# Fir fecundity example using TMB and mle2

Mollie Brooks

November 3, 2016

## Introduction

This vignette may be useful to R users who are familiar with using `mle2` in the bbmle package but are needing a more flexible and robust maximum likelihood estimator. Reasons for making the switch may include trouble with model convergence or needing to add random effects to account for correlation.

The model presented here is found in chapter 6 of *Ecological Models and Data in R* by Bolker 2008 (hereafter EMD book) and the data originally appeared in Silvertown and Dodd 1999, and Dodd and Silvertown 2000. The data on *Abies balsamea* includes counts of the cones produced, measurements of tree size (diameter at breast height, DBH), and the status (wave / non-wave) indicating wave-like die-offs experienced by some populaitons. The mean fir fecundity is predicted to follow a power-law and the number of cones should be negative binomially distributed around the mean.

$\mu = a$ DBH $^b$
cones $\sim$NegativeBinomial($\mu$, $k$)

## 1 Packages

```
library(emdbook) #for data
library(bbmle) #for mle2

## Loading required package:  stats4

library(TMB)
```

## 2 Data

Data organization is the same as in the EMD book.

```
data(FirDBHFec)
dat=na.omit(FirDBHFec[,c("TOTCONES", "DBH", "WAVE_NON")])
dat$TOTCONES=round(dat$TOTCONES)
```

## 3 Fitting the model with `mle2`

The EMD book fits the following model:

```
nbfit.ab = mle2(TOTCONES ~ dnbinom(mu = a * DBH^b,size = k),
                start = list(a=c(1,0),b=c(1,0),k=1), data = dat,
                parameters = list(a ~ WAVE_NON, b ~ WAVE_NON),
                optimizer="nlminb")
summary(nbfit.ab)

## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = TOTCONES ~ dnbinom(mu = a * DBH^b, size = k),
##     start = list(a = c(1, 0), b = c(1, 0), k = 1), optimizer = "nlminb",
##     data = dat, parameters = list(a ~ WAVE_NON, b ~ WAVE_NON))
##
## Coefficients:
##               Estimate Std. Error z value  Pr(z)
## a.(Intercept)  0.28765    0.18056  1.5931 0.1111
## a.WAVE_NONw    0.12043    0.30456  0.3954 0.6925
## b.(Intercept)  2.35524    0.28103  8.3808 <2e-16 ***
## b.WAVE_NONw   -0.20627    0.41244 -0.5001 0.6170
## k              1.50706    0.14310 10.5313 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 2271.353
```

# 4 Fitting the model in TMB

The TMB code for model (in a file named `fir.cpp`) looks like this:

```
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
        DATA_VECTOR(totcones);
        DATA_VECTOR(dbh);
        DATA_MATRIX(wave_non);
        PARAMETER_VECTOR(a);
        PARAMETER_VECTOR(b);
        PARAMETER(log_k); //to keep k positive
        Type k= exp(log_k);
        Type nll;
        vector<Type> a_vec=wave_non*a;
        vector<Type> b_vec=wave_non*b;
        vector<Type> mu=a_vec* pow(dbh, b_vec);
        vector<Type> var=mu+mu*mu/k;
        nll=-sum(dnbinom2(totcones, mu, var, true));
        return(nll);
}
```

## 4.1 Compile TMB code

```
compile("fir.cpp")

## Note: Using Makevars in /Users/molliebrooks1/.R/Makevars
## [1] 0

dyn.load("fir.so")
```

## 4.2 Data structures for TMB

Then I organize the data into the formats for TMB. It takes a list of the observed data `Ldat` and a list of the initial values of the parameters `Lpin`. The object names must be the same as in `fir.cpp`.

```
Ldat=list(totcones= dat$TOTCONES,
                        dbh=dat$DBH,
                        wave_non=model.matrix(~WAVE_NON, data=dat)
                        )
Lpin=list(a=c(1,0), b=c(1,0), log_k=1)
```

### 4.3 Fitting the model via TMB

First we must compile the code. Then run it.

```
mod= MakeADFun(Ldat, Lpin, DLL="fir", silent=TRUE)
fit=nlminb(mod$par, mod$fn, mod$gr)
sdr=sdreport(mod)
summary(sdr)
```

```
##         Estimate Std. Error
## a      0.2876415  0.1803691
## a      0.1204278  0.3043814
## b      2.3552462  0.2807403
## b     -0.2062675  0.4121832
## log_k  0.4101572  0.0949553
```

## 5 Comparing results

```
TMBAIC=function(opt){2*length(opt[["par"]])+2*opt[["objective"]]}
all.equal(TMBAIC(fit), AIC(nbfit.ab))
```

```
## [1] TRUE
```

```
all.equal(unname(fit$par)[1:4], unname(coef(nbfit.ab))[1:4])
```

```
## [1] "Mean relative difference: 3.773851e-06"
```

## 6 Conclusion

We got the same parameter estimates and AIC as the example in the EMD book. Most importantly, TMB could be expanded further and include random effects to account for spatial or temporal

correlations.