TUTORIAL

Estimation of a biological trait heritability using the animal model

How to use the MCMCglmm R package

Octobre 2012

Pierre de Villemereuil

Contents

1 Preamble

1.1 Why this tutorial?

This tutorial is intended for students or researchers in the domain of evolutionary ecology, interested in using the animal model to estimate the heritability of biological traits in a wild population. It aims at bringing theoretical and practical help on three main issues: (*i*) understanding what heritability is, what it quantifies and how the animal model works; (*ii*) learning by practice how to implement animal models using the MCMCglmm R package; and (*iii*) introducing Bayesian statistics (priors, Markov Chain Monte Carlo, etc.). The author tried to use examples of increasing complexity to show best as well as most tedious aspects of MCMC estimation methods. Finally, although this tutorial is directly inspired from J. Hadfield course notes (Hadfield 2010), it tries to bring new information more focused on heritability estimation and good use of MCMC.

1.2 Prior knowledge

The reader is assumed to have an average knowledge of Evolutionary Biology and its problems, along with some notions of quantitative genetics. He is also assumed to have a basic understanding of statistical mixed models framework and its associated vocabulary (fixed and random effects especially) and to have some familiarity with their formulation. A vague knowledge of Bayesian statistics and a familiarity with the R statistical software are needed for a correct understanding of the Part 4, but not for the Parts 2 and 3.

2 What is heritability?

2.1 Definition

Let's study a phenotypic trait in a given wild population. This trait varies among individuals, each having a more or less different value from the others. This variation is quantified by a variance (called *phenotypic variance* or V_P). It originates from different sources, but in a schematic way, we will state it has a genetic component V_G and an environmental component V_E:

$$V_P = V_G + V_E \tag{1}$$

The genetic variance itself originates from different sources. To keep it simple we will assume an *additive* component V_A (i.e. the effect "one by one" of the transmitted alleles) and a *non-additive* component V_{NA} (dominance effects, epistasis, etc.). We thus write:

$$V_P = V_A + V_{NA} + V_E \tag{2}$$

The interest of that variance decomposition lies in the fact that V_A stands for the part of phenotypic variability which is actually transmitted to the descendants (that is to say, at the level of the whole population). For natural selection to have an effect, a part of the phenotypic variability, on which it acts, must be "transmittable" (we say that the trait is *heritable*).

Thus we introduce a quantity, called *heritability*, which allow us to measure how much the phenotypic variability of a trait in a given population is likely to be transmitted to the descendants. The heritability h^2 is defined (*stricto senso*) as the contribution of additive variance into the variability of the phenotype. Using previous notations :

$$h^2 = \frac{V_A}{V_P} \tag{3}$$

The heritability has a value that lies between 0 and 1.

2.2 A few words

- Heritability is *not* heredity! Heredity is the transmission of a phenotypic value from a parent to his offspring, while heritability is the transmission of the phenotypic variability within a population from generation to generation. For example, the number of legs in Human has a no heritability (any variation would be environmental), but is totally hereditary.
- It is important to stress that the heritability of a trait is defined for a given population at a given time. This quantity can vary between populations, and from time to time.
- A weakly heritable trait is not more likely to not be selected. The biological traits that are the closest to fitness have most of the time a small heritability (Mousseau & Roff 1987).

2.3 Possible biases on heritability

A bias on heritability can originates from "artefactual" resemblance between individuals, which could be misinterpreted as additive genetic effect. We thus need to get rid of two main sources of nuisance: the rest of the genetic effects (link to a correct estimation of V_{NA}) and the environmental effect.

2.3.1 Genetic nuisance

They are mainly linked to dominance effects and epistasis, which, if they are badly accounted for, can lead to an overestimation of heritability. Hopefully, some methods are not sensitive to

dominance effects (see Part 2.4). Epistasis is a more complex and generalized issue, but it is a second order effect which is generally neglected by the models. Inbreeding can also lead to an overestimation of heritability, if not accounted for. There is also some factors that are hard to take into account, such as linkage...

2.3.2 Environmental nuisance

They are of various type, yielding greater resemblance between individuals than expected based on kinship. Among others:

- Common environment: Individuals in a common environment (birth spot, habitat, etc.) are likely to show extra phenotypic resemblance.
- Parental effects: Some characters of parents (quality, immunity, etc.) can yield an extra resemblance between their offspring and them; or more generally, extra resemblance within their offspring, independently of their phenotypes[1].
- Assortative mating: A tendency of individuals to mate with partners sharing the same phenotype will lead to extra resemblance within their offspring.

From a practical point of view, the sensitivity to genetic effects (mostly dominance and inbreeding) will depend on the approach used to measure heritability. Environmental effects can sometimes be avoided with a careful design. The animal model approach allows to integrate some of theses effects in the model.

2.4 Which approach to measure heritability in the wild?

Sibling design This approach use the kinship between siblings in order to measure heritability. An ANOVA model compares the within-family variance to the inter-family variance to estimate heritability. Because of an experimental protocol difficult to apply in wild population and an annoying sensitivity to dominance and common environment effects, this approach is less used in wild animal population. It is however to be noticed that the half-sibling design approach is not sensitive to dominance effects, but require some methodological conditions (certain identification of both parents, polygamy or sequential monogamy, etc.).

Parent-offspring regression This approach uses the regression of the phenotype of the mid-parent (or one of the parents) on the mean phenotype of the offspring. Well studied, its properties are well-known (Falconer & Mackay 1996, Roff 1997, Lynch & Walsh 1998). This approach is not sensitive to dominance effects, but ignore issues link to inbreeding. More over,

[1]The difference is important, since the former kind is what most biologists have in mind, whereas only the last kind can be modelized using the animal model, see Part 3.3.

although a correction exists concerning assortative mating (Falconer & Mackay 1996, p.178), the parent-offspring regression remains sensitive to common environment between parents and offspring (territory inheritance, weak dispersal, etc.) and to *transgenerationnal* parental effects (i.e. parental effects which depend on the phenotype of the parent).

Animal model The animal model is a more complex approach, in the sense that it doesn't use only one kind of kinship, but the whole pedigree of the population. It thus uses the maximum of information available, and to take into account inbreeding. It is a mixed model and thus can take into account several factors (mostly environmental) in order to avoid biases described in Part 2.3. We will focus on this approach in the following parts.

3 The animal model in theory

3.1 Basic principle

The animal model uses a pedigree of the wild population. Such a pedigree indicate the father (or sire) and mother (or dam) for each individuals, like in the following array:

```
   individual  mother father
1          A1    NA     NA
2          A2    NA     NA
3          A3    NA     NA        #NA stands for Not Available value
4          A4    NA     NA
5          A5    A2     A4
6          A6    A1     A3
7          A7    A1     A3
8          A8    A1     A3
9          A9    A2     A4
10        A10    A2     A4
```

The individuals for which both the father and the mother are unknown are called *founders*. They generally are individuals from the beggining of the survey or immigrants. The founders are assumed unrelated to each others. This kind of pedigree allows the calculation of kinship among all individuals of the population:

```
     A1  A2  A3  A4  A5  A6  A7  A8  A9  A10
A1    1   0   0   0   0 1/2 1/2 1/2   0   0
A2    0   1   0   0 1/2   0   0   0 1/2 1/2
A3    0   0   1   0   0 1/2 1/2 1/2   0   0
A4    0   0   0   1 1/2   0   0   0 1/2 1/2
A5    0 1/2   0 1/2   1   0   0   0 1/2 1/2
A6  1/2   0 1/2   0   0   1 1/2 1/2   0   0
A7  1/2   0 1/2   0   0 1/2   1 1/2   0   0
A8  1/2   0 1/2   0   0 1/2 1/2   1   0   0
```

```
A9    0 1/2   0 1/2 1/2   0   0   0   1 1/2
A10   0 1/2   0 1/2 1/2   0   0   0 1/2   1
```

Note that, except for founders, the diagonal elements are not always 1, particularly in presence of inbreeding. This matrix is the additive genetic variance-covariance matrix $\mathbf{A}$. It is included in a random or mixed model in order to estimate its associated variance component V_A (i.e. additive genetic variance). These kind of models are called *animal models*.

3.2 Model description

Let's have several measures of a phenotype y on n individuals, during a given period and in a given wild population. These measures are gathered in a vector $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$. These data are given along a pedigree of the population containing informations for all n measured individuals. Our aim is to separate the variation of the Y data between an additive genetic variance V_A and the "rest" (often too quickly assimilated to environmental variance). In order to do that, we will consider the phenotype y_i of the individual i as a variation around the average population phenotype μ in function of the pedigree of the individual and its environment or other uncontrolled factors. We thus write:

$$y_i = \mu + a_i + e_i \tag{4}$$

In this equation, μ stands for the average population phenotype. a_i is called the breeding value and accounts for the influence of the additive effect of the alleles on the phenotype. e_i is a residual accounting for the rest of the possible variation.

We still need to define the distribution of the breeding values a_i and residuals e_i (μ is a simple constant). The breeding values are assumed normally distributed:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \sim \mathcal{N}(0; \mathbf{A}V_A) \tag{5}$$

Let's recall here that $\mathbf{A}$ is the additive genetic matrix, and is related the population pedigree. V_A is the additive genetic variance we are looking to estimate here, in order to estimate the heritability of the trait. The residuals e_i are also normally distributed:

$$\begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} \sim \mathcal{N}(0; \mathbf{I}V_R) \tag{6}$$

where $\mathbf{I}$ stands for the identity matrix and V_R is the residual variance.

From equations 4, 5 and 6, the key assumptions of the animal model are:

- The Y trait is normally distributed (this assumption can be drop using generalized models)
- The breeding values a_i are normally distributed and correlated among related individuals. The function of the pedigree is to structure the correlation between individuals by taking into account their kinship.
- The residuals e_i are normally distributed and uncorrelated. They also are independent from the breeding values (e.g. no environment-genotype interaction).

