Bayesian generalised mixed models with MCMCglmm

Ferran Sayol

20th of December 2018

This is a short guide of running mixed models under a bayesian framework using the MCMCglmm R package. The course is structured in three parts: 1) See the basics of MCMCglmm models (run and check the output); 2) Adding random effects and modify the priors; 3) Correct for phylogenetic effects.

Package Installation.

First we need to install the MCMCglmm package. We also install an additional packages to use phylogenetic trees (phytools).

```
if(!require(MCMCglmm)) install.packages("MCMCglmm")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(phytools)) install.packages("phytools")
```

Next we load up the packages we just installed from the library and we are good to go.

```
library(MCMCglmm)
library(phytools)
library(ggplot2)
```

PART 1: Introduction to MCMCglmm models

First we will load some data as an example. We will use data on morphological measurements and ecology of Pigeons&Doves (Columbidae).

```
getwd() #Check the Working directory
```

```
## [1] "/Users/xsayfe/ownCloud/Research_Resources/MyCourses/MCMCglmm_Erlangen18"
```

```
#setwd() #Change the working directory.
dove <- read.table("data/DB_ColumbidaeData.txt",h=T)
```

This data file is based on a subset of the data used in an analysis on the relation between foraging behaviour and the evolution of morphological adaptations (Lapiedra et al. 2013) Link to paper. Note that from the original data we have generated N repeated measures for each species, adding random noise on each data point.

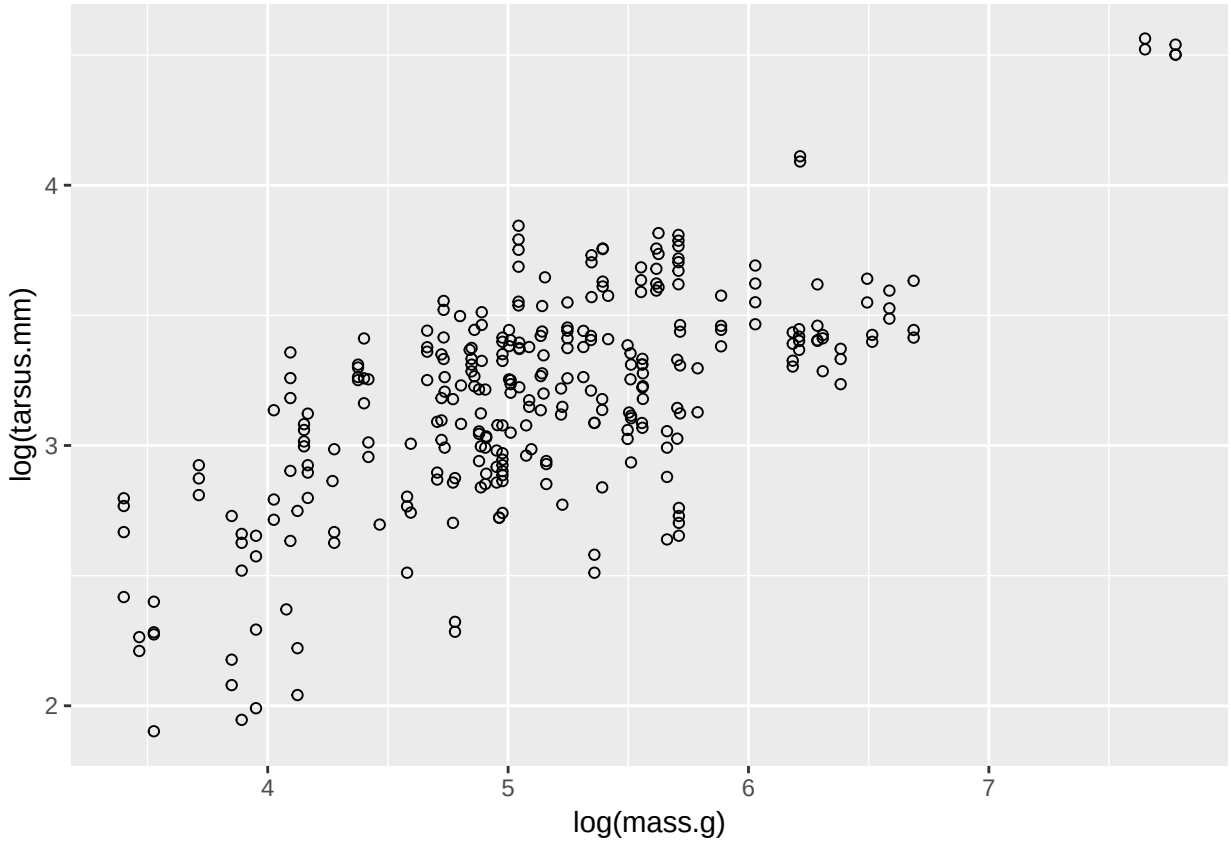
```
head(dove)
```

```
##                    species tarsus.mm tail.mm wing.mm mass.g    foraging
## 110  Ptilinopus_magnificus     35.72  171.32  228.02  360.0    arboreal
## 1403         Treron_waalia     21.50  106.00  173.90  259.5    arboreal
## 67       Geotrygon_violacea     33.02   63.92  147.22  121.5 terrestrial
## 613     Geotrygon_lawrencii     37.70   70.20  137.50  220.0 terrestrial
## 172      Columbina_cruziana     15.32   26.42   84.02   47.0 terrestrial
## 623       Geotrygon_montana     27.40   58.20  141.50  127.5 terrestrial
##          location measureID
## 110  Australasia  measureA
## 1403      Africa  measureB
## 67       America  measureA
```

```
## 613      America  measureB
## 172      America  measureA
## 623      America  measureB
```

