Almost complete list of ADMB types

3darray	4darray
5darray	6darray
7darray	SPinit_3darray
SPinit_4darray	SPinit_bounded_3darray
SPinit_4darray	SPinit_bounded_3darray
SPinit_bounded_matrix	SPinit_bounded_number
SPinit_bounded_vector	SPinit_imatrix
SPinit_int	SPinit_ivector
SPinit_matrix	SPinit_number
SPinit_vector	SPint
SPivector	SPmatrix
SPnumber	SPvector
constant_quadratic_penalty	dll_3darray
dll_adstring	dll_imatrix
dll_init_3darray	dll_init_bounded_number
dll_init_bounded_vector	dll_init_imatrix
dll_init_int	dll_init_matrix
dll_init_number	dll_init_vector
dll_int	dll_matrix
dll_number	dll_random_effects_vector
dll_vector	equality_constraint_vector
gaussian_prior	imatrix
inequality_constraint_vector	init_3darray
init_4darray	init_5darray
init_6darray	init_7darray
init_adstring	init_bounded_3darray
init_bounded_dev_vector	init_bounded_matrix
init_bounded_matrix_vector	init_bounded_number
init_bounded_number_vector	init_bounded_vector
init_bounded_vector_vector	init_imatrix
init_int	init_ivector
init_line_adstring	init_matrix
init_matrix_vector	init_number
init_number_vector	init_vector
init_vector_vector	int
ivector	likeprof_number
matrix	normal_prior
number	objective_function_value
quadratic_penalty	quadratic_prior
random_effects_bounded_matrix	random_effects_bounded_vector
random_effects_matrix	random_effects_vector
sdreport_matrix	sdreport_number
sdreport_vector	splus_3darray
splus_adstring	splus_imatrix
splus_init_3darray	splus_init_bounded_number
splus_init_bounded_vector	splus_init_matrix
splus_init_number	splus_init_vector
splus_int	splus_matrix
splus_number	splus_vector
vector	

Command-line Options

-ainp NAME change default ascii input parameter file name to NAME

-binp NAME change default binary input parameter file name to NAME

-est only do the parameter estimation

-noest do not do the parameter estimation (optimization)

-ind NAME change default input data file name to NAME

-lmn N use limited memory quasi newton – keep N steps

-dd N check derivatives after n function evaluations

-lprof perform profile likelihood calculations

-maxph N increase the maximum phase number to N

-mcdiag use diagonal covariance matrix for mcmc with diagonal values 1

-mcmc [N] perform markov chain monte carlo with N simulations

-mcmult N multiplier N for mcmc default

-mcr resume previous mcmc

-mcrb N reduce the amount of correlation in the covariance matrix 1<=N<=9

-mcnoscale don't rescale step size for mcmc depending on acceptance rate

-nosdmcmc turn off mcmc histogram calcs to make mcsave run faster

-mcgrope N use probing strategy for mcmc with factor N

-mcseed N seed for random number generator for markov chain monte carlo

-mcscale N rescale step size for first N evaluations

-mcsave N save the parameters for every N'th simulation

-mceval Go throught the saved mcmc values from a previous mcsave

-crit N1,N2,... set gradient magnitude convergence criterion to N

-iprint N print out function minimizer report every N iterations

-maxfn N1,N2,... set maximum number opf function eval's to N

-rs if function minmimizer can't make progress rescale and try again

-nox don't show vector and gradient values in function minimizer screen report

-phase N start minimization in phase N

-simplex use simplex algorithm for minimization (new test version)

-nohess don't do hessian or delta method for std dev

-eigvec calculate eigenvectors of the Hessian

-sdonly do delta method for std dev estimates without redoing hessian

-ams N set arrmblsize to N (ARRAY_MEMBLOCK_SIZE)

-cbs N set CMPDIF_BUFFER_SIZE to N (ARRAY_MEMBLOCK_SIZE)

-mno N set the maximum number of independent variables to N

-mdl N set the maximum number of dvariables to N

-master run as PVM master program

-gbs N set GRADSTACK_BUFFER_SIZE to N (ARRAY_MEMBLOCK_SIZE)

