final report

Project code: B.LSM.0060

Prepared by: Christopher M Strickland, Clair L Alston, Graham E Gardner

Date published: September 2015

PUBLISHED BY
Meat & Livestock Australia Limited
Locked Bag 991
NORTH SYDNEY NSW 2059

Efficient Bayesian estimation of mixed effects growth models

Meat & Livestock Australia acknowledges the matching funds provided by the Australian Government to support the research and development detailed in this publication.

Report 1: Bayesian linear mixed effects models

Christopher M Strickland, Clair L. Alston and Graham E. Gardner

0.1 Introduction

Mixed models, often described as a mixture of fixed and random terms, are used in cases where the data are clustered due to subpopulations, such as *sires* in genetics trials, *years* in trials that are conducted annually, *assessors* in experiments where the person obtaining the measurements may be subjective, *individual* in experiments that contain measurements in time on traits such as growth (longitudinal studies) and where higher level terms are considered random. A mixed model handles the different sources of error in data by treating the variation as within and between clusters.

Demidenko (2004), Wu (2010) and Verbeke and Molenberghs (2009) provide substantial references for these models in the frequentist framework, and provide much of the mathematical foundations required for the Bayesian framework. Sorensen and Gianola (2002) provide a comprehensive likelihood and Bayesian outline of these models. Introductory level Bayesian methods are outlined by Carlin and Louis (2009) and Hoff (2009).

Mixed models can be estimated in several Bayesian software packages, for example; WinBugs J. et al. (2000) and library MCMCglmm Hadfield (2010) in the R package, however, the size of the data common in animal trials makes the use of this software impractical. We are developing a module which uses PyMCMC Strickland et al. (2012), to efficiently estimates parameters of the Bayesian linear mixed model when sample size is large.

0.2 Models and Methods

Denoting the $(N \times 1)$ vector of observations $\mathbf{y}$, then the linear mixed model may be represented as,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}; \;\; \boldsymbol{\varepsilon} \sim \mathbf{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_N\right) \tag{1}$$

where $\mathbf{X}$ is an $(N \times k)$ matrix of regressors, $\boldsymbol{\beta}$ is a $(k \times 1)$ vector of fixed effects, $\mathbf{Z}$ is an $(N \times m)$ matrix of regressors for the random effects, $\mathbf{u}$ is an $(m \times 1)$ vector of random effects and $\boldsymbol{\varepsilon}$ is an $(N \times 1)$ normally distributed random vector, with a mean vector $\mathbf{0}$ and a covariance matrix $\sigma^2\mathbf{I}_N$. Here $\mathbf{I}_N$ refers to an $N-$dimensional identity matrix. The random effects are defined such that

$$\mathbf{u} \sim \mathbf{N}\left(\mathbf{0}, \mathbf{D}^{-1}\right), \tag{2}$$

where $\mathbf{D}$ is an $(m \times m)$ precision matrix. To complete the model specification, we need to define priors for $\boldsymbol{\beta}, \mathbf{D}$ and σ. We assume that

$$\boldsymbol{\beta} \sim \mathbf{N}\left(\underline{\boldsymbol{\beta}}, \underline{\mathbf{V}}^{-1}\right)$$

and

$$\sigma \sim \mathbf{IG}\left(\frac{\underline{\nu}}{2}, \frac{\underline{s}}{2}\right), \tag{3}$$

where $\underline{\nu}$ is the prior degrees of freedom and $\underline{s}$ is the prior scale term for the inverted gamma distribution. We assume that $\mathbf{u}$ can be separated into K independent parts,

$\mathbf{u}_i$, where

$$\mathbf{u}_i \sim \mathbf{N}\left(\mathbf{0}, \mathbf{D}_i^{-1} \otimes \mathbf{I}_{m_i}\right),$$

where the variance of $\mathbf{u}_i$, $\mathbf{D}_i^{-1} \otimes \mathbf{I}_{m_i}$, implies that $\mathbf{D} = \oplus_i \left(\mathbf{D}_i \otimes \mathbf{I}_{m_i}\right)$ and $\oplus$ refers to the matrix direct sum and $\otimes$ is the Kronecker product. It is further assumed that the $(l_i \times l_i)$ precision matrix $\mathbf{D}_i$, for $i = 1, 2, \ldots, K$, is assumed to be distributed following

$$\mathbf{D}_i \sim \mathbf{Wishart}\left(\underline{w}_i, \underline{\mathbf{S}}_i\right), \tag{4}$$

where $\underline{w}_i$ is the degrees of freedom parameter and $\underline{\mathbf{S}}_i$ is the inverse of the scale matrix, for the Wishart distribution. The precision matrix $\mathbf{D}_i$ models the correlation between the l_i components in $\mathbf{u}_i = \left(\mathbf{u}_{i,1}^T, \mathbf{u}_{u,2}^T, \ldots, \mathbf{u}_{i,l_i}^T\right)^T$, where each component $u_{i,j}$, for $j = 1, 2, \ldots, l_i$ is a $(J_i \times 1)$ vector and $J_i = \frac{m_i}{l_i}$.