Let's pretend our goal is to study the relation between morphology and ecology. For instance, if the tarsus length is related to foraging behaviour. But we will first start by exploring the relation between tarsus length and body size.

```
ggplot(dove, aes(x=log(mass.g), y=log(tarsus.mm))) +
  geom_point(shape=21)
```



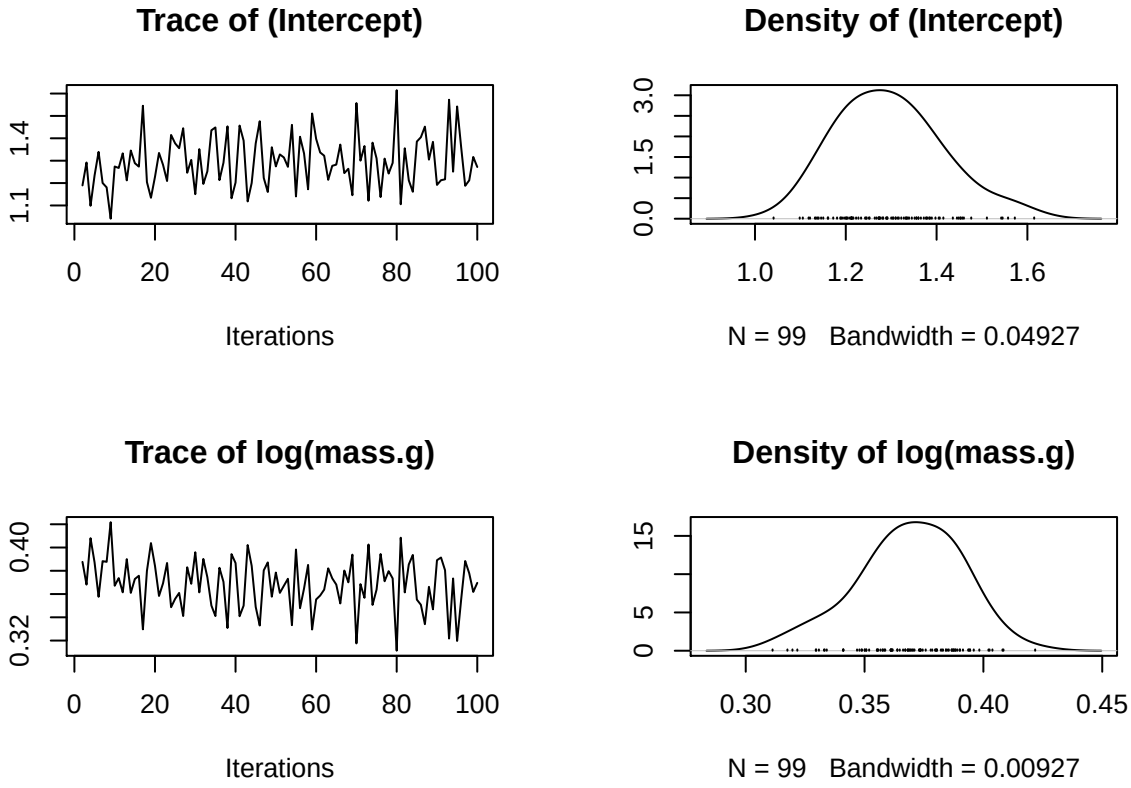
Now let's run a simple model with tarsus.mm as response of body size (mass.g).

When running an MCMCglmm, we need to specify some parameters of the mcmc chain: How many iterations we want to run the chain for (nitt), the burnin we want to discard at the start of the chain (burnin) and also how often we want to sample and store from the chain (`thin`). We discard a burnin as we don't want the starting point of the chain to over-influence our final estimates.

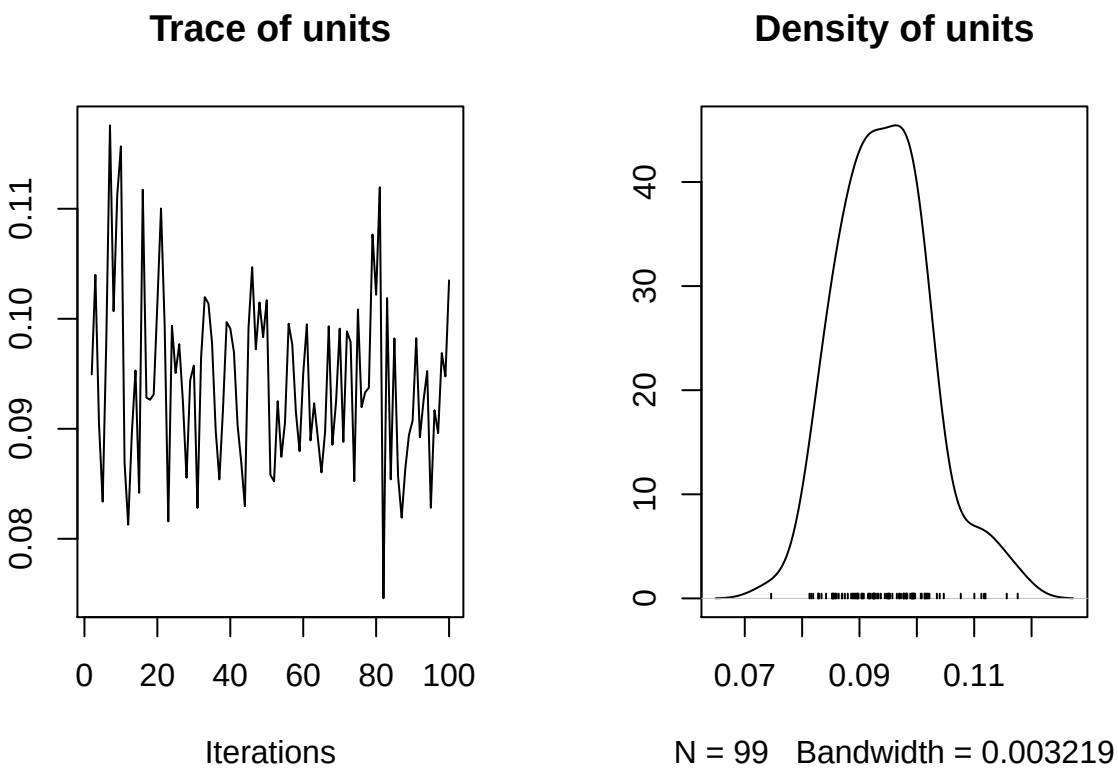
```
prior1 <- list(R=list(V = 1,nu = 0.002)) #We will see this later

mod1.1 <- MCMCglmm(log(tarsus.mm) ~ log(mass.g),
                   data = dove, prior = prior1,verbose=F,
                   nitt = 100, thin=1, burnin = 1)
```

