Model-Fitting in Parallel with the R package `glmm`

Sydney Benson

August 17, 2018

Contents

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 at 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 added an optional argument to the `glmm` command, additional output, and has decreased the time it takes to fit the model.

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 `cores`. The `cores` argument is optional and, by default, R will calculate the optimal number of cores to use for the parallel calculations if `cores` is not specified. The optimal number of cores is one less than the number of cores available in the device. However, if you choose to specify a different number of cores, you may do that using the `cores` argument. In this example, we will use two cores.

Note: you may only specify the number of cores you want to use if it is less than or equal to the optimal number of cores that R defines. If you specify a number greater than the optimal number defined by R, the number of cores used will default to R's optimal number.

Using multiple cores is useful for reducing the time the `glmm` command takes to run, thus allowing for an increased `m` without additional computational expense. We can see the time reduction made by running the calculations in parallel using the `proc.time` command. The times shown here are from fitting a model on a MacBook Air that cost 1000 USD in 2015.

```
> library(glmm)
> data(salamander)
> 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, cores = 1)
> proc.time() - start
```

```
   user  system elapsed
101.479   7.791 341.730
```

```
> 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, cores = 2)
> proc.time() - start
```

```
   user  system elapsed
 90.983   5.009 237.323
```

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

3 Additional Output

Using the `names` command allows us to see the additional output that we can access, beyond the model summary. With the addition of parallel calculations, we also have the addition of the `cores` output. `cores` will tell you how many cores were used during the value, gradient vector and Hessian matrix calculations.

```
> names(sal)
```

```
 [1] "beta"                "nu"                  "likelihood.value"
 [4] "likelihood.gradient" "likelihood.hessian"  "trust.converged"
 [7] "mod.mcml"            "fixedcall"           "randcall"
[10] "x"                   "y"                   "z"
[13] "family.glmm"         "call"                "varcomps.names"
[16] "varcomps.equal"      "umat"                "pvec"
[19] "beta.pql"            "nu.pql"              "u.pql"
[22] "zeta"                "cores"               "debug"
```

```
> sal$cores

[1] 2
```

To read about the model summary or the other additional output provided when using the `glmm` command, please read ”An Introduction to Model-Fitting with the R package `glmm`”