The main outputs of the animal model are an estimate of the additive variance $\hat{V}_A$ and an estimate of the residual variance $\hat{V}_R$. These two variance components sum to the total phenotypic variance V_P. We thus estimate the heritability as:

$$\hat{h}^2 = \frac{\hat{V}_A}{\hat{V}_A + \hat{V}_R} \tag{7}$$

3.3 Going further into the model

3.3.1 Adding random effects

In order to account for some possible biases (common environment, parental effects...), it is possible to add random factors in the model. When those random factors ($F_1, ...,F_K$) are added, they have to be taken into account for the calculation of the total phenotypic variance, such as:

$$\hat{h}^2 = \frac{\hat{V}_A}{\hat{V}_A + \hat{V}_{F_1} + ... + \hat{V}_{F_K} + \hat{V}_R} \tag{8}$$

It is not advised to add too many factors, since it could cause tedious instability in the estimation of variance components.

Parental effects Including the identity of the parents (or only one parent) as a random effect allows to account for possible parental effects. However, it has to be stressed that, included that way, parental effects only take into account the resemblance among siblings from the same parents, but in no way the resemblance between parents and offspring! For example, one famous maternal effect is the quality of mother's milk in mammals: mothers of the best phenotypic "quality" are supposed to give the best milk, which in turns help her offspring to grow healthy. In this context, including the identity of the mother into the model will state that the offspring of a same mother are likely to develop a close phenotype because of the more or less good quality of the mother's milk, but it doesn't imply any resemblance between the

'quality' of the mother and her offspring. In other words, the parental effect is independent from the parental phenotype (this a non transgenerationnal effect)! A kind of parental effects in animal models that are transgenerationnal are *genetic* parental effects (see Wilson *et al.* 2010, for example). This effect assume a hypothetical 'maternal performance' trait which is a summary of potential maternal traits acting on the offspring's phenotype and is heritable. Including a fully parametrized parental effect (including measured maternal traits acting on offspring's phenotype) is not straightforward using animal model, but a more general type of models exists that allow for this (Kirkpatrick & Lande 1989, Räsänen & Kruuk 2007, Day & Bonduriansky 2011).

Dominance effects Dominance effects are problematic only through a supplementary correlation among full-sibs. Since the animal model uses all kind of relationships, it is sensitive to dominance effects, especially if the pedigree contains large siblings groups. A dominance matrix is not necessarily difficult to construct, but the calculation has to be made before the use of MCMCglmm (see Part 4.7).

Repeated measures If repeated measures on individuals are available, it is possible to estimate what is called a *permanent environment* effect, by including the identity of individuals as a random effect. This allows to take into account intra-individual variations and possible measurement errors. The permanent environment effect also accounts for a part of the non-additive genetic effects (but doesn't account for the whole dominance effect for example!). It is important to realize that the total phenotypic variances V_P are not comparable between a design using repeated measures and mean value for each individual (taking the mean reduces V_P).

3.3.2 Adding fixed factors

The animal model is in essence a mixed model, allowing also for fixed effects. It can be interesting to add fixed factors to the model, in order to account for some biological or design-related issues likely to bias our heritability estimation. However it is necessary to recall that including a fixed factor in a model involve the reduction of the residual variance. Each added fixed effect thus possibly involves an artificial raise of the heritability. The aim behind the heritability estimation thus need to be closely examined before adding any fixed effect in the model. For more information about this issue, see the publication of Wilson (2008).

3.3.3 Generalizing the model

Just as the mixed models can be generalized in Generalized Linear Mixed Models (GLMMs), it is possible to generalize the animal model to fit several statistical distributions. Various things

are possible, but we will there focus on binary distribution. In that case, the model is defined on an underlying latent variable l (this is equivalent to a threshold model):

$$l_i = \mu + a_i + e_i \tag{9}$$

We then use a probit link and a binomial distribution to fit binary data:

$$Y_i \sim \mathcal{B}(\text{probit}^{-1}(l_i)) \tag{10}$$

All this is quite common for binary GLMM. However, when calculating heritability, it is necessary to take into account a supplementary source of variance coming from the probit link[2]. We thus have:

$$\hat{h}^2 = \frac{\hat{V}_A}{\hat{V}_A + \hat{V}_R + 1} \tag{11}$$

3.4 Pros and cons

More statistical power, more flexibility... Using the entire pedigree of the population, the animal model has better resources toward a precise estimation of the heritability than the parent-offspring regression. It is also more accurate by taking into account inbreeding and any selection event occurring "since" the founders. Adding random effects also allows for explicitly model dominance effects, common environment effects (nest, habitat, year...) or (non transgenerationnal) parental effects. A multivariate variant is also available, yielding genetic covariance between several traits.

...but everything is not perfect! The fact that animal model use the whole range of relationship in the population is its strength, but also its weakness. For example, the parent-offspring regression is only based on "vertical" relationships between individuals and thus ignores any non transgenerationnal effects: a dominance effect or a non transgenerationnal parental effect will induce no bias on heritability estimation[3]. As a consequence, it is highly important to properly think about all different sources of effects likely to induce a bias on heritability and (if the data structure is sufficient) to indicate them to the model using random effects. It is also important not to neglect simpler approaches such a parent-offspring regression (or half-sibs design if possible), at least as a checking step!

[2]This is justified by the fact that, to be strictly equivalent to a threshold model (where Y is 1 if $l > 0$), we need to include the "variance" of the link transformation into the total variance, which is 1 for a probit link and $\frac{\pi^2}{3}$ for a logit link.

[3]This reasoning is valid in the "horizontal" direction for the half-sibs design.

4 The animal model in practice using MCMCglmm

4.1 Some notions about Bayesian statistics

This section does not aim to be a lecture about Bayesian inference. However some basics are needed for a proper use of the MCMCglmm package. The output of a Bayesian inference is *posterior distribution*, i.e. a probabilistic distribution associating each value of a parameter to a probability (or degree of belief). The inference model is made of a *likelihood* function (in everything identical to the "classical" frequentist counterpart) and a *prior distribution* of the parameter(s) to be estimated. The likelihood model has been described in the previous section, we just need to choose the prior distributions for the parameters to be estimated. For more information about Bayesian and MCMCglmm, I advised the reader to consult Jarrod Hadfield's course notes, who developed the package.

What is MCMC estimation method? The aim of the MCMC algorithm (Markov Chain Monte Carlo) is to approximate the posterior distribution of the parameters. To do so it uses an algorithm based on the proposal, at each step, of a new value for a parameter, as a function of the value of the other parameters. After a convergence phase (often rather small), the MCMC algorithm tends to propose values within the posterior distribution of the parameters. Saving the value of the parameter at each iteration (or a subset, see later), we get a series of values drawing the posterior distribution of interest (just as a series of normally distributed values draw the famous bell curve). In order to get a good picture of this posterior distribution, we thus need a rather large set of draws (say 10,000), for which the correlation is negligible. Indeed, the successive iterations of the MCMC algorithm have the annoying tendency to be correlated from one to another. This is due to the fact that the proposal of a new value is based on the current value of the other parameters. This *autocorrelation* reduce the *effective size* of our sample. The effective size is the size of an uncorrelated sample (all draws are independent) equivalent to ours. For example, 10,000 draws highly correlated might have an effective size of 100 (i.e. they are equivalent to 100 independent draws). Since the iterations are correlated with those in proximity, we pick up in advance a thinning interval which reduces the autocorrelation (for example, we choose to keep only one iteration value every 10 iterations). This allows to spare memory and lightens further analysis. To put it in a nut shell, there are two important issues to monitor when using the MCMC:

- The convergence: It is important to check we waited long enough for the convergence (often called the *burn-in* period) to actually happen, before saving iteration values. Unfortunately, there is no way to tell in advance how long the burn-in has to be, so *post hoc* checks are used.

- The autocorrelation: In order to avoid autocorrelation issues (and mostly to lighten memory usage), it is possible to use a thinning interval. This way, we get a better effective sample size.

4.2 Choosing a prior distribution

4.2.1 What is a good prior?

A good prior distribution is often a *non informative* one, which means that the prior should not influence the estimated posterior distribution. In most cases, 'flat' priors check this criterion, but it is not a golden rule. It is actually quite difficult, without a prior sensitivity study, to predict the influence of the prior. Fortunately, the strength of the prior fades away whith the sample size of the data: with sufficient sample size, this prior issue becomes negligible.
For the definition of priors, the MCMCglmm package has specific distributions already implemented.

4.2.2 Prior distribution for a random effect variance

Regarding the prior distribution of variances, MCMCglmm uses an inverse-Gamma distribution, which is a common choice. In the package, the distribution is parametrized by two parameters `nu` and `V`[4]. A possible set of parameters would be `nu` $= 0.002$ and `V` $= 1$ (Fig. 1). Indeed, it allows for a weakly informative prior on variance components and U-shaped prior (with a very steep shape on the borders in 0 and 1) on the heritability. This choice is obviously not the only one, but it has the quality of being 'classical' (though not appropriate for too small variances, see Gelman (2006)) and generally weakly informative. To define this prior on variances using MCMCglmm, we use an R list as follows:

```
prior <- list(R = list(V=1, nu=0.002), G = list(G1 = list(V=1, nu=0.002)))
```

In this list, the `R` argument stand for the prior on the residual variance. The `G` (itself a list) is for random effects variance (called `G1`, `G2`, etc.). In presence of 3 random effects in the model, we need to define 3 priors in `G`:

```
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), G3=list(V=1, nu=0.002)))
```

4.2.3 Prior distribution of fixed effects

Regarding the prior distribution of fixed effects, the package defaults to a very wide Normal distribution, which is a relevant and consensual choice we don't need to argue or customize further.

[4]On the Wikipedia website, the inverse Gamma is parametrized differently using α and β notations. Since MCMCglmm notations are not usual, here are their 'translation': $\alpha = \frac{\texttt{nu}}{2}$ and $\beta = \frac{\texttt{nu}\times\texttt{V}}{2}$. Thus, a distribution parametrized by `nu` $= 0.002$ and `V` $= 1$ is actually an inverse Gamma$(0.001; 0.001)$.

