MCMCglmm: Markov chain Monte Carlo methods for Generalised Linear Mixed Models.

J. D. Hadfield

January 26, 2009

Contents

MCMCglmm is a package for fitting Generalised Linear Mixed Models using Markov chain Monte Carlo techniques. Most commonly used distributions like the normal and the Poisson are supported together with some useful but less popular ones like the zero-inflated Poisson and the multinomial. Missing values and left, right and interval censoring are accommodated for all traits. The package also supports multi-trait models where the multiple responses can follow different types of distribution. The package allows various residual and random-effect variance structures to be specified including heterogeneous variances, unstructured covariance matrices and random regression (e.g. random slope models). Three special types of variance structure that can be specified are those associated with pedigrees (animal models), phylogenies (the comparative method) and measurement error (meta-analysis). The package makes heavy use of results in Sorensen and Gianola [2002] and Davis [2006] which taken together result in what is hopefully a fast and efficient routine. Most small to medium sized problems should take seconds to a few minutes, but large problems ($>$ 20,000 records) are possible. My interest is in evolutionary biology so there are also several functions for applying tensor analysis [Rice, 2004] to real data and functions for visualising and comparing matrices.

```
> library("MCMCglmm")
```

1 An Empirical Example

To demonstrate how to fit and interpret these models we will use some data from a pilot study on the Indian meal moth (*Plodia interpunctella*) and its granulosis virus, PiGV. This data was collected by Hannah Tidbury & Mike Boots at the University of Sheffield, and highlights a range of model fitting problems that are hard to solve with standard techniques. The aim of the study was to see if there was an association between the amount of phenoloxidase (PO) produced by the caterpillars and resistance to being artificially infected by the virus (measured as successfully reaching pupation). The difficulty with the experiment is that measuring PO is lethal, and so it is not possible to have individual measures for both PO and resistance to the virus. The solution was to take full-sib families and measure half the individuals for PO and infect the other half with the virus. The aim then was to estimate the 'genetic' correlation between the two measures.

```
> data(PlodiaPO)
```

We'll start with a simple analysis of PO which was box-cox transformed so that it was approximately normal.

```
> model1 <- MCMCglmm(PO ~ 1, random = ~FSfamily, data = PlodiaPO,
+     verbose = FALSE)
```

We've modelled an in intercept $\sim$`1` and we've modelled family effects $\sim$`FSfamily`. `MCMCglmm` returns the output as `mcmc` objects from the coda package. The first of these is `Sol` which returns the fixed effect estimates:

```
> plot(model1$Sol)
```

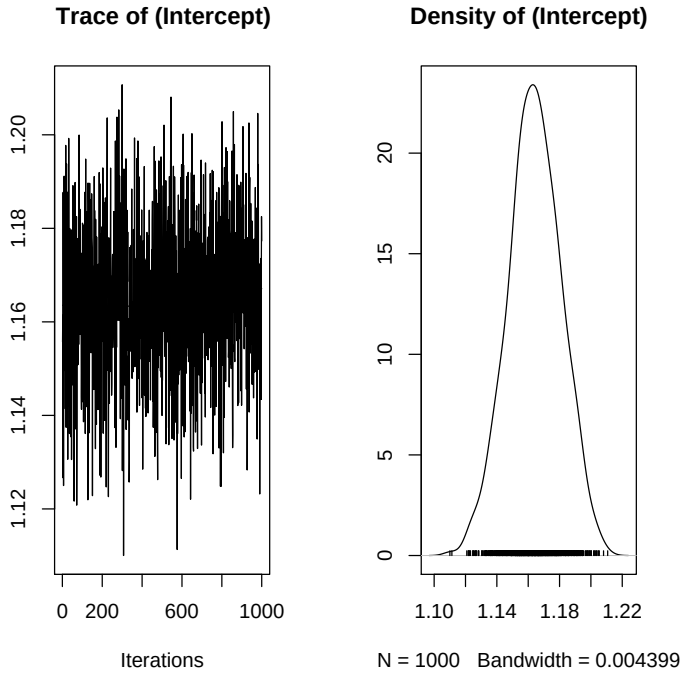

Figure 1: Posterior distribution of the intercept from model1

On the left of Figure 1 is a time series of the parameter as the MCMC iterates, and on the right is a posterior density estimate of the parameter (a smoothed histogram of the output). Making inferences about the parameter(s) from the MCMC output is simple. For example, the most likely value is where the posterior distribution peaks:

```
> posterior.mode(model1$Sol)

[1] 1.163622
```

The probability that the intercept is greater than 1.15 is simply the proportion of the output that is greater than 1.15:

```
> table(model1$Sol > 1.15)/1000

FALSE  TRUE
0.188 0.812
```

and the region of 95% support can be obtained by finding the interval with the highest posterior density:

```
> HPDinterval(model1$Sol, 0.95)

               lower    upper
(Intercept) 1.131230 1.195282
attr(,"Probability")
[1] 0.95
```

However, these metrics are only an accurate description of the true posterior distribution when the chain has converged (there is no systematic trend in the time series) and when each successive value in the output does not show strong dependence with the previous one. The level of dependence can be measured using an autocorrelation statistic:

```
> autocorr(model1$Sol)

, , (Intercept)

       (Intercept)
Lag 0   1.00000000
Lag 1  -0.01300480
Lag 5   0.01733648
Lag 10  0.01776713
Lag 50 -0.02985370
```

As you can see, the autocorrelation between successive values (Lag 1) is small and we probably have obtained close to 1000 independent samples from the posterior. The chain is said to have mixed well. If the chain had not converged or mixed we could extend the burn-in in period (`burnin`), increase the total number of iterations (`nitt`) or increase the interval between which successive samples are stored (`thin`).

