MCMCglmm Course Notes

Jarrod Hadfield (j.hadfield@ed.ac.uk)

April 17, 2019

Introduction

These are (incomplete) course notes about generalised linear mixed models (GLMM). Special emphasis is placed on understanding the underlying structure of a GLMM in order to show that slight modifications of this structure can produce a wide range of models. These include familiar models like regression and ANOVA, but also models with intimidating names: animal models, threshold models, meta-analysis, MANCOVA and random regression ... The primary aim of the course is to show that these models are only daunting by name. The secondary aim is to show how these models can be fitted in a Bayesian framework using Markov chain Monte Carlo (MCMC) methods in the R package `MCMCglmm`. For those not comfortable using Bayesian methods, many of the models outlined in the course notes can be fitted in `asreml` or `lmer` with little extra work. If you do use `MCMCglmm`, please, cite **?**.

Contents

Chapter 1

Bayesian Statistics & Markov chain Monte Carlo

There are fundamental differences between classical and Bayesian approaches, but for those of us interested in applied statistics the hope is that these differences do not translate into practical differences, and this is often the case. My advice would be *if* you can fit the same model using different packages and/or methods do so, and if they give very different answers worry. In some cases differences will exist, and it is important to know why, and which method is more appropriate for the data in hand.

In the context of a generalised linear mixed model (GLMM), here are what I see as the pro's and cons of using (restricted) maximum likelihood (REML) versus Bayesian Markov chain Monte Carlo (MCMC) Bayesian methods. REML is fast and easy to use, whereas MCMC can be slow and technically more challenging. Particularly challenging is the specification of a sensible prior, something which is a non-issue in a REML analysis. However, analytical results for non-Gaussian GLMM are generally not available, and REML based procedures use approximate likelihood methods that may not work well. MCMC is also an approximation but the accuracy of the approximation increases the longer the analysis is run for, being exact at the limit. In addition REML uses large-sample theory to derive approximate confidence intervals that may have very poor coverage, especially for variance components. Again, MCMC measures of confidence are exact, up to Monte Carlo error, and provide an easy and intuitive way of obtaining measures of confidence on derived statistics such as ratios of variances, correlations and predictions.

To illustrate the differences between the approaches lets imagine we've observed several random deviates ($\mathbf{y}$) from a standard normal (i.e. $\mu = 0$ and $\sigma^2 = 1$). The likelihood is the probability of the data given the parameters:

$$Pr(\mathbf{y}|\mu, \sigma^2)$$

This is a conditional distribution, where the conditioning is on the model parameters which are taken as fixed and known. In a way this is quite odd because we've already observed the data, and we don't know what the parameter values are. In a Bayesian analysis we evaluate the conditional probability of the model parameters given the observed data:

$$Pr(\mu, \sigma^2|\mathbf{y})$$

which seems more reasonable, until we realise that this probability is proportional to

$$Pr(\mathbf{y}|\mu, \sigma^2)Pr(\mu, \sigma^2)$$

where the first term is the likelihood, and the second term represents our prior belief in the values that the model parameters could take. Because the choice of prior is rarely justified by an objective quantification of the state of knowledge it has come under criticism, and indeed we will see later that the choice of prior can make a difference.

1.1 Likelihood

We can generate 5 observations from this distribution using `rnorm`:

```
> Ndata <- data.frame(y = rnorm(5, mean = 0, sd = sqrt(1)))
> Ndata$y

[1] -0.1388900  1.1998129 -0.7477224 -0.5752482 -0.2635815
```

We can plot the probability density function for the standard normal using `dnorm` and we can then place the 5 data on it:

```
> possible.y<-seq(-3,3,0.1)                          # possible values of y
> Probability<-dnorm(possible.y, mean=0, sd=sqrt(1))     # density of possible values
> plot(Probability~possible.y, type="l")
> Probability.y<-dnorm(Ndata$y, mean=0, sd=sqrt(1)) # density of actual values
> points(Probability.y~Ndata$y)
```

The likelihood of these data, conditioning on $\mu = 0$ and $\sigma^2 = 1$, is proportional to the product of the densities (read off the y axis on Figure 1.1):

```
> prod(dnorm(Ndata$y, mean = 0, sd = sqrt(1)))

[1] 0.003015919
```

Of course we don't know the true mean and variance and so we may want to ask how probable the data would be if, say, $\mu = 0$, and $\sigma^2 = 0.5$:

```
> prod(dnorm(Ndata$y, mean = 0, sd = sqrt(0.5)))
```

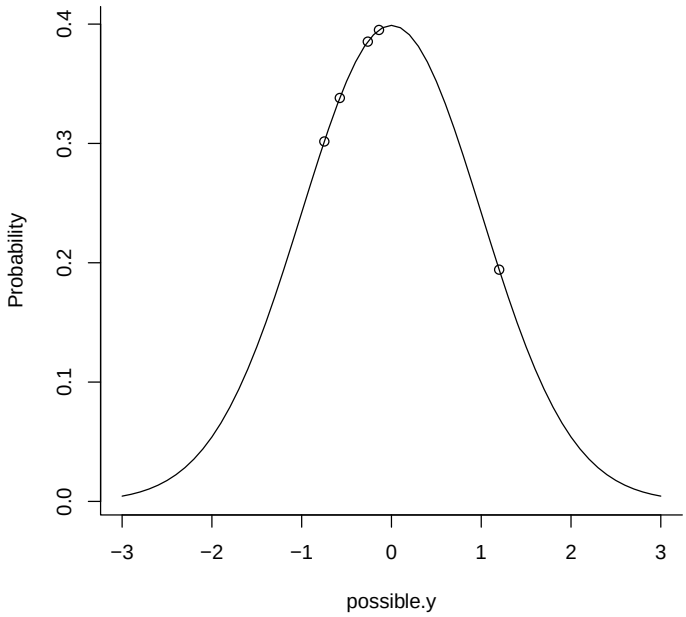

Figure 1.1: Probability density function for the unit normal with the data points overlaid.

```
[1] 0.005091715
```

It would seem that the data are more likely under this set of parameters than the true parameters, which we must expect some of the time just from random sampling. To get some idea as to why this might be the case we can overlay the two densities (Figure 1.2), and we can see that although some data points (e.g. 1.2) are more likely with the true parameters, in aggregate the new parameters produce a higher likelihood.

The likelihood of the data can be calculated on a grid of possible parameter values to produce a likelihood surface, as in Figure 1.3. The densities on the contours have been scaled so they are relative to the density of the parameter values that have the highest density (the maximum likelihood estimate of the two parameters). Two things are apparent. First, although the surface is symmetric about the line $\mu = \hat{\mu}$ (where $\hat{}$ stands for maximum likelihood estimate) the surface is far from symmetric about the line $\sigma^2 = \hat{\sigma}^2$. Second, there are a large range of parameter values for which the data are only 10 times less likely than if the data were generated under the maximum likelihood estimates.

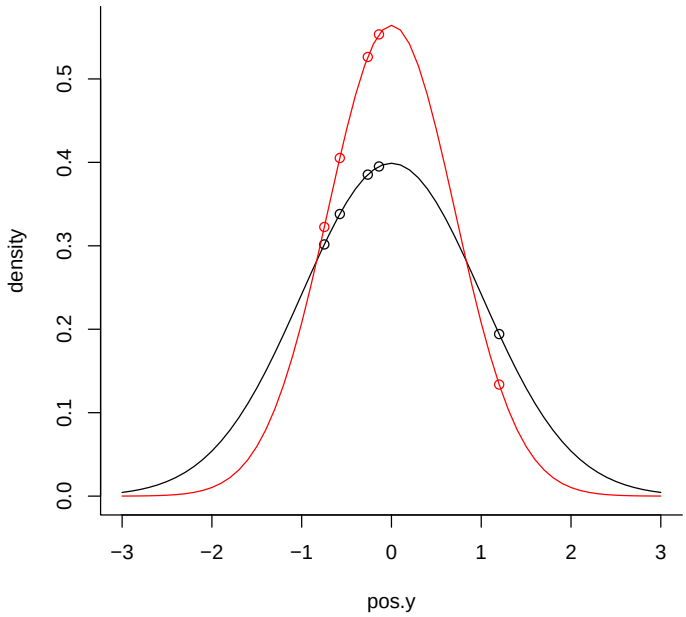

Figure 1.2: Two probability density functions for normal distributions with means of zero, and a variance of one (black line) and a variance of 0.5 (red line). The data points are overlaid.

1.1.1 Maximum Likelihood (ML)

The ML estimator is the combination of μ and σ^2 that make the data most likely. Although we could evaluate the density on a grid of parameter values (as we did to produce Figure 1.3) in order to locate the maximum, for such a simple problem the ML estimator can be derived analytically. However, so we don't have to meet some nasty maths later, I'll introduce and use one of R's generic optimising routines that can be used to maximise the likelihood function (in practice, the log-likelihood is maximised to avoid numerical problems):

```
> loglik <- function(par, y) {
+     sum(dnorm(y, par[1], sqrt(par[2]), log = TRUE))
+ }
> MLest <- optim(c(mean = 0, var = 1), fn = loglik,
+     y = Ndata$y, control = list(fnscale = -1,
+         reltol = 1e-16))$par
```

The first call to `optim` are starting values for the optimisation algorithm, and

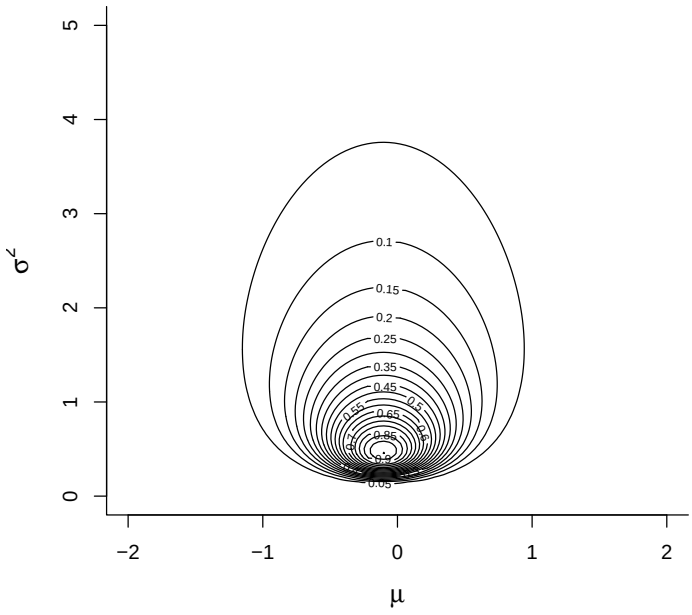

Figure 1.3: Likelihood surface for the likelihood $Pr(\mathbf{y}|\mu, \sigma^2)$. The likelihood has been normalised so that the maximum likelihood has a value of one.

the second argument (`fn`) is the function to be maximised. By default `optim` will try to minimise the function hence multiplying by -1 (`fnscale = -1`). The algorithm has successfully found the mode:

```
> MLest

      mean        var
-0.1051258  0.4726117
```

Alternatively we could also fit the model using `glm`:

```
> m1a.1 <- glm(y ~ 1, data = Ndata)
> summary(m1a.1)

Call:
glm(formula = y ~ 1, data = Ndata)

Deviance Residuals:
       1         2         3         4         5
```

```
-0.03376   1.30494  -0.64260  -0.47012  -0.15846

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1051     0.3437  -0.306    0.775

(Dispersion parameter for gaussian family taken to be 0.5907647)

    Null deviance: 2.3631  on 4  degrees of freedom
Residual deviance: 2.3631  on 4  degrees of freedom
AIC: 14.442

Number of Fisher Scoring iterations: 2
```

Here we see that although the estimate of the mean (intercept) is the same, the estimate of the variance (the dispersion parameter: 0.591) is higher when fitting the model using `glm`. In fact the ML estimate is a factor of $\frac{n}{n-1}$ smaller.

```
> MLest["var"] * (5/4)

      var
0.5907647
```

1.1.2 Restricted Maximum Likelihood (REML)

To see why this happens, imagine if we had only observed the first two values of y (Figure 1.4). The variance is defined as the average squared distance between a random variable and the *true* mean. However, the ML estimator of the variance is the average squared distance between a random variable and the ML *estimate* of the mean. Since the ML estimator of the mean is the average of the two numbers (the dashed line) then the average squared distance will always be smaller than if the true mean was used, unless the ML estimate of the mean and the true mean coincide. This is why we divide by $n - 1$ when estimating the variance from the sum of squares, and is the motivation behind REML.

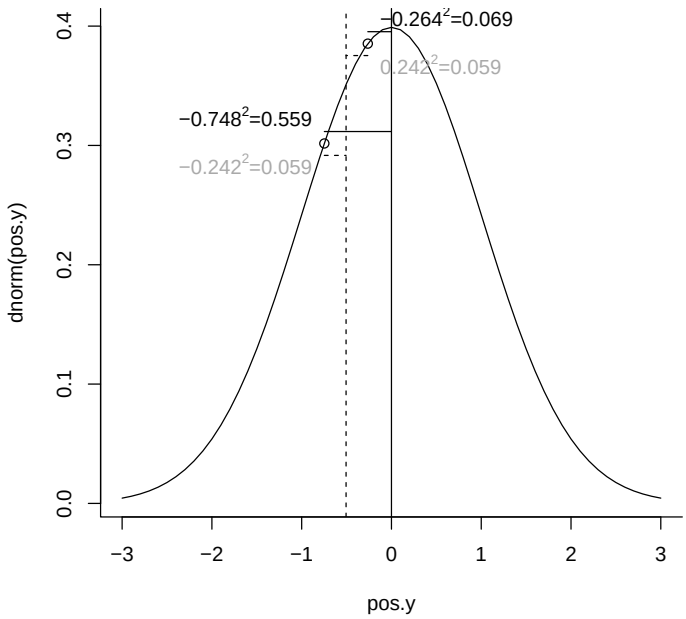

Figure 1.4: Probability density function for the unit normal with 2 realisations overlaid. The solid vertical line is the true mean, whereas the vertical dashed line is the mean of the two realisations (the ML estimator of the mean). The variance is the expected squared distance between the true mean and the realisations. The ML estimator of the variance is the average squared distance between the ML mean and the realisations (horizontal dashed lines), which is always smaller than the average squared distance between the true mean and the realisations (horizontal solid lines)

1.2 Prior Distribution

`MCMCglmm` uses an inverse Wishart prior for the (co)variances and a normal prior for the fixed effects. In versions > 1.13 parameter expanded models can be used which enable prior specifications from the the scaled non-central F-distribution (**?**). Here, we will focus on specifying a prior for a single fixed effect (μ) and a single variance component using the inverse-Wishart to highlight some of the issues. I strongly recommend reading the section 8.0.2 on parameter expanded priors as these can be less informative than the inverse-Wishart under many situations.

For a single variance component the inverse Wishart takes two scalar parameters, `V` and `nu`. The distribution tends to a point mass on `V` as the degree of belief parameter, `nu` goes to infinity. The distribution tends to be right skewed when `nu` is not very large, with a mode of $\frac{\texttt{V}^*\texttt{nu}}{\texttt{nu}+2}$ but a mean of $\frac{\texttt{V}^*\texttt{nu}}{\texttt{nu}-2}$ (which is not defined for $\texttt{nu} < 2$).[1]

As before, we can evaluate and plot density functions in order to visualise what the distribution looks like. Figure 1.5 plots the probability density functions holding `V` equal to one but with `nu` varying.

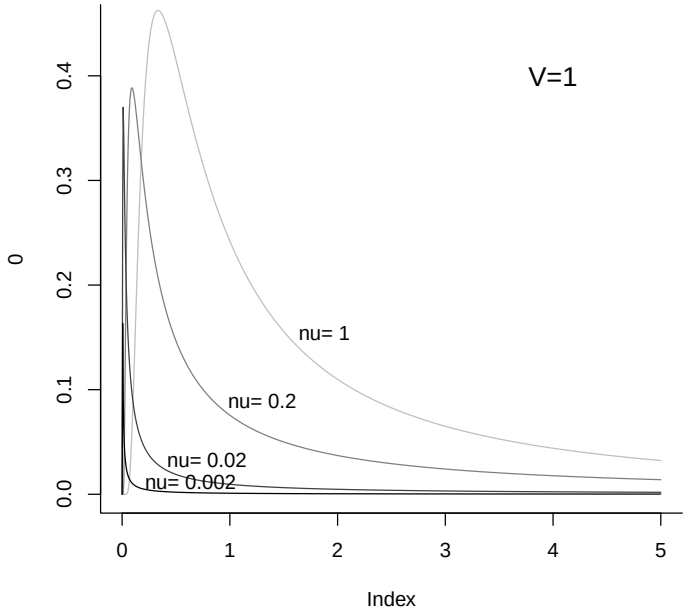

Figure 1.5: Probability density function for a univariate inverse Wishart with the variance at the limit set to 1 (`V=1`) and varying degree of belief parameter (`nu`). With `V=1` these distributions are equivalent to inverse gamma distributions with shape and scale parameters set to `nu`/2.

A probability distribution must integrate to one because a variable must

[1] The inverse gamma is a special case of the inverse Wishart, although it is parametrised using `shape` and `scale`, where $\texttt{nu} = 2 * \texttt{shape}$ and $\texttt{V} = \frac{\texttt{scale}}{\texttt{shape}}$ (or $\texttt{shape} = \frac{\texttt{nu}}{2}$ and $\texttt{scale} = \frac{\texttt{nu}*\texttt{V}}{2}$). `MCMCpack` provides a density function (`dinvgamma`) for the inverse gamma distribution.

have some value. It therefore seems reasonable that when specifying a prior, care must be taken that this condition is met. In the example here where `V` is a single variance this condition is met if `V>0` and `nu>0`. If this condition is not met then the prior is said to be improper, and in WinBUGS (and possibly other software) improper priors cannot be specified. Although great care has to be taken when using improper priors, `MCMCglmm` does allow them as they have some useful properties, and some common improper priors are discussed in section 1.5. However, for now we will use the prior specification `V=1` and `nu=0.002` which is frequently used for variance components. For the mean we will use a diffuse normal prior centred around zero but with very large variance (10^8). If the variance is finite then the prior is always proper.

As before we can write a function for calculating the (log) prior probability:

```
> logprior <- function(par, priorR, priorB) {
+     dnorm(par[1], mean = priorB$mu, sd = sqrt(priorB$V),
+         log = TRUE) + log(dinvgamma(par[2], shape = priorR$nu/2,
+         scale = (priorR$nu * priorR$V)/2))
+ }
```

where `priorR` is a list with elements `V` and `nu` specifying the prior for the variance, and `priorB` is a list with elements `mu` and `V` specifying the prior for the mean. `MCMCglmm` takes these prior specifications as a list:

```
> prior <- list(R = list(V = 1, nu = 0.002), B = list(mu = 0,
+     V = 1e+08))
```

1.3 Posterior Distribution

To obtain a posterior density we need to multiply the likelihood by the prior probability for that set of parameters. We can write a function for doing this:

```
> loglikprior <- function(par, y, priorR, priorB) {
+     loglik(par, y) + logprior(par, priorR, priorB)
+ }
```

and we can overlay the posterior densities on the likelihood surface we calculated before (Figure 1.3).

The prior has some influence on the posterior mode of the variance, and we can use an optimisation algorithm again to locate the mode:

```
> Best <- optim(c(mean = 0, var = 1), fn = loglikprior,
+     y = Ndata$y, priorR = prior$R, priorB = prior$B,
+     method = "L-BFGS-B", lower = c(-1e+05, 1e-05),
+     upper = c(1e+05, 1e+05), control = list(fnscale = -1,
+         factr = 1e-16))$par
> Best
```

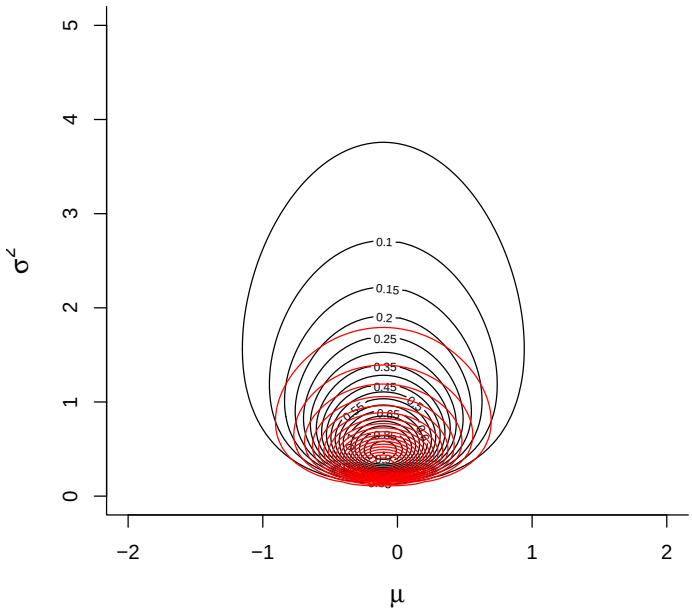

Figure 1.6: Likelihood surface for the likelihood $Pr(\mathbf{y}|\mu, \sigma^2)$ in black, and the posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$ in red. The likelihood has been normalised so that the maximum likelihood has a value of one, and the posterior distribution has been normalised so that the posterior mode has a value of one. The prior distributions $Pr(\mu) \sim N(0, 10^8)$ and $Pr(\sigma^2) \sim IW(\mathtt{V} = 1, \mathtt{nu} = 0.002)$ were used.

```
      mean        var
-0.1051258  0.3377710
```

The posterior mode for the mean is identical to the ML estimate, but the posterior mode for the variance is even less than the ML estimate which is known to be downwardly biased. The reason that the ML estimate is downwardly biased is because it did no take into account the uncertainty in the mean. In a Bayesian analysis we can do this by evaluating the marginal distribution of σ^2 and averaging over the uncertainty in the mean.

1.3.1 Marginal Posterior Distribution

The marginal distribution is often of primary interest in statistical inference, because it represents our knowledge about a parameter given the data:

$$Pr(\sigma^2|\mathbf{y}) \propto \int Pr(\mu, \sigma^2|\mathbf{y}) d\mu$$

after averaging over any nuisance parameters, such as the mean in this case.

Obtaining the marginal distribution analytically is usually impossible, and this is where MCMC approaches prove useful. We can fit this model in `MCMCglmm` pretty much in the same way as we did using `glm`:

```
> m1a.2 <- MCMCglmm(y ~ 1, data = Ndata, prior = prior,
+     thin = 1, verbose = FALSE)
```

The Markov chain is drawing random (but often correlated) samples from the joint posterior distribution (depicted by the red contours in Figure 1.6). The element of the output called `Sol` contains the distribution for the mean, and the element called `VCV` contains the distribution for the variance. We can produce a scatter plot:

```
> points(cbind(m1a.2$Sol, m1a.2$VCV))
```

and we see that MCMCglmm is sampling the same distribution as the posterior distribution calculated on a grid of possible parameter values (Figure 1.8).

A very nice property of MCMC is that we can normalise the density so that it integrates to 1 (a true probability) rather than normalising it with respect to some other aspect of the distribution, such as the density at the ML estimator or the joint posterior mode as in Figures 1.3 and 1.6. To make this clearer, imagine we wanted to know how much more probable the unit normal (i.e. with $\mu = 0$ and $\sigma^2 = 1$) was than a normal distribution with the posterior modal parameters. We can calculate this by taking the ratio of the posterior densities at these two points:

```
> exp(loglikprior(Best, Ndata$y, prior$R, prior$B) -
+     loglikprior(c(0, 1), Ndata$y, prior$R, prior$B))
```

```
[1] 4.522744
```

Now, if we wanted to know the probability that the parameters lay in the region of parameter space we were plotting, i.e. lay in the square $\mu = (-2, 2)$ and $\sigma^2 = (0, 5)$ then this would be more difficult. We would have to evaluate the density at a much larger range of parameter values than we had done, ensuring that we had covered all regions with positive probability. Because MCMC has sampled the distribution randomly, this probability will be equal to the expected probability that we have drawn an MCMC sample from the region. We can obtain an estimate of this by seeing what proportion of our actual samples lie in this square:

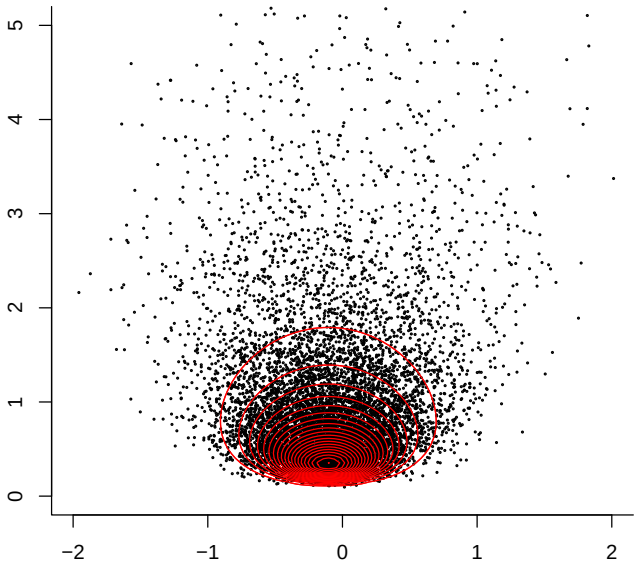

Figure 1.7: The posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$. The black dots are samples from the posterior using MCMC, and the red contours are calculated by evaluating the posterior density on a grid of parameter values. The contours are normalised so that the posterior mode has a value of one.

```
> prop.table(table(m1a.2$Sol > -2 & m1a.2$Sol <
+     2 & m1a.2$VCV < 5))

 FALSE   TRUE
0.0225 0.9775
```

There is Monte Carlo error in the answer (0.978) but if we collect a large number of samples then this can be minimised.

Using a similar logic we can obtain the marginal distribution of the variance by simply evaluating the draws in `VCV` ignoring (averaging over) the draws in `Sol`:

```
> hist(m1a.2$VCV[which(m1a.2$VCV < 5)])
> abline(v = Best["var"], col = "red")
```

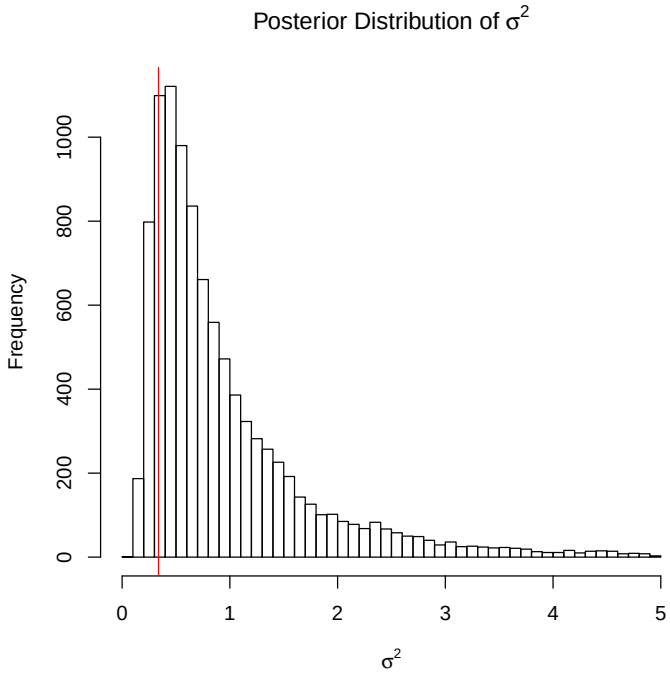

Figure 1.8: Histogram of samples from the marginal distribution of the variance $Pr(\sigma^2|\mathbf{y})$ using MCMC. The vertical line is the joint posterior mode, which differs slightly from the marginal posterior mode (the peak of the marginal distribution).

In this example (see Figure 1.8) the marginal mode and the joint mode are very similar, although this is not necessarily the case and can depend both on the data and the prior. Section 1.5 introduces improper priors that are non-informative with regard to the marginal distribution of a variance.

1.4 MCMC

In order to be confident that `MCMCglmm` has successfully sampled the posterior distribution it will be necessary to have a basic understanding of MCMC methods. MCMC methods are often used when the joint posterior distribution cannot be derived analytically, which is nearly always the case. MCMC relies on the fact that although we cannot derive the complete posterior, we can calculate the height of the posterior distribution at a particular set of parameter

values, as we did to obtain the contour plot in Figure 1.6. However, rather than going systematically through every likely combination of μ and σ and calculate the height of the distribution at regular distances, MCMC moves stochastically through parameter space, hence the name 'Monte Carlo'.

1.4.1 Starting values

First we need to initialise the chain and specify a set of parameter values from which the chain can start moving through parameter space. Ideally we would like to pick a region of high probability, as we do not want to waste time wandering through regions of low probability: we are not so interested in determining the height of the distribution far outside of Figure 1.6 as it is virtually flat and close to zero (or at least we hope so!). Although starting configurations can be set by the user using the `start` argument, in general the heuristic techniques used by `MCMCglmm` seem to work quite well. We will denote the parameter values of the starting configuration (time $t = 0$) as $\mu_{t=0}$ and ${\sigma^2}_{t=0}$. There are several ways in which we can get the chain to move in parameter space, and `MCMCglmm` uses a combination of Gibbs sampling, slice sampling and Metropolis-Hastings updates. To illustrate, it will be easier to turn the contour plot of the posterior distribution into a perspective plot (Figure 1.9).

1.4.2 Metrpolis-Hastings updates

After initialising the chain we need to decide where to go next, and this decision is based on two rules. First we have to generate a candidate destination, and then we need to decide whether to go there or stay where we are. There are many ways in which we could generate candidate parameter values, and `MCMCglmm` uses a well tested and simple method. A random set of coordinates are picked from a multivariate normal distribution that is entered on the initial coordinates $\mu_{t=0}$ and $\sigma^2_{t=0}$. We will denote this new set of parameter values as μ_{new} and σ^2_{new}. The question then remains whether to move to this new set of parameter values or remain at our current parameter values now designated as old $\mu_{old} = \mu_{t=0}$ and $\sigma^2_{old} = \sigma^2_{t=0}$. If the posterior probability for the new set of parameter values is greater, then the chain moves to this new set of parameters and the chain has successfully completed an iteration: ($\mu_{t=1} = \mu_{new}$ and $\sigma^2_{t=1} = \sigma^2_{new}$). If the new set of parameter values has a lower posterior probability then the chain may move there, but not all the time. The probability that the chain moves to low lying areas, is determined by the relative difference between the old and new posterior probabilities. If the posterior probability for μ_{new} and σ^2_{new} is 5 times less than the posterior probability for μ_{old} and σ^2_{old}, then the chain would move to the new set of parameter values 1 in 5 times. If the move is successful then we set $\mu_{t=1} = \mu_{new}$ and $\sigma^2_{t=1} = \sigma^2_{new}$ as before, and if the move is unsuccessful then the chain stays where it is ($\mu_{t=1} = \mu_{old}$ and $\sigma^2_{t=1} = \sigma^2_{old}$). Using these rules we can record where the chain has travelled and generate an approximation of the posterior distribution. Basically, a histogram

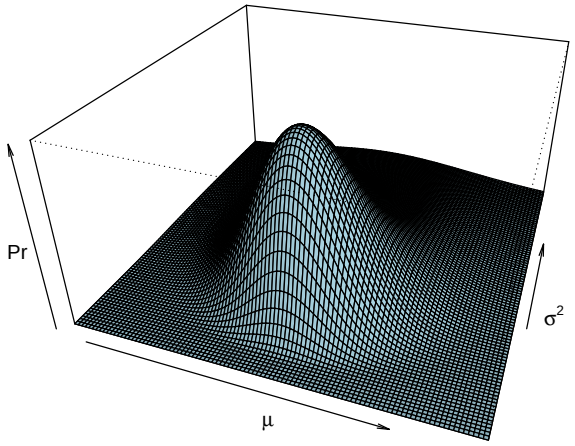

Figure 1.9: The posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$. This perspective plot is equivalent to the contour plot in Figure 1.6

of Figure 1.9.

1.4.3 Gibbs Sampling

Gibbs sampling is a special case of Metropolis-Hastings updating, and `MCMCglmm` uses Gibbs sampling to update most parameters. In the Metropolis-Hastings example above, the Markov Chain was allowed to move in both directions of parameter space simultaneously. An equally valid approach would have been to set up two Metropolis-Hastings schemes where the chain was first allowed to move along the μ axis, and then along the σ^2 axis. In Figure 1.10 I have cut the posterior distribution of Figure 1.9 in half, and the edge of the surface facing left is the conditional distribution of μ given that $\sigma^2 = 1$:

$$Pr(\mu|\sigma^2 = 1, \boldsymbol{y}). \tag{1.1}$$

In some cases, the equation that describes this conditional distribution can be derived despite the equation for the complete joint distribution of Figure

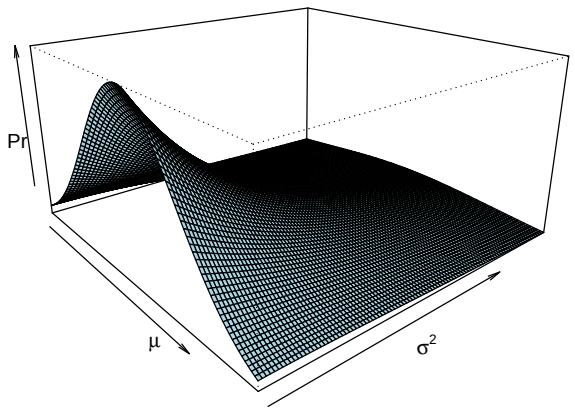

Figure 1.10: The posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$, but only for values of σ^2 between 1 and 5, rather than 0 to 5 (Figure 1.9). The edge of the surface facing left is the conditional distribution of the mean when $\sigma^2 = 1$ ($Pr(\mu|\mathbf{y}, \sigma^2 = 1)$). This conditional distribution follows a normal distribution.

1.9 remaining unknown. When the conditional distribution of μ is known we can use Gibbs sampling. Lets say the chain at a particular iteration is located at $\sigma^2 = 1$. If we updated μ using a Metropolis-Hastings algorithm we would generate a candidate value and evaluate its relative probability compared to the old value. This procedure would take place in the slice of posterior facing left in Figure 1.10. However, because we know the actual equation for this slice we can just generate a new value of μ directly. This is Gibbs sampling. The slice of the posterior that we can see in Figure 1.10 actually has a normal distribution. Because of the weak prior this normal distribution has a mean close to the mean of $\mathbf{y}$ and a variance close to $\frac{\sigma^2}{n} = \frac{1}{n}$. Gibbs sampling can be much more efficient than Metropolis-Hastings updates, especially when high dimensional conditional distributions are known, as is typical in GLMMs. A technical description of the sampling schemes used by `MCMCglmm` is given in appendix 7.2, but is perhaps not important to know.

1.4.4 Slice Sampling

If the distribution can be factored such that one factor is a distribution from which truncated random variables can be drawn, then the slice sampling methods of `?` can be used. The latent variables in univariate binary models can be updated in this way if `slice=TRUE` is specified in the call to `MCMCglmm`. In these models, slice sampling is only marginally more efficient than adaptive Metropolis-Hastings updates when the residual variance is fixed. However, for parameter expanded binary models where the residual variance is not fixed, the slice sampler can be much more efficient.