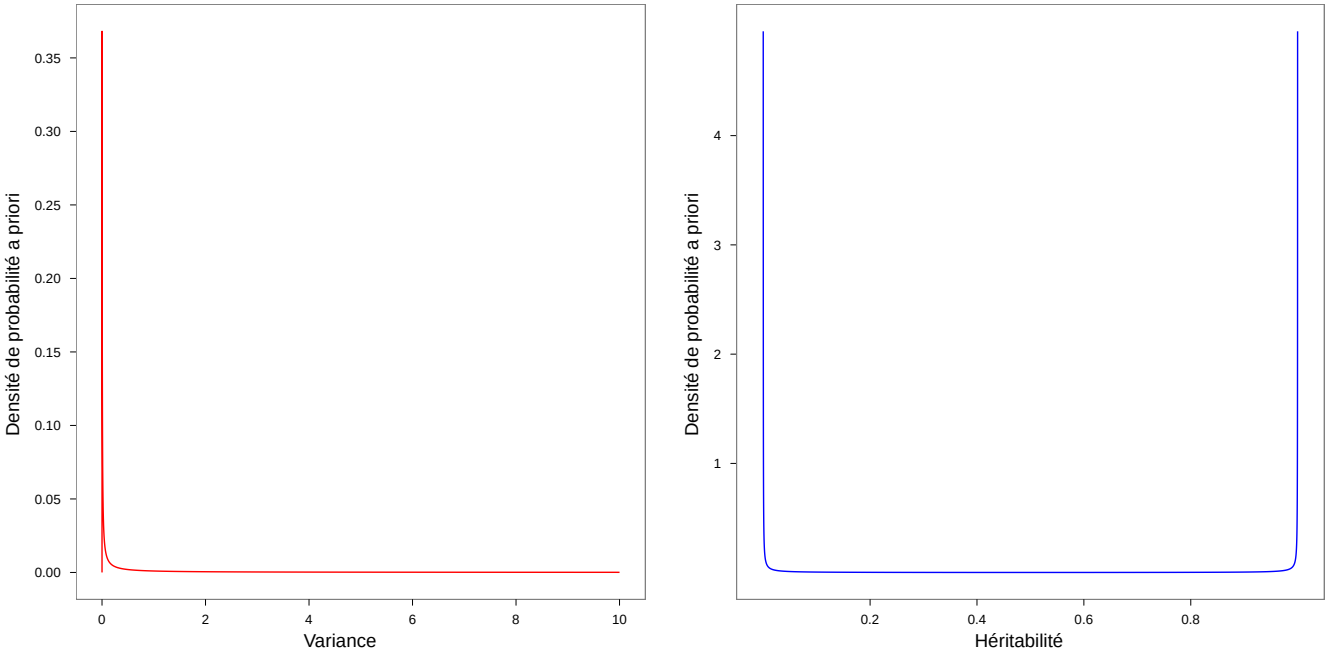

Figure 1: Shape of inverse Gamma$(0.001; 0.001)$ prior distribution on variance components (left) and induced shape of the prior for heritability (right).

4.3 `MCMCglmm()` function parameters

4.3.1 Setting up the pedigree

For `MCMCglmm()` to work properly, we need to provide it with a data frame describing the population pedigree of that kind:

```
pedigree <- read.table("pedigree.txt", header = T)
headtail(pedigree)
```

```
     animal sire  dam
1         1 <NA> <NA>
2         2 <NA> <NA>
3         3 <NA> <NA>
4         4 <NA> <NA>
...     ...  ...  ...
997     997  816  831
998     998  794  852
999     999  837  871
1000   1000  767  845
```

The first column has to be named `animal`. The founders (whose father and mother are unknown) must be placed of the top of the array, because a reproductive individual must appear before its offspring. The function allows for pedigree with missing parents (with `NA` in place of the parent's ID).
The data are also stored in a data frame:

```
data <- read.table("data.txt", header = T)
headtail(data)
```

```
     animal  phen
1         1 12.84
2         2 10.47
3         3 12.44
4         4 10.59
...     ...   ...
997     997 10.28
998     998 10.13
999     999 11.13
1000   1000 11.72
```

The column `animal` does not have to be ordered as in the pedigree. Phenotypic data are here stored in the `phen` column.

4.3.2 How to use `MCMCglmm()` function?

In order to fit a simple model, where we only estimate additive and residual variances, we call the function likewise:

```
prior <- list(R = list(V=1, nu=0.002), G = list(G1 = list(V=1, nu=0.002)))
model <- MCMCglmm(phen ~ 1, random = ~animal, family = "gaussian",
    prior = prior, pedigree = pedigree, data = data, nitt = 100000,
    burnin = 10000, thin = 10)
```

The fist argument `phen ~ 1` is a R formula giving the response variable (`phen`) according to fixed factors (here only the population mean μ of the trait). The argument `random = ~animal` set the random effects: `animal` is a reserved variable to fit an additive genetic effect (in other words, we are stating the model we want to estimate V_A). The argument `family` set the distribution to use for the data (here `gaussian` for a normally distributed trait). The argument `prior` calls the list of parameters for prior distributions stored in the variable `prior`. Arguments `pedigree` and `data` speak for themselves. Finally, the three last arguments are to set up the properties of the MCMC: `nitt` is the total number of iterations, `burnin` is the number of iterations to drop at the beginning (convergence) and `thin` is the number of iterations stored in memory (here, one every ten iterations).

4.4 Results and diagnostic of the MCMC output

4.4.1 Calling the function

The function is called likewise:

```
prior <- list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1,
    nu = 0.002)))
model <- MCMCglmm(phen ~ 1, random = ~animal, family = "gaussian",
    prior = prior, pedigree = pedigree, data = data, nitt = 100000,
    burnin = 10000, thin = 10)
```

```
                    MCMC iteration = 0
                    MCMC iteration = 1000
                    MCMC iteration = 2000
                    MCMC iteration = 3000
                    ...                  ...
                    MCMC iteration = 99000
                    MCMC iteration = 100000
```

Don't forget to store the output of the function in a variable (here `model`), in order to have access to it afterwards. The computation might take some time (even quite a long time), depending of the data, the parameters and, of course, the computer capacities.

4.4.2 Diagnostic of the MCMC

Before even looking at the estimates, the first thing to do is to check the behaviour of our MCMC algorithm. To do so, we need to focus on convergence and autocorrelation of our 'chain' of samples. The output `model` has two main components, which are `model$Sol` and `model$VCV` (respectively for fixed effects and random effects variances). First of all, let's look at the 'trace' of our 'chain' (Fig. 2):

```
plot(model$Sol)
plot(model$VCV)
```

Each of the three couples of graphs shows us the trace (left), i.e. the evolution of the sampled values along the iterations. It allows us to check the convergence (we should not see any trend in the trace) and that autocorrelation is weak (the values are widely spread). On the right of these graphs, we have an estimation of the posterior density function for each component (`Intercept`, `animal` and `units`). We can also have a look at the autocorrelation:

```
autocorr.diag(model$Sol)
```

```
          (Intercept)
Lag 0    1.0000000000
Lag 10   0.0157137339
Lag 50  -0.0035557380
Lag 100 -0.0013154271
Lag 500  0.0004660904
```

```
autocorr.diag(model$VCV)
```

```
            animal        units
Lag 0   1.00000000 1.0000000000
Lag 10  0.40759941 0.2729110582
Lag 50  0.02437235 0.0109679990
Lag 100 0.01591581 0.0043086559
Lag 500 0.01007559 0.0001932177
```

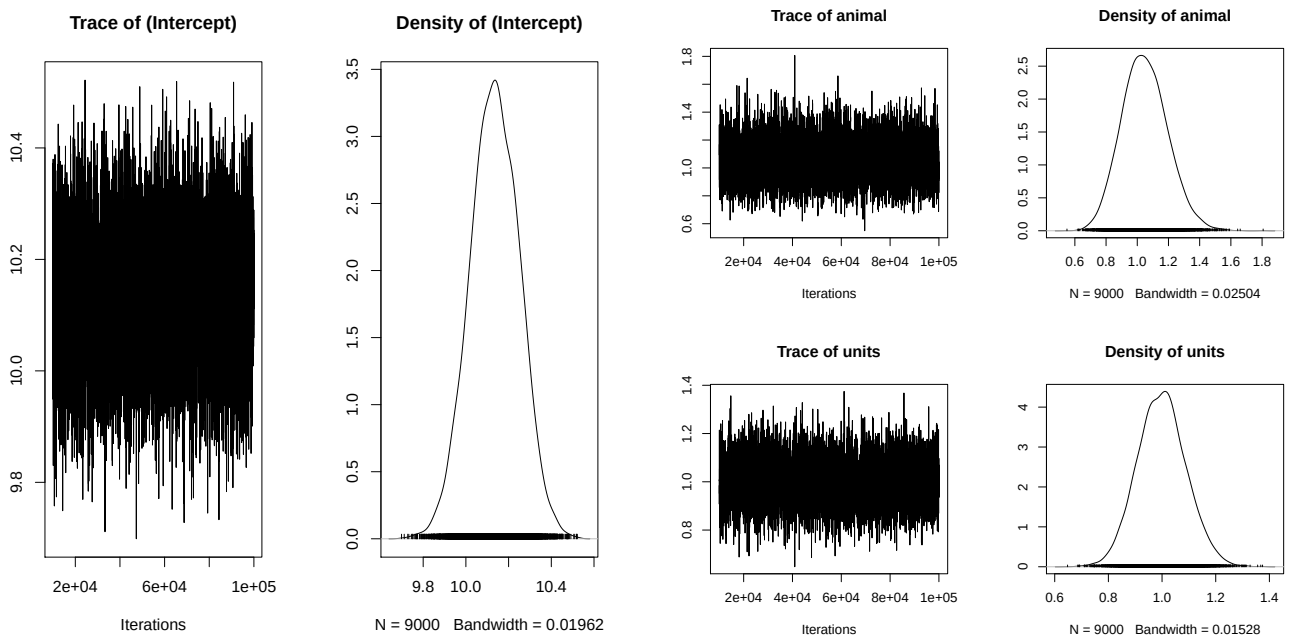

Figure 2: Trace of the mean μ (or intercept, left) and the variances (right). The term `animal` refers to V_A and the term `units` refers to V_R.

