Multievent/HMM capture-recapture with TMB

Olivier Gimenez, with precious help from Mollie Brooks

August 17, 2017

Following my attempts to fit a HMM model to capture-recapture data with Rcpp and to occupancy data with ADMB, a few colleagues suggested TMB as a potential alternative for several reasons (fast, allows for parallel computations, works with R, accomodates spatial stuff, easy implementation of random effects, and probably other reasons that I don't know).

I found materials on the internet to teach myself TMB, at least what I needed to implement a simple HMM model. See here for a linear regression and a Gompertz state space model examples, here for the same linear regression example on Youtube (that's awesome!) and many other examples here. However, I got stuck and posted my desperate request for help on the TMB forum. Guess what, I got an answer less than a few hours after - thank you Mollie Brooks!

First, let's read in the data.

```r
set.seed(1)

# read in data
data = read.table('titis2.txt')
#data = rbind(data,data,data,data,data) # increase sample size artificially

# define various quantities
nh <- dim(data)[1]
k <- dim(data)[2]
km1 <- k-1

# counts
eff <- rep(1,nh)

# compute the date of first capture fc, and state at initial capture init.state
fc <- NULL
init.state <- NULL
for (i in 1:nh){
  temp <- 1:k
  fc <- c(fc,min(which(data[i,]!=0)))
  init.state <- c(init.state,data[i,fc[i]])
}

# init values
binit <- runif(9)

# transpose data
data <- t(data)
```

Now the TMB implementation:

```r
library(TMB)
compile("multievent_tmb.cpp")
```

```
## [1] 0
```

```
dyn.load(dynlib("multievent_tmb"))
```

```
f <- MakeADFun(
  data = list(ch = data, fc = fc, fs = init.state),
  parameters = list(b = binit),
  DLL = "multievent_tmb")
opt <- do.call("optim", f) # optimisation
```

```
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
```

```
f$fn(binit) # evaluate likelihood at the inits
```

```
## [1] 4119.899
```

```
f$report()$B # display B
```

```
##           [,1]      [,2] [,3]
## [1,] 0.2893815 0.2799569    1
## [2,] 0.4686099 0.0000000    0
## [3,] 0.0000000 0.4696758    0
## [4,] 0.2420085 0.2503672    0
```

```
f$report()$BE # display BE
```

```
##           [,1]      [,2] [,3]
## [1,] 0.0000000 0.0000000    1
## [2,] 0.6594396 0.0000000    0
## [3,] 0.0000000 0.6522885    0
## [4,] 0.3405604 0.3477115    0
```

```
f$report()$A # display A
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.1701131 0.4218590 0.4080279
## [2,] 0.3518417 0.2875796 0.3605787
## [3,] 0.0000000 0.0000000 1.0000000
```

```
f$report()$PROP # display PROP
```

```
## [1] 0.56599 0.43401 0.00000
```

```
rep <- sdreport(f)
```

```
## outer mgc:  0.000634904
## outer mgc:  0.1453409
## outer mgc:  0.1452904
## outer mgc:  0.1061278
## outer mgc:  0.1071786
## outer mgc:  0.06220212
## outer mgc:  0.06251711
## outer mgc:  0.06954232
## outer mgc:  0.06969586
## outer mgc:  0.03845126
## outer mgc:  0.03828197
## outer mgc:  0.1258497
## outer mgc:  0.1264165
## outer mgc:  0.09211591
## outer mgc:  0.09240274
## outer mgc:  0.1870961
## outer mgc:  0.1857218
## outer mgc:  0.05774126
## outer mgc:  0.05848487
```

```
rep # get SEs
```

```
## sdreport(.) result
##     Estimate Std. Error
## b  0.8674522 0.13515262
## b  1.4764530 0.10843431
## b  1.6396688 0.14059086
## b -1.2691569 0.14751666
## b -1.2283551 0.27945198
## b  0.2600258 0.11546465
## b  0.3959696 0.13472143
## b -1.4643765 0.09068099
## b  1.0333844 0.28555423
## Maximum gradient component: 0.000634904
```

