Compare glmmTMB, glmmADMB, MCMCglmm, and brms

Mollie Brooks

2016-11-24

Preliminaries

Load packages

```
library(glmmTMB)
library(glmmADMB)
library(MCMCglmm)
library(brms)
library(broom)#for tidy
library(plyr)
library(coefplot)#for position_dodgev, may need to install from source
library(ggplot2); theme_set(theme_bw())
```

Load in data and reorganize it

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

Fitting the same model in 4 packages

glmmTMB

```
time.tmb = unname(system.time(m1.tmb <- glmmTMB(NCalls~(FoodTreatment + ArrivalTime) * SexParent +
    offset(logBroodSize) + (1|Nest), ziformula=~1, data = Owls, family="poisson"))['elapsed'])
```

glmmADMB

```
time.admb = unname(system.time(m1.admb <- glmmadmb(NCalls~(FoodTreatment + ArrivalTime) * SexParent +
    offset(logBroodSize) + (1 | Nest), zeroInflation=TRUE, data = Owls, family="poisson"))['elapsed'])
```

MCMCglmm

```
offvec = c(1,1,2,rep(1,5))
fixef2 = NCalls~trait-1+ ## intercept terms for both count and binary terms
   at.level(trait,1):logBroodSize+
   at.level(trait,1):((FoodTreatment+ArrivalTime)*SexParent)
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=unname(system.time(m1.mcmc <- MCMCglmm(fixef2,
                                      rcov=~idh(trait):units,
                                      random=~idh(trait):Nest,
                                      prior=prior_overdisp_broodoff,
                                      data=Owls,
                                      family="zipoisson",
                                      verbose=FALSE))['elapsed'])
```

brms

```
time.brms = unname(system.time(m1.brms <- brm(NCalls ~(FoodTreatment + ArrivalTime) * SexParent +
    offset(logBroodSize) + (1 | Nest), data = Owls, family="zero_inflated_poisson"))['elapsed'])
```

```
##
## SAMPLING FOR MODEL 'zero_inflated_poisson(log) brms-model' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 6.41661 seconds (Warm-up)
##                3.27425 seconds (Sampling)
##                9.69086 seconds (Total)
##
##
## SAMPLING FOR MODEL 'zero_inflated_poisson(log) brms-model' NOW (CHAIN 2).
##
## Chain 2, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 6.56285 seconds (Warm-up)
##                3.21824 seconds (Sampling)
##                9.78108 seconds (Total)
##
##
## SAMPLING FOR MODEL 'zero_inflated_poisson(log) brms-model' NOW (CHAIN 3).
##
## Chain 3, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 5.61555 seconds (Warm-up)
##                4.21565 seconds (Sampling)
##                9.8312 seconds (Total)
##
##
## SAMPLING FOR MODEL 'zero_inflated_poisson(log) brms-model' NOW (CHAIN 4).
##
## Chain 4, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 6.7394 seconds (Warm-up)
##                3.5606 seconds (Sampling)
##                10.3 seconds (Total)
```

Compare the timings

```
time.tmb
```

```
## [1] 1.494
```

```
time.admb
```

```
## [1] 23.161
```

```
time.mcmc
```

```
## [1] 8.401
```

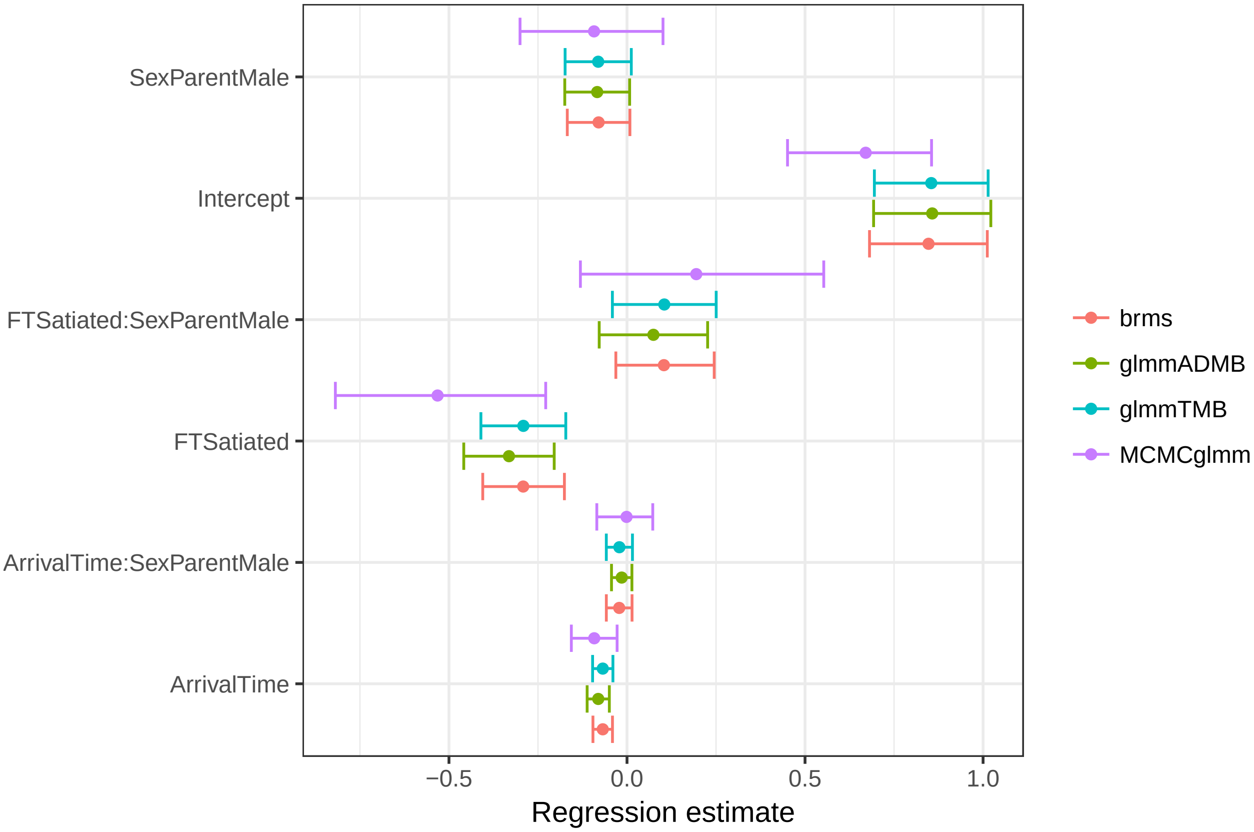
```
time.brms
```

```
## [1] 73.497
```

`glmmTMB` fit the model in less than 2 seconds. Other methods were slower, but `MCMCglmm` was in the same order of magnitude.

Comparing the results

Compare the fits



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.