By default `MCMCglmm` will not store the estimates for each family effect, as generally we're not so interested in individual families but about the variation in these family effects across families. Variance estimates are stored in the object `VCV`:

```
> plot(model1$VCV)
```

There are two variances, one associated with family effects and one associated with `units` (Fig 2). `units` is a reserved variable which has a factor level for each row of the response. The default in `MCMCglmm` is to specify the residual term as `rcov=~units`. In this instance each data point corresponds to a unique level of `units` and therefore we simply interpret the units variance as we would the residual variance in most models. Often we're interested in knowing the

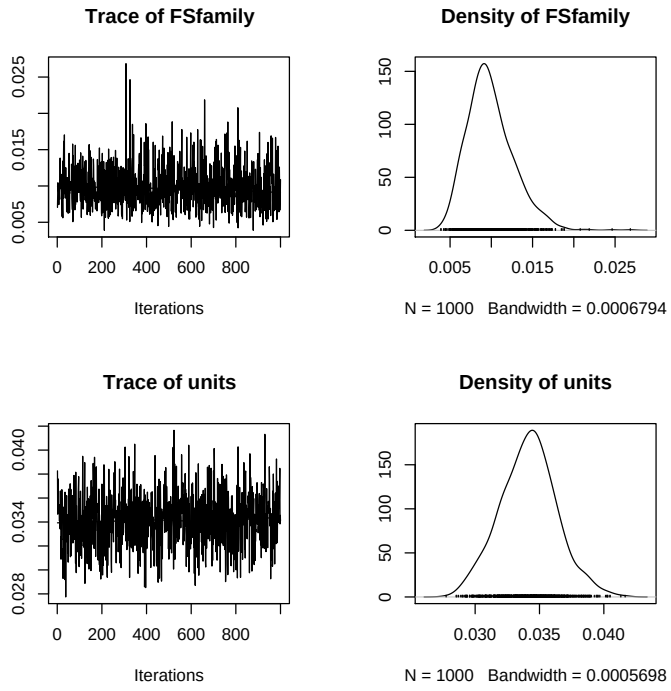

Figure 2: Posterior distribution of the variance components from `model1`

proportion of total variance explained by the random effect, which would involve dividing the family variation by the sum of the two variance components. Obtaining some measure of confidence for this proportion is not straightforward using standard methods such as REML, but with MCMC it is easy. For example, the region of 95% support:

```
> HPDinterval(model1$VCV[, "FSfamily"]/(model1$VCV[, "FSfamily"] +
+     model1$VCV[, "units"]))

         lower     upper
var1 0.1290778 0.3210156
attr(,"Probability")
[1] 0.95
```

A valid posterior for any transformation of model parameters can be obtained by applying that transformation to each sample and analysing the result.

This model was fitted without explicitly specifying a prior and the default priors were used. In certain circumstances this can lead to inferential and numer-

ical problems[1] and Section 2.5 should be read carefully. To assess the sensitivity of the analysis to alternative prior specifications lets have a weak prior (n=1) equal to half the variance in PO for each variance component:

```
> halfV <- var(PlodiaPO$PO)/2
> prior = list(R = list(n = 1, V = halfV), G = list(G1 = list(n = 1,
+     V = halfV)))
> model2 <- MCMCglmm(PO ~ 1, random = ~FSfamily, data = PlodiaPO,
+     verbose = FALSE, prior = prior)
> plot(mcmc.list(model1$VCV, model2$VCV))
```

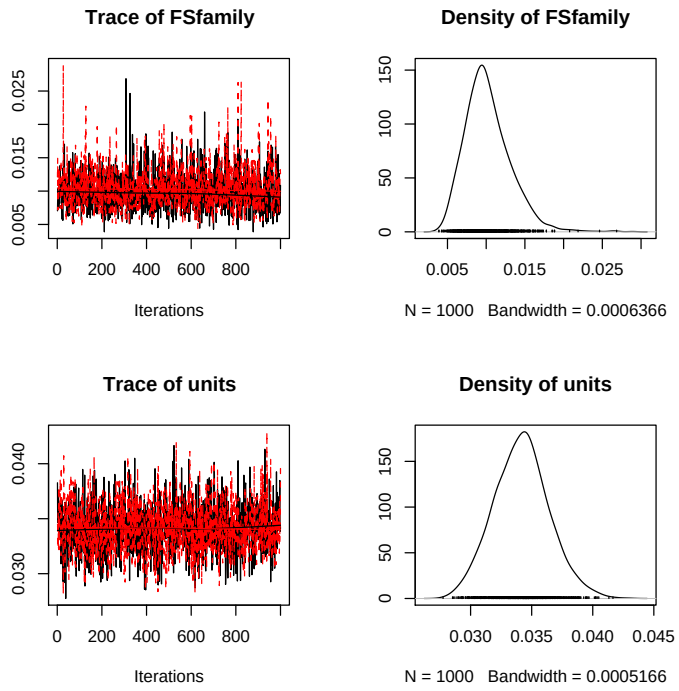

Figure 3: MCMC output from `model1` (black trace) which was fitted without a prior on the variance components (n=0) and `model2` in which a weak but proper prior (n=1, V=0.02) was used (red trace)

As you can see from Figure 3 the two models do not completely coincide but they are very close; most of the information is coming from the data.

[1] When this results in numerical problems (variances close to zero or correlations close to -1 or 1) MCMCglmm can fail, although I've tried make it do so nicely.

