Notes on AD Model Builder by a novice

Ben Bolker

November 10, 2010

Mostly created 11 August 2009, not significantly updated since then. Some issues may have been resolved.

This document describes my discoveries and frustrations as I started to learn how to use AD Model Builder (and ADMB-RE). Note to ADMB developers: please take this all as constructive criticism! ADMB is a great tool and I hope these comments will contribute to its development and ease of uses for others . . .

1. I got quite confused by the distinction between `tpl2cpp` and `tpl2rem`: I now know that these are used for constructing models without and with random effects, respectively, but this wasn't clear initially when I was poking around in the `bin` directory trying to figure out what to do. There seem to be a confusing number of different possible scripts and wrappers: the ADMB-RE manual refers to an `admb` script, which didn't come with the binary version of ADMB that I downloaded (despite the fact that the binary did allow the construction of ADMB-RE models). I eventually downloaded from `http://admb-project.org/community/editing-tools/admb-ide/scripts-linux.zip/view?searchterm=Scripts` and got the `admb` script. Appendix C of the ADMB-RE documentation is really useful . . . it might also be useful to mention `adlink` and `adcomp`, which are used in the `admb` script . . . a Makefile would be nice (so one could rebuild only the necessary pieces), but it might be overkill and might be too hard to make it properly cross-platform (since Windows users etc. would have to get a make executable from somewhere).

   Probably most of this confusion would have been avoided if the binary distribution I downloaded had actually had an `admb` script, and if that was consistently referred to throughout both the ADMB and the ADMB-RE manual. ADMB manual only mentions `tpl2cpp` (of course, since it preceded ADMB-RE), ADMB-RE mentions both but never *explicitly* says exactly what they are.

2. At one point I thought that file names with dashes were somehow broken, but now I think I must just have been confused — can't replicate.

3. ADMB is case **sensitive** in a slightly nasty way, which must come from its Windows/DOS heritage but is a little odd on a *nix system. In particular,

it converts file names to lower case — if the TPL file is called `Simple.tpl`, it will look for a DAT file called `simple.dat` (and fail to find one called `Simple.dat` ...

4. If parameters are specified, the order in the PIN file must match the order in which they are defined in the TPL file ... this is perhaps obvious, but screwed me up a few times (I have gotten lazy and used to the expectation that things are named and matched up according to their names). Seems one could write an automatic checker for this reasonably easily?

5. ADMB "does the right thing" with scalar*vector arithmetic (and vector*matrix and ?? vector+vector), but one must remember to use `elem_prod` for elementwise vector multiplication ... (this is mostly a problem for R/S-PLUS users, since C/C++ has no such expectation and MATLAB uses .* for elementwise operations ...

6. No errors thrown on vector size mismatch????? (there is something in `http://courses.washington.edu/fish507/ADMB%20living%20doc.doc` that describes how to turn on bounds-checking, but it looks like this only works in an IDE?? ...)

7. is there an equivalent of parallel min/max (`pmin`/`pmax` in R)? It would be useful for bounds checking, although the built-in `posfun` does work.

8. various experimentation with bounds, etc. — bounding parameters at zero where sensible seems to help even when it doesn't appear that there is such a problem at the end point

9. `tpl2rem` (needed for random effects models) appears to be fussy about dos vs unix mode (not sure about this any more, don't know if I can replicate it)

10. One can (apparently) ignore errors of the form `cat: xxalloc4.tmp: No such file or directory cat: xxalloc5.tmp: No such file or directory Error executing command cat xxglobal.tmp xxhtop.tmp header.tmp xxalloc1.tmp xxalloc2.tmp xxalloc3.tmp xxalloc4.tmp xxalloc5.tmp xxtopm.tmp xxalloc6.tmp > mccoypred3.cpp`. I think (??) these are produced by ADMB-RE models that are "too simple" (??)

11. some warnings when compiling code ... many fewer as of a recent update — would be nice if legal code compiled cleanly, without warnings. My ADMB models now compile cleanly, but my ADMB-RE code gives

```
In file included from ./df1b2fun.h:126,
                 from mccoypred5.cpp:2:
./df32fun.h:98: warning: extra qualification df3_two_vector:: on member allocate
./df32fun.h:150: warning: extra qualification df3_two_matrix:: on member df3_two_matrix
In file included from mccoypred5.cpp:2:
./df1b2fun.h:715: warning: extra qualification df1b2_gradlist:: on member write_pass1
```

```
In file included from ./df1b2fun.h:1555,
                 from mccoypred5.cpp:2:
./df3fun.h:87: warning: extra qualification df3_one_vector:: on member allocate
./df3fun.h:139: warning: extra qualification df3_one_matrix:: on member df3_one_matrix
```

12. I had trouble deciding whether random effect values needed to be included as a starting value or not: since they are specified as `random_effects_vector` rather than `init_vector` it wasn't clear to me. Dave Fournier points out that they are not *required*; my best guess is that the default behavior is to make all starting values equal to zero if a PIN file is not specified (and if the values are not specified as bounded in the `PARAMETER_SECTION` and if no explicit assignments are made in the `INITIALIZATION_SECTION`), but it's a little hard to tell. He says "the easiest way to make a pin file is to run the model with the command line option -maxfn 0 [sets maximum number of function evaluations to zero] and copy the output par file (*.p01) in this case to *.pin and edit it." (P. 1-28 of the ADMB manual says "If the model finds the parameter file (whose default name is `admodel.par`) ..." Should that be `admodel.pin`??)

13. Likelihood profiles: are likelihood profiles (`-lprof`/`likeprof_number`) supposed to work for ADMB-RE models? Is `likeprof_number` supposed to be used in addition to the standard `init_` definition, or in its place? Where is the likelihood profile information stored?

    If I add a `likeprof_number` declaration to my non-RE code, I get:

    ```
    In file included from mccoypred2.cpp:6:
    ./mccoypred2.htp:52: error: declaration of param_likeprof_number model_parameters::g
    ./mccoypred2.htp:49: error: conflicts with previous declaration
                   param_init_number model_parameters::g
    ```

    With RE code I get:

    ```
    In file included from mccoypred5.cpp:24:
    ./mccoypred5.htp:72: error: declaration of param_likeprof_number model_parameters::g
    ./mccoypred5.htp:59: error: conflicts with previous declaration
             param_init_bounded_number model_parameters::g
    ./mccoypred5.htp:126: error: declaration of df1b2variable df1b2_parameters::g
    ./mccoypred5.htp:113: error: conflicts with previous declaration
         df1b2_init_bounded_number df1b2_parameters::g
    mccoypred5.cpp: In constructor model_parameters::model_parameters(int, int, char**):
    mccoypred5.cpp:68: error: no matching function for call to
         param_init_bounded_number::allocate(const char [2])
    /usr/local/src/admb/include/admodel.h:1012: note: candidates are:
        void param_init_bounded_number::allocate(double, double, int, const char*)
    /usr/local/src/admb/include/admodel.h:1013: note:
         void param_init_bounded_number::allocate(double, double, const char*)
    ```

```
mccoypred5.cpp: In member function virtual void df1b2_parameters::allocate():
mccoypred5.cpp:238: error: no matching function for call to
    df1b2_init_bounded_number::allocate(const char [2])
./df1b2fun.h:1315: note: candidates are:
    void df1b2_init_bounded_number::allocate(double, double, int, const char*)
./df1b2fun.h:1318: note:
    void df1b2_init_bounded_number::allocate(double, double, const char*)
```

If I don't add `likeprof_number` declarations, running the code with the `-lprof` switch doesn't do anything bad, but doesn't seem to produce profiles either (at least, I can't find it)

14. check likelihood on scaled $N(0, 1)$ variables? (p. 18 of ADMB-RE manual); try this both ways ...
15. (non-significant) typos in manuals — where should I report these?
16. note: negative binomial log-likelihood `log_negbinomial_density` is parameterized with (variance/mean), which is $(1 + \mu/k)$ or $(1 + \theta\mu)$ in the more usual (overdispersion parameter) parameterizations

Thoughts/wishes:

1. it will be nice to have a full SVN setup, comparable to more typical *nix/open-source situations, where checking out the source directories and running "./configure; ./make" in the root directory Just Works. At the moment, I can get all of the source files, but it's fairly difficult to figure out the appropriate flags etc. etc. for building and linking the appropriate library files.
2. any chance of a "1-click" (or small-number-of-clicks) installer for Windows? Does the IDE version put everything in the right places so that ADMB "just works"?
3. Can I get admb-mode from the google code pages without getting the full 66MB zip file and digging through it?
4. how about .deb/.rpm packages for Linux ...? Debian repositories so I can just say `apt-get install admb`?
5. it would be nice to have `df1b2vector pow(const df1b2vector& v,const df1b2variable & x)` defined in ADMB-RE, so I could leave it out of my TPL file ...
6. it seems a little odd that `sdreport` must be explicitly turned on to get MCMC results for a parameter, when otherwise all parameters are reported on — is this just to save space when the parameter vector is long?

7. R integration: there are various attempts, but I think (IM perhaps NSHO) that they could be improved and made more "R-ish". I get the sense that there is not a huge amount of overlap (alas) between people who know ADMB and R idioms well ... The primary integration attempts are

   - some code in `glmmADMB` (which I have made use of in my stuff, especially `dat_write` and `pin_write`). See below for nit-picks ...
   - `ADMB2R`, which
   - `PBSadmb` (`http://code.google.com/p/pbs-software/`); this might be great, but it's GUI and not cross-platform, both of which put me off a little bit ... I am used to developing in R and would like to work that way. Glancing over the PBSadmb user's manual, I'm a little bit surprised that the authors haven't used (at least) S3 classes to define a class of `admb` (or `admb_fit`) objects and associated methods (plot, print, summary etc.); maybe this is because of the GUI approach?

     In any case, I wouldn't want to reinvent the wheel too thoroughly ...

8. would be nice to have negative log-likelihood helper functions defined for some other standard distributions (Poisson, Cauchy, neg. binomial are defined, but not binomial, beta-binomial, Beta ...)

9. It might be useful to write some vignettes showing how one could use `model.matrix()` and perhaps some digging in the guts of `lmer` models to easily construct model matrices for the random and fixed effects? Again, wouldn't want to re-invent too much of glmmADMB ...

10. `glmmADMB` is nice, but frustrated me a bit:

    (a) documentation holes and general missing pieces (e.g., some of the documentation still has stubs in it (see `help("glmmADMB-package")`); offsets are implemented but not documented (at least in version 0.3); I get a warning about `cannot remove file 'nbmm.std'` when I run `example(glmm.admb)`)

    (b) I find the banner "Welcome to glmmADMB" irritating — I prefer packages that don't announce themselves;

    (c) the license is undefined ...

    (d) general binomial (with $N > 1$) variables aren't possible (although the package description only says "Poisson or negative binomial response distributions", so even Bernoulli is better than documented)

    (e) it prints all the intermediate sampling results (I would rather have it be silent, or at worst produce some kind of estimated-progress bar ... although that could be a little tricky for a standard progress bar since we don't really know how long it's going to take — but we could just print "progress dots" every few steps)

(f) the one time I tried to use importance sampling it failed (granted I didn't try too hard). (I will mention that Dave Fournier immediately sent back ADMB code that would do what I wanted — but, since I wasn't ready to plunge into "full" ADMB at the time, I didn't try it out.)

There are plenty of other nice extensions (e.g. allow MCMC, retrieval of MCMC results) that would be straightforward.

What are all the files that admb spits out?

- `admodel.dep`:
- `admodel.hes` (binary):
- `hesscheck`
- `hessian.bin`
- `*.eva`
- `*.log`
- `*.luu`
- `*.rhes`
- `*.bar`
- `*.par`: parameter input
- `*.p01`: parameter output
- `*.b01`: (binary) parameter output