Parallel Computing in the R package `glmm`

Sydney Benson

December 11, 2018

Contents

1 Introduction **2**

2 Additional Model-Fitting Arguments **2**

1 Introduction

The R package `glmm` approximates the likelihood function for generalized linear mixed models (GLMMs) with a canonical link. `glmm` calculates and maximizes the Monte Carlo likelihood approximation (MCLA) to find Monte Carlo maximum likelihood estimates (MCMLEs) for the fixed effects and variance components. The value, gradient vector, and Hessian matrix of the MCLA are calculated to maximize the likelihood and the MCMLEs. The Hessian of the MCLA is used to calculate the standard errors for the MCMLEs.

In version 1.2.4, the R package `glmm` has been revised to calculate the value, gradient vector and Hessian matrix in parallel. This addition has incorporated an optional argument to the `glmm` command, an increased number of outputs, and has decreased the time it takes to fit the model in most cases.

2 Additional Model-Fitting Arguments

In the following code, we fit the model using the `glmm` command and save the model under the name `sal`. The final argument for the `glmm` command is `cluster`. The `cluster` argument is optional and, by default, R will create a cluster that uses only a single core. This will force the calculations for the value, gradient vector and Hessian matrix to be done sequentially instead of simultaneously. However, if you choose to specify a cluster using a different number of cores, you may do that using the `cluster` argument. In this example, we will use a cluster with two cores.

Using a cluster with multiple cores is useful for reducing the time the `glmm` command takes to run, thus allowing for an increased `m` without additional computational expense. An increased `m` allows the model to provide more accurate estimates, but can also increase the model-fitting time exponentially. Using multiple cores can help mitigate this problem. We can see the time reduction made by running the calculations in parallel using the `proc.time` command. First, the model-fitting time for the command using one core and $m = 10^4$ is given.

```
library(glmm)
data(salamander)
```

```
clust <- makeCluster(1)
set.seed(1234)
start <- proc.time()
sal <- glmm(Mate ~ 0 + Cross, random = list(~ 0 + Female,
~ 0 + Male), varcomps.names = c("F", "M"), data = salamander,
family.glmm = bernoulli.glmm, m = 10^4, debug = TRUE, cluster = clust)
proc.time() - start

##    user  system elapsed
##  43.379   0.288 144.095

stopCluster(clust)
```

Next, the model-fitting time for the command using two cores with the same Monte Carlo sample size is found.

```
clust <- makeCluster(2)
set.seed(1234)
start <- proc.time()
sal <- glmm(Mate ~ 0 + Cross, random = list(~ 0 + Female,
~ 0 + Male), varcomps.names = c("F", "M"), data = salamander,
family.glmm = bernoulli.glmm, m = 10^4, debug = TRUE, cluster = clust)
proc.time() - start

##    user  system elapsed
##  43.048   0.160  76.887

stopCluster(clust)
```

To read about the other arguments in the `glmm` command, please read "An Introduction to Model-Fitting with the R package `glmm`".