Supplementary File 5: *Tutorial for MCMCglmm version*

Tutorial 1 (MCMCglmm) - Estimating the heritability of birth weight

This tutorial will demonstrate how to run a univariate animal model using the `R` package `MCMCglmm` and example data files provided. Additionally, 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 ways of 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 tutorial, which is available from CRAN (the Comprehensive R Archive Network, `http://cran.r-project.org/`, or can be accessed in `R` using the command `vignette("Tutorial","MCMCglmm")`). These tutorials (1-3) 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 in the documentation that is distributed with `MCMCglmm`, and available from CRAN.

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.

Data files

Open `gryphonped.txt` and `gryphon.txt` in your text editor. The structure and contents of these files is fairly self-explanatory. The pedigree file `gryphonped.txt` contains three columns containing unique identifiers 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.txt. Columns correspond to individual identity (`ANIMAL`), maternal identity (`MOTHER`), year of birth (`BYEAR`), sex (`SEX`,1=female, 2=male), birth weight (`BWT`), and tarsus length (`TARSUS`). Each row of the data file contains a record for a different individual.

`gryphonped.txt` and `gryphon.txt` were prepared specifically for use with `ASREML`. However, tab delimited files such as these are easily read into `R`. The following code reads the data in the two files into data frames called Ped and Data (the names are arbitrary, almost any character string will do for names). This code also modifies some of the structure of the data in some simple ways. First, we rename the column origianally called ANIMAL to lower case, as the lower case "animal" is a reserved word for `MCMCglmm` that we will use when fitting animal models. This code also ensures that `R` recognizes those data that we will use as factors and numeric data as such. The `head()` commands show us just the first few lines of the phenotypic and pedigree data frames, just so that we can reassure ourselves that we've read in and handled the data properly. We could view the entire data frames by typing their names and pressing return.

```
> Data <- as.data.frame(read.table(file =
+       "./gryphon.txt", header = TRUE))
> names(Data)[1] <- "animal"
> Data$animal <- as.factor(Data$animal)
> Data$MOTHER <- as.factor(Data$MOTHER)
> Data$BYEAR <- as.factor(Data$BYEAR)
> Data$SEX <- as.factor(Data$SEX)
> Data$BWT <- as.numeric(Data$BWT)
> Data$TARSUS <- as.numeric(Data$TARSUS)
> head(Data)
```

```
  animal MOTHER BYEAR SEX   BWT TARSUS
1   1029   1145   968   1 10.77  24.77
2   1299    811   968   1  9.30  22.46
3    643    642   970   2  3.98  12.89
4   1183   1186   970   1  5.39  20.47
5   1238   1237   970   2 12.12     NA
6    891    895   970   1    NA     NA
```

```
> Ped <- as.data.frame(read.table(file = "./gryphonped.txt", header = TRUE))
> for (x in 1:3) Ped[, x] <- as.factor(Ped[, x])
> head(Ped)
```

```
    ID FATHER MOTHER
1 1306   <NA>   <NA>
2 1304   <NA>   <NA>
3 1298   <NA>   <NA>
4 1293   <NA>   <NA>
5 1290   <NA>   <NA>
6 1288   <NA>   <NA>
```

This code is intended to run in a Linux/Unix environment. The only difference on a PC is that the file location would have to be specified slightly differently. For example if the files were in a folder called `JAE_MCMCglmm` on the root `C:` drive, the two lines above that read the data might be

```
Data<-as.data.frame(read.table(file=
                    "C:\JAE_MCMCglmm\gryphon.txt",header=TRUE))
Ped<-as.data.frame(read.table(file=
                    "C:\JAE_MCMCglmm\gryphonped.txt",header=TRUE))
```

Similarly on a Mac (OSX) if the files were in a folder called `JAE_MCMCglmm` in the documents area, the two lines above that read the data might be

```
Data<-as.data.frame(read.table(file=
    "//users//name/documents/JAE_MCMCglmm/gryphon.txt",header=TRUE))
Ped<-as.data.frame(read.table(file=
    "//users//name/documents/JAE_MCMCglmm/gryphonped.txt",header=TRUE))
```

Running the model

First load MCMCglmm:

```
> library(MCMCglmm)
```

You should check that you have the most current version of `MCMCglmm` witih the command `help(package=MCMCglmm)`. You can check the number of the current version on CRAN. If you need to update (or install) `MCMCglmm`, use `install.packages()` and follow the prompted instructions.

