Appendix C: Compare zero-inflated mixed models across R packages

Mollie Brooks and Ben Bolker

2017-05-01

In this appendix, we analyze counts of begging behavior by owl nestlings. This example previously appeared in Zuur et al. (2009) and Bolker et al. (2013) and was originally published by Roulin and Bersier (2007). The response variable is the number of calls from chicks (`NCalls`) in a nest. Since this should directly scale with the number of chicks (i.e. brood size), `logBroodSize` is used as an offset term. Since nests were repeatedly measured, `Nest` is included as a random effect. Covariates of interest include the sex of the parent visiting the nest (`SexParent`), whether the chicks were satiatited or not (`FoodTreatment`), and the timing of the parent's arrival (`ArrivalTime`).

Preliminaries

Load packages

```
library(glmmTMB)
library(glmmADMB)
library(MCMCglmm)
library(brms)
library(INLA)
library(broom) #for tidy
library(plyr)
library(dplyr) #tidyverse
library(ggplot2); theme_set(theme_bw())
library(ggstance)#for position_dodgev
```

Data organization and helper functions (hidden)

```
data(Owls)
Owls = plyr::rename(Owls, c(SiblingNegotiation="NCalls"))
Owls = transform(Owls, ArrivalTime=scale(ArrivalTime, center=TRUE, scale=FALSE))
```

Constant zero-inflation

Here we fit the model with zero-inflation assumed to be constant across the data set, i.e. zero-inflation is independent of the predictor variables.

glmmTMB

```
form = NCalls~(FoodTreatment + ArrivalTime) * SexParent +
    offset(logBroodSize) + (1|Nest)
time.tmb = tfun(m1.tmb <<- glmmTMB(form,
                                   ziformula=~1, data = Owls, family=poisson))
```

glmmADMB

```
time.admb = tfun(m1.admb <<- glmmadmb(form,
                 zeroInflation=TRUE, data = Owls, family="poisson"))
```

MCMCglmm

Code for this example was copied from Bolker et al. (2013); a more complete description appears in the supplementary material for that paper.

```
offvec = c(1,1,2,rep(1,5)) # 1=non-offset; 2=offset
fixef2 = NCalls~trait-1+ # intercept terms for both count and binary terms
    # other fixed-effect terms only apply to count term
    at.level(trait,1):logBroodSize+
   at.level(trait,1):((FoodTreatment+ArrivalTime)*SexParent)
# residual variances independent for count and binary terms;
#    fixed to 1 for binary term
# random-effects variances independent for count and binary terms;
#    fixed very small (1e-6) for binary term
prior_overdisp  = list(R=list(V=diag(c(1,1)),nu=0.002,fix=2),
                       G=list(list(V=diag(c(1,1e-6)),nu=0.002,fix=2)))
prior_overdisp_broodoff = c(prior_overdisp,
                            list(B=list(mu=c(0,1)[offvec],
                              V=diag(c(1e8,1e-6)[offvec]))))
time.mcmc=tfun(m1.mcmc <<- MCMCglmm(fixef2,
                                    rcov=~idh(trait):units,
                                    random=~idh(trait):Nest,
                                    prior=prior_overdisp_broodoff,
                                    data=Owls,
                                    family="zipoisson",
                                    verbose=FALSE))
```

brms

```
time.brms = tfun(m1.brms <<- brm(form, data = Owls,
                                 family="zero_inflated_poisson",
                                 save_dso=TRUE))
```

```
## Compiling the C++ model
```

```
time.brms2 = tfun(m1.brms2 <<- update(m1.brms))
```

INLA

```
time.inla = tfun(m1.inla <<- inla(NCalls~(FoodTreatment + ArrivalTime) * SexParent +
                                    f(Nest, model="iid"),
                                  offset = logBroodSize,
                                  family= "zeroinflatedpoisson1",
                                  data=Owls))
```

Comparing the results

Timings

```
sort(c(glmmTMB=time.tmb,glmmADMB=time.admb,MCMCglmm=time.mcmc,brms=time.brms,
  brms2=time.brms2,INLA=time.inla))
```

```
##      INLA  glmmTMB MCMCglmm glmmADMB    brms2     brms
##     1.274    1.626    8.937   23.619   40.727   75.837
```

(Time is recorded in seconds.)