Before we even look at our model putput we can check if the model ran appropriately. We can do this by visually inspecting the chains. We can extract the full chains using `model$Sol` for the fixed effects and `model$VCV` for the variance terms. So `Sol[,1]` will give you the first fixed term, in this case the intercept, and `VCV[,1]` will give you the first random term, which is just the residual term here. As our model is an mcmc object when we use the plot function we get a trace plot.

```
#plot the fist fixed term, the intercpet.
plot(mod1.1$Sol)
```




```
#plot the fist variance term, the residual error term.
plot(mod1.1$VCV)
```



On the right hand side of the plots is the posterior distributions for each of the terms. On the left side of these plots are the traces of the mcmc chain for each estimate. What we want to see in these trace plots has an aparent random pattern. That is a trace with no obvious trend that is bouncing around some stable point.

Another thing we also want to check is the level of auto-correlation in the chain traces. We can do this using autocorr.diag() which gives the level of correlation along the chain between some lag sizes.

Let's see some diagnosis (Autocorrelation)

```
autocorr.diag(mod1.1$Sol) #Solutions (coeficients)
```

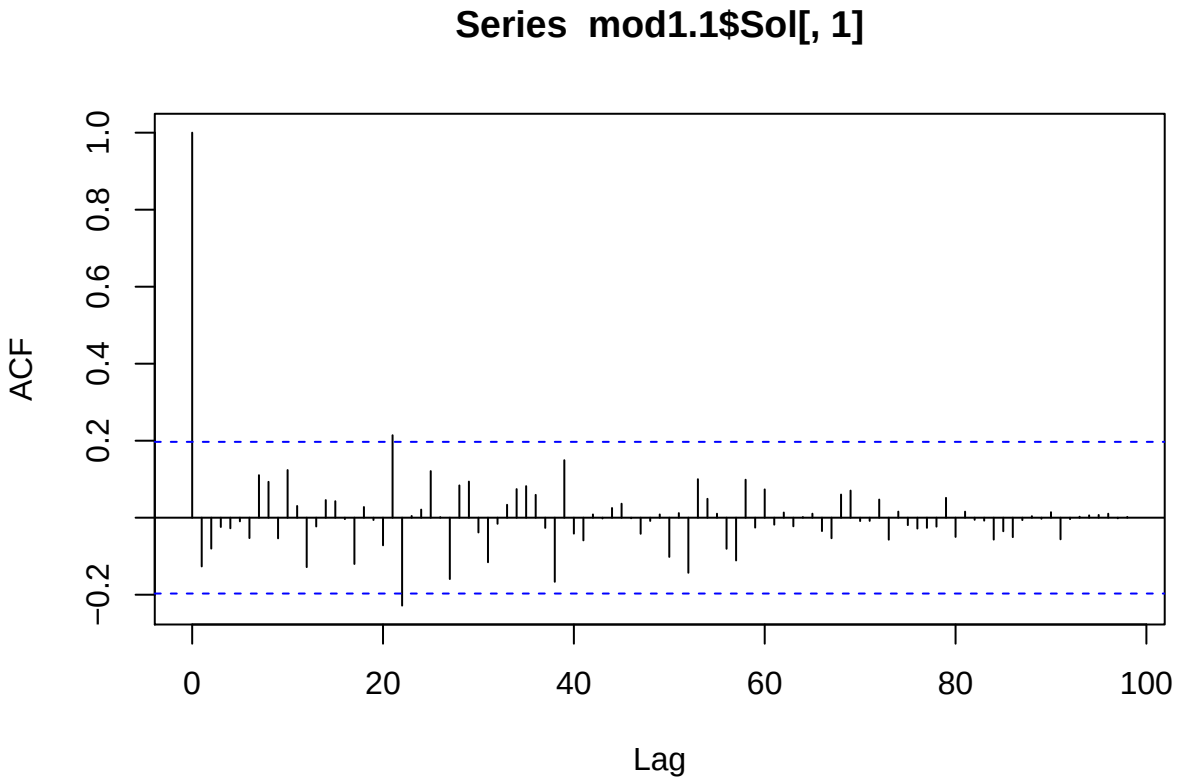
```
##         (Intercept)  log(mass.g)
## Lag 0   1.000000000  1.000000000
## Lag 1  -0.126478809 -0.129550217
## Lag 5  -0.009363355  0.001990293
## Lag 10  0.123724412  0.108658739
## Lag 50 -0.101782568 -0.097937275
```

```
autocorr.diag(mod1.1$VCV) #Variance
```

```
##              units
## Lag 0   1.000000000
## Lag 1   0.110732654
## Lag 5  -0.143570133
## Lag 10 -0.006355195
## Lag 50 -0.018886711
```

Another way is to look at autocorrelation plots for each of the traces. For example, let's check the auto-correlation in the intercept chain using the `acf` function

```
#acf plot for the first fixed estimate in our model (the intercept)
acf(mod1.1$Sol[,1],lag.max =100)
```



Ideally, we have to make sure that the autocorrelation is less than 0.1. The thinning is used to help reduce autocorrelation in our sample, how much you use often depends on how much autocorrelation you find and we can reduce autocorrelation by increasing the thining interval. As a result, we might have to increase the total number of iterations as well to have a sample of at least 1000. We can also set a Burn-in (normally 5-10% of samples) to get rid of the first samples that have not converged yet.