1.1 Simple univariate model - Binomial response

The data on resistance to the virus have been aggregated into a binomial response for each family

```
> data(PlodiaR)
```

For multinomial data, of which the binomial is a special case, we need to specify for each category (in this case `Pupated` (not infected) and `Infected`) the number of counts that were recorded, and we do this by passing the multiple response variables using `cbind`

```
> model3 <- MCMCglmm(cbind(Pupated, Infected) ~ 1, family = "multinomial2",
+     data = PlodiaR, verbose = FALSE)
```

Note that the family is `multinomial` and the final number is the number of categories (in this case 2). Also, we have not fitted a random effect. Because the data are counts for each family we cannot fit `FSfamily` levels as random effects because they would be confounded with the residuals. The residual variance is however estimated and we can interpret this over-dispersion as variation in the binomial probability across families. Figure 4 shows that there is ample over-dispersion (the variance is estimated to be well away from zero) leading us to believe that families vary in their probability of resistance.

```
> plot(model3$VCV)
```

Variation in the probabilities are modelled on the logit scale and are assumed to be normal on that scale. We can get a feel for what this looks like on the data scale by getting the posterior modes for the intercept and the variance, and generating random numbers from this distribution. It is clear from Figure 5 that the variation between families in the probability of resistance is non-negligible: many families are likely to have a 0.1 probability but many are also likely to have a 0.5 probability.

```
> PMmu <- posterior.mode(model3$Sol)
> PMv <- posterior.mode(model3$VCV)
> PMprob <- inv.logit(rnorm(10000, PMmu, sqrt(PMv)))
> hist(PMprob)
```

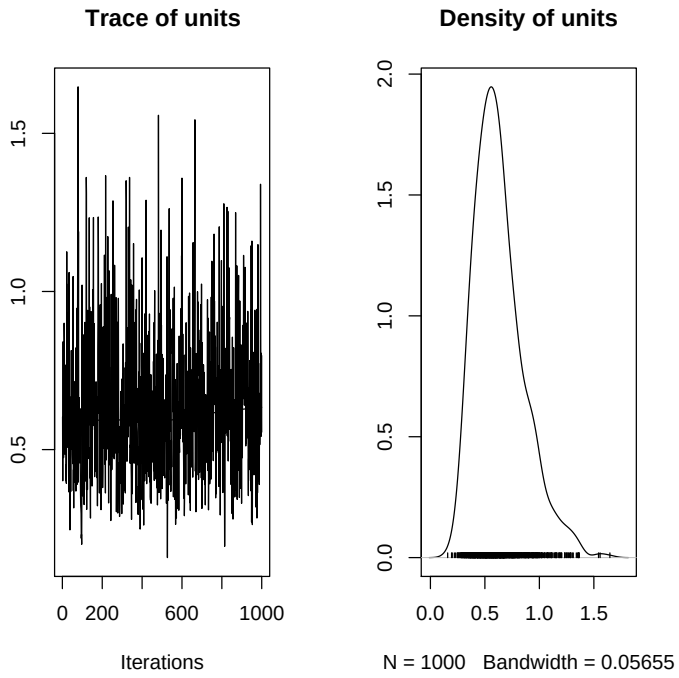

Figure 4: Residual variance from `model3` which models binomial over-dispersion

1.2 Simple univariate model - Binary response

There is another way we could have fitted `model3`, and that is by expanding the binomial response for each family into a binary response for each individual within each family.

```
> data(PlodiaRB)
> prior = list(R = list(V = 1, n = 0, fix = 1), G = list(G1 = list(V = 1,
+     n = 0)))
> model4 <- MCMCglmm(Pupated ~ 1, random = ~FSfamily, family = "categorical",
+     data = PlodiaRB, prior = prior, verbose = FALSE)
```

Binary responses (or other responses that are discrete) are of the family `"categorical"`. They pose a special problem because the residual variance cannot be estimated because the variance is uniquely determined by the mean. When proper priors are used this does not pose a problem to a Bayesian analyst, but we do have to be aware that all the information regarding the parameter is coming from the prior. Because of this it is usual to fix these parameters at some value (conventionally one for variances and 0 for covariances) and this

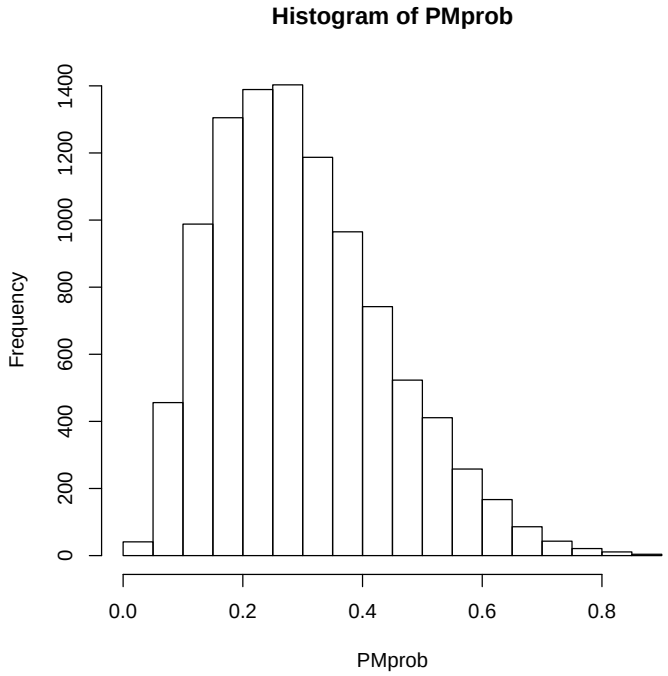

Figure 5: The predicted distribution of resistance probabilities across families assuming the posterior modes of `model3` are correct.

can be done by specifying `fix=1` in the appropriate prior element. An identical procedure for simple models such as this would be to specify a very informative prior (`n` is large) around `V`:

```
> prior = list(R = list(V = 1, n = 1e+06), G = list(G1 = list(V = 1,
+     n = 0)))
```

You will notice that the parameters of `model3` and `model4` do not take on the same values. This is because in `model3` we implicitly assumed that the within individual variance was zero (as in standard quasi-binomial models) rather than 1. This is essentially an arbitrary choice because we have no way of seeing this variation in real data. This may seem a bit disconcerting but it should not worry you unduly; the models are just reparameterisations of each other and we can see this by approximating the posterior distribution of the mean on the data scale:

```
> mu.data.scale <- function(logit.mu, logit.var) {
+     mean(inv.logit(rnorm(10000, logit.mu, sqrt(logit.var))))
```

```
+ }
> mu.model3 <- mcmc(mapply(mu.data.scale, model3$Sol, model3$VCV))
> mu.model4 <- mcmc(mapply(mu.data.scale, model4$Sol, rowSums(model4$VCV)))
> summary(mu.model3)

Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
     0.2976256      0.0259197      0.0008197      0.0009884

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.2493 0.2793 0.2974 0.3140 0.3503

> summary(mu.model4)

Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.294072       0.026090       0.000825       0.001039

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.2473 0.2774 0.2926 0.3093 0.3506
```

The posterior summaries are virtually identical up to Monte Carlo error.

