Getting More from Generalized Linear Mixed Models in R

Sydney Benson
Advisor: Christina Knudson, Ph.D.

bens0104@stthomas.edu
www.github.com/bensonsyd
www.linkedin.com/in/bensonsyd

Introduction

In the R package **glmm**, more valuable parameter estimates for generalized linear mixed models come with a costly increase in the Monte Carlo sample size. Thus, to alleviate some of the cost associated with improved estimates, a parallel computing component has been added to the package. The details of parallel computing in R and the effect of this component in the **glmm** package is described below.

The R Package glmm

The goal of the R Package **glmm** is to linearly model a data set without assuming independent or normally distributed responses, assumptions commonly made by other modeling techniques. These assumptions are frequently broken, as illustrated in the examples to the right.

Modeling Techniques

Simple linear model assumptions:
- Independent responses
- Normally distributed responses
- Responses have equal variance

Generalized linear models assume independent, but not normally distributed responses. This occurs in the case of:
- Log odds of your favorite sports team winning a game (binomial)
- Log mean number of students per class at your university (Poisson or negative binomial)

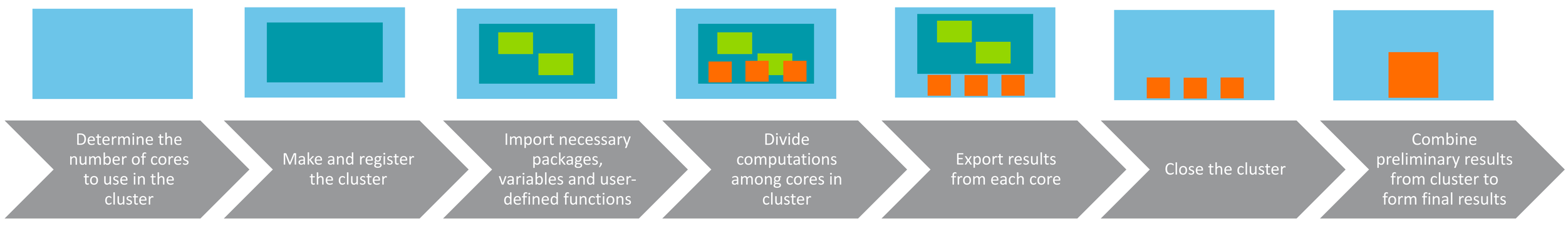
Linear mixed models assume normally distributed, but not independent responses. This occurs in the case of:
- Repeated measurements on individuals
- Measurements on related individuals (i.e., siblings, parent/child, other relatives)

Generalized linear mixed models assume neither independent, nor normally distributed responses. These models use fixed and random effects to account for the differences seen within a group of related responses. Link functions are used to account for non-normally distributed responses.

Parallel Computing

A type of computation where calculations or processes are executed simultaneously rather than sequentially. Consequently, processing time decreases.

Parallel Computing in R



Setting Up the Cluster
For the R package **glmm**, the user creates a cluster and includes it as one of the arguments in the glmm function. The cluster is then registered in the function.

Using the Cluster
Variables and user-defined functions are imported into the cluster multiple times by the *glmm* function. First, the cluster is used to generate the random effects matrix; each core generates part of the matrix. Then, the cluster is used when calculations for the value of the log-likelihood and gradient are performed; Finally, the cluster is used when calculations for the Hessian are performed. For each of these calculations the cores use their respective partial-matrices to obtain a preliminary value of the log-likelihood, gradient and Hessian. These values are exported from the cluster by naming the results of the parallel calculations.

Closing the Cluster and Obtaining Results
For the R package **glmm**, the cluster can be closed by the user after the *glmm* function has been executed. The final log-likelihood value, gradient and Hessian are obtained by recombining the preliminary results from each core.

A Practical Example: Salamander Mating

The Salamander data set is a classic example for generalized linear mixed models. It comes from an experiment conducted at the University of Chicago in 1986 which studied the interbreeding of two populations of Mountain Dusky salamanders. The question they aimed to answer was:

Do salamanders prefer to mate with salamanders from their own population?

