Fitting models via maximum likelihood using RTMB

Chris Cahill and Jim Bence

21 July 2023

Quantitative Fisheries Center
MICHIGAN STATE UNIVERSITY

Outline

- Demonstrate a mind-blowing advance with AD
- Show a few examples that may be useful
  - Start with equations, move to code
- Talk about debugging via `browser()`
- Tips and trickery
- Code for all of this available at:
  https://github.com/QFCatMSU/RTMB/tree/main

What is RTMB?

- RTMB is a new package that provides a native R interface for a subset of TMB so you can avoid coding in C++
- See https://kaskr.r-universe.dev/RTMB
- In all applications we have tried all TMB functionality was available!
- Because code is all in R, both easier to code and for others to read that code - a game changer!
- A game changer *if* you know how to code in R, create an objective f(x) for your model
- No compiling or compiling errors!
- Bottom line: less time developing and testing models, more intuitive code

Linear regression in RTMB

- The math:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \mathrm{N}(0, \sigma)$$

```
1  par(mar=c(5,6,4,1) + 0.1)
2  plot(y_obs~x1, xlab = "x1", ylab = "observed data", cex.lab = 2.5)
```
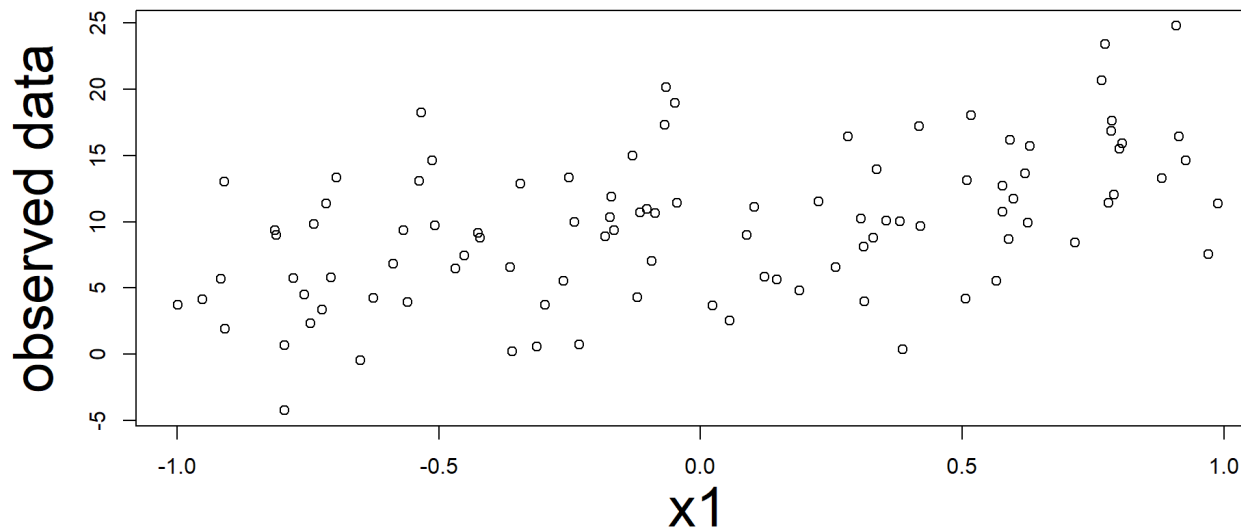
Linear regression (RTMB)

```r
1  # set up tagged data + parameters lists:
2  data = list(
3    n = n,
4    y_obs = y_obs,
5    x1 = x1
6  )
7
8  pars = list(
9    b0 = 1,
10   b1 = 1,
11   log_sd = log(3)
12 )
```

- see `linreg.r`

Linear regression (RTMB)

```r
 1  library(RTMB)
 2
 3  # write an objective function returning negative log-likelihood
 4  f = function(pars) {
 5    getAll(data, pars) # replaces DATA_XX, PARAMETER_YY
 6    y_pred = b0 + b1 * x1
 7    nll = -sum(dnorm(y_obs, y_pred, exp(log_sd), log = TRUE))
 8    nll
 9  }
10
11  obj = MakeADFun(f, pars)
12  obj$fn() # objective function
```

```
[1] 777.5154
```

```r
 1  obj$gr() # gradients
```

```
outer mgc:  1051.521
           [,1]      [,2]       [,3]
[1,] -96.88602 -13.2133 -1051.521
```

- Stick to base R

Linear regression (RTMB)

```
1  opt = nlminb(obj$par, obj$fn, obj$gr)
```

```
outer mgc:   1051.521
outer mgc:   54.72716
outer mgc:   36.6083
outer mgc:   21.88026
outer mgc:   35.7811
outer mgc:   5.686432
outer mgc:   1.353547
outer mgc:   1.508322
outer mgc:   0.08695211
outer mgc:   0.04554657
outer mgc:   0.0305969
outer mgc:   0.002009656
outer mgc:   1.470605e-05
```

Linear regression (RTMB)

```
1  sdr = sdreport(obj)
```

```
outer mgc:   1.470605e-05
outer mgc:   0.004344401
outer mgc:   0.004344313
outer mgc:   0.001393873
outer mgc:   0.00140084
outer mgc:   0.1997855
outer mgc:   0.2002149
```

- We are done.

