How to fit an animal model

Julien Martin

12-02-2021

Table des Matières

Preface **5**

1 Introduction **7**
1.1 Data . . . 7
1.2 R . . . 8

2 Univariate animal model **11**
2.1 Scenario and data . . . 11
2.2 Asreml-R . . . 12
2.3 gremlin . . . 18
2.4 MCMCglmm . . . 18
2.5 brms . . . 28

3 Multivariate animal model **31**
3.1 Scenario and data . . . 31
3.2 Asreml-R . . . 32
3.3 gremlin . . . 37
3.4 MCMCglmm . . . 37
3.5 brms . . . 42

4 A repeated measures animal model **47**
4.1 Scenario and data . . . 47
4.2 Asreml-R . . . 48
4.3 gremlin . . . 53
4.4 MCMCglmm . . . 53
4.5 brms . . . 57

5 Quick comparison of codes **59**
5.1 Univariate model with repeated measures . . . 59
5.2 bivariate model . . . 59

Preface

This book is a collection of tutorial from the excellent paper by (Wilson et al., 2010). Instead of just copy pasting the tutorial in a bookdown format, the tutorials have been updated to work with the newest version of the softwares and extended to present other softwares. **However, this is still a work in progress.**

Contributors

List of people who contributed to update and extend tutorials:

- Eric Postma
- Julien Martin

Chapitre 1

Introduction

The book is provides a series of tutorials (and accompanying data files) to fit animal model in `R` using different packages (`ASReml-R`, `gremlin`, `MCMCglmm` and `brms`) . You will need to carefully follow the instructions below to download the data files and install the R packages. Before beginning the tutorial, we assume the reader has successfully installed the chosen R package on their computer and has saved the required data files to an appropriate directory from which they will be read. Full instructions for how to do this are provided with software distributions.

To work though the different tutorial I would recommend to create a folder where you will save your different `R` scripts for the tutorials. In addition, I recommand to create a subfolder `data` to save the files needed.

1.1 Data

1.1.1 Data files

You will need to download 3 data files for the tutorial in `R`:

- gryphon.csv: data on gryphon birth weight
- gryphonRM.csv: data
- gryphonped.csv

In addition, some models presented in the tutorials can take a while to run (sometimes $>$ 1 hour), thus we are also providing model the model outputs to allow you continue the tutorial without waiting for the model to run.

The files are available here

I recommend to save the data and Rdata files in a subfolder `data` in the folder you will use as your working directory for R and where you will save your R scripts. It should be noted that the tutorial are using this structure to read or save data.

1.1.2 Notes on data and pedigree

It is always important to take time to think carefully about the strengths and potential limitations of your pedigree information before embarking on quantitative genetic analyses. Pedigree Viewer,

written by Brian Kinghorn, is an extremely useful application for visualising pedigrees, and can be downloaded from: http://www-personal.une.edu.au/~bkinghor/pedigree.htm. `Pedantics` an R package written by Michael Morrissey and distributed through CRAN (http://cran.r-project.org/) can also be used for this and offers some nice additional features for visualising pedigree structures and generating associated statistics. Before you begin running through the tutorials, we advise taking a moment to look at the pedigree files provided with them using Pedigree Viewer or Pedantics.

1.2 R

describe R Briefly You should check that you have the most current version of R and R packages. You can check the number of the current version on CRAN. If you need to update (or install) R packages, use `install.packages()` and follow the prompted instructions.

1.2.1 R packages

describe briefly the differnet r packages indicate how to load and install provide warning on cost and complexity indicating that we are just touching the surface of what they can do

1.2.1.1 asreml-r

ASReml-R is commercial software published by VSN international (http://www.vsni.co.uk/software/asreml/).

1.2.1.2 gremlin

`gremlin` is a little monster appearing if you wet a mugwai and feed it after midnight. It is also a great and promising software to fit mixed models using a freuqentist approach.

1.2.1.3 MCMCglmm

`MCMCglmm` is an R package for Bayesian mixed model analysis written by Jarrod Hadfield. It is freeware distributed through CRAN (http://cran.r-project.org/). Information about the package, together with a user manual and vignettes are available at http://cran.r-project.org/web/packages/MCMCglmm/index.html . Reference: (Hadfield, 2010, Hadfield (2021)).

This module provides some information that applies to MCMCglmm-based analyses in general, but that will not be included in other tutorials. Most importantly, this applies to some of the simplest waysof determining the performance of a run using MCMCglmm, i.e., verification of the validity of of the posterior distribution. This tutorial is not a substitute for working through the MCMCglmm course notes, which is available from CRAN (the Comprehensive R ArchiveNetwork, http://cran.r-project.org/, or can be accessed in R using the command vignette("CourseNotes","MCMCglmm")). These tutorials do not introduce one of the main advantages of using MCMCglmm for analyses of data from natural populations -the ability to properly model non-normal responses. These capabilities are introduced inthe documentation that is distributed with MCMCglmm, and available from CRAN.

1.2.1.4 brms

`brms` provides an interface to fit Bayesian generalized multivariate (non-)linear multilevel models using `Stan`, which is a C++ package for obtaining full Bayesian inference (see https://mc-stan.org/). The formula syntax is an extended version of the syntax applied in the 'lme4' package to provide a familiar and simple interface for performing regression analyses.

It should be noted that if `brms` is able to fit animal model the parametrization used based on Kroeneker products is not the most efficient and can take quite longer than using a DAG parametrisation directly in `stan`.

Chapitre 2

Univariate animal model

This tutorial will demonstrate how to run a univariate animal model to estimate genetic variance in birth weight in the mighty gryphons.

2.1 Scenario and data

2.1.1 Scenario

In a population of gryphons there is strong positive selection on birth weight with heavier born individuals having, on average higher fitness. To find out whether increased birth weight will evolve in response to the selection, and if so how quickly, we want to estimate the heritability of birth weight.

2.1.2 Data files

Open `gryphonped.csv` and `gryphon.csv` in your text editor. The structure and contents of these files is fairly self-explanatory. The pedigree file `gryphonped.csv` contains three columns containing unique IDs that correspond to each animal, its father, and its mother. Note that this is a multigenerational pedigree, with the earliest generation (for which parentage information is necessarily missing) at the beginning of the file. For later-born individuals maternal identities are all known but paternity information is incomplete (a common situation in real world applications). The phenotype data, as well as additional factors and covariates that we may wish to include in our model are contained in `gryphon.csv`. Columns correspond to individual identity (`animal`), maternal identity (`mother`), year of birth (`byear`), sex (`sex`, where `1` is female and `2` is male), birth weight (`bwt`), and tarsus length (`tarsus`). Each row of the data file contains a record for a different offspring individual. Note that all individuals included in the data file must be included as offspring in the pedigree file.

We can read teh data file, using `read.csv()` which consider by default that `NA` is the symbol for missing values and that the first line of the file contains the column headers.

It is a good idea to make sure that all variables are correctly assigned as numeric or factors:

```
gryphon$animal <- as.factor(gryphon$animal)
gryphon$mother <- as.factor(gryphon$mother)
gryphon$byear <- as.factor(gryphon$byear)
gryphon$sex <- as.factor(gryphon$sex)
gryphon$bwt <- as.numeric(gryphon$bwt)
gryphon$tarsus <- as.numeric(gryphon$tarsus)
```

Similarly we can read in the pedigree file, using `read.csv()` which consider by default that `NA` is the symbol for missing values and that the first line of the file contains the column headers.

```
## 'data.frame':    1309 obs. of  3 variables:
##  $ id    : int  1306 1304 1298 1293 1290 1288 1284 1283 1282 1278 ...
##  $ father: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ mother: int  NA NA NA NA NA NA NA NA NA NA ...
```

```
gryphonped$id <- as.factor(gryphonped$id)
gryphonped$father <- as.factor(gryphonped$father)
gryphonped$mother <- as.factor(gryphonped$mother)
```

Now that we have imported the data and the pedigree file, we are ready to fit an animal model.

2.2 Asreml-R

2.2.1 Running the model

First we need to load the `asreml` library:

```
library(asreml)
```

To be able to fit an animal model Asreml-r needs (the inverse of) the relationship matrix:

```
ainv <- ainverse(gryphonped)
```

We are now ready to specify our first model:

```
model1 <- asreml(
  fixed = bwt ~ 1, random = ~ vm(animal, ainv),
  residual = ~ idv(units),
  data = gryphon,
  na.action = na.method(x = "omit", y = "omit")
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:29 2021
```

```
##           LogLik        Sigma2     DF     wall    cpu
##  1     -4128.454           1.0    853 17:19:29    0.0
##  2     -3284.272           1.0    853 17:19:29    0.0
##  3     -2354.992           1.0    853 17:19:29    0.0
##  4     -1710.357           1.0    853 17:19:29    0.0
##  5     -1363.555           1.0    853 17:19:29    0.0
##  6     -1263.516           1.0    853 17:19:29    0.0
##  7     -1247.854           1.0    853 17:19:29    0.0
##  8     -1247.185           1.0    853 17:19:29    0.0
##  9     -1247.183           1.0    853 17:19:29    0.0
```

In this model, `bwt` is the response variable and the only fixed effect is the mean (the intercept, denoted as `1`). The only random effect we have fitted is `animal`, which will provide an estimate of V_A. Our random `animal` effect is connected to the inverse related matrix `ainv`. `data=` specifies the name of the dataframe that contains our variables. Finally, we tell `asreml()` what to when it encounters `NA`s in either the dependent or predictor variables (in this case we choose to remove the records).

A note of the specification of the structure of the residuals: This simple univariate model will run fine without `residual=~idv(units)`. However, if you are going to use `vpredict()` to calculate the heritability (see below), not specifying the residuals in this way will result in a standard error for the heritability that is incorrect.

To see the estimates for the variance components, we run:

```
summary(model1)$varcomp
```

```
##                   component std.error  z.ratio bound %ch
## vm(animal, ainv)  3.395398 0.6349915 5.347154     P   0
## units!units       3.828602 0.5185919 7.382687     P   0
## units!R           1.000000        NA       NA     F   0
```

We fitted a single random effect so have partitioned the phenotypic variance into two components. The `vm(animal, ainv)` variance component is V_A and is estimated as 3.4. Given that the ratio of V_A to its standard error (`z.ratio`) is considerably larger than 2 (*i.e.* the parameter estimate is more than 2 SEs from zero) this looks likely to be highly significant. The `units!units` component refers to the residual variance V_R, and `units$R` should be ignored. If you don't include `residual=~idv(units)`in your model specification, `units$R` will provide you with the residual variance.

2.2.2 Estimating heritability

We can calculate the h^2 of birth weight from the components above since $h^2 = V_A/V_P = V_A/(V_A + V_R)$. Thus according to this model, h^2 = 3.4 / (3.4 + 3.83) = 0.47.

Alternatively we can use the `vpredict()` function to calculate h^2 and its standard error:

```
vpredict(model1, h2.bwt ~ V1 / (V1 + V2))
```

```
##         Estimate         SE
## h2.bwt 0.4700163 0.07650881
```

2.2.3 Adding fixed effects

To add fixed effects to a univariate model simply modify the model statement. For example we might know (or suspect) that birth weight is a sexually dimorphic trait and therefore fit a model

```
model2 <- asreml(
  fixed = bwt ~ 1 + sex,
  random = ~ vm(animal, ainv),
  residual = ~ idv(units),
  data = gryphon,
  na.action = na.method(x = "omit", y = "omit")
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:29 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -3364.126           1.0    852 17:19:29    0.0
##  2     -2702.117           1.0    852 17:19:29    0.0
##  3     -1978.916           1.0    852 17:19:29    0.0
##  4     -1487.834           1.0    852 17:19:29    0.0
##  5     -1236.350           1.0    852 17:19:29    0.0
##  6     -1172.771           1.0    852 17:19:29    0.0
##  7     -1165.270           1.0    852 17:19:29    0.0
##  8     -1165.093           1.0    852 17:19:29    0.0
##  9     -1165.093           1.0    852 17:19:29    0.0
```

Now we can look at the fixed effects parameters and assess their significance with a conditional Wald F-test:

```
summary(model2, coef = TRUE)$coef.fixed
wald.asreml(model2, ssType = "conditional", denDF = "numeric")
```

```
##             solution std error  z.ratio
## sex_1       0.000000        NA       NA
## sex_2       2.206996 0.1619974 13.62365
## (Intercept) 6.058669 0.1718244 35.26082
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:29 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -1165.093           1.0    852 17:19:30    0.0
```

```
##  2     -1165.093           1.0    852 17:19:30    0.0
## Calculating denominator DF
```

```
##
##             Df denDF  F.inc  F.con Margin          Pr
## (Intercept)  1   251 3491.0 3491.0        0.00000e+00
## sex          1   831  185.6  185.6      A 2.70204e-38
```

The very small probability (`Pr`) in the Wald test above shows that `sex` is a highly significant fixed effect, and from the parameter estimates we can see that the average male (sex 2) is 2.2 kg ($\pm$ 0.16 SE) heavier than the average female (sex 1). However, when we look at the variance components in the model including `sex` as a fixed effect, we see that they have changed slightly from the previous model:

```
summary(model2)$varcomp
```

```
##                   component std.error  z.ratio bound %ch
## vm(animal, ainv)  3.060441 0.5243571 5.836558     P   0
## units!units       2.938412 0.4161473 7.060991     P   0
## units!R           1.000000        NA       NA     F   0
```

In fact since `sex` effects were previously contributing to the residual variance of the model, our estimate of V_R (denoted `units!R` in the output) is now slightly lower than before. This has an important consequence for estimating heritability since if we calculate V_P as V_A+V_R then as we include fixed effects we will soak up more residual variance driving V_P. Assuming that V_A is more or less unaffected by the fixed effects fitted then as V_P goes down we expect our estimate of h^2 will go up:

```
vpredict(model2, h2.bwt ~ V1 / (V1 + V2))
```

```
##        Estimate         SE
## h2.bwt 0.510171 0.07432388
```

Here h^2 has increased slightly from 0.47 to 0.51. Which is the better estimate? It depends on what your question is. The first is an estimate of the proportion of variance in birth weight explained by additive effects, the latter is an estimate of the proportion of variance in birth weight *after conditioning on sex* that is explained by additive effects.