`glmmTMB` fit the model in less than 5 seconds. Other methods were slower, but `MCMCglmm` was in the same order of magnitude (`brms` and `brms2` are times including and excluding compilation time, respectively).

Estimated fixed-effect coefficients

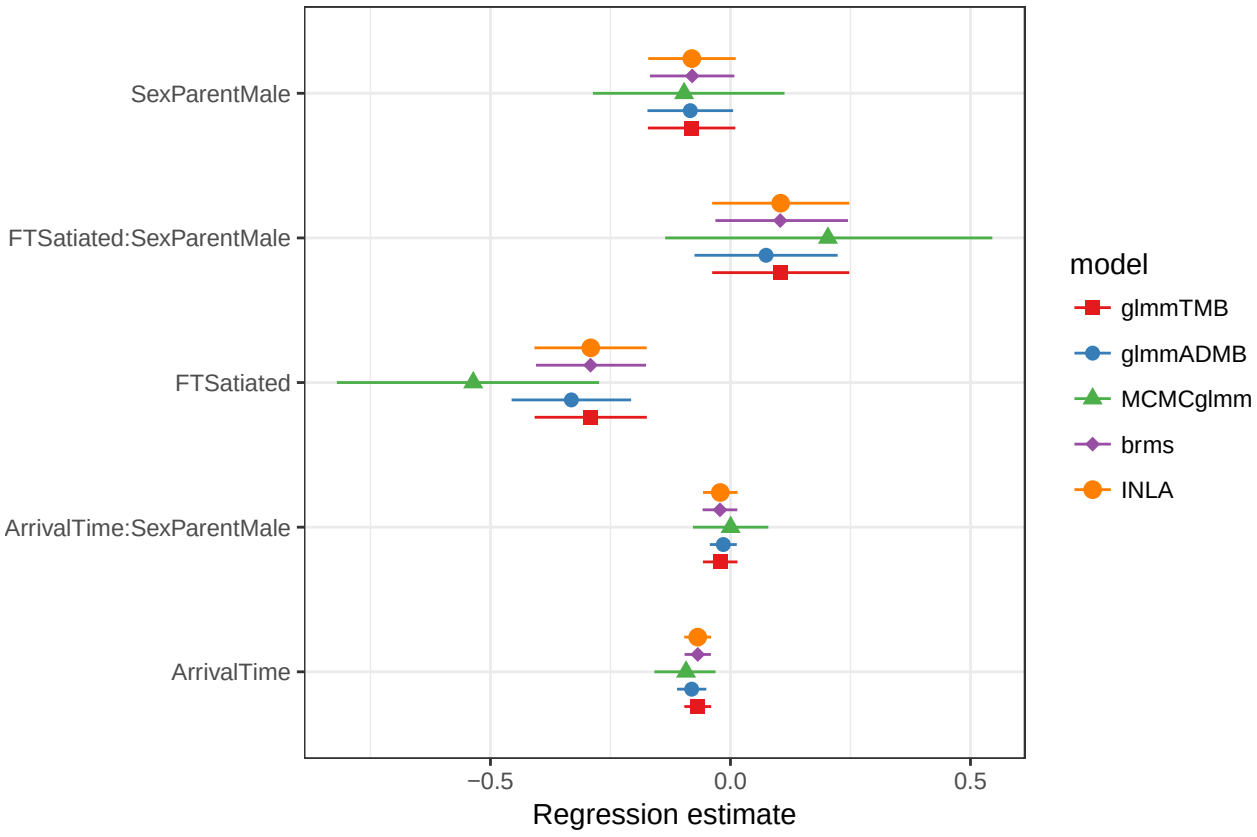

Figure C.1 – Estimated fixed-effect coefficients Estimates are from the same zero-inflated Poisson model fit using functions `glmmTMB`, `glmmadmb`, `MCMCglmm`, `brm`, and `inla`.