It is interesting to note that the only difference between the random effects, $\mathbf{u}$, and the fixed effects, $\boldsymbol{\beta}$, is the extra layer of hierarchy in the prior specification, of the random effects.

The generic form, for the linear mixed model in (1), is a compact representation that nests many useful forms, including the specific models used in case studies 1 and 2, which we now describe.

0.2.1 MCMC estimation

The aim of Markov chain Monte Carlo (MCMC) estimation is to sample from the joint posterior distribution for the unknown parameters, $\boldsymbol{\beta}, \mathbf{u}, \mathbf{D}$ and σ. An MCMC scheme at iteration j is as follows:

Algorithm 1

1. Jointly sample $\boldsymbol{\beta}^{(j)}$ and $\mathbf{u}^{(j)}$ from $p\left(\boldsymbol{\beta}, \mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z}, \mathbf{D}^{(j-1)}, \sigma^{(j-1)}\right)$.
2. Sample $\mathbf{D}^{(j)}$ from $p\left(\mathbf{D}|\mathbf{y}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}^{(j)}, \mathbf{u}^{(j)}, \sigma^{(j-1)}\right)$.
3. Sample $\sigma^{(j)}$ from $p\left(\sigma|\mathbf{y}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}^{(j)}, \mathbf{u}^{(j)}, \mathbf{D}^{(j)}\right)$.

Step 1, of Algorithm 1, is undertaken using standard Bayesian linear regression theory. Observing that Equation (1), once conditioned on $\mathbf{D}$, is simply a standard Bayesian linear regression model; see Chapter Ref: StricklandLinearRegression for specific details. If we define $\mathbf{W} = (\mathbf{X}; \mathbf{Z})$ and $\boldsymbol{\delta} = \left(\boldsymbol{\beta}^T, \mathbf{u}^T\right)^T$, then we can express Equation (1) as

$$\mathbf{y} = \mathbf{W}\boldsymbol{\delta} + \boldsymbol{\varepsilon}$$

and it follows that

$$\boldsymbol{\delta}|\mathbf{W}, \sigma \sim \mathbf{N}\left(\overline{\boldsymbol{\delta}}, \overline{\mathbf{H}}\right), \tag{5}$$

where $\overline{\boldsymbol{\delta}} = \left(\frac{\mathbf{W}^T\mathbf{W}}{\sigma^2} + \underline{\mathbf{H}}\right)^{-1}\left(\frac{\mathbf{W}^T\mathbf{y}}{\sigma^2} + \underline{\mathbf{H}}\underline{\boldsymbol{\delta}}\right)$, with $\underline{\boldsymbol{\delta}} = \left(\underline{\boldsymbol{\beta}}^T, \mathbf{0}^T\right)^T$ and $\underline{\mathbf{H}} = \begin{bmatrix} \underline{\mathbf{V}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}$. It is important to note that

Step 2, of Algorithm 1, takes advantage of the conjugacy of the prior specification for $\mathbf{D}$, in Equation (3). To sample $\mathbf{D}$, we make use of the fact that the blocks, $\mathbf{D}_i$, described in Equation (4) are independent of each other. We therefore sample each precision matrix, $\mathbf{D}_i$, for $i = 1, 2, \ldots, K$, separately, using

$$\mathbf{D}_i|\mathbf{u}_i \sim \mathbf{Wishart}\left(\overline{\mathbf{w}}_i, \overline{\mathbf{S}}_i\right),$$

where $\overline{w}_i = \underline{w}_i + J_i$ and $\overline{\mathbf{S}}_i = \underline{\mathbf{S}}_i + \sum_{j=1}^{J_i} \mathbf{u}_{i,j}\mathbf{u}_{i,j}^T$.

Step 3, of Algorithm 1, also takes advantage of the conjugacy of the prior specification of σ, in Equation (3). In this case the posterior for σ is given as

$$\sigma|\mathbf{y}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}, \mathbf{u} \sim \mathbf{IG}\left(\frac{\overline{\nu}}{2}, \frac{\overline{s}}{2}\right),$$

where $\overline{\nu} = \underline{\nu} + N$ and $\overline{s} = \underline{s} + (\mathbf{y} - \mathbf{W}\boldsymbol{\delta})^T(\mathbf{y} - \mathbf{W}\boldsymbol{\delta})$.

This method is coded in python using sparse matrix techniques to ensure efficient computation.

0.3 Case study

0.3.1 Large Data Set

The data set which will primarily be the focus of this work is an extensive 5 year study of growth in newborn lambs. There are 17525 individuals in the study, with a total of 141206 measurements of weights taken over time.