2.2.4 Adding random effects

This is done by simply modifying the model statement in the same way. For instance fitting

```
model3 <- asreml(
  fixed = bwt ~ 1 + sex,
  random = ~ vm(animal, ainv) + byear,
  residual = ~ idv(units),
  data = gryphon,
```

```
  na.action = na.method(x = "omit", y = "omit")
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:30 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -2742.658           1.0    852 17:19:30    0.0
##  2     -2237.268           1.0    852 17:19:30    0.0
##  3     -1690.453           1.0    852 17:19:30    0.0
##  4     -1328.910           1.0    852 17:19:30    0.0
##  5     -1154.597           1.0    852 17:19:30    0.0
##  6     -1116.992           1.0    852 17:19:30    0.0
##  7     -1113.809           1.0    852 17:19:30    0.0
##  8     -1113.772           1.0    852 17:19:30    0.0
##  9     -1113.772           1.0    852 17:19:30    0.0
```

results in an additional variance component of birth year:

```
summary(model3)$varcomp
```

```
##                  component std.error  z.ratio bound %ch
## byear            0.8862604 0.2695918 3.287416     P   0
## vm(animal, ainv) 2.7068665 0.4422140 6.121169     P   0
## units!units      2.3092415 0.3451025 6.691466     P   0
## units!R          1.0000000        NA       NA     F   0
```

Here the variance in `bwt` explained by `byear` is 0.886 and, based on the `z.ratio`, appears to be significant. Thus we would conclude that year-to-year variation (*e.g.*, in weather, resource abundance) contributes to V_P. Note that although V_A has changed somewhat, as most of what is now partitioned as a birth year effect was previously partitioned as V_R. Thus what we have really done here is to partition environmental effects into those arising from year-to-year differences versus everything else, and we do not really expect much change in h^2 (since now $h^2 = V_A/(V_A + V_{BY} + V_R)$).

However, we get a somewhat different result if we also add a random effect of `mother` to test for maternal effects:

```
model4 <- asreml(
  fixed = bwt ~ 1 + sex,
  random = ~ vm(animal, ainv) + byear + mother,
  residual = ~ idv(units),
  data = gryphon,
  na.action = na.method(x = "omit", y = "omit")
)
```

```
## Model fitted using the sigma parameterization.
```

```
## ASReml 4.1.0 Fri Feb 12 17:19:30 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -2033.178           1.0    852 17:19:30    0.0
##  2     -1723.734           1.0    852 17:19:30    0.0
##  3     -1396.354           1.0    852 17:19:30    0.0
##  4     -1193.012           1.0    852 17:19:30    0.0
##  5     -1107.946           1.0    852 17:19:30    0.0
##  6     -1095.327           1.0    852 17:19:30    0.0
##  7     -1094.816           1.0    852 17:19:30    0.0
##  8     -1094.815           1.0    852 17:19:30    0.0
```

Gives estimated variance components of

```
summary(model4)$varcomp
```

```
##                  component std.error  z.ratio bound %ch
## byear            0.8820313 0.2632455 3.350604     P   0
## mother           1.1184698 0.2386239 4.687167     P   0
## vm(animal, ainv) 2.2985320 0.4962496 4.631806     P   0
## units!units      1.6290034 0.3714154 4.385934     P   0
## units!R          1.0000000        NA       NA     F   0
```

Here partitioning of significant maternal variance has resulted in a further decrease in V_R but also a decrease in V_A. The latter is because maternal effects of the sort we simulated (fixed differences between mothers) will have the consequence of increasing similarity among maternal siblings. Consequently they can look very much like additive genetic effects and if present, but unmodelled, represent a type of "common environment effect"" that can - and will - cause upward bias in V_A and so h^2.

2.2.5 Testing significance of random effects

A final point to note in this tutorial is that while the `z.ratio` (`component`/`std.error`) reported is a good indicator of likely statistical significance (>1.96?), the standard errors are approximate and are not recommended for formal hypothesis testing. A better approach is to use likelihood-ratio tests.

For example, to test the significance of maternal effects we could compare models with and without the inclusion of maternal identity as a random effect and compare the final log-likelihoods of these models.

```
model4$loglik
```

```
## [1] -1094.815
```

shows that the model including maternal identity has a log-likelihood of -1094.815, and

```
model3$loglik
```

```
## [1] -1113.772
```

shows that the model excluding maternal identity has a log-likelihood of -1113.772.

A test statistic equal to twice the absolute difference in these log-likelihoods is assumed to be distributed as Chi square with one degree of freedom. In this case we would conclude that the maternal effects are highly significant since: $2 \times$ (-1094.8145793 - -1113.7719147) equals 37.9146708, and the p-value that comes with this is:

```
1 - pchisq(2 * (model4$loglik - model3$loglik), 1)
```

```
## [1] 7.390738e-10
```

As $P < 0.0001$ we would therefore conclude that the additional of maternal identity as a random effect significantly improves the model, given an increase in log-likelihood of approximately 19.

2.3 gremlin

TODO (maybe just bother Matthew to do it)

2.4 MCMCglmm

2.4.1 Running the model

First load MCMCglmm:

```
library(MCMCglmm)
```

The first model we will fit is a simple animal model with no fixed effects, and only an 'animal' random effect relating individuals to their additive genetic values through the pedigree. First we are going to define priors. In a way we might want to avoid using priors, because we would like all of the infromation in our analysis to come from our data. By default MCMCglmm uses improper priors, but this can cause inferential and numerical problems. We will specify priors for the animal effect and the residual variance using the following code:

```
prior1.1 <- list(
  G = list(G1 = list(V = 1, nu = 0.002)),
  R = list(V = 1, nu = 0.002)
)
```

This prior specification used to be used a lot because it was believed to be relatively uninformative, and is equivalent to an inverse-gamma prior with shape and scale equal to 0.001. In many cases it is relatively uninformative but when the posterior distribution for the variances has suport close

to zero it can behave poorly. Parameter expanded priors (See Chapter 8 of the CourseNotes) are gaining in popularity due to their better behaviour but for the purposes of this tutorial we will stick with the inverse-gamma prior. We have told MCMCglmm to pay little heed to our prior expectaion (V) by specifying a small degree of belief parameter (nu) of 0.002. Since this is a univariate analysis, the priors are matricies of order 1 and thus nu>0 is the smallest degree of belief that provides what is known as a 'proper' prior, avoiding numerical problems. In fact, there is a lot of information in the data regarding the marginal distributions of the parameters, and MCMCglmm will run most of the models that we suggest in these tutorials without priors. However, this is poor practice, and we will therefore use priors throughout these tutorials. We can now fit an animal model. The model to decompose variation in birth weight into genetic and residual effects is as follows:

The lower case "animal" is a can be a **special** word for MCMCglmm. If a `pedigree` argument is provided then `MCMCglmm` will recognize the term `animal` as the term to use to estimate additive genetic variance. When the argument `pedigree` is not provided then the word `animal` is not different than any other variable. However, instead of providing a pedigree argument to the call to MCMCglmm function it is much more fexible to use the `ginv` argument to specify the random effect that must be linked to the pedigree (with the inverse relatedness matrix). We thus first estimate teh inverse relatedness matrix using `inverseA()` then fit the animal model.

```
Ainv <- inverseA(gryphonped)$Ainv
model1.1 <- MCMCglmm(bwt ~ 1, random = ~animal, ginv = list(animal = Ainv), data = gryphon
```

```
##
##                          MCMC iteration = 0
##
##                          MCMC iteration = 1000
##
##                          MCMC iteration = 2000
##
##                          MCMC iteration = 3000
##
##                          MCMC iteration = 4000
##
##                          MCMC iteration = 5000
##
##                          MCMC iteration = 6000
##
##                          MCMC iteration = 7000
##
##                          MCMC iteration = 8000
##
##                          MCMC iteration = 9000
##
##                          MCMC iteration = 10000
##
##                          MCMC iteration = 11000
##
```

```
##                          MCMC iteration = 12000
##
##                          MCMC iteration = 13000
```

After typing this code, MCMCglmm will run, taking about 20 seconds on a modern desk- top computer. The progress of the run will be printed to the screen. Also, note the warning message will be printed at the end of the run. This is natural too. In order for the MCMC algorithm to work, MCMCglmm must keep track of effects associated with unmeasured individuals appearing in the pedigree. This will not affect the answers, but when many unmeasured individuals exist, it can hinder the ability of the algorithm to explore the parameter space (more on this, and a solution, later). Lets have a look at the MCMCglmm outputs. First we will evaluate how confident we can be that MCMCglmm found good answers. By entering

```
plot(model1.1$Sol)
```

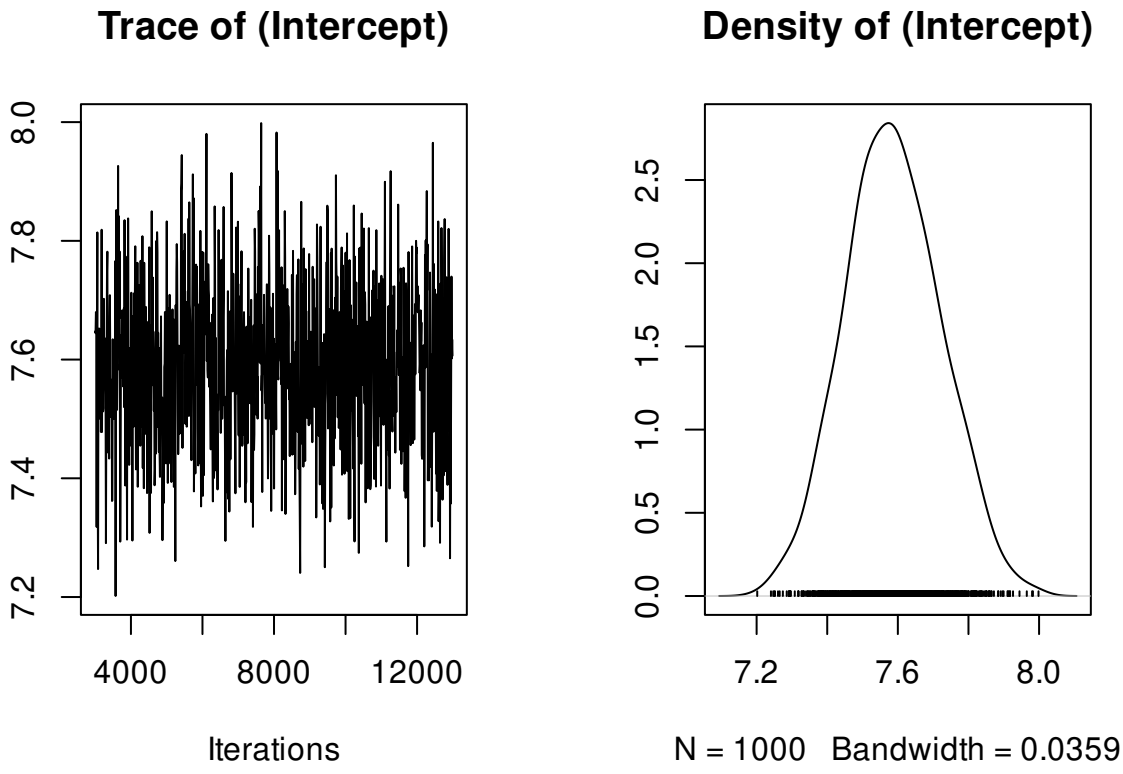


Figure 2.1: The posterior distribution of the fixed effect (the intercept, or mean) in model 1.1

in the console, we get Figure 1 (p. 5). The plot on the left shows a time series of the values of 1000 samples of the posterior distribution of the the model intercept (mean birthweight). The plot on the right shows the same data as a distribution. Complicated statistical methods for estimating population means are of course of little interest; rather, we are examining these outputs to check that MCMCglmm's algorithms worked well for our data and for this model. The important point here is that a consistent amount of variation around a largely unchanging mean value of the intercept was obtained, and the posterior distribution of the intercept appears to be valid. More rigorous means of evaluation the independence of the samples in the posterior distribution

(evaluating autocorrelation) are discussed in the MCMCglmm CourseNotes, available from CRAN. Note that your output for model 1.1 may not be identical to this due to Monte Carlo (random number) error.