1.4.5 MCMC Diagnostics

When fitting a model using `MCMCglmm` the parameter values through which the Markov chain has travelled are stored and returned. The length of the chain (the number of iterations) can be specified using the `nitt` argument[2] (the default is 13,000), and should be long enough so that the posterior approximation is valid. If we had known the joint posterior distribution in Figure 1.9 we could have set up a Markov chain that sampled directly from the posterior. If this had been the case, each successive value in the Markov chain would be independent of the previous value after conditioning on the data, **y**, and a thousand iterations of the chain would have produced a histogram that resembled Figure 1.9 very closely. However, generally we do not know the joint posterior distribution of the parameters, and for this reason the parameter values of the Markov chain at successive iterations are usually not independent and care needs to be taken regarding the validity of the approximation. `MCMCglmm` returns the Markov chain as `mcmc` objects, which can be analysed using the `coda` package. The function `autocorr` estimates the level of non-independence between successive samples in the chain:

```
> autocorr(m1a.2$Sol)

, , (Intercept)

         (Intercept)
Lag 0   1.0000000000
Lag 1  -0.0157652146
Lag 5   0.0094886774
Lag 10  0.0093923394
Lag 50  0.0002389178

> autocorr(m1a.2$VCV)

, , units

              units
```

[2] The double `t` is because I cannot spell.

```
Lag 0   1.000000000
Lag 1   0.175580402
Lag 5  -0.007972959
Lag 10 -0.011741307
Lag 50  0.003373268
```

The correlation between successive samples is low for the mean (-0.016) but a bit high for the variance (0.176). When auto-correlation is high the chain needs to be run for longer, and this can lead to storage problems for high dimensional problems. The argument `thin` can be passed to `MCMCglmm` specifying the intervals at which the Markov chain is stored. In model `m1a.2` we specified `thin=1` meaning we stored every iteration (the default is `thin=10`). I usually aim to store 1,000-2,000 iterations and have the autocorrelation between successive *stored* iterations less than 0.1.

The approximation obtained from the Markov chain is conditional on the set of parameter values that were used to initialise the chain. In many cases the first iterations show a strong dependence on the starting parametrisation, but as the chain progresses this dependence may be lost. As the dependence on the starting parametrisation diminishes the chain is said to converge and the argument `burnin` can be passed to `MCMCped` specifying the number of iterations which must pass before samples are stored. The default burn-in period is 3,000 iterations. Assessing convergence of the chain is notoriously difficult, but visual inspection and diagnostic tools such as `gelman.diag` often suffice.

```
> plot(m1a.2$Sol)
```

On the left of Figure 1.11 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). If the model has converged there should be no trend in the time series. The equivalent plot for the variance is a little hard to see on the original scale, but on the log scale the chain looks good (Figure 1.12):

```
> plot(log(m1a.2$VCV))
```

1.5 Improper Priors

When improper priors are used their are two potential problems that may be encountered. The first is that if the data do not contain enough information the posterior distribution itself may be improper, and any results obtained from `MCMCglmm` will be meaningless. In addition, with proper priors there is a zero probability of a variance component being exactly zero but this is not necessarily the case with improper priors. This can produce numerical problems (trying to divide through by zero) and can also result in a reducible chain. A reducible

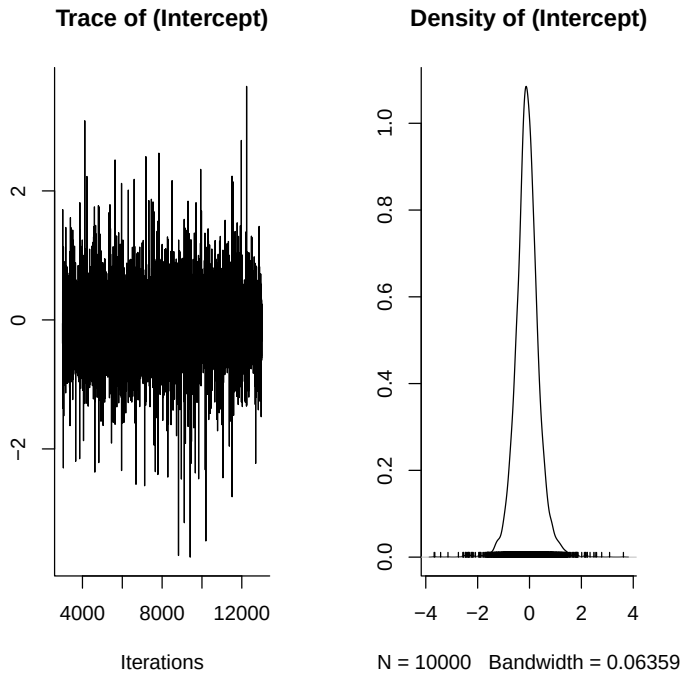

Figure 1.11: Summary plot of the Markov Chain for the intercept. The left plot is a trace of the sampled posterior, and can be thought of as a time series. The right plot is a density estimate, and can be thought of a smoothed histogram approximating the posterior.

chain is one which gets 'stuck' at some parameter value and cannot escape. This is usually obvious from the `mcmc` plots but `MCMCglmm` will often terminate before the analysis has finished with an error message of the form:

```
ill-conditioned G/R structure: use proper priors ...
```

However, improper priors do have some useful properties.

1.5.1 Flat Improper Prior

The simplest improper prior is one that is proportional to some constant for all possible parameter values. This is known as a flat prior and the posterior density in such cases is equal to the likelihood:

$$Pr(\mu, \sigma^2|\mathbf{y}) \propto Pr(\mathbf{y}|\mu, \sigma^2)$$

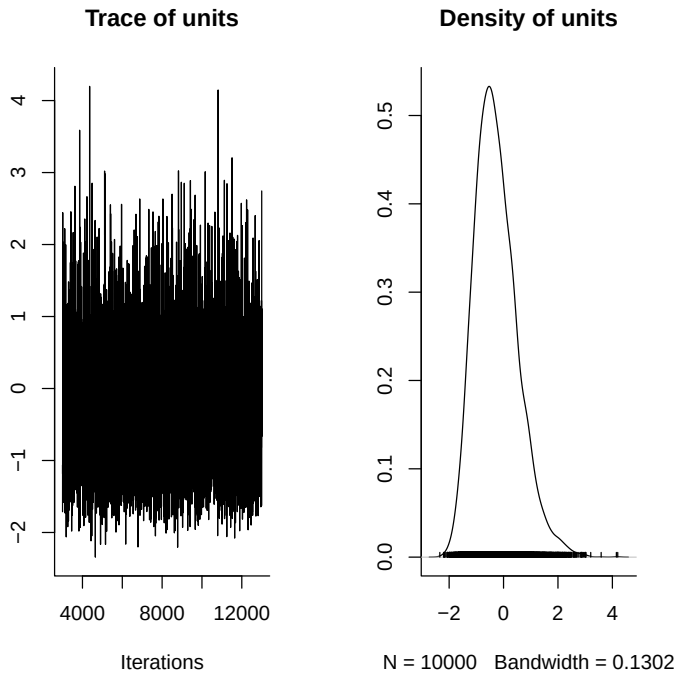

Figure 1.12: Summary plot of the Markov Chain for the logged variance. The logged variance was plotted rather than the variance because it was easier to visualise. The left plot is a trace of the sampled posterior, and can be thought of as a time series. The right plot is a density estimate, and can be thought of a smoothed histogram approximating the posterior.

It is known that although such a prior is non-informative for the mean it is informative for the variance. We can specify a flat prior on the variance component by having `nu=0` (the value of `V` is irrelevant) and the default prior for the mean is so diffuse as to be essentially flat across the range $(-10^6, 10^6)$.

```
> prior.m1a.3 <- list(R = list(V = 1, nu = 0))
> m1a.3 <- MCMCglmm(y ~ 1, data = Ndata, thin = 1,
+     prior = prior.m1a.3, verbose = FALSE)
```

We can overlay the joint posterior distribution on the likelihood surface (1.13) and see that the two things are in close agreement, up to Monte Carlo error.

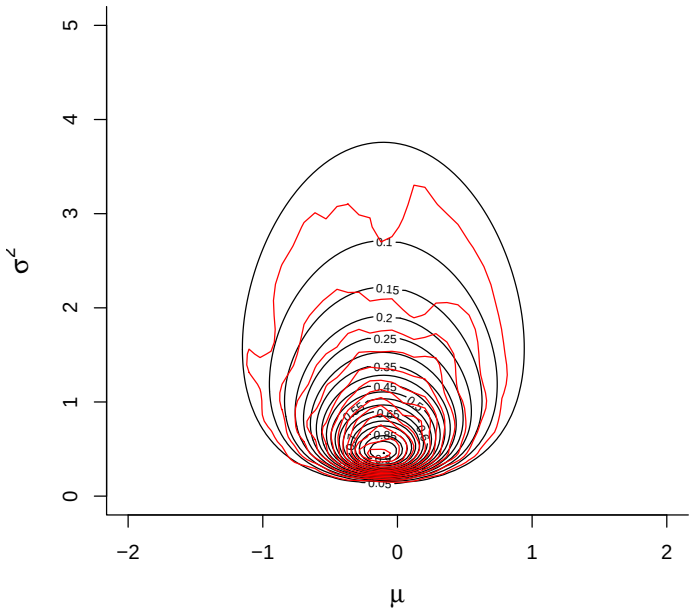

Figure 1.13: Likelihood surface for the likelihood $Pr(\mathbf{y}|\mu, \sigma^2)$ in black, and an MCMC approximation for the posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$ in red. The likelihood has been normalised so that the maximum likelihood has a value of one, and the posterior distribution has been normalised so that the posterior mode has a value of one. Flat priors were used ($Pr(\mu) \sim N(0, 10^8)$ and $Pr(\sigma^2) \sim IW(\mathtt{V} = 0, \mathtt{nu} = 0)$) and so the posterior distribution is equivalent to the likelihood.

1.5.2 Non-Informative Improper Prior

Although inverse-Wishart distributions with negative degree of belief parameters are not defined, the resulting posterior distribution can be defined if there is sufficient replication. Specifying `V=0` and `n=-1` is equivalent to a uniform prior for the standard deviation on the the interval $(0, \infty]$, and specifying `V=0` and `n=-2` is non-informative for a variance component.

```
> prior.m1a.4 <- list(R = list(V = 1e-16, nu = -2))
> m1a.4 <- MCMCglmm(y ~ 1, data = Ndata, thin = 1,
+     prior = prior.m1a.4, verbose = FALSE)
```

The joint posterior mode does not coincide with either the ML or REML estimator (Figure 1.14).

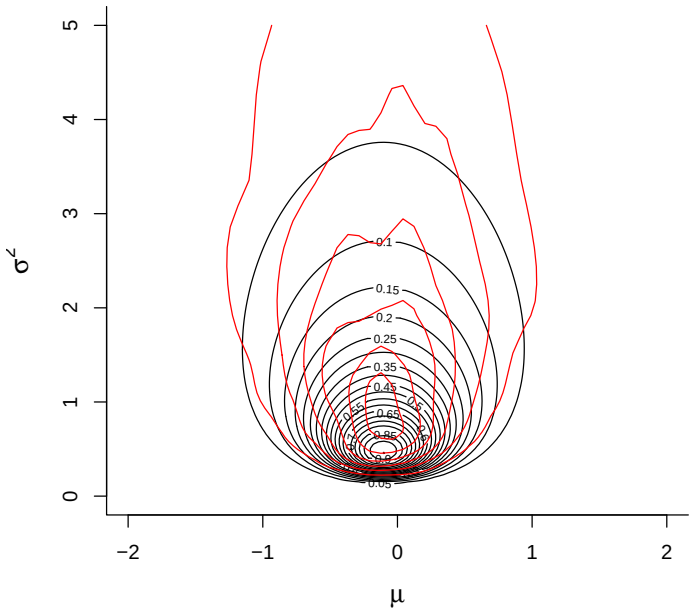

Figure 1.14: Likelihood surface for the likelihood $Pr(\mathbf{y}|\mu, \sigma^2)$ in black, and an MCMC approximation for the posterior distribution $Pr(\mu, \sigma^2|\mathbf{y})$ in red. The likelihood has been normalised so that the maximum likelihood has a value of one, and the posterior distribution has been normalised so that the posterior mode has a value of one. A non-informative prior was used ($Pr(\mu) \sim N(0, 10^8)$ and $Pr(\sigma^2) \sim IW(\mathtt{V} = 0, \mathtt{nu} = -2)$)

but the marginal distribution of the variance component is equivalent to the REML estimator (See Figure 1.15):

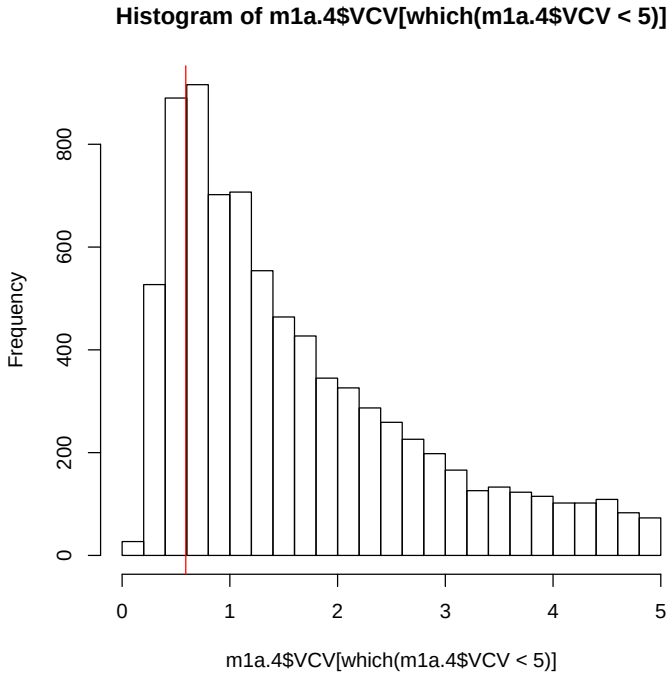

Figure 1.15: An MCMC approximation for the marginal posterior distribution of the variance $Pr(\sigma^2|\mathbf{y})$. A non-informative prior specification was used ($Pr(\mu) \sim N(0, 10^8)$ and $Pr(\sigma^2) \sim IW(\mathtt{V} = 0, \mathtt{nu} = -2)$) and the REML estimator of the variance (red line) coincides with the marginal posterior mode.

Chapter 2

Generalised Linear Mixed Models (GLMM)

2.1 Linear Model (LM)

A linear model is one in which unknown parameters are multiplied by observed variables and then added together to give a prediction for the response variable. As an example, lets take the results from a Swedish experiment from the sixties:

```
> data(Traffic, package = "MASS")
> Traffic$year <- as.factor(Traffic$year)
> Traffic[c(1, 2, 184), ]
```

```
    year day limit  y
1   1961   1    no  9
2   1961   2    no 11
184 1962  92   yes  9
```

The experiment involved enforcing speed limits on Swedish roads on some days, but on other days letting everyone drive as fast as they liked. The response variable (`y`) was how many of their citizens were injured in road accidents! The experiment was conducted in 1961 and 1962 for 92 days in each year. As a first attempt we could specify the linear model:

```
y ~ limit + year + day
```

but what does this mean?

2.1.1 Linear Predictors

The model formula defines a set of simultaneous (linear) equations

$$
\begin{aligned}
E[y\texttt{[1]}] &= \beta_1 + \beta_2(\texttt{limit[1]=="yes"}) + \beta_3(\texttt{year[1]=="1962"}) + \beta_4\texttt{day[1]} \\
E[y\texttt{[2]}] &= \beta_1 + \beta_2(\texttt{limit[2]=="yes"}) + \beta_3(\texttt{year[2]=="1962"}) + \beta_4\texttt{day[2]} \\
\vdots &= \vdots \\
E[y\texttt{[184]}] &= \beta_1 + \beta_2(\texttt{limit[184]=="yes"}) + \beta_3(\texttt{year[184]=="1962"}) + \beta_4\texttt{day[184]}
\end{aligned}
$$

where the β's are the unknown coefficients to be estimated, and the variables in `this font` are observed predictors. Continuous predictors such as `day` remain unchanged, but categorical predictors are expanded into a series of binary variables of the form '*do the data come from 1961, yes or no?*', '*do the data come from 1962, yes or no?*', and so on for as many years for which there are data.

It is cumbersome to write out the equation for each data point in this way, and a more compact way of representing the system of equations is

$$E[\mathbf{y}] = \mathbf{X}\boldsymbol{\beta} \tag{2.1}$$

where $\mathbf{X}$ is called a design matrix and contains the predictor information, and $\boldsymbol{\beta} = [\beta_1\ \beta_2\ \beta_3\ \beta_4]'$ is the vector of parameters.

```
> X <- model.matrix(y ~ limit + year + day, data = Traffic)
> X[c(1, 2, 184), ]

    (Intercept) limityes year1962 day
1             1        0        0   1
2             1        0        0   2
184           1        1        1  92
```

The binary predictors *do the data come from 1961, yes or no?* and *there was no speed limit, yes or no?* do not appear. These are the first factor levels of `year` and `limit` respectively, and are absorbed into the global intercept (β_1) which is fitted by default in R. Hence the expected number of injuries for the four combinations (on day zero) are β_1 for 1961 with no speed limit, $\beta_1 + \beta_2$ for 1961 with a speed limit, $\beta_1 + \beta_3$ for 1962 with no speed limit and $\beta_1 + \beta_2 + \beta_3$ for 1962 with a speed limit.

The simultaneous equations defined by Equation (2.1) cannot be solved directly because we do not know the expected value of y. We only know the observed value, which we assume is distributed around the expected value with some error. In a normal linear model we assume that these errors are normally distributed so that the data are also normally distributed (after taking into account the predictor variables):

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma_e^2\mathbf{I}) \tag{2.2}$$

I is an identity matrix. It has ones along the diagonal, and zeros in the off-diagonals. The zero off-diagonals imply that the residuals are uncorrelated, and the ones along the diagonal imply that they have the same variance σ_e^2. We could use `glm` to estimate β and σ_e^2 assuming that y is normally distributed:

```
> m2a.1 <- glm(y ~ limit + year + day, data = Traffic)
```

but the injuries are count data and the residuals show the typical right skew:

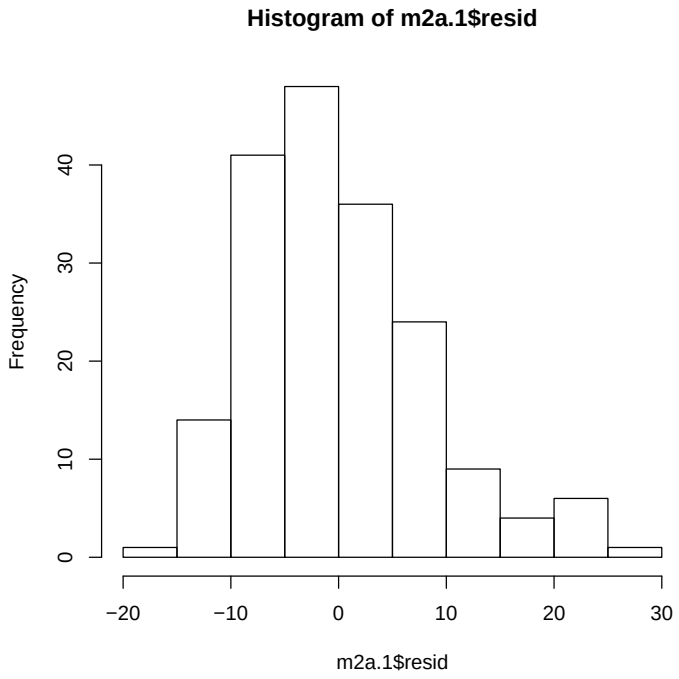

Figure 2.1: Histogram of residuals from model `m2a.1` which assumed they followed a Gaussian distribution.

Its not extreme, and the conclusions probably won't change, but we could assume that the data follow some other distribution.

2.2 Generalised Linear Model (GLM)

Generalised linear models extend the linear model to non-Gaussian data. They are essentially the same as the linear model described above, except they differ in two aspects. First, it is not necessarily the mean response that is predicted, but

some function of the mean response. This function is called the link function. For example, with a log link we are trying to predict the logged expectation:

$$\log(E[\mathbf{y}]) = \mathbf{X}\boldsymbol{\beta} \tag{2.3}$$

or alternatively

$$E[\mathbf{y}] = \exp(\mathbf{X}\boldsymbol{\beta}) \tag{2.4}$$

where exp is the inverse of the log link function. The second difference is that many distributions are single parameter distributions for which a variance does not need to be estimated, because it can be inferred from the mean. For example, we could assume that the number of injuries are Poisson distributed, in which case we also make the assumption that the variance is equal to the expected value. There are many different types of distribution and link functions and those supported by `MCMCglmm` can be found in Table 7.1. For now we will concentrate on a Poisson GLM with log link (the default link function for the Poisson distribution):

```
> m2a.2 <- glm(y ~ limit + year + day, family = poisson,
+     data = Traffic)
> summary(m2a.2)

Call:
glm(formula = y ~ limit + year + day, family = poisson, data = Traffic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.0467406  0.0372985  81.685  < 2e-16 ***
limityes    -0.1749337  0.0355784  -4.917 8.79e-07 ***
year1962    -0.0605503  0.0334364  -1.811   0.0702 .
day          0.0024164  0.0005964   4.052 5.09e-05 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: 1467.2

Number of Fisher Scoring iterations: 4
```

The results look fairly straightforward, having a speed limit reduces the number of injuries significantly, there are fewer injuries in 1962 (although significance is marginal) and there is a significant increase in the number of injuries over the year. Are these big effects or small effects? The coefficients are on the log scale so to get back to the data scale we need to exponentiate. The exponent of the intercept is the predicted number of injuries on day zero in 1961 without a speed limit:

```
> exp(m2a.2$coef["(Intercept)"])

(Intercept)
   21.04663
```

To get the prediction for the same day with a speed limit we need to add the `limityes` coefficient

```
> exp(m2a.2$coef["(Intercept)"] + m2a.2$coef["limityes"])

(Intercept)
   17.66892
```

With a speed limit there are expected to be 0.840 times less injuries than if there were no speed limits. This value can be more directly obtained:

```
> exp(m2a.2$coef["limityes"])

 limityes
0.8395127
```

and holds true for any given day in either year. For example, without a speed limit on the final day of the year (92) in 1961 we expect 24.742 injuries:

```
> exp(m2a.2$coef["(Intercept)"] + m2a.2$coef["year1962"] +
+     92 * m2a.2$coef["day"])

(Intercept)
   24.74191
```

and 20.771 injuries if a speed limit had been in place:

```
> exp(m2a.2$coef["(Intercept)"] + m2a.2$coef["limityes"] +
+     m2a.2$coef["year1962"] + 92 * m2a.2$coef["day"])

(Intercept)
   20.77115
```

The proportional change is identical because the model is *linear* on the log scale.

2.3 Over-dispersion

Most count data do not conform to a Poisson distribution because the variance in the response exceeds the expectation. This is known as over-dispersion and it is easy to see how it arises, and why it is so common. In the summary to `m2a.2` note that the ratio of the residual deviance to the residual degrees of freedom is 3.162 which means, roughly speaking, there is 3.2 times as much variation in the residuals than what we expect.

If the predictor data had not been available to us then the only model we could have fitted was one with just an intercept:

```
> m2a.3 <- glm(y ~ 1, data = Traffic, family = "poisson")
> summary(m2a.3)

Call:
glm(formula = y ~ 1, family = "poisson", data = Traffic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6546  -1.4932  -0.3378   0.9284   5.0601

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.07033    0.01588   193.3   <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 625.25  on 183  degrees of freedom
AIC: 1517.2

Number of Fisher Scoring iterations: 4
```

for which the residual variance exceeds that expected by a factor of 3.5. Of course, the variability in the residuals must go up if there are factors that influence the number of injuries, but which we hadn't measured. Its likely that in most studies there are things that influence the response that haven't been measured, and even if each thing has small effects individually, in aggregate they can cause substantial over-dispersion.

2.3.1 Multiplicative Over-dispersion

There are two ways of dealing with over-dispersion. With `glm` the distribution name can be prefixed with `quasi` and a dispersion parameter estimated:

```
> m2a.4 <- glm(y ~ limit + year + day, family = quasipoisson,
+     data = Traffic)
> summary(m2a.4)

Call:
glm(formula = y ~ limit + year + day, family = quasipoisson,
    data = Traffic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.046741   0.067843  44.909  < 2e-16 ***
limityes    -0.174934   0.064714  -2.703  0.00753 **
year1962    -0.060550   0.060818  -0.996  0.32078
day          0.002416   0.001085   2.227  0.02716 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 3.308492)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

`glm` uses a multiplicative model of over-dispersion and so the estimate is roughly equivalent to how many times greater the variance is than expected, after taking into account the predictor variables. You will notice that although the parameter estimates have changed very little ,the standard errors have gone up and the significance gone down. Over-dispersion, if not dealt with, can result in extreme anti-conservatism because the assumption of independence is contravened. For example, the second lowest number of accidents (8) occurred on the 91^{st} day of 1961 without a speed limit. This should have been the second worst day for injuries over the whole two years, and the probability of observing 9 or less accidents on this day, under the assumption of independence is almost 1 in a 100,000:

```
> ppois(9, exp(m2a.2$coef["(Intercept)"] + 91 *
+     m2a.2$coef["day"]))

[1] 9.80056e-05
```

However, perhaps it was Christmas day and everything was under 5 foot of snow. Although the accidents may have been independent in the sense that all 9 cars didn't crash into each other, they are non-independent in the sense that they all happened on a day where the underlying probability may be different from that underlying any other day (data point).

2.3.2 Additive Over-dispersion

I believe that a model assuming all relevant variables have been measured or controlled for, should **not** be the *de facto* model, and so when you specify `family=poisson` in `MCMCglmm`, over-dispersion is always dealt with[1]. However, `MCMCglmm` does not use a multiplicative model, but an additive model.

```
> prior <- list(R = list(V = 1, nu = 0.002))
> m2a.5 <- MCMCglmm(y ~ limit + year + day, family = "poisson",
+     data = Traffic, prior = prior, verbose = FALSE,
+     pl = TRUE)
```

The element `Sol` contains the posterior distribution of the coefficients of the linear model, and we can plot their marginal distributions:

Notice that the `year1962` coefficient has a high posterior density around zero, in agreement with the over-dispersed `glm` model, and that in general the estimates for the two models are broadly similar. This agreement is superficial.

With additive over-dispersion the linear predictor includes a 'residual', for which a residual variance is estimated (hence our prior specification).

$$E[\mathbf{y}] = \exp(\mathbf{X}\boldsymbol{\beta} + \mathbf{e}) \tag{2.5}$$

At this point it will be handy to represent the linear model in a new way:

$$\mathbf{l} = \boldsymbol{\eta} + \mathbf{e} \tag{2.6}$$

where $\mathbf{l}$ is a vector of latent variables ($\log(E[\mathbf{y}])$ in this case) and eta ($\boldsymbol{\eta}$) the usual symbol for the linear predictor ($\mathbf{X}\boldsymbol{\beta}$). The data we observe are assumed to be Poisson variables with expectation equal to the exponentiated latent variables:

$$\mathbf{y} \sim Pois(\exp(\mathbf{l})) \tag{2.7}$$

Note that the latent variable does not exactly predict y, as it would if the data were Gaussian, because there is additional variability in the Poisson process. In the call to `MCMCglmm` I specified `pl=TRUE` to indicate that I wanted to store the posterior distributions of latent variables. This is not usually necessary and can require a lot of memory (we have 1000 realisations for each of the 182

[1] This is a bit disingenuous - it is no coincidence that the Markov chain without over-dispersion would be reducible

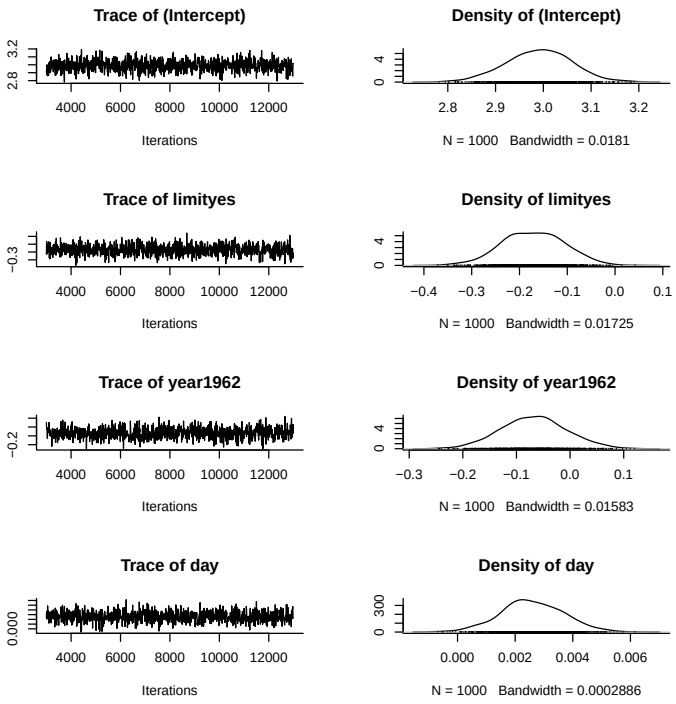

Figure 2.2: MCMC summary plot for the coefficients from a Poisson glm (model `m2a.5`).

data points). However as an example we can obtain the posterior mean residual for data point 92 which is the data from day 92 in 1961 when there was no speed limit:

```
> lat92 <- m2a.5$Liab[, 92]
> eta92 <- m2a.5$Sol[, "(Intercept)"] + m2a.5$Sol[,
+     "day"] * Traffic$day[92]
> resid92 <- lat92 - eta92
> mean(resid92)
```

```
[1] -0.1417317
```

This particular day has a negative expected residual indicating that the probability of getting injured was less than expected for this *particular* realisation of that day in that year. If that *particular* day could be repeated it does not necessarily mean that the actual number of injuries would always be less than expected, because it would follow a Poisson distribution with rate parameter λ =exp(`lat92`)=21.920. In fact there would be a 21.767% chance of having more injuries than if the residual had been zero:

```
> 1 - ppois(exp(mean(eta92)), exp(mean(lat92)))

[1] 0.2176698
```

Like residuals in a Gaussian model, the residuals are assumed to be independently and normally distributed with an expectation of zero and an estimated variance. If the residual variance was zero then **e** would be a vector of zeros and the model would conform to the standard Poisson GLM. However, the posterior distribution of the residual variance is located well away form zero:

```
> plot(m2a.5$VCV)
```

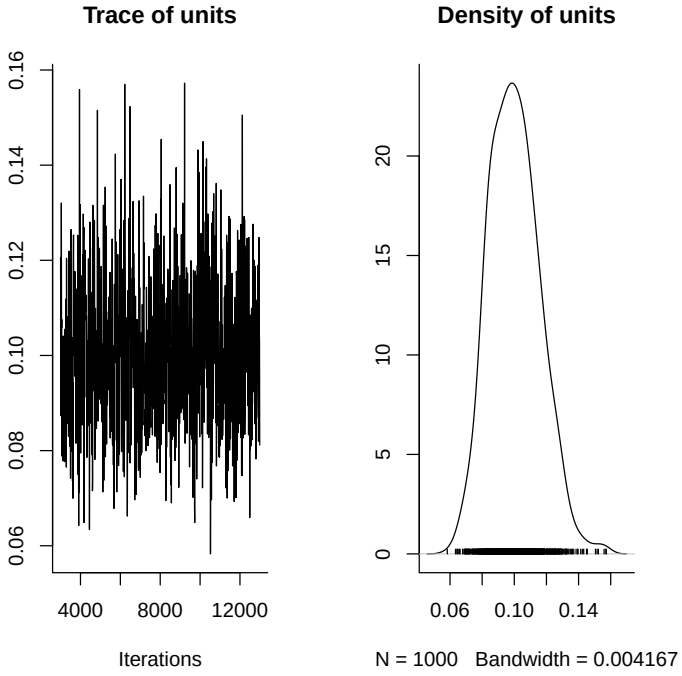

Figure 2.3: MCMC summary plot for the residual (`units`) variance from a Poisson glm (model `m2a.5`). The residual variance models any over-dispersion, and a residual variance of zero implies that the response conforms to a standard Poisson.

The forces that created this residual were only realised on day 92 in 1961, however we could ask hypothetically what if those forces were present on another day. Figure 2.4 plots the first 92 residuals as function of day (red lines) as scatter around the expectation on the log scale (solid black line). Each residual

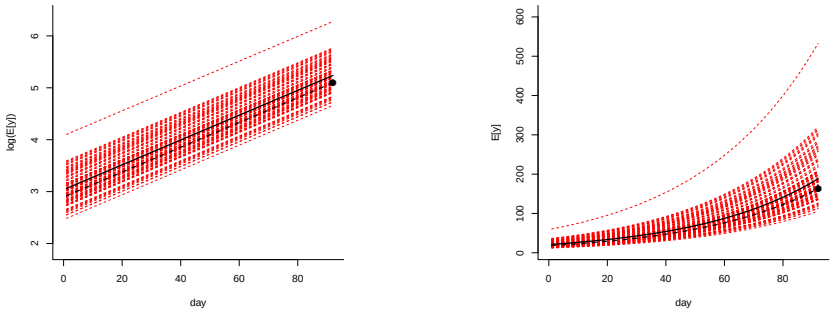

Figure 2.4: The predicted number of injuries on the log scale (left) and data scale (right) as a function of the continuous covariate `day` for 1961 without a speed limit. In order to highlight a point, the slope of the plotted relationship is an order of magnitude steeper than the model `m2a.5` estimate. The solid black line is the value of the linear predictor, and the red dashed lines represent noise around the linear predictor. Each dashed line is a residual from the model, which is only observed for a particular data point. The vertical distance between the black dot and the solid black line is the observed residual on day 92. The black dashed line is the predicted value of a data point observed on other days but with the same residual value. All lines are parallel and linear on the log scale, but this is not the case on the data scale.

is only realised once, and the black dashed line is the hypothetical `resid92` which happened to be observed on day 92 (black circle).

It is perhaps more interesting to know the expected number of injuries that would occur on this date if we had randomly sampled one of these other residuals. To indicate an expectation taken over residuals I have subscripted expectations with e. In Figure 2.5 I have plotted the distribution of the latent variables on day 92. On the log scale the expectation is simply the solid black line η. However, because the exponent function is non-linear this does not translate to the data scale and η is actually equal to the median value on the data scale.

In the `Traffic` example the non linearities are small so the differences in parameter estimates are not large using either multiplicative or additive models. However, multiplying the intercept in model `m2a.5` by half the residual variance is in closer agreement with the quasipoisson model than the raw intercept:

```
> exp(mean(m2a.5$Sol[, "(Intercept)"] + 0.5 * m2a.5$VCV[,
+     1]))

[1] 20.92171

> exp(mean(m2a.5$Sol[, "(Intercept)"]))
```

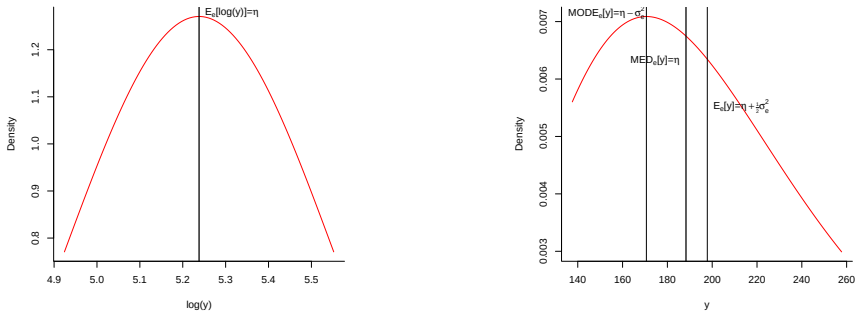

Figure 2.5: The hypothetical distribution for the number of injuries on the log scale (left) and data scale (right) for day 92 in 1961 without a speed limit. These can viewed as vertical slices from Figure 2.4 on day 92. On the log scale the distribution is assumed to be normal and so the residuals are symmetrically distributed around the linear predictor. As a consequence the linear predictor (η) is equal to the mean, median and mode of the distribution on the log scale. Because the exponential function is non-linear this symmetry is lost on the data scale, and the different measures of central tendency do not coincide. Since the residuals are normal on the log scale, the distribution on the data scale is log-normal and so analytical solutions exist for the mean, mode and median. σ^2 is the residual variance.