1.3 Bivariate model - Binary and Gaussian response

Expanding the binomial data into binary data allows us to fit a model in which we can estimate what we're interested in: the correlation between PO and probability of resistance at the family level. We start by creating missing PO values for those individuals in the resistance experiment and missing Pupated/Infected

values for those individuals only measured for PO. We can do this by creating id's for each data point and then merging the two data frames:

```
> PlodiaPO$ID <- 1:dim(PlodiaPO)[1]
> PlodiaRB$ID <- dim(PlodiaPO)[1] + 1:dim(PlodiaRB)[1]
> PlodiaPORB <- merge(PlodiaPO, PlodiaRB, all = TRUE)
```

Fitting a sensible model to bivariate data requires a bit more work, because usually we will need to specify different models for each response but also specify a model that accounts for dependence between the two responses (otherwise we may as well just fit to univariate models). Fitting an appropriate model to the Plodia data is particularly fiddly but it serves as a useful example of the types of variance structures that can be fitted.

```
> prior = list(R = list(V = diag(2), n = 0, fix = 2), G = list(G1 = list(V = diag(2),
+     n = 1)))
> model5 <- MCMCglmm(cbind(PO, Pupated) ~ trait - 1, random = ~us(trait):FSfamily,
+     rcov = ~idh(trait):units, family = c("gaussian", "categorical"),
+     data = PlodiaPORB, prior = prior, verbose = FALSE)
```

For multivariate models we will usually want to make use of the reserved variable `trait` which indexes columns of the response. By fitting `trait` as a fixed effect we allow the two responses to have different means. I usually fit the model ∼`trait-1` rather than just ∼`trait` as this fits trait specific intercepts rather than an intercept for trait 1 (`PO`) and a contrast for trait 2 (`Pupated`). Likewise, just fitting a single family variance using ∼`FSfamily` implies that families have respond in the same way to both traits, so we usually want to form an interaction. The simplest would be ∼`trait:FSfamily` but this would imply that the family variance for each trait is equal and the family effects for trait 1 are uncorrelated with the family effects for trait 2. More reasonable variance structures can be formed using `idh` and `us`. `idh` allows different variances across the traits but assumes that the family effects for trait 1 are uncorrelated with the family effects for trait 2. `us` is the most general variance structure and allows different variances across the traits and allows covariances to exist between them. In both these cases the variance structure becomes a 2x2 matrix (because there are two traits) rather than a scalar (1x1 matrix).

$$\begin{aligned} \texttt{idh(trait):FSfamily} &= \begin{bmatrix} \sigma^2_{F(T_1)} & 0 \\ 0 & \sigma^2_{F(T_2)} \end{bmatrix} \\ \texttt{us(trait):FSfamily} &= \begin{bmatrix} \sigma^2_{F(T_1)} & \sigma_{F(T_1,T_2)} \\ \sigma_{F(T_2,T_1)} & \sigma^2_{F(T_2)} \end{bmatrix} \end{aligned} \tag{1}$$

Here, $\sigma^2_{F(T_1)}$ stands for the variance across families (F) for trait 1 (T_1) and $\sigma_{F(T_1,T_2)}$ the covariance between trait 1 and trait 2 across families. Section 2.1 covers this in more detail.

In `model5` we have used an `idh` structure for the residual component because both traits have never been measured on the same individual so the residual covariance cannot be estimated. As before we have fixed the residual variance of the binary trait to 1 because it too cannot be estimated from the data. Family members on the other hand have been measured for both traits and the covariance can be estimated, indeed it was the purpose of the experiment.

We can get the posterior distribution of the correlation between family effects by using the distribution of the correlation estimated for each iteration of the chain (Figure 6)

```
> Pcor <- model5$VCV[, 2]/sqrt(model5$VCV[, 1] * model5$VCV[, 4])
> plot(Pcor)
```

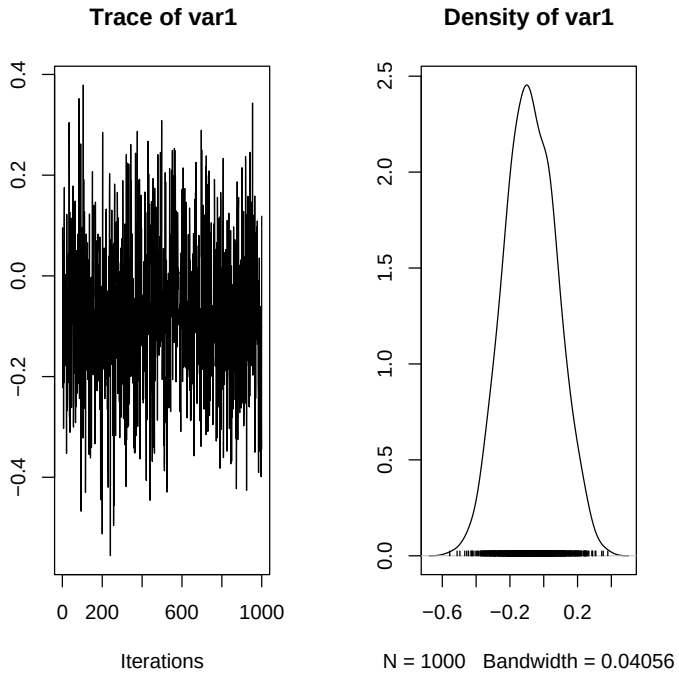

Figure 6: The posterior distribution of the family level correlation between resistance and PO from `model5`.