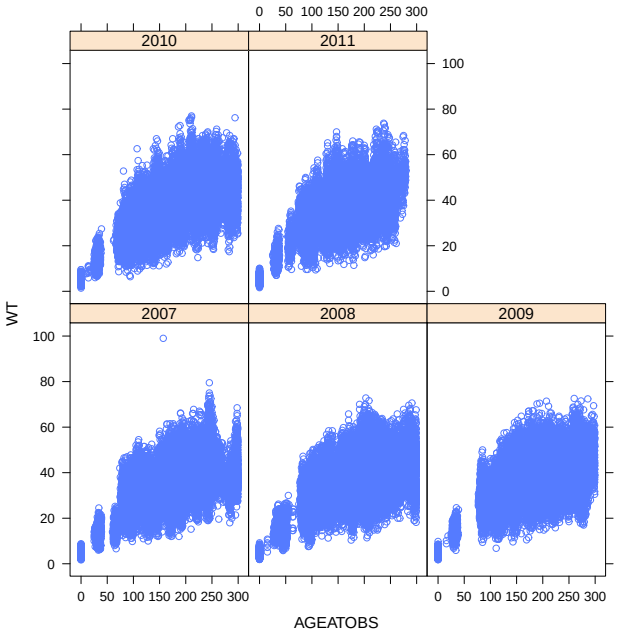
The model that is required to be fitted in this case is

WT = Constant +poly(AGEATOBS,3)+AGEOFDAM+SEX+SIRETYPE+FLOCK+drop+btrt+dambreedST +poly(AGEATOBS,3):AGEOFDAM +poly(AGEATOBS,3):SEX +poly(AGEATOBS,3):SIRETYPE +poly(AGEATOBS,3):FLOCK +poly(AGEATOBS,3):drop +poly(AGEATOBS,3):btrt +poly(AGEATOBS,3):dambreedST +poly(AGEATOBS,3):pwwt +poly(AGEATOBS,3):pfat +FLOCK:drop +FLOCK:SEX +FLOCK:btrt +FLOCK:SIRETYPE +FLOCK:dambreedST +drop:btrt +drop:SIRETYPE +drop:dambreedST +SEX:btrt +SEX:SIRETYPE +btrt:AGEOFDAM +btrt:SIRETYPE +btrt:dambreedST +pwwt +pwwt:pwwt +pwwt:FLOCK +pwwt:btrt +pfat +pfat:pfat +pfat:btrt , random= sSire+dam:drop+us(1+poly(AGEATOBS,2)):id

This is a total of 311 fixed effects coefficients and 65430 random coefficients. The sparseness of this model requires efficient statistical methods to compute the MCMC estimates efficiently.

0.3.2 HCWT

In this report, we focus on a simpler example, Hot carcass weight (HCWT). This is the weight of a carcase when it has had its non-usable meat products, such as head,



cmcm

Figure 1 Plot of weight against the age of the animal. Each year is represented in a separate graph.

intestinal tract and internal organs removed, as well as the hide. This weight is taken on room temperature carcases.

In this case study, the HCWT of 3282 animals has been recorded. Researchers are interested in the effect of age at observation, sex, sire breed and body fat (and possible interactions) on the hot carcase weight. However, as an observational data set, these factors are highly unbalanced. For example, there are 1040 females and 2242 males and the 6 different breeds have samples sizes ranging from 164 to 1134.

Figure 2 (top) is a plot of hot carcase weight vs age. generally, these animals reach maturity by around 150 days, and as such we would not be expecting a large response in terms of growth in this data set. However, it can be observed that there may be an increase in weight with age. However, the bottom plot indicates that other factors, such as breed, may also account for this behaviour.

At first glance, there does not appear to be too much "activity" in this data set, however, the mixed model is used to investigate the contributions of the random effects, as removing these may make the signal of the data clearer. There are several factors that are likely to contribute to the overall variance, but are not of interest in themselves. These factors, known as random effects, are sire (149), flock (15) and the group in which the animals were slaughtered (kill group, 59). Figure 3 illustrates the mean and range of hot carcase weights in each of these factors.