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 a prior, but we will tell `MCMCglmm` to pay very little heed to its specific value. 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))
```

We have provided a guess for the animal (the G1 structure) and residual effects (the R structure) by dividing the observed phenotypic variation in birth weight in half (the V terms). We have made this assumption for the purposes of this tutorial, but in reality careful consideration of priors is necessary. We have told `MCMCglmm` to pay little heed to these numbers by specifying degree of belief parameters (`n`) of 1. Since this is a univariate analysis, the priors are matricies of order 1 and thus n = 1 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:

```
> model1.1 <- MCMCglmm(BWT ~ 1, random = ~animal, pedigree = Ped,
+     data = Data, prior = prior1.1)
```

```
                         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

In addition: Warning messages:
1: In MCMCglmm(BWT ~ 1, random = ~animal, pedigree = Ped, data = Data,  :
  some combinations in animal do not exist and 225 missing records have
  been generated
```

After typing this code, MCMCglmm will run, taking about 20 seconds on a modern desktop computer. The R console may temporarily appear to crash under windows. Unless an error message appears, just let it run, the jitters result from poor communication between R and the C routines in MCMCglmm. 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)
```

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,

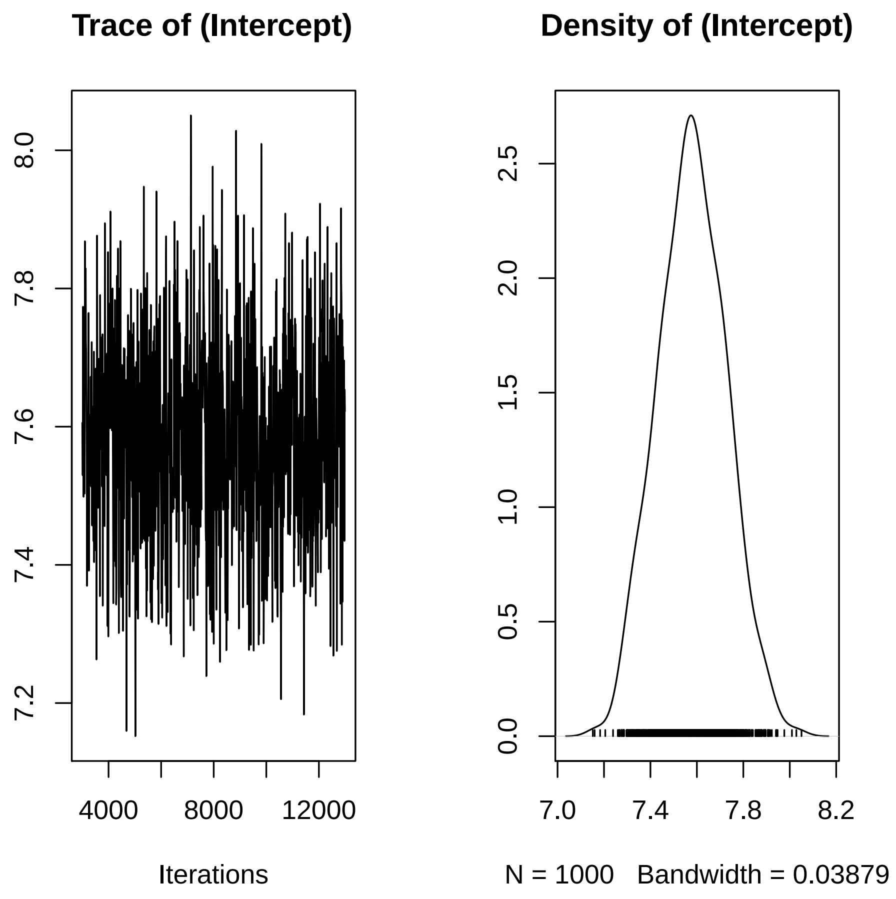

Figure 1: The posterior distribution of the fixed effect (the intercept, or mean) in model 1.1.

