

# Bayesian generalised mixed models with MCMCglmm

*Ferran Sayol*

*5th November 2020*

*Course editions:* 5th November 2020, Gothenburg (Sweden) 20th December 2018, Erlangen (Germany)

*Summary* This is a short guide of running mixed models under a bayesian framework using the MCMCglmm R package (Hadfield 2010).

The course is structured in three parts: 1) See the basics of MCMCglmm models (run and check the output); 2) Adding random effects; 3) Correct for phylogenetic effects.

## Used packages

First we need to install the MCMCglmm package. We also install an additional packages to do plots (ggplot2) and work with phylogenetic trees (phytools). Next, we load the packages we have just installed from the library.

```
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 use "setwd()" if necessary
```

```
## [1] "/Users/xsayfe/Google Drive/1_ResearchOrganization/GitHub/MCMCglmmCourse"
```

```
cdata <- read.table("data/ColumbidaeTraits.txt",h=T)
```

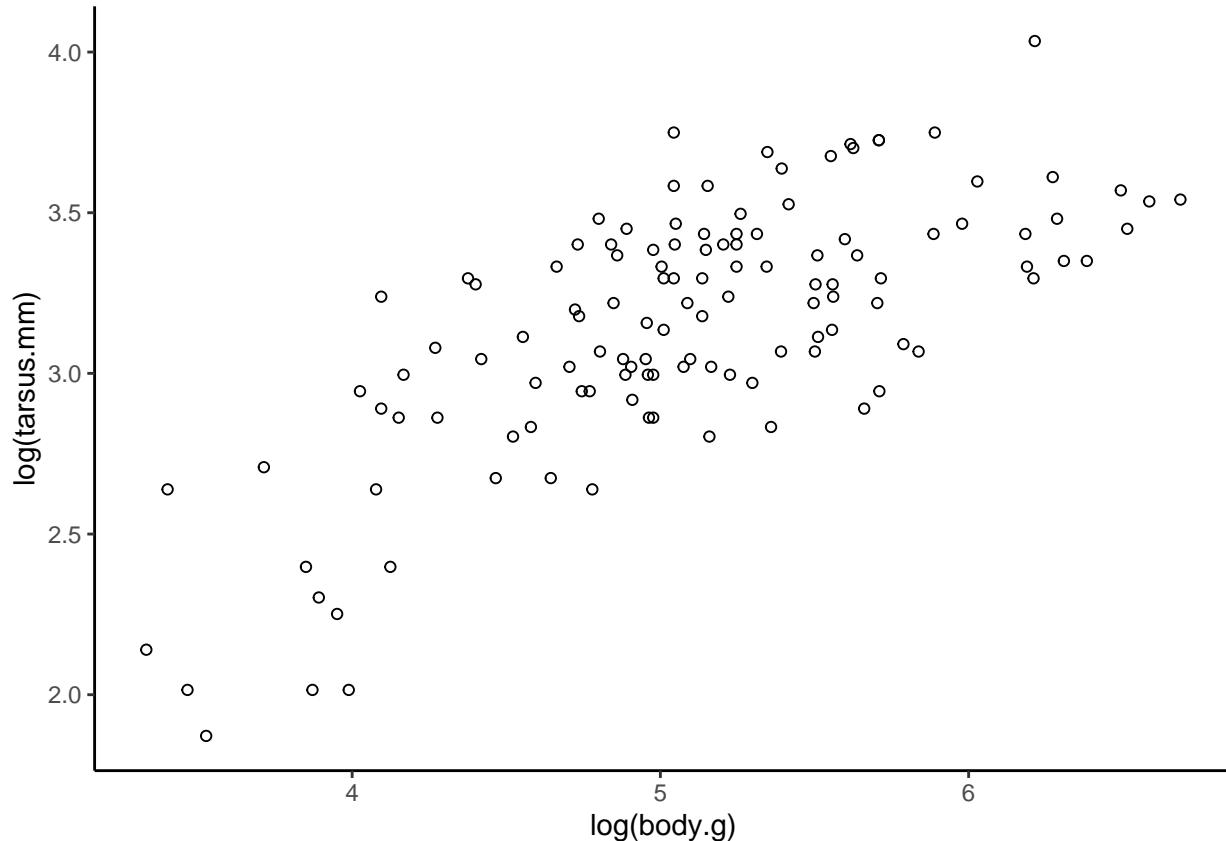
This data file is based on a subset of the data used in an analysis on the relation between foraging behaviour and the evolution of morphological adaptations (Lapiedra et al. 2013).

```
head(cdata)
```

```
##           species tarsus.mm tail.mm wing.mm body.g   foraging
## 1 Uropelia_campestris      8.5   55.5   64.0    28 terrestrial
## 2  Geopelia_cuneata     14.0  102.5   91.0    30 terrestrial
## 3 Columbina_passerina      7.5   30.0   78.0    32 terrestrial
## 4  Columbina_minuta      6.5   29.5   64.5    34 terrestrial
## 5      Oena_capensis     15.0  140.5  107.0    41 terrestrial
## 6 Columbina_cruziana     11.0   29.5   83.0    47 terrestrial
##           region      measure
## 1  Americas ResearcherA
## 2 Australasia ResearcherC
## 3  Americas ResearcherC
## 4  Americas ResearcherB
## 5    Africa ResearcherB
## 6  Americas ResearcherA
```