-master run as PVM master program

-slave run as PVM slave program

-pvmtime record timing information for PVM performance analysis

-info Contributors acknowledgements

-nr N maximum number of Newton-Raphson steps

-imaxfn N maximum number of fevals in quasi-Newton inner optimization

-is N set importance sampling size to n for random effects

-isf N set importance sampling size funnel blocksto n for random effects

-isdiag print importance sampling diagnostics

-hybrid do hybrid Monte Carlo version of MCMC

-hbf set the hybrid bounded flag for bounded parameters

-hyeps mean step size for hybrid Monte Carlo

-hynstep number of steps for hybrid Monte Carlo

-noinit do not initialize random effects before inner optimzation

-ndi N set maximum number of separable calls

-ndb N set number of blocks for derivatives for random effects (reduces temporary file sizes)

-ddnr use high precision Newton-Raphson for inner optimization for banded hessian case ONLY even if implemented

-nrdbg verbose reporting for debugging newton-raphson

-mm N do minimax optimization

-shess use sparse Hessian structure inner optimzation

-? list of options

-help list of option

Handy-dandy Reference Card

John Sibert Anders Nielsen

August 13, 2009

ADMB Syntax Notes

- `_SECTIONS` must begin in the first position of a line.
- Other ADMB statements and variable declarations must begin in the third position of the line or beyond.
- `LOCAL_CALCS` and `END_CALCS` must begin in the second position.
- Semicolons are not required after statements in either the `DATA_SECTION` or the `PARAMETER_SECTION`, but should be used elsewhere. Since semicolons are required by C++, it is harmless to use them everywhere.
- `!!` preceding a statement indicates a line of C++ code that will be passed unchanged to the C++ compiler; a semicolon must end the line.
- Template file errors are indicated by the line number in the template file and the first letter of the offending text.
- Beware of editors intended for word processing (e.g. WordPad) that may insert extra invisible formatting characters in files.
- Some Windows versions of ADMB may require a blank line after the last line of the program.
- Adopt a consistent programming style. Here are a couple of reasonable sets of guidlines: `http://corelinux.sourceforge.net/cppstnd/cppstnd.php` and `http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml`.
- Avoid directories (folders) and file names that contain spaces. They will only cause grief and tears. Some operating systems do not distinguish between upper-case and lower-case letters, so it's best not to mix.

Example Template File (linear regression)

```
DATA_SECTION
  init_int N
  init_vector x(1,N)
  init_vector Y(1,N)

PARAMETER_SECTION
  init_number a
  init_number b
  init_number logSigma
  sdreport_number ss
  objective_function_value nll

PROCEDURE_SECTION
  ss=exp(2.0*logSigma);
  nll=0.5*(N*log(2.0*M_PI*ss)+sum(square(Y-(a+b*x)))/ss);
```

Compile: `makeadm <programname>` creates a executable
Run: `<programname>` runs the executable

Template File SECTIONs

Sections are discussed in detail in the ADMB manual. Every ADMB program must contain these three sections.

`DATA_SECTION` Describes data and specifies how they are read and possible transformed.

`PARAMETER_SECTION` Describes model parameters, valid ranges and sequence of estimation. The variable holding the value of the objective function is specified here.

`PROCEDURE_SECTION` Contains the details of the model and the likelihood computation. Semi-colons are required at the end of each statement.

Other sections have specialized purposes.

`FUNCTION` Begins definition of a function or "method" in the `PROCEDURE_SECTION` `LOCAL_CALCS` and `END_CALCS` bracket C++ code transmitted without modification to the compiler. Semi-colons are required at the end of each statement.

`INITIALIZATION_SECTION` Used to initialize parameters declared in the `PARAMETER_SECTION`.

`REPORT_SECTION` Used to create a customized report. Uses the pre-defined `ofstream` variable `report` for output. For example `report << "a = " << a << endl;` would place the value of the variable `a` in the file `<programname>.rep`.