The posterior distributions of the the variance components are generally of more inter- est to animal model users. We can view plots of the posterior distribution for the variance components for model 1.1 by

```
plot(model1.1$VCV)
```

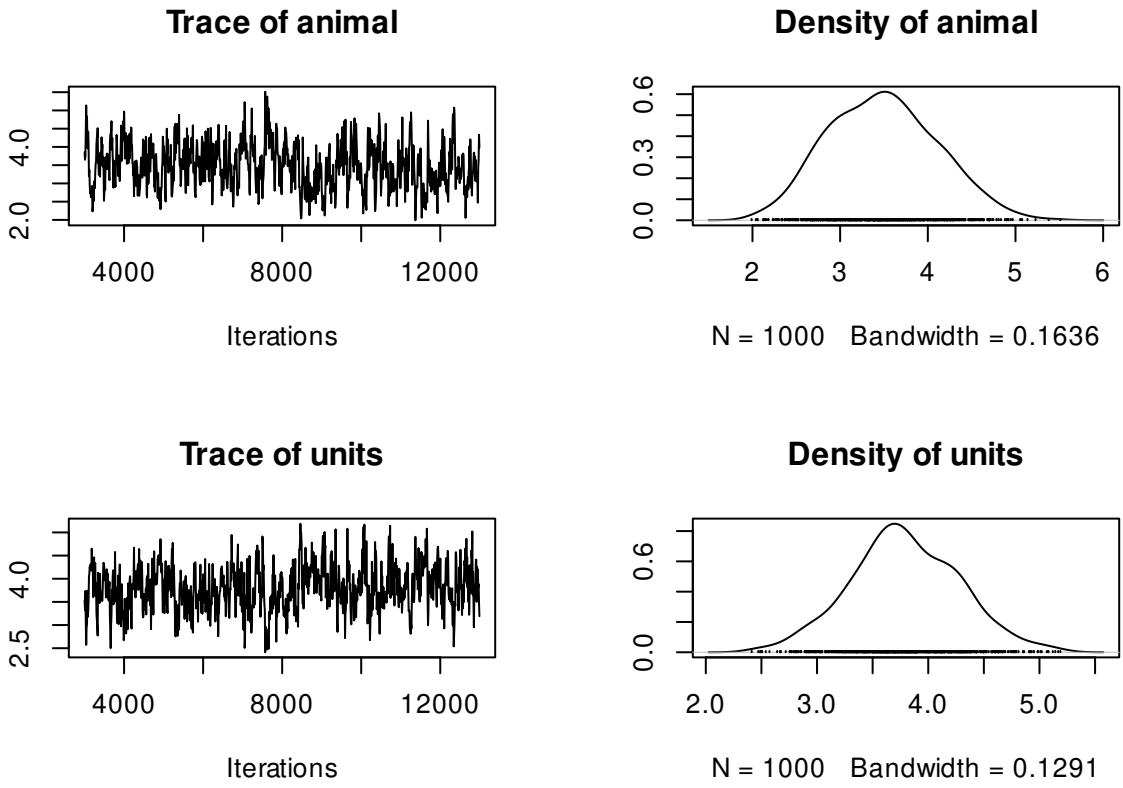


Figure 2.2: The posterior distributions of the variance components of model 1.1, based on an analysis with the default values for nitt, burnin, and thin in MCMCglmm

which generates Figure 2 (p. 6). Here we see distributions of the estimates of the additive genetic (animal) and residual (units) effects. These samples contain some au- tocorrelation, i.e., trends are apparent in the left-hand plot. We can deal with this easily.

We will simply re-run the model for a longer number of iterations, and sample the chain less frequently. So far we have been running MCMCglmm with its default values. These defaults are a total run length of 13000 iterations, the first 3000 of which are discarded as a 'burn-in' period to make sure that the converges to the part of the parameter space where the maximum likelihood exists. The remaining 10000 iterations are sampled (es- timates retained) every 10 iterations (the thinning interval). Because the values in the left-hand plots in figure 2 to appear to have different values at the beginning of the run, we might suspect that a longer burn-in period might be required. We can reduce the autocorrelation by lengthening the rest of the run and sampling the chain less

frequently. The following code runs the same model 1.1, but is likely to produce better samples of the posterior distributions. This model should take about two minutes to analyze.

```
model1.1 <- MCMCglmm(bwt ~ 1, random = ~animal, ginv = list(animal = Ainv), data = gryphon
```

Notice that we have now included the command verbose=FALSE in the MCMCglmm call. We will continue this throughout the tutorial so that more complete screen outputs can be included in this document without using too much space. Now produce the plots of the samples of the fixed and random effects (they have not been included in this document). Note that the autocorrelation is much reduced. A more compact way to evaluate the validity of the posterior distributions is to calculate autocorrelation among samples, as follows:

```
autocorr.diag(model1.1$VCV)
```

```
##                animal        units
## Lag 0     1.000000000 1.0000000000
## Lag 50    0.177204004 0.1319444721
## Lag 250  -0.008856328 0.0099887594
## Lag 500  -0.008189573 0.0059353491
## Lag 2500 -0.020351937 0.0003427092
```

We will consider these levels of autocorrelation acceptable, at least for the purposes of this tutorial. Ideally, all samples of the posterior distribution should be independent, and the autocorrelation for all lag values greater than zero should be near zero. However, in practice this will not strictly be achievable for all analytical scenarios. Certainly the levels of autocorrelation observed here should not be tollerated in any formal analysis. Note that the validity of posterior distributions of any analysis should always be checked; however, for brevity we will not continue to be so consistently diligent throughout the rest of these tutorials. We can now proceed with confidence to recover some more information from these samples. We can obtain estimates of the additive genetic and residual variance by calculating the modes of the posterior distributions:

```
posterior.mode(model1.1$VCV)
```

```
##   animal    units
## 3.199249 3.981221
```

We can obtain the Bayesian equivalent of confidence intervals by calculating the the values of the estimates that bound 95% (or any other proportion) of the posterior distri- butions:

```
HPDinterval(model1.1$VCV)
```

```
##           lower    upper
## animal 2.413592 4.733803
## units  2.784487 4.831548
## attr(,"Probability")
## [1] 0.95
```

We specified weak priors in this analyses. Now we will check whether or not proper priors would have influenced the results that we obtained. The simplest way to do this is to rerun the model with different priors. Here we construct priors with a larger degree of belief parameter, and we will specify that a large proportion of the variation is under genetic control:

```
p.var <- var(gryphon$bwt, na.rm = TRUE)
prior1.1.2 <- list(G = list(G1 = list(
  V = matrix(p.var * 0.05),
  nu = 1
)), R = list(V = matrix(p.var * 0.95), nu = 1))
model1.1.2 <- MCMCglmm(bwt ~ 1,
  random = ~animal, ginv = list(animal = Ainv),
  data = gryphon, prior = prior1.1.2, nitt = 65000, thin = 50,
  burnin = 15000, verbose = FALSE
)
posterior.mode(model1.1$VCV)
```

```
##   animal    units
## 3.199249 3.981221
```

```
posterior.mode(model1.1.2$VCV)
```

```
##   animal    units
## 3.105066 4.032622
```

and we can therefore conclude that the difference in the priors has little effect on the outcome of the analysis. This is typical for an analysis where lots of data are available relative to the complexity of the model, but is often not the case. In all cases, it is important to check the effect of priors on conclusions drawn from a model.

2.4.2 Estimating heritability

A useful property of Bayesian posterior distributions is that we can apply almost any transformation to these distributions and they will remain valid. This applies to the calculation of heritabilities. We can obtain an estimate of the heritability by applying the basic formula h 2 =V A /V P to each sample of the posterior disribution:

```
posterior.heritability1.1 <- model1.1$VCV[, "animal"] / (model1.1$VCV[, "animal"] + model1
HPDinterval(posterior.heritability1.1, 0.95)
```

```
##          lower     upper
## var1 0.3332045 0.6110076
## attr(,"Probability")
## [1] 0.95
```

```
posterior.mode(posterior.heritability1.1)
```

```
##      var1
## 0.4996456
```

Generate a plot of the posterior distribution of this heritability estimate:

```
plot(posterior.heritability1.1)
```

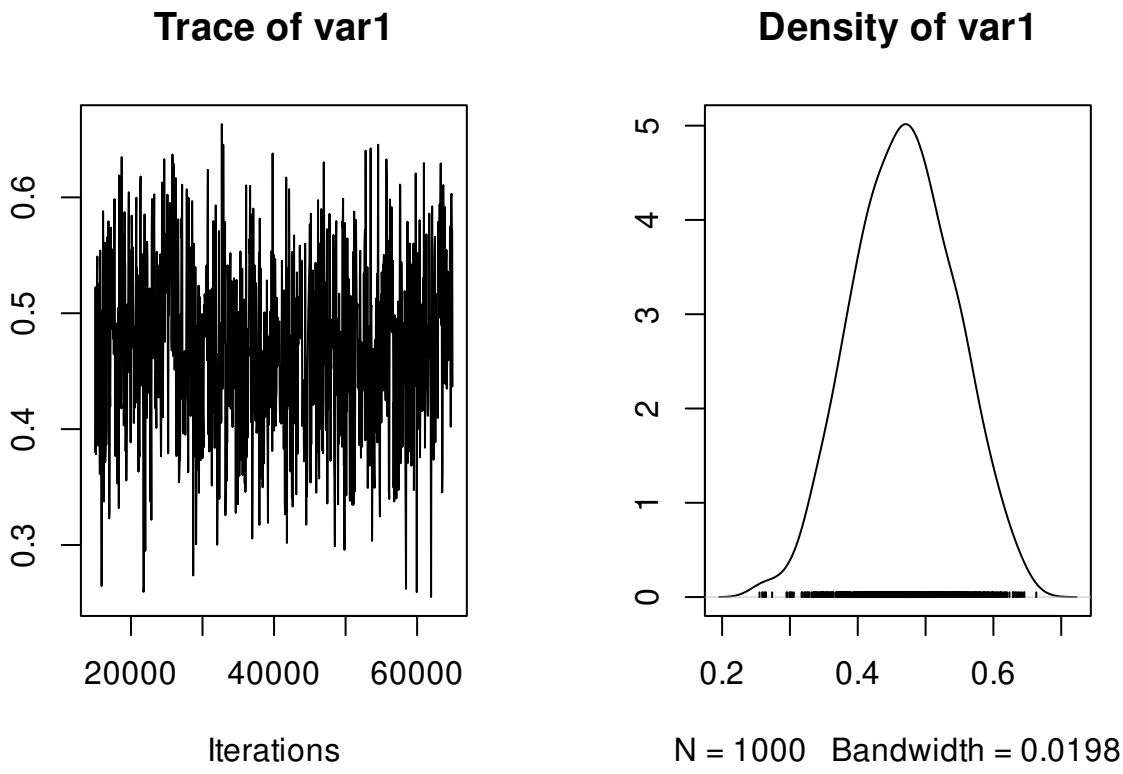


Figure 2.3: The posterior distributions the heritability from model 1.1

2.4.3 Adding fixed effects

To add effects to a univariate model we simply modify the fixed effect portion of the the model specification:

```
model1.2 <- MCMCglmm(bwt ~ sex, random = ~animal, ginv = list(animal = Ainv), data = gryph
```

We can assess the significance of sex as a fixed effect by examining its posterior distribution.

```
posterior.mode(model1.2$Sol[, "sex2"])
```

```
##     var1
## 2.082665
```

```
HPDinterval(model1.2$Sol[, "sex2"], 0.95)
```

```
##         lower    upper
## var1 1.920319 2.547148
## attr(,"Probability")
## [1] 0.95
```

The posterior distribution of the sex2 term does not overlap zero. Thus the we can infer that sex has a statistical effect on birthweight in this model and is a useful addition to the model, for most purposes. MCMCglmm has designated sex2 as the contrast between the two factor levels (male and female). It is also worth noting that the variance components have changed slightly:

```
posterior.mode(model1.2$VCV)
```

```
##   animal    units
## 3.013331 3.000154
```

In fact since sex effects were previously contributing to the residual variance of the model our estimate of V R (denoted 'units' in the output) is now slightly lower than before. This has an important consequence for estimating heritability since if we calculate V P as V A +V R then as we include fixed effects we will soak up more residual variance driving V P . Assuming that V A is more or less unaffected by the fixed effects fitted then as V P goes down we expect our estimate of h 2 will go up.

```
posterior.heritability1.2 <- model1.2$VCV[, "animal"] / (model1.2$VCV[, "animal"] + model1
posterior.mode(posterior.heritability1.2)
```

```
##      var1
## 0.4973786
```

```
HPDinterval(posterior.heritability1.2, 0.95)
```

```
##          lower    upper
## var1 0.3790183 0.655496
## attr(,"Probability")
## [1] 0.95
```

Here h 2 has increased slightly from 0.4829 to 0.5079 (again, your values may differ slightly due to Monte Carlo error). Which is the better estimate? It depends on what your question is. The first is an estimate of the proportion of variance in birth weight explained by additive effects, the latter is an estimate of the proportion of variance in birth weight after conditioning on sex that is explained by additive effects.