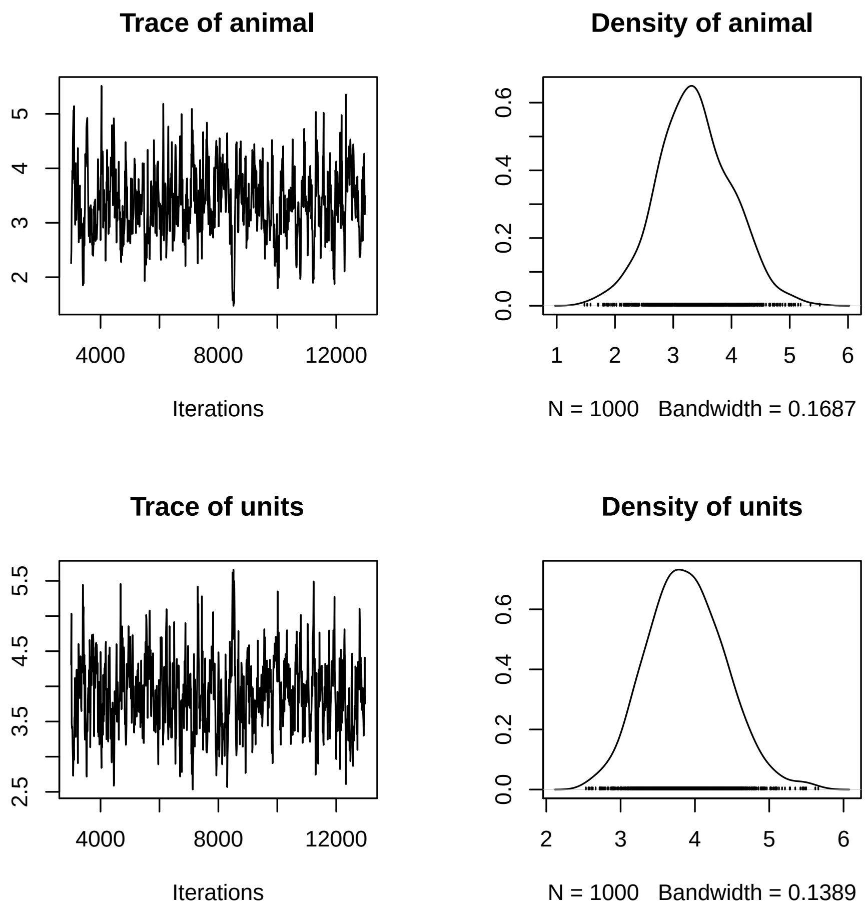

Figure 2: The posterior distributions of the variance components of model 1.1, based on an analysis with the default values for nitt, burnin, and thin in MCMCglmm.

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 tutorial, 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 interest 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)
```

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 autocorrelation, 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 (estimates 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, pedigree = Ped,
+      data = Data, nitt = 65000, thin = 50,
+      burnin = 15000, prior = prior1.1,
+      verbose = FALSE)
```

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(model1.1$VCV)

, , animal

              animal       units
Lag 0     1.00000000 -0.79054324
Lag 50    0.14876528 -0.16511231
Lag 250  -0.04273714  0.02747032
Lag 500   0.02057043 -0.02496050
Lag 2500  0.04595415 -0.05561558

, , units

               animal        units
Lag 0    -0.790543237  1.000000000
Lag 50   -0.142312942  0.149116354
Lag 250   0.040387068 -0.028593604
Lag 500   0.017841149  0.007001413
Lag 2500 -0.003795404  0.010794216
```

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
3.327839 3.915565
```

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 distributions:

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

```
          lower    upper
animal 2.302286 4.655025
units  3.002575 4.901992
attr(,"Probability")
[1] 0.95
```

We specified very 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. Rather than splitting the observed phenotypic variance evenly between the priors for the genetic and residual effects, we will construct priors with the same degree of belief parameters, but we will specify a much larger proportion of genetic control:

```
> p.var <- var(Data$BWT, na.rm = TRUE)
> prior1.1.2 <- list(G = list(G1 = list(V = matrix(p.var * 0.05),
+     n = 1)), R = list(V = matrix(p.var * 0.95), n = 1))
> model1.1.2 <- MCMCglmm(BWT ~ 1, random = ~animal, pedigree = Ped,
+     data = Data, prior = prior1.1.2, nitt = 65000, thin = 50,
+     burnin = 15000, verbose = FALSE)
> posterior.mode(model1.1$VCV)
```

```
  animal    units
3.327839 3.915565
```

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

```
  animal    units
3.281250 3.988396
```

and we can therefore conclude that the prior 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. Were this not the case, it is possible that numerical problems would have been encountered by `MCMCglmm`, and priors such as those used to get model1.1.2 would have had to have been used. In all cases, it is important to check the effect of priors on conclusions drawn from a model.