Access fit

```
1  opt
```

```
$par
      b0        b1    log_sd
9.730622 4.775419 1.568146

$objective
[1] 298.7085

$convergence
[1] 0

$iterations
[1] 12

$evaluations
function gradient
```

Access fit

```
1 sdr
```

```
sdreport(.) result
      Estimate Std. Error
b0     9.730622 0.47978081
b1     4.775419 0.84596427
log_sd 1.568146 0.07071065
Maximum gradient component: 1.470605e-05
```

RTMB scales to (much) more complicated models

von Bertalanffy growth model: the math

$$l_i = l_\infty \left(1 - e^{-k(a_i - t_0)}\right) + \varepsilon_i$$

$$\varepsilon_i \sim N\left(0, \sigma^2\right)$$

RTMB objective f(x) for a von Bertalanffy growth model:

```r
 1  f = function(pars) {
 2    getAll(data, pars)
 3    linf = exp(log_linf)
 4    vbk = exp(log_vbk)
 5    sd = exp(log_sd)
 6    l_pred = linf * (1 - exp(-vbk * (age_i - t0)))
 7    nll = -sum(dnorm(l_obs_i, l_pred, sd, TRUE))
 8    REPORT(linf)
 9    REPORT(vbk)
10    ADREPORT(sd)
11    nll
12  }
```

- see `vonB.r`
- can use REPORT(), ADREPORT()

Poisson GLMM: the math

$$y_{i,site} \sim \text{Poisson}\left(\lambda_{i,site}\right)$$

$$\log(\lambda_{i,site}) = \beta_0 + \beta_1 \cdot x_1 + \epsilon_{site}$$

$$\epsilon_{site} \sim \text{N}(0, \sigma^2_{site})$$

- β_0 is global intercept shared among sites
- x_1 is a covariate
- ϵ_{site} is normally distributed random effect

Objective f(x) for a Poisson GLMM:

```r
 1  f = function(pars) {
 2    getAll(data, pars)
 3    sd_site = exp(log_sd_site)                        # back transform
 4    jnll = 0                                          # initialize
 5    jnll = jnll - sum(dnorm(eps_site, 0, sd_site, TRUE)) # Pr(random effects)
 6    lam_i = exp(                                      # link f(x)
 7      Xmat %*% bvec +                                 # fixed effects
 8      eps_site                                        # random effects
 9    )
10    jnll = jnll - sum(dpois(yobs, lam_i, TRUE))       # Pr(observations)
11    jnll
12  }
```

- see `glmm.r`

Objective f(x) for a hierarchical normal selectivity model, adapted from Millar and Freyer 1999:

```r
 1  f = function(pars) {
 2    getAll(data, pars)
 3    jnll = 0
 4    jnll = jnll - sum(dnorm(k1_dev, 0, exp(ln_sd_k1), TRUE))
 5    jnll = jnll - sum(dnorm(k2_dev, 0, exp(ln_sd_k2), TRUE))
 6    k1 = exp(ln_k1 + k1_dev)
 7    k2 = exp(ln_k2 + k2_dev)
 8    sel_mat = phi_mat = matrix(0, nrow(catches), ncol(catches))
 9    for (i in 1:nrow(sel_mat)) {
10      for (j in 1:ncol(sel_mat)) {
11        sel_mat[i, j] = exp(-(lens[i] - k1[site[i]] * rel_size[j])^2 /
12          (2 * k2[site[i]]^2 * rel_size[j]^2))
13      }
14    }
15    sel_sums = rowSums(sel_mat)
16    for (i in 1:nrow(phi_mat)) {
17      for (j in 1:ncol(phi_mat)) {
18        phi_mat[i, j] = sel_mat[i, j] / sel_sums[i]
19      }
20    }
21    jnll = jnll - sum(catches * log(phi_mat))
22    jnll
23  }
```

Spatially explicit Poisson GLMM: the math

$$y_{i,site} \sim \text{Poisson}\left(\lambda_{i,site}\right)$$

$$\log(\lambda_{i,site}) = \beta_0 + \epsilon_{site}$$

$$\epsilon_{site} \sim \text{MVN}(0, \Sigma_{site})$$

- where Σ_{site} is calculated via some correlation f(x)

- we'll use exponential

Objective f(x) for a spatially explicit GLMM:

```r
1  f = function(pars){
2    getAll(data, pars)
3    SIGMA = gp_sigma * exp(-dist_sites / gp_theta)
4    jnll = 0
5    jnll = jnll - sum(dmvnorm(eps_s, SIGMA, TRUE))
6    y_hat = exp(beta0 + eps_s) # index on site_i in more complex examples
7    jnll = jnll - sum(dpois(y_obs, y_hat), TRUE)
8    jnll
9  }
```

- majicks
- see `grf.r`

Jim Bence's fancy vonB: math

- ?

Objective f(x) for a Bence's fancy vonB:

```r
1  f = function(pars) {
2    getAll(data, pars)
3    Linfmn = exp(logLinfmn)
4    logLinfsd = exp(loglogLinfsd)
5    Linfs = exp(logLinfs)
6    K = exp(logK)
7    Sig = exp(logSig)
8    nponds = length(Linfs)
9    nages = length(A)
10   predL = matrix(0, nrow = nages, ncol = nponds)
11   # fill one column (pond) at a time:
12   for (i in 1:nponds) {
13     predL[, i] = Linfs[i] * (1 - exp(-K * (A - t0)))
14   }
15   nll = -sum(dnorm(x = L, mean = predL, sd = Sig, log = TRUE))
16   nprand = -sum(dnorm(x = logLinfs, mean = logLinfmn, sd = logLinfsd, log = TRUE))
17   jnll = nll + nprand
18   jnll
19 }
```

- see `multilinf.r`

Debugging

- Because RTMB is written in R, can use debugging tools (!)
- `browser()` allows you to step through and check calculations
  - no longer need `cout` or `REPORT()` calls to check calculations
- Can also use breakpoints
- Jump to demonstration

Tips

- Talk about `advector` error messages
  - Often(?) means you are using something that isn't supported by RTMB
- Make sure you are using base R / RTMB f(x)'s
- Hash out each line to find the offending line(s)
- Discuss other oddities

Questions?

- cahill11@msu.edu