2.4.4 Adding random effects

This is done by simply modifying the model statement in the same way, but requires addition of a prior for the new random effect. For instance, we can fit an effect of birth year:

```
prior1.3 <- list(G = list(G1 = list(V = 1, nu = 0.002), G2 = list(V = 1, nu = 0.002)), R =
model1.3 <- MCMCglmm(bwt ~ sex, random = ~ animal + byear, ginv = list(animal = Ainv), dat
posterior.mode(model1.3$VCV)
```

```
##    animal     byear     units
## 2.5855155 0.8071505 2.3493597
```

Here the variance in birth weight explained by birth year is 0.7887. Note that although V A has changed somewhat, most of what is now partitioned as a birth year effect was previously partitioned as V R . Thus what we have really done here is to partition environ- mental effects into those arising from year to year differences versus everything else, and we do not really expect much change in h 2 (since now h 2 = V A /(V A + V BY + V R)). However, we get a somewhat different result if we also add a random effect of mother to test for maternal effects:

```
p.var <- var(gryphon$bwt, na.rm = TRUE)
prior1.4 <- list(G = list(G1 = list(V = 1, nu = 0.002), G2 = list(
  V = 1,
  nu = 0.002
), G3 = list(V = 1, nu = 0.002)), R = list(
  V = 1,
  nu = 0.002
))
model1.4 <- MCMCglmm(bwt ~ sex,
  random = ~ animal + byear + mother,
  ginv = list(animal = Ainv), data = gryphon, nitt = 65000, thin = 50, burnin = 15000,
  prior = prior1.4, verbose = FALSE
)
posterior.mode(model1.4$VCV)
```

```
##   animal    byear   mother    units
## 2.401821 0.793273 1.145345 1.605242
```

Here partitioning of significant maternal variance has resulted in a further decrease in V R but also a decrease in V A . The latter is because maternal effects of the sort we simulated (fixed differences between mothers) will have the consequence of increasing similarity among maternal siblings. Consequently they can look very much like additive genetic effects and if present, but unmodelled, represent a type of 'common environment effect' that can - and will- cause upward bias in V A and so h 2 . Let's compare the estimates of heritability from each of models 1.2, 1.3 and 1.4:

```
posterior.heritability1.3 <- model1.3$VCV[, "animal"] / (model1.3$VCV[, "animal"] + model1
posterior.heritability1.4 <- model1.4$VCV[, "animal"] / (model1.4$VCV[, "animal"] + model1
posterior.mode(posterior.heritability1.2)
```

```
##      var1
## 0.4973786
```

```
posterior.mode(posterior.heritability1.3)
```

```
##      var1
## 0.4477868
```

```
posterior.mode(posterior.heritability1.4)
```

```
##      var1
## 0.4027288
```

2.4.5 Testing significance of variance components

While testing the significance of fixed effects by evaluating whether or not their posterior distributions overlap zero was simple and valid, this approach does not work for vari- ance components. Variance components are bound to be positive (given a proper prior), and thus even when a random effect is not meaningful, its posterior distribution will never overlap zero. Model comparisons can be performed using the deviance information criterion (DIC), although it should be noted that the properties of DIC are not well un- derstood and that the DIC may be focused at the wrong level for most people's intended level of infernce - particularly with non-Gaussian responses. The implementation of DIC in MCMCglmm is further described in the reference manual. DIC values are calculated by MCMCglmm by default. Briefly, DIC like other information criteria balance model fit and model complexity simultaneously, and small values of DIC are prefered. We can compare models 1.4 and 1.3, i.e., models with and without the mother term:

```
model1.3$DIC
```

```
## [1] 3550.392
```

```
model1.4$DIC
```

```
## [1] 3314.328
```

model 1.4 has a much lower DIC value. Since the maternal effect term is the only difference between the models, we can consider the inclusion of this term statistically justifiable. We should note however that DIC has a large sampling variance and should probably only be calculated based on much longer MCMC runs.

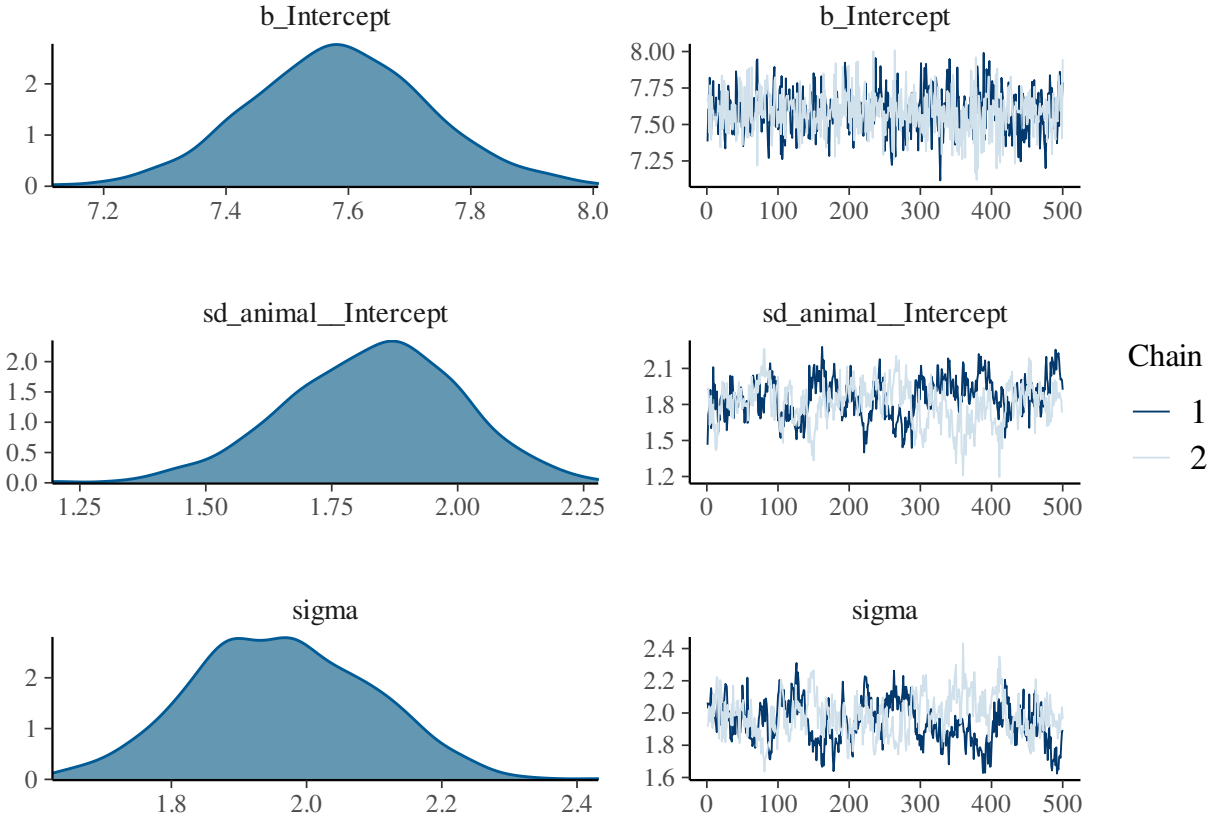
2.5 brms

```
library(brms)
Amat <- as.matrix(nadiv::makeA(gryphonped))
brms_m1.1 <- brm(
  bwt ~ 1 + (1 | gr(animal, cov = Amat)),
  data = gryphon,
  data2 = list(Amat = Amat),
  family = gaussian(),
  chains = 2, cores = 2, iter = 1000
)
save(brms_m1.1, file = "data/brms_m1_1.rda")
```

```
load("data/brms_m1_1.rda")
summary(brms_m1.1)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: bwt ~ 1 + (1 | gr(animal, cov = Amat))
##    Data: gryphon (Number of observations: 854)
## Samples: 2 chains, each with iter = 1000; warmup = 500; thin = 1;
##          total post-warmup samples = 1000
##
## Group-Level Effects:
## ~animal (Number of levels: 854)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.83      0.17     1.47     2.15 1.04       60      224
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     7.58      0.15     7.29     7.88 1.00      639      558
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     1.96      0.13     1.71     2.22 1.04       57      165
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(brms_m1.1)
```



Chapitre 3

Multivariate animal model

This tutorial will demonstrate how to run a multivariate animal model looking at birth weight and tarsus length of the phenomenal gryphons.

3.1 Scenario and data

3.1.1 Scenario

Since natural selection rarely acts on single traits, to understand how birth weight might evolve in our population of gryphons, we may also want to think about possible covariance with other traits. If tarsus length at fledging is also under positive selection, what implications does this have for birth weight and vice versa? If the two traits are positively genetically correlated then this will facilitate evolution of larger size (since response of one trait will induce a positively correlated response in the other). If there is negative genetic covariance then this could act as an evolutionary constraint.

Using multivariate models allows the estimation of parameters relating to each trait alone (*i.e.* V_A, h^2, etc), but also yields estimates of covariance components between traits. These include the (additive) genetic covariance COV_A which is often rescaled to give the additive genetic correlation r_A. However, covariance can also arise through other random effects (*e.g.* maternal covariance) and these sources can also be explicitly modelled in a bivariate analysis.

3.1.2 gryphon files

gryphonpedigree and phenotypic data files are the same as those used in tutorial 1 (*i.e*, `gryphonped.csv` and `gryphon.csv` respectively).

Reading the data

```
gryphon <- read.csv("data/gryphon.csv")
gryphon$animal <- as.factor(gryphon$animal)
gryphon$mother <- as.factor(gryphon$mother)
gryphon$byear <- as.factor(gryphon$byear)
```

```
gryphon$sex <- as.factor(gryphon$sex)
gryphon$bwt <- as.numeric(gryphon$bwt)
gryphon$tarsus <- as.numeric(gryphon$tarsus)
```

Reading the pedigree

```
gryphonped <- read.csv("data/gryphonped.csv")
gryphonped$id <- as.factor(gryphonped$id)
gryphonped$father <- as.factor(gryphonped$father)
gryphonped$mother <- as.factor(gryphonped$mother)
```

3.2 Asreml-R

3.2.1 Running the model

For running multivariate analyses in ASReml-R, the code is slightly more complex than for the univariate case. This is because ASReml-R allows us to make different assumptions about the way in which traits might be related and so we need to explicitly code a model of the (co)variance structure we want to fit. We have also specified some starting values. These are can be very approximate *guestimates*, but having reasonable starting values can aid convergence. Finally, we have increased the default maximum number of iterations (`maxiter`) which can help to achieve convergence for more complicated models.

```
ainv <- ainverse(gryphonped)
modela <- asreml(
  fixed = cbind(bwt, tarsus) ~ trait,
  random = ~ us(trait):vm(animal, ainv), init = c(1, 0.1, 1),
  residual = ~ id(units):us(trait, init = c(1, 0.1, 1)),
  data = gryphon,
  maxit = 20
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:47 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -5118.122           1.0   1535 17:19:47    0.0
##  2     -4358.769           1.0   1535 17:19:47    0.0
##  3     -3540.792           1.0   1535 17:19:47    0.0
##  4     -3004.970           1.0   1535 17:19:47    0.0
##  5     -2747.831           1.0   1535 17:19:47    0.0
##  6     -2687.807           1.0   1535 17:19:47    0.0
##  7     -2680.057           1.0   1535 17:19:47    0.0
##  8     -2679.743           1.0   1535 17:19:47    0.0
##  9     -2679.741           1.0   1535 17:19:47    0.0
```

This has fitted a bivariate model of `bwt` and `tarsus`, with the mean for each of the traits as a fixed effect (`trait`). The additive genetic variance-covariance matrix (**G**) is unstructured (`us`; *i.e.* all elements are free to vary) and the starting values for V_A for `bwt`, COV_A between `bwt` and `tarsus`, and V_A for `tarsus` are set to 1, 0.1 and 1, respectively. Similarly, the residual matrix is unstructured and uses the same starting values.

Let's have a look at the variance components, and notice that there are now six (co)variance components reported in the table:

```
summary(modela)$varcomp
```

```
##                                          component std.error  z.ratio bound
## trait:vm(animal, ainv)!trait_bwt:bwt      3.368498 0.6349441 5.305188     P
## trait:vm(animal, ainv)!trait_tarsus:bwt   2.460195 1.0735865 2.291567     P
## trait:vm(animal, ainv)!trait_tarsus:tarsus 12.346837 3.0753775 4.014739     P
## units:trait!R                             1.000000        NA       NA     F
## units:trait!trait_bwt:bwt                 3.849836 0.5199897 7.403677     P
## units:trait!trait_tarsus:bwt              3.312978 0.9128834 3.629136     P
## units:trait!trait_tarsus:tarsus          17.645584 2.6668492 6.616641     P
##                                          %ch
## trait:vm(animal, ainv)!trait_bwt:bwt      0.0
## trait:vm(animal, ainv)!trait_tarsus:bwt   0.2
## trait:vm(animal, ainv)!trait_tarsus:tarsus 0.1
## units:trait!R                             0.0
## units:trait!trait_bwt:bwt                 0.0
## units:trait!trait_tarsus:bwt              0.1
## units:trait!trait_tarsus:tarsus           0.1
```

The first three terms relate to the genetic matrix and, in order are $V_{A,bwt}$, COV_A, $V_{A,tarsus}$. Below is again a line where the `units:traitr!R` component equals 1, which again can be ignored. The final three terms relate to the residual matrix and correspond to $V_{R,bwt}$, COV_R, $V_{R,tarsus}$. Based on our quick and dirty check (is `z.ratio` > 1.96?) all components look to be statistically significant.