1.4 An Animal model

Quantitative geneticists will have noticed that the heritability of PO is not the proportion of variance explained by family in `model1`, because full-sibs only

share 50% of their genes. With such a simple experimental design the estimate for the heritability can be obtained easily by multiplying the proportion of variance explained by family by two (Figure 7):

```
> h2 <- 2 * model1$VCV[, "FSfamily"]/(model1$VCV[, "FSfamily"] +
+     model1$VCV[, "units"])
> plot(h2)
```

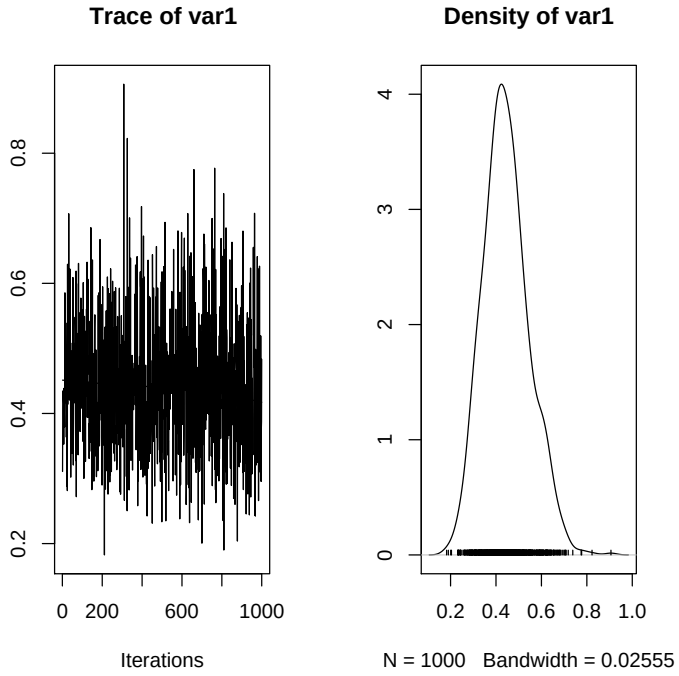

Figure 7: The posterior distribution of PO heritability from `model1`.

In this instance the proportion of the variance explained by family was small enough and known with enough precision that the posterior did not exceed 0.5. If it had of done then the resulting heritability would have had support at values greater than 1. This would be inconsistent with the model because it would mean that genes explain more of the variation than actually exists. An alternative model is to explicitly factor in the proportion of genes shared by two individuals. This has the advantage that the genetic variance can't exceed the total variance, and allows more complicated family structures to be modelled. For example, when some individuals share 50% of their genes and others 25% and so on. This type of model is known as an animal model and to be able to fit them we need to have a pedigree table that represents the genealogy of

the individuals. This is a 3 column table with id, mother, and father in each column. Individuals with unknown parents have `NA` for their mother and/or father, and all individuals that appear as parents must be represented in the id column. The Plodia data were collected on 50 families named `F1` through to `F50`. We can make a pedigree for the individuals in the data by adding 'dummy' parents (e.g. `F1mother` and `F1father`):

```
> ID <- c(paste("F", 1:50, "mother", sep = ""), paste("F", 1:50,
+     "father", sep = ""), PlodiaPO$ID)
> DAM <- c(rep(NA, 100), paste(PlodiaPO$FSfamily, "mother", sep = ""))
> SIRE <- c(rep(NA, 100), paste(PlodiaPO$FSfamily, "father", sep = ""))
> pedigree <- cbind(ID, DAM, SIRE)
```

The model has the same form as `model1` except `animal` is fitted as a random effect rather than `FSfamily`. `animal` is a special variable in `MCMCglmm` and it will always be associated with the id levels in the first column of the pedigree.

```
> PlodiaPO$animal <- 1:dim(PlodiaPO)[1]
> model6 <- MCMCglmm(PO ~ 1, random = ~animal, data = PlodiaPO,
+     pedigree = pedigree, verbose = FALSE)
```

A warning message appears saying that missing records have been added. This is because there are 100 parents in the pedigree file that do not have data records. To allow the model to run these missing data are augmented and integrated over. For this experiment, fitting the model as an animal model is inefficient because of this and the chain mixes poorly compared to `model1`

```
> autocorr(model6$VCV)

, , animal

            animal        units
Lag 0   1.00000000 -0.779125329
Lag 1   0.62893205 -0.577521640
Lag 5   0.02671291 -0.004937284
Lag 10 -0.04527634  0.046379767
Lag 50 -0.12431376  0.098593830

, , units

            animal        units
Lag 0  -0.77912533  1.000000000
Lag 1  -0.54949653  0.508155180
Lag 5  -0.01988529 -0.008323217
Lag 10  0.05968535 -0.047905166
Lag 50  0.07254662 -0.060898826
```

That being said the two analyses give the same answer we just have to run the second for longer to get the same degree of accuracy[2](Figure 8):

```
> h2AM <- model6$VCV[, 1]/(model6$VCV[, 1] + model6$VCV[, 2])
> plot(mcmc.list(h2, h2AM))
```

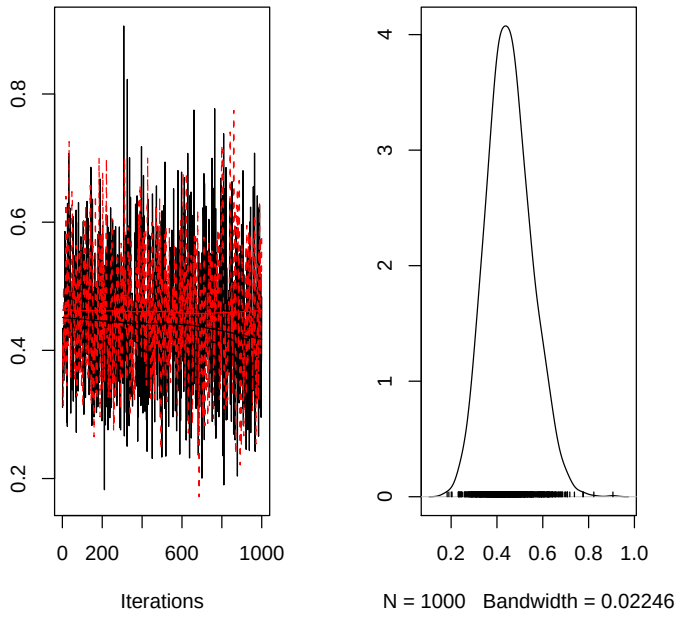

Figure 8: The posterior distribution of PO heritability from the animal model `model6`.