Here `Lag 10` stands for 'autocorrelation every 10 iteration values'. Since our `thin` parameter was 10, this refers actually to the correlation of every sampled value with the following one. We can see that there little autocorrelation on the mean (`Intercept`). On the contrary, the autocorrelation on variance components becomes negligible only with a lag of 50. Theoretically, it should be good to re-run a longer MCMC to increase the effective sample size. In practice, an autocorrelation less than 0.1 for the first `Lag` (i.e. from one sampled value to the other) is reasonable. Autocorrelation *per se* is not really an issue, but it does shrink the effective sample size. Let's check this out:

```
effectiveSize(model$Sol)
```

```
(Intercept)
   8720.559
```

```
effectiveSize(model$VCV)
```

```
  animal    units
3787.308 4747.467
```

We see that the effective sample size of the mean (`Intercept`) is larger than the effective sample size for variance components, for which the autocorrelation is greater. We need to recall how important is the effective size parameter: the aim of our sample (or 'chain') is to estimate

the posterior distribution of the parameter of interest. To do so, we need the largest number of independent values as possible, which means a large effective sample size. In practice, an effective size above 1,000 is recommended, 10,000 being a quite comfortable goal. Here, we get medium effective size. The autocorrelation is still a bit too strong, but considering the shape of traces and posterior density (Fig. 2, the curves are relatively symmetrical, unimodal and not aberrant), we will content ourselves with it for this tutorial. For more confidence in our estimates, we could run the MCMC for a longer number of iterations (`nitt`), and maybe increase the `thin` parameter a bit to save memory. This would help to get a larger effective sample size.

Using MCMCglmm, the convergence is often really fast. Here, it happens from the very first iterations (Fig. 3):

```
modelburnin <- MCMCglmm(phen ~ 1, random = ~animal, family = "gaussian",
    prior = prior, pedigree = pedigree, data = data, nitt = 5000,
    burnin = 1, thin = 1)
plot(modelburnin$VCV)
```

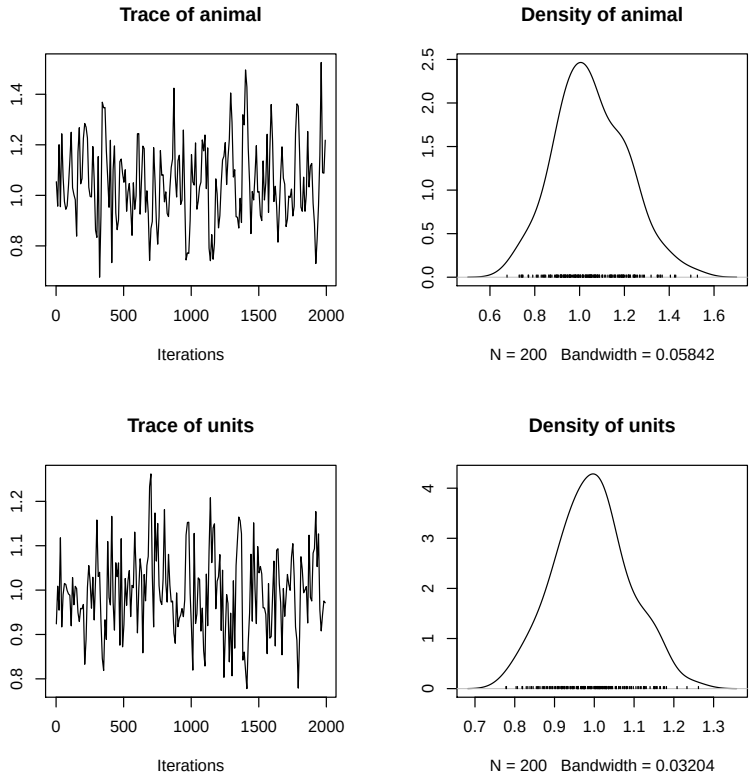

Figure 3: Trace of the variances with only 5000 iterations.

Note that there is diagnostic tests of convergence, as the Heidelberg stationarity test (here, to bend the rules, the *p-values* must exceed 0.05):

```
heidel.diag(model$VCV)
```

```
       Stationarity start     p-value
       test         iteration
animal passed       1         0.613
units  passed       1         0.993

       Halfwidth Mean  Halfwidth
       test
animal passed    1.048 0.00416
units  passed    0.999 0.00242
```

These tests do not however spare one of a graphical check using the trace and **certainly not** from an autocorrelation study (convergence and autocorrelation issues are separate!).

4.4.3 Results from a MCMC algorithm

The output of the `MCMCglmm()` function has the following structure:

```
summary(model)
```

```
 Iterations = 10001:99991        #MCMC parameters
 Thinning interval  = 10
 Sample size  = 9000  #Actual sample size (not effective)

 DIC: 3203.707         #Somewhat like AIC

 G-structure:  ~animal        #Random effects section

       post.mean l-95% CI u-95% CI eff.samp
animal     1.048   0.7683    1.334     3787  #Here is an effective sample size

 R-structure:  ~units        #Residual variance section

      post.mean l-95% CI u-95% CI eff.samp
units    0.9989   0.8256    1.178     4747

 Location effects: phen ~ 1         #Fixed effects section

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)     10.13     9.91    10.35     8721 <1e-04 ***
```

The first part reminds us the characteristics of the sample. Then, the function gives the DIC (Deviance Information Criterion) associated to the model. This DIC may be used for model selection, like the AIC. Be careful however, because the behaviour of this statistical tool has still to be precisely evaluated. The rest of the output is separated into three sections: the first, `G-structure`, informs us about the variance estimates of the random effects (here, we only have V_A, called `animal`); the second, `R-structure`, is about the residual variance estimation

(V_R here called `units`); and the third section, `Location effects`, gives us the results regarding the fixed effects (here only the population mean μ called `Intercept`).
Concerning the two first parts, we get various information about our estimations. The column gives us the posterior mean of the posterior distribution (actually, it is simply the mean of the MCMC sample). However, it has to be noted that the median is sometimes a better summary statistics than the mean, especially for very asymmetric posterior distribution. We then have the lower and upper limits of the 95% credible interval (i.e. our parameter has a posterior probability of 0.95 to lie within this interval). Finally, the function gives us the effective sample size associated to the parameter. Concerning the fixed effects, we have the same information plus a `pMCMC` column. This latter is the posterior probability associated to the event: "the parameter is not different from zero"[5]. It is not a *p-value*, but provides the same kind of information. Here, the `pMCMC` is very weak indicating that the population mean is very different from zero (which also makes sense when looking at the credible interval).

4.5 Computation of heritability estimate

One big advantage of the MCMC is the possibility to straightforwardly compute the posterior distribution of any function of the variance components. Thus we have a simple way to obtain the posterior distribution of the heritability:

```
herit <- model$VCV[, "animal"]/(model$VCV[, "animal"] + model$VCV[, "units"])
```

We can then use all the tools we just mentioned on the heritability:

```
effectiveSize(herit)
```

```
    var1
3494.684
```

```
mean(herit)
```

```
[1] 0.5104629
```

```
HPDinterval(herit)        #Display 95% credible interval
```

```
         lower     upper
var1 0.4062324 0.6104063
attr(,"Probability")
[1] 0.95
```

We can also plot the trace and the density function using `plot()` (Fig. 4). The heritability of the trait `phen` is about 0.51 with 95% of probability to lies between 0.40 and 0.61.

[5]To be precise, the event is "the parameter is negative" if the posterior mean is positive and "the parameter is positive" if the mean is negative.

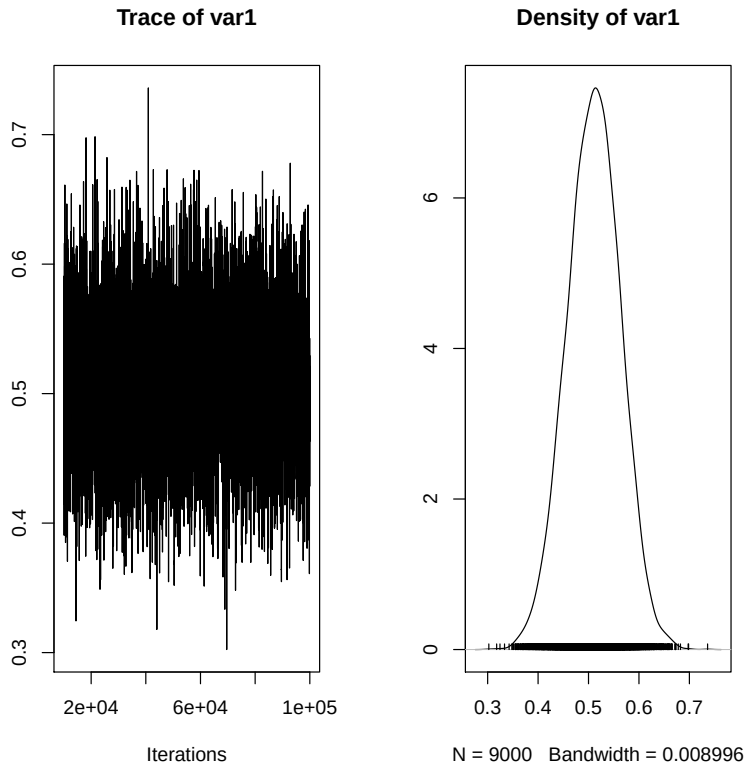

Figure 4: Trace (left) and posterior density (right) of the heritability.