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"] + model1.1$VCV[, "units"])
> HPDinterval(posterior.heritability1.1, 0.95)

         lower     upper
var1 0.3286145 0.6111165
attr(,"Probability")
[1] 0.95

> posterior.mode(posterior.heritability1.1)

     var1
0.4820407
```

Generate a plot of the posterior distribution of this heritability estimate (Figure 3, p. 10):

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

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,
+     pedigree = Ped, data = Data, prior = prior1.1,
+     nitt = 65000, thin = 50, burnin = 15000,
+     verbose = FALSE)
```

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

```
> posterior.mode(model1.2$Sol[, "SEX2"])
```

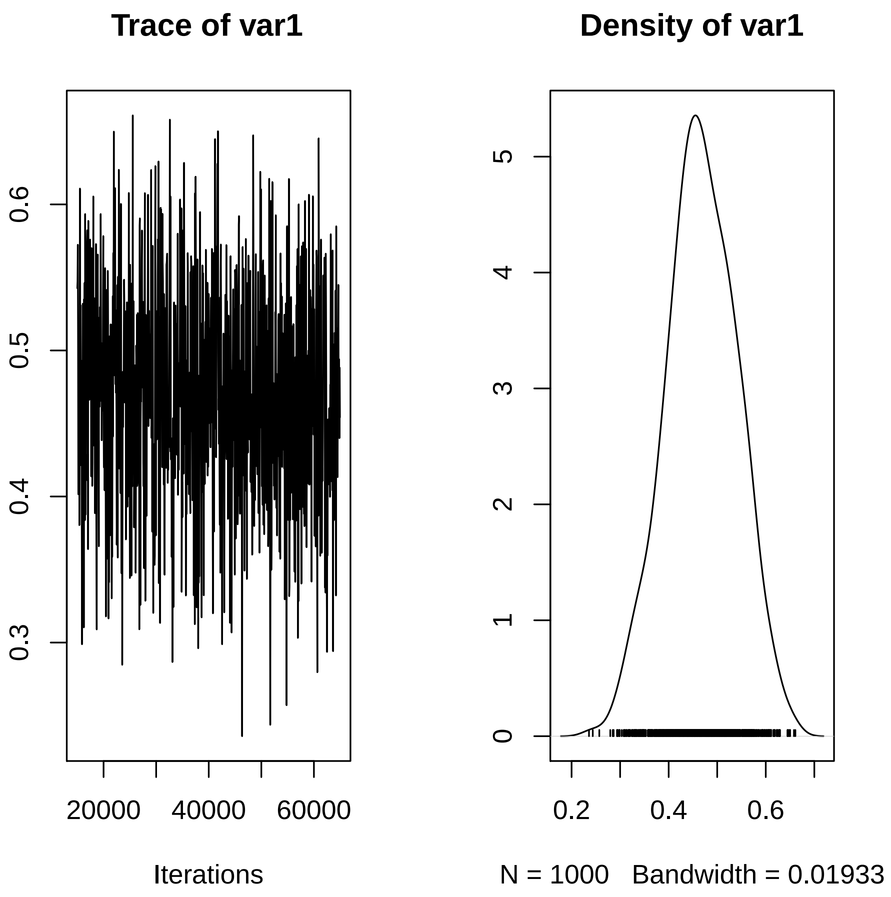

Figure 3: The posterior distributions the heritability from model 1.1.

```
    var1
2.160593

> HPDinterval(model1.2$Sol[, "SEX2"], 0.95)

        lower    upper
var1 1.893699 2.519132
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
2.826757 2.827430
```