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

```
ggplot(cdata, aes(x=log(body.g), y=log(tarsus.mm))) +  
  geom_point(shape=21)+theme_classic()
```



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

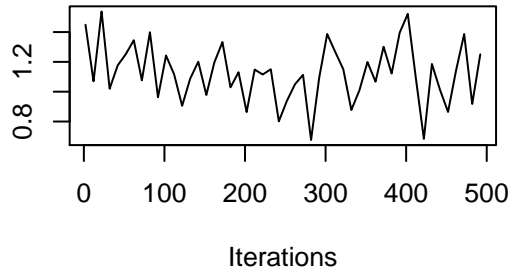
When running an MCMCglmm, we need to specify some parameters of the mcmc chain: How many iterations we want to run the chain for (nitt), the burnin we want to discard at the start of the chain (burnin) and also how often we want to sample and store from the chain (thin). We discard a burnin as is normal practice in Bayesian analyses.

```
prior1 <- list(R=list(V = 1,nu = 0.002)) #We will see this later  
  
mod1.1 <- MCMCglmm(log(tarsus.mm) ~ log(body.g),  
  data = cdata, prior = prior1,verbose=F,  
  nitt = 500, thin=10, burnin = 1)
```

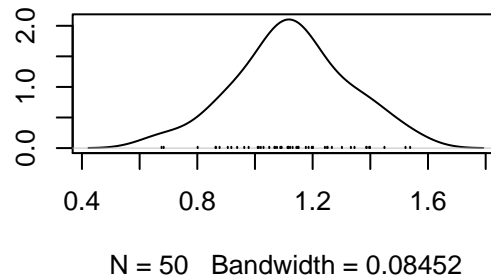
Before we even look at our model output we can check if the model ran appropriately (e.g. if the model converged and if there is any issue in chain mixing). We can do this by visually inspecting the chains. For the fixed effects, the samples are stored in *Sol* and for the variance terms, in *VCV*. So *modelSol[,1]* \* *will give you the first fixed term, in this case the intercept, and \* model 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 first fixed term, the intercept.  
plot(mod1.1$Sol)
```

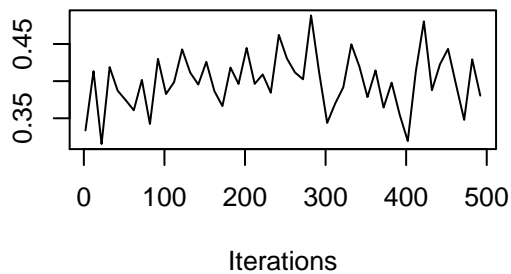
**Trace of (Intercept)**



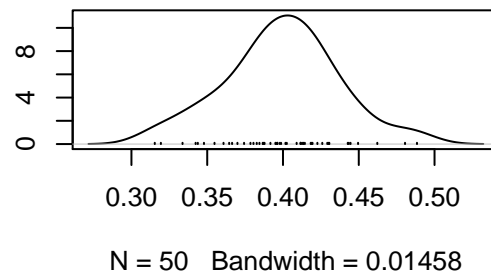
**Density of (Intercept)**



**Trace of log(body.g)**

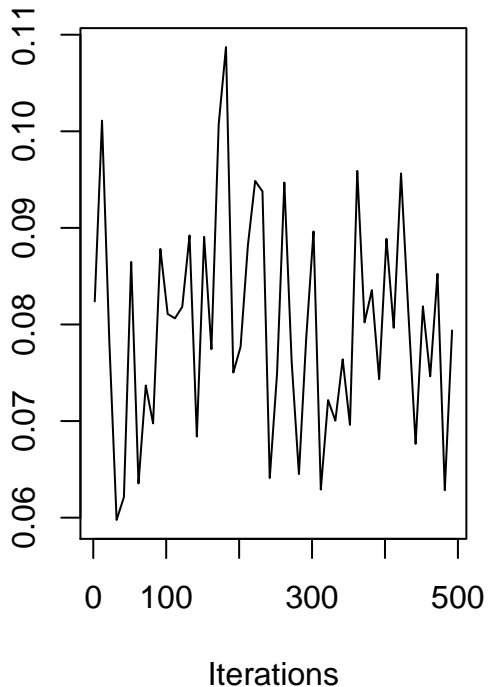


**Density of log(body.g)**

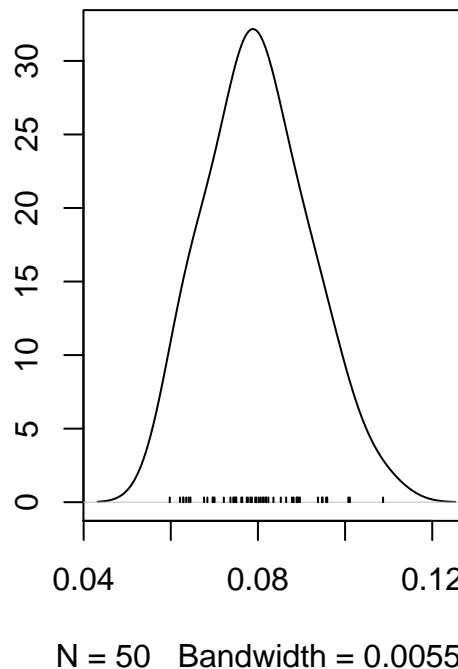