2 A Mathematical Tour

The empirical examples only cover a small set of possible models that are possible using `MCMCglmm`. In this section I cover a wider range of models using a combination of code and maths. I will use a notation that is flexible enough to cover multivariate generalised linear mixed models when the multiple responses

[2]note that they are not perfectly identical because the default prior of `n=0` is not uninformative. For a single variance component `V=0, n=-2` is uninformative and using these priors the two methods give indistinguishable results at moderate heritabilities.

come from different distributions. The multiple responses are passed to `MCMCglmm` as a matrix ($\mathbf{Y}$) using `cbind()` but it will be easier to think of them concatenated column-wise into a vector denoted by $\mathbf{y}$. Each element of $\mathbf{y}$ is associated with some probability distribution from the exponential family such as the Poisson or the normal (See Table 2.9). I denote the canonical parameter of the distribution as l as it is often called a latent variable or liability in quantitative genetics. The canonical parameter is related to the more familiar distribution parameter through the canonical link function $g()$. For example, the mean of the Poisson distribution $\lambda = g^{-1}(l)$ where $g()$ is log() and $g^{-1}()$ is exp(). The probability of the i^{th} data point conditional on l_i is denoted as

$$p_i(y_i|l_i) \tag{2}$$

where p_i is the probability density function associated with y_i. The linear mixed model is applied to l not y:

$$\mathbf{l} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e} \tag{3}$$

A special case is when p_i is the normal distribution. In this case $l_i = y_i$ (because the canonical link function is the identity link) and the probability $p_i(y_i|l_i = y_i)$ is always unity. In this case we can replace the LHS of Equation 3 with $\mathbf{y}$ to obtain the standard linear mixed model. The notation for the linear model is fairly standard: $\mathbf{X}$ is a design matrix relating fixed predictors to the data and $\mathbf{Z}$ is a design matrix relating random predictors to the data. These predictors have associated parameter vectors $\boldsymbol{\beta}$ and $\mathbf{u}$. In Bayesian analyses both these parameter vectors are technically random, but I will stick with the frequentist terminology of referring to them as fixed effects and random effects, respectively. The key difference between $\boldsymbol{\beta}$ and $\mathbf{u}$ is that u's are assumed to come from some distribution, the parameters of which are usually estimated, whereas the β's are not[3]. MCMCglmm assumes that the n_u u's follow a n_u-dimensional multivariate normal distribution with null mean vector and a structured (co)variance matrix. We will call this structure the G-structure. $\mathbf{e}$ is a vector of residuals which also come from a structured multivariate normal distribution the parameters of which are usually estimated. We will refer to this (co)variance structure as the R-structure.

The variables `trait` and `units` can be used as predictors in the model formulae to index the column and row to which an element of $\mathbf{l}$ originally belonged in $\mathbf{L}$, where $\mathbf{L}$ is the liability equivalent of the data before concatenation ($\mathbf{Y}$). The distribution associated with a data point can be specified in one of two ways. The easiest way is to specify them in the `family` argument where each distribution name is associated with a column of $\mathbf{Y}$. Alternatively, `family=NULL` can be specified if `data` contains a variable `family`.[4]

[3] This is not strictly true in a Bayesian analysis because $\boldsymbol{\beta}$ are assumed to be distributed according to the prior

[4] only `gaussian`, `poisson`, `exponential` and `categorical` distributed data can be specified this way, not variables where the dimension of $\mathbf{Y}$ and $\mathbf{L}$ are not equal.

2.1 Variance structures

The standard variance structure is to have effects i.i.d (independent and identically distributed). In matrix form this is represented by a diagonal matrix with a single variance parameter. For example,

$$\mathbf{R} = \begin{bmatrix} \sigma_e^2 & 0 & \dots & 0 \\ 0 & \sigma_e^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_e^2 \end{bmatrix} = \sigma_R^2 \mathbf{I} \tag{4}$$

where all covariances are set to zero (implying independence) and all diagonal elements are the same (implying identically distributed). However, there are alternatives. Imagine that two traits are both measured on a set of individuals. It would be natural to have different residual variances for each trait, and also allow residual covariances between the two traits measured in the same individual. The appropriate R structure would then have the form:

$$\mathbf{R} = \mathbf{V}_R \otimes \mathbf{I} = \begin{bmatrix} \sigma_{e_1}^2 & 0 & \dots & \sigma_{e_1,e_2} & 0 & \dots \\ 0 & \sigma_{e_1}^2 & \dots & 0 & \sigma_{e_1,e_2} & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \sigma_{e_2,e_1} & 0 & \dots & \sigma_{e_2}^2 & 0 & \dots \\ 0 & \sigma_{e_2,e_1} & \dots & 0 & \sigma_{e_2}^2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5}$$

where the residual covariance matrix has the form

$$\mathbf{V}_R = \begin{bmatrix} \sigma_{e_1}^2 & \sigma_{e_1,e_2} \\ \sigma_{e_2,e_1} & \sigma_{e_2}^2 \end{bmatrix} \tag{6}$$

This type of structure can be fitted using the `us` function: `us(trait):individual`.