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"] + model1.2$VCV[, "units"])
> posterior.mode(posterior.heritability1.2)

     var1
0.4910329

> HPDinterval(posterior.heritability1.2, 0.95)

         lower     upper
var1 0.3802627 0.6633944
attr(,"Probability")
[1] 0.95
```

Here h^2 has increased slightly from 0.482 to 0.491 (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.

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,
+     n = 0.002), G2 = list(V = 1,
+     n = 0.002)), R = list(V = 1, n = 0.002))
> model1.3 <- MCMCglmm(BWT ~ SEX, random = ~
+     animal + BYEAR, pedigree = Ped,
+     data = Data, nitt = 65000, thin = 50,
+     burnin = 15000, prior = prior1.3,
+     verbose = FALSE)
> posterior.mode(model1.3$VCV)
```

```
   animal     BYEAR     units
2.6890654 0.7530438 2.0917447
```

Here the variance in birth weight explained by birth year is 0.753. Note that although V_A has changed somewhat, most of what is now partitioned as a birth year effect was previously partitioned as VR. 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:

```
> p.var <- var(Data$BWT, na.rm = TRUE)
> prior1.4 <- list(G = list(G1 = list(V = 1,
+     n = 0.002), G2 = list(V = 1,
+     n = 0.002), G3 = list(V = 1, n
+     = 0.002)), R = list(V = 1,
+     n = 0.002))
> model1.4 <- MCMCglmm(BWT ~ SEX, random =
+     ~animal + BYEAR + MOTHER,
+     pedigree = Ped, data = Data, nitt = 65000,
+     thin = 50, burnin = 15000,
+     prior = prior1.4, verbose = FALSE)
> posterior.mode(model1.4$VCV)
```

```
   animal     BYEAR    MOTHER     units
2.2573296 0.7717659 1.1122087 1.6098344
```

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

```
     var1
0.4910329
```

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

```
     var1
0.4578524
```

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

```
     var1
0.3766595
```

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 variance 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. We can test the statistical significance of variance components using the deviance information criterion (DIC). It should be noted that the properties of DIC are not as well understood. The implementation of DIC in `MCMCglmm` is further described in the reference manual. DIC values are calculated by `MCMCglmm` by default. Briefly, DIC balances model fit and parameter number simultaneously, and small values of DIC are prefered. We can test the statistical significance of the maternal effect by comparing models 1.4 and 1.3, i.e., models with and without the mother term:

```
> model1.3$DIC
```

```
[1] 3546.683
```

```
> model1.4$DIC
```

```
[1] 3313.364
```

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.

Tutorial 2 (MCMCglmm) – A bivariate animal model

This tutorial will demonstrate how to run a multivariate animal model using the R package MCMCglmm and example data files provided.

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 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 genetic correlation r_G. However, covariance can also arise through other random effects (e.g. maternal covariance) and these sources can be explicitly modelled in a bivariate analysis.

Data

Pedigree and phenotypic data files are the same as those used in tutorial 1 (i.e, `gryphonped .txt` and `gryphon.txt` respectively). See tutorial 1, section 2 for information on loading these data into `R`. Note that if you are continuing from tutorial number 1 that these data will still be loaded and no further processing of the raw data is required at this point.

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:

```
> prior2.1 <- list(G = list(G1 = list(
+     V = diag(2), n = 1.002)),
+     R = list(V = diag(2), n = 1.002))
> model2.1 <- MCMCglmm(cbind(BWT, TARSUS) ~
+     trait - 1, random = ~us(trait):animal,
+     rcov = ~us(trait):units, family =
+     c("gaussian", "gaussian"),
+     pedigree = Ped, data = Data, prior =
+     prior2.1, verbose = FALSE)
```