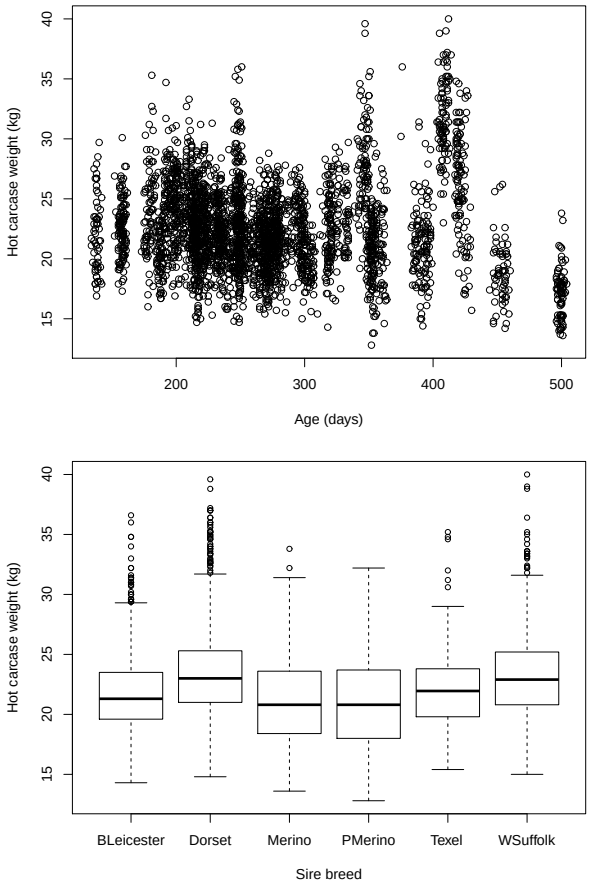
Apart from the obvious mean differences of HCWT between the 149 sires, it is also possible that there may be a sire by age interaction, as shown in Figure 4. These are data taken for a sub-sample of sires, and indicate that these sires may have a different response in terms of weight gain with increasing age. For example, sires 1, 2 and 6 seem to have a clear increase with age, whereas sires 3, 4 and 5 seem to have reached full maturity and are not increasing with additional age. Therefore, the fixed effect of age may be "blurred" by the different sires. In our analysis, we will also test the importance of this random term.

0.3.3 Model for Case Study

The observed variable, **HCWT**, is modeled as

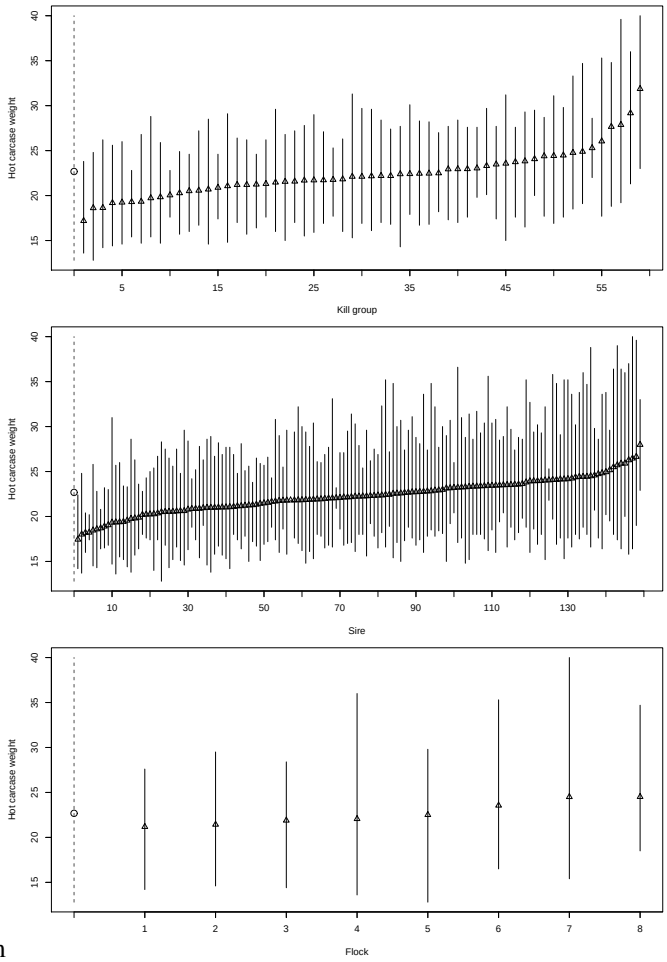
$$\mathbf{HCWT} = \beta_0 + \mathbf{Age}\beta_1 + \mathbf{S}\beta_2 + \mathbf{AS}\beta_3 + \mathbf{B}\boldsymbol{\beta}_4 + \mathbf{AB}\boldsymbol{\beta}_5 + \boldsymbol{PF}\beta_6 + \mathbf{Z}_s\boldsymbol{u}_s + \mathbf{Z}_{fl}\boldsymbol{u}_{fl} + \mathbf{Z}_{kg}\boldsymbol{u}_{kg} + \boldsymbol{\varepsilon}, \tag{6}$$

where **Age** is the age of the sheep, **S** is the sex of the sheep, **AS** is an interaction term between the age and sex of the sheep, **B** is the breed of the sheep, **AB** is an interaction term between the age and the breed of the sheep, **PF** is the percentage of fat in the sheep, $\mathbf{Z}_s$ is an $(N \times m_s)$ design matrix for the random effects that defines the sire of the sheep, $\mathbf{Z}_{fl}$ is an $(N \times m_{fl})$ design matrix for the random effects that specifies the flock, $\mathbf{Z}_{kg}$ is an $(N \times m_{kg})$ design matrix for the random effects for the kill group, and $\boldsymbol{\varepsilon}$ is the residual vector. The random effects, $\mathbf{u}_s$, $\mathbf{u}_{fl}$, $\mathbf{u}_{kg}$, are