4.6 Adding random effects

We will here use a common environmental effect, the breeding location (here called `patch`) of the individuals:

```
pedigreeEC <- read.table("pedigreeEC.txt", header = T)
dataEC <- read.table("dataEC.txt", header = T)
head(dataEC)
```

```
  animal   phen patch
1      1 10.245   P18
2      2  9.785   P12
3      3 13.386   P25
4      4 10.571   P18
5      5  8.278    P5
6      6  8.444   P23
```

Because of a dispersal rate of 50%, the individuals have one chance over two to breed in the same patch as their parents. Yet the characteristics of the patch have an effect on their phenotypes (say a spatial variability makes some patches better than others in terms of resources). We can first look into what happens if we ignore this effect:

```
prior <- list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1,
    nu = 0.002)))
modelnaif <- MCMCglmm(phen ~ 1, random = ~animal, family = "gaussian",
    prior = prior, pedigree = pedigreeEC, data = dataEC, nitt = 1e+05,
    burnin = 10000, thin = 10)
```

After all the usual checks (!!), we calculate the heritability:

```
heritnaif <- modelnaif$VCV[, "animal"]/(modelnaif$VCV[, "animal"] +
    modelnaif$VCV[, "units"])
```

We can then compute the posterior mean and credible interval:

```
mean(heritnaif)
```

```
[1] 0.6419418
```

```
HPDinterval(heritnaif)
```

```
         lower     upper
var1 0.5537633 0.7365198
attr(,"Probability")
[1] 0.95
```

So, here we estimate a heritability around 0.64. The value 0.5 does not even lie within the credible interval. However, when we add the `patch` random effect into the model (without forgetting to change the priors to include a new variance component!):

```
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)))
modelEC <- MCMCglmm(phen ~ 1, random = ~animal + patch, family = "gaussian",
    prior = prior, pedigree = pedigreeEC, data = dataEC, nitt = 1e+05,
    burnin = 10000, thin = 10)
```

We still do the usual checks and calculate the heritability including the `patch` effect:

```
heritEC <- modelEC$VCV[, "animal"]/(modelEC$VCV[, "animal"]
    + modelEC$VCV[, "units"] + modelEC$VCV[, "patch"])
```

We here see that the heritability is estimated around 0.5 (which is the actual value used to simulate data):

```
mean(heritEC)
```

```
[1] 0.5200806
```

```
HPDinterval(heritEC)
```

```
         lower     upper
var1 0.4070969 0.6282084
attr(,"Probability")
[1] 0.95
```

Through this little example, we can see the importance of a well defined model, including all the environmental effects (and other kinds!) to correctly estimate heritability.

4.7 Adding a dominance effect

4.7.1 Computation of the dominance matrix D

When we specify a random effect to the function `MCMCglmm()`, this last one compute itself the corresponding variance-covariance matrix. For example, using the pedigree, the function automatically computes the inverse of the additive genetic matrix **A** corresponding to the `animal` effect. Yet nothing is implemented in the function to calculate the variance-covariance matrix associated to a dominance effect. We thus have to compute ourselves this dominance matrix. In absence of inbreeding, this matrix is made of 1 on the diagonal, 0.25 when individuals of the row and the column are full-siblings and 0 elsewhere. Note that, from the dominance perspective, parent/offspring and half-sibling relationships are null (such that only full-siblings matter)[6]. In order to compute this dominance matrix, a R package exits, called nadiv and developed by Wolak (2012):

```
pedigreedom<-read.table('pedigreedom.txt',header=T)
library(nadiv)
listD<-makeD(pedigreedom)
```

```
[1] "starting to make D"
[1] "D made"
[1] "starting to invert D"
[1] "done inverting D"
```

```
Dinv<-listD$Dinv
```

We obtain the `Dinv` matrix (i.e. the inverse of the dominance matrix). The non inverse dominance matrix can be obtained using `listD$D`. Here, the data have a supplementary column, identical to the `animal` column:

```
datadom<-read.table('datadom.txt',header=T)
headtail(datadom)
```

```
      animal     dom  phen
1    R187557 R187557 10.69
2    R187559 R187559 11.21
3    R187568 R187568 12.17
4    R187518 R187518 11.59
...      ...     ...   ...
1037 R187732 R187732 11.14
1038 R187073 R187073  8.47
1039 R187905 R187905  9.35
1040 R187002 R187002 13.55
```

[6] Actually, double first cousins have a dominance correlation of $\frac{1}{16}$, but they are quite improbable in nature.

4.7.2 How to implement dominance effects in MCMCglmm?

To implement dominance effects, we use the argument `ginverse` of the function `MCMCglmm()`, which allows to 'customize' a variance-covariance matrix associated to a random effect. We link the `Dinv` matrix to our dominance effect:

```
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)))
modeldom1<-MCMCglmm(phen~1,random=~animal+dom,ginverse=list(dom=Dinv),
  family="gaussian",prior=prior,pedigree=pedigreedom,data=datadom,
  nitt=100000,burnin=1000,thin=10)
```

Note that the data have a `dom` column identical to the `animal` column. This former is needed for the function what `dom` refers to here. However, a quick look at the results shows a slight issue:

```
summary(modeldom1)
```

```
 Iterations = 1001:99991
 Thinning interval  = 10
 Sample size  = 9900

 DIC: 3490.966

 G-structure:  ~animal

       post.mean l-95% CI u-95% CI eff.samp
animal     1.811    1.126    2.499     1705

              ~dom

    post.mean  l-95% CI u-95% CI eff.samp
dom    0.5699 0.0002564    1.921    25.19

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units      1.45 0.007292    2.132     28.4

 Location effects: phen ~ 1

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)     9.968    9.763   10.189     9613 <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, contrary to the previous cases, the autocorrelation becomes so huge that the effective sample size of our variances are ridiculously small (almost 10,000 iterations for only 25 independent values!). A glance at the trace of the MCMC gives us a clue about what is going on (Fig.5, left): sometimes the dominance effect variance (`dom`) or the residual variance (`units`) get 'stuck' on zero... In this situation, the MCMC is, in a way, lacking of variance to change its state and the autocorrelation increases a lot! We can see the effects on the posterior distribution of both variances (Fig.5, left). Even for the residual variance, which is less 'stuck' in zero, we see a secondary mode of density in zero. This is a perfect opportunity to show what happens with a longer run. We will choose a thinning interval 50 larger (500) and let the MCMC run for 500 times more iterations:

```
modeldom1<-MCMCglmm(phen~1,random=~animal+dom,ginverse=list(dom=Dinv),
  family="gaussian",prior=prior,pedigree=pedigreedom,data=datadom,
  nitt=50000000,burnin=1000,thin=500)
```

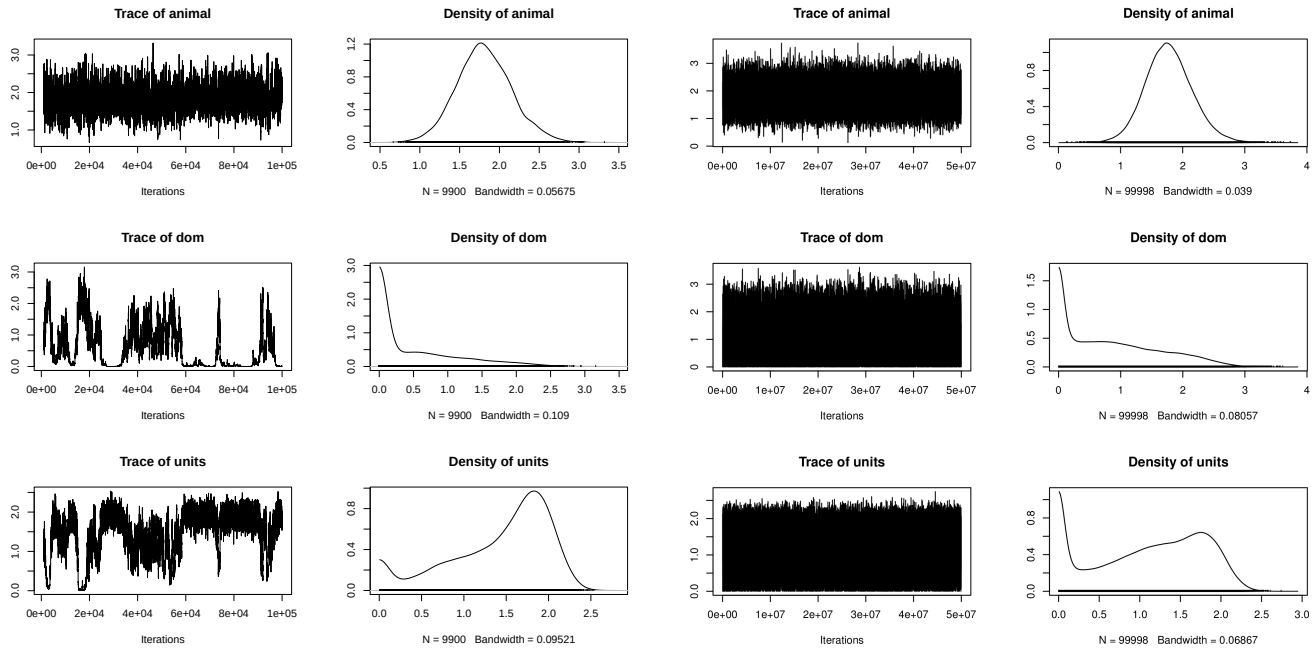

Figure 5: Trace and posterior density of variance for the first (left) and second (right, with more iterations) MCMC runs with dominance. The term `animal` refers to V_A, `dom` to V_D and `units` to V_R.

The effective sample size is a bit more satisfying:

```
summary(modeldom2)
```

```
Iterations = 1001:49999501
Thinning interval  = 500
Sample size  = 99998

DIC: 2718.913
```

```
 G-structure:  ~animal

       post.mean l-95% CI u-95% CI eff.samp
animal     1.778    1.054    2.554    86135

               ~dom

    post.mean  l-95% CI u-95% CI eff.samp
dom     0.928 0.0001706    2.323     7522

 R-structure:  ~units

      post.mean  l-95% CI u-95% CI eff.samp
units     1.161 0.0001896    2.052     7331

 Location effects: phen ~ 1

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)     9.965    9.750   10.180    1e+05 <1e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This example allows us to stress out the main drawbacks of the MCMC as an estimation method. First, the comparison of the two `summary` shows how much the estimates might rely on the fact that the MCMC works smoothly (we speak about good or poor mixing properties of the MCMC chains). Importantly, the DIC is quite sensitive to a bad mixing (3490 for the badly mixed chain against 2719 here). Second, it is really important to use everything to check the mixing of the chain. Our effective size are comfortable, they however are to be compared to other parameters' (for example, `animal` has a far better mixing than `units` with 86,135 against 7331). The autocorrelation is indeed very high despite our efforts:

```
autocorr.diag(modeldom2$VCV)
```

```
               animal          dom         units
Lag 0     1.000000000  1.000000000  1.0000000000
Lag 500   0.014052835  0.694609597  0.7002058074
Lag 2500  0.003985196  0.392770278  0.3964285607
Lag 5000  0.007108644  0.236290457  0.2397001041
Lag 25000 0.001947910 -0.001554258 -0.0003827729
```

This autocorrelation is not without influence on our variances estimations (Fig.5, right). The trace is here thicker, and it is difficult to distinguish if the variances get stuck in zero, but a look at the posterior distributions (Fig.5, right) shows that letting the model run for longer did

not change much this behaviour... Nevertheless, this example is a good demonstration of how the autocorrelation might bias estimates in the MCMC![7]

4.7.3 How to improve the mixing of our chains?

Adding a dominance effect certainly had complicated the model. However, the issue here raise from two main sources. First, we used small variances ($V_A = 2$, $V_D = 1$ et $V_R = 1$) to simulate data. With larger variances, we might have not encountered this type of problems. Second, a close population model has been used to simulate the pedigree, leading to a strong inbreeding, which makes the dominance matrix more complex.