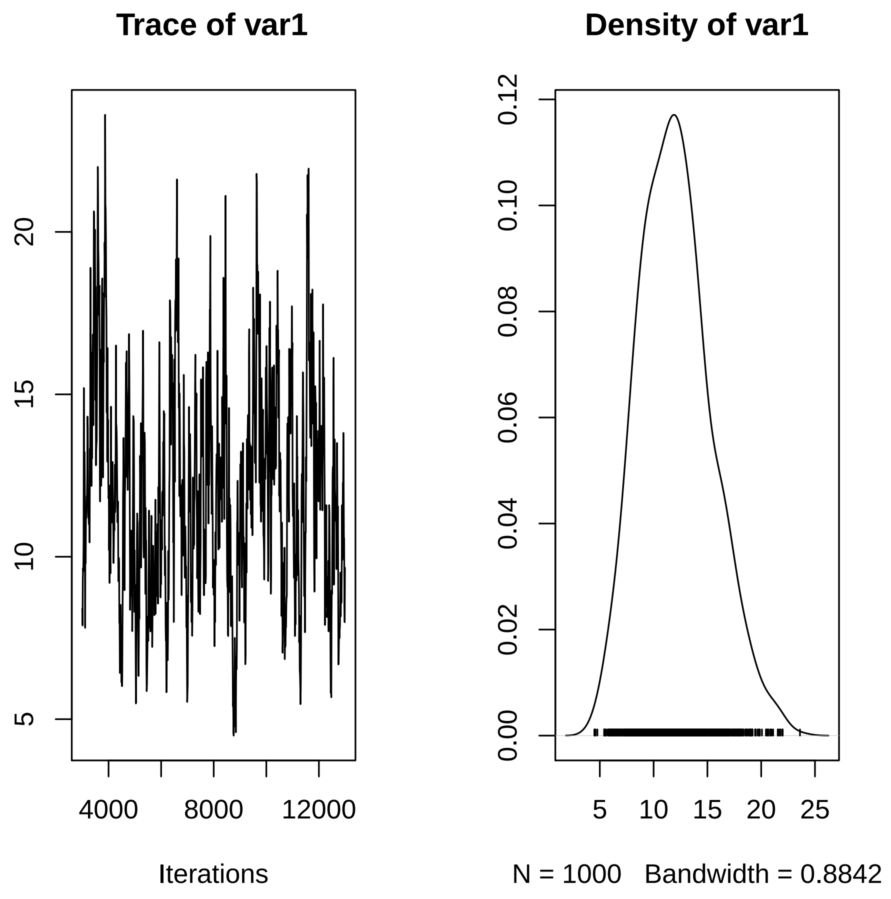

Figure 4: The posterior distribution of the additive genetic effect for tarsus length in a MCMCglmm run with default values.

```
> plot(model2.1$VCV[, "TARSUS:TARSUS.animal"])
```

We have constructed the prior similarly to the those in the univariate models in tutorial 1, only we are specifying a matrix based on the the phenotypic variances (and arbitrarily ignoring covariance). In order to provide proper priors, we have set the degree of belief parameter to the order of the matrices describing the priors, i.e., 2. We have used the `R` command `cbind`, "column-bind", to specify the two response variables, body weight and tarsus length. The nature of genetic and residual (co)variance (between and) of the traits is specified as unstructured matrices, the most general structure available for this type of data. Finally, we have specified that we wish to treat both traits as gaussian responses.

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 five samples (50 iterations given this MCMCglmm run with default values). This lag of 0.4965 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:

```
model2.1<-MCMCglmm(cbind(BWT,TARSUS)~trait-1,
              random=~us(trait):animal,
              rcov=~us(trait):units,
              family=c("gaussian","gaussian"),
              pedigree=Ped,data=Data,
              nitt=130000,thin=100,burnin=30000,
              prior=prior2.1,verbose=FALSE)

# the autocorrelation of the genetic variance of TARSUS at Lag 5
autocorr(model2.2$VCV)[,,"TARSUS:TARSUS.animal"][3,4]
```

However, this run would 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 `./model1point1LongRun.Rdat` is available at the specified location:

```
> load(file = "./model2point1LongRun.Rdat")
> autocorr(model2.1$VCV)[, , "TARSUS:TARSUS.animal"][3, 4]

[1] 0.01200772
```

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)

      BWT:BWT.animal    TARSUS:BWT.animal    BWT:TARSUS.animal
            3.078625             2.627429             2.627429
TARSUS:TARSUS.animal        BWT:BWT.units     TARSUS:BWT.units
           11.996661             3.947790             3.115639
    BWT:TARSUS.units  TARSUS:TARSUS.units
            3.115639            18.017630

> heritability.BWT2.1 <- model2.1$VCV[,
+     "BWT:BWT.animal"]/(model2.1$VCV[,
+     "BWT:BWT.animal"] + model2.1$VCV[,
+     "BWT:BWT.animal"])
> posterior.mode(heritability.BWT2.1)

     var1
0.4999336
```

```
> heritability.TARSUS2.1 <- model2.1$VCV[,
+     "TARSUS:TARSUS.animal"]/(model2.1$VCV[,
+     "TARSUS:TARSUS.animal"] + model2.1$VCV[,
+     "TARSUS:TARSUS.units"])
> posterior.mode(heritability.TARSUS2.1)
```