Each salamander was studied in multiple different mating-pair scenarios, so there are multiple measurements on each individual and the responses are correlated. The responses are either mating success or mating fail, so the responses are not normally distributed. It is assumed that each salamander's personalized tendency to mate is unique and independent of the other salamanders. These assumptions result in this code, which will build the generalized linear mixed model using the R package **glmm**:

```
sal <- glmm(Mate ~ 0 + Cross,
            random = list( ~ 0 + Female, ~ 0 + Male ),
            varcomps.names = c( "F" , "M" ),
            data = salamander,  m = 10^5,
            family.glmm = bernoulli.glmm)
```

A Mountain Dusky Salamander.

For the *fixed* argument, the `0 + Cross` notation is used to generate coefficient estimates for the crosses without using a reference group. For the *random* argument, the `0 + Female/0 + Male` notation is used to center the random effects at 0. *m* is the Monte Carlo sample size and a larger *m* will provide more accurate estimates.

Below is some of the output produced by the model. *R* indicates a Rough Butt salamander and *W* indicates a White Side salamander. The first letter denotes the female salamander type and the second letter denotes the male salamander type. The estimates are the log-odds for each cross type and the significance of each estimate is displayed. These estimates are given as probabilities in the table below.

```
Fixed Effects:
         Estimate Std. Error z value Pr(>|z|)
CrossR/R   0.9612     0.5123   1.876   0.0606 .
CrossR/W   0.3848     0.8188   0.470   0.6384
CrossW/R  -2.0454     0.5204  -3.931 8.47e-05 ***
CrossW/W   1.0294     0.4677   2.201   0.0277 *
```

Cross	RR	WW	RW	WR
Probability of Mating	0.7234	0.7368	0.5950	0.1145

Users of the R package **glmm** can implement the new parallel computing procedures by creating a cluster of their chosen size and type and naming it. The cluster can then be fed into the glmm function using the *cluster* argument.

In the example below, a cluster with 3 cores has been created using the *makeCluster* command from the R package **parallel**. The cluster was named "cluster" and the command to create the generalized linear mixed model, implementing parallel computing, is altered from the command above by adding the *cluster* argument:

```
sal <- glmm(Mate ~ 0 + Cross,
            random = list( ~ 0 + Female, ~ 0 + Male ),
            varcomps.names = c( "F" , "M" ),
            data = salamander,  m = 10^5,
            family.glmm = bernoulli.glmm,
             cluster = cluster)
```

Using a Monte Carlo sample size of 10^5, the function takes approximately 19.9 minutes to build the model prior to the parallel computing implementation and only 15.5 minutes when three cores are used. The time saved using the function with parallel computing increases as the Monte Carlo sample size increases.

Monte Carlo Sample Size

The Monte Carlo sample size is denoted by *m* in the function arguments. Monte Carlo Likelihood Approximation (MCLA), an iterative importance sampling procedure, is used in this package to determine the true parameters of the model.

Since a larger *m* equates to more iterations of the sampling procedure, it follows that a larger *m* will provide improved parameter estimates for the model. The drawback of using a larger *m* is that the model takes longer to be created. A larger *m* means that model building is a more computationally expensive process.

To decrease the computational expense of using a large *m*, the ability to compute the value of the log-likelihood approximation, gradient and Hessian in parallel has been added to this package. Details of this addition and how the user can utilize this change are specified to the right.

Selected References

1. Knudson C. (2015). *glmm: Generalized Linear Mixed Models via Monte Carlo Likelihood Approximation*. R package version 1.0.2, URL http://CRAN.R-project.org/package=glmm.
2. Ripley B., Tierney L., Urbanek S. (2017). *Package 'parallel'* R package version 3.3.1, URL http://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf.
3. Microsoft Corporation and Steve Weston (2017). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.11, URL https://CRAN.R-project.org/package=doParallel.
4. Microsoft and Steve Weston (2017). *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.4, URL https://CRAN.R-project.org/package=foreach.
5. Steve Weston and Hadley Wickham (2014). *itertools: Iterator Tools*. R package version 0.1.3, URL https://CRAN.R-project.org/package=itertools.

For a full list of references and parallel computing details, visit https://github.com/bensonsyd/GLMM/blob/master/Univ_St_Thomas/R_CAM_Paper.pdf.