```
[1] 19.89901

> exp(m2a.3$coef["(Intercept)"])

(Intercept)
   21.54891
```

Analytical results for these transformations can be obtained for the Poisson log-normal, but for other distributions this is not always the case. Section 5.2 gives prediction functions for other types of distribution. One could reasonably ask, why have this additional layer of complexity, why not just stick with the multiplicative model? This brings us to random effects.

2.4 Random effects

In some cases we may have measured variables whose effects we would like to treat as random. Often the distinction between fixed and random is given by example; things like population, species, individual and vial are random, but sex, treatment and age are not. Or the distinction is made using rules of thumb; if there are few factor levels and they are interesting to other people they are fixed. However, this doesn't really confer any understanding about what it means to treat something as fixed or random, and doesn't really allow judgements to be made regarding ambiguous variables (for example year) or give any insight into the fact that in a Bayesian analysis all effects are technically random.

When we treat an effect as fixed we believe that the only information regarding its value comes from data associated with that particular level. If we treat an effect as random we also use this information, but we weight it by what other data tell us about the likely values that the effects could take. In a Bayesian analysis this additional information could come from data not formally included in the analysis, in which case it would be called a prior. In hierarchical models this additional information comes from data associated with other factor levels of the same type.

The degree to which this additional information is important depends on the variability of the effects, as measured by the estimated variance component, and the degree of replication within a particular level. If variability is high then most of the information must come from data associated with an individual effect, particularly if replication within that effect is high. However, if variability and replication are low then extreme mean values of the response for a given level are more likely to be due to sampling error alone, and so the estimates are shrunk towards zero.

It is common to hear things like 'year is a random effect' as if you just have to estimate *a* single effect for all years. It is also common to hear things like 'years is random' as if years were sampled at random. Better to say year effects

are random and understand that it is the effects that are random not the years, and that we're trying to estimate as many effects as there are years. In this sense they're the same as fixed effects, and we can easily treat the year effects as random to see what difference it makes.

Random effect models are often expressed as:

$$E[\mathbf{y}] = \exp(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}) \tag{2.8}$$

where $\mathbf{Z}$ is a design matrix like $\mathbf{X}$, and $\mathbf{u}$ is a vector of parameters like $\boldsymbol{\beta}$. We can specify simple random effect models in the same way that we specified the fixed effects:

```
random =  ~ year
```

although we don't need anything to the left of the $\sim$ because the response is known from the fixed effect specification. In addition, the global intercept is suppressed by default, so in fact this specification produces the design matrix:

```
> Z <- model.matrix(~year - 1, data = Traffic)
> Z[c(1, 2, 184), ]

    year1961 year1962
1          1        0
2          1        0
184        0        1
```

Earlier I said that there was no distinction between fixed and random effects in a Bayesian analysis - all effects are random - so lets not make the distinction and combine the design matrices ($\mathbf{W} = [\mathbf{X}, \mathbf{Z}]$) and combine the vectors of parameters ($\boldsymbol{\theta} = [\boldsymbol{\beta}^{'}, \mathbf{u}^{'}]^{'}$):

$$E[\mathbf{y}] = \exp(\mathbf{W}\boldsymbol{\theta} + \mathbf{e}) \tag{2.9}$$

If we drop year from the fixed terms, the new fixed effect design matrix looks like:

```
> X2 <- model.matrix(y ~ limit + day, data = Traffic)
> X2[c(1, 2, 184), ]

    (Intercept) limityes day
1             1        0   1
2             1        0   2
184           1        1  92
```

and

```
> W <- cbind(X2, Z)
> W[c(1, 2, 184), ]
```

```
    (Intercept) limityes day year1961 year1962
1             1        0   1        1        0
2             1        0   2        1        0
184           1        1  92        0        1
```

You will notice that this new design matrix is exactly equivalent to the original design matrix `X` except we have one additional variable `year1961`. In our first model this variable was absorbed in to the global intercept because it could no be uniquely estimated from the data. What has changed that could make this additional parameter estimable? As is usual in a Bayesian analysis, if there is no information in the data it has to come from the prior. In model `m2a.5` we used the default normal prior for the fixed effects with means of zero, large variances of 10^8, and no covariances. Lets treat the year effects as random, but rather than estimate a variance component for them we'll fix the variance at 10^8 in the prior:

```
> prior <- list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1e+08,
+     fix = 1)))
> m2a.6 <- MCMCglmm(y ~ limit + day, random = ~year,
+     family = "poisson", data = Traffic, prior = prior,
+     verbose = FALSE, pr = TRUE)
> plot(m2a.6$Sol)
```

The estimates for the intercept, day and the effect of a speed limit now appear completely different (Figure 2.6). However, in the original model (`m2a.5`) the prediction for each year is obtained by:

```
> y1961.m2a.5 <- m2a.5$Sol[, "(Intercept)"]
> y1962.m2a.5 <- m2a.5$Sol[, "(Intercept)"] + m2a.5$Sol[,
+     "year1962"]
```

However, for this model we have to add the intercept to both random effects to get the year predictions. `MCMCglmm` does not store the posterior distribution of the random effects by default, but because we specified `pr=TRUE`, the whole of $\boldsymbol{\theta}$ is stored rather than just $\boldsymbol{\beta}$:

```
> y1961.m2a.6 <- m2a.6$Sol[, "(Intercept)"] + m2a.6$Sol[,
+     "year.1961"]
> y1962.m2a.6 <- m2a.6$Sol[, "(Intercept)"] + m2a.6$Sol[,
+     "year.1962"]
```

We can merge the two posterior distributions to see how they compare:

```
> y.m2a.5 <- mcmc(cbind(y1961 = y1961.m2a.5, y1962 = y1962.m2a.5))
> y.m2a.6 <- mcmc(cbind(y1961 = y1961.m2a.6, y1962 = y1962.m2a.6))
> plot(mcmc.list(y.m2a.5, y.m2a.6))
```

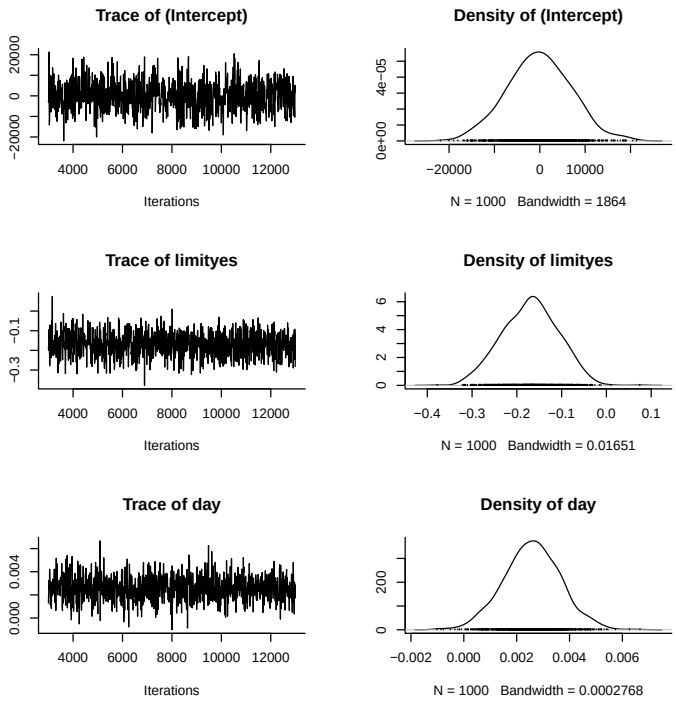

Figure 2.6: MCMC summary plots for the intercept, speed limit and day coefficients from model `m2a.6` where year effects were treated as random. Note the high posterior variance for the intercept.

The posterior distributions are very similar (Figure 2.7; but see Section 2.7 why they are not identical), highlighting the fact that effects that are fixed are those associated with a variance component which has been set *a priori* to something large (10^8 in this case), where effects that are random are associated with a variance component which is not set *a priori* but is estimated from the data. As the variance component tends to zero then no matter how many random effects there are, we are effectively only estimating a single parameter (the variance). This makes sense, if there were no differences between years we only need to estimate a global intercept and not separate effects for each year. Alternatively if the variance is infinite then we need to estimate separate effects for each year. In this case the intercept is confounded with the average value of the random effect, resulting in a wide marginal distribution for the intercept, and strong posterior correlations between the intercept and the mean of the random effects:

```
> plot(c(m2a.6$Sol[, "year.1961"] + m2a.6$Sol[,
+     "year.1962"])/2, c(m2a.6$Sol[, "(Intercept)"]))
```

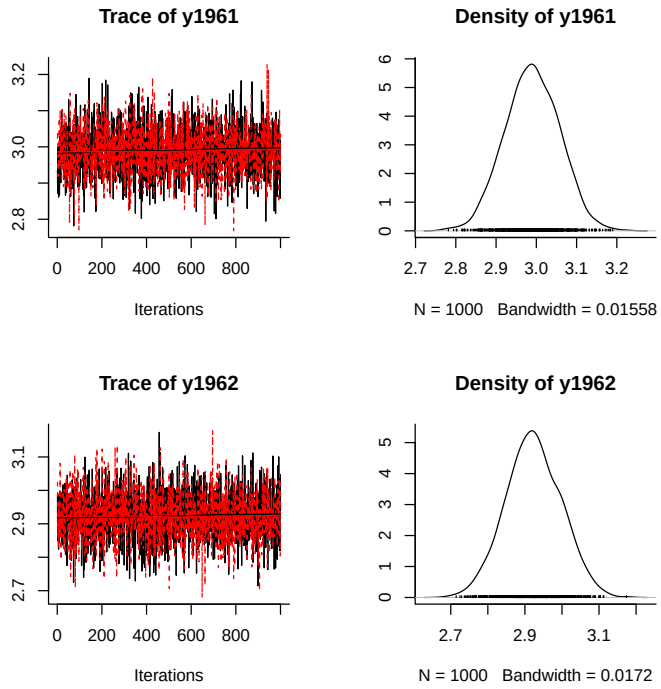

Figure 2.7: MCMC summary plots for the year effects from a model where year effects were treated as fixed (black) and where they were treated as random (red) but with the variance component set at a large value rather than being estimated. The posterior distributions are virtually identical.

With only two levels, there is very little information to estimate the variance, and so we would often make the *a priori* decision to treat year effects as fixed, and fix the variance components to something large (or infinity in a frequentist analysis).

At the moment we have day as a continuous covariate, but we could also have random day effects and ask whether the number of injuries on the same day but in different years are correlated. Rather than fixing the variance component at something large, we'll use the same weaker prior that we used for the residual variance:

```
> Traffic$day <- as.factor(Traffic$day)
> prior <- list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1,
+     nu = 0.002)))
> m2a.7 <- MCMCglmm(y ~ year + limit + as.numeric(day),
+     random = ~day, family = "poisson", data = Traffic,
```

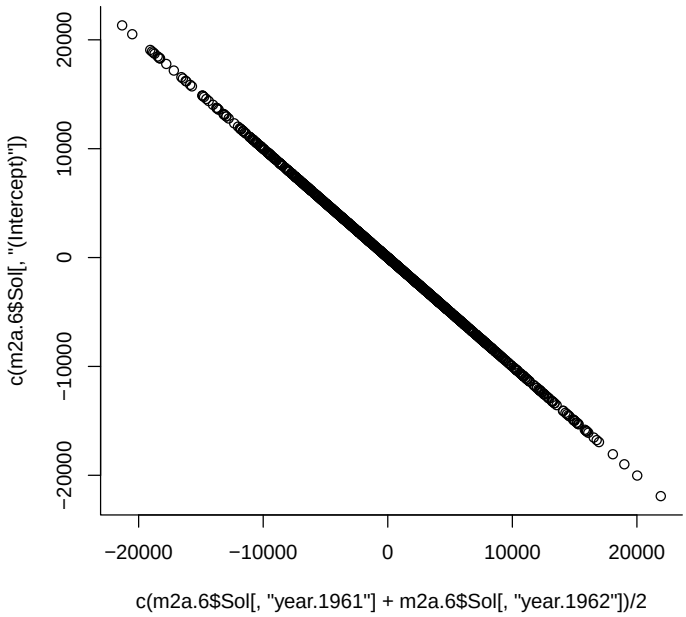

Figure 2.8: Joint posterior distribution of the intercept and the mean of the two random year effects. The variance component associated with year was fixed at a large value (10^8) and so the effects are almost completely confounded.

```
+     prior = prior, verbose = FALSE)
```

`day` has also gone in the fixed formula, but as a numeric variable, in order to capture any time trends in the number of injuries. Most of the over-dispersion seems to be captured by fitting day as a random term (Figure 2.9):

```
> plot(m2a.7$VCV)
```

In fact it explains so much that the residual variance is close to zero and mixing seems to be a problem. The chain would have to be run for longer, and the perhaps an alternative prior specification used.

2.5 Prediction with Random effects

In section 2.3.2 we showed that for non-Gaussian data the expectation of the response variable y is different from the linear predictor if we wish to average

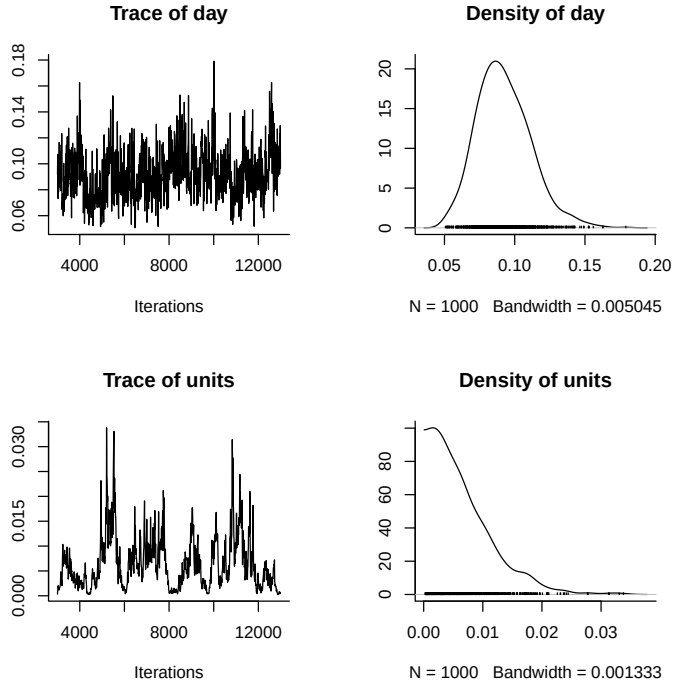

Figure 2.9: MCMC summary plot of the variance component associated with day (top) and the residual variance component (below). The trace for the residual variance shows strong autocorrelation and needs to be ran for longer.

over the residuals. Often it is important to get the expectation after marginalising residuals, and indeed after marginalising other random effects. For example we may not be so interested in knowing the expected number of injuries on the average day, but knowing the expected number of injuries on any random day.

For the Poisson mixed model:

$$E[y] = \mathtt{exp}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Zu} + \mathbf{e}) \tag{2.10}$$

we can marginalise with respect to the random effects, including the over-dispersion residual:

$$E_{u,e}[y] = \exp(\mathbf{X}\boldsymbol{\beta} + 0.5\sigma^2) \tag{2.11}$$

where σ^2 is the sum of the variance components.

For the Binomial mixed model with logit link

$$E[y] = \text{logit}^{-1}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Zu} + \mathbf{e}) \tag{2.12}$$

it is not possible to marginilse with respect to the random effects analytically, but two approximations exist. The first

$$E_{u,e}[y] \approx \text{logit}^{-1}(\mathbf{X}\boldsymbol{\beta} - 0.5\sigma^2 \tanh(\mathbf{X}\boldsymbol{\beta}(1 + 2\exp(-0.5\sigma^2))/6))) \tag{2.13}$$

can be found on p452 in **?** and the second (and possibly less accurate) approximation in **?**:

$$E_{u,e}[y] \approx \text{logit}^{-1}\left(\frac{\mathbf{X}\boldsymbol{\beta}}{\sqrt{1 + (\frac{16\sqrt{3}}{15\pi})^2\sigma^2}}\right) \tag{2.14}$$

A predict function has been implemented for `MCMCglmm` although it is currently incomplete and needs further testing. However, for simpler distributions it should be OK - for example, we can predict the laibality on the latent scale after marginalising the random effects in model `m2a.7`:

```
> predict(m2a.7, marginal = ~day, type = "terms")[1:5]

[1] 3.005080 3.007689 3.010298 3.012907 3.015515
```

or we can predict on the data scale:

```
> predict(m2a.7, marginal = ~day, type = "response")[1:5]

[1] 21.27195 21.32569 21.37960 21.43369 21.48796
```

In addition, credible intervals can be obtained

```
> predict(m2a.7, marginal = ~day, type = "response",
+     interval = "confidence")[1:5, ]

       fit      lwr      upr
1 21.27195 18.38979 24.60332
2 21.32569 18.45328 24.57738
3 21.37960 18.54746 24.59088
4 21.43369 18.64484 24.62558
5 21.48796 18.70649 24.61990
```

as can prediction intervals through posterior predictive simulation:

```
> predict(m2a.7, marginal = ~day, type = "response",
+     interval = "prediction")[1:5, ]

     fit lwr upr
1 20.971   6  37
2 21.112   8  38
3 21.177   6  38
4 21.200   7  37
5 21.706   7  38
```

2.6 Categorical Data

Response variables consisting of levels of some categorical factor are best analysed using `family="categorical"` if the levels have no natural ordering, or `family="ordinal"` if the levels do have a natural ordering, such as never < sometimes < always. The simplest variable of this type is binary data where the response variable is either a zero or a one, and can be analysed as `family="categorical"` (logit link) or `family="ordinal"` (probit link). A binary distribution is a special case of the binomial distribution where the number of trials (`size`) is equal to 1. One way of interpreting a binomial response is to expand it into a series of binary variables and treat the zero's and ones as repeated measures. For example, we could generate two binomial variates each with 5 trials:

```
> success <- rbinom(2, size = 5, prob = c(0.4, 0.6))
> failure <- 5 - success
> binom <- rbind(success, failure)
> colnames(binom) <- c("u.1", "u.2")
> binom

        u.1 u.2
success   1   2
failure   4   3
```

and then expand them into success or failure:

```
> binary <- matrix(rep(c(1, 0, 1, 0), binom), 1,
+     10)
> colnames(binary) <- rep(c("u.1", "u.2"), each = 5)
> binary

     u.1 u.1 u.1 u.1 u.1 u.2 u.2 u.2 u.2 u.2
[1,]   1   0   0   0   0   1   1   0   0   0
```

We can then interpret the `units` variance in a binomial GLMM as accounting for any similarity between repeated measurements made within the same observational unit. If the binary variables within the binomial observation are correlated, this means that the underlying probability for each binomial response differs to a greater degree than can be predicted from the linear predictor. In this example the two probabilities were 0.4 and 0.6 which means that the repeated binary measures would be correlated if we only fitted the intercept (0.5).

If the original data are already binary then there is no information to measure how repeatable trials are within a binomial unit because we only have a single trial per observation. This does not necessarily mean that heterogeneity in the underlying probabilities does not exist, only that we can't estimate it. Imagine we are in a room of 100 people and we are told that 5% of the people will be dead the following day. If the people in the room were a random sample

from the UK population I would worry - *I* probably have a 5% chance of dying. If on the other hand the room was a hospital ward and I was a visitor, I may not worry too much for *my* safety. The point is that in the absence of information, the binary data look the same if each person has a 5% chance of dying or if 5 people have a 100% chance of dying. Most programs set the residual variance to zero and assume the former, but it is important to understand that this is a convenient but arbitrary choice. Given this, it is desirable that any conclusions drawn from the model do not depend on this arbitrary choice. Worryingly, both the location effects (fixed and random) and variance components are completely dependent on the magnitude of the residual variance.

To demonstrate we will use some data from a pilot study on the Indian meal moth (*Plodia interpunctella*) and its granulosis virus (PiGV) collected by Hannah Tidbury & Mike Boots at the University of Sheffield.

```
> data(PlodiaRB)
```

The data are taken from 874 moth pupae for which the `Pupated` variable is zero if they failed to pupate (because they were infected with the virus) or one if they successfully pupated. The 874 individuals are spread across 49 full-sib families, with family sizes ranging from 6 to 38.

To start we will fix the residual variance at 1:

```
> prior.m2b.1 = list(R = list(V = 1, fix = 1), G = list(G1 = list(V = 1,
+     nu = 0.002)))
> m2b.1 <- MCMCglmm(Pupated ~ 1, random = ~FSfamily,
+     family = "categorical", data = PlodiaRB, prior = prior.m2b.1,
+     verbose = FALSE)
```

and then fit a second model where the residual variance is fixed at 2:

```
> prior.m2b.2 = list(R = list(V = 2, fix = 1), G = list(G1 = list(V = 1,
+     nu = 0.002)))
> m2b.2 <- MCMCglmm(Pupated ~ 1, random = ~FSfamily,
+     family = "categorical", data = PlodiaRB, prior = prior.m2b.2,
+     verbose = FALSE)
```

The posterior distribution for the intercept differs between the two models (see Figure 2.10):

```
> plot(mcmc.list(m2b.1$Sol, m2b.2$Sol))
```

as do the variance components (see Figure 2.11):

```
> plot(mcmc.list(m2b.1$VCV, m2b.2$VCV))
```

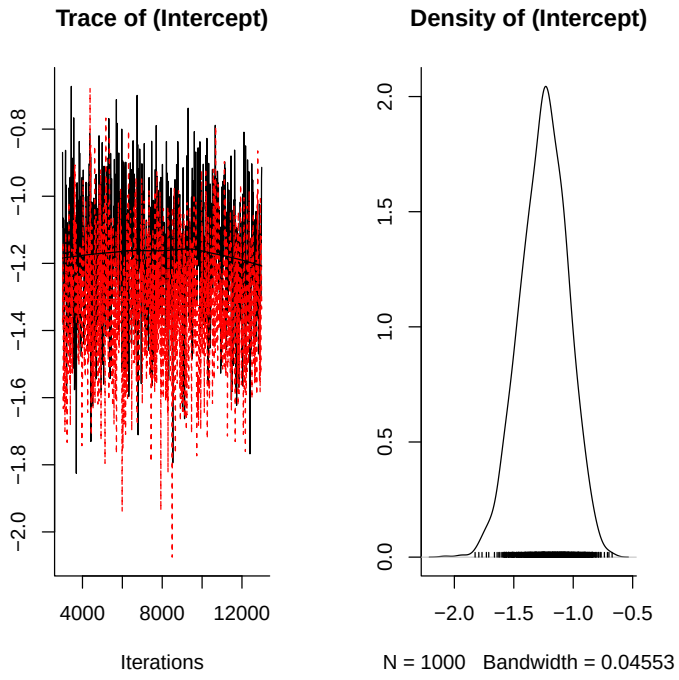

Figure 2.10: MCMC summary plots for the intercept of a binary GLMM where the residual variance was fixed at one (black) and two (red).

Should we worry? Not really. We just have to be careful about how we express the results. Stating that the family variance is 0.744 is meaningless without putting it in the context of the assumed residual variance. It is therefore more appropriate to report the intraclass correlation which in this context is the expected correlation between the state Pupated/Not Pupated, for members of the same family. It can be calculated as:

$$\texttt{IC} = \frac{\sigma^2_{\texttt{FSfamily}}}{\sigma^2_{\texttt{FSfamily}} + \sigma^2_{\texttt{units}} + \pi^2/3} \tag{2.15}$$

for the logit link, which is used when `family=categorical`, or

$$\texttt{IC} = \frac{\sigma^2_{\texttt{FSfamily}}}{\sigma^2_{\texttt{FSfamily}} + \sigma^2_{\texttt{units}} + 1} \tag{2.16}$$

for the probit link, which is used if `family=ordinal` was specified.

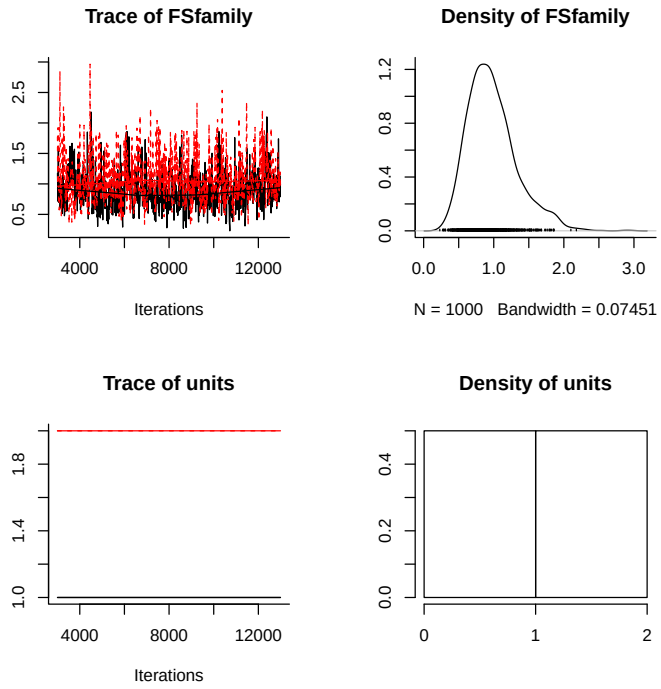

Figure 2.11: MCMC summary plots for the between family variance component of a binary GLMM where the residual variance was fixed at one (black) and two (red).

Obtaing the posterior distribution of the intra-calss correlation for each model shows that they are sampling very similar posterior distributions (see Figure 2.12)

```
> IC.1 <- m2b.1$VCV[, 1]/(rowSums(m2b.1$VCV) + pi^2/3)
> IC.2 <- m2b.2$VCV[, 1]/(rowSums(m2b.2$VCV) + pi^2/3)
> plot(mcmc.list(IC.1, IC.2))
```

Using the approximation due to **?** described earlier we can also rescale the estimates by the estimated residual variance ($\sigma^2_{\texttt{units}}$) in order to obtain the posterior distributions of the parameters under the assumption that the actual residual variance (σ^2_e) is equal to some other value. For location effects the posterior distribution needs to be multiplied by $\sqrt{\frac{1+c^2\sigma^2_e}{1+c^2\sigma^2_{\texttt{units}}}}$ and for the variance components the posterior distribution needs to be multiplied by $\frac{1+c^2\sigma^2_e}{1+c^2\sigma^2_{\texttt{units}}}$ where c is some constant that depends on the link function. For the probit

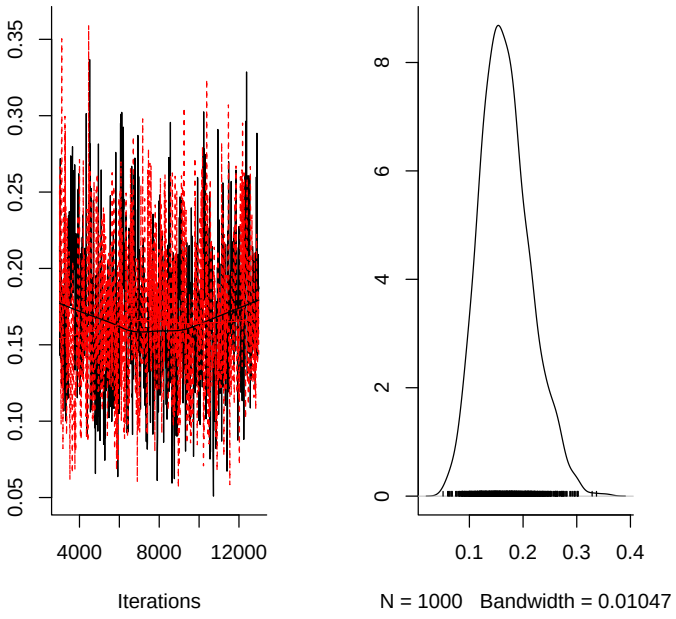

Figure 2.12: MCMC summary plots for the intra-family correlation from a binary GLMM where the residual variance was fixed at one (black) and two (red).

$c = 1$ and for the logit $c = 16\sqrt{3}/15\pi$. We can obtain estimates under the assumption that $\sigma^2_e = 0$:

```
> c2 <- ((16 * sqrt(3))/(15 * pi))^2
> Int.1 <- m2b.1$Sol/sqrt(1 + c2 * m2b.1$VCV[, 2])
> Int.2 <- m2b.2$Sol/sqrt(1 + c2 * m2b.2$VCV[, 2])
> plot(mcmc.list(as.mcmc(Int.1), as.mcmc(Int.2)))
```

The posteriors should be virtually identical under a flat prior (See Figure 2.13) although with different priors this is not always the case. Remarkably, **?** show that leaving a diffuse prior on $\sigma^2_{\texttt{units}}$ and rescaling the estimates each iteration, a Markov chain with superior mixing and convergence properties can be obtained (See section 8).

It should also be noted that a diffuse prior on the logit scale is not necessarily weakly informative on the probability scale. For example, the default setting for the prior on the intercept is $N(0, 10^8)$ on the logit scale, which although

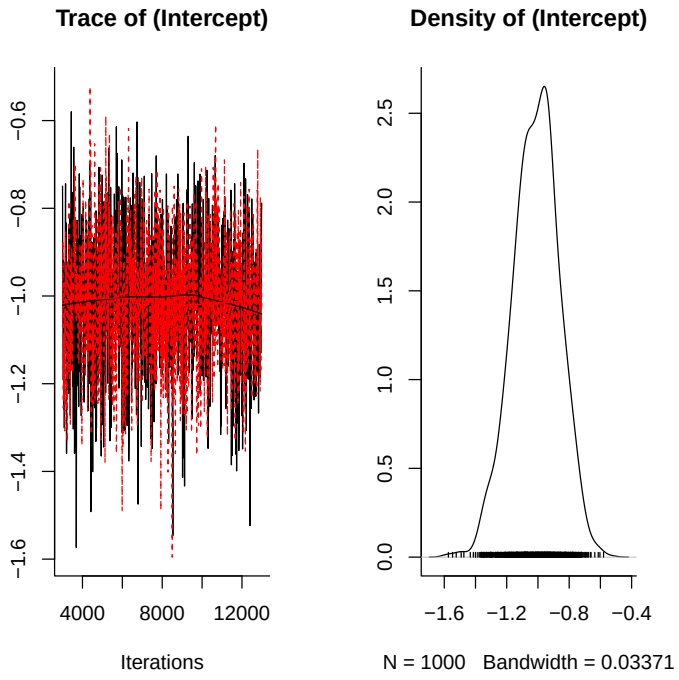

Figure 2.13: MCMC summary plots for the expected proportion of caterpillars pupating from a binary GLMM where the residual variance was fixed at one (black) and two (red).

relatively flat across most of the probability scale, has a lot of density close to zero and one:

```
> hist(plogis(rnorm(1000, 0, sqrt(1e+08))))
```

This diffuse prior can cause problems if there is complete (or near complete) separation. Generally this happens when the binary data associated with some level of a categorical predictor are all success or all failures. For example, imagine we had 50 binary observations from an experiment with two treatments, for the first treatment the probability of success is 0.5 but in the second it is only one in a thousand:

```
> treatment <- gl(2, 25)
> y <- rbinom(50, 1, c(0.5, 0.001)[treatment])
> data.bin <- data.frame(treatment = treatment,
+     y = y)
> table(data.bin)
```

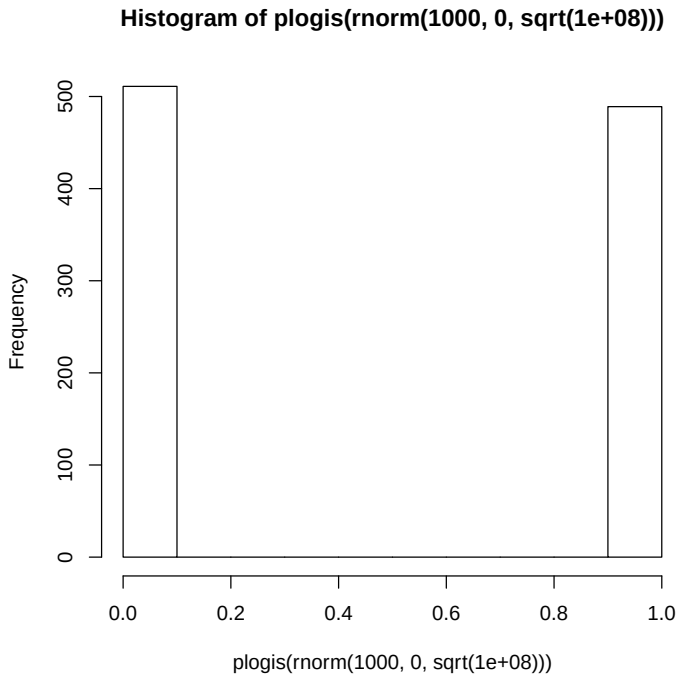

Figure 2.14: Histogram of 1000 random deviates from a normal distribution with a mean of zero and a large variance (10^8) after undergoing an inverse logit transformation.

```
          y
treatment  0  1
        1 15 10
        2 25  0
```

if we analyse using `glm` we see some odd behaviour:

```
> m2c.1 <- glm(y ~ treatment, data = data.bin, family = "binomial")
> summary(m2c.1)

Call:
glm(formula = y ~ treatment, family = "binomial", data = data.bin)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.01077  -1.01077  -0.00008  -0.00008   1.35373
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.4055     0.4082  -0.993    0.321
treatment2   -19.1606  2150.8026  -0.009    0.993

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 50.040  on 49  degrees of freedom
Residual deviance: 33.651  on 48  degrees of freedom
AIC: 37.651

Number of Fisher Scoring iterations: 18
```

the effect of treatment does not appear significant despite the large effect size. This is in direct contrast to an exact binomial test:

```
> m2c.2 <- binom.test(table(data.bin)[2, 2], 25)
> m2c.2

        Exact binomial test

data:  table(data.bin)[2, 2] and 25
number of successes = 0, number of trials = 25,
p-value = 5.96e-08
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.0000000 0.1371852
sample estimates:
probability of success
                     0
```

where the 95% confidence interval for the probability of success is 0.000 to 0.137.

The default `MCMCglmm` model also behaves oddly (see Figure 2.15):

```
> prior.m2c.3 = list(R = list(V = 1, fix = 1))
> m2c.3 <- MCMCglmm(y ~ treatment, data = data.bin,
+     family = "categorical", prior = prior.m2c.3,
+     verbose = FALSE)
> plot(m2c.3$Sol)
```

For these types of problems, I usually remove the global intercept (`-1`) and use the prior $N(0, \sigma^2_{\texttt{units}} + \pi^2/3)$ because this is reasonably flat on the probability scale when a logit link is used. For example,

```
> prior.m2c.4 = list(B = list(mu = c(0, 0), V = diag(2) *
+     (1 + pi^2/3)), R = list(V = 1, fix = 1))
```

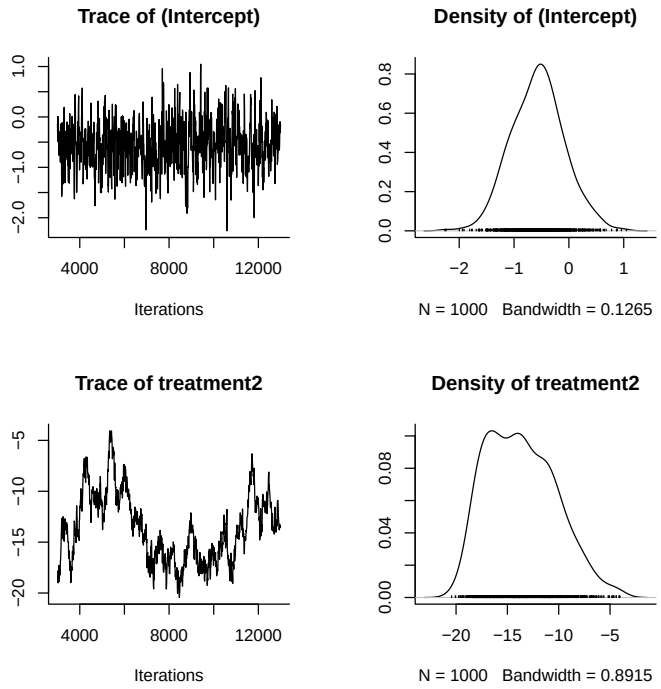

Figure 2.15: MCMC summary plots for the intercept and treatment effect in a binary GLM. In treatment 2 all 25 observations were failures and so the ML estimator on the probability scale is zero and $-\infty$ on the logit scale. With a flat prior on the treatment effect the posterior distribution is improper, and with a diffuse prior (as used here) the posterior is dominated by the high prior densities at extreme values.