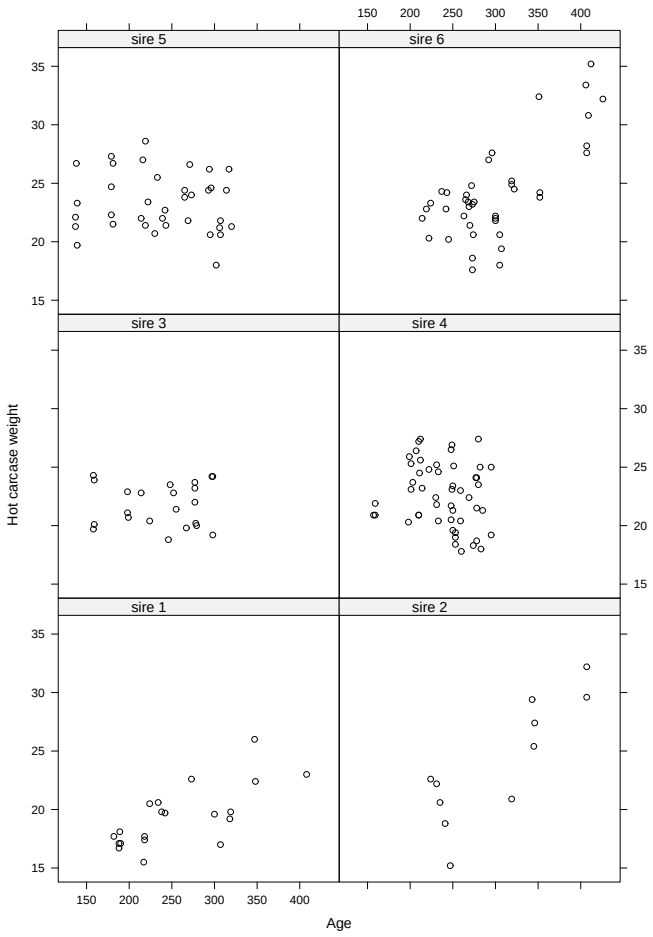
cmcm

Figure 2 Top: Plot of Hot carcase weight against the age of the animal at slaughter. Researchers would like to know if additional age (after maturity) increases the carcase weight. Bottom: Boxplot of hot carcase weight against sire breed, which is one of several factors which may influence final weight.



cmcm

Figure 3 Top: Ordered plot of hot carcase weight (mean and range) against the kill group for animals at slaughter. These groups contain unbalanced data in terms of the fixed effects (Sire breed, age, sex) and can be seen to vary widely in mean and range. The grey dashed line represents mean and range for whole data set. Middle: Ordered plot of hot carcase weight (mean and range) against the individual sires for animals at slaughter Bottom: Ordered plot of hot carcase weight (mean and range) against the flock from which the animals originated at slaughter.



cmcm

Figure 4 A selection of the sire response vs age against hot carcase weight.

further defined, such that

$$\begin{aligned}\mathbf{u}_s &\sim \mathbf{N}\left(0, d_s^{-1}\right), \\ \mathbf{u}_{fl} &\sim \mathbf{N}\left(0, d_{fl}^{-1}\right), \\ \mathbf{u}_{kg} &\sim \mathbf{N}\left(0, d_{kg}^{-1}\right). \end{aligned} \tag{7}$$

The design matrices $\mathbf{Z}_s$, $\mathbf{Z}_{fl}$ and $\mathbf{Z}_{kg}$ are designed in a similar fashion. As an example, if we define the i_{th} row of $\mathbf{Z}_s$ as $\mathbf{z}_{s,i}^T$ then if the i^{th} carcase belongs to group j, where $j \in \{1, 2, \ldots, m_s\}$, then set the j^{th} element in $z_{s,i}^T$ equal to 1 and the remaining elements equal to zeros. For instance, if $m_s = 3$ and the i^{th} carcase is from the second sire, then $z_{s,i}^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$.