If the two traits were measured on different individuals then there would be no reason to expect covariation between the residuals and an appropriate residual covariance matrix may be:

$$\mathbf{V}_R = \begin{bmatrix} \sigma_{e_1}^2 & 0 \\ 0 & \sigma_{e_2}^2 \end{bmatrix} \tag{7}$$

giving

$$\mathbf{R} = \mathbf{V}_R \otimes \mathbf{I} = \begin{bmatrix} \sigma_{e_1}^2 & 0 & \dots & 0 & 0 \\ 0 & \sigma_{e_1}^2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_{e_2}^2 & 0 \\ 0 & 0 & \dots & 0 & \sigma_{e_2}^2 \end{bmatrix} \tag{8}$$

implying independence but not identical distribution. This type of structure can be fitted using the `idh` function: `idh(trait):individual`.

Only a single variance structure can define the R-structure, but more than one random effect can be fitted by using the `+` operator, and the the resulting G-structure is formed from the direct product of each variance structure. This sounds complicated but it just means that more than one random effect can be fitted and that there is no covariance between them.

2.2 Pedigrees and Phylogenies

The variance structures above are formed through a Kronecker product involving an identity matrix, and therefore have a simple and highly patterned form. For certain types of model the identity matrix is not appropriate because covariances may be expected between different levels of the random effect. For example, if individuals are related we may expect them to be more similar if the response variable is heritable. However, if individuals are related to different degrees (e.g. some brothers, some second cousins) we don't expect them to be equally similar. Under certain assumptions we can predict the degree of similarity through the proportion of genes shared by two individuals. This can be represented using the matrix $\mathbf{A}$, and for a nuclear family consisting of two parents followed by their two children this would look like:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0.5 & 0.5 \\ 0 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \end{bmatrix} \tag{9}$$

The diagonal elements are one because an individual shares all of its genes with itself[5] and the off-diagonal elements between parents and offspring, and between siblings, are 0.5 because on average they share 50% of their genes. The diagonal element between the parents is zero because they are assumed to be unrelated. This model is known as an animal model [Henderson, 1976], and when `animal` is fitted as a random effect then $\mathbf{I}$ in Equations 4:8 is replaced with $\mathbf{A}$, where $\mathbf{A}$ is calculated from a pedigree passed to the `pedigree` argument of `MCMCglmm`. For example, the syntax `us(trait):animal` in a bivariate model would fit a G-structure of the form $\mathbf{V}_G \otimes \mathbf{A}$ where $\mathbf{V}_G$ has the genetic variances for trait 1 and trait 2 along the diagonal, and the genetic covariance between the two traits in the off-diagonals. The same model can be applied to phylogenies where a `phylo` object from the `ape` package is passed to the pedigree argument. In this case the `animal` term would not contain individual identifiers but taxa names, possibly species.

[5] I've left the interpretation a little loose; the diagonals are not necessarily one with inbreeding.

2.3 Meta-analysis

In meta-analysis each data point has some associated measurement error which may be known. If the data are published test-statistics the variance around the true value due to measurement error may be assumed to be roughly the square of the standard error. These measurement error variances can be passed to `MCMCglmm` through the argument `mev` to fit a random effect meta-analysis. The variance structure in this case will have the form $\mathbf{V}_G \otimes \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix of measurement error variances and $\mathbf{V}_G$ is a scalar fixed at one.

2.4 Random regression

Another variation on the simple mixed effect model is random regression. In standard models $\mathbf{Z}$ is a matrix of zeros and ones, with a one in row i and column j indicating that the i^{th} data point is associated with the j^{th} level of the random effect. However for random regression models, of which the random slope model is a special case, the elements of $\mathbf{Z}$ can take on alternative numeric values. For example, lets imagine individuals were weighed twice, once at 6 months (time=0.5) and once after 18 months (time=1.5). If the individuals were grouped into families we could ask a) whether individuals of certain families are generally larger at both ages and b) whether individuals of certain families grow faster between the two time periods. For two individuals from different families, $\mathbf{Z}$ would have the form:

$$\begin{bmatrix} 1 & 0 & \dots & 0.5 & 0 \\ 1 & 0 & \dots & 1.5 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0.5 \\ 0 & 1 & \dots & 0 & 1.5 \end{bmatrix} \tag{10}$$

where the first and second columns are associated with the (random) intercept's for families 1 and 2, and the third and fourth columns are associated with the (random) slopes for families 1 and 2. `leg(time, n=1):family` fits a random intercept-slope model across families for the covariate time. `n` specifies the order of the polynomial with `n=1` having intercept and slope, and `n=2` having an intercept, slope and quadratic term[6]. An unstructured (co)variance matrix is fitted, meaning that the covariance between intercept and slope is always estimated.

2.5 Priors

The prior specification is passed to `MCMCglmm` by the `prior` argument. It takes a list of 3 elements: `R`, `G` and `B`, which specify the priors for the R-structure,

[6]The design matrix $\mathbf{Z}$ is not actually 1's for the intercept and `time` for the slope because I use normailsed Legendre polynmials rather than ordinary polynomials. `legendre.polynomials(n, TRUE)` will give the appropriate scalings.

G-structure and the fixed effects. `G` is also a list with an element for each random effect. The covariance matrices are assumed to be (conditional) inverse-Wishart distributed and individual elements for each variance structure take the arguments `V`, `n` and `split` which specify the (co)variance matrix, the degree of freedom parameter, and the partition to condition on. For example, the residual error structure for a trivariate model may look something like:

```
> prior = list(R = list(V = diag(3), n = 3))
```

which is inverse Wishart distributed with inverse scale matrix `solve(n*V)` and degree of belief parameter `n`. You can get a feel for this using the `rIW` function which generates random samples from the distribution. The default prior is `n=0` and is not proper, but it is informative. The specification `V=diag(k)*0, n=k-1` where `k` is the dimension of `V` is uninformative, but I strongly recommend you use proper priors where `n` is greater than or equal to `k`. If nothing else, `MCMCglmmm` may baulk after many iterations if the parameters are not well identified. When `idh()` structures are used each diagonal element of the matrix is independently distributed *a priori*. The distribution of the diagonal element has the same marginal distribution as that element would have in the standard inverse-Wishart distribution (i.e. $\sim IW(n-k+1, (n\sigma^2)^{-1})$) such that the prior specification above would actually be equivalent to three priors on each diagonal element with `V=1` and `n=1`.

The fixed effects have a multivariate normal distribution equal in dimension to the number of fixed effects. `B` has two arguments a mean vector `mu` and a (co)variance matrix `V`. The default has a zero mean vector and a diagonal variance matrix with large variances (1e+10).

2.6 Fixing (co)variances

In meta-analysis we fix the variance component to one because we are assuming we know the measurement error variances. In some instances however, we may want to fix the variances because there is no information in the data and we cannot estimate them. We can fix certain elements of a variance structure by giving `fix` a value in the prior specification. For example, we could modify the previous prior specification:

```
> prior = list(R = list(V = diag(3), n = 3, fix = 2))
```

The fix argument partitions V into (potentially) 4 sub-matrices where the partition occurs on the `fix`th diagonal element. In this case the partition has the form

$$\mathbf{V} = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \tag{11}$$

and the lower right sub-matrix is fixed and not estimated. When `fix=1` the whole matrix is fixed. The default prior for the variance structures do not fix any component.

2.7 Starting values

Starting values are passed to `MCMCglmm` by the `start` argument. It takes a list of 4 elements: `R`, `G` and `liab` specify the starting R-structure, G-structures and the liabilities. As with the prior argument, `G` must also be a list with an element for each random effect. The starting liabilities are passed as a matrix equal in dimension to $\mathbf{L}$. By default `MCMCglmm` will use some very rough and ready methods for obtaining reasonable starting values for the liabilities, however, this can be suppressed by passing the fourth element of `start` as `QUASI=FALSE` which will then sample the starting liabilities from a normal distribution with mean zero and variance one when the `liab` element is `NULL`.

2.8 Tuning parameters

For certain types and combinations of distribution the liabilities have to be sampled using Metropolis-Hastings (MH) updates rather than Gibbs sampling. The choice of proposal distribution for the MH sampler can strongly effect the rate of convergence and the mixing properties of the chain. By default an adaptive MH sampler is used which modifies the proposal distribution so it has an efficient rate of jumping. Once the burnin period is over this proposal distribution is fixed so as to ensure that the posterior distribution is valid. Alternatively, a matrix equal in dimension to the R-structure can be passed to the tune argument, and this will serve as the covariance matrix for the proposal distribution centred on the previous value of the liabilities.

2.9 Distributions

Currently the distributions listed in Table 2.9 are those that can be used, although this list will be extended to cover some additional distributions such as the Weibull.

Multinomial and categorical models are parametrised in $k-1$ dimensions where k is the number of categories. For multinomial data that only have a single count the data can be represented by a column of factors. If they are passed in this way then the baseline category is the first factor level. I've done this so that if a binary trait is passed, the model is predicting ones not zeros as is usual. If the distribution is specified as multinomial$k$ then the data is expected to have k columns. In this instance the final column is treated as the baseline category in keeping with the usual binomial specification `cbind(successes, failures)` where the model is predicting successes (not failures).

Distribution Type	No. Data (**Y**) columns	No. liability (**L**) columns	Recommended R-structure	Recommended R-constraint
`"gaussian"`	1	1	standard	none
`"poisson"`	1	1	standard	none
`"categorical"`	1	k-1	`us(trait):units`	`fix=1, V=`$0.5(\mathbf{I}+\mathbf{J})$
`"multinomial`k`"`	k	k-1	standard	none
`"exponential"`	1	1	standard	none
`"cengaussian"`	2	1	standard	none
`"cenpoisson"`	2	1	standard	none
`"cenexponential"`	2	1	standard	none
`"zipoisson"`	1	2	idh(trait):units	`fix=2, V=as.matrix(c(a,0,0,1),2,2)`

Table 1: Distribution types that can fitted using `MCMCglmm`; their dimension and recommended residual variance structures. For multi-trait analyses there are no hard and fast rules although us(trait):units is often a good starting point for the residual variance structure. The prefix `"cen"` standards for censored, and the prefix `"zi"` stands for zero-inflated. k stands for the number of categories in the multinomial/categorical distributions and this must be specified in the family argument for the multinomial distribution. $\mathbf{I}$ is the identity matrix and $\mathbf{J}$ the unit matrix of all ones. In this context they both have dimension $k-1$

For censored responses two data columns must be passed. The first column should contain the minimum value the data could take and the second column the maximum. If these values are finite the data are said to be interval censored. For left censored data use `-Inf` in the first column and for right censored data use `Inf` in the second column. If a particular data point is not censored have the same value in both columns.

Although only a single column of data is passed for the zero-inflated Poisson, the response is actually expanded to form a bivariate model. The first `trait` is the Poisson part of the model and the residual variance can be estimated (to account for over-disperion) but the second `trait` is the zero-inflation and as with a binary model the residual variance cannot be estimated. In addition the residual (co)variance between the Poisson term and the zero-infaltion term cannot be estimated because we can't observe both processes in a single individual. Nevertheless this does not mean that these covariances cannot be estimated at the level of some random effect.

3 Acknowledgements

This work would not have been possible without the CSparse library written by Tim Davis and the comprehensive book on MCMC and mixed models by Sorensen & Gianola. I am grateful to Loeske Kruuk for providing funding for this work through the Leverhulme Trust, and Hannah Tidbury and Mike Boots for allowing me to use their data to demonstrate the package. Laura Ross did battle with the tutorial.

References

T. A. Davis. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. SIAM, Philadelphia, 2006.

C. R. Henderson. Simple method for computing inverse of a numerator relationship matrix used in prediction of breeding values. *Biometrics*, 32(1):69–83, 1976. Times Cited: 296.

S. H. Rice. Developmental associations between traits: Covariance and beyond. *Genetics*, 166(1):513–526, 2004. Genetics.

D. Sorensen and D. Gianola. *Likelihood, Bayesian and MCMC Methods in Quantitative Genetics*. Statistics for Biology and Health. Springer-Verlag, New York, 2002.