`RUNTIME_SECTION` Used to control the behavior of the function minimizer. Useful to change stopping criteria during initial phases of an estimation.

`PRELIMINARY_CALCULATIONS_SECTION` or `PRELIMINARY_CALCS_SECTION` Intended to do preliminary calculations on the data prior to starting the model. Largely supplanted by `LOCAL_CALCS` and `END_CALCS` code fragments.

`BETWEEN_PHASES_SECTION` Code executed between estimation phases.

`GLOBALS_SECTION` Used to insert any valid C++ statements prior to the defination of the `main()` function. Useful to include header files and to declare global objects.

`TOP_OF_MAIN_SECTION` Used to set AUTODIF global variables. Useful to reduce size of temporary gradient files.

`FINAL_SECTION`

`SLAVE_SECTION`

ADMB Variable Types

ADMB uses two fundamental data types: the standard C++ `double` for which no derivative information is generated, and the AUTODIF library `dvariable` for which derivative information is generated. See the AUTODIF manual for details. The prefix `init_` in the `DATA_SECTION` tells ADMB to read the value of the variable from the file `<programname>.dat`. The prefix `init_` in the `PARAMETER_SECTION` tells ADMB to estimate the value of the parameter using the model. Qualifiers in brackets `[...]` are optional. Refer to the ADMB manual for a complete descriptions.

Declaration in tpl	DATA_SECTION	PARAMETER_SECTION
[init_]int	int	int
[init_][bounded_]number	double	dvariable
[init_][bounded_][dev_]vector	dvector	dvar_vector
[init_][bounded_]matrix	dmatrix	dvar_matrix
[init_]3darray	3D double array	3D dvariable array
4darray	4D double array	4D dvariable array
5darray	5D double array	5D dvariable array
6darray	6D double array	6D dvariable array
7darray	7D double array	7D dvariable array
sdreport_number	na	dvariable
likeprof_number	na	dvariable
sdreport_vector	na	dvar_vector
sdreport_matrix	na	dvar_matrix

ADMB Utilities

`ADMB_HOME` environment variable is the folder in which ADMB was installed. Used by other utilities, compilers and make files to access ADMB

`makeadm` and `makeadms` build executable files from `.tpl` files. The terminal letter 's' invokes the "safe" library for subscript checking.

`tpl2cpp` and `tpl2rem` translate `.tpl` to C++ code.

`mygcco` `mygccs` `mygccopt` `mygcco-re` comple ADMB c++ code into object files.

`linkadm` and `linkadms` link ADMB object files into executables.

`admb` Makes ADMB applications. Takes -s -r and -d command line options. Type `admb --help` for more information.

`adcomp` and `adlink` Used by `admb` for compiling and linking.

`check-expected-results` I have no idea what this does.

The following should probably not be mentioned: `sed1.exe` `sedcmd` `sedcmd2` `sedcmd3` `seddf1b2` `seddf1b3` `seddf1b4` `sed.exe` `sedf1b2a` `sedf1b2c` `sedf1b2d` `sedflex`

ADMB Files

`<programname>.tpl` ADMB template file specifying a complete model.

`<programname>.dat` The default file for reading in data to an ADMB application.

`<programname>.htp` and `<programname>.cpp` C++ header and source code files produced by `tpl2cpp`.

`<programname>.par` Estimated values of parameters.

`<programname>.std` Estimated values of parameters and the standard deviations of the parameter estimates computed by the the inverse Hessian method.

`<programname>.cor` Correlation matrix of the estimated parameters.

`<programname>.pin` File containing the initial values of the paramters to be used to start the numerical estimation. Format is the same as `<programname>.par`

Run-time Interventions

Pressing Control-C (press the control key and then C) after the minimizer has started will interrupt the program and cause it to display the prompt, "`press q to quit or c to invoke derivative checker:`". Pressing 'q' and then "Return" (or "Enter"), will cause the program to leave the minimizer and enter the report phase. Pressing 'c' and then "Return" will invoke the derivative checker and additional prompts will be displayed.