The model in (6) and (7) can be expressed in terms of the general model in (1) and (2), by defining $\mathbf{X} = (\mathbf{1}_N; \mathbf{Age}; \mathbf{S}; \mathbf{AS}; \mathbf{B}; \mathbf{AB}; \mathbf{PF})$, where the generic notation $(\mathbf{A}; \mathbf{B})$ is used to denote that the columns of $\mathbf{A}$ are concatenated with the columns of $\mathbf{B}$, and $\mathbf{1}_N$ is a vector of ones of order N. Further, define $\boldsymbol{\beta} = \left(\beta_0, \beta_1, \beta_2, \boldsymbol{\beta}_3^T, \boldsymbol{\beta}_4^T, \boldsymbol{\beta}_5, \beta_6\right)^T$, $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_s & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_{fl} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Z}_{kg} \end{bmatrix}$, $\mathbf{u} = \left(\mathbf{u}_s^T, \mathbf{u}_{fl}^T, \mathbf{u}_{kb}\right)^T$ and $\mathbf{D} = \begin{bmatrix} d_s\mathbf{I}_{m_s} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & d_{fl}\mathbf{I}_{m_{fl}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & d_{kg}\mathbf{I}_{m_{kg}} \end{bmatrix}$.

The prior hyperparameters for this case study are defined as $\underline{w} = 10$ and $\underline{S} = 0.01$, for each random effect, $\underline{\boldsymbol{\beta}} = \mathbf{0}$, $\underline{\mathbf{V}} = 0.01\mathbf{I}_{15}$

0.4 The python library LMM

The class names LMM has been added to the PyMCMC library. The coding of this class is relatively straight-forward and will be familiar to the commands animal scientists are used to working with in R and SAS.

```
class LMM(LMM_Data):
    """Class for LMM (linear mixed model) class
    arguments:
        data - An list (or tuple) or dictionary of the data
        observations - Position (if data is a list) or string (if data is a
                       dictionary) with gives specifies the position of the
                       vector of observations in data.

        nit - The number of iterations to run the MCMC scheme. The default value
              is 20000 iterations.

        burn - The length of the burnin for the MCMC scheme. The default value
               is 5000 iterations. These will be discarded from the 20000
               iterations.

        optional arguments:
            fixed_effects - a list (or tuple) of the positions (data is a list)
                            or strings (data is a dictionary), of the fixed effects.
                            Note the special keyword 'CONSTANT' can be used to
```

```
                add a constant.

constant - Should be set to False if there is no constant in the
           model (in the case factors are in the model. Otherwise
           when the factors are constructed it is
           either assumed that the user has defined one of the fixed
           effects to be a contant or that they have used the
           special keyword 'CONSTANT'.

random_effects - a list (or tuple)  of positions (data is a list)
or strings (data is a dictionary), of the random effects.

factors - a list (or tuple) of positions (data is a list)
          or strings (data is a dictionary), of the factors.

prior_fixed_effects - Is specified by a list. Currently the options
    are, ['normal', mean, precision], where mean, should
    either be a vector of the same dimension of the number of
    regressors, or a scalar, in which case the entire mean vector
    will equal the scalar parameter value.

    The precision, likewise, can be the entire precision matrix, or
    a scalar, where the scalar case implies a diagonal precision
    matrix where all diagonal elements are equal to the scalar. If
    no prior for the fixed effects is specified a flat prior is
    used.

prior_random_effects - Is specified by a list. The most simple case
    is ['wishart', nu, S], where nu is the degrees of freedom parameter
    and S is in the inverse scale matrix for the Wishart distribution.
    If the prior is specified as above each of the scale parameters for
    the random effects will have the same prior, in which case S should
    be a prior.

    Alternatively, one may specify the prior completely. This is
    best explained by an example. Suppose the list of random effects
    are ['var1', 'var2', ('var1', 'var3')]. Suppose we want to
    assume that the scale parameters for 'var1' and ('var1', 'var3')
    are correlated. In this case we would specify the prior as
    [['var2',['wishart', nu1, S1]],
    [('var1', ('var1', 'var2'), ['wishart', nu2, S2]]], where in the
    case nu1, nu2 and S1 are scalars and S2 is a (2 x 2) precision
    matrix. Note currently we only allow joint priors between a
    random_efect and an interation with the same random effect.

    NOTE: If not prior is specified a flat prior will be used for
    each individual random effects.

prior_scale - Is specified as a list. One one option is currently
    available. That is ['gamma' ,nu, s], where nu is the degrees of
    freedom parameter and s is the inverse scale parameter for the
    gamma distribution. If no prior is specified a flat prior will
    be used.
```