When such autocorrelation issues arise, is usual to change a bit the MCMC algorithm. Here, using MCMCglmm, we are a bit limited, but it is possible to use the so-called *parameter extension* (Gelman 2006) to allow for more flexibility during MCMC iterations. The idea behind the parameter extensions lies into splitting the random effect u_i into two independent components:

$$\begin{aligned} u_i &= \alpha\, \eta_i \\ \text{with } \eta_i &\sim \mathcal{N}(0; V_\eta) \end{aligned} \tag{12}$$

Doing so, we implicitly split the variance of the random effect u in two: $V_u = \alpha^2 V_\eta$. We have the following prior distributions for α and V_η:

$$\begin{aligned} \alpha &\sim \mathcal{N}(0; V_\alpha) \\ V_\eta &\sim \text{Inverse-Gamma}(\texttt{V}; \texttt{nu}) \end{aligned} \tag{13}$$

We are a bit technical here, but it is sufficient to keep in mind that we implicitly here define a prior distribution on V_u such as V_u/V_α follows a Fisher distribution of degree of freedom 1 and `nu`.

Our problem comes from the fact that our variances are weak. Indeed, the inverse-Gamma distribution has the (really) tedious trends to 'pull' too strongly around zero. Yet, our new 'Fisher' prior puts less weight in zero. Better, the parameter extension allows the MCMC to depart more easily from zero when it is stuck. Let's try this new prior:

```
prior_ext<-list(R=list(V=1,nu=1),
  G=list(G1=list(V=1,nu=1,alpha.mu=0,alpha.V=1000),
  G2=list(V=1,nu=1,alpha.mu=0,alpha.V=1000)))
```

[7] It has to be noted here that reducing the autocorrelation by using a bigger thinning interval might not be a good solution: indeed, if the MCMC get stuck in zero for the variances, then even in getting rid of most of the iterations, we still will have an artificially high probability to sample near zero if we sample during a 'stucked' phase. Increasing the thinning interval will reduce the issue, but not overcome it!

A few things. First, the parameter extension is not (yet) available for the residual variance. However, we have fixed the `nu` parameter to 1 in order to put less weight in zero[8]. Regarding the random effects, the parameter extension is defined by the parameters `V` and `nu` (pour V_η) and `alpha.mu` and `alpha.V` (for α). Let's look at the results obtained using this prior (with 5 times less iterations than for `modeldom2`):

```
modeldom3<-MCMCglmm(phen~1,random=~animal+dom,ginverse=list(dom=Dinv),
  family="gaussian",prior=prior_ext,pedigree=pedigreedom,data=datadom,
  nitt=10000000,burnin=1000,thin=500)
```

First, we observe that the estimates are closer to the simulated values, with correct effective size:

```
summary(modeldom3)
```

```
 Iterations = 1001:9999501
 Thinning interval  = 500
 Sample size  = 19998

 DIC: 3487.879

 G-structure:  ~animal

       post.mean l-95% CI u-95% CI eff.samp
animal     1.767    1.066    2.523    19998

               ~dom

    post.mean  l-95% CI u-95% CI eff.samp
dom      1.01 1.203e-08    1.954    16289

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units     1.098   0.2538    1.948    16413

 Location effects: phen ~ 1

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)     9.962    9.756   10.184    20830 <5e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

These better results are explained by a weaker autocorrelation (but let's keep in mind that the thinning interval is 500 here):

[8]Beware! This prior is an inverse-Gamma$(0.5; 0.5)$ which is not really 'classical'! Its use needs to be relevantly justified!

```
autocorr.diag(modeldom3$VCV)
```

```
                animal         dom       units
Lag 0       1.000000000 1.000000000 1.000000000
Lag 500     0.009773732 0.088038981 0.098434370
Lag 2500   -0.012963423 0.008679641 0.007021305
Lag 5000    0.002886355 0.009259605 0.005525710
Lag 25000  -0.004708470 0.001587171 0.004412308
```

The posterior densities (Fig.6) show 'prettier' curves. We however see that the `dom` component still has a secondary mode in zero (which could be not so abnormal after all). The parameter extension allows thus for better mixing properties for our MCMC chains, but it is not to forget the informative character of our priors. It is often hard to find a trade-off between a non informative prior and MCMC mixing properties, especially while speaking of variance components. Unfortunately, each dataset is unique and no universal consensus exists. Doing a sensitivity analysis (by using different priors and testing the sensitivity of the posterior distribution to the priors) stays the best method to avoid any suspicion!

5 Going further...

5.1 Generalized animal model: binary data

When using binary data (presence/absence of a character or success/failure for example), some caution is needed. We will illustrate this with a new binary example:

```
pedigreebin <- read.table("pedigreebin.txt", header = T)
databin <- read.table("databin.txt", header = T)
head(databin)
```

```
  animal phen
1      1    0
2      2    1
3      3    1
4      4    0
5      5    1
6      6    0
```

5.1.1 Defining a prior

The choice of the prior requires already some modifications. Indeed, it is impossible to evaluate simultaneously the additive genetic variance V_A and the residual variance V_R[9]. Because of this

[9]Indeed, whatever the value of $V_P = V_A + V_R$, the variance of the binary data will always be $p(1-p)$ where p is the probability of success (and does not depend on V_P).

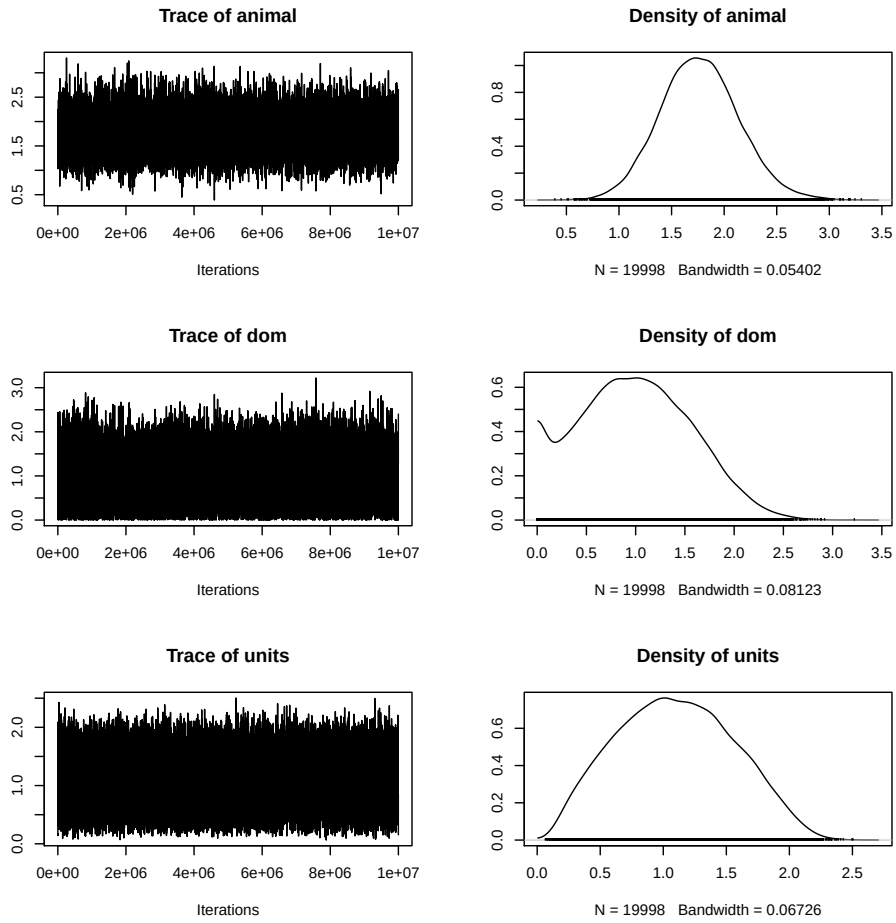

Figure 6: Trace and posterior density of the variances for the MCMC run using parameter expansion. The term `animal` refers to V_A, `dom` to V_D and `units` to V_R.

identifiability issue, we usually fix V_R to 1 and estimate V_A solely (when more random factors are included, they can be estimated as well, but V_R still must be fixed to 1[10]. Because of this, it becomes tricky to define a good prior for V_A (and other possible random effects variance). On Fig. 7, we see the difference between the prior we use so far for V_A and a new prior (a χ^2 with 1 degree of freedom). We see that neither is quite perfect. Indeed the first puts too much weight in 1 while the second puts too much weight in 0. However, a glance at the cumulative density function (Fig. 7, right) shows that the first really puts almost all the weight in 1 while the probabilistic weight of the second is more spread between 0 and 1. We thus advice to use the χ^2 prior distribution when estimating heritability of binary data. This is not an ideal choice, but unfortunately, we do not have too much of a choice for prior distributions when using MCMCglmm. As always though, the influence of the prior distribution fade away with a sufficient sample size.

[10]It is however not recommended to include lots of random factors with binary data: because of the lack of information provided by this kind of data, the variance components estimation might become unstable

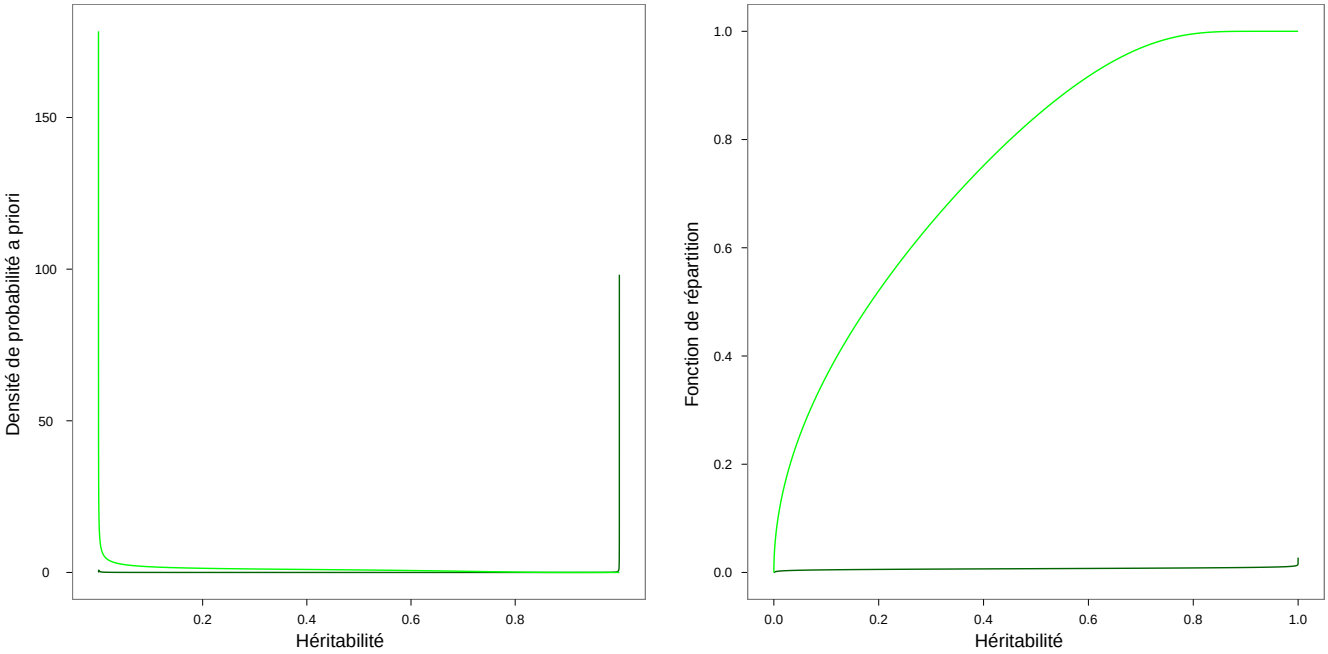

Figure 7: Prior density function (left) and prior cumulative density function (CDF, right) of the heritability using two kind of priors for additive genetic variance V_A: (i) inverse Gamma (dark green) and (ii) χ^2 with 1 degree of freedom (green).

To use this new prior, we use the parameter extension with the supplementary parameters `alpha.mu` and `alpha.V`:

```
prior <- list(R = list(V = 1, fix = 1), G = list(G1 =
    list(V = 1, nu = 1000, alpha.mu = 0, alpha.V = 1)))
```

Then, for the model, all we need is to change the parameter `family` for a binomial distribution. In `MCMCglmm()`, there is two kinds of family that fit our needs: `categorical` which uses a *logit* link and `ordinal` which uses a *probit* link (and consider that the categories are ordered, but that has no effect on binary data). Since the probit link is more natural for animal models, we will choose the last one. At last, it is quite common to get strong autocorrelation in the MCMC chains with binary data. We thus multiply the `thin` parameter by 10 and the total number of iterations (`nitt`) accordingly:

```
modelbin <- MCMCglmm(phen ~ 1, random = ~animal, family = "ordinal",
    prior = prior, pedigree = pedigreebin, data = databin, nitt = 1e+06,
    burnin = 10000, thin = 100)
```

```
                    MCMC iteration = 0
 Acceptance ratio for latent scores = 0.000647

                    MCMC iteration = 1000
 Acceptance ratio for latent scores = 0.443566
```

```
                      MCMC iteration = 2000
  Acceptance ratio for latent scores = 0.442072

       ...                 ...                 ...

                      MCMC iteration = 999000
  Acceptance ratio for latent scores = 0.468743

                      MCMC iteration = 1000000
  Acceptance ratio for latent scores = 0.471385
```

For some reasons, the function gives us some information about the acceptance ratio of the latent scores, which is recommended to be around 0.44. After the usual checking steps, we compute the posterior distribution of the heritability likewise (not forgetting to add the last 1 due to the 'variance' of the probit link, see section 3.3.3):

```
heritbin <- modelbin$VCV[, "animal"]/(modelbin$VCV[, "animal"] +
    modelbin$VCV[, "units"] + 1)
mean(heritbin)
```

```
[1] 0.5322876
```

```
HPDinterval(heritbin)
```

```
         lower     upper
var1 0.3775965 0.6869016
attr(,"Probability")
[1] 0.95
```

Note that the effective size for the heritability estimate is not really large, despite a thinning interval of 100, suggesting an quite strong autocorrelation:

```
effectiveSize(heritbin)
```

```
    var1
3372.556
```

5.2 A multi-traits model

5.2.1 Why a multi-traits model?

It might be interesting to address the genetic basis of several phenotypic traits at the same time, notably to avoid any confounding effect between these traits, but also because the genetic correlation may be a parameter of biological interest! Here, we will focus on bivariate model[11] composed of measurements on two different phenotypic traits:

```
pedigreemulti<-read.table('pedigreemulti.txt',header=T)
datamulti<-read.table('datamulti.txt',header=T)
headtail(datamulti)
```

```
     animal phen1 phen2
1         1 11.01 -1.86
2         2 10.05 -0.77
3         3   9.3  0.84
4         4  9.87  0.77
...     ...   ...   ...
997     997   9.9  0.06
998     998 12.82 -0.44
999     999  9.25  1.03
1000   1000  7.74  0.48
```

5.2.2 A word on priors

As always, difficulties arise when it comes to the choice of a good prior. Here, we do not talk about simple variances anymore, but about a variance-covariance matrix. In two dimensions, such a matrix is shaped like:

$$\begin{pmatrix} V_1 & C_{1,2} \\ C_{1,2} & V_2 \end{pmatrix} \tag{14}$$

where V_i is the variance (for example additive genetic) associated to the trait i and $C_{1,2}$ is the covariance (for example genetic) between trait 1 and trait 2. We thus need a kind of prior distribution relevant for matrices. To do so, we use a matrix distribution called *inverse Wishart*. Here, we have two choices. The following prior:

```
prior<-
  list(R=list(V=diag(2)*(0.002/1.002),nu=1.002),
  G=list(G1=list(V=diag(2)*(0.002/1.002),nu=1.002)))
```

is roughly equivalent to the one we used so far, which is a (marginal) inverse-Gamma(0.001,0.001) on variances; and the (marginal) prior distribution of correlation follows to some extent a

[11] We do not recommend to fit models with a too large number of traits, unless having a very large sample size and enough time for the MCMC to run!

Beta(0.001,0.001) distribution[12]. However, this inverse Wishart distribution is not well defined and might cause instability in the MCMC. A more 'proper' parametrization would be the following:

```
prior<-
  list(R=list(V=diag(2)/2,nu=2),
  G=list(G1=list(V=diag(2)/2,nu=2)))
```

However, we here define (marginal) inverse Gamma(0.5,0.5) and (still to some extent) Beta(0.5,0.5) distributions for variances and correlation. Yet this prior is more informative than the last one, notably for small variances. We here thus have a choice to make according to sample size, expected variance estimations and stability of the MCMC. In the following and to stay consistent with the previous parts, we will use the first prior.

5.2.3 Using `MCMCglmm()` with a multi-traits model

On its multivariate shape, the function must be used likewise:

```
modelmulti<-MCMCglmm(cbind(phen1,phen2)~trait-1,random=~us(trait):animal,
  rcov=~us(trait):units,family=c("gaussian","gaussian"),prior=prior,
  pedigree=pedigreemulti,data=datamulti,nitt=100000,burnin=10000,thin=10)
```

Here, the 'response variable' is an array with two columns binded using the `cbind()` function, each line containing the measure of both traits for one individual. On the other side of the formula, the notation `trait-1` allows us to estimate the mean of each trait, instead of the contrast between the two traits. Note that the argument `family` now requires a vector with the data distribution of each trait. Finally, the argument `random` has changed and a new argument `rcov` appeared: these two arguments allow us to define the structure of the variance-covariance matrix for the random effects (`random`) or the residual variances (`rcov`). The command `us(trait):animal` states that we define these matrix exactly as written is Eq.14. This is the role of the function `us()`. If we simply had used `random=~animal`, we would have defined the following variance-covariance matrix:

$$\begin{pmatrix} V & V \\ V & V \end{pmatrix} \tag{15}$$

Doing so, we assume that both traits have the same variance ($V_1 = V_2 = V$) and are perfectly correlated ($\rho_{1,2} = 1$). Beside the function `us()`, the function `idh()` exists, which fixes the

[12] In fact, from the covariance $C_{1,2}$, we can define the correlation as $\rho_{1,2} = \frac{C_{1,2}}{\sqrt{V_1 V_2}}$. We then know that if the parameter `V` is a diagonal matrix, we have $\frac{\rho_{1,2}+1}{2} \sim \text{Beta}(\frac{\texttt{nu}-1}{2}, \frac{\texttt{nu}-1}{2})$

covariances to 0. Indeed the call `random=~idh(trait):animal` defines the following matrix:

$$\begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix} \tag{16}$$

5.2.4 Results of a bivariate MCMC

The previous MCMC run give us the following results:

```
summary(modelmulti)
```

```
 Iterations = 10001:99991
 Thinning interval  = 10
 Sample size  = 9000

 DIC: 6050.479

 G-structure:  ~us(trait):animal

                   post.mean l-95% CI u-95% CI eff.samp
phen1:phen1.animal    1.1564   0.8659   1.4589   3222.0
phen2:phen1.animal    0.7026   0.5017   0.8995   1513.4
phen1:phen2.animal    0.7026   0.5017   0.8995   1513.4
phen2:phen2.animal    0.5205   0.3377   0.7147    446.3

 R-structure:  ~us(trait):units

                  post.mean l-95% CI u-95% CI eff.samp
phen1:phen1.units    0.9461  0.77350   1.1254   3560.0
phen2:phen1.units    0.1452  0.02207   0.2668   1350.3
phen1:phen2.units    0.1452  0.02207   0.2668   1350.3
phen2:phen2.units    0.9579  0.80731   1.1062    611.7

 Location effects: cbind(phen1, phen2) ~ trait - 1

           post.mean l-95% CI u-95% CI eff.samp  pMCMC
traitphen1  10.13235  9.89961 10.34914     8557 <1e-04 ***
traitphen2   0.07419 -0.10057  0.22675     7314  0.364
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In this summary, the notation `phen1:phen1` stands for the variance for the trait 1 and `phen1:phen2` stands for the covariance between the two traits. We see that the additive genetic variance is different for the two traits (1.15 for trait 1 and 0.52 for trait 2) while the residual variances are almost identical. We thus expect different heritabilities. Note the small effective size for `phen2:phen2.animal` and `phen2:phen2.units`. Ideally, we would have to run again the MCMC with more iterations, but we will content ourselves with this result in this tutorial.

