AD Model Builder Introductory Workshop

http://admb-project.org/

Getting Started with ADMB

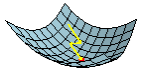
ADMB Foundation

sibert@hawaii.edu

Getting Started

What you must have:

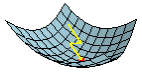
1. Good text editor – **NOT** a word processing program such as MS Word or Wordpad.
2. C++ compiler
3. ADMB utilities and libraries
4. Superficial familiarity with C++ helps a lot.



Installation — 1

Installation Steps

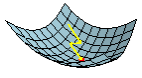
- Unpack archive into (any) directory. The archive contains the correct directory structure that the ADMB utilities expect.
- Create the environment variable `ADMB_HOME` so that it points to the installation directory.
- Modify existing `PATH` environment variable so that it includes `ADMB_HOME\bin`.
- Make sure the compiler looks in `ADMB_HOME\include` and `ADMB_HOME\lib` for include files and libraries.



Installation — 2

Easy Installation in Windows

- Installation is always dependent on operating system and compiler, so ...
- The ADMB Project offers an specialized packages and installation scripts for different OS and compiler combinations.
- The self extracting Windows archive is an integrated package that includes a free compiler, MingGW, and the appropriate ADMB libraries.
- The setup script at http://admb-project.googlecode.com/files/admb-9.0.363-win32.exe will correctly install **everything** and create a specialized command line window to run ADMB utilities.



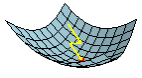
Creating an ADMB application

1. Create an ADMB script using a text editor and save in a file with the suffix `.tpl`, for instance `simple.tpl`.
2. Translate the the `.tpl` into C++ using `tpl2cpp` (or `tpl2rem` for random effects models), i.e. `tpl2cpp simple`.
3. Compile the resulting `simple.cpp` into an object file, `simple.obj`.
4. Link the object file with the ADMB libraries to create an executable file.

ADMB utilities will do these steps with a single command. For instance, type

```
admb simple
```

to create the `simple` ADMB example application from `simple.tpl`.

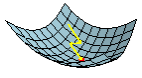


Syntax

```
// Simple linear model
DATA_SECTION
  init_int N
  !! cout << N << endl;
  init_vector Y(1,N)
  init_vector x(1,N)
 LOCAL_CALCS
   cout << "X = " << X << endl;
   cout << "Y = " << Y << endl;
 END_CALCS
PARAMETER_SECTION
  init_number a
  init_number b
  init_number logSigma;
  sdreport_number sigma;
  objective_function_value nll

PROCEDURE_SECTION
  sigma=exp(logSigma);
  nll=0.5*(N*log(2*M_PI*square(sigma))+
      sum(square(Y-(a+b*x)))/square(sigma));
```

- `DATA_SECTION`, `PARAMETER_SECTION`, and `PROCEDURE_SECTION` must be present in every valid `.tpl` file.
- `_SECTION` keywords must begin in the first character of a line.
- Statements must begin on at least third character of a line.
- Semicolons `;` are required in the `PROCEDURE_SECTION` and in all C++ statements.
- Program statements beginning with `!!` are C++ statements passed compiler without modification.
- "Sections" of code beginning with `LOCAL_CALCS` and ending with `END_CALCS` are C++ statements passed to the compiler without modification. `LOCAL_CALCS` and `END_CALCS` must begin in the second character of a line.
- C++ comments can be used anywhere in a `.tpl` file; they are not passed to the compiler.



Exercises

- Modify and test the simple linear regression example to verify that the data input is correct.
- Modify the example to attempt to print the value of x(0), i.e. add `!!cout << x(0) << endl;` to the `DATA_SECTION`.
  What happens when you try to run the program?
- Rebuild the example using `admb -s simplelm`.
  Now what happens when you run the program?