The class LMM is then imported into standard python code for use in a statistical analysis. A typical python script file will look like the following

```
#Python code for hot carcase weight example 2

import data_manipulation as dm
from lmm import LMM
```

```
data = dm.data_dict('Data/HCWTPyMCMC.txt', ' ')
#functions convert names variables to floating point arrays in dictionary
dm.float_dict(['AGEATOBS','HCWT', 'PFAT'], data)


#functions enumerate dictionary for named variables
#Note returns a dictonary that maps the name to the index

map_name_index = dm.enum_dict(['FLOCK', 'Killgroup', 'SEX','SIREBREED',
                               'sSire'], data)

#Prior fixed effects
#Normal prior with a mean vector = 0. and a precision = diag(0.1* np.eye(k)
#where k is the number of regressors
prior_fixed = ['normal', 0., 0.01]

#prior for random effects
prior_RE = ['wishart', 10, 0.1]

#instantiate class for linear mixed model
lmm = LMM(data, 'HCWT', 10000, 1000,
          fixed_effects = ['CONSTANT', 'AGEATOBS', 'SEX', 'SIREBREED',
                           'PFAT', ('AGEATOBS', 'SEX'),
                           ('AGEATOBS', 'SIREBREED')],
              random_effects = ['sSire', 'FLOCK', 'Killgroup'],
              factors = ['SIREBREED', 'SEX', 'sSire', 'FLOCK',
                         'Killgroup'],
              prior_fixed_effects = prior_fixed,
              prior_random_effects = prior_RE)

lmm.output()
```

0.5 Data Analysis and Results

Hot carcase weight analysis

The default output for the linear mixed model analysis is as follows. The runtime output bar which appears after the burnin and iteration information allows the user to see how far the job has progressed, giving an indication of the expected end time.

The user is also informed of the number of random effects to be estimated. The time the MCMC sampler took to run is provided followed by the parameter estimates.

```
Number of random effects = 216
PyMCMC is now running

The number of iterations = 10000
The length of the burnin = 5000

[########################################]

-----------------------------------------------------

The time (seconds) for the MCMC sampler = 33.40
Number of blocks in MCMC sampler =  3
```

	mean	sd	2.5%	97.5%	IFactor
beta[0]	16.6	0.531	15.6	17.7	3.57
beta[1]	0.0161	0.00214	0.012	0.0202	3.56
beta[2]	2.16	0.45	1.26	3.01	3.56
beta[3]	10.7	0.693	9.3	12	3.75
beta[4]	3.52	0.517	2.54	4.57	3.54
beta[5]	9.16	0.678	7.74	10.4	3.56
beta[6]	2.62	0.483	1.66	3.52	3.53
beta[7]	3.15	0.666	1.89	4.51	3.58
beta[8]	0.131	0.105	-0.0849	0.328	3.57
beta[9]	-0.00466	0.00177	-0.00804	-0.00107	3.57
beta[10]	-0.0312	0.00209	-0.0353	-0.0271	3.54
beta[11]	-0.00635	0.00209	-0.0103	-0.00213	3.75
beta[12]	-0.0274	0.00204	-0.0312	-0.0234	3.75
beta[13]	-0.00193	0.00196	-0.00577	0.00178	3.75
beta[14]	-0.00897	0.00281	-0.0142	-0.00326	3.45
sigma	2.35	0.03	2.29	2.41	3.49
d_fl	0.0345	0.00908	0.0204	0.0519	5.31
d_sire	0.9	0.0767	0.753	1.05	5.26
d_kg	2.17	0.196	1.8	2.54	3.52

```
Acceptance rate  beta  =  1.0
Acceptance rate  sigma  =  1.0
Acceptance rate  d_fl  =  1.0
Acceptance rate  d_sire  =  1.0
Acceptance rate  d_kg  =  1.0
```

The posterior mean, standard deviation (sd) and 95% credible intervals are provided for each model parameter. The 95% credible interval is a convenient measure to determine the significance of a term relative to the base "no effect" model. The inefficiency factor (IFactor) is also provided. This gives an indication

of the effectiveness of the MCMC sampler. For example, an IFactor of 2 would indicate that the MCMC would need to be run for double the number of iterations if we require a sample size of 5000 independent posterior draws. In general, an IFactor of less then 50 is considered quite adequate.

As can be noted in the output, the coefficients are currently returned as "beta[0]", "beta[1]", etc. These coefficients are returned in the order they are placed in the model, in this case, beta[0] = constant, beta[1] = AGEATOBS, beta[2] = Male, beta[3] = PMerino, beta[4] = WSuffolk, beta[5] = Merino, beta[6] = Dorset, beta[7] = Texel, beta[8] = PFAT, ...

These generic labels will be replaced with the actual labels in the near future. This system of reliable ordering will be of assistance to researchers, who will be able to write script code to complete other tasks whilst using this output. This is currently an issue in using MCMCglmm() in R.

The magnitude of the random sources can be seen in the comparison of overall sigma (2.35), flock d_{fl} (0.0345), sire d_{sire} (0.9) and killgroup d_{kg} (2.17). Kill group was the biggest source of variation, followed by sire and then flock.

