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

Functions

ADMB contains a very large number of functions. "Pseudo-prototypes" of some commonly used functions showing the general argument types and return types. ADMB has multiple overloads of many functions with different combinations of argument types.

```
number gammln(number);
vector gammln(vector);
number square(number);
number cube(number);
```

A more complete, but only partially documented, list of functions can be found in rhe "Modules" tab of the draft API documentation at http://admb-project.org/documentation/api/doxygen

Matrix and vector operations

The syntax of ADMB Matrix and vector operations follows normal mathematical conventions as much as possible. If u and v are vectors and M is a matrix, `u*M` is a normal matrix multiplication and `u*v` is a dot product. Element-wise multiplications and divisions are accomplished by `elem_prod(u, v)` and `elem_div(u, v)` respectively. Both functions return a vector. `inv(M)` returns the inverse of a matrix. `trans(M)` returns the transpose of a matrix. `det(M)` returns the determinant of a matrix. `norm(...)` returns the norm of a vector or matrix. `norm2(...)` returns the square of the norm of a vector or matrix.

Gratuitous Advice

- Beware of editors intended for word processing (e.g. WordPad) that may insert extra invisible formatting characters in files.
- 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.

Handy-dandy Reference Card

John Sibert Anders Nielsen

September 25, 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.

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. Must include a declaration of one instance of a variable of type `objective_function_value`.

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
objective_function_value	na	dvariable

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. The folder `$ADMB_HOME\bin` contains executables files that can be used to build ADMB applications:

`admb` Very handy tool for building 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.

`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 Files

Every ADMB application requires a template file and a data file.

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

`<programname>.dat` The default ADMB data file. 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.

Command-line Options

A large number of command line options are available for ADMB applications. For a complete list, type `<programname> -?`. A few commonly used options are given below:

`-est` do the parameter estimation only (skip Hessian and S. D. computations)

`-dd N` derivatives after n function evaluations

`-lprof` profile likelihood calculations

`-mcmc [N]` chain monte carlo with N simulations

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

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