- EXERCISE 1: *Increase the thining interval and the number of iterations to make sure there is no autocorrelation and to have a sample of >1000.*

```
prior1 <- list(R=list(V = 1,nu = 0.002)) #We will see this later

mod1.1 <- MCMCglmm(log(tarsus.mm) ~ log(mass.g),
                   data = dove, prior = prior1,verbose=T,
                   nitt = 101000, thin=100, burnin = 1000)
```

```
##
##                       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
##
##                         MCMC iteration = 14000
##
##                         MCMC iteration = 15000
##
##                         MCMC iteration = 16000
##
##                         MCMC iteration = 17000
##
##                         MCMC iteration = 18000
##
##                         MCMC iteration = 19000
##
##                         MCMC iteration = 20000
##
##                         MCMC iteration = 21000
##
##                         MCMC iteration = 22000
##
##                         MCMC iteration = 23000
##
##                         MCMC iteration = 24000
##
##                         MCMC iteration = 25000
##
##                         MCMC iteration = 26000
##
##                         MCMC iteration = 27000
##
##                         MCMC iteration = 28000
##
##                         MCMC iteration = 29000
##
##                         MCMC iteration = 30000
##
##                         MCMC iteration = 31000
##
##                         MCMC iteration = 32000
##
##                         MCMC iteration = 33000
```

```
##
##                         MCMC iteration = 34000
##
##                         MCMC iteration = 35000
##
##                         MCMC iteration = 36000
##
##                         MCMC iteration = 37000
##
##                         MCMC iteration = 38000
##
##                         MCMC iteration = 39000
##
##                         MCMC iteration = 40000
##
##                         MCMC iteration = 41000
##
##                         MCMC iteration = 42000
##
##                         MCMC iteration = 43000
##
##                         MCMC iteration = 44000
##
##                         MCMC iteration = 45000
##
##                         MCMC iteration = 46000
##
##                         MCMC iteration = 47000
##
##                         MCMC iteration = 48000
##
##                         MCMC iteration = 49000
##
##                         MCMC iteration = 50000
##
##                         MCMC iteration = 51000
##
##                         MCMC iteration = 52000
##
##                         MCMC iteration = 53000
##
##                         MCMC iteration = 54000
##
##                         MCMC iteration = 55000
##
##                         MCMC iteration = 56000
##
##                         MCMC iteration = 57000
##
##                         MCMC iteration = 58000
##
##                         MCMC iteration = 59000
##
##                         MCMC iteration = 60000
```

```
##
##                         MCMC iteration = 61000
##
##                         MCMC iteration = 62000
##
##                         MCMC iteration = 63000
##
##                         MCMC iteration = 64000
##
##                         MCMC iteration = 65000
##
##                         MCMC iteration = 66000
##
##                         MCMC iteration = 67000
##
##                         MCMC iteration = 68000
##
##                         MCMC iteration = 69000
##
##                         MCMC iteration = 70000
##
##                         MCMC iteration = 71000
##
##                         MCMC iteration = 72000
##
##                         MCMC iteration = 73000
##
##                         MCMC iteration = 74000
##
##                         MCMC iteration = 75000
##
##                         MCMC iteration = 76000
##
##                         MCMC iteration = 77000
##
##                         MCMC iteration = 78000
##
##                         MCMC iteration = 79000
##
##                         MCMC iteration = 80000
##
##                         MCMC iteration = 81000
##
##                         MCMC iteration = 82000
##
##                         MCMC iteration = 83000
##
##                         MCMC iteration = 84000
##
##                         MCMC iteration = 85000
##
##                         MCMC iteration = 86000
##
##                         MCMC iteration = 87000
```

```
##
##                         MCMC iteration = 88000
##
##                         MCMC iteration = 89000
##
##                         MCMC iteration = 90000
##
##                         MCMC iteration = 91000
##
##                         MCMC iteration = 92000
##
##                         MCMC iteration = 93000
##
##                         MCMC iteration = 94000
##
##                         MCMC iteration = 95000
##
##                         MCMC iteration = 96000
##
##                         MCMC iteration = 97000
##
##                         MCMC iteration = 98000
##
##                         MCMC iteration = 99000
##
##                         MCMC iteration = 100000
##
##                         MCMC iteration = 101000
```

Now, let's explore the output of model.

```
summary(mod1.1)
```

```
##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 142.6552
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.09419   0.0799   0.1104    835.9
##
##  Location effects: log(tarsus.mm) ~ log(mass.g)
##
##             post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)    1.2933   1.0503   1.4939     1000 <0.001 ***
## log(mass.g)    0.3692   0.3275   0.4132     1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see the estimates for the fixed factor. Each parameter has a measure of the effect size under post.mean

and a lower and higher 95% credible interval (CI).

Another way to directly look at the posterior means and confidence intervals for the factors is with the following commands.

Posterior mean of fixed factors:

```
posterior.mode(mod1.1$Sol)
```

```
## (Intercept) log(mass.g)
##   1.2684827   0.3718734
```

Posterior mean of fixed factors:

```
HPDinterval(mod1.1$Sol)
```

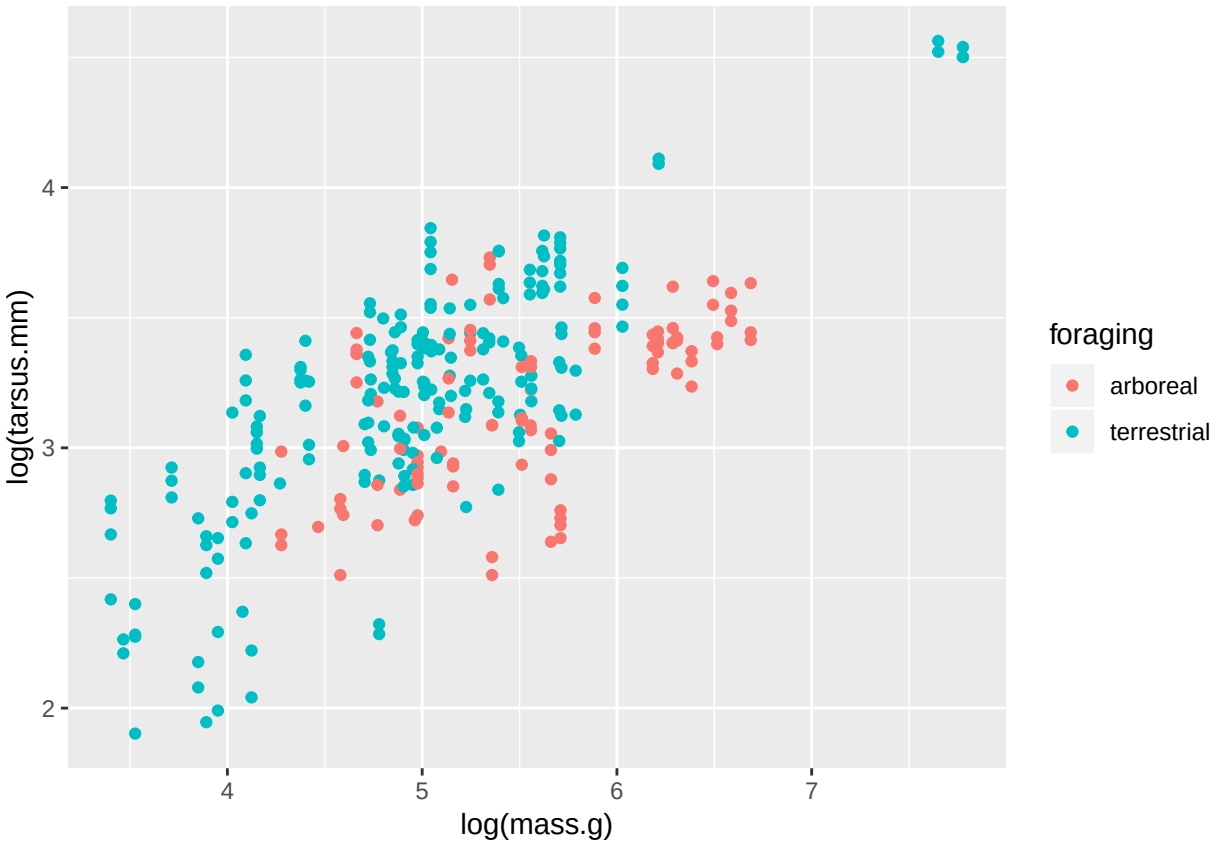
```
##                 lower     upper
## (Intercept) 1.0502943 1.4938624
## log(mass.g) 0.3275394 0.4132225
## attr(,"Probability")
## [1] 0.95
```

We also have the effective sample size (*eff.samp*) and the *pMCMC* which calculated as two times the probability that the estimate is either $>$ or $<$ 0, using which ever one is smaller. However, since our data has been mean centred and expressed in units of standard deviation we can simply look at what proportion of our posterior is on either side of zero. This mean centering and expression in of our data units of standard deviation hence allows us to use a cut off point like a p-value but without boiling down the whole distribution to one value.

We also have the *DIC* which is a Bayesian version of *AIC*. Like *AIC* it is a measure of the trade-off between the "fit" of the model and the number of parameters, with a lower number better.

Comming back to theinitial question, we want to see if foraging behavioir can explain tarsus length while accounting for body size. Let's do a plot first:

```
ggplot(dove, aes(x=log(mass.g), y=log(tarsus.mm), color=foraging)) +
  geom_point(shape=19)
```



It looks like foraging behaviour can explain relative differences in tarsus length. But we need to test this:

- EXERCISE 2: *Run a new model (mod1.2) including also the foraging ecology together with body size as predictor and compare the DIC with mod1.1 / Which of the models is better?*

```
prior1 <- list(R=list(V = 1,nu = 0.002)) #We will see this later

names(dove)
```

```
## [1] "species"   "tarsus.mm" "tail.mm"   "wing.mm"   "mass.g"    "foraging"
## [7] "location"  "measureID"
```