```
    var1
0.387315
```

```
> genetic.correlation2.1 <- model2.1$VCV[,
+     "BWT:TARSUS.animal"]/sqrt(model2.1$VCV[,
+     "BWT:BWT.animal"] * model2.1$VCV[,
+     "TARSUS:TARSUS.animal"])
> posterior.mode(genetic.correlation2.1)
```

```
     var1
0.3379037
```

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:

```
prior2.2<-list(G=list(G1=list(V=diag(2),n=1.002),
                 G2=list(V=diag(2),n=1.002),
                 G3=list(VV=diag(2),n=1.002)),
                 R=list(V=diag(2),n=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"),
          pedigree=Ped,data=Data,
          nitt=130000,thin=100,burnin=30000,
          prior=prior2.2,verbose=FALSE)
```

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

```
> load(file = "./model2point2LongRun.Rdat")
> autocorr(model2.2$VCV)[, , "TARSUS:TARSUS.animal"][3, 4]
```

```
[1] 0.02443699
```

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

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

```
       BWT:BWT.animal    TARSUS:BWT.animal    BWT:TARSUS.animal
            1.8938967            2.9376207            2.9376207
TARSUS:TARSUS.animal        BWT:BWT.BYEAR     TARSUS:BWT.BYEAR
            9.8658027            1.0051499            0.2848888
    BWT:TARSUS.BYEAR  TARSUS:TARSUS.BYEAR       BWT:BWT.MOTHER
            0.2848888            3.5068545            1.1759621
   TARSUS:BWT.MOTHER    BWT:TARSUS.MOTHER TARSUS:TARSUS.MOTHER
           -1.6842097           -1.6842097            4.3827364
        BWT:BWT.units    TARSUS:BWT.units     BWT:TARSUS.units
            1.9037183            3.9196186            3.9196186
 TARSUS:TARSUS.units
           12.6003031
```

```
> genetic.correlation2.2 <- model2.2$VCV[,
+      "BWT:TARSUS.animal"]/sqrt(model2.2$VCV[,
+     "BWT:BWT.animal"] * model2.2$VCV[,
+     "TARSUS:TARSUS.animal"])
> maternal.correlation2.2 <- model2.2$VCV[,
+     "BWT:TARSUS.MOTHER"]/sqrt(model2.2$VCV[,
+     "BWT:BWT.MOTHER"] * model2.2$VCV[,
+     "TARSUS:TARSUS.MOTHER"])
> posterior.mode(genetic.correlation2.2)
```

```
     var1
0.8011656
```

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

```
      var1
-0.7334039
```

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 0.5286517 0.9414846
attr(,"Probability")
[1] 0.95
```

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

```
          lower      upper
var1 -0.9297604 -0.3658335
attr(,"Probability")
[1] 0.95
```

Neither or these posterior distributions overlapps zero, so we can consider them both statistically supported. Alternatively, we could constrain one or both covariances to zero using `idh()` in place of `us()` in the `MCMCglmm` model specifications, and proceed with significance testing using DIC.

Tutorial 3 (MCMCglmm) – A repeated measures animal model

This tutorial will demonstrate how to run a univariate animal model for a trait with repeated observations using the software `MCMCglmm` and example data files provided.

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 Jan 1) for individual females of varying 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.

Data

We use for these examples the same pedigree as was used for tutorials 1 and 2, and we assume that it is still loaded. If this is not the case, revisit tutorial 1, section 2 for instructions for loading the pedigree. We are however going to use different phenotypic data collected from the same population of gryphons, and we need to load these data. We can load the data from the file `gryphonRM.txt` into an `R` data frame using the following code

```
> DataRM <- as.data.frame(read.table(file = "./gryphonRM.txt",
+     header = TRUE))
> names(DataRM)[1] <- "animal"
> DataRM$animal <- as.factor(DataRM$animal)
> DataRM$BYEAR <- as.factor(DataRM$BYEAR)
> DataRM$AGE <- as.factor(DataRM$AGE)
> DataRM$YEAR <- as.factor(DataRM$YEAR)
> DataRM$LAYDATE <- as.numeric(DataRM$LAYDATE)
> head(DataRM)
```

```
  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
```

```
> DataRM$ID <- DataRM$animal
> head(DataRM)

  animal BYEAR AGE YEAR LAYDATE ID
1      1   990   2  992      19  1
2      1   990   3  993      23  1
3      1   990   4  994      24  1
4      1   990   5  995      23  1
5      1   990   6  996      29  1
6      2   990   2  992      21  2
```