```
> m2c.4 <- MCMCglmm(y ~ treatment - 1, data = data.bin,
+     family = "categorical", prior = prior.m2c.4,
+     verbose = FALSE)
> plot(m2c.4$Sol)
```

looks a little better (see Figure 2.15), and the posterior distribution for the probability of success in treatment 2 is consistent with the exact binomial test for which the 95% CI were (0.000 - 0.137). With such a simple model, the prediction for observation 26 is equal to the treatment 2 effect and so we can get the the credible interval (on the data scale) for treatment 2 using the predict function:

```
> predict(m2c.4, interval = "confidence")[26, ]
```

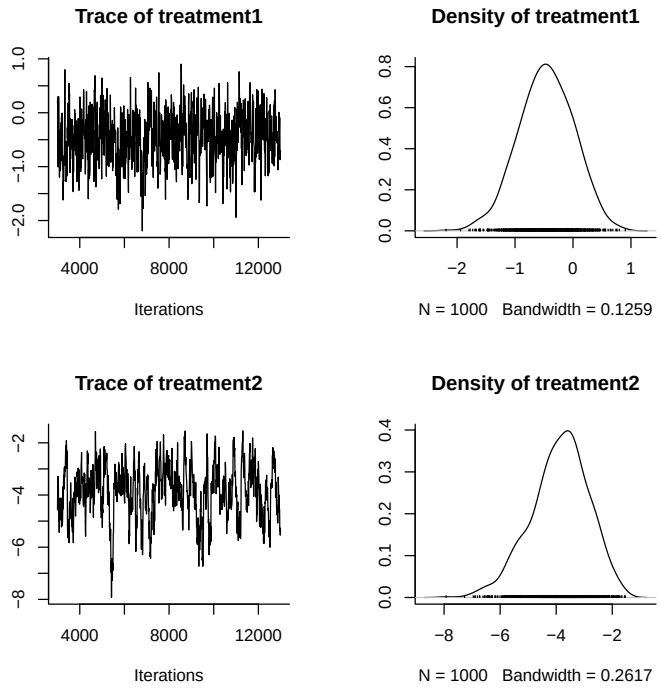

Figure 2.16: MCMC summary plots for the intercept and treatment effect in a binary GLM. In treatment 2 all 25 observations were failures and so the ML estimator on the probability scale is zero and $-\infty$ on the logit scale. A flat prior on the probability scale was used and the posterior distribution is better behaved than if a flat prior on the logit scale had been used (see Figure 2.15).

```
         fit          lwr          upr
0.0446684094 0.0005919544 0.1225549385
```

2.7 A note on fixed effect priors and covariances

Fixed and random effects are essentially the same thing. The only difference is that the variance component for the fixed effects is usually fixed at some large value, whereas the variance component for the random effects is estimated. In section 2.4 I demonstrated this by claiming that a model where year effects were fixed (`m2a.5`) was identical to one where they were treated as random, but with the variance component set to a large value (`m2a.6`). This was a white lie as I did not want to distract attention from the main point. The reason why they were not identical is as follows:

In the fixed effect model (`m2a.5`) we had the prior:

$$\begin{bmatrix} \beta_{\texttt{(Intercept)}} \\ \beta_{\texttt{year1962}} \end{bmatrix} \sim \begin{bmatrix} 10^8 & 0 \\ 0 & 10^8 \end{bmatrix} \tag{2.17}$$

Where $\beta_{\texttt{(Intercept)}}$ and $\beta_{\texttt{year1962}}$ are the fixed effects to be estimated.

Remembering the identity $\sigma^2_{(a+b)} = \sigma^2_a + \sigma^2_b + 2\sigma_{a,b}$, this implies:

$$\begin{bmatrix} \beta_{1961} \\ \beta_{1962} \end{bmatrix} = \begin{bmatrix} \beta_{\texttt{(Intercept)}} \\ \beta_{\texttt{(Intercept)}} + \beta_{\texttt{year1962}} \end{bmatrix} \sim \begin{bmatrix} 10^8 & 10^8 \\ 10^8 & 10^8 + 10^8 \end{bmatrix} = \begin{bmatrix} 10^8 & 10^8 \\ 10^8 & 20^8 \end{bmatrix} \tag{2.18}$$

where β_{1961} and β_{1962} are the actual year effects, rather than the global intercept and the contrast. In hindsight this is a bit odd, for one thing we expect the 1962 effect to be twice as variable as the 1961 effect. With such weak priors it makes little difference, but lets reparameterise the model anyway.

Rather than having a global intercept and a year contrast, we will have separate intercepts for each year:

```
> X3 <- model.matrix(y ~ year - 1, data = Traffic)
> X3[c(1, 2, 184), ]

    year1961 year1962
1          1        0
2          1        0
184        0        1
```

and a prior that has a covariance between the two year effects:

```
> PBV.yfixed <- diag(2) * 1e+08
> PBV.yfixed[1, 2] <- PBV.yfixed[2, 1] <- 1e+08/2
> PBV.yfixed

      [,1]  [,2]
[1,] 1e+08 5e+07
[2,] 5e+07 1e+08

> prior.m2a.5.1 <- list(B = list(mu = rep(0, 2),
+     V = PBV.yfixed), R = list(V = 1, nu = 0.002))
```

This new model:

```
> m2a.5.1 <- MCMCglmm(y ~ year - 1, family = "poisson",
+     data = Traffic, prior = prior.m2a.5.1, verbose = FALSE)
```

has the same form as a mixed effect model with a prior variance of $\frac{10^8}{2}$ for the intercept, and the variance component associated with the random year effects also fixed at $\frac{10^8}{2}$:

```
> prior.m2a.6.1 <- list(B = list(mu = 0, V = 1e+08/2),
+     R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1e+08/2,
+         fix = 1)))
```

This arises because the two random effects have the joint prior distribution:

$$\begin{bmatrix} \beta_{\texttt{year.1961}} \\ \beta_{\texttt{year.1962}} \end{bmatrix} \sim \begin{bmatrix} \frac{10^8}{2} & 0 \\ 0 & \frac{10^8}{2} \end{bmatrix} \tag{2.19}$$

which when combined with the prior for the intercept, $N(0, \frac{10^8}{2})$, gives:

$$\begin{bmatrix} \beta_{1961} \\ \beta_{1962} \end{bmatrix} = \begin{bmatrix} \beta_{\texttt{(Intercept)}} + \beta_{\texttt{year.1961}} \\ \beta_{\texttt{(Intercept)}} + \beta_{\texttt{year.1962}} \end{bmatrix} \sim \begin{bmatrix} \frac{10^8}{2} + \frac{10^8}{2} & \frac{10^8}{2} \\ \frac{10^8}{2} & \frac{10^8}{2} + \frac{10^8}{2} \end{bmatrix} = \begin{bmatrix} 10^8 & \frac{10^8}{2} \\ \frac{10^8}{2} & 10^8 \end{bmatrix} \tag{2.20}$$

which is equivalent to the `PBV.yfixed` parameteristaion of for the two years.

The model:

```
> m2a.6.1 <- MCMCglmm(y ~ 1, random = ~year, family = "poisson",
+     data = Traffic, prior = prior.m2a.6.1, verbose = FALSE,
+     pr = TRUE)
```

is therefore sampling from the same posterior distribution as model `m2a.5.1`.

Chapter 3

Categorical Random Interactions

Random effect specification is a common cause of confusion, especially when we want to form interactions in the random terms. To illustrate the possibilities we'll use data collected on Blue tits.

```
> data(BTdata)
```

The data are morphological measurements (`tarsus` length and `back` colour) made on 828 blue tit chicks from 106 mothers (`dam`). Half the offspring from each mother were swapped with half the offspring from another mother soon after hatching. The nest they were reared in is recorded as `fosternest`.

```
> prior = list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1,
+     nu = 0.002), G2 = list(V = 1, nu = 0.002)))
> m3a.1 <- MCMCglmm(tarsus ~ sex, random = ~dam +
+     fosternest, data = BTdata, verbose = FALSE,
+     prior = prior)
```

fits `sex` as a fixed effect, and `dam` and `fosternest` as random effects.

```
> diag(autocorr(m3a.1$VCV)[2, , ])

        dam  fosternest       units
 0.11102705  0.30977997 -0.02760598
```

```
> plot(m3a.1$VCV)
```

Perhaps the autocorrelation for the `fosternest` variance is a little higher than we would like, and so we may like to run it for longer.

```
> effectiveSize(m3a.1$VCV)
```

```
       dam  fosternest       units
0.11102705  0.30977997 -0.02760598
```

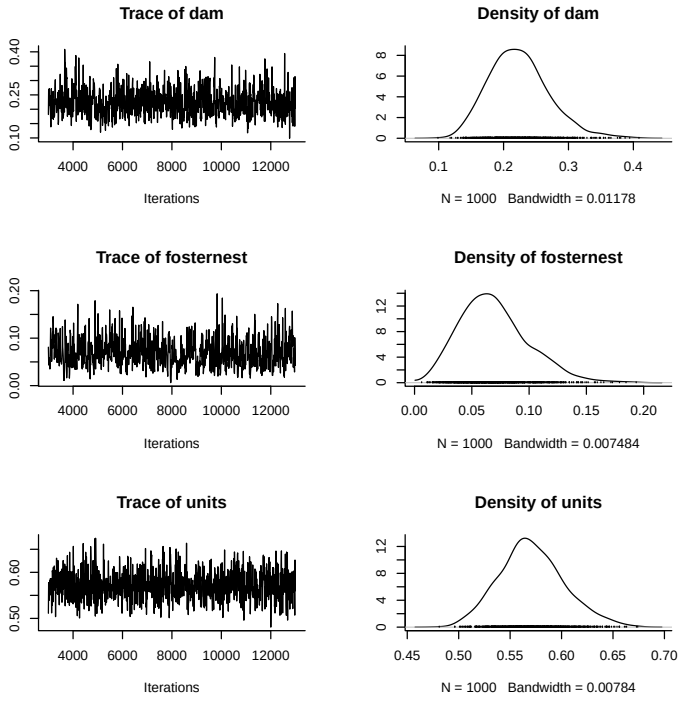

Figure 3.1: MCMC summary plot for the variance components from model `m3a.1`.

```
      dam fosternest      units
  799.3353   472.3695  1000.0000
```

Indeed, we've only sampled the fosternest variance about half as well as the other two variance components.

The posterior correlation between the parameters is low

```
> cor(m3a.1$VCV)
```

```
                  dam fosternest       units
dam         1.00000000 -0.1831774 -0.03579065
fosternest -0.18317736  1.0000000 -0.16361545
units      -0.03579065 -0.1636154  1.00000000
```

which is not that surprising give the data come from an experiment which was designed in order to estimate these variance components. In general, vari-

ance components will show negative posterior correlations because the the total variance is being divided up. Imagine cutting a piece of string; making one bit longer has to reduce the size of the other bits, by necessity. If we hadn't experimentally manipulated the birds then all chicks with the same mother, would be raised in the same nest, and there would be no information in the data to separate these terms. In this case the posterior correlation between these parameters would approach -1 as the prior information goes to zero.

The lower 95% credible interval for the `fosternest` variance is low

```
> HPDinterval(m3a.1$VCV)

               lower     upper
dam        0.13766735 0.3101992
fosternest 0.01904245 0.1300758
units      0.51064413 0.6352781
attr(,"Probability")
[1] 0.95
```

and perhaps a model without it would be better supported, although the DIC suggest not:

```
> priorb <- prior
> priorb[[2]] <- priorb[[2]][-2]
> m3a.2 <- MCMCglmm(tarsus ~ sex, random = ~dam,
+     data = BTdata, verbose = FALSE, prior = priorb)
> m3a.2$DIC

[1] 2014.776

> m3a.1$DIC

[1] 1991.29
```

The tarsus lengths were standardised prior to analysis - this is not recommended, but was done in the original analyses of these data (**?**) so that comparisons would be scale invariant. The original analyses were done in REML where it is hard to get accurate confidence intervals for functions of variance components. With MCMC procedures this is simple. For example if we want to know what proportion of the total variance is explained by dams

```
> HPDinterval(m3a.1$VCV[, 1]/rowSums(m3a.1$VCV))

         lower     upper
var1 0.1792323 0.3452472
attr(,"Probability")
[1] 0.95
```

One nice thing though about standardised data is that effect sizes are immediately apparent. For example, fixed effects are in standard deviation units and the sex effects are non-trivial:

```
> summary(m3a.1)

 Iterations = 3001:12991
 Thinning interval  = 10
 Sample size  = 1000

 DIC: 1991.29

 G-structure:  ~dam

    post.mean l-95% CI u-95% CI eff.samp
dam    0.2237   0.1377   0.3102    799.3

              ~fosternest

           post.mean l-95% CI u-95% CI eff.samp
fosternest   0.06953  0.01904   0.1301    472.4

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units    0.5712   0.5106   0.6353     1000

 Location effects: tarsus ~ sex

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)   -0.4045  -0.5353  -0.2768   1000.0 <0.001 ***
sexMale        0.7693   0.6667   0.8886    979.8 <0.001 ***
sexUNK         0.2101  -0.0512   0.4613   1000.0  0.128
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given that the sexes differ in their mean phenotype it may be worth exploring whether they vary in other ways. For example, perhaps there are sex-limited genes that mean that related brothers resemble each other more than they do their sisters. Perhaps females are less sensitive to environmental variation? To fit these models it will be necessary to understand how the variance functions, such as `us()` and `idh()`, work. We could refit the model `m3a.1` using the random effect specifications:

```
> random = ~us(1):dam + us(1):fosternest
```

```
> random = ~idh(1):dam + idh(1):fosternest
```

and these would give exactly the same answer as the model specified as `~dam+fosternest`. The term inside the brackets is a model formula and is interpreted exactly how you would interpret any R formula expect the intercept is not fitted by default. These formula are therefore fitting an intercept which is interacted with the random effects. For the dam terms we can get a representation of the interaction for the first few levels of `dam`:

```
> levels(BTdata$dam)[1:5]
```

```
[1] "Fem2"    "Fem20"   "Fem3"    "Fem5"    "K983388"
```

	Fem2	Fem20	Fem3	Fem5	K983388	...
(1)	(1).Fem2	(1).Fem20	(1).Fem3	(1).Fem5	(1).K983388	...

Across the top, we have the original dam effects in red, and along the side we have the term defined by the variance structure formula (just the intercept in this case). The interaction forms a new set of factors. Although they have different names from the original `dam` levels, it is clear that there is a one to one mapping between the original and the new factor levels and the models are therefore equivalent. For more complex interactions this is not the case.

We could also fit `sex` in the variance structure model, (i.e. `us(sex):dam` or `idh(sex):dam`)[1]:

	Fem2	Fem20	Fem3	Fem5	K983388	...
Fem	Fem.Fem2	Fem.Fem20	Fem.Fem3	Fem.Fem5	Fem.K983388	...
Male	Male.Fem2	Male.Fem20	Male.Fem3	Male.Fem5	Male.K983388	...
UNK	UNK.Fem2	UNK.Fem20	UNK.Fem3	UNK.Fem5	UNK.K983388	...

which creates three times as many random factors, one associated with offspring of each sex for each each dam.

3.1 `idh` Variance Structure

The different variance functions make different assumptions about how the effects associated with these different factors are distributed. First, we may want

[1] Remember that a global intercept is not fitted by default for variance structure models, and the model formula is essentially `~sex-1`. To add the global intercept, `us(1+sex):dam` could be fitted but this can be harder to interpret because the effects are then `Fem`, `Male-Fem` and `UNK-Fem`. If a `us` structure is fitted, the two models are equivalent reparameterisations of each other although the priors have to be modified accordingly. This is not the case if the variance function is `idh`. In this case the sex-specific variances are allowed to vary as before, but a constant covariance equal to $\sigma^2_{\texttt{Fem}}$ is also assumed

to allow the variance in the effects to be different for each row of factors; i.e. does the identity of a chicks mother explain different amounts of variation depending on the sex of the chick. We can fit this model using the `idh` function and represent our belief in how the effects are distributed as a 3×3 covariance matrix $\mathbf{V}$:

$$\mathbf{V}_{\texttt{dam}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & 0 & 0 \\ 0 & \sigma^2_{\texttt{Male}} & 0 \\ 0 & 0 & \sigma^2_{\texttt{UNK}} \end{bmatrix}$$

In the simpler models we had fitted in Chapters 1 and 2 $\mathbf{V}$ was a scalar ($\mathbf{V} = \sigma^2$) rather than a matrix, and the prior specification was relatively simple. We will come back to prior specifications for covariance matrices in Section 3.6.3, but for now note that the prior for the dam component has `V` as a 3×3 identity matrix:

```
> priorb = list(R = list(V = diag(1), nu = 0.002),
+     G = list(G1 = list(V = diag(3), nu = 0.002),
+         G2 = list(V = 1, nu = 0.002)))
> m3a.3 <- MCMCglmm(tarsus ~ sex, random = ~idh(sex):dam +
+     fosternest, data = BTdata, verbose = FALSE,
+     prior = priorb)
```

The sex specific variances for males and females look pretty similar, but the sex-specific variance for birds with unknown sex is not behaving well. This is not that surprising given that there are only 47 birds with unknown sex and these tend to be thinly spread across dams. This variance component is likely to be dominated by the prior, but for now we'll leave the model as it is and come back to some possible alternative solutions later.

We can extract the marginal means for each variance and place them into a matrix:

```
> Vdam.3 <- diag(colMeans(m3a.3$VCV)[1:3])
> colnames(Vdam.3) <- colnames(m3a.3$VCV)[1:3]
> Vdam.3
```

```
     sexFem.dam sexMale.dam sexUNK.dam
[1,]  0.1757593   0.0000000 0.00000000
[2,]  0.0000000   0.1717858 0.00000000
[3,]  0.0000000   0.0000000 0.06184668
```

Note, that they are in general less than the marginal mean of the dam variance in model `m3a.1` (0.224) where a sex interaction was not fitted. Because the dam effects are assumed to be multivariate normal we can plot an ellipsoid that completely represents their distribution (you can rotate the figure in R):

```
> plotsubspace(Vdam.3, axes.lab = TRUE)
```

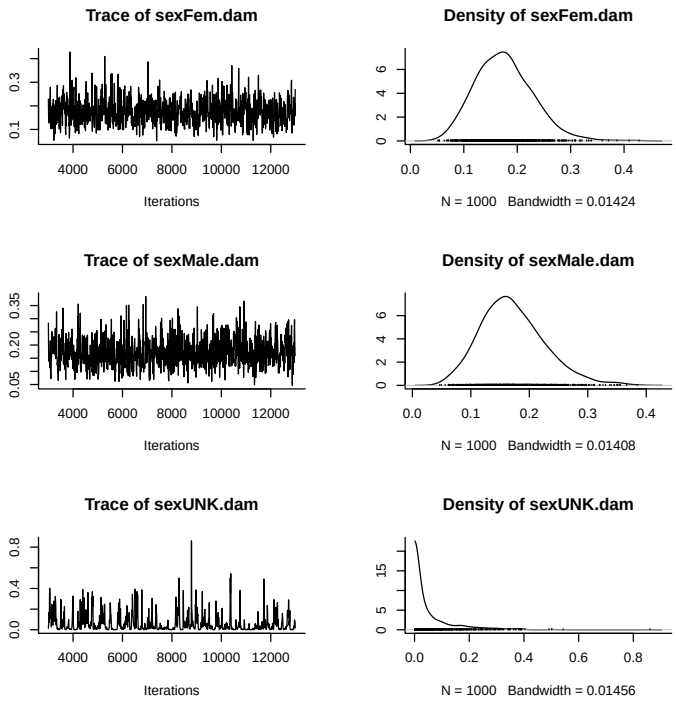

Figure 3.2: MCMC summary plot for the sex-specific dam variance components from model `m3a.3`. The number of chicks with unknown (`UNK`) sex is low, with very little replication within dams. The posterior distribution for the `UNK` variance component is dominated by the prior which has a marginal distribution of `V=1` and `nu=0.002`.

If we had measured the offspring of a lot of dams, and for each dam we had measured a very large number of offspring of each sex, then we could calculate the average tarsus lengths within a nest for males, females and unknowns separately. If we produced a scatter plot of these means the data would have the same shape as this ellipsoid and 95% of the data would lie inside.

3.2 `us` Variance Structure

The oddity of the model, and the meaning of the off-diagonal zeros, should become apparent. We have assumed that the different sexes with in a nest are independent. If we plotted the average tarsus lengths for males against the average tarsus lengths for females for each dam this model implies we should see no relationship. We can relax this assumption using the `us` function which estimates the matrix:

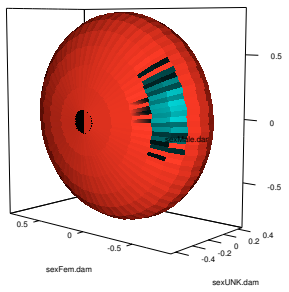

Figure 3.3: Ellipsoid that circumscribes 95% of the expected dam effects as estimated in model `m3a.3`. This can be thought of as a scatter plot of the dam effects between each sex, if the dam effects could be directly measured. Because the covariances of the dam effects between the sexes were set to zero the axes of the ellipsoids are all parallel to the figure axes.

$$\mathbf{V}_{\texttt{dam}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & \sigma_{\texttt{Female,Male}} & \sigma_{\texttt{Female,UNK}} \\ \sigma_{\texttt{Female,Male}} & \sigma^2_{\texttt{Male}} & \sigma_{\texttt{Male,UNK}} \\ \sigma_{\texttt{Female,UNK}} & \sigma_{\texttt{Male,UNK}} & \sigma^2_{\texttt{UNK}} \end{bmatrix}$$

We will now use a prior for the covariance matrix where nu=4 (1 more than the dimension of $\mathbf{V}$) and the prior covariance matrix is an diagonal matrix with small variances. This may seem surprising but the motivation is laid out in Section 3.6.3:

```
> prior.m3a.4 = list(R = list(V = diag(1), nu = 0.002),
+     G = list(G1 = list(V = diag(3) * 0.02, nu = 4),
+         G2 = list(V = 1, nu = 0.002)))
> m3a.4 <- MCMCglmm(tarsus ~ sex, random = ~us(sex):dam +
+     fosternest, data = BTdata, verbose = FALSE,
+     prior = prior.m3a.4)
```

The posterior mean (co)variances for this model show that the covariances are almost the same magnitude as the variances suggesting strong correlations:

```
> Vdam.4 <- matrix(colMeans(m3a.4$VCV)[1:9], 3,
+     3)
> colnames(Vdam.4) <- colnames(m3a.4$VCV)[1:3]
> Vdam.4
```

```
     sexFem:sexFem.dam sexMale:sexFem.dam sexUNK:sexFem.dam
[1,]         0.2311007          0.2004611         0.2039762
[2,]         0.2004611          0.2135747         0.1964712
[3,]         0.2039762          0.1964712         0.2424483
```

The distribution of dam effects in this model looks substantially different (Figure 3.4):

```
> plotsubspace(Vdam.4, axes.lab = TRUE, wire.frame = T)
```

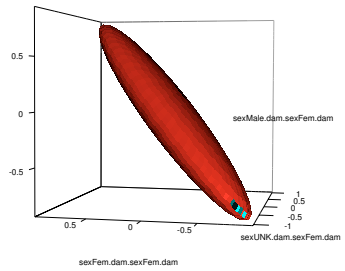

Figure 3.4: Ellipsoid that circumscribes 95% of the expected dam effects as estimated in model `m3a.4`. This can be thought of as a scatter plot of the dam effects between each sex, if the dam effects could be directly measured. The correlations of the dam effects between the sexes were estimated and found to be close to one, and the sex-specific variances were all roughly equal in magnitude. Consequently the major axis of the ellipsoid lies at 45^o to the figure axes.

Covariances can be hard to interpret, and I usually find correlations easier to think about. They can also be useful for detecting problems in the chain. In model `m3a.1` the dam variance for chicks with unknown sex was behaving badly and was getting 'trapped' at zero. When fitting a 2×2 covariance matrix similar things can happen when correlations are close to -1 and 1, and this may not be obvious from the marginal distribution of the covariances:

```
> plot(posterior.cor(m3a.4$VCV[, 1:9])[, c(2, 3,
+     7)])
```

All the correlations are very close to one, and the variances all pretty equal so we'd probably consider the simpler model. We could try using DIC to compare models, although given the different prior specifications for the two models it is unclear whether this would be meaningful. However, the simpler model does seem to have better support as intuition suggests:

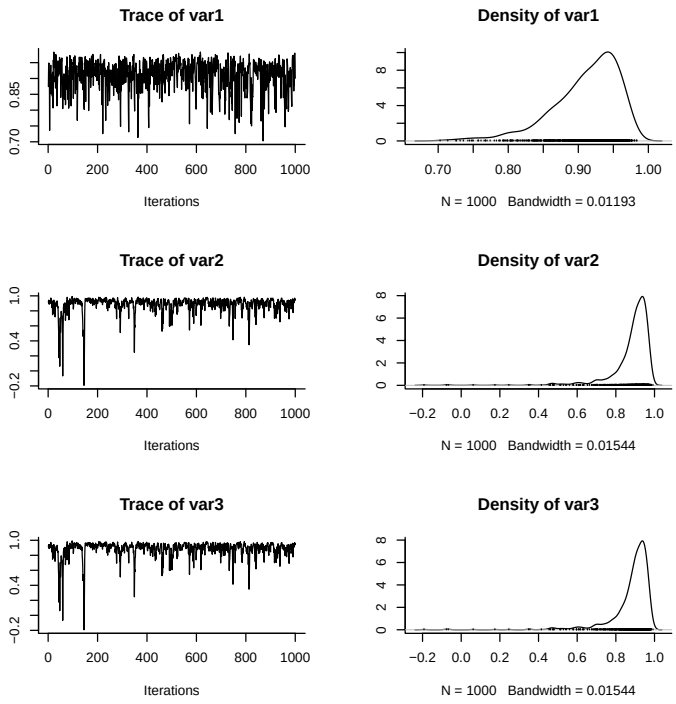

Figure 3.5: MCMC summary plot for the between sex correlations in dam effects from model `m3a.4`.

```
> m3a.4$DIC

[1] 1997.34

> m3a.1$DIC

[1] 1991.29
```

3.3 Compound Variance Structures

There are also ways of specifying models that lie somewhere between the simple model (`mBT`), where dam effects are assumed to be equivalent across the sexes, and the most complex model (`mBT2`), where dam effects are allowed to vary across the sexes and covary between the sexes to different degrees. Some alternatives are listed in Table 3.1.

To be completed ....

lmer	MCMCglmm/asreml	No. Parameters	Variance	Correlation
(1|dam)	dam	1	$\begin{bmatrix} V & V & V \\ V & V & V \\ V & V & V \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
(sex-1|dam)	us(sex):dam	6	$\begin{bmatrix} V_{1,1} & C_{1,2} & C_{1,3} \\ C_{1,2} & V_{2,2} & C_{2,3} \\ C_{1,3} & C_{2,3} & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{1,2} & 1 & r_{2,3} \\ r_{1,3} & r_{2,3} & 1 \end{bmatrix}$
(1|sex:dam)	sex:dam	1	$\begin{bmatrix} V & 0 & 0 \\ 0 & V & 0 \\ 0 & 0 & V \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
(1|dam)+(1|sex:dam)	dam+sex:dam	2	$\begin{bmatrix} V_1+V_2 & V_1 & V_1 \\ V_1 & V_1+V_2 & V_1 \\ V_1 & V_1 & V_1+V_2 \end{bmatrix}$	$\begin{bmatrix} 1 & r & r \\ r & 1 & r \\ r & r & 1 \end{bmatrix}$
-	idh(sex):dam	3	$\begin{bmatrix} V_{1,1} & 0 & 0 \\ 0 & V_{2,2} & 0 \\ 0 & 0 & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
-	corg(sex):dam	3	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{1,2} & 1 & r_{2,3} \\ r_{1,3} & r_{2,3} & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{1,2} & 1 & r_{2,3} \\ r_{1,3} & r_{2,3} & 1 \end{bmatrix}$
-	corgh(sex):dam	3	$\begin{bmatrix} V_{1,1} & r_{1,2}\sqrt{V_{1,1}V_{2,2}} & r_{1,3}\sqrt{V_{1,1}V_{2,2}} \\ r_{1,2}\sqrt{V_{1,1}V_{2,2}} & V_{2,2} & r_{2,3}\sqrt{V_{2,2}V_{2,3}} \\ r_{1,3}\sqrt{V_{1,1}V_{3,3}} & r_{2,3}\sqrt{V_{2,2}V_{3,3}} & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{1,2} & 1 & r_{2,3} \\ r_{1,3} & r_{2,3} & 1 \end{bmatrix}$

Table 3.1: Different random effect specifications in `lmer`, `MCMCglmm` and `asreml`. `sex` is a factor with three levels so the resulting matrix is 3×3. Continuous variables can also go on the LHS of the pipe, or within the variance structure functions (e.g. `us`,`idh`). In this case the associated parameters are regression coefficients for which a variance is estimated. For example, if the chicks were of different ages (or we'd measured the same chicks at different ages) we may want to see if the growth rate is more similar for chicks raised by the same mother. `(1+age|dam)` or `us(1+age):dam` estimates a 2×2 matrix which includes the variance in intercepts (when `age`=0), the variance in slopes, and the covariance that exists between them. Fixed parameters are in red.

3.4 Heterogenous Residual Variance

To be started... In short - if you've fitted a sex by dam interaction I would always allow the sexes to have different residual variances. Use `rcov=~idh(sex):units`.

3.5 Contrasts and Covariances

A general method for seeing what a particular random specification means in terms of the original variables is to realise that

$$\mathbf{\Sigma} = \mathbf{Z}\mathbf{V}\mathbf{Z}^{'} \tag{3.1}$$

where $\mathbf{\Sigma}$ is the covariance matrix for the original set of variables and $\mathbf{V}$ the variances associated with the variance structure model. $\mathbf{Z}$ is the random effect design matrix. Equation 3.1 implies:

$$\mathbf{V} = \mathbf{Z}^{-1}\mathbf{\Sigma}(\mathbf{Z}^{'})^{-1} \tag{3.2}$$

or alternatively:

$$\mathbf{V} = (\mathbf{Z}\mathbf{Z}^{'})^{-}\mathbf{Z}^{'}\mathbf{\Sigma}\mathbf{Z}(\mathbf{Z}^{'}\mathbf{Z})^{-} \tag{3.3}$$

if $\mathbf{Z}$ is non-square and/or singular, where $^{-}$ is a generalised inverse.

3.6 Priors for Covariance Matrices

Priors for covariance matrices are tricky. What maybe non-informative for a covariance may be informative for a correlation and *vice versa*.

3.6.1 Priors for `us` structures

A useful result is that the marginal distribution of a variance is also inverse - Wishart distributed:

$$\sigma_1^2 \sim IW\left(\texttt{nu}^{*}\texttt{=nu-dim(V)+1},\ \texttt{V}^{*} = \frac{\texttt{nu}}{\texttt{nu}^{*}}\texttt{V[1,1]}\right)$$

using the first variance as an example, and indicating the new parameters with an asterisk.

An uninformative prior for the correlations is an improper prior with `V=diag(dim(V))*0` and `nu=dim(V)+1`. For the 3×3 sex by dam covariance matrix in model `m3a.4` we used a proper prior with `V=diag(3)*0.02` and `nu=4` in the hope that this would be relatively uninformative for the correlations. We can plot the marginal density of the variances for this distribution as we did in Chapter 1:

```
> nu.ast <- prior.m3a.4$G$G1$nu - dim(prior.m3a.4$G$G1$V)[1] +
+     1
> V.ast <- prior.m3a.4$G$G1$V[1, 1] * (prior.m3a.4$G$G1$nu/nu.ast)
> xv <- seq(1e-16, 1, length = 100)
> dv <- MCMCpack::dinvgamma(xv, shape = nu.ast/2,
+     scale = (nu.ast * V.ast)/2)
> plot(dv ~ xv, type = "l")
```

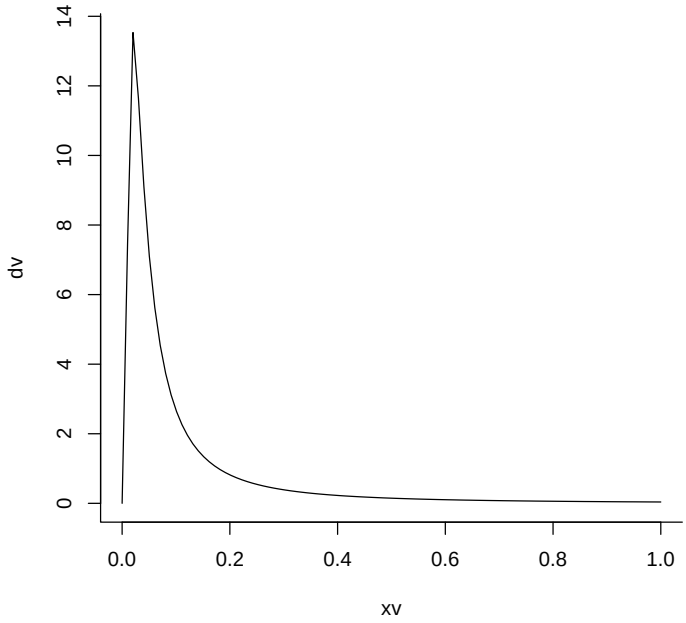

Figure 3.6: Marginal prior distribution of a variance using an inverse Wishart prior for the covariance matrix with `V=diag(3)*0.02` and `nu=4`.

In Chapter 2 we saw that a non-informative prior for a variance component was `V=0` and `nu=-2`. This result generalises to covariance matrices where the improper prior `V=diag(dim(V))*0` and `nu=dim(V)-3` is non-informative for the variances and covariances. This can be verified for the variances using the results derived above for the marginal distribution:

$$\begin{aligned} \sigma_1^2 \sim & \quad IW\left(\texttt{nu}^*\texttt{=dim(V)-3-dim(V)+1},\ \texttt{V}^* = \frac{\texttt{nu}}{\texttt{nu}^*}\texttt{0}\right) \\ \sim & \quad IW\left(\texttt{nu}^*\texttt{=-2},\ \texttt{V}^* = \texttt{0}\right) \end{aligned}$$

3.6.2 Priors for `idh` structures

For `idh` the diagonal elements of the matrix are independent and each variance is distributed as[2]:

$$\sigma_1^2 \sim IW\left(\texttt{nu}^*\texttt{=nu},\ \texttt{V}^* = \texttt{V[1,1]}\right)$$

3.6.3 Priors for `corg` and `corgh` structures

For `corg` and `corgh` structures[3] the diagonals of `V` define the fixed variances (`corgh`) or are ignored and the variances set to one (`corg`). I use the prior specification in **?** where `nu` controls how much the correlation matrix approaches an identity matrix. The marginal distribution of individual correlations (r) is given by **?** (and **?**):

$$Pr(r) \propto (1-r^2)^{\frac{\texttt{nu-dim(V)-1}}{2}}, \quad |r| < 1 \tag{3.4}$$

and as shown above setting `nu` =`dim(V)+1` results in marginal correlations that are uniform on the interval [-1,1].