```
mod1.2 <- MCMCglmm(log(tarsus.mm) ~ log(mass.g)+foraging,
                   data = dove, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)
mod1.3 <- MCMCglmm(log(tarsus.mm) ~ log(mass.g)+foraging-1,
                   data = dove, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)

summary(mod1.2)
```

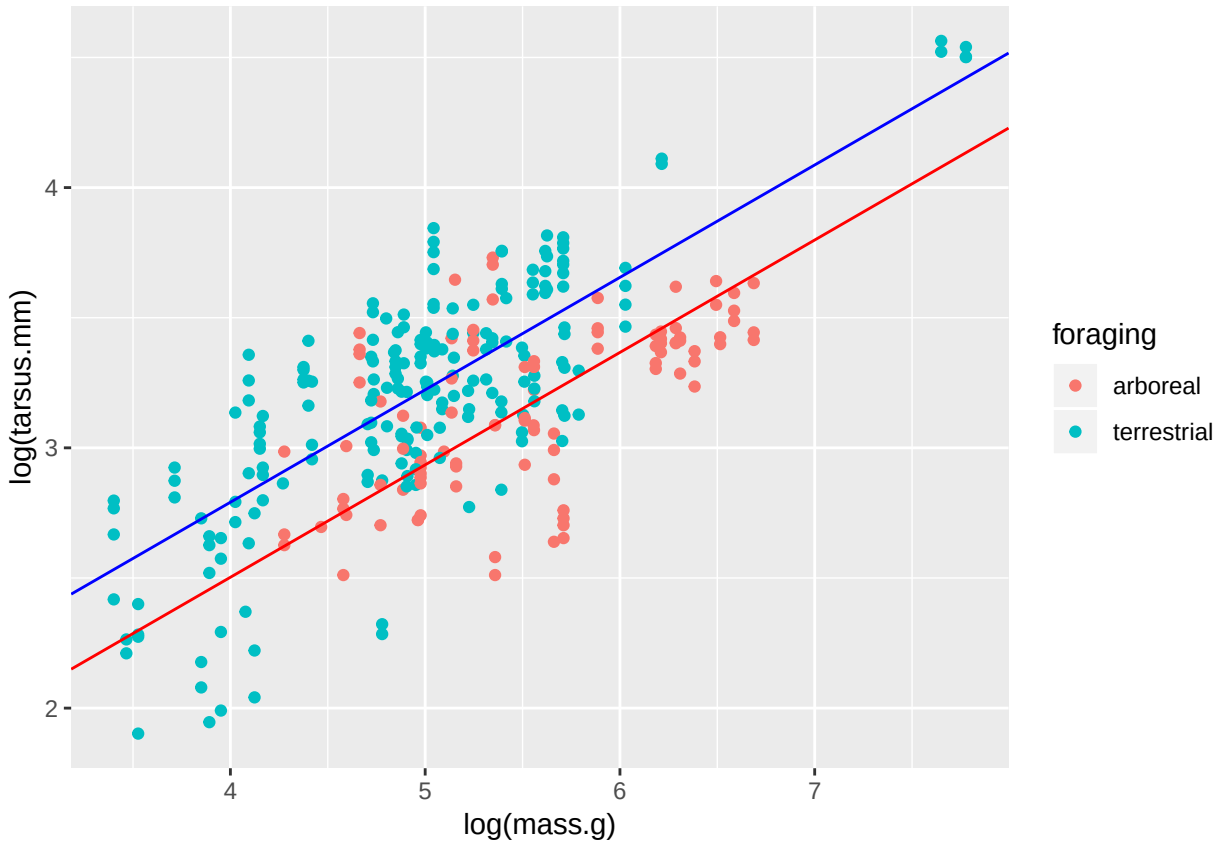
```
##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 89.69631
##
```

```
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.07834  0.06568  0.09083     1000
##
##  Location effects: log(tarsus.mm) ~ log(mass.g) + foraging
##
##                     post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)            0.8145   0.5936   1.0681    817.0 <0.001 ***
## log(mass.g)            0.4255   0.3777   0.4623    816.6 <0.001 ***
## foragingterrestrial    0.2839   0.2163   0.3575   1437.8 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod1.3)
```

```
##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 89.70875
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.07848  0.06684  0.09166     1000
##
##  Location effects: log(tarsus.mm) ~ log(mass.g) + foraging - 1
##
##                     post.mean l-95% CI u-95% CI eff.samp  pMCMC
## log(mass.g)            0.4262   0.3847   0.4713     1000 <0.001 ***
## foragingarboreal       0.8116   0.5616   1.0601     1000 <0.001 ***
## foragingterrestrial    1.0925   0.8919   1.3152     1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ggplot(dove, aes(x=log(mass.g), y=log(tarsus.mm), color=foraging)) +
  geom_point(shape=19) +
  geom_abline(intercept = posterior.mode(mod1.3$Sol)[2],slope = posterior.mode(mod1.3$Sol)[1],color="re
  geom_abline(intercept = posterior.mode(mod1.3$Sol)[3],slope = posterior.mode(mod1.3$Sol)[1],color="blu
```



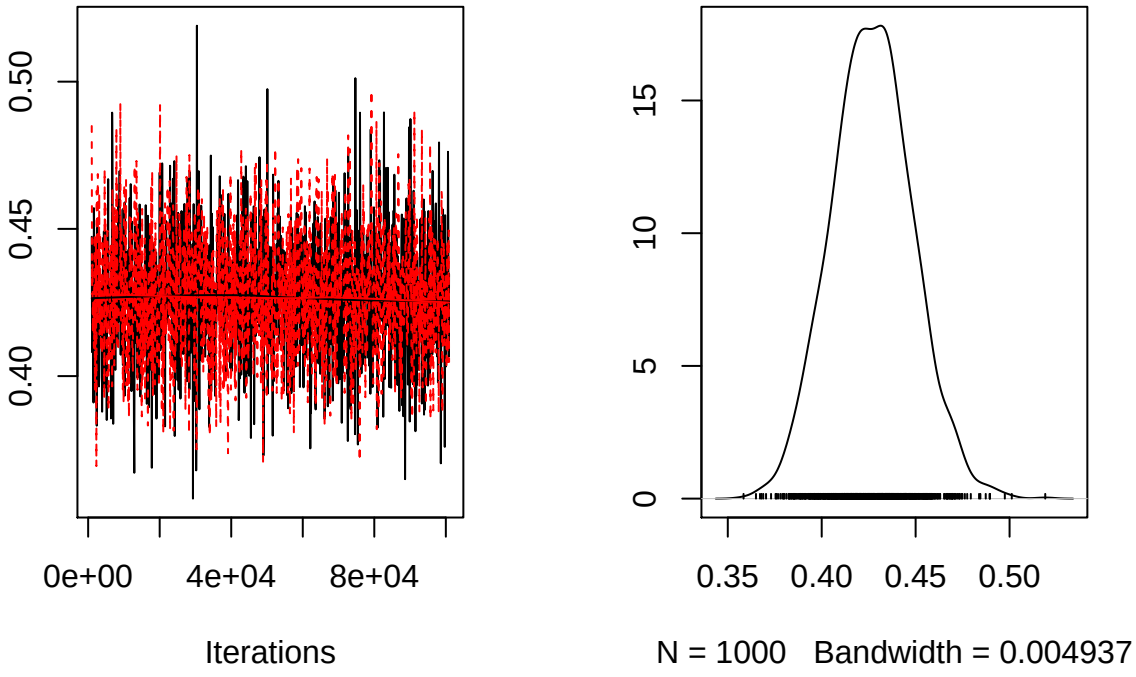

