sensibly. (I.e., on Unix systems, it will run a "locate" command (if one is available) to try to find the binaries, and thereafter check if they are installed in the default location (/usr/local/admb); on Windows it will assume they are installed in the default location (C:/ADMB).) (2) If ADMB_HOME is set and admb_home is not specified, it will leave the original setting alone. (3) If admb_home is specified, it will set the environment variable ADMB_HOME to that value.

The function also prepends the admb_home value to the PATH variable.

Value

A character vector containing the name of the current ADMB_HOME.

Author(s)

Ben Bolker

Examples

```
 orig <- Sys.getenv("ADMB_HOME")
## this doesn't make sense but won't break anything
## until you actually try to run AD Model Builder
 setup_admb("elsewhere")
 Sys.setenv(ADMB_HOME="") ## erase environment variable
## Not run:
 setup_admb()             ## auto-locate (fails if ADMB not found)

## End(Not run)
 Sys.setenv(ADMB_HOME=orig) ## restore sanity
```

write_pin — *Write parameter and data files for ADMB*

Description

Given base filenames and lists, write output files for starting parameter values and data in a format suitable for input by AD Model Builder from glmmADMB, by Hans Skaug

Usage

```
write_pin(name,L)

      write_dat(name, L, append=FALSE)
```

Arguments

`name`	(character) the base name of the file
`L`	a list of objects to be written to file
`append`	(logical) append to existing file?

Value

Returns nothing; creates files in the current working directory as a side effect

Note

numeric vectors and matrices are the only objects that can be written (at present)

Author(s)

Hans Skaug

See Also

`read_pars`

Index