We can calculate the genetic correlation as $COV_A / \sqrt{V_{A,bwt} \cdot V_{A,tarsus}}$. Thus this model gives an estimate of $r_A = 0.38$. Although we can calculate this by hand, we can also use `vpredict()`, which also provides an (approximate) standard error:

```
vpredict(modela, rA ~ V2 / sqrt(V1 * V3))
```

```
##     Estimate       SE
## rA 0.3814816 0.129991
```

Of course we can also calculate the heritability of `bwt` and `tarsus` from this model:

```
vpredict(modela, h2.bwt ~ V1 / (V1 + V5))
```

```
##         Estimate         SE
```

```
## h2.bwt 0.4666586 0.07672219
```

```
vpredict(modela, h2.tarsus ~ V3 / (V3 + V7))
```

```
##            Estimate        SE
## h2.tarsus 0.4116652 0.0930738
```

3.2.2 Adding fixed and random effects

Fixed and random effects can be added just as for the univariate case. Given that our full model of bwt from tutorial 1 had sex as a fixed effect as well as random effects of birth year and mother we could specify a bivariate formulation of this using:

```
modelb <- asreml(
  fixed = cbind(bwt, tarsus) ~ trait + trait:sex,
  random = ~ us(trait, init = c(1, 0.1, 1)):vm(animal, ainv) +
    us(trait, init = c(1, 0.1, 1)):byear +
    us(trait, init = c(1, 0.1, 1)):mother,
  residual = ~ id(units):us(trait, init = c(1, 0.1, 1)),
  data = gryphon,
  maxit = 20
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:47 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -4672.301           1.0   1533 17:19:47    0.0
##  2     -4005.615           1.0   1533 17:19:47    0.0
##  3     -3271.483           1.0   1533 17:19:47    0.0 (1 restrained)
##  4     -2761.414           1.0   1533 17:19:47    0.0 (1 restrained)
##  5     -2481.356           1.0   1533 17:19:47    0.0
##  6     -2395.858           1.0   1533 17:19:47    0.0
##  7     -2381.050           1.0   1533 17:19:47    0.0
##  8     -2380.251           1.0   1533 17:19:47    0.0
##  9     -2380.246           1.0   1533 17:19:47    0.0
```

Note that we have specified a covariance structure for each random effect and an estimate of the effect of sex on both birth weight and tarsus length by interacting sex with trait in the fixed effect structure.

There will now be twelve (co)variance components reported after running the code:

```
summary(modelb)$varcomp
```

```
##                                          component std.error    z.ratio
## trait:byear!trait_bwt:bwt                0.9746500 0.2825643  3.4493030
```

```
## trait:byear!trait_tarsus:bwt                 0.1624455 0.4185658  0.3881003
## trait:byear!trait_tarsus:tarsus              3.7384670 1.2069893  3.0973489
## trait:mother!trait_bwt:bwt                   1.1445030 0.2302062  4.9716424
## trait:mother!trait_tarsus:bwt               -1.5566953 0.4051695 -3.8420841
## trait:mother!trait_tarsus:tarsus             4.8206658 1.3200970  3.6517512
## trait:vm(animal, ainv)!trait_bwt:bwt         1.9893623 0.4410265  4.5107542
## trait:vm(animal, ainv)!trait_tarsus:bwt      3.3168101 0.9030192  3.6730229
## trait:vm(animal, ainv)!trait_tarsus:tarsus 10.2277887 2.8065673  3.6442343
## units:trait!R                                1.0000000        NA         NA
## units:trait!trait_bwt:bwt                    1.8443114 0.3443161  5.3564488
## units:trait!trait_tarsus:bwt                 4.0144042 0.7411868  5.4161844
## units:trait!trait_tarsus:tarsus             12.4857174 2.2890289  5.4545915
##                                             bound %ch
## trait:byear!trait_bwt:bwt                       P 0.0
## trait:byear!trait_tarsus:bwt                    P 0.0
## trait:byear!trait_tarsus:tarsus                 P 0.0
## trait:mother!trait_bwt:bwt                      P 0.0
## trait:mother!trait_tarsus:bwt                   P 0.0
## trait:mother!trait_tarsus:tarsus                P 0.0
## trait:vm(animal, ainv)!trait_bwt:bwt            P 0.0
## trait:vm(animal, ainv)!trait_tarsus:bwt         P 0.1
## trait:vm(animal, ainv)!trait_tarsus:tarsus      P 0.1
## units:trait!R                                   F 0.0
## units:trait!trait_bwt:bwt                       P 0.0
## units:trait!trait_tarsus:bwt                    P 0.0
## units:trait!trait_tarsus:tarsus                 P 0.1
```

3.2.3 Significance testing

Under the model above r_M is estimated as -0.66 and the `z.ratio` associated with the corresponding covariance (COV_M) is >2 (in absolute terms). We might therefore infer that there is evidence for a strong negative correlation between the traits with respect to the mother and that while maternal identity explains variance in both traits those mothers that tend to produce heavier offspring actually tend to produce offspring with shorter tarsus lengths.

To formally test if COV_M is significantly different from zero, we can compare the log-likelihood for this model:

```
modelb$loglik
```

```
## [1] -2380.246
```

to a model in which we specify that COV_M=0. Since this constraint reduces the number of parameters to be estimated by one, we can use a likelihood ratio test with one degree of freedom. To run the constrained model, we modify the G structure definition for the `mother` random effect to diagonal (`diag`), which means we only estimate the variances (the diagonal of the matrix) but not the covariance:

```
modelc <- asreml(
  fixed = cbind(bwt, tarsus) ~ trait + trait:sex,
  random = ~ us(trait, init = c(1, 0.1, 1)):vm(animal, ainv) +
    us(trait, init = c(1, 0.1, 1)):byear +
    diag(trait, init = c(1, 1)):mother,
  residual = ~ id(units):us(trait, init = c(1, 0.1, 1)),
  data = gryphon,
  maxit = 20
)
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:19:48 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -4677.820           1.0   1533 17:19:48    0.1
##  2     -4010.442           1.0   1533 17:19:48    0.0
##  3     -3275.409           1.0   1533 17:19:48    0.0
##  4     -2763.519           1.0   1533 17:19:48    0.0
##  5     -2483.732           1.0   1533 17:19:48    0.0
##  6     -2400.242           1.0   1533 17:19:48    0.0
##  7     -2386.663           1.0   1533 17:19:48    0.0
##  8     -2386.049           1.0   1533 17:19:48    0.0
##  9     -2386.045           1.0   1533 17:19:48    0.0
```

You can run `summary(modelc)$varcomp` to confirm this worked. We can now obtain the log-likelihood of this model and compare this to that of `modelb` using a likelihood ratio test:

```
modelc$loglik
```

```
## [1] -2386.045
```

We can see that the model log-likelihood is now -2386.05. And comparing the models using a likelihood ratio test:

```
2 * (modelb$loglik - modelc$loglik)
```

```
## [1] 11.59831
```

So our chi-square test statistic is χ^2_1= 11.6. The p-value that goes with this is obtained by:

```
1 - pchisq(2 * (modelb$loglik - modelc$loglik), 1)
```

```
## [1] 0.000660117
```

We would therefore conclude that the maternal covariance is significantly different from zero.

We could apply the same procedure to show that the residual (environmental) covariance and the genetic covariance estimates are significantly greater than zero (*i.e.*, heavier individuals tend to

have longer tarsus lengths). In contrast, we should find that the byear covariance between the two traits is non-significant.

3.3 gremlin

Not implemented yet

3.4 MCMCglmm

All others software will remove automatically lines with missing data from the dataset use to fit the model. `MCMCglmm`, however, will not do it and will try to fit the model use latent variables for missing data. For comparison, we will remove the missing values from the data before fitting the model.

```
gryphon <- subset(gryphon, !is.na(bwt) & !is.na(tarsus))
```

3.4.1 Fitting the model

Fitting a multivariate model in MCMCglmm involves several new consideration above those for fitting univariate models. First, we have to fit multivariate priors; second, we have to specify the ways in which effects on different traits may covary, including the nature of residual (co)variation; and third, we will have to be a little more specific when specifying to MCMCglmm what type of distributions from which we assume our data are drawn. Our most basic model can be specified as:

```
library(MCMCglmm)
Ainv <- inverseA(gryphonped)$Ainv
prior2.1 <- list(
  G = list(G1 = list(V = diag(2), nu = 1.002)),
  R = list(V = diag(2), nu = 1.002)
)
model2.1 <- MCMCglmm(cbind(bwt, tarsus) ~ trait - 1,
  random = ~ us(trait):animal, rcov = ~ us(trait):units, family = c("gaussian", "gaussian"
  ginv = list(animal = Ainv), data = gryphon, prior = prior2.1, verbose = FALSE
)
plot(model2.1$VCV[, "traittarsus:traittarsus.animal"])
```

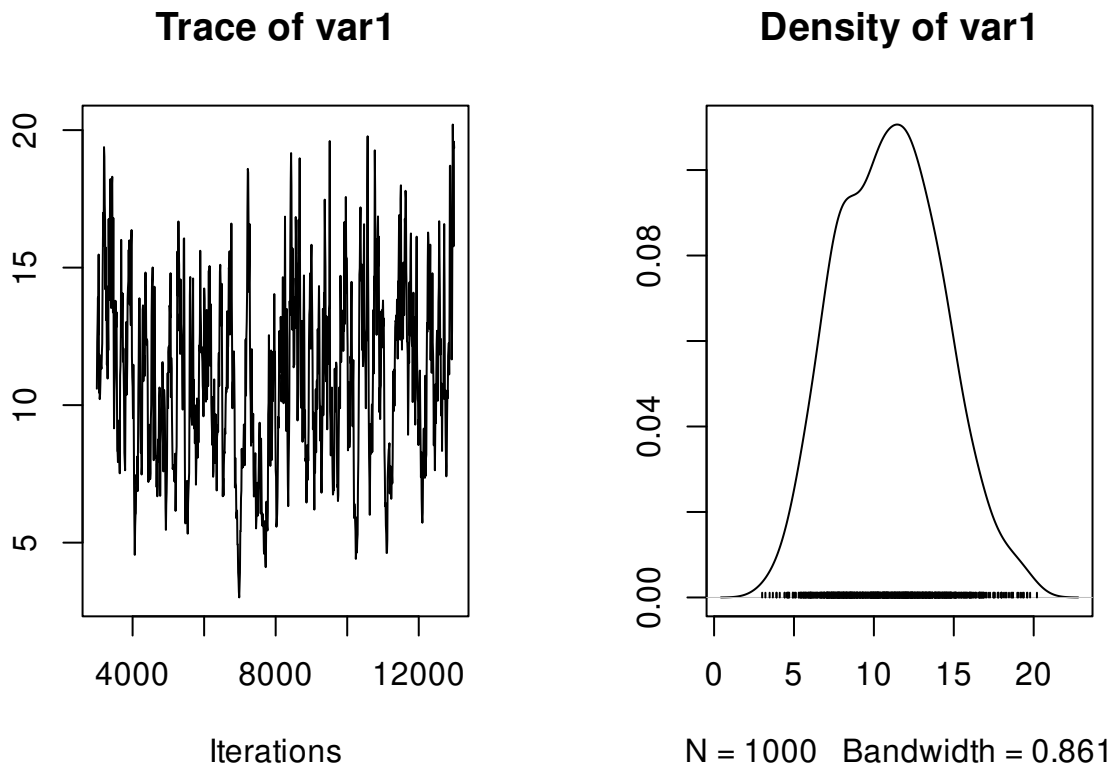


Figure 3.1: The posterior distribution of the additive genetic effect for tarsus length in a MCMCglmm run with default values

We have constructed the prior similarly to the those in the univariate models in tutorial 1, only we are specifying a 2x2 covariance matrix rather than a single variance. In order to provide proper priors, we have set the degree of belief parameter to greater than 1 (1.002). Those priors are not necessarily weak or uninformative in all circumstances. We will consider them adequate nonetheless for this tutorial. Please the vignette of the MCMCglmm packages (Hadfield, 2021) for more infomration on priors. In tutorial 1, we used full autocorrelation tables to evaluate the validity of the posterior distribution. Note that we have not done this here. For a bivariate model this table can become very complex. Nonetheless, it is worth evaluating, rather it is simply to large to include here. It can be viewed in the console as before. Here we have displayed only the autocorrelation for estimates of additive genetic effects for tarsus length with a lag of one samples (10 iterations given this MCMCglmm run with default values). This lag of 0.8457909 is clearly unacceptable. The posterior distribution of the additive genetic effect on tarsus length is shown in Figure 4 (p. 15), note the autocorrelation evident in the left-hand plot. We will opt to run the analysis for longer. This longer run could be run using the following code (including a line to save the output):

```
model2.1 <- MCMCglmm(cbind(bwt, tarsus) ~ trait - 1,
  random = ~ us(trait):animal,
  rcov = ~ us(trait):units,
  family = c("gaussian", "gaussian"),
  ginv = list(animal = Ainv), data = gryphon,
  nitt = 130000, thin = 100, burnin = 30000,
```

```
  prior = prior2.1, verbose = FALSE
)
save(model2.1, file = "data/MCMCglmm_model2_1_LongRun.rda")
```