In most cases correlation matrices do not have known form and so cannot be directly Gibbs sampled. `MCMCglmm` uses a method proposed by **?** with the target prior as in **?**. Generally this algorithm is very efficient as the Metropolis-Hastings acceptance probability only depends on the degree to which the candidate prior and the target prior (the prior you specify) conflict. The candidate prior is equivalent to the prior in **?** with `nu=0` so as long as a diffuse prior is set, mixing is generally not a problem. If `nu=0` is set (the default) then the Metropolis-Hastings steps are always accepted resulting in Gibbs sampling. However, a prior of this form puts high density on extreme correlations which can cause problems if the data give support to correlations in this region.

[2]IMPORTANT: In versions < 2.05 priors on each variance of an `idh` structure were distributed as $IW\left(\texttt{nu}^*\texttt{=nu-dim(V)+1},\ \texttt{V}^* = \texttt{V[1,1]}\right)$ but this was a source of confusion and was changed.

[3]In versions < 2.18 `cor` fitted what is now a `corg` structure. The reason for the change is to keep the `asreml` and `MCMCglmm` syntax equivalent. However, the `corgh` structure in asreml is a reparameterised `us` structure whereas in `MCMCglmm` the variances are fixed in the prior.

Chapter 4

Continuous Random Interactions

In Lecture 3 we saw how we could define a linear model within a variance function and then interact these terms with a random effect. In the example, we did this in order to fit a `sex` by `dam` interaction:

```
us(sex):dam
```

The term entering into the variance function model was categorical, and we saw that by fitting the interaction we were essentially estimating the parameters of the covariance matrix:

$$\mathbf{V}_{\texttt{dam}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & \sigma_{\texttt{Female,Male}} & \sigma_{\texttt{Female,UNK}} \\ \sigma_{\texttt{Female,Male}} & \sigma^2_{\texttt{Male}} & \sigma_{\texttt{Male,UNK}} \\ \sigma_{\texttt{Female,UNK}} & \sigma_{\texttt{Male,UNK}} & \sigma^2_{\texttt{UNK}} \end{bmatrix}$$

We are also free to define the variance function model with continuous covariates, or even a mixture of continuous and categorical factors, and the resulting covariance matrix is interpreted in the same way.

4.1 Random Regression

As an example, we'll use a longitudinal data set on chicken growth (See Figure 4.1):

```
> data(ChickWeight)
```

The data consist of body weights (`weight`) for 50 chicks (`Chick`) measured up to 12 times over a 3 week period. The variable `Time` is the number of days since hatching, and `Diet` is a four level factor indicating the type of protein diet the chicks received.

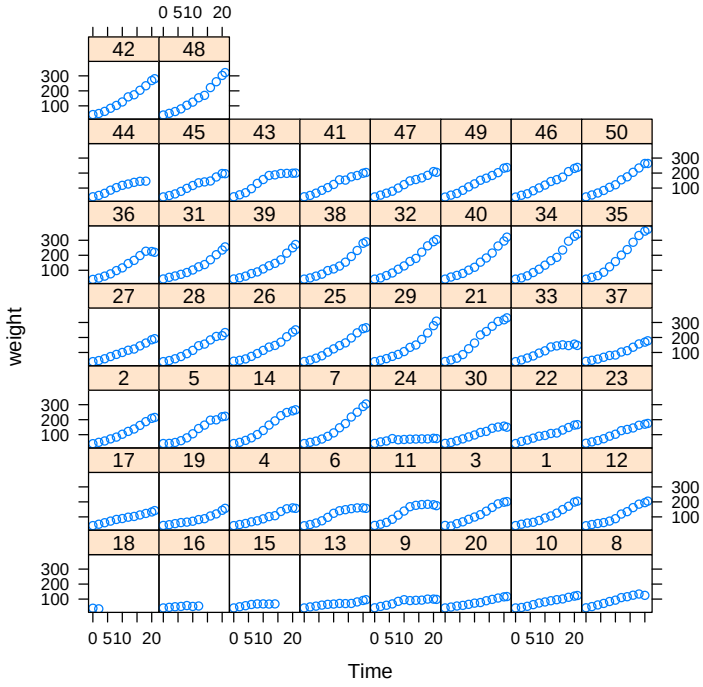

Figure 4.1: Weight data of 50 chicks from hatching until three weeks old.

```
> xyplot(weight ~ Time | Chick, data = ChickWeight)
```

Growth curves tend to be sigmoidal and so one of the non-linear growth curves such as the Gompertz or logistic may be a good starting model. However, these can be tricky to use and an alternative is to try and capture the form of the curve using polynomials. We'll start with a quadratic function at the population level and and fit chick as a random term:

```
> prior.m4a.1 <- list(R = list(V = 1e-07, n = -2),
+     G = list(G1 = list(V = 1, n = 1)))
> m4a.1 <- MCMCglmm(weight ~ Diet + poly(Time, 2,
+     raw = TRUE), random = ~Chick, data = ChickWeight,
+     verbose = FALSE, pr = TRUE, prior = prior.m4a.1,
+     saveX = TRUE, saveZ = TRUE)
```

We've saved the random chick effects so we can plot the predicted growth functions for each bird. For now we will just predict the growth function assuming that all birds were on Diet 1 (the intercept):

```
> pop.int <- posterior.mode(m4a.1$Sol[, 1])
> pop.slope <- posterior.mode(m4a.1$Sol[, 5])
> pop.quad <- posterior.mode(m4a.1$Sol[, 6])
> chick.int <- posterior.mode(m4a.1$Sol[, c(7:56)])
```

We need to combine these parameter estimates with the polynomials for `Time` which are just the sequence $\texttt{Time}^0, \texttt{Time}^1, \texttt{Time}^2 \ldots$ and so on. We can then plot the population expected population growth curve, and around that the predicted growth curves for each chick (we don't need to bother with $\texttt{Time}^0$ since this is always one):

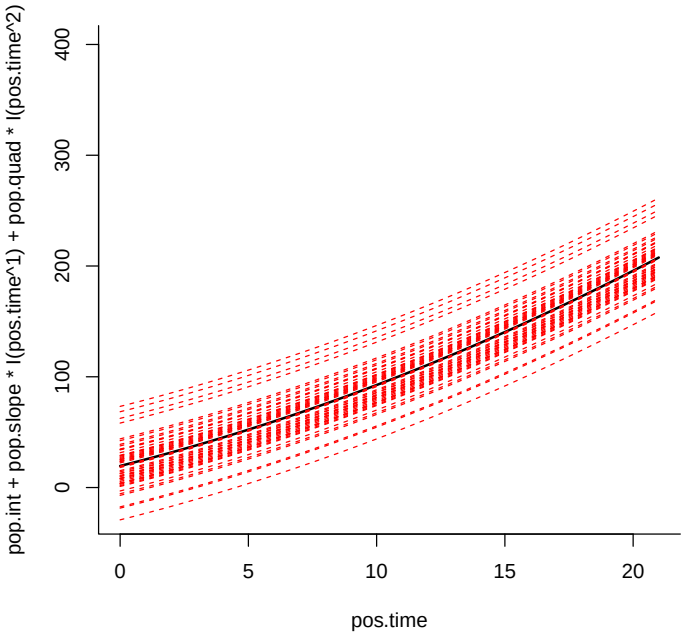

Figure 4.2: Predicted weights of each chick as a function of age. A quadratic population growth curve (black) is fitted with random chick intercepts.

The population growth curve is slightly convex because of the quadratic term, and the predictions for each chick are parallel to this curve. By fitting chick as a random effect we have allowed variation in the intercept only, and often this is not enough. We can get a feel for how well the model fits the data by overlaying the predictions with actual values. In the call to `MCMCglmm` we specified `saveX=TRUE` and `saveZ=TRUE` indicating that we wanted to save the design matrices. We can combine these matrices into the design matrix $\mathbf{W}$ and multiply by the parameter vector $\boldsymbol{\theta}$ to get the predictions (See Eq. 2.9):

```
> W.1<-cbind(m4a.1$X, m4a.1$Z)
> prediction.1<-W.1%*%posterior.mode(m4a.1$Sol)
> xyplot(weight+prediction.1@x~Time|Chick, data=ChickWeight)
```

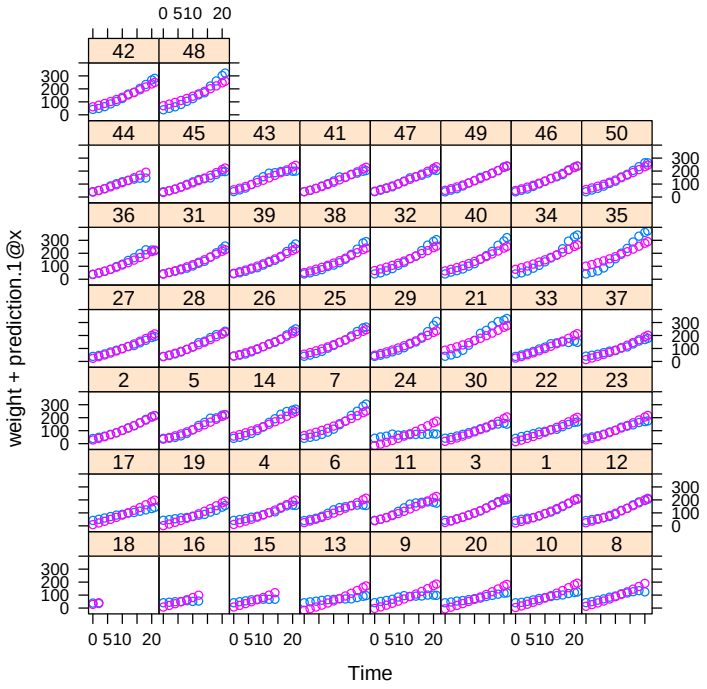

Figure 4.3: Weights of each chick as a function of age in blue, with the predicted weights in purple. A quadratic population growth curve was fitted with random chick intercepts.

The predictions don't look that bad, but you will notice that for some chicks (e.g. 13,19,34) the slope of the predicted growth seems either to shallow, or too steep. To account for this we can start by fitting `us(1+time):Chick`. The linear model inside the variance function has two parameters, an intercept (`1`) and a regression slope associated with `Time` which define the set of interactions:

	Chick1	Chick2	Chick3	...
(Intercept)	(Intercept).Chick1	(Intercept).Chick2	(Intercept).Chick3	...
Time	Time.Chick1	Time.Chick2	Time.Chick3	...

Each chick now has an intercept and a slope, and because we have used the `us` function we are estimating the 2×2 matrix:

$$\mathbf{V}_{\texttt{Chick}} = \begin{bmatrix} \sigma^2_{\texttt{(Intercept)}} & \sigma_{\texttt{(Intercept),Time}} \\ \sigma_{\texttt{(Intercept),Time}} & \sigma^2_{\texttt{Time}} \end{bmatrix}$$

$\sigma^2_{\texttt{(Intercept)}}$ is the amount of variation in intercepts between chicks, and $\sigma^2_{\texttt{Time}}$ is the amount of variation in the regression slopes between chicks. If the `idh` function had been used the covariance would have been set to zero and we could have interpreted variation in intercepts as variation in overall size, and variation in slopes as variation in growth rate. However, there is often covariance between intercepts and slopes and it is usually a good idea to use the `us` function and estimate them (see Section 4.3). We shall do so:

```
> prior.m4a.2 <- list(R = list(V = 1e-07, nu = -2),
+     G = list(G1 = list(V = diag(2), nu = 2)))
> m4a.2 <- MCMCglmm(weight ~ Diet + poly(Time, 2,
+     raw = TRUE), random = ~us(1 + Time):Chick,
+     data = ChickWeight, verbose = FALSE, pr = TRUE,
+     prior = prior.m4a.2, saveX = TRUE, saveZ = TRUE)
```

The traces look OKish for the chick (co)variance matrices (Figure 4.4) but notice that the the estimate of intercept-slope correlation is close to the boundary of parameter space (-1):

```
> int.slope.cor <- m4a.2$VCV[, 2]/sqrt(m4a.2$VCV[,
+     1] * m4a.2$VCV[, 4])
> posterior.mode(int.slope.cor)

      var1
-0.9817454
```

and shows strong autocorrelation

```
> autocorr(int.slope.cor)

, , 1

                [,1]
Lag 0     1.000000000
Lag 10    0.198725570
Lag 50    0.002783530
Lag 100   0.005720720
Lag 500  -0.001004745
```

and we should run it for longer in order to sample the posterior adequately. For now we will carry on and obtain the predictions from the model we ran, but using the `perdict` function rather than dong it 'by hand':

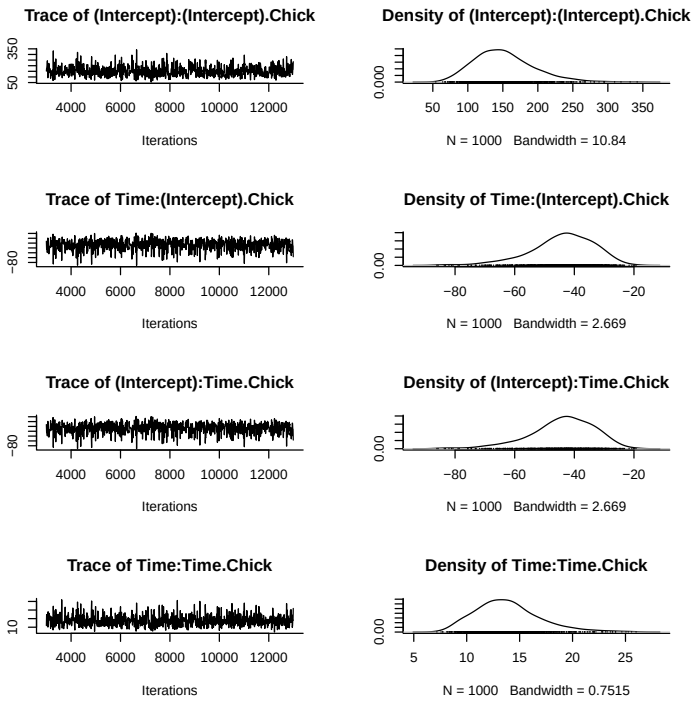

Figure 4.4: MCMC summary plots for the chick covariance components from model `m4a.2`. The lower and upper plots are the intercept and slope variance components respectively, and the middle two plots are the intercept-slope covariance.

```
> xyplot(weight + predict(m4a.2, marginal = NULL) ~
+     Time | Chick, data = ChickWeight)
```

and we can see that the fit is much better (See Figure 4.5). In theory we could fit higher order random regressions (data and prior permitting) and use something like DIC to choose which is the best compromise between the fit of the model to the data and how many effective parameters were fitted. For example we could go from the 1^{st} order random regression to a 2^{nd} order model:

```
> prior.m4a.3 <- list(R = list(V = 1, n = 0.002),
+     G = list(G1 = list(V = diag(3), n = 3)))
> m4a.3 <- MCMCglmm(weight ~ Diet + poly(Time, 2,
+     raw = TRUE), random = ~us(1 + poly(Time, 2,
+     raw = TRUE)):Chick, data = ChickWeight, verbose = FALSE,
+     pr = TRUE, prior = prior.m4a.3, saveX = TRUE,
+     saveZ = TRUE)
```

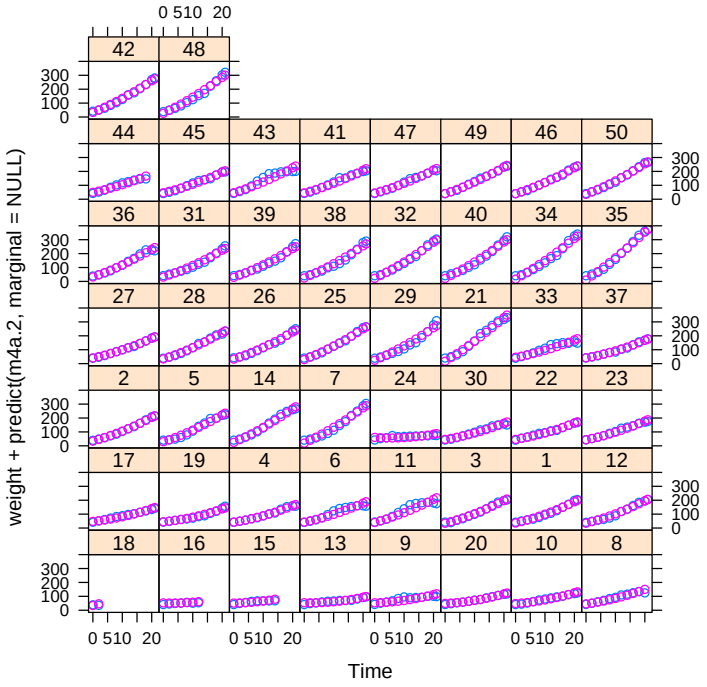

Figure 4.5: Weights of each chick as a function of age in blue, with the predicted weights in purple. A quadratic population growth curve was fitted with a first order random regression for chicks (i.e. a random intercept-slope model).

and obtain the 3×3 covariance matrix:

$$\mathbf{V}_{\texttt{Chick}} = \begin{bmatrix} \sigma^2_{\texttt{(Intercept)}} & \sigma_{\texttt{(Intercept),Time}} & \sigma_{\texttt{(Intercept),Time}^2} \\ \sigma_{\texttt{(Intercept),Time}} & \sigma^2_{\texttt{Time}} & \sigma_{\texttt{Time,Time}^2} \\ \sigma_{\texttt{(Intercept),Time}^2} & \sigma_{\texttt{Time,Time}^2} & \sigma^2_{\texttt{Time}^2} \end{bmatrix}$$

The model predicts the chick weights to an even better degree (See Figure 4.6)

```
> xyplot(weight + predict(m4a.3, marginal = NULL) ~
+     Time | Chick, data = ChickWeight)
```

and the DIC has gone down, suggesting that the model is better:

```
> m4a.1$DIC

[1] 5525.147
```

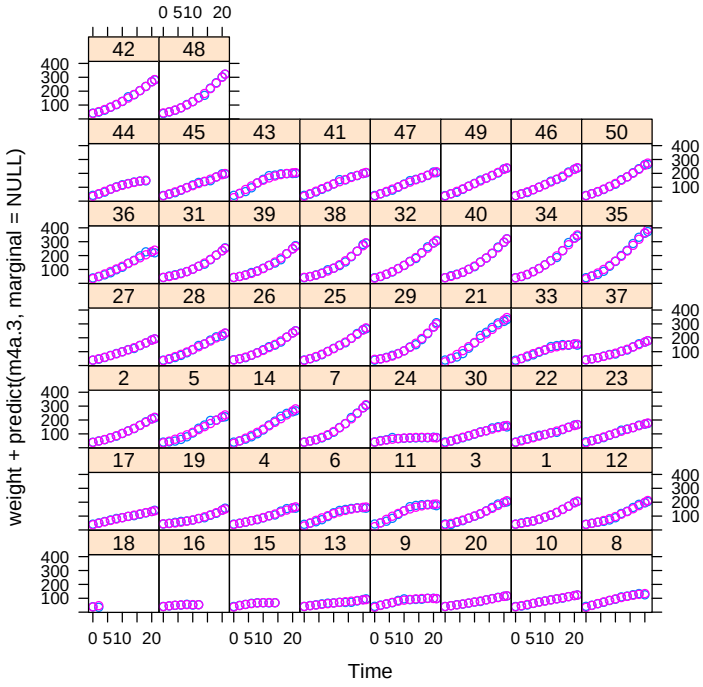

Figure 4.6: Weights of each chick as a function of age in blue, with the predicted weights in purple. A quadratic population growth curve was fitted with a second order random regression for chicks (i.e. a random intercept-slope-quadratic model).

```
> m4a.2$DIC

[1] 4544.509

> m4a.3$DIC

[1] 3932.327
```

It is worth seeing whether using an AIC measure using REML also suggests the highest order model is the better model.

```
> library(lme4, warn.conflicts = FALSE)
> m5a.1.REML <- lmer(weight ~ Diet + poly(Time,
+     2, raw = TRUE) + (1 | Chick), data = ChickWeight)
> AIC(m5a.1.REML)

[1] 5578.963
```

```
> m5a.2.REML <- lmer(weight ~ Diet + poly(Time,
+     2, raw = TRUE) + (poly(Time, 1, raw = TRUE) |
+     Chick), data = ChickWeight)
> AIC(m5a.2.REML)

[1] 4732.387

> m5a.3.REML <- lmer(weight ~ Diet + poly(Time,
+     2, raw = TRUE) + (poly(Time, 2, raw = TRUE) |
+     Chick), data = ChickWeight)
> AIC(m5a.3.REML)

[1] 4267.013

> detach(package:lme4)
```

4.2 Expected Variances and Covariances

Random regression models make strong assumptions about how the variance should change as a function of the predictor variable. Imagine that the intercept variance was zero, such that all regressions give the same prediction when `Time`=0. Imagine also that there was variance for slope, the predictions would look something like this (Figure 4.7):

```
> slope <- rnorm(30)
> plot(0, type = "n", xlim = c(-1, 1), ylim = c(-3,
+     3), ylab = "y", xlab = "Time")
> for (i in 1:30) {
+     lines(c(-1, 1), c(-slope[i], slope[i]))
+ }
```

with the variance increasing at extreme values of `Time` and being zero at `Time`=0. For an intercept-slope model such as this the expected variance is quadratic in the predictor, and for a intercept-slope-quadratic model the variance is cubic in the predictor. Generally the expected variance can be obtained using:

$$VAR[y] = \text{diag}(\mathbf{ZVZ}^{'})$$

and we can use this to predict the change in variance as a function of `Time` for the three models:

```
> pos.time <- seq(0, 21, length = 100)
> polynomial <- cbind(rep(1, 100), poly(pos.time,
+     2, raw = TRUE))
> beta.1 <- c(posterior.mode(m4a.1$Sol[, 1]), posterior.mode(m4a.1$Sol[,
+     5]), posterior.mode(m4a.1$Sol[, 6]))
```

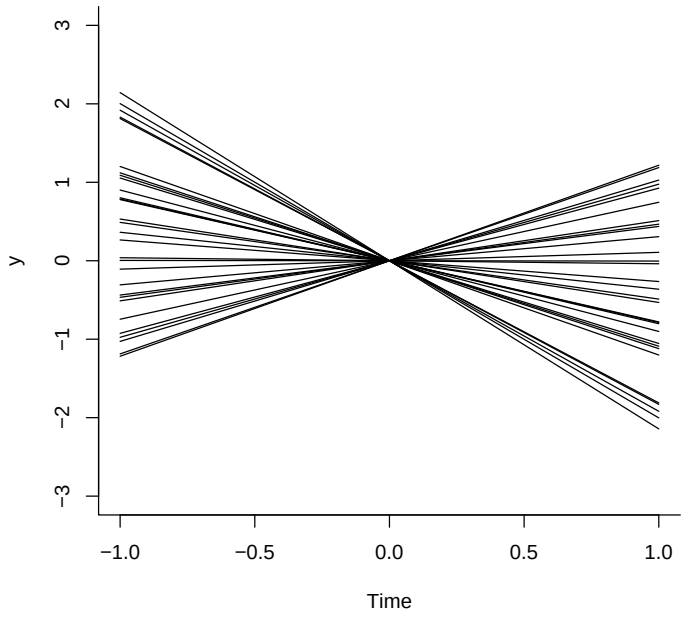

Figure 4.7: Hypothetical regression lines where the variance in slopes is one but the variance in intercepts is zero. The expected variance of y is a quadratic function of Time, being zero when Time=0, and increasing with positive or negative values.

```
> beta.2 <- c(posterior.mode(m4a.2$Sol[, 1]), posterior.mode(m4a.2$Sol[,
+     5]), posterior.mode(m4a.2$Sol[, 6]))
> beta.3 <- c(posterior.mode(m4a.3$Sol[, 1]), posterior.mode(m4a.3$Sol[,
+     5]), posterior.mode(m4a.3$Sol[, 6]))
> VCV.1 <- matrix(posterior.mode(m4a.1$VCV)[1],
+     1, 1)
> VCV.2 <- matrix(posterior.mode(m4a.2$VCV)[1:(2^2)],
+     2, 2)
> VCV.3 <- matrix(posterior.mode(m4a.3$VCV)[1:(3^2)],
+     3, 3)
> units.1 <- posterior.mode(m4a.1$VCV)[2]
> units.2 <- posterior.mode(m4a.2$VCV)[5]
> units.3 <- posterior.mode(m4a.3$VCV)[10]

> plot(weight ~ Time, data = ChickWeight, cex.lab = 1.5)
> mu.1 <- polynomial %*% beta.1
```

```
> sd.1 <- sqrt(units.1 + diag(polynomial[, 1, drop = FALSE] %*%
+     VCV.1 %*% t(polynomial[, 1, drop = FALSE])))
> lines(mu.1 ~ pos.time, lwd = 2)
> lines(I(mu.1 + 1.96 * sd.1) ~ pos.time, lty = 2,
+     lwd = 2)
> lines(I(mu.1 - 1.96 * sd.1) ~ pos.time, lty = 2,
+     lwd = 2)
```

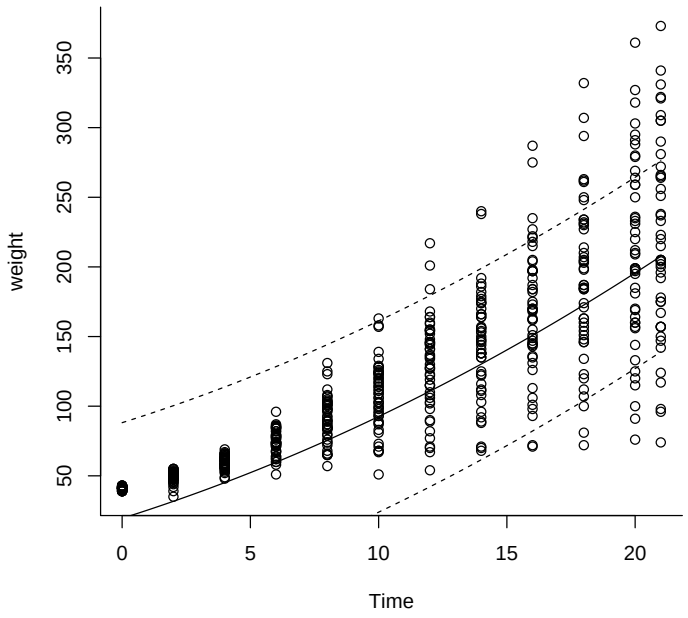

Figure 4.8: Chick weights plotted as a function of time. 95% of the data are expected to fall within the dashed lines assuming the model with random intercepts is the correct model, and the diet treatments have small effects.

The simple model, without a slope term has constant variance across the range, and is clearly inconsistent with the data (Figure 4.8). The second model on the other hand

```
> plot(weight ~ Time, data = ChickWeight, cex.lab = 1.5)
> mu.2 <- polynomial %*% beta.2
> sd.2 <- sqrt(units.2 + diag(polynomial[, 1:2,
+     drop = FALSE] %*% VCV.2 %*% t(polynomial[,
+     1:2, drop = FALSE])))
```

```
> lines(mu.2 ~ pos.time, lwd = 2)
> lines(I(mu.2 + 1.96 * sd.2) ~ pos.time, lty = 2,
+     lwd = 2)
> lines(I(mu.2 - 1.96 * sd.2) ~ pos.time, lty = 2,
+     lwd = 2)
```

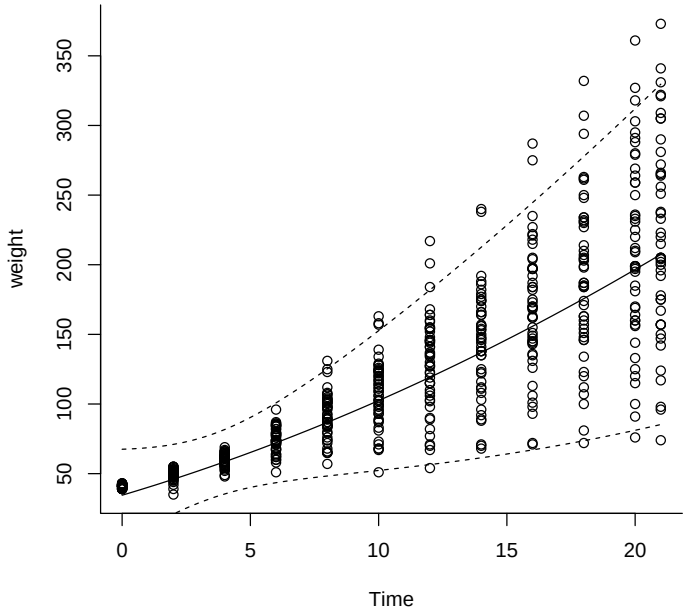

Figure 4.9: Chick weights plotted as a function of time. 95% of the data are expected to fall within the dashed lines assuming the model with random intercepts and slopes is the correct model, and the diet treatments have small effects.

has an expected variance structure reasonably close to that observed (Figure 4.9). The highest order model, which was the best using information criteria such as AIC an DIC, also does badly (Figure 4.10):

```
> plot(weight ~ Time, data = ChickWeight, ylim = c(-150,
+     600), cex.lab = 1.5)
> mu.3 <- polynomial %*% beta.3
> sd.3 <- sqrt(units.3 + diag(polynomial[, 1:3,
+     drop = FALSE] %*% VCV.3 %*% t(polynomial[,
+     1:3, drop = FALSE])))
```

```
> lines(mu.3 ~ pos.time, lwd = 2)
> lines(I(mu.3 + 1.96 * sd.3) ~ pos.time, lty = 2,
+     lwd = 2)
> lines(I(mu.3 - 1.96 * sd.3) ~ pos.time, lty = 2,
+     lwd = 2)
```

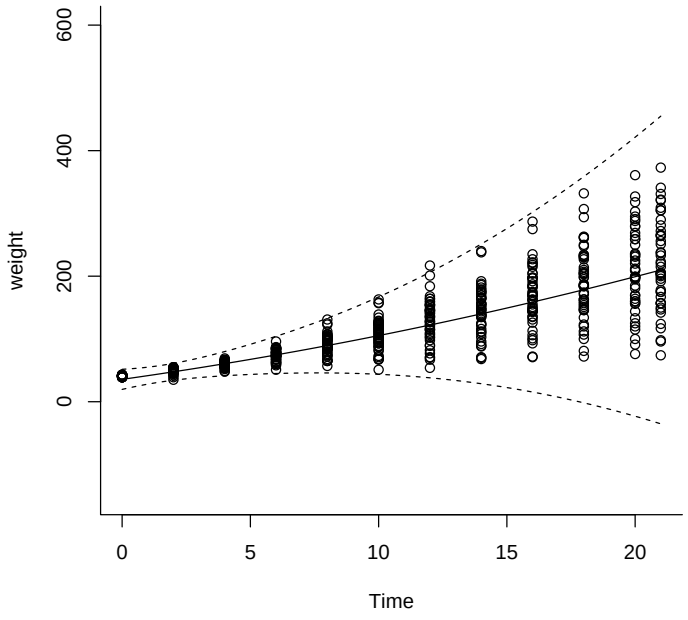

Figure 4.10: Chick weights plotted as a function of time. 95% of the data are expected to fall within the dashed lines assuming the model with random intercepts, slopes and quadratic effects is the correct model, and the diet treatments have small effects.

In general, I would not draw conclusions about changes in variance from random regression models (**?**).

4.3 `us` versus `idh` and mean centering

bad
reparametersitaion

```
> prior.m4b.1 <- list(R = list(V = 1e-07, nu = -2),
+     G = list(G1 = list(V = diag(2), nu = 2)))
> m4b.1 <- MCMCglmm(weight ~ Diet + poly(Time, 2,
+     raw = TRUE), random = ~us(1 + I(Time - 3)):Chick,
+     data = ChickWeight, verbose = FALSE, pr = TRUE,
+     prior = prior.m4b.1, saveX = TRUE, saveZ = TRUE)
```

4.4 Meta-analysis

Random intercept-slope models implicitly assume that the variance changes as a quadratic function of the predictor. This can be used to our advantage because it allows us to fit meta-analytic models. In meta-analysis the data are usually some standardised statistic which has been estimated with different levels of measurement error. If we wanted to know the expected value of these statistics we would want to weight our answer to those measurements made with the smallest amount of error. If we assume that measurement error around the true value is normally distributed then we could assume the model:

$$y_i = \beta_1 + m_i + e_i \tag{4.1}$$

where β_1 is the expected value, m_i is some deviation due to measurement error, and e_i is the deviation of the statistic from the global intercept not due to measurement error. Some types of meta-analysis presume e_i does not exist and that the only variation between studies is due to measurement error. This is not realistic, I think. Often, standard errors are reported in the literature, and these can be viewed as an approximation to the expected standard deviation of the measurement error. If we put the standard errors for each statistic as a column in the data frame (and call it `SE`) then the random term `idh(SE):units` defines a diagonal matrix with the standard errors on the diagonal. Using results from Equation 4.2

$$\begin{aligned} \mathrm{VAR}[\mathbf{m}] &= \mathbf{ZVZ}' \\ &= \mathbf{Z}\sigma^2_m\mathbf{IZ}' \\ &= \sigma^2_m\mathbf{ZZ}' \end{aligned} \tag{4.2}$$

fixing $\sigma^2_m = 1$ in the prior, the expected variance in the measurement errors are therefore the standard errors squared (the sampling variance) and all measurement errors are assumed to be independent of each other. The random regression therefore fits a random effect meta-analysis.

4.5 Splines

blah blah

```
> random = ~idv(spl(covariate))
```

fits a penalised thin-plate spline,

```
> random = ~idv(smspline(covariate))
```

fits a penalised cubic spline. The coefficients are random effects, stored in the `Sol` element of the model output and the single variance component (the penalising bit) is in the `VCV` element. Its usually a good idea to scale the covariate to lie in the interval [0,1] or some such thing.

Chapter 5

Multi-response models

So far we have only fitted models to a single response variable. Multi-response models are not that widely used, except perhaps in quantitative genetics, and deserve wider use. They allow some of the assumptions of single response models to be relaxed and can be an effective way of dealing with missing data problems.

5.1 Relaxing the univariate assumptions of causality

Imagine we knew how much money 200 people had spent on their holiday and on their car in each of four years, and we want to know whether a relationship exists between the two. A simple correlation would be one possibility, but then how do we control for the repeated measures? An often used solution to this problem is to choose one variable as the response (lets say the amount spent on a car) and have the other variable as a fixed covariate (the amount spent on a holiday). The choice is essentially arbitrary, highlighting the belief that any relationship between the two types of spending maybe in part due to unmeasured variables, rather than being completely causal.