Because we ran `brms` with flat priors, the estimates are very close to the maximum likelihood estimates of `glmmTMB`. Maximum likelihood estimates from `glmmTMB` and `glmmADMB` differ slightly because `glmmADMB` uses some numerical tricks to increase robustness and these change the objective function by a small amount.

Complex zero-inflation

Here we fit the model with zero-inflation depending on some of the predictor variables. We can no longer use `glmmADMB` and `INLA`.

glmmTMB

```
ziform = ~FoodTreatment+(1|Nest)
time.tmb_czi = tfun(m1.tmb_czi <<- glmmTMB(form,
                            ziformula=ziform, data = Owls, family=poisson))
```

MCMCglmm

```
offvec_czi = c(1,1,2,rep(1,6)) # 1=non-offset; 2=offset
fixef3 = NCalls~trait-1+ # intercept terms for both count and binary terms
    # fixed-effect terms for count term
    at.level(trait,1):logBroodSize+
    at.level(trait,1):((FoodTreatment+ArrivalTime)*SexParent)+
    # fixed-effect terms for binary term
    at.level(trait,2):FoodTreatment
# residual variances independent for count and binary terms;
#    fixed to 1 for binary term
# random-effects variances now allow estimated variance for binary term
#   as well
prior_overdisp_czi  = list(R=list(V=diag(c(1,1)),nu=0.002,fix=2),
                       G=list(list(V=diag(c(1,1)),nu=0.002)))
prior_overdisp_broodoff_czi = c(prior_overdisp_czi,
                             list(B=list(mu=c(0,1)[offvec_czi],
                               V=diag(c(1e8,1e-6)[offvec_czi]))))
time.mcmc_czi=tfun(m1.mcmc_czi <<- MCMCglmm(fixef3,
                                    rcov=~idh(trait):units,
                                    random=~idh(trait):Nest,
                                    prior=prior_overdisp_broodoff_czi,
                                    data=Owls,
                                    family="zipoisson",
                                    verbose=FALSE))
```

```
## Warning in MCMCglmm(fixef3, rcov = ~idh(trait):units, random =
## ~idh(trait):Nest, : some fixed effects are not estimable and have
## been removed. Use singular.ok=TRUE to sample these effects, but use an
## informative prior!
```

brms

```
time.brms_czi = tfun(m1.brms_czi <<- brm(brmsformula(form,zi=ziform),
                                  data = Owls,
                                  family="zero_inflated_poisson",
                                  save_dso=TRUE))
```

```
## Compiling the C++ model
```

```
time.brms_czi2 = tfun(m1.brms_czi2 <<- update(m1.brms_czi))
```

Comparison

Timings:

```
sort(c(TMB=time.tmb_czi,MCMCglmm=time.mcmc_czi,brms=time.brms_czi,
       brms2=time.brms2))
```

```
##       TMB MCMCglmm    brms2     brms
##     1.651    8.519   40.727   87.326
```

Coefficients:

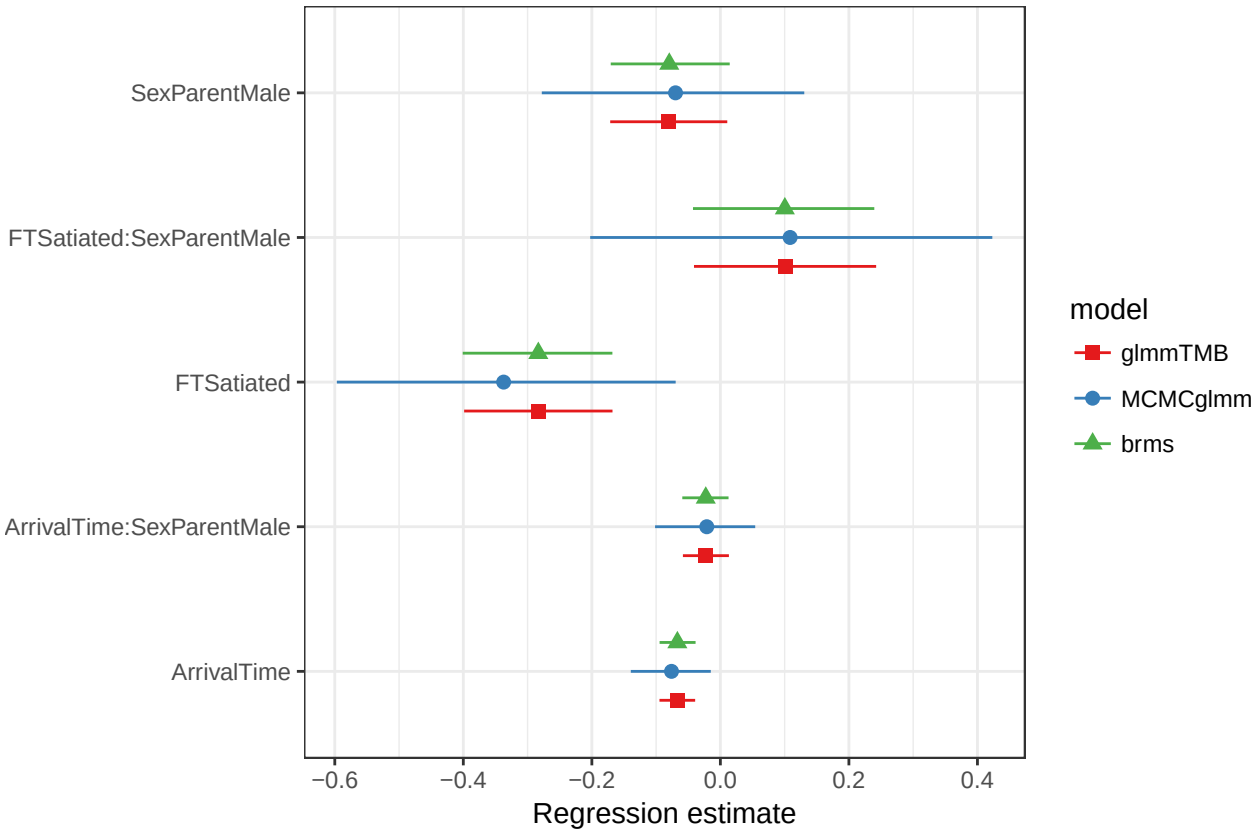

Figure C.2 – Estimated fixed-effect coefficients Estimates are from the same zero-inflated Poisson model with predictors on zero-inflation fit using functions `glmmTMB`, `MCMCglmm`, and `brm`.

References

Bolker, Benjamin M., Beth Gardner, Mark Maunder, Casper W. Berg, Mollie Brooks, Liza Comita, Elizabeth Crone, et al. 2013. "Strategies for Fitting Nonlinear Ecological Models in R, AD Model Builder, and BUGS." Edited by Satu Ramula. *Methods in Ecology and Evolution* 4 (6): 501–12. doi:10.1111/2041-210X.12044.

Roulin, Alexandre, and Louis-Felix Bersier. 2007. "Nestling Barn Owls Beg More Intensely in the Presence of Their Mother Than in the Presence of Their Father." *Animal Behaviour* 74 (4): 1099–1106. doi:10.1016/j.anbehav.2007.01.027.

Zuur, Alain F., Elena N. Ieno, Neil J. Walker, Anatoly A. Saveliev, and Graham M. Smith. 2009. *Mixed Effects Models and Extensions in Ecology with R*. Springer.