However, this run might take as long as an hour. For the purpose of this tutorial we have provided an output for such a run. It can be obtained and manipulated as follows, assuming that the file `MCMCglmm_model2_1_LongRun.rda` is available at the specified location:

```
load(file = "data/MCMCglmm_model2_1_LongRun.rda")
autocorr.diag(model2.1$VCV)[, "traittarsus:traittarsus.animal"][2]
```

```
##   Lag 100
## 0.2608752
```

This level of autocorrelation is more acceptable, at least for the purpose of demonstration in this tutorial. We can recover variance components, heritabilities, and genetic correlations from the posterior distribution of this model:

```
posterior.mode(model2.1$VCV)
```

```
##       traitbwt:traitbwt.animal    traittarsus:traitbwt.animal
##                       3.370616                       2.581839
##    traitbwt:traittarsus.animal traittarsus:traittarsus.animal
##                       2.581839                      12.463915
##        traitbwt:traitbwt.units     traittarsus:traitbwt.units
##                       3.761401                       2.982413
##     traitbwt:traittarsus.units  traittarsus:traittarsus.units
##                       2.982413                      19.556443
```

```
heritability.bwt2.1 <- model2.1$VCV[, "traitbwt:traitbwt.animal"] / (model2.1$VCV[, "trait
posterior.mode(heritability.bwt2.1)
```

```
##      var1
## 0.4999336
```

```
heritability.tarsus2.1 <- model2.1$VCV[, "traittarsus:traittarsus.animal"] / (model2.1$VCV
posterior.mode(heritability.tarsus2.1)
```

```
##      var1
## 0.4038754
```

```
genetic.correlation2.1 <- model2.1$VCV[, "traitbwt:traittarsus.animal"] / sqrt(model2.1$VC
posterior.mode(genetic.correlation2.1)
```

```
##      var1
## 0.3691503
```

3.4.2 Adding fixed and random effects

Fixed and random effects can be added just as for the univariate case. Given that our full model of bwt from tutorial 1 had sex as a fixed effect as well as random effects of byear and mother we could specify a bivariate formulation of this using the following code (including a line to save the output):

```
prior2.2 <- list(
  G = list(
    G1 = list(V = diag(2), nu = 1.002),
    G2 = list(V = diag(2), nu = 1.002),
    G3 = list(V = diag(2), nu = 1.002)
  ),
  R = list(V = diag(2), nu = 1.002)
)
model2.2 <- MCMCglmm(cbind(bwt, tarsus) ~ trait - 1 + trait:sex,
  random = ~ us(trait):animal + us(trait):byear + us(trait):mother,
  rcov = ~ us(trait):units,
  family = c("gaussian", "gaussian"),
  ginv = list(animal = Ainv), data = gryphon,
  nitt = 130000, thin = 100, burnin = 30000,
  prior = prior2.2, verbose = FALSE
)
save(model2.2, file = "data/MCMCglmm_model2_2_LongRun.rda")
```

Again we have provided the data from one such run. It can be accessed using the code:

```
load(file = "data/MCMCglmm_model2_2_LongRun.rda")
autocorr(model2.2$VCV)[, , "traittarsus:traittarsus.animal"][3, 4]
```

```
## [1] 0.5231926
```

As before we can obtain the raw variance component estimates and genetic correlations for the random effects:

```
posterior.mode(model2.2$VCV)
```

```
##       traitbwt:traitbwt.animal    traittarsus:traitbwt.animal
##                     1.32761860                     1.98076467
##    traitbwt:traittarsus.animal traittarsus:traittarsus.animal
##                     1.98076467                     0.07827994
##        traitbwt:traitbwt.byear     traittarsus:traitbwt.byear
##                     0.83300490                    -0.14691430
```

```
##     traitbwt:traittarsus.byear  traittarsus:traittarsus.byear
##                     -0.14691430                     2.79242863
##       traitbwt:traitbwt.mother    traittarsus:traitbwt.mother
##                      1.29299116                    -1.97967150
##    traitbwt:traittarsus.mother traittarsus:traittarsus.mother
##                     -1.97967150                     3.71594577
##        traitbwt:traitbwt.units     traittarsus:traitbwt.units
##                      2.53632072                     5.22764927
##     traitbwt:traittarsus.units  traittarsus:traittarsus.units
##                      5.22764927                    16.57931045
```

```
genetic.correlation2.2 <- model2.2$VCV[, "traitbwt:traittarsus.animal"] / sqrt(model2.2$VC
maternal.correlation2.2 <- model2.2$VCV[, "traitbwt:traittarsus.mother"] / sqrt(model2.2$V
posterior.mode(genetic.correlation2.2)
```

```
##      var1
## 0.9942065
```

```
posterior.mode(maternal.correlation2.2)
```

```
##       var1
## -0.9920879
```

Evaluation of the statistical support for these genetic and maternal correlations is straightforward. Because we imposed no constraint on their estimation, we can evaluate the extent to which the posterior distributions overlap zero:

```
HPDinterval(genetic.correlation2.2, 0.95)
```

```
##          lower     upper
## var1 0.3935369 0.9990187
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(maternal.correlation2.2, 0.95)
```

```
##           lower      upper
## var1 -0.9980476 -0.9443838
## attr(,"Probability")
## [1] 0.95
```

Neither or these posterior distributions overlapps zero, so we can consider them both statistically supported.

3.5 brms

```
library(brms)
Amat <- as.matrix(nadiv::makeA(gryphonped))
bf_bwt <- bf(bwt ~ 1 + (1 | p | gr(animal, cov = Amat)))
bf_tarsus <- bf(tarsus ~ 1 + (1 | p | gr(animal, cov = Amat)))
brms_m2.1 <- brm(
  bf_bwt + bf_tarsus + set_rescor(TRUE),
  data = gryphon,
  data2 = list(Amat = Amat),
  chains = 2, cores = 2, iter = 1000
)
save(brms_m2.1, file = "data/brms_m2_1.rda")
```

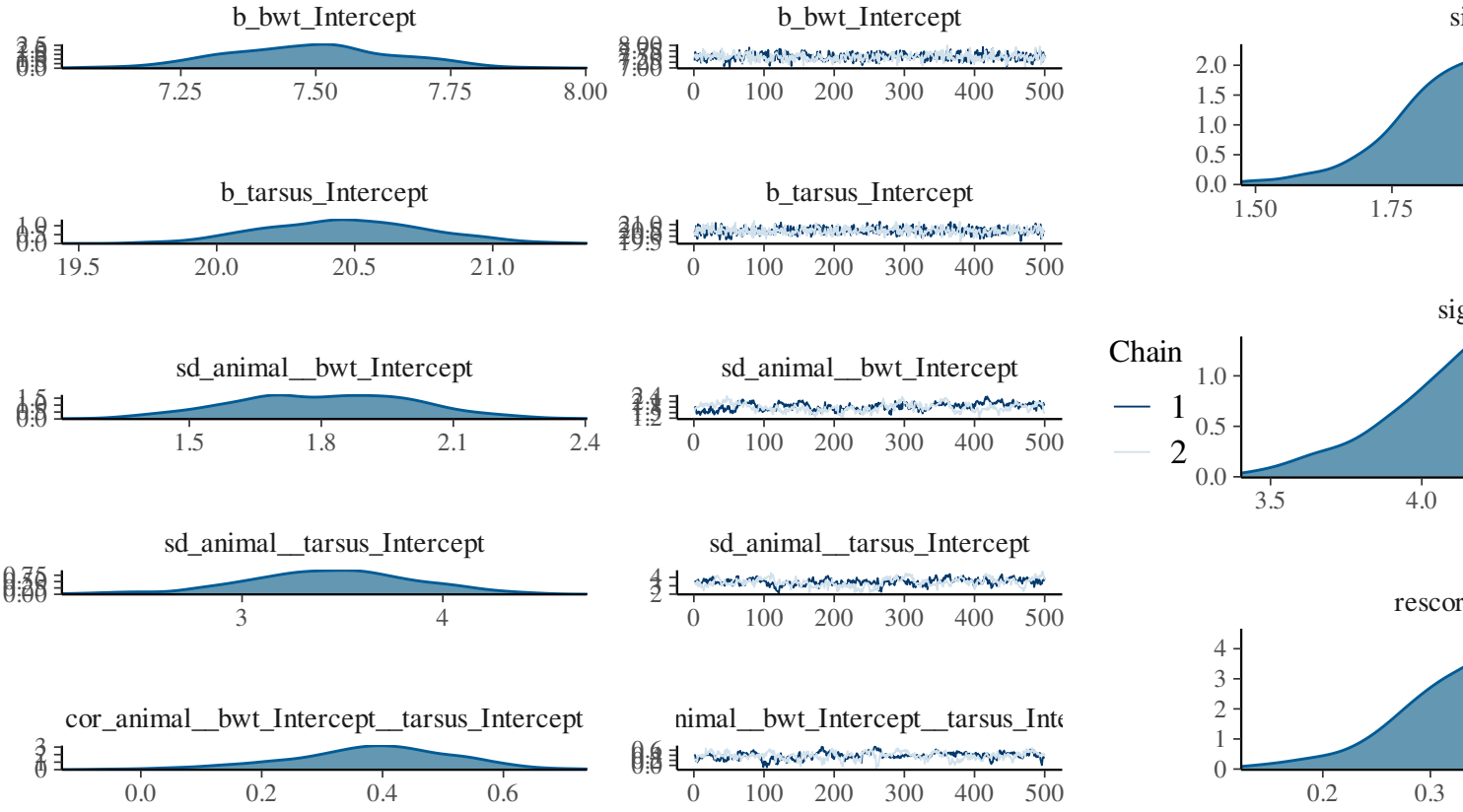
```
load("data/brms_m2_1.rda")
summary(brms_m2.1)
```

```
## Warning: Parts of the model have not converged (some Rhats are > 1.05). Be
## careful when analysing the results! We recommend running more iterations and/or
## setting stronger priors.
```

```
##  Family: MV(gaussian, gaussian)
##   Links: mu = identity; sigma = identity
##          mu = identity; sigma = identity
## Formula: bwt ~ 1 + (1 | p | gr(animal, cov = Amat))
##          tarsus ~ 1 + (1 | p | gr(animal, cov = Amat))
##    Data: gryphon (Number of observations: 683)
## Samples: 2 chains, each with iter = 1000; warmup = 500; thin = 1;
##          total post-warmup samples = 1000
##
## Group-Level Effects:
## ~animal (Number of levels: 683)
##                                     Estimate Est.Error l-95% CI u-95% CI Rhat
## sd(bwt_Intercept)                       1.81      0.21     1.41     2.20 1.06
## sd(tarsus_Intercept)                    3.44      0.43     2.49     4.25 1.05
## cor(bwt_Intercept,tarsus_Intercept)     0.38      0.14     0.08     0.62 1.02
##                                     Bulk_ESS Tail_ESS
## sd(bwt_Intercept)                         32      192
## sd(tarsus_Intercept)                      61      173
## cor(bwt_Intercept,tarsus_Intercept)      101      232
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## bwt_Intercept       7.49      0.16     7.20     7.79 1.00      607      839
## tarsus_Intercept   20.47      0.30    19.92    21.03 1.00      868      803
```

```
##
## Family Specific Parameters:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma_bwt        1.97      0.16     1.66     2.28 1.06       27      172
## sigma_tarsus     4.24      0.30     3.63     4.82 1.04       71      162
##
## Residual Correlations:
##                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## rescor(bwt,tarsus)     0.39      0.09     0.21     0.55 1.02       95      179
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(brms_m2.1)
```




```
VarCorr(brms_m2.1)
```

```
## $animal
## $animal$sd
##                   Estimate Est.Error     Q2.5    Q97.5
## bwt_Intercept    1.808171 0.2050233 1.412824 2.204805
## tarsus_Intercept 3.438368 0.4283612 2.491218 4.245264
##
```

```
## $animal$cor
## , , bwt_Intercept
##
##                   Estimate Est.Error       Q2.5     Q97.5
## bwt_Intercept    1.0000000 0.0000000 1.00000000 1.0000000
## tarsus_Intercept 0.3814062 0.1380014 0.07581464 0.6209038
##
## , , tarsus_Intercept
##
##                   Estimate Est.Error       Q2.5     Q97.5
## bwt_Intercept    0.3814062 0.1380014 0.07581464 0.6209038
## tarsus_Intercept 1.0000000 0.0000000 1.00000000 1.0000000
##
##
## $animal$cov
## , , bwt_Intercept
##
##                  Estimate Est.Error      Q2.5    Q97.5
## bwt_Intercept    3.311473 0.7430185 1.9960721 4.861167
## tarsus_Intercept 2.440166 1.0901689 0.3870783 4.668720
##
## , , tarsus_Intercept
##
##                   Estimate Est.Error      Q2.5    Q97.5
## bwt_Intercept     2.440166  1.090169 0.3870783  4.66872
## tarsus_Intercept 12.005688  2.918741 6.2061701 18.02226
##
##
##
## $residual__
## $residual__$sd
##        Estimate Est.Error     Q2.5    Q97.5
## bwt    1.970532 0.1597581 1.658782 2.276074
## tarsus 4.244704 0.2984518 3.632824 4.820109
##
## $residual__$cor
## , , bwt
##
##         Estimate  Est.Error      Q2.5     Q97.5
## bwt    1.0000000 0.00000000 1.0000000 1.0000000
## tarsus 0.3888754 0.08510488 0.2127907 0.5526631
##
## , , tarsus
##
##         Estimate  Est.Error      Q2.5     Q97.5
## bwt    0.3888754 0.08510488 0.2127907 0.5526631
## tarsus 1.0000000 0.00000000 1.0000000 1.0000000
```

```
##
##
## $residual__$cov
## , , bwt
##
##        Estimate Est.Error     Q2.5    Q97.5
## bwt    3.908493 0.6282892 2.751557 5.180511
## tarsus 3.289995 0.9305960 1.572647 5.147133
##
## , , tarsus
##
##         Estimate Est.Error      Q2.5     Q97.5
## bwt     3.289995  0.930596  1.572647  5.147133
## tarsus 18.106495  2.530138 13.197409 23.233452
```

Chapitre 4

A repeated measures animal model

This tutorial will demonstrate how to run a univariate animal model for a trait with repeated observations using different R packages and example data files provided.