In practice does this matter? Lets imagine there was only one unmeasured variable: disposable income. There are repeatable differences between individuals in their disposable income, but also some variation within individuals across the four years. Likewise, people vary in what proportion of their disposable income they are willing to spend on a holiday versus a car, but this also changes from year to year. We can simulate some toy data to get a feel for the issues:

```
> id<-gl(200,4)                  # 200 people recorded four times
> av_wealth<-rlnorm(200, 0, 1)
> ac_wealth<-av_wealth[id]+rlnorm(800, 0, 1)
> # expected disposable incomes + some year to year variation
>
```

```
> av_ratio<-rbeta(200,10,10)
> ac_ratio<-rbeta(800, 2*(av_ratio[id]), 2*(1-av_ratio[id]))
> # expected proportion spent on car + some year to year variation
>
> y.car<-(ac_wealth*ac_ratio)^0.25     # disposable income * proportion spent on car
> y.hol<-(ac_wealth*(1-ac_ratio))^0.25 # disposable income * proportion spent on holiday
> Spending<-data.frame(y.hol=y.hol, y.car=y.car, id=id)
```

A simple regression suggests the two types of spending are negatively related but the association is weak with the $R^2 = 0.012$.

```
> summary(lm(y.car ~ y.hol, data = Spending))

Call:
lm(formula = y.car ~ y.hol, data = Spending)

Residuals:
     Min       1Q   Median       3Q      Max
-0.88552 -0.19737 -0.00205  0.18858  1.26133

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.11702    0.03873   28.84  < 2e-16 ***
y.hol       -0.11453    0.03731   -3.07  0.00221 **
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2967 on 798 degrees of freedom
Multiple R-squared:  0.01167,        Adjusted R-squared:  0.01043
F-statistic: 9.425 on 1 and 798 DF,  p-value: 0.002213
```

With `id` added as a random term to deal with the the repeated measures, a similar conclusion is reached although the estimate is more negative:

```
> m5a.1 <- MCMCglmm(y.car ~ y.hol, random = ~id,
+     data = Spending, verbose = FALSE)
> summary(m5a.1$Sol[, "y.hol"])

Iterations = 3001:12991
Thinning interval = 10
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
   -0.198854       0.037917       0.001199       0.001199

2. Quantiles for each variable:

   2.5%     25%     50%     75%   97.5%
-0.2772 -0.2236 -0.1986 -0.1757 -0.1268
```

We may be inclined to stop there, but lets proceed with a multi-response model of the problem. The two responses are passed as a matrix using `cbind()`, and the rows of this matrix are indexed by the reserved variable `units`, and the columns by the reserved variable `trait`.

It is useful to think of a new data frame where the response variables have been stacked column-wise and the other predictors duplicated accordingly. Below is the original data frame on the left (`Spending`) and the stacked data frame on the right:

	y.hol	y.car	id
1	1.063890	0.191558	1
2	1.135753	0.724603	1
⋮	⋮	⋮	
800	0.983501	1.310201	200

$\Longrightarrow$

	y	trait	id	units
1	1.063890	y.hol	1	1
2	1.135753	y.hol	1	2
⋮	⋮	⋮	⋮	
800	0.983501	y.hol	200	800
801	0.191558	y.car	1	1
802	0.724603	y.car	1	2
⋮	⋮	⋮	⋮	
1600	1.310201	y.car	200	800

From this we can see that fitting a multi-response model is a direct extension to how we fitted models with categorical random interactions (Chapter 3):

```
> m5a.2 <- MCMCglmm(cbind(y.hol, y.car) ~ trait -
+     1, random = ~us(trait):id, rcov = ~us(trait):units,
+     data = Spending, family = c("gaussian", "gaussian"),
+     verbose = FALSE)
```

We have fitted the fixed effect `trait` so that the two types of spending can have different intercepts. I usually suppress the intercept (`-1`) for these types of models so the second coefficient is not the difference between the intercept for the first level of `trait` (`y.hol`) and the second level (`y.car`) but the actual trait specific intercepts. In other words the design matrix for the fixed effects has the form:

$$\begin{bmatrix} \texttt{trait[1]=="y.hol"} & \texttt{trait[1]=="y.car"} \\ \texttt{trait[2]=="y.hol"} & \texttt{trait[2]=="y.car"} \\ \vdots & \vdots \\ \texttt{trait[800]=="y.hol"} & \texttt{trait[800]=="y.car"} \\ \texttt{trait[801]=="y.hol"} & \texttt{trait[801]=="y.car"} \\ \texttt{trait[802]=="y.hol"} & \texttt{trait[802]=="y.car"} \\ \vdots & \vdots \\ \texttt{trait[1600]=="y.hol"} & \texttt{trait[1600]=="y.car"} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix}$$

A 2×2 covariance matrix is estimated for the random term where the diagonal elements are the variance in consistent individual effects for each type of spending. The off-diagonal is the covariance between these effects which if positive suggests that people that consistently spend more on their holidays consistently spend more on their cars. A 2×2 residual covariance matrix is also fitted. In Section 3.4 we fitted heterogeneous error models using `idh():units` which made sense in this case because each level of `unit` was specific to a particular datum and so any covariances could not be estimated. In multi-response models this is not the case because both traits have often been measured on the same observational unit and so the covariance can be measured. In the context of this example a positive covariance would indicate that in those years an individual spent a lot on their car they also spent a lot on their holiday.

A univariate regression is defined as the covariance between the response and the predictor divided by the variance in the predictor. We can therefore estimate a regression coefficient for these two levels of random variation, and compare them with the regression coefficient we obtained in the simpler model:

```
> id.regression <- m5a.2$VCV[, 2]/m5a.2$VCV[, 1]
> units.regression <- m5a.2$VCV[, 6]/m5a.2$VCV[,
+     5]
> plot(mcmc.list(m5a.1$Sol[, "y.hol"], id.regression,
+     units.regression), density = FALSE)
```

The regression coefficients (see Figure 5.1) differ substantially at the within individual (green) and between individual (red) levels, and neither is entirely consistent with the regression coefficient from the univariate model (black). The process by which we generated the data gives rise to this phenomenon - large variation between individuals in their disposable income means that people who are able to spend a lot on their holiday can also afford to spend a lot on their holidays (hence positive covariation between `id` effects). However, a person that spent a large proportion of their disposable income in a particular year on a holiday, must have less to spend that year on a car (hence negative residual (within year) covariation).

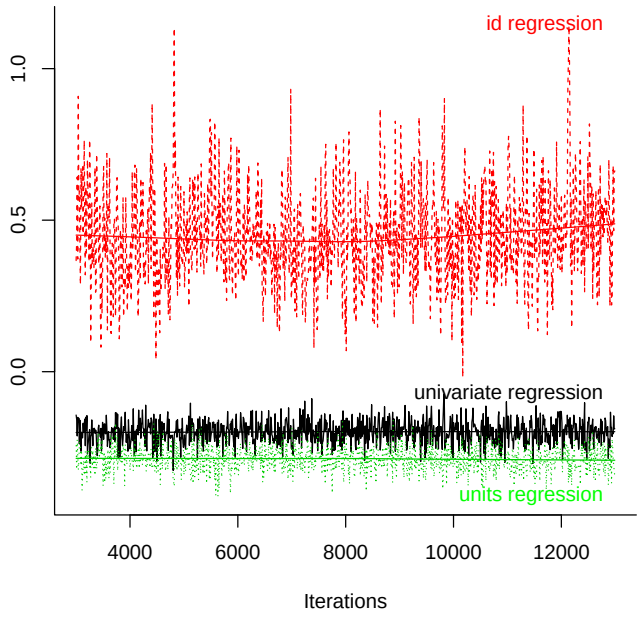

Figure 5.1: MCMC summary plot of the coefficient from a regression of car spending on holiday spending in black. The red and green traces are from a model where the regression coefficient is estimated at two levels: within an individual (green) and across individuals (red). The relationship between the two types of spending is in part mediating by a third unmeasured variable, disposable income.

When fitting the simpler univariate model we make the assumption that the effect of spending money on a car directly effects how much you spend on a holiday. If this relationship was purely causal then all regression coefficients would have the same expectation, and the simpler model would be justified.

For example, we could set up a simpler model where two thirds of the variation in holiday expenditure is due to between individual differences, and holiday expenditure directly affects how much an individual will spend on their car (using a regression coefficient of -0.3). The variation in car expenditure not caused by holiday expenditure is also due to individual differences, but in this case they only explain a third of the variance.

```
> Spending$y.hol2 <- rnorm(200, 0, sqrt(2))[Spending$id] +
+     rnorm(800, 0, sqrt(1))
```

```
> Spending$y.car2 <- Spending$y.hol2 * -0.3 + rnorm(200,
+     0, sqrt(1))[Spending$id] + rnorm(800, 0, sqrt(2))
```

We can fit the univariate and multivariate models to these data, and compare the regression coefficients as we did before. Figure 5.2 shows that the regression coefficients are all very similar and a value of -0.3 has a reasonably high posterior probability. However, it should be noted that the posterior standard deviation is smaller in the simpler model because the more strict assumptions have allowed us to pool information across the two levels to get a more precise answer.

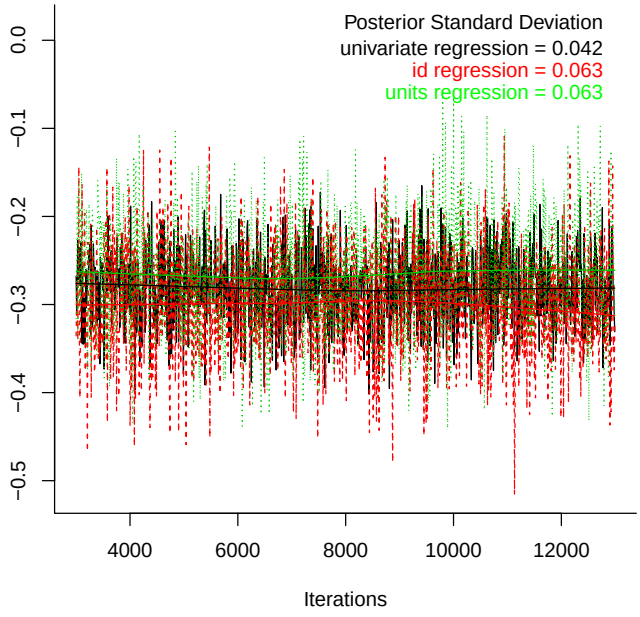

Figure 5.2: MCMC summary plot of the coefficient from a regression of car spending on holiday spending in black. The red and green traces are from a model where the regression coefficient is estimated at two levels: within an individual (green) and across individuals (red). In this model the relationship between the two types of spending is causal and the regression coefficients have the same expectation. However, the posterior standard deviation from the simple regression is smaller because information from the two different levels is pooled.

5.2 Multinomial Models

Multinomial models are difficult - both to fit and interpret. This is particularly true when each unit of observation only has a single realisation from the multinomial. In these instances the data can be expressed as a single vector of factors, and the family argument can be specified as `categorical`. To illustrate, using a very simple example, we'll use data collected on 666 Soay sheep from the island of Hirta in the St. Kilda archipelago (**?**, Table A2.5).

```
> data(SShorns)
> head(SShorns)

  id    horn    sex
1  1 scurred female
2  2 scurred female
3  3 scurred female
4  4 scurred female
5  5  polled female
6  6  polled female
```

The sex and horn morph were recorded for each individual, giving the contingency table:

```
> Ctable <- table(SShorns$horn, SShorns$sex)
> Ctable

         female male
  normal      83  352
  polled      65    0
  scurred     96   70
```

and we'll see if the frequencies of the three `horn` types differ, and if the trait is sex dependent. The usual way to do this would be to use a Chi square test, and to address the first question we could add the counts of the two sexes:

```
> chisq.test(rowSums(Ctable))

        Chi-squared test for given probabilities

data:  rowSums(Ctable)
X-squared = 329.52, df = 2, p-value < 2.2e-16
```

which strongly suggests the three morphs differ in frequency. We could then ask whether the frequencies differ by sex:

```
> chisq.test(Ctable)
```

```
        Pearson's Chi-squared test

data:  Ctable
X-squared = 202.3, df = 2, p-value < 2.2e-16
```

which again they do, which is not that surprising since the trait is partly sex limited, with males not expressing the polled phenotype.

If there were only two horn types, polled and normal for example, then we could have considered transforming the data into the binary variable *polled or not?* and analysing using a glm with sex as a predictor. In doing this we have reduced the dimension of the data from $J = 2$ categories to a single $(J - 1 = 1)$ contrast. The motivation for the dimension reduction is obvious; if being a male increased the probability of expressing normal horns by 10%, it must by necessity reduce the probability of expressing polled horn type by 10%, because an individual cannot express both horn types simultaneously. The dimension reduction essentially constrains the probability of expressing either horn type to unity:

$$Pr(\texttt{horn[i]} = \text{normal}) + Pr(\texttt{horn[i]} = \text{polled}) = 1 \tag{5.1}$$

These concepts can be directly translated into situations with more than two categories where the unit sum constraint has the general form:

$$\sum_{k=1}^{J} Pr(y_i = k) = 1 \tag{5.2}$$

For binary data we designated one category to be the success (polled) and one category to be the failure (normal) which we will call the baseline category. The latent variable in this case was the log-odds ratio of succeeding versus failing:

$$l_i = \log\left(\frac{Pr(\texttt{horn[i]} = \text{polled})}{Pr(\texttt{horn[i]} = \text{normal})}\right) = \text{logit}\left(Pr(\texttt{horn[i]} = \text{polled})\right) \tag{5.3}$$

With more than two categories we need to have $J - 1$ latent variables, which in the original horn type example are:

$$l_{i,\text{polled}} = \log\left(\frac{Pr(\texttt{horn[i]} = \text{polled})}{Pr(\texttt{horn[i]} = \text{normal})}\right) \tag{5.4}$$

and

$$l_{i,\text{scurred}} = \log\left(\frac{Pr(\texttt{horn[i]} = \text{scurred})}{Pr(\texttt{horn[i]} = \text{normal})}\right) \tag{5.5}$$

The two latent variables are indexed as `trait`, and the unit of observation (i) as `units`, as in multi-response models. As with binary models the residual variance is not identified, and can be set to any arbitrary value. For reasons that will become clearer later I like to work with the residual covariance matrix $\frac{1}{J}(\mathbf{I} + \mathbf{J})$ where $\mathbf{I}$ and $\mathbf{J}$ are $J - 1$ dimensional identity and unit matrices, respectively.

To start we will try a simple model with an intercept:

```
> IJ <- (1/3) * (diag(2) + matrix(1, 2, 2))
> prior = list(R = list(V = IJ, fix = 1))
> m5c.1 <- MCMCglmm(horn ~ trait - 1, rcov = ~us(trait):units,
+     prior = prior, data = SShorns, family = "categorical",
+     verbose = FALSE)
```

The posterior distribution for the intercepts is shown in Figure 5.3, and the model clearly needs to be run for longer (Figure 5.3). However...

The problem can also be represented using the contrast matrix $\mathbf{\Delta}$ (**?**):

$$\mathbf{\Delta} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.6}$$

where the rows correspond to the factor levels (`normal`, `polled` and `scurred`) and the columns to the two latent variables. For example column one corresponds to $l_{i,\text{polled}}$ which on the log scale is $Pr(\texttt{horn[i]} = \text{polled}) - Pr(\texttt{horn[i]} = \text{normal})$.

$$\exp\left((\mathbf{\Delta}\mathbf{\Delta}^{'})^{-1}\mathbf{\Delta}\mathbf{l}_i\right) \propto E \begin{bmatrix} Pr(\texttt{horn[i]} = \text{normal}) \\ Pr(\texttt{horn[i]} = \text{polled}) \\ Pr(\texttt{horn[i]} = \text{scurred}) \end{bmatrix} \tag{5.7}$$

The residual and any random effect covariance matrices are for estimability purposes estimated on the $J - 1$ space with $\boldsymbol{V} = \mathbf{\Delta}^{'}\tilde{\mathbf{V}}\mathbf{\Delta}$ where $\tilde{\mathbf{V}}$ is the covariance matrix estimated on the $J - 1$ space. To illustrate, we will rescale the intercepts as if the residual covariance matrix was zero (see Sections and) and predict the expected probability for each horn type:

```
> Delta <- cbind(c(-1, 1, 0), c(-1, 0, 1))
> c2 <- (16 * sqrt(3)/(15 * pi))^2
> D <- ginv(Delta %*% t(Delta)) %*% Delta
> Int <- t(apply(m5c.1$Sol, 1, function(x) {
+     D %*% (x/sqrt(1 + c2 * diag(IJ)))
+ }))
> summary(mcmc(exp(Int)/rowSums(exp(Int))))
```

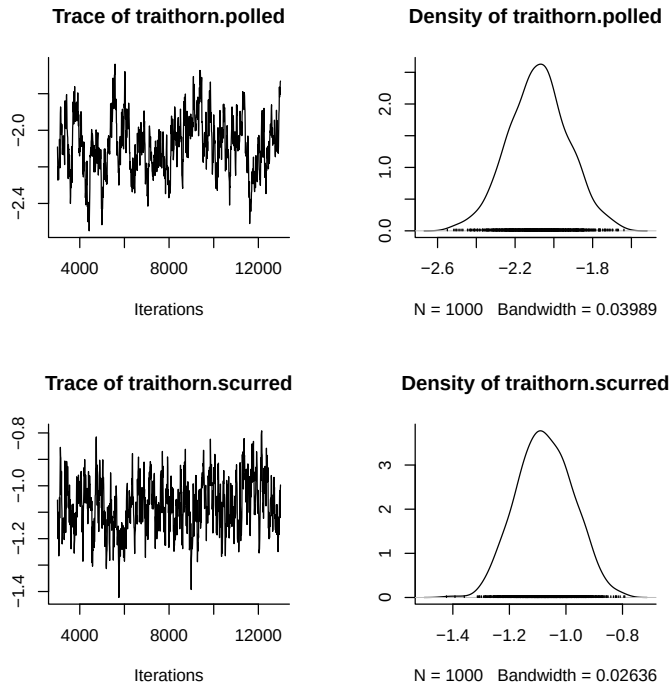

Figure 5.3: Posterior distribution of fixed effects from model `m5c.1`: a simple multinomial logit model with intercepts only

```
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
[1,] 0.6515 0.01797 0.0005683       0.001566
[2,] 0.1007 0.01246 0.0003941       0.001880
[3,] 0.2478 0.01654 0.0005229       0.001633

2. Quantiles for each variable:

        2.5%    25%    50%    75%  97.5%
var1 0.61605 0.6396 0.6521 0.6640 0.6862
```

```
var2 0.07762 0.0921 0.1002 0.1083 0.1289
var3 0.21775 0.2363 0.2473 0.2588 0.2803
```

which agrees well with those observed:

```
> prop.table(rowSums(Ctable))

   normal    polled   scurred
0.6531532 0.0975976 0.2492492
```

To test for the effects of sex specific expression we can also fit a model with a sex effect:

```
> m5c.2 <- MCMCglmm(horn ~ trait + sex - 1, rcov = ~us(trait):units,
+     data = SShorns, family = "categorical", prior = prior,
+     verbose = FALSE)
```

In this case we have not interacted sex with trait, and so we are estimating the difference between the sexes in their expression of normal and polled+scurred jointly. The posterior distribution is plotted in Figure 5.4 and clearly shows that males are more likely to express the normal horn phenotype than females.

A more general model would be to estimate separate probabilities for each cell, but the contingency table indicates that one cell (polled males) has zero counts which will cause extreme separation problems. We could choose to have a better prior for the fixed effects, that is close to being flat for the two-way (i.e. polled vs scurred, normal vs.scurred & polled vs. normal) marginal probabilities within each sex:

```
> prior$B = list(mu = rep(0, 4), V = kronecker(IJ,
+     diag(2)) * (1.7 + pi^2/3))
> m5c.3 <- MCMCglmm(horn ~ at.level(sex, 1):trait +
+     at.level(sex, 2):trait - 1, rcov = ~us(trait):units,
+     data = SShorns, family = "categorical", prior = prior,
+     verbose = FALSE)
```

The female specific probabilities appear reasonable:

```
> Int <- t(apply(m5c.3$Sol[, 1:2], 1, function(x) {
+     D %*% (x/sqrt(1 + c2 * diag(IJ)))
+ }))
> summary(mcmc(exp(Int)/rowSums(exp(Int))))

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

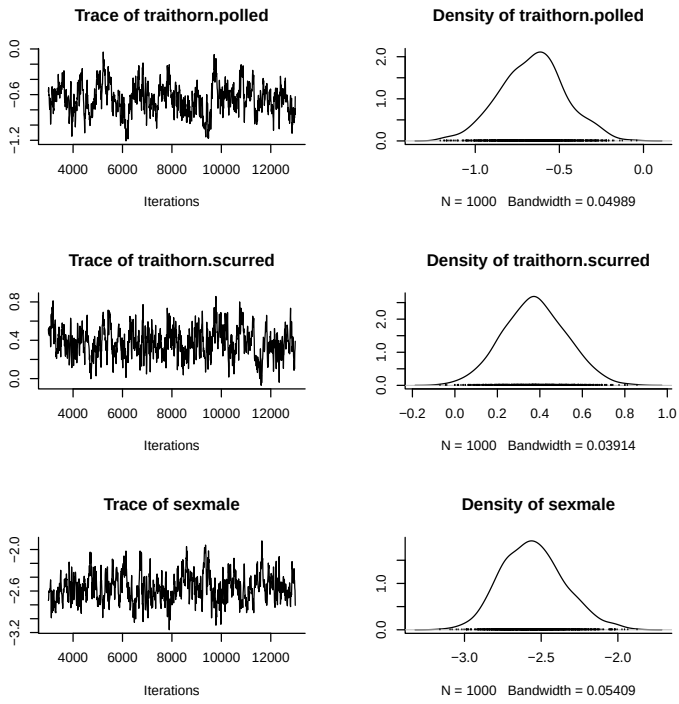

Figure 5.4: Posterior distribution of fixed effects from model `m5c.2` in which a main effect of sex was included

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

      Mean      SD  Naive SE Time-series SE
[1,] 0.3480 0.02921 0.0009237       0.002742
[2,] 0.2594 0.02759 0.0008724       0.002239
[3,] 0.3926 0.02952 0.0009334       0.003481

2. Quantiles for each variable:

      2.5%    25%    50%    75%  97.5%
var1 0.2897 0.3279 0.3477 0.3677 0.4084
var2 0.2099 0.2403 0.2584 0.2775 0.3159
var3 0.3365 0.3731 0.3930 0.4126 0.4488
```

compared to the observed frequencies:

```
> prop.table(Ctable[, 1])
```

```
   normal    polled   scurred
0.3401639 0.2663934 0.3934426
```

as do the male probabilities:

```
> Int <- t(apply(cbind(m5c.3$Sol[, 3:4]), 1, function(x) {
+     D %*% (x/sqrt(1 + c2 * diag(IJ)))
+ }))
> summary(mcmc(exp(Int)/rowSums(exp(Int))))

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
[1,] 0.82402 0.017006 0.0005378       0.001926
[2,] 0.00596 0.002979 0.0000942       0.001080
[3,] 0.17002 0.016710 0.0005284       0.001913

2. Quantiles for each variable:

         2.5%      25%     50%      75%  97.5%
var1 0.790824 0.813142 0.82304 0.834855 0.8598
var2 0.001774 0.003841 0.00575 0.007553 0.0138
var3 0.135724 0.158641 0.17109 0.181380 0.2021
```

compared to the observed frequencies:

```
> prop.table(Ctable[, 2])

   normal    polled   scurred
0.8341232 0.0000000 0.1658768
```

5.3 Zero-inflated Models

Each datum in a zero-inflated model is associated with two latent variables. The first latent variable is associated with the named distribution and the second latent variable is associated with zero inflation. I'll work through a zero-inflated Poisson (ZIP) model to make things clearer. As the name suggests, a ZIP distribution is a Poisson distribution with extra zero's. The observed zeros are modelled as a mixture distribution of zero's originating form the Poisson process and zero's arising through zero-inflation. It is the probability (on the

logit scale) that a zero is from the zero-inflation process that we aim to model with the second latent variable. The likelihood has the form:

$$\begin{aligned} Pr(y=0) = &\quad \texttt{plogis}(l_2) + \texttt{plogis}(-l_2) * \texttt{dpois}(0, \texttt{exp}(l_1)) \\ Pr(y|y>0) = &\quad \texttt{plogis}(-l_2) * \texttt{dpois}(y, \texttt{exp}(l_1)) \end{aligned} \tag{5.8}$$

pscl fits zero-inflated models very well through the `zeroinfl` function, and I strongly recommend using it if you do not want to fit random effects. To illustrate the syntax for fitting ZIP models in MCMCglmm I will take one of their examples:

```
> data("bioChemists", package = "pscl")
> head(bioChemists)
```

```
  art   fem     mar kid5  phd ment
1   0   Men Married    0 2.52    7
2   0 Women  Single    0 2.05    6
3   0 Women  Single    0 3.75    6
4   0   Men Married    1 1.18    3
5   0 Women  Single    0 3.75   26
6   0 Women Married    2 3.59    2
```

`art` is the response variable - the number of papers published by a Ph.D student - and the remaining variables are to be fitted as fixed effects. Naively, we may expect zero-inflation to be a problem given 30% of the data are zeros, and based on the global mean we only expect around 18%.

```
> table(bioChemists$art == 0)
```

```
FALSE  TRUE
  640   275
```

```
> ppois(0, mean(bioChemists$art))
```

```
[1] 0.1839859
```

As with binary models we do not observe any residual variance for the zero-inflated process, and in addition the residual covariance between the zero-inflation and the Poisson process cannot be estimated because both processes cannot be observed in a single data point. To deal with this I've fixed the residual variance for the zero-inflation at 1, and the covariance is set to zero using the idh structure. Setting `V=diag(2)` and `nu=0.002`[1] we have the inverse-gamma prior with `shape=scale=0.001` for the residual component of the Poisson process which captures over-dispersion:

[1] Earlier versions of the CourseNotes had `nu=1.002`. In versions <2.05 the marginal prior of a variance associated with an `idh` structure was inverse-Wishart with $\texttt{nu}^* = \texttt{nu} - 1$ where $\texttt{nu}^*$ is the marginal degree of belief. In versions >=2.05 I changed this so that $\texttt{nu}^* = \texttt{nu}$ as it was leading to confusion.

```
> prior.m5d.1 = list(R = list(V = diag(2), nu = 0.002,
+     fix = 2))
> m5d.1 <- MCMCglmm(art ~ trait - 1 + at.level(trait,
+     1):fem + at.level(trait, 1):mar + at.level(trait,
+     1):kid5 + at.level(trait, 1):phd + at.level(trait,
+     1):ment, rcov = ~idh(trait):units, data = bioChemists,
+     prior = prior.m5d.1, family = "zipoisson",
+     verbose = FALSE)
```

As is often the case the parameters of the zero-inflation model mixes poorly (See Figure 5.5) especially when compared to equivalent hurdle models (See Section 5.4). Poor mixing is often associated with distributions that may *not* be zero-inflated but instead over-dispersed.

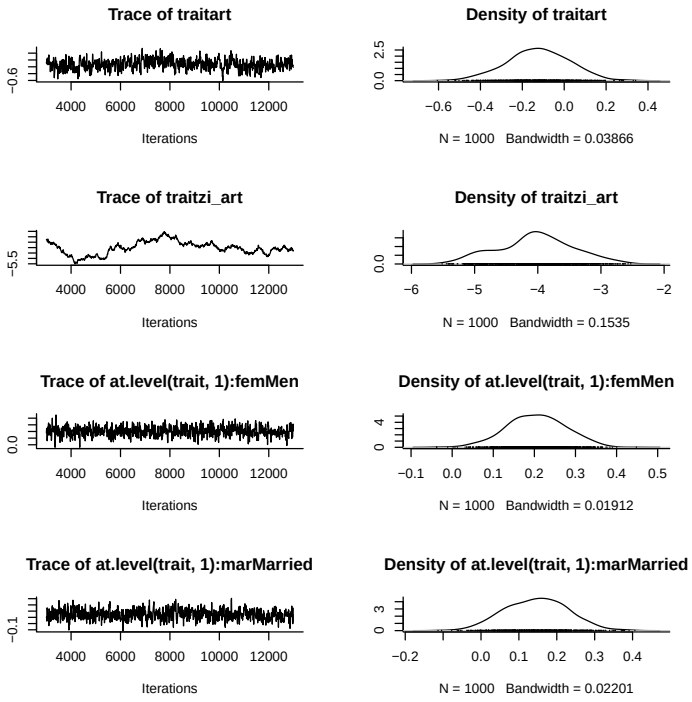

Figure 5.5: Posterior distribution of fixed effects from model `m5d.1` in which trait 1 (`art`) is the Poisson process and trait 2 (`zi.art`) is the zero-inflation.

The model would have to be run for (much) longer to say something concrete about the level of zero-inflation but my guess would be it's not a big issue, given the probability is probably quite small:

```
> quantile(plogis(m5d.1$Sol[, 2]/sqrt(1 + c2)))

         0%         25%         50%         75%        100%
0.008547321 0.021516547 0.030095780 0.041048332 0.101711517
```

5.3.1 Posterior predictive checks

Another useful check is to fit the standard Poisson model and use posterior predictive checks to see how many zero's you would expect under the simple model:

```
> prior.m5d.2 = list(R = list(V = diag(1), nu = 0.002))
> m5d.2 <- MCMCglmm(art ~ fem + mar + kid5 + phd +
+     ment, data = bioChemists, prior = prior.m5d.2,
+     family = "poisson", saveX = TRUE, verbose = FALSE)
> nz <- 1:1000
> oz <- sum(bioChemists$art == 0)
> for (i in 1:1000) {
+     pred.l <- rnorm(915, (m5d.2$X %*% m5d.2$Sol[i,
+         ])@x, sqrt(m5d.2$VCV[i]))
+     nz[i] <- sum(rpois(915, exp(pred.l)) == 0)
+ }
```

Figure 5.6 shows a histogram of the posterior predictive distribution of zero's (`nz`) from the model compared to the observed number of zeros (`oz`). The simpler model seems to be consistent with the data, suggesting that a ZIP model may not be required.

5.4 Hurdle Models

Hurdle models are very similar to zero-inflated models but they can be used to model zero-deflation as well as zero-inflation and seem to have much better mixing properties in `MCMCglmm`. As in ZIP models each datum in the hurdle model is associated with two latent variables. However, whereas in a ZIP model the first latent variable is the mean parameter of a Poisson distribution the equivalent latent variable in the hurdle model is the mean parameter of a zero-truncated Possion distribution (i.e. a Poisson distribution without the zeros observed). In addition the second latent variable in a ZIP model is the probability that an observed zero is due to zero-inflation rather than the Poisson process. In hurdle models the second latent variable is simply the probability (on the logit scale) that the response variable is zero or not. The likelihood is:

$$\begin{aligned} Pr(y=0) = &\quad \texttt{plogis}(l_2) \\ Pr(y|y>0) = &\quad \texttt{plogis}(-l_2) * \texttt{dpois}(y, \texttt{exp}(l_1))/(1-\texttt{ppois}(0, \texttt{exp}(l_1))) \end{aligned} \tag{5.9}$$

To illustrate, we will refit the ZIP model (`m5d.1`) as a hurdle-Poisson model.

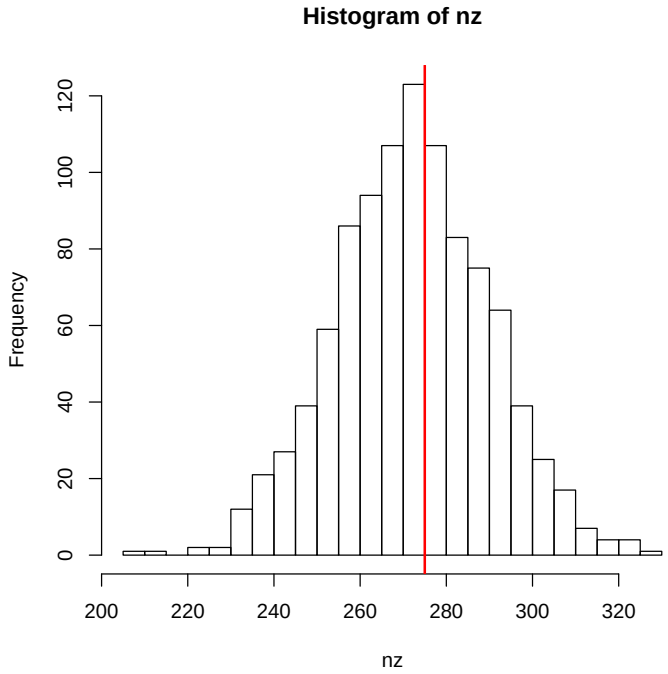

Figure 5.6: Posterior predictive distribution of zeros from model `m5d.2` with the observed number in red.

```
> m5d.3 <- MCMCglmm(art ~ trait - 1 + at.level(trait,
+     1):fem + at.level(trait, 1):mar + at.level(trait,
+     1):kid5 + at.level(trait, 1):phd + at.level(trait,
+     1):ment, rcov = ~idh(trait):units, data = bioChemists,
+     prior = prior.m5d.1, family = "hupoisson",
+     verbose = FALSE)
```

Plotting the Markov chain for the equivalent parameters that were plotted for the ZIP model shows that the mixing properties are much better (compare Figure 5.5 with Figure 5.7).

The interpretation of the model is slightly different. Fitting just an intercept in the hurdle model implies that the proportion of zeros observed across different combinations of those fixed effects fitted for the Poisson process is constant. Our 95% credible intervals for this proportion is (See section 5.2):

```
> c2 <- (16 * sqrt(3)/(15 * pi))^2
> HPDinterval(plogis(m5d.3$Sol[, 2]/sqrt(1 + c2)))
```

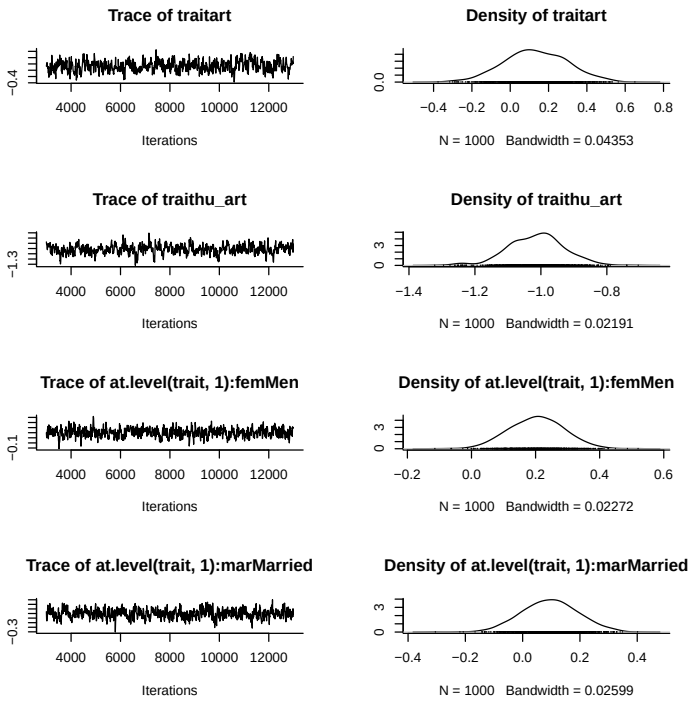

Figure 5.7: Posterior distribution of fixed effects from model `m5d.3` in which trait 1 (`art`) is the zero-truncated Poisson process and trait 2 (`hu.art`) is the binary trait zero or non-zero.

```
        lower     upper
var1 0.2682566 0.3260789
attr(,"Probability")
[1] 0.95
```

and we can compare this to the predicted number of zero's from the Poisson process if it had not been zero-truncated:

```
> HPDinterval(ppois(0, exp(m5d.3$Sol[, 1] + 0.5 *
+     m5d.3$VCV[, 1])))

        lower     upper
var1 0.1504119 0.3605811
attr(,"Probability")
[1] 0.95
```

The credible intervals largely overlap, strongly suggesting a standard Poisson model would be adequate. However, our prediction for the number of zero's that

would arise form a non-truncated Poisson process only involved the intercept term. This prediction therefore pertains to the number of articles published by single women with no young children who obtained their Ph.D's from departments scoring zero for prestige (`phd`) and whose mentors had published nothing in the previous 3 years. Our equivalent prediction for men is a little lower

```
> HPDinterval(ppois(0, exp(m5d.3$Sol[, 1] + m5d.3$Sol[,
+     3] + 0.5 * m5d.3$VCV[, 1])))

         lower    upper
var1 0.0903996 0.289349
attr(,"Probability")
[1] 0.95
```

suggesting that perhaps the number of zero's is greater than we expected for this group. However, this may just be a consequence of us fixing the proportion of zero's to be constant across these groups. We can relax this assumption by fitting a separate term for the proportion of zeros for men:

```
> m5d.4 <- MCMCglmm(art ~ trait - 1 + at.level(trait,
+     1:2):fem + at.level(trait, 1):mar + at.level(trait,
+     1):kid5 + at.level(trait, 1):phd + at.level(trait,
+     1):ment, rcov = ~idh(trait):units, data = bioChemists,
+     prior = prior.m5d.1, family = "hupoisson",
+     verbose = FALSE)
```

which reveals that although this proportion is expected to be (slightly) smaller:

```
> HPDinterval(plogis((m5d.4$Sol[, 2] + m5d.4$Sol[,
+     4])/sqrt(1 + c2)))

         lower     upper
var1 0.2300976 0.3060856
attr(,"Probability")
[1] 0.95
```

the proportion of zeros expected for men is probably still less than what we expect from a non-truncated Poisson process for which the estimates have changed very little:

```
> HPDinterval(ppois(0, exp(m5d.4$Sol[, 1] + m5d.4$Sol[,
+     3] + 0.5 * m5d.4$VCV[, 1])))

          lower     upper
var1 0.07802243 0.2463355
attr(,"Probability")
[1] 0.95
```

This highlights one of the disadvantages of hurdle models. If explanatory variables have been fitted that affect the expectation of the Poisson process then this implies that the proportion of zero's observed will also vary across these same explanatory variables, even in the absence of zero-inflation. It may then be necessary to fit an equally complicated model for both processes even though a single parameter would suffice in a ZIP model. However, in the absence of zero-inflation the intercept of the zero-inflation process in a ZIP model is $-\infty$ on the logit scale causing numerical and inferential problems. An alternative type of model are zero-altered models.