```
summary(mod1.1)
```

```
##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 142.6552
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.09419   0.0799   0.1104    835.9
##
##  Location effects: log(tarsus.mm) ~ log(mass.g)
##
##             post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)    1.2933   1.0503   1.4939     1000 <0.001 ***
## log(mass.g)    0.3692   0.3275   0.4132     1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model convergence

One last thing to check is that our MCMC chain has properly converged and that our estimate is not the result of some type of transitional behaviour. That is have our chains "found" the optimum or do we need to let them run longer before they settle around some estimate. To check this we will run a second model and see if it converges on the same estimates as our first model.

```
mod1.3 <- MCMCglmm(log(tarsus.mm) ~ log(mass.g)+foraging,
                   data = dove, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)
mod1.3b <- MCMCglmm(log(tarsus.mm) ~ log(mass.g)+foraging,
                   data = dove, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)
#They have reached the same solution?
plot(mcmc.list(mod1.3$Sol[,2], mod1.3b$Sol[,2]))
```




```
summary(mod1.3b)
```

```
##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 89.70255
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.07817  0.06632  0.09195     1000
##
##  Location effects: log(tarsus.mm) ~ log(mass.g) + foraging
##
##                     post.mean l-95% CI u-95% CI eff.samp  pMCMC
```

```
## (Intercept)            0.8052   0.5594   1.0368     1000 <0.001 ***
## log(mass.g)            0.4271   0.3863   0.4692     1000 <0.001 ***
## foragingterrestrial    0.2840   0.2129   0.3523     1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

PART 2: Modify priors and add random factors

Since we are using a Bayesian approach we will need to set up the priors. In most cases we want to use a non-informative prior that doesn't influence the estimated posterior distribution. We are basically saying that we don't know anything about the expected values for our parameters. That is we have no prior information.

To give priors for MCMCglmm we need to make an object that is in a list format that includes terms B (fixed effects), R (residual terms) and G (random effects).

In our model we have 3 fixed terms B (1 intercept + 2 factors) and the residual term R.

```
prior2.1 <- list(R=list(V = 1,nu = 0.002))

mod2.1 <- MCMCglmm(log(tarsus.mm) ~ foraging+log(mass.g),
                   data = dove, prior = prior2.1,verbose=F,family="gaussian",
                   nitt = 1100, thin=10, burnin = 100)
summary(mod2.1)
```

```
##
##  Iterations = 101:1091
##  Thinning interval  = 10
##  Sample size  = 100
##
##  DIC: 89.60799
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.07874  0.06518  0.09173      100
##
##  Location effects: log(tarsus.mm) ~ foraging + log(mass.g)
##
##                     post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)            0.8088   0.5971   1.0592      100 <0.01 **
## foragingterrestrial    0.2880   0.2269   0.3588      100 <0.01 **
## log(mass.g)            0.4256   0.3831   0.4672      100 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For fixed effects (B) the terms mu and V give the variance and mean of a normal distribution. Here we set mu as 0 and the variance as a large number to make these priors uninformative. Since we have three fixed terms (two intercepts and one slope) we can use the diag function to create a matrix to store a prior for each. Normally we don't need to set this as MCMCglmm will set non-informative priors automatically for fixed terms. Then, we can set the prior by only specifying the R term:

```
prior2.1 <- list(R=list(V = 1,nu = 0.002))
```

For any of the variance terms (R or G) we need to make sure that the distribution is bounded at zero as the variance term needs to be positive. In MCMCglmm the variance is described the parameters *nu* and *V*. As

again we don't have any prior information, we will use weakly informative prior values such as descripted as $V = 1$ and $nu = 0.002$.

Mixed models: Adding random factors to our MCMCglmm

Just to remember, mixed models are referred to models that contain both fixed and random factors.

Let's add a random term of measurement ("measureID" > Who took the measure) in the *dove* example. Like before, we need to set up the prior however we will let the model estimte the fixed effects this time. To add a random term we now add a G structure that acts just like the other random varience term and is defined using nu and V.

```
prior2.2 <- list(G = list(G1 = list(nu=0.002, V=1),G2 = list(nu=0.002, V=1)),
              R = list(nu=0.002, V=1))
```

We can now include the random term in the model in the section `random= ~`.