If researchers wish to obtain more then the posterior summary above, it is a simple matter to add an additional line of code to save the MCMC chains to a file, such as below.

```
mcmc.CODAoutput(filename = 'betaout')
```

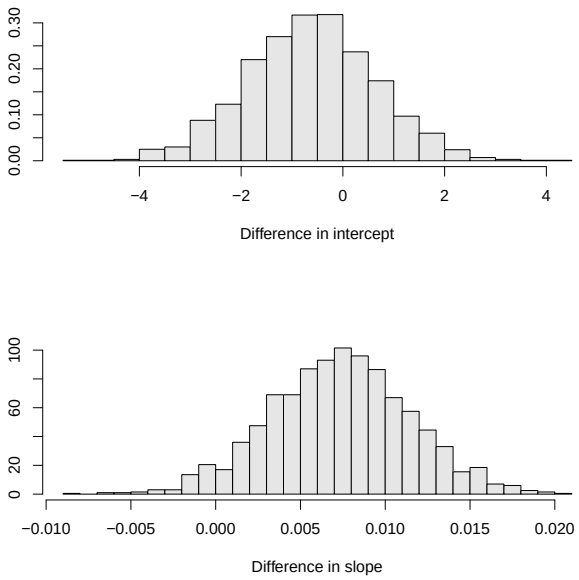
This is an important feature as a key advantage of using Bayesian modelling in mixed effects model is for ease of comparison between effects such as breed in a post-hoc fashion.

For example, it is relatively easy to test for differences between breeds by comparing the values at each iteration of the MCMC chain. An example of this is given in Figure 5 where we test for differences in birth weight (HSCW) and growth rate (response to age) between the Texel and Dorset breeds. We note from the top graph that Texel breeds are not significantly heavier than Dorset at birth (intercept), however, there is evidence of Texel being less responsive to age, with 4% of values falling below zero.

0.6 Discussion

We have implemented the code employing both functional and unit testing frameworks. Unit tests are particularly important as they test individual components of the program using mock data and make it easy to test each component under many different situations. This provides us a way to test for an extremely wide range of variations that arise from the different possible ways the users may use the class LMM to specify the model they are interested in. As a result we can expect far fewer bugs resulting from the users entering data into the program in unexpected ways, or setting up model variations that we have not considered.

We are currently completing the code for interaction terms in the random effects. The code is written, but further testing and debugging is required. The most complex



cmcm

Figure 5 Density of the difference between MCMC draws for the parameters of birth weight (left) and growth rate (right) for Dorset and Texel breeds. The differences between birth weight between the the two breeds are non-significant (p=0.32), however, the growth rates are significantly different (p=0.04), with Dorset breed growing faster than Texel.

part of the code is written in Python, and works correctly, however, the Fortran implementation (which is required in this part of the procedure for efficiency reasons) still requires further debugging. This should not take much longer.

We have also decided to provide for the option of Stochastic search variable selection for the fixed effects. This is essentially a method of automatic variable selection. We believe this option will be particularly useful as it provides a set of most probable variables for inclusion in the model and the saves the user from re-running the analysis and taking variables out base on the output and re-running the model and so on. It is also statistically more valid. Given in applied work the researchers are faced with many possible alternative explanatory variables, we think this inclusion is of substantial practical importance. We already have classes to implement the stochastic search variable selection, as a part of the PyMCMC library so we just need to integrate this method carefully to ensure that we retain efficiency in the current implementation. We do not believe this will cause us too much trouble.

References

Carlin BP and Louis TA 2009 *Bayesian methods for data analysis*. Chapman & Hall / CRC press, Boca Raton.

Chib S and Carlin BP 1999 On MCMC sampling in hierarchical longitudinal models. *Statistics and Computing* **9**, 17–26.

Chib S, Greenberg E and Winkelmann R 1998 Posterior simulation and Bayes factors in panel count data model. *Journal of Econometrics* **86**, 33–54.

Demidenko E 2004 *Mixed models: Theory and applications*. Wiley, New Jersey.

Hadfield JD 2010 MCMC methods for Multi-response Generalised Linear Mixed models: The MCMCglmm R Package. *Journal of Statistical Software* **33**, 1–22.

Hoff P 2009 *A first course in Bayesian statistical methods*. Springer, Seattle.

J. D, Thomas A, Best N and Spiegelhalter D 2000 WinBUGS – a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing* **10**, 325–37.

Sorensen D and Gianola D 2002 *Likelihood, Bayesian, and MCMC methods in quantative genetics*. Springer, New York.

Strickland CM, Denham R, Alston CL and Mengersen KL 2012 A python package for Bayesian estimation using Markov chain Monte Carlo. Under review.

Verbeke G and Molenberghs G 2009 *Linear mixed models for longitudinal data*. Springer, New York.

Wu L 2010 *Mixed effects models for complex data*. Chapman & Hall / CRC press, Boca Raton.