Now, let's implement the same model with standard R code:

```
devMULTIEVENT <- function(b,data,eff,e,garb,nh,km1){

# data encounter histories, eff counts
# e vector of dates of first captures
# garb vector of initial states
# km1 nb of recapture occasions (nb of capture occ - 1)
# nh nb ind

# OBSERVATIONS (+1)
# 0 = non-detected
# 1 = seen and ascertained as non-breeder
# 2 = seen and ascertained as breeder
# 3 = not ascertained

# STATES
# 1 = alive non-breeder
# 2 = alive breeder
# 3 = dead

# PARAMETERS
# phiNB  survival prob. of non-breeders
# phiB  survival prob. of breeders
# pNB  detection prob. of non-breeders
# pB  detection prob. of breeders
# psiNBB transition prob. from non-breeder to breeder
# psiBNB transition prob. from breeder to non-breeder
# piNB prob. of being in initial state non-breeder
# deltaNB prob to ascertain the breeding status of an individual encountered as non-breeder
# deltaB prob to ascertain the breeding status of an individual encountered as breeder

# logit link for all parameters
# note: below, we decompose the state and obs process in two steps composed of binomial events,
# which makes the use of the logit link appealing;
# if not, a multinomial (aka generalised) logit link should be used
    par = plogis(b)
    piNB <- par[1]
    phiNB <- par[2]
    phiB <- par[3]
    psiNBB <- par[4]
    psiBNB <- par[5]
    pNB <- par[6]
    pB <- par[7]
    deltaNB <- par[8]
    deltaB <- par[9]

# prob of obs (rows) cond on states (col)
    B1 = matrix(c(1-pNB,pNB,0,1-pB,0,pB,1,0,0),nrow=3,ncol=3,byrow=T)
    B2 = matrix(c(1,0,0,0,0,deltaNB,0,1-deltaNB,0,0,deltaB,1-deltaB),nrow=3,ncol=4,byrow=T)
    B = t(B1 %*% B2)

# first encounter
    BE1 = matrix(c(0,1,0,0,0,1,1,0,0),nrow=3,ncol=3,byrow=T)
    BE2 = matrix(c(1,0,0,0,0,deltaNB,0,1-deltaNB,0,0,deltaB,1-deltaB),nrow=3,ncol=4,byrow=T)
    BE = t(BE1 %*% BE2)

# prob of states at t+1 given states at t
```

```
    A1 <- matrix(c(phiNB,0,1-phiNB,0,phiB,1-phiB,0,0,1),nrow=3,ncol=3,byrow=T)
    A2 <- matrix(c(1-psiNBB,psiNBB,0,psiBNB,1-psiBNB,0,0,0,1),nrow=3,ncol=3,byrow=T)
    A <- A1 %*% A2

# init states
    PI <- c(piNB,1-piNB,0)

# likelihood
    l <- 0
    for (i in 1:nh) # loop on ind
   {
      ei <- e[i] # date of first det
      oe <- garb[i] + 1 # init obs
      evennt <- data[,i] + 1 # add 1 to obs to avoid 0s in indexing
      ALPHA <- PI*BE[oe,]
     for (j in (ei+1):(km1+1)) # cond on first capture
     {
        if ((ei+1)>(km1+1)) {break}
        ALPHA <- (ALPHA %*% A)*B[evennt[j],]
      }
      l <- l + log(sum(ALPHA))#*eff[i]
    }
    l <- -l
    l
  }
```

Let’s do some benchmarking:

```
# The optimization is not stochastic, but depending on what else I'm doing,
# computation times may vary, hence a benchmark
library(microbenchmark)
res = microbenchmark(
  optim(binit,devMULTIEVENT,NULL,hessian=F,data,eff,fc,init.state,nh,km1,method="BFGS"),
  do.call("optim", f),
  times=5
 )
```

```
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
## outer mgc:  325.077
## outer mgc:  221.2212
## outer mgc:  160.3872
## outer mgc:  145.6448
## outer mgc:  125.8119
## outer mgc:  89.23869
## outer mgc:  94.46672
## outer mgc:  67.25231
## outer mgc:  68.50366
## outer mgc:  52.73606
## outer mgc:  54.96598
## outer mgc:  25.97314
## outer mgc:  20.21621
## outer mgc:  21.97169
## outer mgc:  21.67411
## outer mgc:  23.46248
## outer mgc:  19.97417
## outer mgc:  12.68817
## outer mgc:  6.547335
## outer mgc:  4.064836
## outer mgc:  2.625934
## outer mgc:  1.73165
## outer mgc:  1.579185
## outer mgc:  2.633778
## outer mgc:  1.580973
## outer mgc:  1.600656
## outer mgc:  1.605499
## outer mgc:  1.223522
## outer mgc:  0.9529237
## outer mgc:  0.9356155
## outer mgc:  0.07340839
## outer mgc:  0.007294368
```

```
res2 = summary(res)
```

Now the TMB code is 327.5042507 times faster than basic R!!