Here, we will include "measureID" as a random effect, as we have repeated measures for each of the species.

```
table(dove$measureID)
```

```
##
## measureA measureB
##      156      141
```

```
table(dove$location)
```

```
##
##      Africa     America Australasia     Eurasia
##          39         102         113          43
```

```
names(dove)
```

```
## [1] "species"   "tarsus.mm" "tail.mm"   "wing.mm"   "mass.g"    "foraging"
## [7] "location"  "measureID"
```

```
mod2.2 <- MCMCglmm(log(tarsus.mm) ~ foraging+log(mass.g),
                    random= ~measureID+location,
                    data = dove, prior = prior2.2,verbose=F,
                    nitt = 1100, thin=10, burnin = 100)
summary(mod2.2)
```

```
##
##  Iterations = 101:1091
##  Thinning interval  = 10
##  Sample size  = 100
##
##  DIC: 90.16813
##
##  G-structure:  ~measureID
##
##           post.mean l-95% CI u-95% CI eff.samp
## measureID    0.1357    3e-04   0.7396      100
##
##                ~location
##
##          post.mean  l-95% CI u-95% CI eff.samp
## location  0.008612 0.0002984  0.02866      100
```

```
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.0786  0.06727  0.08992      100
##
##  Location effects: log(tarsus.mm) ~ foraging + log(mass.g)
##
##                      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)             0.7905   0.3708   1.1559      100 <0.01 **
## foragingterrestrial     0.2935   0.2329   0.3626      100 <0.01 **
## log(mass.g)             0.4240   0.3799   0.4618      100 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

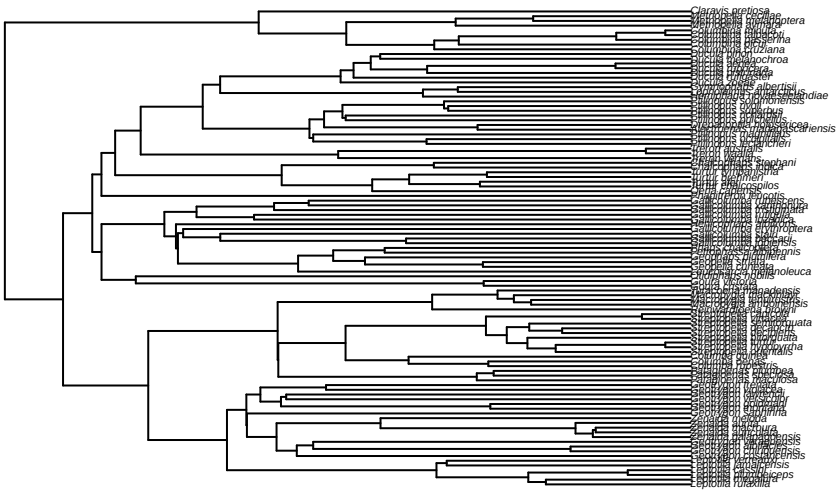
- EXECISE 3: *Include also the geographical location ("location") as a random effect. Remember you will need to specify the prior for this new factor as well.*

PART 3: Phylogenetic effects and random variance.

As species are not independent of each other due to shared ancestry, we need to take this into account. MCMCglmm allows to include phylogenetic similarity as a random effect. For this, we only need a phylogenetic tree and a column in our data called 'animal' that corresponds to the phylogenetic tips of the tree.

We open the tree and plot it:

```
tree <- read.tree("data/ColumbidaeTree.tre")
plot(tree,cex=0.3)
```



Now, we add a column in our data with the tips of the tree. We already have a column "species", but MCMCglmm need a column names "animal" to associate with the tree.

```
dove$animal <- dove$species
prior3.1 <- list(G = list(G1 = list(nu=0.002, V=1),G2 = list(nu=0.002, V=1)),
                 R = list(nu=0.002, V=1))
```

```
mod3.1 <- MCMCglmm(log(tarsus.mm) ~ 1+log(mass.g),
                   random= ~animal + measureID,
                   data = dove, prior = prior3.1,verbose=F,
                   pedigree=tree,
                   nitt = 11000, thin=10, burnin = 100)
summary(mod3.1)
```

```
##
##  Iterations = 101:10991
##  Thinning interval  = 10
##  Sample size  = 1090
##
##  DIC: -221.8788
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal   0.05942  0.04047  0.08222     1090
##
##                ~measureID
##
##           post.mean  l-95% CI u-95% CI eff.samp
## measureID     1.488 0.0001699   0.4676     1090
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.02144  0.01677  0.02596     1090
##
##  Location effects: log(tarsus.mm) ~ 1 + log(mass.g)
##
##             post.mean l-95% CI u-95% CI eff.samp   pMCMC
## (Intercept)    1.6420   1.0625   2.0873     1090 0.00734 **
## log(mass.g)    0.3127   0.2414   0.3877     1090 < 9e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that different random factors explain different proportion of the variance. We can explore the effect of each random factor:

```
posterior.mode(mod3.1$VCV)
```

```
##        animal     measureID         units
##  0.0505612850 -0.0008784083  0.0203641524
```

However, it's more appropriate to report the intraclass correlation as the proportion of variance explained by each random factor in relative terms. This is done by dividing the variance of factor X by the sum of all varainces.

So the proportion of variance explained by the phylogeny is:

```
total.variance <- sum(posterior.mode(mod3.1$VCV))
IC.animal <- posterior.mode(mod3.1$VCV)[1]/total.variance #animal variance
IC.animal
```

```
##    animal
## 0.7218191
```

IC.animal is expressed in relation to the total variance (1). If we want the % we can do:

```
round(IC.animal*100,2) #round to 2 decimals and multiple by 100
```

```
## animal
##  72.18
```

This is a useful propierty of the MCMCglmm models, as we can see which is the phylogenetic effect (or heterability) of traits or can calculate the repeatability of measurements.

EXERCISE 4: Compare the proportion of variance Include the "measureID" and "location" as a random factor in a new model. Which factors explains more proportion of the variance?*

For more information see:

https://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf