Using AD Model Builder and R together: getting started with the R2admb package

Ben Bolker

August 3, 2010

1 Installation

The R2admb package can be installed in R in the standard way, using install.packages (at the moment, since it's on the development platform r-forge, you'll have to say

```
> install.packages("R2admb",repos="http://r-forge.r-project.org")
```

You can also download the file and use `R CMD INSTALL` from the command line, or (depending on your operating system/R interface) use a package menu.

However, you'll also need AD Model Builder installed: see e.g.

- http://admb-project.org/
- http://admb-project.org/downloads
- http://code.google.com/p/admb-project/
- http://code.google.com/p/admb-project/downloads/list

You may also need to install a C++ compiler (in particular, the MacOS installation instructions will probably ask you to install gcc/g++ from the Xcode package). You will need to have the scripts `admb`, `adcomp`, and `adlink` in the `bin` directory of your ADMB installation (let's hope this Just Works).

2 Basics

Here's a very simple example that can easily be done with built-in tools in R.

```
> library(R2admb)
> library(ggplot2) ## for pictures
```

The data are from [1], on the numbers of reed frog tadpoles killed by predators as a function of size (`TBL` is total body length, `Kill` is the number killed out of 10 tadpoles exposed to predation). Figure 1 shows the data.

So if $p(\text{kill}) = c((S/d)\exp(1-(S/d)))^g$ (a function for which the peak occurs at $S = d$, peak height=c) then a reasonable starting set of estimates would be $c = 0.45$, $d = 13$.

```
> ReedfrogSizepred <-
   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))
```

Here is the code to fit a binomial model with `mle2` using these starting points:

```
> library(bbmle)
> m0 <- mle2(Kill~dbinom(c*((TBL/d)*exp(1-TBL/d))^g,size=10),
            start=list(c=0.45,d=13,g=1),data=ReedfrogSizepred,
            method="L-BFGS-B",
            lower=c(c=0.003,d=10,g=0),
            upper=c(c=0.8,d=20,g=20),
            control=list(parscale=c(c=0.5,d=10,g=1)))
```

Generate predicted values:

```
> TBLvec = seq(9.5,36,length=100)
> predfr <-
   data.frame(TBL=TBLvec,
              Kill=predict(m0,newdata=data.frame(TBL=TBLvec)))
```

Here is a `minimal` TPL (AD Model Builder definition) file:

```
PARAMETER_SECTION

  vector prob(1,nobs)    // per capita mort prob

PROCEDURE_SECTION

  dvariable fpen=0.0;          // penalty variable
  // power-Ricker
  prob = c*pow(elem_prod(TBL/d,exp(1-TBL/d)),g);
```

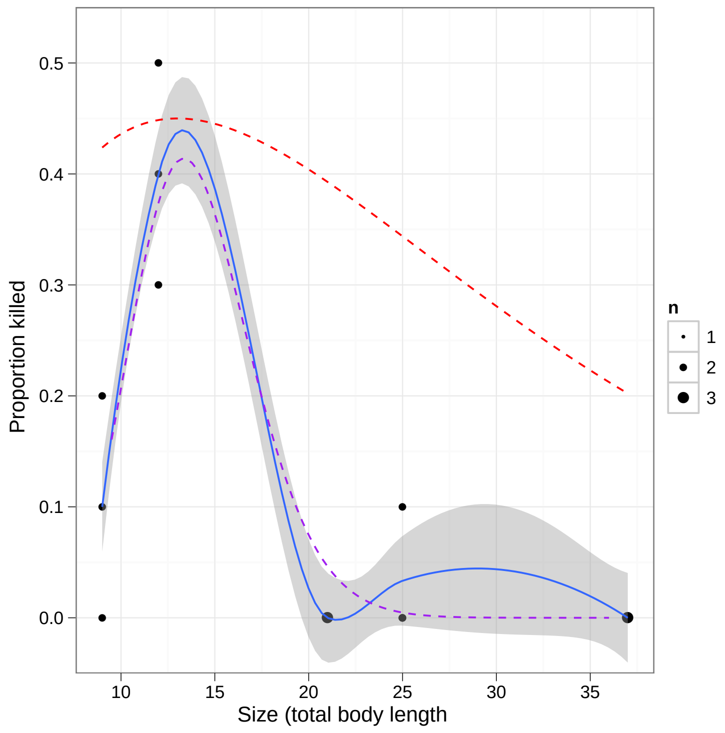


Figure 1: Proportions of reed frogs killed by predators, as a function of total body length in mm. Red: starting estimate.

```
  // penalties: constrain 0.001 <= prob <= 0.999
  prob = posfun(prob,0.001,fpen);
  f += 1000*fpen;
  prob = 1-posfun(1-prob,0.001,fpen);
  f += 1000*fpen;
  // binomial negative log-likelihood
  f -= sum( log_comb(nexposed,Kill)+
            elem_prod(Kill,log(prob))+
            elem_prod(nexposed-Kill,log(1-prob)));
```

```
> setup_admb()
> m1 <- do_admb("ReedfrogSizepred0",
                data=c(list(nobs=nrow(ReedfrogSizepred),
                  nexposed=rep(10,nrow(ReedfrogSizepred))),
                  ReedfrogSizepred),
                params=list(c=0.45,d=13,g=1),
                bounds=list(c=c(0,1),d=c(0,50),g=c(-1,25)),
                checkparam="write",
                checkdata="write",
                clean=TRUE)
> ## clean up leftovers:
> unlink(c("reedfrogsizepred0.tpl",
           "reedfrogsizepred0_gen.tpl",
           "reedfrogsizepred0"))
```

The `data`, `params`, and `bounds` (parameter bounds) arguments should be reasonably self-explanatory. When `checkparam="write"` and `checkdata="write"` are specified, `R2admb` attempts to write appropriate DATA and PARAMETER sections into a modified TPL file, leaving the results with the suffix `_gen.tpl` at the end of the run. Here's what the augmented file looks like:

```
DATA_SECTION

  init_int nobs
  init_vector nexposed(1,15)
  init_vector TBL(1,15)
  init_vector Kill(1,15)

PARAMETER_SECTION

  objective_function_value f
  init_number c
  init_number d
  init_number g
  vector prob(1,nobs)    // per capita mort prob
PROCEDURE_SECTION

  dvariable fpen=0.0;         // penalty variable
  // power-Ricker
  prob = c*pow(elem_prod(TBL/d,exp(1-TBL/d)),g);
  // penalties: constrain 0.001 <= prob <= 0.999
  prob = posfun(prob,0.001,fpen);
  f += 1000*fpen;
  prob = 1-posfun(1-prob,0.001,fpen);
  f += 1000*fpen;
  // binomial negative log-likelihood
  f -= sum( log_comb(nexposed,Kill)+
            elem_prod(Kill,log(prob))+
            elem_prod(nexposed-Kill,log(1-prob)));
```

You might instead choose to compose the whole TPL file yourself, in which case you can add comments appropriately:

```
DATA_SECTION
  init_int nobs                // # of observations
  init_vector nexposed(1,nobs) // # exposed per trial
  init_vector TBL(1,nobs)      // total body length
  init_vector Kill(1,nobs)     // # killed per trial
PARAMETER_SECTION
  init_bounded_number c(0,1) // baseline mort prob
  init_bounded_number d(0,50) // size scaling factor
  init_bounded_number g(-1,25) // size scaling power
  vector prob(1,nobs)    // per capita mort prob
  objective_function_value f
  sdreport_number rc;    // this & following for MCMC
  sdreport_number rd;
  sdreport_number rg;
PROCEDURE_SECTION
  rc = c; rd = d; rg = g;     // set MCMC reporting
  dvariable fpen=0.0;         // penalty variable
  // power-Ricker
  prob = c*pow(elem_prod(TBL/d,exp(1-TBL/d)),g);
  // penalties: constrain 0.001 <= prob <= 0.999
  prob = posfun(prob,0.001,fpen);
  f += 1000*fpen;
  prob = 1-posfun(1-prob,0.001,fpen);
```

```
f += 1000*fpen;
// binomial negative log-likelihood
f -= sum( log_comb(nexposed,Kill)+
          elem_prod(Kill,log(prob))+
          elem_prod(nexposed-Kill,log(1-prob)));
```

Here are some of the basic extractor methods provided:

- Basic information about the fit and coefficient estimates:

```
> m1

Model file: reedfrogsizepred0
Negative log-likelihood: 12.8938
Coefficients:
         c          d          g
 0.4138327 13.3508231 18.2478388
```

- Coefficients only: ■coef■= coef(m1)
- Coefficient table including standard errors and (approximate!!) p values:

```
> summary(m1)

Model file: reedfrogsizepred0
Negative log-likelihood: 12.8938
Coefficients:
  Estimate Std. Error z value Pr(>|z|)
c   0.4138     0.1257   3.292 0.000996 ***
d  13.3508     0.8111  16.461  < 2e-16 ***
g  18.2478     6.0331   3.025 0.002489 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Extract the table of coefficients from the summary:

```
> coef(summary(m1))

    Estimate Std. Error   z value     Pr(>|z|)
c  0.4138327    0.12572  3.291701 9.958327e-04
d 13.3508231    0.81107 16.460753 7.022115e-61
g 18.2478388    6.03310  3.024621 2.489452e-03
```

- Variance-covariance matrix of the parameters:

  ```
  > vcov(m1)
  ```

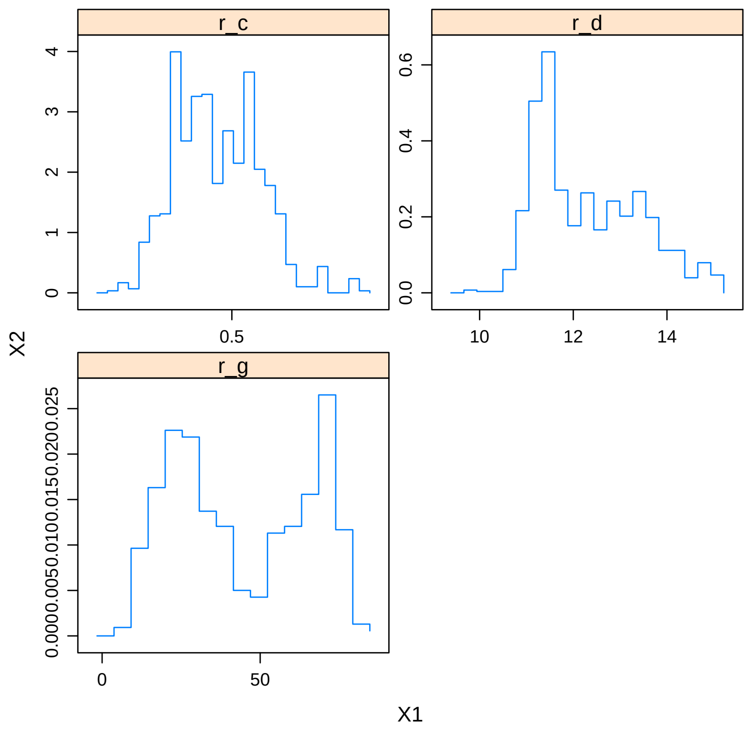
  ```
              c         d          g
  c 0.01580552 0.0578055  0.5043901
  d 0.05780550 0.6578345  2.2464986
  g 0.50439009 2.2464986 36.3982956
  ```

  Log-likelihood, deviance, AIC:

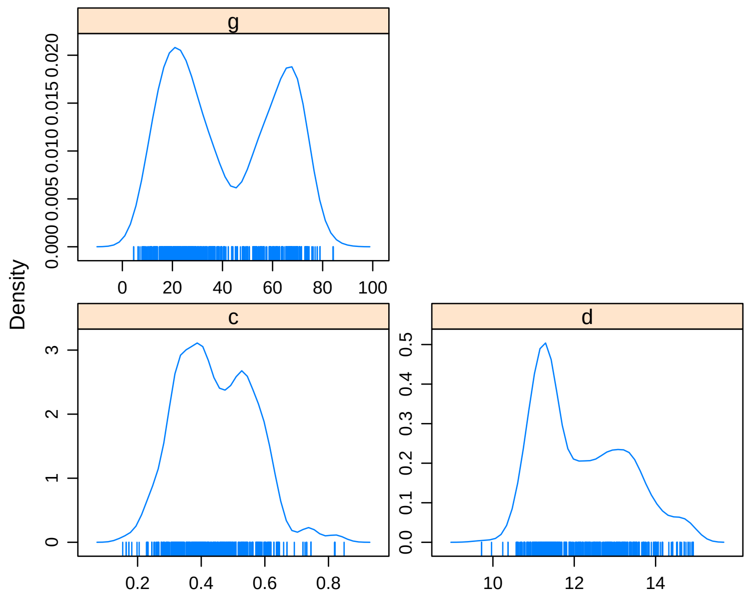
  ```
  > c(logLik(m1),deviance(m1),AIC(m1))
  ```

  ```
  [1] -12.8938  25.7876  31.7876
  ```

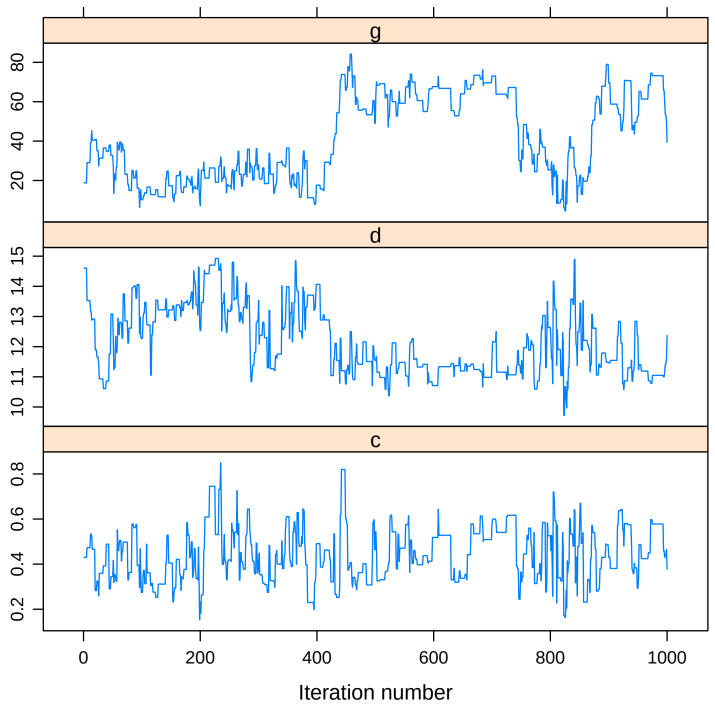
```
> m1MC <- do_admb("ReedfrogSizepred",
              data=c(list(nobs=nrow(ReedfrogSizepred),
                nexposed=rep(10,nrow(ReedfrogSizepred))),
                ReedfrogSizepred),
                params=list(c=0.45,d=13,g=1),
                bounds=list(c=c(0,1),d=c(0,50),g=c(-1,25)),
                checkparam="write",
                checkdata="write",
                mcmc=TRUE,
                mcmcpars=c("c","d","g"),
                clean=TRUE)
> ## clean up leftovers:
> unlink(c("reedfrogsizepred0.tpl",
         "reedfrogsizepred0_gen.tpl",
         "reedfrogsizepred0"))

> print(plot(m1MC$hist))
```



```
> library(coda)
> HPDinterval(as.mcmc(m1MC$mcmc))

       lower      upper
c  0.2278136  0.6366696
d 10.5913713 14.4189225
g 11.2400600 73.8567544
attr(,"Probability")
[1] 0.95

> print(densityplot(as.mcmc(m1MC$mcmc)))
```



```
> print(xyplot(as.mcmc(m1MC$mcmc)))
```



3 Random effects: gopher tortoise data

References

[1] James R. Vonesh and Benjamin M. Bolker. Compensatory larval responses shift tradeoffs associated with predator-induced hatching plasticity. *Ecology*, 86(6):1580–1591, 2005.