4.1 Scenario and data

4.1.1 scenario

Since gryphons are iteroparous, multiple observations of reproductive traits are available for some individuals. Here we have repeated measures of lay date (measured in days after January 1) for individual females varying in age from 2 (age of maturation) up until age 6. Not all females lay every year so the number of observations per female is variable. We want to know how repeatable the trait is, and (assuming it is repeatable) how heritable it is.

4.1.2 Data files

The pedigree file `gryphonped.csv` is that used in the preceding tutorials but we now use a new data file `gryphonRM.csv`. Columns correspond to individual identity (`animal`), birth year (`byear`), age in years (`age`), year of measurement (`year`) and lay date (`laydate`). Each row of the data file corresponds to a single phenotypic observation. Here data are sorted by identity and then age so that the repeated observations on individuals are readily apparent. However this is not a requirement for analysis - data could equally be sorted by some other variable (*e.g.*, measurement year) or be in a random order.

```
str(gryphonRM)
```

```
## 'data.frame':    1607 obs. of  5 variables:
##  $ animal : Factor w/ 469 levels "1","2","3","8",..: 1 1 1 1 1 2 2 2 3 3 ...
##  $ byear  : Factor w/ 34 levels "968","970","971",..: 22 22 22 22 22 22 22 22 22 22 ...
##  $ age    : Factor w/ 5 levels "2","3","4","5",..: 1 2 3 4 5 1 2 3 1 2 ...
##  $ year   : Factor w/ 39 levels "970","971","972",..: 23 24 25 26 27 23 24 25 23 24 ...
##  $ laydate: num  19 23 24 23 29 21 17 21 20 20 ...
```

```
head(gryphonRM)
```

```
##   animal byear age year laydate
## 1      1   990   2  992      19
## 2      1   990   3  993      23
## 3      1   990   4  994      24
## 4      1   990   5  995      23
## 5      1   990   6  996      29
## 6      2   990   2  992      21
```

```
## 'data.frame':    1309 obs. of  3 variables:
##  $ id    : int  1306 1304 1298 1293 1290 1288 1284 1283 1282 1278 ...
##  $ father: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ mother: int  NA NA NA NA NA NA NA NA NA NA ...
```

```
gryphonped$id<-as.factor(gryphonped$id)
gryphonped$father<-as.factor(gryphonped$father)
gryphonped$mother<-as.factor(gryphonped$mother)
```

4.2 Asreml-R

4.2.1 Estimating repeatability

With repeated measures on individuals it is often of interest, prior to fitting a genetic model, to see how repeatable a trait is. We can estimate the repeatability of a trait as the proportion of phenotypic variance explained by individual identity.

```
modelv <- asreml(fixed = laydate ~ 1,
                 random =~ animal,
                 residual=~idv(units),
                 data=gryphonRM,
                 na.action = na.method(x="omit", y="omit"))
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:20:24 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1    -10182.83           1.0   1606 17:20:24    0.0
##  2     -8266.10           1.0   1606 17:20:24    0.0
##  3     -6145.01           1.0   1606 17:20:24    0.0
##  4     -4651.57           1.0   1606 17:20:24    0.0
##  5     -3819.31           1.0   1606 17:20:24    0.0
##  6     -3554.22           1.0   1606 17:20:24    0.0
##  7     -3501.56           1.0   1606 17:20:24    0.0
##  8     -3497.58           1.0   1606 17:20:24    0.0
```

```
##  9      -3497.54           1.0   1606 17:20:24    0.0
## 10      -3497.54           1.0   1606 17:20:24    0.0
```

Note that since we want to estimate the amount of variance explained by individual identity (rather than by additive effects), we fit `animal` as a normal random effect and we don't associate it with the pedigree.

This model partitions the phenotypic variance in `laydate` as follows:

```
summary(modelv)$varcomp
```

```
##              component std.error   z.ratio bound %ch
## animal        11.08634 1.1794319  9.399728     P   0
## units!units   21.29643 0.8896196 23.938798     P   0
## units!R        1.00000        NA        NA     F   0
```

Between-individual variance is given by the `animal` component, while the residual component (`units!units`) represents within-individual variance. Here then the repeatability of the trait can be determined by hand as 0.34 (*i.e.*, as 11.086/(11.086 + 21.296)).

Mean lay date might change with age, so we could ask what the repeatability of lay date is after conditioning on age. This would be done by adding `age` into the model as a fixed effect.

```
modelw <- asreml(fixed = laydate ~ age,
                 random =~ animal,
                 residual=~idv(units),
                 data=gryphonRM,
                 na.action = na.method(x="omit", y="omit"))
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:20:24 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -8402.968           1.0   1602 17:20:24    0.0
##  2     -6912.361           1.0   1602 17:20:24    0.0
##  3     -5274.379           1.0   1602 17:20:24    0.0
##  4     -4143.634           1.0   1602 17:20:24    0.0
##  5     -3541.895           1.0   1602 17:20:24    0.0
##  6     -3372.909           1.0   1602 17:20:24    0.0
##  7     -3347.670           1.0   1602 17:20:24    0.0
##  8     -3346.655           1.0   1602 17:20:24    0.0
##  9     -3346.652           1.0   1602 17:20:24    0.0
```

```
summary(modelw)$varcomp
```

```
##              component std.error  z.ratio bound %ch
## animal        12.28982  1.156115 10.63027     P   0
## units!units   16.37989  0.686619 23.85586     P   0
```

```
## units!R       1.00000        NA       NA     F    0
```

The repeatability of lay date, after accounting for age effects, is now estimated as 0.43 (*i.e.*, as $12.29/(12.29 + 16.38)$). So, just as we saw when estimating h^2 in Tutorial 1, the inclusion of fixed effects will alter the estimated effect size if we determine total phenotypic variance as the sum of the variance components. Thus, proper interpretation is vital.

Here age is modelled as a 5-level factor (specified using the function `as.factor()` at the beginning of the analysis). We could equally have fitted it as a continuous variable, in which case, given potential for a late life decline, we would probably also include a quadratic term.

4.2.2 Partitioning additive and permanent environment effects

Generally we expect that the repeatability will set the upper limit for heritability since, while additive genetic effects will cause among-individual variation, so will other types of effects. Non-additive contributions to fixed among-individual differences are normally referred to as *permanent environment effects*. If a trait has repeated measures then it is necessary to model permanent environment effects in an animal model to prevent upward bias in V_A.

To illustrate this fit the animal model:

```
modelx <- asreml(fixed = laydate ~ age,
                 random =~ vm(animal, ainv),
                 residual=~idv(units),
                 data=gryphonRM,
                 na.action = na.method(x="omit", y="omit"))
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:20:24 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -8751.390           1.0   1602 17:20:24    0.0
##  2     -7169.205           1.0   1602 17:20:24    0.0
##  3     -5427.604           1.0   1602 17:20:24    0.0
##  4     -4219.598           1.0   1602 17:20:24    0.0
##  5     -3569.815           1.0   1602 17:20:24    0.0
##  6     -3382.341           1.0   1602 17:20:24    0.0
##  7     -3352.867           1.0   1602 17:20:24    0.0
##  8     -3351.565           1.0   1602 17:20:24    0.0
##  9     -3351.560           1.0   1602 17:20:24    0.0
```

Variance components are almost unchanged:

```
summary.asreml(modelx)$varcomp
```

```
##                  component std.error   z.ratio bound %ch
## vm(animal, ainv)  13.91784  1.443968  9.638607     P   0
## units!units       16.84008  0.707365 23.806768     P   0
## units!R            1.00000        NA        NA     F   0
```

This suggests that all of the among-individual variance is – rightly or wrongly – being partitioned as V_A here. To instead obtain an unbiased estimate of V_A we need to allow for both additive genetic *and* non-genetic sources of individual variation. We do this by fitting `animal` twice, once with a pedigree, and once without a pedigree (using `ide()`).

```
modely <- asreml(fixed = laydate ~ age,
                 random =~ vm(animal, ainv) + ide(animal),
                 residual=~idv(units),
                 data=gryphonRM,
                 na.action = na.method(x="omit", y="omit"))
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:20:24 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -7731.394           1.0   1602 17:20:24    0.0
##  2     -6426.548           1.0   1602 17:20:24    0.0
##  3     -4997.252           1.0   1602 17:20:24    0.0
##  4     -4018.486           1.0   1602 17:20:24    0.0
##  5     -3504.988           1.0   1602 17:20:24    0.0
##  6     -3363.160           1.0   1602 17:20:24    0.0
##  7     -3341.611           1.0   1602 17:20:24    0.0
##  8     -3340.682           1.0   1602 17:20:24    0.0
##  9     -3340.679           1.0   1602 17:20:24    0.0
```

```
summary(modely)$varcomp
```

```
##                   component std.error   z.ratio bound %ch
## vm(animal, ainv)  4.876101 1.8087709  2.695809     P   0
## ide(animal)       7.400983 1.7280113  4.282948     P   0
## units!units      16.380188 0.6866189 23.856300     P   0
## units!R           1.000000        NA        NA     F   0
```

The estimate of V_A is now much lower since the additive and permanent environment effects are being properly separated. We can estimate h^2 and the repeatability from this model:

```
vpredict(modely, h2 ~ V1/(V1+V2+V3))
```

```
##     Estimate         SE
## h2 0.1701523 0.06073974
```

```
vpredict(modely, repeatability ~ (V1+V2)/(V1+V2+V3))
```

```
##                Estimate         SE
## repeatability 0.4284108 0.02741602
```

4.2.3 Adding additional effects and testing significance

Models of repeated measures can be extended to include other fixed or random effects. For example try including year of measurement (`year`) and birth year (`byear`) as random effects.

```
modelz <- asreml(fixed = laydate ~ age,
                 random =~ vm(animal, ainv) + ide(animal) +
                 year + byear,
                 residual=~idv(units),
                 data=gryphonRM,
                 na.action = na.method(x="omit", y="omit"))
```

```
## Model fitted using the sigma parameterization.
## ASReml 4.1.0 Fri Feb 12 17:20:24 2021
##           LogLik        Sigma2     DF     wall    cpu
##  1     -4650.748           1.0   1602 17:20:24    0.0
##  2     -4088.264           1.0   1602 17:20:24    0.0
##  3     -3494.147           1.0   1602 17:20:24    0.0
##  4     -3127.161           1.0   1602 17:20:25    0.0 (1 restrained)
##  5     -2976.449           1.0   1602 17:20:25    0.0 (1 restrained)
##  6     -2955.785           1.0   1602 17:20:25    0.0 (1 restrained)
##  7     -2955.097           1.0   1602 17:20:25    0.0 (1 restrained)
##  8     -2955.095           1.0   1602 17:20:25    0.0 (1 restrained)
##  9     -2955.095           1.0   1602 17:20:25    0.0
```

```
summary(modelz)$varcomp
```

```
##                      component std.error   z.ratio bound %ch
## byear             1.650876e-07        NA        NA     B   0
## year              7.938576e+00 1.9344619  4.103765     P   0
## vm(animal, ainv)  4.815136e+00 1.6682351  2.886365     P   0
## ide(animal)       8.433325e+00 1.5495778  5.442337     P   0
## units!units       7.795560e+00 0.3324411 23.449443     P   0
## units!R           1.000000e+00        NA        NA     F   0
```

This model will return additional variance components corresponding to variation in lay dates between years of measurement and between birth cohorts of females. V_{byear} is very low and if you compare this model to a reduced model with byear excluded the log-likelihood remains unchanged.

`year` effects could alternatively be included as fixed effects (try this!). This will reduce V_R and increase the estimates of heritability and repeatability, which must now be interpreted as proportions of phenotypic variance after conditioning on both age and year of measurement effects.

4.3 gremlin

4.4 MCMCglmm

4.4.1 Estimating repeatability

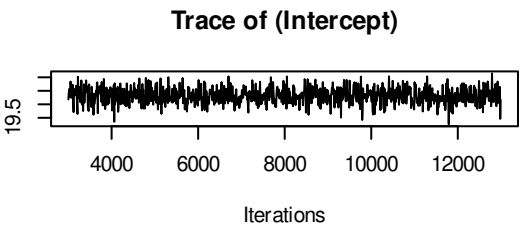
With repeated measures on individuals it is often of interest, prior to fitting a genetic model, to see how repeatable a trait is. We can estimate the repeatability of a trait as the proportion of phenotypic variance explained by individual identity using the commands below

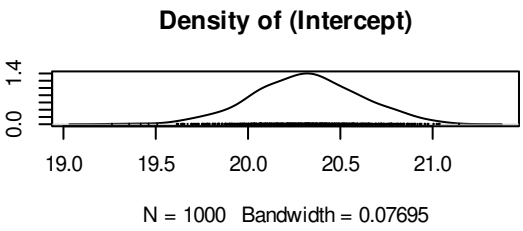
```
p.var <- var(gryphonRM$laydate, na.rm = TRUE)
prior3.1 <- list(G = list(G1 = list(V = 1, nu = 0.002)), R = list(V = 1,
nu = 0.002))
model3.1 <- MCMCglmm(laydate ~ 1, random = ~ animal, data = gryphonRM,
prior = prior3.1, verbose = FALSE)
posterior.mode(model3.1$VCV)
```

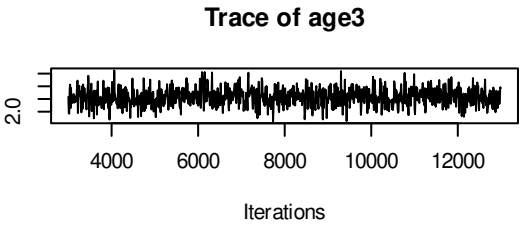
```
##   animal    units
## 10.87766 20.78441
```

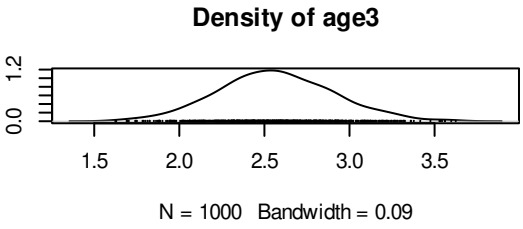
Note the use of the term `animal`, since neither the `pedigree` nor the `ginv` argument are provided, it represents the between-individual variance, while the residual compo- nent (Variance) therefore represents within-individual variance. Here then the repeata- bility of the trait can be determined by as 0.353 (i.e., 11.4532/(11.4532+21.1432). Given that we set up the simulation such that mean lay date changes with age (initially increas- ing to age 5 before a late life decline) we might ask what the repeatability of lay date is after conditioning on age effect. This would be done by adding age into the model as a fixed effect.

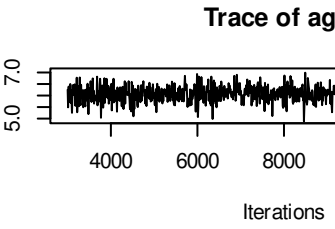
```
model3.2 <- MCMCglmm(laydate ~ age, random = ~ animal, data = gryphonRM,
prior = prior3.1, verbose = FALSE)
plot(model3.2$Sol)
```

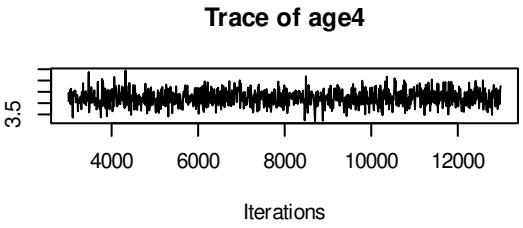


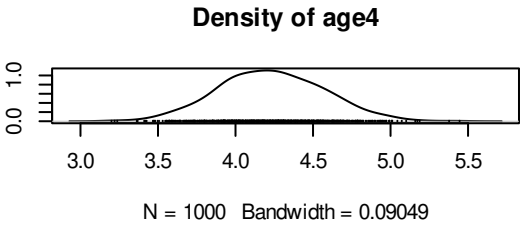


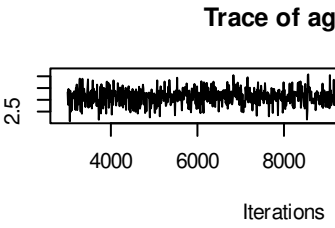




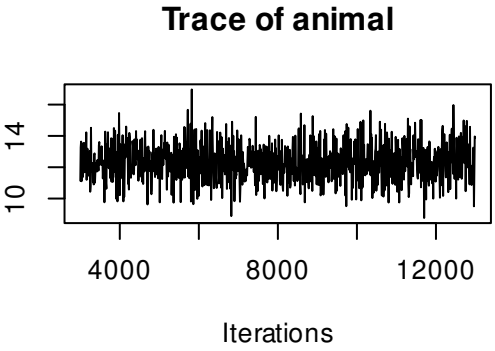


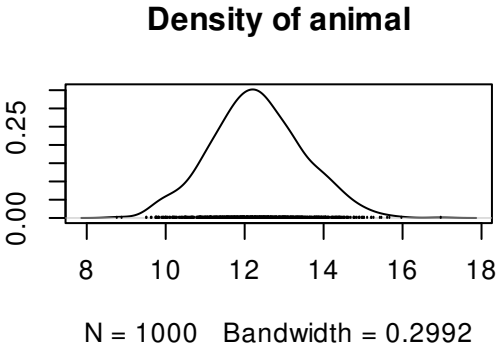


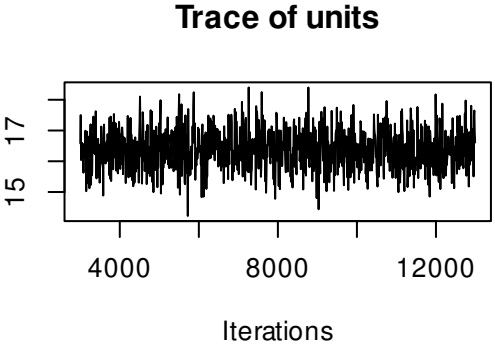


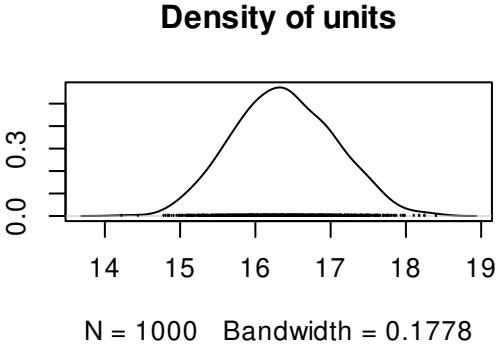



```
plot(model3.2$VCV)
```









```
posterior.mode(model3.2$VCV)
```

```
##   animal    units
## 11.95532 16.35622
```

Note that the random effect structure has remained unchaged, and so we have not modified the prior between models 3.1 and 3.2. So that the repeatability of laydate, after accounting for age effects, is now estimated as 0.445 (i.e., 12.6379/(12.6379+16.465). So, just as we saw when estimating h 2 in tutorial 1, the inclusion of fixed effects will alter the estimated effect size if we determine total phenotypic variance as the sum of the variance components. Thus, proper interpretation is vital.

Here age is modelled as a 5 level factor (see the convertion of age to a factor in section 3.2). We could equally have fitted it as a continuous variable instead in which case, given the late life decline, we would probably also include a quadratic term.

4.4.2 Partitioning additive and permanent environment effects

Generally we expect that the repeatability will set the upper limit for heritability since, while additive genetic effects will cause among-individual variation, so will other types of effect. Non-additive contributions to fixed among-individual differences are normally referred to as 'permanent environment effects', although 'non-heritable effects' that are consistent within individuals may be a better way to think of modelling this effect. If a trait has repeated measures then it is necessary to model permanent environment effects in an animal model to prevent upward bias in V A . To illustrate this fit the animal model

```
Ainv <- inverseA(gryphonped)$Ainv
model3.3 <- MCMCglmm(laydate ~ 1 + age, random = ~ animal, ginv = list(animal = Ainv),
  data = gryphonRM, prior = prior3.1, verbose = FALSE)
posterior.mode(model3.3$VCV)
```

```
##   animal    units
## 13.48495 17.16223
```

This suggests that all of the among-individual variance is - rightly or wrongly - being partitioned as V A here. In fact here the partition is wrong since the simulation included both additive genetic effects and additional fixed heterogeneity that was not associated with the pedigree structure (i.e. permanent environment effects). In order to fit both permanent environemnt and additive genetic effects, we need to fit the individual identity twice in the model: once linked to the pedigree (genetic effect) and once not linked to the pedigree (permanenet environemnt effect). To do so, we need to duplicate the variable containing the individual identity and give it a new name. An more appropriate estimate of V A is given by the model:

```
gryphonRM$animal_pe <- gryphonRM$animal
p.var <- var(gryphonRM$laydate, na.rm = TRUE)
prior3.4 <- list(G = list(G1 = list(V = 1, nu = 0.002), G2 = list(V = 1,
nu = 0.002)), R = list(V = 1, nu = 0.002))
```

```
model3.4 <- MCMCglmm(laydate ~ 1 + age, random = ~animal + animal_pe,
ginv = list(animal = Ainv), data = gryphonRM, prior = prior3.4, verbose = FALSE)
posterior.mode(model3.4$VCV)
```

```
##    animal animal_pe     units
##  4.792356  7.586770 16.585880
```

The estimate of V A is now much lower (reduced from 13.6735 to 5.1238) since the ad- ditive and permanent environment effects are being properly separated. We could obtain estimates of h 2 and of the repeatability from this model using the following commands:

```
model3.4.VP <- model3.4$VCV[, "animal"] + model3.4$VCV[, "animal_pe"] + model3.4$VCV[, "un
model3.4.PE_VA <- model3.4$VCV[, "animal"] + model3.4$VCV[,"animal_pe"]
posterior.mode(model3.4.PE_VA/model3.4.VP)
```

```
##      var1
## 0.4163568
```

```
posterior.mode(model3.4$VCV[, "animal"]/model3.4.VP)
```

```
##     var1
## 0.159411
```

4.4.3 Adding additional effects and testing significance

Models of repeated measures can be extended to include other fixed or random effects. For example try including year of measurement (year).

```
p.var <- var(gryphonRM$laydate, na.rm = TRUE)
prior3.5 <- list(G = list(G1 = list(V = 1, nu = 0.002), G2 = list(V = 1,
nu = 0.002), G3 = list(V = 1, nu = 0.002), G4 = list(V = 1,
nu = 0.002)), R = list(V = 1, nu = 0.002))
model3.5 <- MCMCglmm(laydate ~ 1 + age, random = ~animal + animal_pe +
year + byear, ginv = list(animal = Ainv), data = gryphonRM, prior = prior3.5,
verbose = FALSE)
posterior.mode(model3.5$VCV)
```

```
##     animal  animal_pe       year      byear      units
## 5.21481674 7.87072250 7.91353832 0.00114482 7.78745344
```

This model will return additional variance components corresponding to year of mea- surement effects and birth year (of the female effects). The latter were not simulated as should be apparent from the parameter estimate (and by the support interval derivable from the posterior distribution and from DIC-based comparison of model3.5 and a model from which the birth year term had been eliminated, see tutorial 1). However, year ef- fects were simulated as should be apparent from the

from the modal estimate and from the support interval (try this yourself using HPDinterval()) and this could be formally confirmed by comparison of DIC. year effects could alternatively be included as fixed effects (try this, you should be able to handle the new prior specification at this point). Since we simulated large year of measurement effects this treatment will reduce V R and increase the the estimates of heritability and repeatability which must now be interpreted as proportions of phenotypic variance after conditioning on both age and year of measurement effects.

4.5 brms

```
library(brms)
Amat <- nadiv::makeA(Ped)

model_simple1.1 <- brm(
  bwt ~ 1 + (1|animal) + (1|animal_pe), data = gryphonRM,
  family = gaussian(), cov_ranef = list(animal = Amat),
  chains = 2, cores = 2, iter = 1000
)

summary(model_simple1.1)
plot(model_simple1.1)
```

Chapitre 5

Quick comparison of codes

5.1 Univariate model with repeated measures

5.1.1 Asreml-R

5.1.2 gremlin

5.1.3 MCMCglmm

5.1.4 brms

5.2 bivariate model

5.2.1 Asreml-R

5.2.2 gremlin

5.2.3 MCMCglmm

5.2.4 brms

Bibliography

Hadfield, J. (2021). *MCMCglmm: MCMC Generalised Linear Mixed Models*. R package version 2.30.

Hadfield, J. D. (2010). Mcmc methods for multi-response generalized linear mixed models: The MCMCglmm R package. *Journal of Statistical Software*, 33(2):1–22.

Wilson, A. J., Réale, D., Clements, M. N., Morrissey, M. M., Postma, E., Walling, C. A., Kruuk, L. E. B., and Nussey, D. H. (2010). An ecologist's guide to the animal model. *Journal of Animal Ecology*, 79(1):13–26.