Note that we have created a new factor called ID that is the same as the 'animal' factor. MCMCglmm will recognize the term animal and use it to relate individuals to their records in a pedigree. The factor ID allows us to disassociate individual records from the pedigree, which is an important part of analyzing repeated measures data, as we will see.

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(DataRM$LAYDATE, na.rm = TRUE)
> prior3.1 <- list(G = list(G1 = list(V = 1,
+     nu = 0.002)), R = list(V = 1,
+     nu = 0.002))
> model3.1 <- MCMCglmm(LAYDATE ~ 1, random = ~ID,
+     data = DataRM, prior = prior3.1, verbose = FALSE)
> posterior.mode(model3.1$VCV)

      ID    units
11.80216 21.17339
```

Note the use of the new `ID` factor that we created in the previous section.

Between-individual variance is given by the `ID` component, while the residual component (Variance) therefore represents within-individual variance. Here then the repeatability of the trait can be determined by as 0.371 (i.e., 11.8022/(11.8022+21.1734). Given that we set up the simulation such that mean lay date changes with age (initially increasing 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 = ~ID, data = DataRM,
+     prior = prior3.1, verbose = FALSE)
> plot(model3.2$Sol)
> plot(model3.2$VCV)
> posterior.mode(model3.2$VCV)

      ID    units
12.26418 16.26076
```

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 0.435 (i.e., 12.2642/(12.2642+16.2608). 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.

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

```
> model3.3 <- MCMCglmm(LAYDATE ~ 1 + AGE,
+     random = ~animal, pedigree = Ped,
+     data = DataRM, prior = prior3.1,
+     verbose = FALSE)
> posterior.mode(model3.3$VCV)

  animal    units
14.37259 16.88468
```

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). An more appropriate estimate of V_A is given by the model:

```
> p.var <- var(DataRM$LAYDATE, na.rm = TRUE)
> prior3.4 <- list(G = list(G1 = list(V =
+     1, n = 0.002), G2 = list(V = 1,
+     n = 0.002)), R = list(V = 1, n = 0.002))
> model3.4 <- MCMCglmm(LAYDATE ~ 1 +
+     AGE, random = ~animal + ID,
+     pedigree = Ped, data = DataRM, prior =
+     prior3.4, verbose = FALSE)
> posterior.mode(model3.4$VCV)

   animal         ID     units
 4.385704  7.652516 16.202916
```

The estimate of V_A is now much lower (reduced from 14.3726 to 4.3857) since the additive 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[, "ID"] +
+     model3.4$VCV[, "units"]
> model3.4.IDplusVA <- model3.4$VCV[, "animal"] + model3.4$VCV[,
+     "ID"]
> posterior.mode(model3.4.IDplusVA/model3.4.VP)

     var1
0.4248554

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

     var1
0.1524774
```

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(DataRM$LAYDATE, na.rm = TRUE)
> prior3.5 <- list(G = list(G1 = list(V =
+     1, n = 0.002), G2 = list(V = 1,
+     n = 0.002), G3 = list(V = 1, n =
+     0.002), G4 = list(V = 1,
+     n = 0.002)), R = list(V = 1, n =
+     0.002))
> model3.5 <- MCMCglmm(LAYDATE ~ 1 + AGE,
+     random = ~animal + ID +
+     YEAR + BYEAR, pedigree = Ped, data =
+     DataRM, prior = prior3.5,
+     verbose = FALSE)
> posterior.mode(model3.5$VCV)

     animal          ID        YEAR       BYEAR       units
4.439354078 8.286966613 7.719833939 0.003340065 7.935489610
```

This model will return additional variance components corresponding to year of measurement 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` effects 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 $\mathrm{V_R}$ 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.