5.2.5 Computing heritabilities and genetic correlations

Almost like in the univariate case, the posterior distribution of the heritability is computed as:

```
herit1<-modelmulti$VCV[,'phen1:phen1.animal']/
 (modelmulti$VCV[,'phen1:phen1.animal']+modelmulti$VCV[,'phen1:phen1.units'])
herit2<-modelmulti$VCV[,'phen2:phen2.animal']/
 (modelmulti$VCV[,'phen2:phen2.animal']+modelmulti$VCV[,'phen2:phen2.units'])
mean(herit1)
```

```
[1] 0.5484367
```

```
mean(herit2)
```

```
[1] 0.3509364
```

Like expected, the two heritabilities are different, the trait 2 being less heritable than the trait 1. Let's now compute the genetic correlation:

```
 corr.gen<-modelmulti$VCV[,'phen1:phen2.animal']/
sqrt(modelmulti$VCV[,'phen1:phen1.animal']*modelmulti$VCV[,'phen2:phen2.animal'])
 mean(corr.gen)
```

```
[1] 0.9099979
```

We see that the genetic correlation is strong, around 0.91, which involve some genetics in common for the traits.

6 Some references

Regarding quantitative genetics, and notably heritability estimation, three books are very comprehensive (Falconer & Mackay 1996, Roff 1997, Lynch & Walsh 1998). They however are quite old references, and develop little around the animal model. Fortunately, a consequent literature is currently build around the quantitative genetics of wild populations: various reviews (Kruuk 2004, Postma & Charmantier 2007, Kruuk *et al.* 2008, Visscher *et al.* 2008) and a guide (Wilson *et al.* 2010) are available on the animal model (see also the associated Wild Animal Model Wiki[13]). Two interesting retrospectives on quantitative genetics give a previous insight into its fundamentals (Roff 2007, Hill & Kirkpatrick 2010). Some comparison between the different approaches are also available (Kruuk 2004, Kruuk & Hadfield 2007). Regarding MCMCglmm, the course notes by its developer Jarrod Hadfield form a really nice and reachable

[13]Be careful! The prior definitions in this Wiki are erroneous for the MCMCglmm part: NEVER use the data to be analyzed in order to define a prior distribution!

teaching document about mixed models and Bayesian, while about the handling of the package itself. Gelman *et al.* (2004) is a very comprehensive book about Bayesian inference and Ellison (2004) is a review focused on what Bayesian statistics can bring into Ecology and Evolution. Some methodological papers: about including fixed effects in animal models (Wilson 2008), the importance of taking into account the common environmental effects (Kruuk & Hadfield 2007), the influence of extra-pair copulations (Charmantier & Réale 2005) or the influence of data and pedigree quality for heritability estimation (Quinn *et al.* 2006). A mailing list exists for help about mixed models in R, in which Jarrod Hadfield is very active. To subscribe, see here. Finally, two recent articles about an alternative Bayesian method for animal model estimations (Holand *et al.* 2011) or about molecular approaches (Sillanpää 2011) using genetic data.

References

Charmantier A. & Réale D. (2005). How do misassigned paternities affect the estimation of heritability in the wild? *Molecular Ecology* **14**: 2839–2850. doi:10.1111/j.1365-294X.2005.02619.x. Cited page(s): 34

Day T. & Bonduriansky R. (2011). A unified approach to the evolutionary consequences of genetic and nongenetic inheritance. *The American Naturalist* **178**: E18–E36, URL: http://www.jstor.org/stable/10.1086/660911. Cited page(s): 7

Ellison A.M. (2004). Bayesian inference in ecology. *Ecology Letters* **7**: 509–520. doi:10.1111/j.1461-0248.2004.00603.x. Cited page(s): 34

Falconer D.S. & Mackay T.F. (1996). *Introduction to Quantitative Genetics*, 4th Ed. Benjamin Cummings, Harlow, Essex (UK). Cited page(s): 3, 4, 33

Gelman A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian analysis* **1**: 515–533. doi:10.1214/06-BA117A. Cited page(s): 10, 24

Gelman A., Carlin J.B., Stern H.S. & Rubin D.B. (2004). *Bayesian data analysis*, 2nd Ed. Text in Statisctical Science. Chapman & Hall/CRC Press, Boca Raton, Florida (US). Cited page(s): 34

Hadfield J.D. (2010). MCMCglmm course notes. MCMCglmm R package. URL: http://cran.r-project.org/web/packages/MCMCglmm/index.html. Cited page(s): 1

Hill W.G. & Kirkpatrick M. (2010). What animal breeding has taught us about evolution. *Annual Review of Ecology, Evolution, and Systematics* **41**: 1–19. doi:10.1146/annurev-ecolsys-102209-144728. Cited page(s): 33

Holand A.M., Steinsland I., Martino S. & Jensen H. (2011). Animal models and integrated nested laplace approximations. *Statistics* **Preprints**, URL: http://www.math.ntnu.no/preprint/statistics/2011/S4-2011.pdf. Cited page(s): 34

Kirkpatrick M. & Lande R. (1989). The evolution of maternal characters. *Evolution* **43**: 485–503. doi:10.2307/2409054. Cited page(s): 7

Kruuk L.E., Slate J. & Wilson A.J. (2008). New answers for old questions: The evolutionary quantitative genetics of wild animal populations. *Annual Review of Ecology, Evolution, and Systematics* **39**: 525–548. doi:10.1146/annurev.ecolsys.39.110707.173542. Cited page(s): 33

Kruuk L.E.B. (2004). Estimating genetic parameters in natural populations using the 'animal model'. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* **359**: 873 –890. doi:10.1098/rstb.2003.1437. Cited page(s): 33

Kruuk L.E.B. & Hadfield J.D. (2007). How to separate genetic and environmental causes of similarity between relatives. *Journal of Evolutionary Biology* **20**: 1890–1903. doi:10.1111/j.1420-9101.2007.01377.x. Cited page(s): 33, 34

Lynch M. & Walsh B. (1998). *Genetics and analysis of quantitative traits*. Sinauer Associates, Sunderland, Massachussets (US). Cited page(s): 3, 33

Mousseau T.A. & Roff D.A. (1987). Natural selection and the heritability of fitness components. *Heredity* **59**: 181–197, URL: http://dx.doi.org/10.1038/hdy.1987.113. Cited page(s): 2

Postma E. & Charmantier A. (2007). What 'animal models' can and cannot tell ornithologists about the genetics of wild populations. *Journal of Ornithology* **148**: 633–642. doi:10.1007/s10336-007-0191-8. Cited page(s): 33

Quinn J.L., Charmantier A., Garant D. & Sheldon B.C. (2006). Data depth, data completeness, and their influence on quantitative genetic estimation in two contrasting bird populations. *Journal of Evolutionary Biology* **19**: 994–1002. doi:10.1111/j.1420-9101.2006.01081.x. Cited page(s): 34

Roff D.A. (1997). *Evolutionary quantitative genetics*. Chapman & Hall, New York, New York (US). Cited page(s): 3, 33

Roff D.A. (2007). A centennial celebration for quantitative genetics. *Evolution* **61**: 1017–1032. doi:10.1111/j.1558-5646.2007.00100.x. Cited page(s): 33

Räsänen K. & Kruuk L.E.B. (2007). Maternal effects and evolution at ecological time-scales. *Functional Ecology* **21**: 408–421. doi:10.1111/j.1365-2435.2007.01246.x. Cited page(s): 7

Sillanpää M.J. (2011). On statistical methods for estimating heritability in wild populations. *Molecular Ecology* **20**: 1324–1332. doi:10.1111/j.1365-294X.2011.05021.x. Cited page(s): 34

Visscher P.M., Hill W.G. & Wray N.R. (2008). Heritability in the genomics era — concepts and misconceptions. *Nature Reviews Genetics* **9**: 255–266. doi:10.1038/nrg2322. Cited page(s): 33

Wilson A.J. (2008). Why h2 does not always equal VA/VP? *Journal of Evolutionary Biology* **21**: 647–650. doi:10.1111/j.1420-9101.2008.01500.x. Cited page(s): 7, 34

Wilson A.J., Réale D., Clements M.N., Morrissey M.M., Postma E., Walling C.A., Kruuk L.E.B. & Nussey D.H. (2010). An ecologist's guide to the animal model. *Journal of Animal Ecology* **79**: 13–26. doi:10.1111/j.1365-2656.2009.01639.x. Cited page(s): 7, 33

Wolak M.E. (2012). nadiv: an R package to create relatedness matrices for estimating non-additive genetic variances in animal models. *Methods in Ecology and Evolution* **3**: 792–796, URL: `http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2012.00213.x/full`. Cited page(s):