5.5 Zero-altered Models

Zero-altered Poisson (ZAP) models are identical to Poisson-hurdle models except a complementary log-log link is used instead of the logit link when modeling the proportion of zeros. However for reasons that will become clearer below, the zero-altered process (`za`) is predicting non-zeros as opposed to the ZIP and hurdle-Poisson models where it is the number of zeros. The likelihood is:

$$\begin{aligned} Pr(y=0) = &\quad 1 - \texttt{pexp}(\texttt{exp}(l_2)) \\ Pr(y|y>0) = &\quad \texttt{pexp}(\texttt{exp}(l_2)) * \texttt{dpois}(y, \texttt{exp}(l_1))/(1 - \texttt{ppois}(0, \texttt{exp}(l_1))) \end{aligned} \tag{5.10}$$

since the inverse of the complementary log-log transformation is the distribution function of the extreme value (log-exponential) distribution.

It happens that $\texttt{ppois}(0, \texttt{exp}(l)) = \texttt{dpois}(0, \texttt{exp}(l)) = 1 - \texttt{pexp}(\texttt{exp}(l))$ so that if $l = l_1 = l_2$ then the likelihood reduces to:

$$\begin{aligned} Pr(y=0) = &\quad \texttt{dpois}(0, \texttt{exp}(l)) \\ Pr(y|y>0) = &\quad \texttt{dpois}(y, \texttt{exp}(l)) \end{aligned} \tag{5.11}$$

which is equivalent to a standard Poisson model.

We can then test for zero-flation by constraining the over-dispersion to be the same for both process using a `trait` by `units` interaction in the R-structure, and by setting up the contrasts so that the zero-altered regression coefficients are expressed as differences from the Poisson regression coefficients. When this difference is zero the variable causes no zero-flation, when it is negative it causes zero-inflation and when it is positive it causes zero-deflation:

```
> m5d.5 <- MCMCglmm(art ~ trait * (fem + mar + kid5 +
+     phd + ment), rcov = ~trait:units, data = bioChemists,
+     family = "zapoisson", verbose = FALSE)
> summary(m5d.5)

 Iterations = 3001:12991
 Thinning interval  = 10
```

```
 Sample size  = 1000

 DIC: 3039.935

 R-structure:  ~trait:units

           post.mean l-95% CI u-95% CI eff.samp
trait:units     0.369   0.2641   0.4736    49.25

 Location effects: art ~ trait * (fem + mar + kid5 + phd + ment)

                       post.mean  l-95% CI  u-95% CI
(Intercept)            0.320876 -0.023992  0.647538
traitza_art           -0.534159 -1.052208 -0.040673
femWomen              -0.196701 -0.371203 -0.035722
marMarried             0.097212 -0.112003  0.293646
kid5                  -0.128691 -0.240225 -0.005758
phd                    0.017548 -0.064325  0.099296
ment                   0.019478  0.012636  0.026614
traitza_art:femWomen   0.027842 -0.243108  0.302421
traitza_art:marMarried 0.138033 -0.188627  0.408963
traitza_art:kid5      -0.077758 -0.285256  0.122673
traitza_art:phd        0.013288 -0.121300  0.152226
traitza_art:ment       0.028583  0.012840  0.044232
                      eff.samp  pMCMC
(Intercept)              174.4  0.072 .
traitza_art              165.1  0.046 *
femWomen                 279.6  0.030 *
marMarried               325.4  0.340
kid5                     342.9  0.028 *
phd                      235.1  0.726
ment                     390.9 <0.001 ***
traitza_art:femWomen     199.7  0.874
traitza_art:marMarried   249.9  0.360
traitza_art:kid5         239.1  0.430
traitza_art:phd          207.9  0.842
traitza_art:ment         128.5 <0.001 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we can see from this that the more papers a mentor produces, the more zero-deflation (or conversely the less papers a mentor produces, the more zero-inflation).

Chapter 6

Pedigrees and Phylogenies

```
> library(kinship2)
```

Pedigrees and phylogenies are similar things: they are both ways of representing shared ancestry. Under a quantitative genetic model of inheritance, or a Brownian motion model of evolution, GLMM's can be readily extended to model the similarities that exist between the phenotypes of related individuals or taxa. In the context of quantitative genetics these models are known as 'animal' models (**?**), and in the context of the comparative method these models are known as phylogenetic mixed models (**?**). The two models are almost identical, and are relatively minor modifications to the basic mixed model (**?**).

6.1 Pedigree and phylogeny formats

6.1.1 Pedigrees

`MCMCglmm` handles pedigrees stored in 3-column tabular form, with each row representing a single individual. The first column should contain the unique identifier of the individual, and columns 2 and 3 should be the unique identifiers of the individual's parents. Parents must appear before their offspring in the table. I usually have the dam (mother) in the first column and the sire (father) in the third. I prefer the words dam and sire because if I subscript things with m and f I can never remember whether I mean male and female, or mother and father. In hermaphrodite systems the same individual may appear in both columns, even within the same row if an individual was produced through selfing. This is not a problem, but `MCMCglmm` will issue a warning in case hermaphrodites are not present and a data entry mistake has been made. Impossible pedigrees (for example individual's that give birth to their own mother) are a problem and `MCMCglmm` will issue an error, hopefully with an appropriate message, when impossibilities are detected.

If the parent(s) of an individual are unknown then a missing value (NA) should be assigned in the relevant column. All individuals appearing as dams or sires need to have their own record, even if both of their parents are unknown. Often the number of individuals in a pedigree will be greater than the number of individuals for which phenotypic data exist. MCMCglmm can handle this, as long as all the individuals appearing in the data frame passed to data also appear in the pedigree.

To illustrate, we can load a pedigree for a population of blue tits and display the pedigree for the nuclear family that has individuals "R187920" and "R187921" as parents:

```
> data(BTped)
> Nped <- BTped[which(apply(BTped, 1, function(x) {
+     any(x == "R187920" | x == "R187921")
+ })), ]
> Nped
```

```
      animal     dam    sire
66   R187920    <NA>    <NA>
172  R187921    <NA>    <NA>
325  R187726 R187920 R187921
411  R187724 R187920 R187921
503  R187723 R187920 R187921
838  R187613 R187920 R187921
932  R187612 R187920 R187921
1030 R187609 R187920 R187921
```

Both parents form part of what is known as the base population - they are outbred and unrelated to anybody else in the pedigree.

MCMCglmm and MasterBayes have several pedigree manipulation functions. (MasterBayes::orderPed) orders a pedigree so parents appear before their offspring, (MasterBayes::insertPed) inserts records for individuals that only appear as parents (or a vector of specified individuals). When the number of individuals with phenotypic data is less than the number of individuals in the pedigree it is sometimes possible to remove uninformative individuals from the pedigree and thus reduce the computation time. This is known as pruning the pedigree and is implemented in the MCMCglmm function prunePed. A vector of measured individuals is specified in the argument keep and specifying make.base=TRUE implements the most complete pruning. Note, make.base=FALSE is the default argument so you'll need to explicitly specify TRUE in the call to prunePed. Michael Morrissey's pedantics package, and the kinship2 package also have many other useful pedigree orientated functions. In fact, the orderPed function in MasterBayes is built around functions provided by kinship2.

6.1.2 Phylogenies

Phylogenies can be expressed in tabular form, although only two columns are required because each species only has a single parent. In general however, phylogenies are not expressed in this form presumably because it is hard to traverse phylogenies (and pedigrees) backwards in time when they are stored this way. For phylogenetic mixed models we generally only need to traverse phylogenies forward in time (if at all) but I have stuck with convention and used the `phylo` class from the `ape` package to store phylogenies. As with pedigrees, all species appearing in the data frame passed to `data` need to appear in the phylogeny. Typically, this will only include species at the tips of the phylogeny and so the measured species should appear in the `tip.label` element of the phylo object. An error message will be issued if this is not the case. Data may also exist for ancestral species, or even for species present at the tips but measured many generations before. It is possible to include these data as long as the phylogeny has labelled internal nodes. If nodes are unlabeled then `MCMCglmm` names them internally using the default arguments of `makeNodeLabel` from `ape`.

To illustrate, lets take the phylogeny of bird families included in the `ape` package, and extract the phylogeny in tabular form for the Paridae (Tits), Certhiidae (Treecreepers), Gruidae (Cranes) and the Struthionidae (Ostriches):

```
> data("bird.families")
> bird.families <- makeNodeLabel(bird.families)
> some.families <- c("Certhiidae", "Paridae", "Gruidae",
+     "Struthionidae")
> Nphylo <- drop.tip(bird.families, setdiff(bird.families$tip.label,
+     some.families))
> INphylo <- inverseA(Nphylo)
> INphylo$pedigree

     node.names
[1,] "Node58"        NA        NA
[2,] "Node122"       "Node58"  NA
[3,] "Struthionidae" NA        NA
[4,] "Gruidae"       "Node58"  NA
[5,] "Certhiidae"    "Node122" NA
[6,] "Paridae"       "Node122" NA
```

The full phylogeny, with these families and their connecting notes displayed, is shown in Figure 6.1. You will notice that `Node1` - the root - does not appear in the phylogeny in tabular form. This is because the root is equivalent to the base population in a pedigree analysis, an issue which we will come back to later. Another piece of information that seems to be lacking in the tabular form is the branch length information. Branch lengths are equivalent to inbreeding coefficients in a pedigree. As with pedigrees the inbreeding coefficients are calculated by `inverseA`:

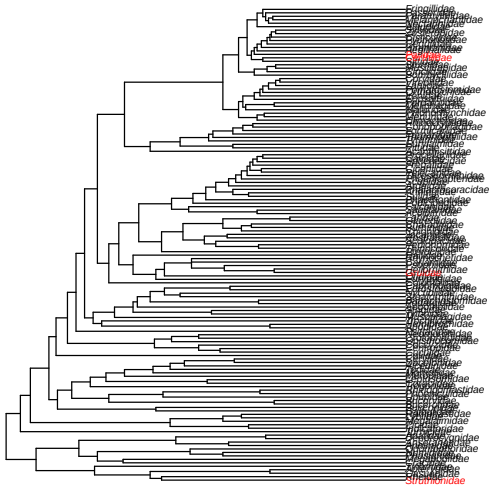

Figure 6.1: A phylogeny of bird families from **?** The families in red are the Tits (Paridae), Treecreepers (Certhiidae), Cranes (Gruidae) and the Ostriches (Struthionidae) from top to bottom. Blue tits are in the Paridae, and the word pedigree comes from the french for crane's foot.

```
> INphylo$inbreeding
```

```
[1] 0.2285714 0.3857143 1.0000000 0.7714286 0.3857143
[6] 0.3857143
```

You will notice that the Struthionidae have an inbreeding coefficient of 1 because we used the default `scale=TRUE` in the call to `inverseA`. Only ultrametric trees can be scaled in `MCMCglmm` and in this case the sum of the inbreeding coefficients connecting the root to a terminal node is one. To take the Paridae as an example:

```
> sum(INphylo$inbreeding[which(INphylo$pedigree[,
+     1] %in% c("Paridae", "Node122", "Node58"))])
```

```
[1] 1
```

The inbreeding coefficients for the members of the blue tit nuclear family are of course all zero:

```
> inverseA(Nped)$inbreeding
```

```
[1] 0 0 0 0 0 0 0 0
```

6.2 The animal model and the phylogenetic mixed model

The structure of pedigrees and phylogenies can be expressed in terms of the relatedness matrix **A**. This matrix is symmetric, square, and has dimensions equal to the number of individuals in the pedigree (or the number of taxa in the phylogeny). For pedigrees, element $A_{i,j}$ is twice the probability that an allele drawn from individual i is identical by descent to an allele in individual j. For phylogenies, element $A_{i,j}$ is the amount of time that elapsed (since the common ancestor of all sampled taxa) before the speciation event that resulted in taxa i and j. Simple, but perhaps slow, recursive methods exist for calculating **A** in both cases:

```
> Aped <- 2 * kinship2::kinship(Nped[, 1], Nped[,
+     2], Nped[, 3])
> Aped
```

```
        R187920 R187921 R187726 R187724 R187723 R187613
R187920     1.0     0.0     0.5     0.5     0.5     0.5
R187921     0.0     1.0     0.5     0.5     0.5     0.5
R187726     0.5     0.5     1.0     0.5     0.5     0.5
R187724     0.5     0.5     0.5     1.0     0.5     0.5
R187723     0.5     0.5     0.5     0.5     1.0     0.5
R187613     0.5     0.5     0.5     0.5     0.5     1.0
R187612     0.5     0.5     0.5     0.5     0.5     0.5
R187609     0.5     0.5     0.5     0.5     0.5     0.5
        R187612 R187609
R187920     0.5     0.5
R187921     0.5     0.5
R187726     0.5     0.5
R187724     0.5     0.5
R187723     0.5     0.5
R187613     0.5     0.5
R187612     1.0     0.5
R187609     0.5     1.0
```

```
> Aphylo <- vcv.phylo(Nphylo, cor = T)
> Aphylo
```

```
              Struthionidae   Gruidae Certhiidae   Paridae
Struthionidae             1 0.0000000  0.0000000 0.0000000
Gruidae                   0 1.0000000  0.2285714 0.2285714
```

```
Certhiidae              0 0.2285714  1.0000000 0.6142857
Paridae                 0 0.2285714  0.6142857 1.0000000
```

Note that specifying `cor=T` is equivalent to scaling the tree as we did in the argument to `inverseA`.

In fact, all of the mixed models we fitted in earlier sections also used an $\mathbf{A}$ matrix, but in those cases the matrix was an identity matrix (i.e. $\mathbf{A} = \mathbf{I}$) and we didn't have to worry about it. Let's reconsider the Blue tit model `m3a.1` from Section 3 where we were interested in estimating `sex` effects for `tarsus` length together with the amount of variance explained by genetic mother (`dam`) and foster mother (`fosternest`):

```
> m3a.1 <- MCMCglmm(tarsus ~ sex, random = ~dam +
+     fosternest, data = BTdata, verbose = FALSE)
```

All individuals that contributed to that analysis are from a single generation and appear in `BTped` together with their parents. However, individuals in the parental generation do not have tarsus length measurements so they do not have their own records in `BTdata`.

The model can be expressed as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_1\mathbf{u}_1 + \mathbf{Z}_2\mathbf{u}_2 + \mathbf{e} \tag{6.1}$$

where the design matrices contain information relating each individual to a `sex` ($\mathbf{X}$) a `dam` ($\mathbf{Z}_1$) and a `fosternest`($\mathbf{Z}_2$). The associated parameter vectors ($\boldsymbol{\beta}$, $\mathbf{u}_1$ and $\mathbf{u}_2$) are the effects of each `sex`, `mother` and `fosternest` on `tarsus` length, and $\mathbf{e}$ is the vector of residuals.

In the model, the u's are treated as random so we estimate their variance instead of fixing it in the prior at some (large) value, as we did with the β's. We can be a little more explicit about what this means:

$$\mathbf{u}_1 \sim N(\mathbf{0},\ \mathbf{I}\sigma^2_1) \tag{6.2}$$

where $\sim$ stands for 'is distributed as' and N a (multivariate) normal distribution. The distribution has two sets of parameters; a vector of means and a covariance matrix. We assume the random effects are deviations around the fixed effect part of the model and so they have a prior expectation of zero. The (co)variance matrix of the random effects is $\mathbf{I}\sigma^2_1$ where σ^2_1 is the variance component to be estimated. The use of the identity matrix makes two things explicit. First, because all off-diagonal elements of an identity matrix are zero we are assuming that all `dam` effects are independent (no covariance exists between any two `dam` effects). Second, all diagonal elements of an identity matrix are 1 implying that the range of possible values the dam effect *could* take is equivalent for every `dam`, this range being governed by the magnitude of the

variance component.

Since `dam`'s have very little interaction with the subset of offspring that were moved to a `fosternest`, we may be willing to assume that any similarity that exists between the tarsus lengths of this susbset and the subset that remained at home must be due to genetic effects. Although not strictly true we can assume that individuals that shared the same dam also shared the same sire, and so share around 50% of their genes.

to be completed ...

Chapter 7

Technical Details

7.1 Model Form

The probability of the i^{th} data point is represented by:

$$f_i(y_i|l_i) \tag{7.1}$$

where f_i is the probability density function associated with y_i. For example, if y_i was assumed to be Poisson distributed and we used the canonical log link function, then Equation 7.1 would have the form:

$$f_P\left(y_i|\lambda = \exp(l_i)\right) \tag{7.2}$$

where λ is the canonical parameter of the Poisson denisty function f_P. Table 7.1 has a full list of supported distributions and link functions.

The vector of latent variables follow the linear model

$$\mathbf{l} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e} \tag{7.3}$$

where $\mathbf{X}$ 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 $\mathbf{u}$, and $\mathbf{e}$ is a vector of residuals. In the Poisson case these residuals deal with any over-dispersion in the data after accounting for fixed and random sources of variation.

The location effects ($\boldsymbol{\beta}$ and $\mathbf{u}$), and the residuals ($\mathbf{e}$) are assumed to come from a multivariate normal distribution:

$$\begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\beta}_0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}\right) \tag{7.4}$$

where $\boldsymbol{\beta}_0$ is a vector of prior means for the fixed effects with prior (co)variance $\mathbf{B}$, and $\mathbf{G}$ and $\mathbf{R}$ are the expected (co)variances of the random effects and residuals respectively. The zero off-diagonal matrices imply *a priori* independence between fixed effects, random effects, and residuals. Generally, $\mathbf{G}$ and $\mathbf{R}$ are large square matrices with dimensions equal to the number of random effects or residuals. Typically they are unknown, and must be estimated from the data, usually by assuming they are structured in a way that they can be parameterised by few parameters. Below we will focus on the structure of $\mathbf{G}$, but the same logic can be applied to $\mathbf{R}$.

At its most general, **MCMCglmm** allows variance structures of the form:

$$\mathbf{G} = (\mathbf{V}_1 \otimes \mathbf{A}_1) \oplus (\mathbf{V}_2 \otimes \mathbf{A}_2) \oplus \dots \tag{7.5}$$

where the parameter (co)variance matrices ($\mathbf{V}$) are usually low-dimensional and are to be estimated, and the structured matrices ($\mathbf{A}$) are usually high dimensional and treated as known.

In the case of ordinal probit models with > 2 categories (i.e. `"threshold"` or `"ordinal"` models), f_T/f_O depends on an extra set of parameters in addition to the latent variable: the $\max(y)+1$ cutpoints $\boldsymbol{\gamma}$. The probability of y_i is then:

$$f_T(y_i|l_i, \boldsymbol{\gamma}) = 1 \text{ if } \gamma_{y_i+1} < l_i < \gamma_{y_i} \tag{7.6}$$

and

$$f_O(y_i|l_i, \boldsymbol{\gamma}) = F_N(\gamma_{y_i}|l_i, 1) - F_N(\gamma_{y_i+1}|l_i, 1) \tag{7.7}$$

where F_N is the cumulative density function for the normal. Note that the two models can be made equivalent.

7.2 MCMC Sampling Schemes

7.2.1 Updating the latent variables l

The conditional density of l is given by:

$$Pr(l_i|\mathbf{y}, \boldsymbol{\theta}, \mathbf{R}, \mathbf{G}) \propto f_i(y_i|l_i) f_N(e_i|\mathbf{r}_i\mathbf{R}_{/i}^{-1}\mathbf{e}_{/i}, r_i - \mathbf{r}_i\mathbf{R}_{/i}^{-1}\mathbf{r}_i^{'}) \tag{7.8}$$

where f_N indicates a Multivariate normal density with specified mean vector and covariance matrix. Equation 7.2.1 is the probability of the data point y_i from distribution f_i with latent varaible l_i, multiplied by the probability of the linear predictor residual. The linear predictor residual follows a conditional normal distribution where the conditioning is on the residuals associated with data points other than i. Vectors and matrices with the row and/or column associated with i removed are denoted $/i$. Three special cases exist for which we sample directly from Equation 7.2.1: i) When y_i is normal $f_i(y_i|l_i) = 1$ if

$y_i = l_i$ and zero otherwise so $l_i = y_i$ with out the need for updating, ii) when y_i is discrete and modelled using `family="threshold"` then Equation defines a truncated normal distribution and can be slice sampled (**?**) and iii) when y_i is missing $f_i(y_i|l_i)$ is not defined and samples can drawn directly from the normal.

In practice, the conditional distribution in Equation 7.2.1 only involves other residuals which are expected to show some form of residual covariation, as defined by the $\mathbf{R}$ structure. Because of this we actually update latent variables in blocks, where the block is defined as groups of residuals which are expected to be correlated:

$$Pr(\mathbf{l}_j|\mathbf{y}, \boldsymbol{\theta}, \mathbf{R}, \mathbf{G}) \propto \prod_{i \in j} p_i(y_i|l_i) f_N(\mathbf{e}_j|\mathbf{0}, \mathbf{R}_j) \tag{7.9}$$

where j indexes blocks of latent variables that have non-zero residual covariances. For response variables that are neither Gaussian nor threshold, the density in equation 7.9 is in non-standard form and so Metropolis-Hastings updates are employed. We use adaptive methods during the burn-in phase to determine an efficient multivariate normal proposal distribution centered at the previous value of $\mathbf{l}_j$ with covariance matrix $m\mathbf{M}$. For computational efficiency we use the same $\mathbf{M}$ for each block j, where $\mathbf{M}$ is the average posterior (co)variance of $\mathbf{l}_j$ within blocks and is updated each iteration of the burn-in period **?**. The scalar m is chosen using the method of **?** so that the proportion of successful jumps is optimal, with a rate of 0.44 when $\mathbf{l}_j$ is a scalar declining to 0.23 when $\mathbf{l}_j$ is high dimensional (**?**).

A special case arises for multi-parameter distributions in which each parameter is associated with a linear predictor. For example, in the zero-inflated Poisson two linear predictors are used to model the same data point, one to predict zero-inflation, and one to predict the Poisson variable. In this case the two linear predictors are updated in a single block even when the residual covariance between them is set to zero, because the first probability in Equation 7.9 cannot be factored:

$$Pr(\mathbf{l}_j|\mathbf{y}, \boldsymbol{\theta}, \mathbf{R}, \mathbf{G}) \propto p_i(y_i|\mathbf{l}_j)(\mathbf{e}_j|\mathbf{0}, \mathbf{R}_j) \tag{7.10}$$

When the block size is one (i.e. a univariate analysis) then the latent variables can be slice sampled for two-category `ordinal` and `categorical` models if `slice=TRUE` is passed to `MCMCglmm`.

7.2.2 Updating the location vector $\boldsymbol{\theta} = \left[\boldsymbol{\beta}' \ \mathbf{u}'\right]'$

? provide a method for sampling $\boldsymbol{\theta}$ as a complete block that involves solving the sparse linear system:

$$\tilde{\boldsymbol{\theta}} = \mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}(\mathbf{l} - \mathbf{W}\boldsymbol{\theta}_\star - \mathbf{e}_\star) \tag{7.11}$$

where $\mathbf{C}$ is the mixed model coefficient matrix:

$$\mathbf{C} = \mathbf{W}^{'}\mathbf{R}^{-1}\mathbf{W} + \begin{bmatrix} \mathbf{B}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} \end{bmatrix} \tag{7.12}$$

and $\mathbf{W} = [\mathbf{X}\ \mathbf{Z}]$, and $\mathbf{B}$ is the prior (co)variance matrix for the fixed effects.

$\boldsymbol{\theta}_\star$ and $\mathbf{e}_\star$ are random draws from the multivariate normal distributions:

$$\boldsymbol{\theta}_\star \sim N\left(\begin{bmatrix} \boldsymbol{\beta}_0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{bmatrix}\right) \tag{7.13}$$

and

$$\mathbf{e}_\star \sim N(\mathbf{0}, \mathbf{R}) \tag{7.14}$$

$\tilde{\boldsymbol{\theta}} + \boldsymbol{\theta}_\star$ gives a realisation from the required probability distribution:

$$Pr(\boldsymbol{\theta}|\mathbf{l}, \mathbf{W}, \mathbf{R}, \mathbf{G}) \tag{7.15}$$

Equation 7.11 is solved using Cholesky factorisation. Because $\mathbf{C}$ is sparse and the pattern of non-zero elements fixed, an initial symbolic Cholesky factorisation of $\mathbf{PCP}^{'}$ is preformed where $\mathbf{P}$ is a fill-reducing permutation matrix (**?**). Numerical factorisation must be performed each iteration but the fill-reducing permutation (found via a minimum degree ordering of $\mathbf{C}+\mathbf{C}^{'}$) reduces the computational burden dramatically compared to a direct factorisation of $\mathbf{C}$ (**?**).

Forming the inverse of the variance structures is usually simpler because they can be expressed as a series of direct sums and Kronecker products:

$$\mathbf{G} = (\mathbf{V}_1 \otimes \mathbf{A}_1) \oplus (\mathbf{V}_2 \otimes \mathbf{A}_2) \oplus \ldots \tag{7.16}$$

and the inverse of such a structure has the form

$$\mathbf{G}^{-1} = \left(\mathbf{V}_1^{-1} \otimes \mathbf{A}_1^{-1}\right) \oplus \left(\mathbf{V}_2^{-1} \otimes \mathbf{A}_2^{-1}\right) \oplus \ldots \tag{7.17}$$

which involves inverting the parameter (co)variance matrices ($\mathbf{V}$), which are usually of low dimension, and inverting $\mathbf{A}$. For many problems $\mathbf{A}$ is actually an identity matrix and so inversion is not required. When $\mathbf{A}$ is a relationship matrix associated with a pedigree, **??** give efficient recursive algorithms for obtaining the inverse, and **?** derive a similar procedure for phylogenies.

7.2.3 Updating the variance structures G and R

Components of the direct sum used to construct the desired variance structures are conditionally independent. The sum of squares matrix associated with each component term has the form:

$$\mathbf{S} = \mathbf{U}^{'}\mathbf{A}^{-1}\mathbf{U} \tag{7.18}$$

where $\mathbf{U}$ is a matrix of random effects where each column is associated with the relevant row/column of $\mathbf{V}$ and each row associated with the relevant row/column of $\mathbf{A}$. The parameter (co)variance matrix can then be sampled from the inverse Wishart distribution:

$$\mathbf{V} \sim IW((\mathbf{S}_p + \mathbf{S})^{-1},\ n_p + n) \tag{7.19}$$

where n is the number of rows in $\mathbf{U}$, and $\mathbf{S}_p$ and n_p are the prior sum of squares and prior degree's of freedom, respectively.

In some models, some elements of a parameter (co)variance matrix cannot be estimated from the data and all the information comes from the prior. In these cases it can be advantageous to fix these elements at some value and **?** provide a strategy for sampling from a conditional inverse-Wishart distribution which is appropriate when the rows/columns of the parameter matrix can be permuted so that the conditioning occurs on some diagonal sub-matrix. When this is not possible Metropolis-Hastings updates can be made.

7.2.4 Ordinal Models

For ordinal models it is necessary to update the cutpoints which define the bin boundaries for latent variables associated with each category of the outcome. To achieve good mixing we used the method developed by (**?**) that allows the latent variables and cutpoints to be updated simultaneously using a Hastings-with-Gibbs update.

7.2.5 Path Analyses

Elements of the response vector can be regressed on each other using the `sir` and `path` functions. Using the matrix notation of **?**, Equation 7.3 can be rewritten as:

$$\mathbf{\Lambda l} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Zu} + \mathbf{e} \tag{7.20}$$

where $\mathbf{\Lambda}$ is a square matrix of the form:

$$\mathbf{\Lambda} = \quad \mathbf{I} - \textstyle\sum_l \mathbf{\Psi}^{(l)} \lambda_l \tag{7.21}$$

This sets up a regression where the i^{th} element of the response vector acts as a weighted (by $\Psi^{(l)}_{i,j}$) predictor for the j^{th} element of the response vector with associated regression parameter λ_l. Often $\mathbf{\Psi}^{(l)}$ is an incidence matrix with the patterns of ones determining which elements of the response are regressed on each other.

Conditional on the vector of regression coefficients $\boldsymbol{\lambda}$, the location effects and variance structures can be updated as before by simply substituting $\mathbf{l}$ for

$\mathbf{\Lambda l}$ in the necessary equations. **?** provide a simple scheme for updating $\boldsymbol{\lambda}$. Note that Equation 7.20 can be rewritten as:

$$\begin{aligned} \mathbf{l} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u} = & \quad \mathbf{e} + \textstyle\sum_l \boldsymbol{\Psi}^{(l)}\mathbf{l}\lambda_l \\ = & \quad \mathbf{e} + \mathbf{L}\boldsymbol{\lambda} \end{aligned} \tag{7.22}$$

where $\mathbf{L}$ is the design matrix $\left[\boldsymbol{\Psi}^{(1)}\mathbf{l}, \boldsymbol{\Psi}^{(2)}\mathbf{l} \ldots \boldsymbol{\Psi}^{(L)}\mathbf{l}\right]$ for the L path coefficients. Conditional on $\boldsymbol{\beta}$ and $\boldsymbol{u}$, $\boldsymbol{\lambda}$ can then be sampled using the method of **?** with $\mathbf{l} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u}$ as response and $\mathbf{L}$ as predictor. However, only in a fully recursive system (there exists a row/column permutation by which all $\boldsymbol{\Psi}$'s are triangular) are the resulting draws from the appropriate conditional distribution, which requires multiplication by the Jacobian of the transform: $|\mathbf{\Lambda}|$. An extra Metropolis Hastings step is used to accept/reject the proposed draw when $|\mathbf{\Lambda}| \neq 1$.

When the response vector is Gaussian and fully observed, the latent variable does not need updating. For non-Gaussian data, or with missing responses, updating the latent variable is difficult because Equation 7.2.1 becomes:

$$Pr(l_i|\mathbf{y}, \boldsymbol{\theta}, \mathbf{R}, \mathbf{G}, \boldsymbol{\lambda}) \propto f_i(y_i|l_i) f_N((\mathbf{\Lambda}^{-1}\mathbf{e})_i|\mathbf{q}_i\mathbf{Q}_{/i}^{-1}\mathbf{e}_{/i}, q_i - \mathbf{q}_i\mathbf{Q}_{/i}^{-1}\mathbf{q}_i^{'}) \tag{7.23}$$

where $\mathbf{Q} = \mathbf{\Lambda}^{-1}\mathbf{R}\mathbf{\Lambda}^{-\top}$. In the general case $\mathbf{Q}$ will not have block diagonal structure like $\mathbf{R}$ and so the scheme for updating latent variables within residual blocks (i.e. Equation 7.9) is not possible. However, in some cases $\mathbf{\Lambda}$ may have the form where all non-zero elements correspond to elements of the response vector that are in the same residual block. In such cases updating the latent variables remains relatively simple:

$$Pr(\mathbf{l}_j|\mathbf{y}, \boldsymbol{\theta}, \mathbf{R}, \mathbf{G}) \propto p_i(y_i|\mathbf{l}_j)(\mathbf{\Lambda}_j^{-1}\mathbf{e}_j|\mathbf{0}, \mathbf{\Lambda}_j^{-1}\mathbf{R}_j\mathbf{\Lambda}_j^{-\top}) \tag{7.24}$$

7.2.6 Deviance and DIC

The deviance D is defined as:

$$D = -2\log(\Pr(\mathbf{y}|\boldsymbol{\Omega})) \tag{7.25}$$

where $\boldsymbol{\Omega}$ is some parameter set of the model. The deviance can be calculated in different ways depending on what is in 'focus', and MCMCglmm calculates this probability for the lowest level of the hierarchy (**?**). For fully-observed Gaussian response variables in the likelihood is the density:

$$f_N(\mathbf{y}|\mathbf{W}\boldsymbol{\theta},\ \mathbf{R}) \tag{7.26}$$

where $\boldsymbol{\Omega} = \{\boldsymbol{\theta},\ \mathbf{R}\}$. For discrete response variables in univariate analyses modeled using `family="threshold"` the density is

$$\prod_i F_N(\gamma_{y_i}|\mathbf{w}_i\boldsymbol{\theta},\ r_{ii}) - F_N(\gamma_{y_i+1}|\mathbf{w}_i\boldsymbol{\theta},\ r_{ii}) \tag{7.27}$$

where $\boldsymbol{\Omega} = \{\boldsymbol{\gamma},\ \boldsymbol{\theta},\ \mathbf{R}\}$. For other response variables variables (including discrete response variables modeled using `family="ordinal"`) it is the product:

$$\prod_i f_i(y_i|l_i) \tag{7.28}$$

with $\boldsymbol{\Omega} = \mathbf{l}$.

For multivariate models with mixtures of Gaussian (g), threshold (t) and other non-Gaussian (n) data (including missing data) we can define the deviance in terms of three conditional densities:

$$\begin{aligned} Pr(\mathbf{y}|\boldsymbol{\Omega}) = &\ \Pr(\mathbf{y}_g, \mathbf{y}_t, \mathbf{y}_n|\boldsymbol{\gamma}, \boldsymbol{\theta}_g, \boldsymbol{\theta}_t, \boldsymbol{l}_n, \boldsymbol{R}) \\ = &\ \Pr(\mathbf{y}_t|\boldsymbol{\gamma}, \boldsymbol{\theta}_t, \mathbf{y}_g, \boldsymbol{l}_n, \boldsymbol{R}) \Pr(\mathbf{y}_g|\boldsymbol{\theta}_g, \boldsymbol{l}_n, \boldsymbol{R}) \Pr(\mathbf{y}_n|\boldsymbol{l}_n) \end{aligned} \tag{7.29}$$

with $\boldsymbol{\Omega} = \{\boldsymbol{\gamma},\ \boldsymbol{\theta}_{/n},\ \boldsymbol{l}_n\ \mathbf{R}\}$. Have $(\boldsymbol{W\theta})_{a|b} = \boldsymbol{W}_a\boldsymbol{\theta}_a + \mathbf{R}_{a,b}\mathbf{R}_{b,b}^{-1}(\mathbf{l}_b - \boldsymbol{W}_b\boldsymbol{\theta}_b)$ and $\mathbf{R}_{a|b} = \mathbf{R}_{a,a} - \mathbf{R}_{a,b}\mathbf{R}_{b,b}^{-1}\mathbf{R}_{a,b}$ where the subscripts denote rows of the data vector/design matrices or rows/columns of the $\mathbf{R}$-structure. Then, the conditional density of $\mathbf{y}_g$ in Equation 7.29 is:

$$f_N\left(\mathbf{y}_g|(\boldsymbol{W\theta})_{g|n},\ \mathbf{R}_{g|n}\right) \tag{7.30}$$

The conditional density of $\mathbf{y}_n$ in Equation 7.29 is identical to that given in Equation 7.28, and for a single `"threshold"` trait

$$\prod_i F_N(\gamma_{y_i}|(\boldsymbol{W\theta})_{ti|g,n},\ r_{ti|g,n}) - F_N(\gamma_{y_i+1}|(\boldsymbol{W\theta})_{ti|g,n},\ r_{ti|g,n}) \tag{7.31}$$

is the conditional density for $\mathbf{y}_t$ in Equation 7.29, where $(\boldsymbol{W\theta})_{ti|g,n}$ is the i^{th} element of $(\boldsymbol{W\theta})_{t|g,n}$. Currently the deviance (and hence the DIC) will not be returned if there is more than one threshold trait.

The deviance is calculated at each iteration if `DIC=TRUE` and stored each `thin`th iteration after burn-in. However, for computational reasons the deviance is calculated mid-iteration such that the deviance returned at iteration i uses $\boldsymbol{\Omega}_i = \{\boldsymbol{\gamma}_i,\ \boldsymbol{\theta}_{/n,i},\ \boldsymbol{l}_{n,i-1}\ \mathbf{R}_i\}$. The mean deviance ($\bar{D}$) is calculated over all iterations, as is the mean of the latent variables ($\mathbf{l}$) the $\mathbf{R}$-structure and the vector of predictors ($\mathbf{W\theta}$). The deviance is calculated at the mean estimate of the parameters ($D(\bar{\boldsymbol{\Omega}})$) and the deviance information criterion calculated as:

$$\text{DIC} = 2\bar{D} - D(\bar{\boldsymbol{\Omega}}) \tag{7.32}$$

Distribution type	No. Data columns	No. latent columns	Density	function
`"gaussian"`	1	1	$Pr(y) =$	$f_N(\mathbf{w}\boldsymbol{\theta}, \sigma_e^2)$
`"poisson"`	1	1	$Pr(y) =$	$f_P(\exp(l))$
`"categorical"`	1	J-1	$Pr(y = k\|k \neq 1) =$	$\frac{\exp(l_k)}{1+\sum_{j=1}^{J-1}\exp(l_j)}$
			$Pr(y = 1) =$	$\frac{1}{1+\sum_{j=1}^{J-1}\exp(l_j)}$
`"multinomial`J`"`	J	J-1	$Pr(y_k = n_k\|k \neq J) =$	$\left(\frac{\exp(l_k)}{1+\sum_{j=1}^{J-1}\exp(l_j)}\right)^{n_k}$
			$Pr(y_k = n_k\|k = J) =$	$\left(\frac{1}{1+\sum_{j=1}^{J-1}\exp(l_j)}\right)^{n_k}$
`"ordinal"`	1	1	$Pr(y = k) =$	$F_N(\gamma_k\|l, 1) - F_N(\gamma_{k+1}\|l, 1)$
`"threshold"`	1	1	$Pr(y = k) =$	$F_N(\gamma_k\|\mathbf{w}\boldsymbol{\theta}, \sigma_e^2) - F_N(\gamma_{k+1}\|\mathbf{w}\boldsymbol{\theta}, \sigma_e^2)$
`"exponential"`	1	1	$Pr(y) =$	$f_E(\exp(-l))$
`"geometric"`	1	1	$Pr(y) =$	$f_G(\frac{\exp(l)}{1+\exp(l)})$
`"cengaussian"`	2	1	$Pr(y_1 > y > y_2) =$	$F_N(y_2\|\mathbf{w}\boldsymbol{\theta}, \sigma_e^2) - F_N(y_1\|\mathbf{w}\boldsymbol{\theta}, \sigma_e^2)$
`"cenpoisson"`	2	1	$Pr(y_1 > y > y_2) =$	$F_P(y_2\|l) - F_P(y_1\|l)$

`"cenexponential"`	2	1	$Pr(y_1 > y > y_2) =$	$F_E(y_2\|l) - F_E(y_1\|l)$
`"zipoisson"`	1	2	$Pr(y = 0) =$	$\frac{\exp(l_2)}{1+\exp(l_2)} + \left(1 - \frac{\exp(l_2)}{1+\exp(l_2)}\right) f_P(y\|\exp(l_1))$
			$Pr(y\|y > 0) =$	$\left(1 - \frac{\exp(l_2)}{1+\exp(l_2)}\right) f_P(y\|\exp(l_1))$
`"ztpoisson"`	1	1	$Pr(y) =$	$\frac{f_P(y\|\exp(l))}{1-f_P(0\|\exp(l))}$
`"hupoisson"`	1	2	$Pr(y = 0) =$	$\frac{\exp(l_2)}{1+\exp(l_2)}$
			$Pr(y\|y > 0) =$	$\left(1 - \frac{\exp(l_2)}{1+\exp(l_2)}\right) \frac{f_P(y\|\exp(l_1))}{1-f_P(0\|\exp(l_1))}$
`"zapoisson"`	1	2	$Pr(y = 0) =$	$1 - \exp(\exp(l_2))$
			$Pr(y\|y > 0) =$	$\exp(\exp(l_2)) \frac{f_P(y\|\exp(l_1))}{1-f_P(0\|\exp(l_1))}$
`"zibinomial"`	2	2	$Pr(y_1 = 0) =$	$\frac{\exp(l_2)}{1+\exp(l_2)} + \left(1 - \frac{\exp(l_2)}{1+\exp(l_2)}\right) f_B(0, n = y_1 + y_2\|\frac{\exp(l_1)}{1+\exp(l_1)})$
			$Pr(y_1\|y_1 > 0) =$	$\left(1 - \frac{\exp(l_2)}{1+\exp(l_2)}\right) f_B(y_1, n = y_1 + y_2\|\frac{\exp(l_1)}{1+\exp(l_1)})$

Table 7.1: Distribution types that can fitted using `MCMCglmm`. The prefixes `"zi"`, `"zt"`, `"hu"` and `"za"` stand for zero-inflated, zero-truncated, hurdle and zero-altered respectively. The prefix `"cen"` standards for censored where y_1 and y_2 are the upper and lower bounds for the unobserved datum y. J stands for the number of categories in the multinomial/categorical distributions and this must be specified in the family argument for the multinomial distribution. The density function is for a single datum in a univariate model with $\mathbf{w}$ being a row vector of $\mathbf{W}$. f and F are the density and distribution functions for the subscripted distribution (N=Normal, P=Poisson, E=Exponential, G=Geometric, B=Binomial). The $J - 1$ γ's in the ordinal models are the cutpoints, with γ_1 set to zero.

Chapter 8

Parameter Expansion

As the covariance matrix approaches a singularity the mixing of the chain becomes notoriously slow. This problem is often encountered in single-response models when a variance component is small and the chain becomes stuck at values close to zero. Similar problems occur for the EM algorithm and **?** introduced parameter expansion to speed up the rate of convergence. The idea was quickly applied to Gibbs sampling problems (**?**) and has now been extensively used to develop more efficient mixed-model samplers (e.g. **???**).

The columns of the design matrix ($\mathbf{W}$) can be multiplied by the non-identified working parameters $\boldsymbol{\alpha} = [1,\ \alpha_1,\ \alpha_2,\ \ldots \alpha_k]'$:

$$\mathbf{W}_\alpha = [\mathbf{X}\ \mathbf{Z}_1\alpha_1\ \mathbf{Z}_2\alpha_2\ \ \ldots\ \ \mathbf{Z}_k\alpha_k] \tag{8.1}$$

where the indices denote submatrices of $\mathbf{Z}$ which pertain to effects associated with the same variance component. Replacing $\mathbf{W}$ with $\mathbf{W}_\alpha$ we can sample the new location effects $\boldsymbol{\theta}_\alpha$ as described above, and rescale them to obtain $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\mathbf{I}_\beta \oplus_{i=1}^{k} \mathbf{I}_{u_i}\ \alpha_i)\boldsymbol{\theta}_\alpha \tag{8.2}$$

where the identity matrices are of dimension equal to the length of the subscripted parameter vectors.

Likewise, the (co)variance matrices can be rescaled by the set of α's associated with the variances of a particular variance structure component ($\boldsymbol{\alpha}_\mathcal{V}$):

$$\mathbf{V} = Diag(\boldsymbol{\alpha}_\mathcal{V})\mathbf{V}_\alpha Diag(\boldsymbol{\alpha}_\mathcal{V}) \tag{8.3}$$

The working parameters are not identifiable in the likelihood, but do have a proper conditional distribution. Defining the $n \times (k+1)$ design matrix $\mathbf{X}_\alpha$ with each column equal to the submatrices in Equation 8.1 postmultiplied by the relevant subvectors of $\boldsymbol{\theta}_\alpha$, we can see that $\boldsymbol{\alpha}$ is a vector of regression coefficients:

$$\mathbf{l} = \quad \mathbf{X}_\alpha\boldsymbol{\alpha} + \mathbf{e} \tag{8.4}$$

and so the methods described above can be used to update them.

8.0.1 Variances close to zero

To use parameter exapnsion in `MCMCglmm` it is necessary to specify a prior covariance matrix for $\boldsymbol{\alpha}$ which is non-null. In section 8.0.2 I discuss what this prior means in the context of posterior inference but for now we will specify two models, one parameter expanded and the other not. To illustrate I will fit a model that estimates the between mother variation in offsping sex ratio using parameter expansions:

```
> BTdata$sex[which(BTdata$sex == "UNK")] <- NA
> BTdata$sex <- gdata::drop.levels(BTdata$sex)
> prior1b = list(R = list(V = 1, fix = 1), G = list(G1 = list(V = 1,
+     nu = 1, alpha.mu = 0, alpha.V = 1000)))
> m7b.1 <- MCMCglmm(sex ~ 1, random = ~dam, data = BTdata,
+     family = "categorical", prior = prior1b, verbose = FALSE)
```

and fit a model that does not use parameter expansion:

```
> prior2b = list(R = list(V = 1, fix = 1), G = list(G1 = list(V = 1e-10,
+     nu = -1)))
> m7b.2 <- MCMCglmm(sex ~ 1, random = ~dam, data = BTdata,
+     family = "categorical", prior = prior2b, verbose = FALSE)
```

The prior densities in the two models are very similar across the range of variances with reasonable posterior support, and running the models for long enough will verify that they are sampling from very similar posterior densities. However, the mixing properties of the two chains are very different, with the non-parameter expanded chain (in red) getting stuck at values close to zero (Figure 8.1).

The parameter expanded model is 25% slower per iteration but the effective sample size is 3.049 times greater:

```
> effectiveSize(m7b.1$VCV[, 1])

    var1
194.9393

> effectiveSize(m7b.2$VCV[, 1])

    var1
63.93423
```

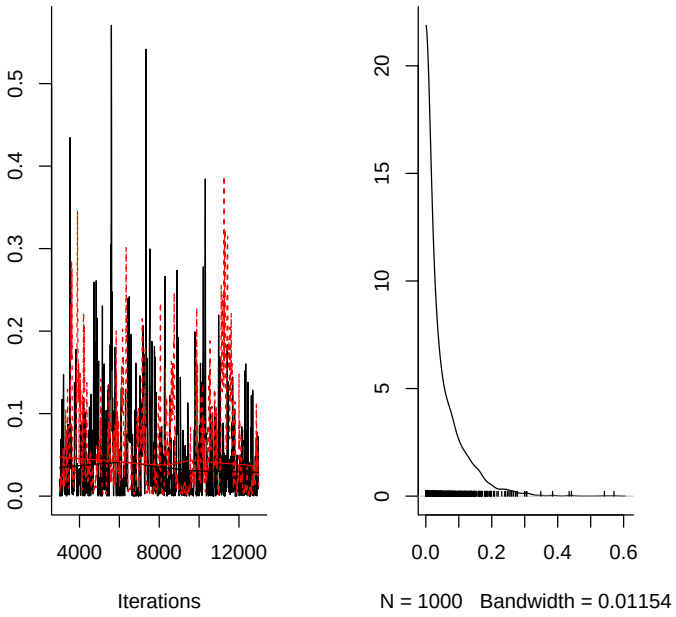

Figure 8.1: Traces of the sampled posterior distribution for between female variance in sex ratio. The black trace is from a parameter expanded model, and the red trace from a non-parameter expanded model.

8.0.2 Parameter expanded priors

The original aim of applying parameter expanded methods to Gibbs sampling was to speed up the convergence and mixing properties of the chain. They achieve this by introducing parameters that are not identified in the likelihood, and for which all information comes from the prior distribution. By placing priors on these parameters we can induce different prior distributions for the variance components. These priors are all from the non-central scaled F-distribution, which implies the prior for the standard deviation is a non-central folded scaled t-distribution (**?**). To use parameter expansion it is necessary to specify the prior means (`alpha.mu`) and prior covariance matrix (`alpha.V`) in the prior. Without loss of generality `V` can be set to one, so that the prior for the variance (`v`) has density function:

```
> df(v/alpha.V, df1 = 1, df2 = nu, ncp = (alpha.mu^2)/alpha.V)
```

and the prior for the standard deviation:

```
> 2 * dt(sqrt(v)/sqrt(alpha.V), df = nu, ncp = alpha.mu/sqrt(alpha.V))
```

where v> 0.

To illustrate I'll use the original Schools example from (**?**)

```
> data(schools)
> head(schools)
```

```
  school estimate   sd
1      A    28.39 14.9
2      B     7.94 10.2
3      C    -2.75 16.3
4      D     6.82 11.0
5      E    -0.64  9.4
6      F     0.63 11.4
```

The response variable `estimate` is the relative effect of Scholastic Aptitude Test coaching programs in 8 `schools`, and `sd` are the standard errors of the estimate. In the original example Gelman focused on the standard deviation of the between school effects and so we will place an improper flat prior on the standard deviation:

```
> prior1 <- list(R = list(V = diag(schools$sd^2),
+     fix = 1), G = list(G1 = list(V = 1e-10, nu = -1)))
> m7a.1 <- MCMCglmm(estimate ~ 1, random = ~school,
+     rcov = ~idh(school):units, data = schools,
+     prior = prior1, verbose = FALSE)
```

In this example there is information on the between school variance although we only have a single estimate for each school. This is possible because the within school variance was available for each school and we were able to fix the residual variance for each school at this value (See Section 4.4). The posterior distribution of the between school standard deviation is shown in Figure 8.2 with the flat prior shown as a solid line.

We can also use the inverse-gamma prior with scale and shape equal to 0.001:

```
> prior2 <- list(R = list(V = diag(schools$sd^2),
+     fix = 1), G = list(G1 = list(V = 1, nu = 0.002)))
> m7a.2 <- MCMCglmm(estimate ~ 1, random = ~school,
+     rcov = ~idh(school):units, data = schools,
+     prior = prior2, verbose = FALSE)
```

but Figure 8.3 indicates that such a prior in this context may put too much density and values close to zero.

For the final prior we have `V=1`, `nu=1`, `alpha.mu=0` which is equivalent to a proper Cauchy prior for the standard deviation with scale equal to $\sqrt{\texttt{alpha.V}}$. Following `Gelman.2006` we use a scale of 25:

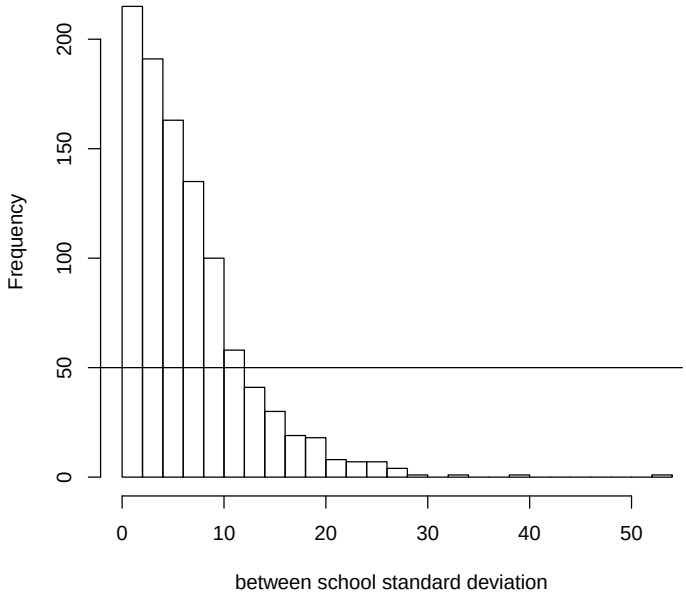

Figure 8.2: Between school standard deviation in educational test scores, with an improper uniform prior

```
> prior3 <- list(R = list(V = diag(schools$sd^2),
+     fix = 1), G = list(G1 = list(V = 1, nu = 1,
+     alpha.mu = 0, alpha.V = 25^2)))
> m7a.3 <- MCMCglmm(estimate ~ 1, random = ~school,
+     rcov = ~idh(school):units, data = schools,
+     prior = prior3, verbose = FALSE)
```

and Figure 8.4 shows that the prior may have better properties than the inverse-gamma, and that the posterior is less distorted.

8.0.3 Binary response models

When analysing binary responses the residual variance is not identified in the likelihood and without a prior the posterior is improper. If a weak prior is placed on the residual variance then the chain appears to mix poorly and the MCMC output often looks terrible. However, this poor mixing is in some ways superficial. As discussed in section 5.2 we can rescale the location effects and variances by the estimated residual variance to obtain the posterior distribution

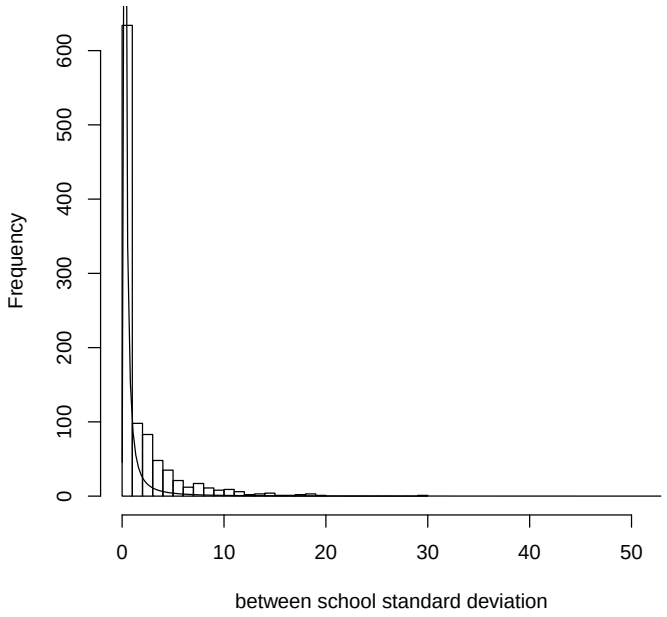

Figure 8.3: Between school standard deviation in educational test scores, with an inverse-gamma prior with shape and scale set to 0.001

for some fixed value of the actual residual variance. For example, we can refit the sex ratio model using a residual variance fixed at ten rather than one:

```
> prior3b = list(R = list(V = 10, fix = 1), G = list(G1 = list(V = 1,
+     nu = 1, alpha.mu = 0, alpha.V = 1000)))
> m7b.3 <- MCMCglmm(sex ~ 1, random = ~dam, data = BTdata,
+     family = "categorical", prior = prior3b, verbose = FALSE)
```

The two models appear to give completely different posteriors (Figure 8.6)

```
> plot(mcmc.list(m7b.1$VCV[, 1], m7b.3$VCV[, 1]))
```

but rescaling indicates that they are very similar:

```
> c2 <- (16 * sqrt(3)/(15 * pi))^2
> plot(mcmc.list(m7b.1$VCV[, 1]/(1 + c2 * m7b.1$VCV[,
+     "units"]), m7b.3$VCV[, 1]/(1 + c2 * m7b.3$VCV[,
+     "units"])))
```

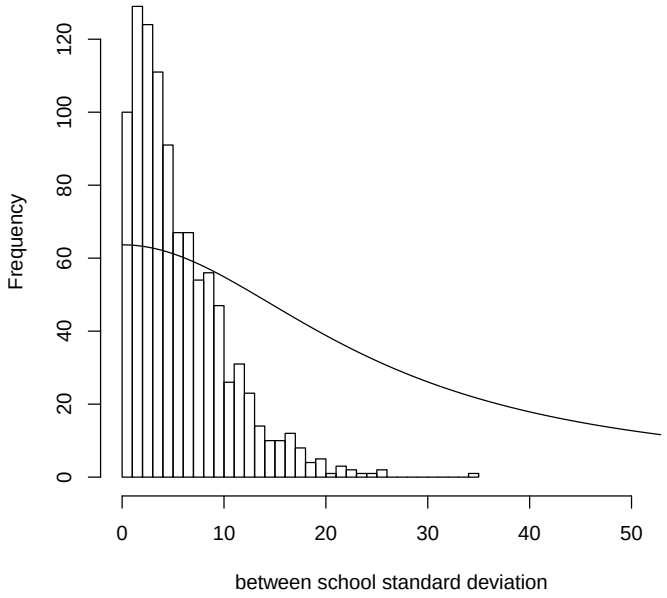

Figure 8.4: Between school standard deviation in educational test scores, with a Cauchy prior with a scale of 25.

The prior specification for the between mother variance is different in the two models but Figure 8.6 suggests that the difference has little influence. However, the mixing properties of the second chain are much better (**?**):

```
> effectiveSize(m7b.1$VCV[, 1]/(1 + c2 * m7b.1$VCV[,
+     "units"]))

    var1
194.9393

> effectiveSize(m7b.3$VCV[, 1]/(1 + c2 * m7b.3$VCV[,
+     "units"]))

    var1
636.7686
```

Although the chain mixes faster as the residual variance is set to be larger, numerical problem are often encountered because the latent variables can take

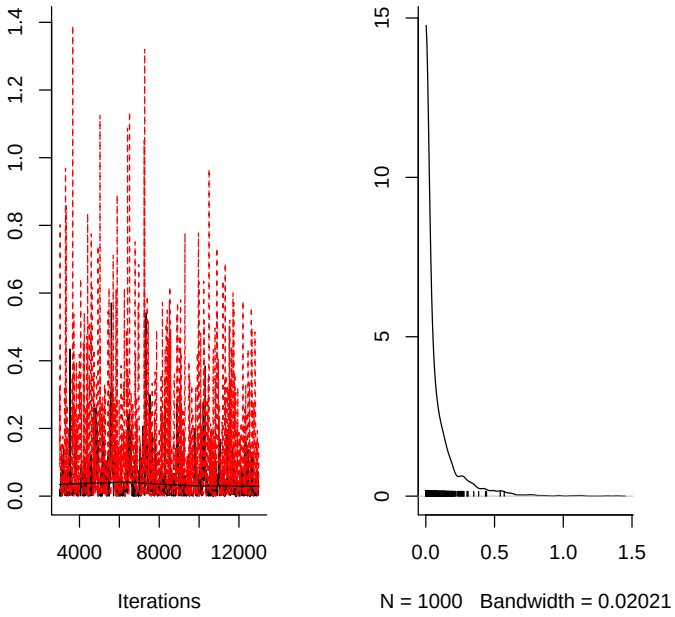

Figure 8.5: Between mother variation in sex ratio with the residual variance fixed at 1 (black trace) and 10 (red trace).

on extreme values. For most models a variance of 1 is safe, but care needs to be taken so that the absolute value of the latent variable is less than 20 in the case of the logit link and less than 7 for the probit link. If the residual variance is not fixed but has an alternative proper prior placed on it then the Metropolis-Hastings proposal distribution for the latent variables may not be well suited to the local properties of the conditional distribution and the acceptance ratio may fluctuate widely around the optimal 0.44. This can be fixed by using the slice sampling methods outlined in `?` by passing `slice=TRUE` to `MCMCglmm`. Slice sampling can also be more efficient even if the prior is fixed at some value:

```
> m7b.4 <- MCMCglmm(sex ~ 1, random = ~dam, data = BTdata,
+     family = "categorical", prior = prior3b, verbose = FALSE,
+     slice = TRUE)
> effectiveSize(m7b.4$VCV[, 1]/(1 + c2 * m7b.3$VCV[,
+     "units"]))

    var1
625.4758
```

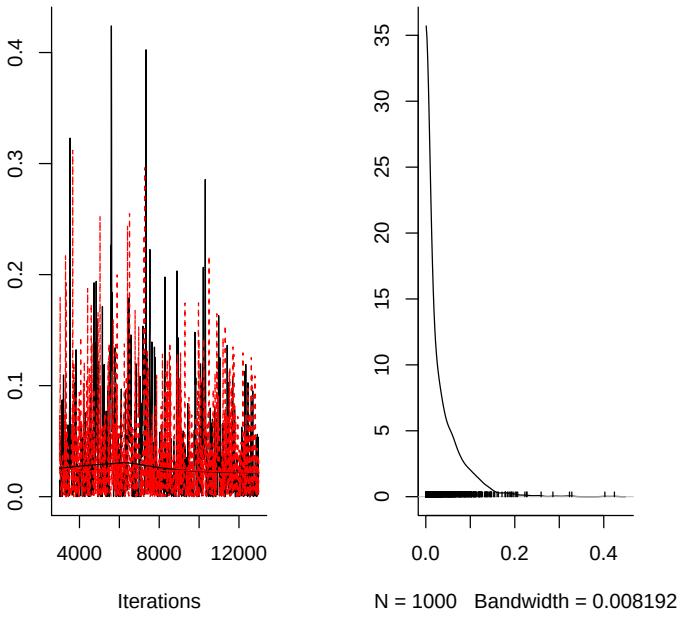

Figure 8.6: Between mother variation in sex ratio with the residual variance fixed at 1 (black trace) and 10 (red trace) but with both estimates rescaled to what would be observed under no residual variance.

Chapter 9

Path Analysis & Antedependence Structures

There are many situations where it would seem reasonble to put some apsect of a response variable in as a predictor, and the only thing that stops us is some (often vague) notion that this is a bad thing to do from a statistical point of view. The approach appears to have a long history in economics but I came across the idea in a paper written by **?**. The notation of this section, and indeed the sampling strategy employed in MCMCglmm is derived from this paper.

9.1 Path Anlaysis

$\boldsymbol{\Psi}^{(l)}$ is a square matrix of dimension $N \times N$ where $\Psi^{(l)}_{i,j} = k$ sets up the equation $y_i = \lambda_l k y_j \ldots$. In matrix terms:

$$\boldsymbol{\Lambda}\mathbf{y} = \mathbf{y} - \sum_l \boldsymbol{\Psi}^{(l)}\mathbf{y}\lambda_l \tag{9.1}$$

Having $\boldsymbol{\Psi} = \left[\boldsymbol{\Psi}^{(1)}\ \boldsymbol{\Psi}^{(2)}\ \ldots \boldsymbol{\Psi}^{(L-1)}\ \boldsymbol{\Psi}^{(L)}\right]$ we have

$$\begin{aligned} \boldsymbol{\Lambda}\mathbf{y} = &\quad \mathbf{y} - \boldsymbol{\Psi}\left(\mathbf{I}_L \otimes \mathbf{y}\right)\boldsymbol{\lambda} \\ = &\quad \mathbf{y} - \boldsymbol{\Psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_N\right)\mathbf{y} \end{aligned} \tag{9.2}$$

where $\boldsymbol{\lambda} = \left[\lambda_1\ \lambda_2 \ldots \lambda_{L-1}\ \lambda_L\right]^\top$, and $\boldsymbol{\Lambda} = \mathbf{I}_N - \boldsymbol{\Psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_N\right)$

Each $\boldsymbol{\Psi}^{(l)}$ can be formed using the function `sir` which takes two formulae. $\boldsymbol{\Psi} = \mathbf{X}_1\mathbf{X}_2^\top$ where $\mathbf{X}_1$ and $\mathbf{X}_2$ are the model matrices defined by the formulae (with intercept removed). $\mathbf{X}_1$ and $\mathbf{X}_2^\top$ have to be conformable, and although this could be achieved in many ways, one way to ensure this is to have categorical predictors in each which have common factor levels. To give a concrete example, lets take a sample of individuals measured a variable number of times for 2 traits:

```
> id <- sample(1:100, 100, replace = T)
> y1 <- rnorm(100)
> y2 <- rnorm(100)
> y <- c(y1, y2)
> trait <- gl(2, 100)
```

Lets then imagine that each of these indiviuals interacts with another randomly chosen individual - indexed in the vector `id1`

```
> id1 <- sample(id, 100, replace = T)
> id <- as.factor(c(id, id))
> id1 <- factor(c(id1, id1), levels = levels(id))
```

we will adopt a recursive model where by the phenotypes of indiviuals in the `id1` vector affect those in the `id` vector:

```
> Psi <- sir(~id1, ~id)
```

we can see that the first record for individual `id[1]`=10 is directly affected by individual `id1[1]`=48's traits:

```
> Psi[1, which(id == id1[1])]

 31  90 131 190
  1   1   1   1
```

i.e indiviual `id1[1]`=48 has 4 records.

We can build on this simple model by stating that only trait 2 affects trait 1:

```
> Psi <- sir(~id1:at.level(trait, 1), ~id:at.level(trait,
+     2))
> Psi[c(1, 101), which(id == id1[1])]

    31 90 131 190
1    0  0   1   1
101  0  0   0   0
```

or that trait 2 affect both trait 2 and trait 1:

```
> Psi <- sir(~id1, ~id:at.level(trait, 2))
> Psi[c(1, 101), which(id == id1[1])]

    31 90 131 190
1    0  0   1   1
101  0  0   1   1
```

```
> my.data <- data.frame(y1 = y1, y2 = y2, id = id[1:100],
+     id1 = id1[1:100], x = rnorm(100))
> m1 <- MCMCglmm(y1 ~ x + sir(~id1, ~id) + y2, data = my.data,
+     verbose = FALSE)
```

One problem is that $\mathbf{e}^\star$ the residual vector that appears in the likelihood for the latent variable does not have a simple (block) diagonal structure when (as in the case above) the elements of the response vector that are regressed on each other are not grouped in the R-structure:

$$\mathbf{e}^\star \sim N\left(\mathbf{0}, \mathbf{\Lambda}^{-1}\mathbf{R}\mathbf{\Lambda}^{-\top}\right) \tag{9.3}$$

Consequently, analyses that involve latent variables (i.e. non-Gaussian data, or analyses that have incomplete records for determining the R-structure) are currently not implemented in MCMCglmm.

The `path` function is a way of specifying path models that are less general than those formed by `sir` but are simple enough to allow updating of the latent variables associated with non-Gaussian data. Imagine a residual structure is fitted where the N observations are grouped into n blocks of k. For instance this might be k different characteristics measured in n individuals. A path model may be entertained whereby an individual's characteristics only affect their own characteristics rather than anyone elses. In this case, $\mathbf{\Psi}^{(l)} = \boldsymbol{\psi}^{(l)} \otimes \mathbf{I}_n$ is block diagonal, and $\mathbf{\Psi} = \boldsymbol{\psi} \otimes \mathbf{I}_n$ where $\boldsymbol{\psi} = \left[\boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)} \ldots \boldsymbol{\psi}^{(L)}\right]$. Consequently,

$$\begin{aligned}
\mathbf{\Lambda} = & \mathbf{I}_N - \mathbf{\Psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_N\right) \\
= & \mathbf{I}_N - \left(\boldsymbol{\psi} \otimes \mathbf{I}_n\right)\left(\boldsymbol{\lambda} \otimes \mathbf{I}_N\right) \\
= & \left(\mathbf{I}_k \otimes \mathbf{I}_n\right) - \left(\boldsymbol{\psi} \otimes \mathbf{I}_n\right)\left(\boldsymbol{\lambda} \otimes \mathbf{I}_k \otimes \mathbf{I}_n\right) \\
= & \left(\mathbf{I}_k - \boldsymbol{\psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_k\right)\right) \otimes \mathbf{I}_n
\end{aligned} \tag{9.4}$$

and so $\mathbf{\Lambda}^{-1} = \left(\mathbf{I}_k - \boldsymbol{\psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_k\right)\right)^{-1} \otimes \mathbf{I}_n$ and $|\mathbf{\Lambda}| = |\mathbf{I}_k - \boldsymbol{\psi}\left(\boldsymbol{\lambda} \otimes \mathbf{I}_k\right)|^n$

Mean centering responses can help mixing, because $\boldsymbol{theta}$ and $\boldsymbol{lambda}$ are not sampled in a block. (Jarrod - I guess when $|\mathbf{\Lambda}| = 1$ this could be detected and updating occur in a block?)

9.2 Antedependence

An $n \times n$ unstructured covariance matrix can be reparameterised in terms of regression coefficeints and residual variances from a set of n nested multiple regressions. For example, for $n = 3$ the following 3 multiple regressions can be defined:

$$\begin{aligned}
u_3 = & \; u_2\beta_{3|2} + u_1\beta_{3|1} + e_{u_3} \\
u_2 = & \; u_1\beta_{2|1} + e_{u_2} \\
u_1 = & \; e_{u_1}
\end{aligned} \tag{9.5}$$

Arranging the regression coefficients and residual ('innovation') variances into a lower triangular matrix and diagonal matrix respectively:

$$\mathbf{L}_u = \begin{bmatrix} 1 & 0 & 0 \\ -\beta_{2|1} & 1 & 0 \\ -\beta_{3|1} & -\beta_{3|2} & 1 \end{bmatrix} \tag{9.6}$$

and

$$\mathbf{D}_u = \begin{bmatrix} \sigma^2_{e_{u_1}} & 0 & 0 \\ 0 & \sigma^2_{e_{u_2}} & 0 \\ 0 & 0 & \sigma^2_{e_{u_3}} \end{bmatrix} \tag{9.7}$$

gives

$$\mathbf{V}_u = \mathbf{L}_u \mathbf{D}_u \mathbf{L}_u^\top \tag{9.8}$$

Rather than fit the saturated model (in this case all 3 regression coefficients) k^{th} order antedependence models seek to model $\mathbf{V}_u$ whilst constraining the regression coefficients in $\mathbf{L}_u$ to be zero if they are on sub-diagonals¿k. For example, a first order antedependence model would set the regression coefficients in the second off-diagonal (i.e. $\beta_{3|1}$) to zero, but estimate those in the first sub-diagonal (i.e. $\beta_{2|1}$ and $\beta_{3|2}$). For a 3×3 matrix, a second order antedependence model would fit a fully unstructured covariance matrix. In terms of Gibbs sampling this parameterisation is less efficient because $\mathbf{V}_u$ is sampled in two blocks (the regression coefficients followed by the innovation variances) rather than in a single block from the inverse Wishart. However, more flexible conjugate prior specifications are possible by placing multivariate normal priors on the regression coefficients and independent inverse Wishart priors on the innovation variances. By constraining arbitrary regression coefficients to be zero in a fully unstructured model allows any fully recursive path model to be constructed for a set of random effects.

9.3 Scaling

The path analyses described above essentially allow elements of the response vector to be regressed on each other. Regressing an observation on itself would seem like a peculiar thing to do, although with a little work we can show that by doing this we can allow two sets of observations to conform to the same model except for a difference in scale.

Acknowledgments

MCMCglmm relies heavily on sparse matrix operations facilitated by the CSparse library written by Tim Davis, whom I thank. Countless people have given me feedback, suggestions and bug reports. In particular, I'd like to thank Shinichi Nakagawa, Michael Morrissey & Laura Ross, and also Loeske Kruuk who provided funding for this work through a Leverhulme trust award. More recently this work has been funded by NERC and the Royal Society.

Bibliography