```
#plot the first variance term, the residual error term.  
plot(mod1.1$VCV)
```

**Trace of units**



**Density of units**



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. (Note that this is the same thing we were looking at for another purpose yesterday when we disucsed temporal autocorrelation for CO2 and temperature)

Let's see some diagnosis (Autocorrelation)

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

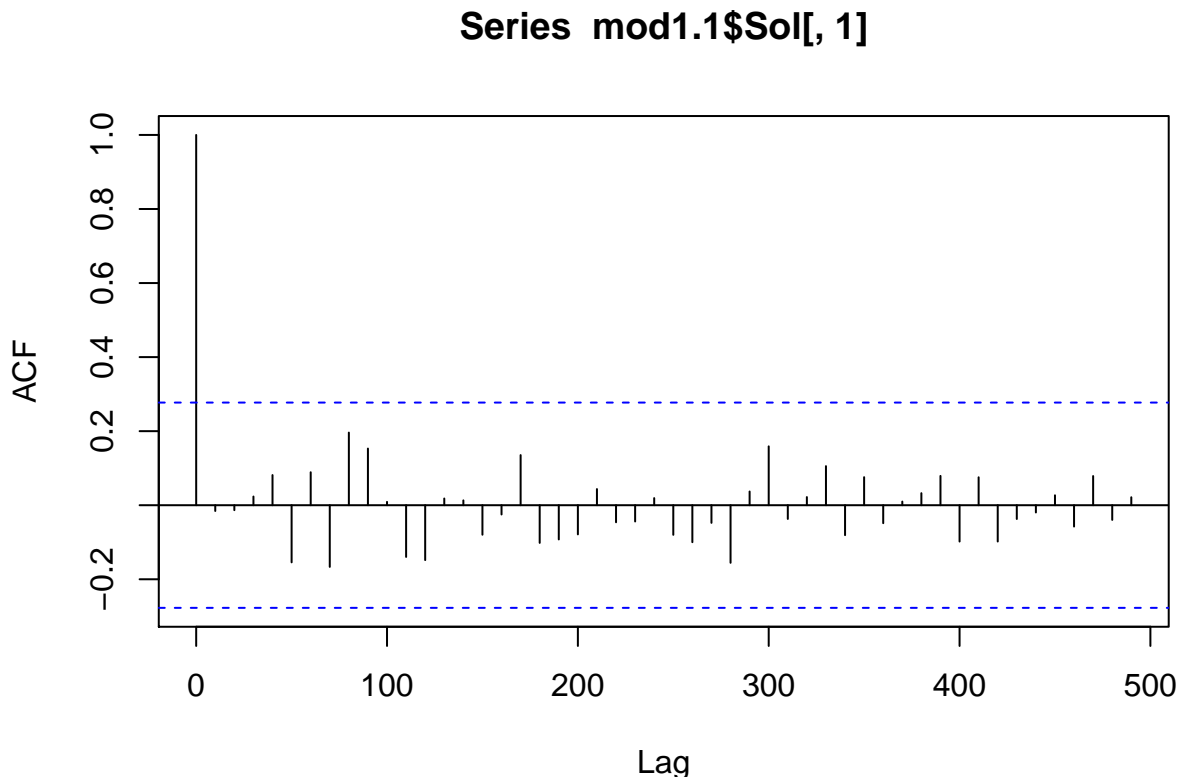
```
##           (Intercept)  log(body.g)
## Lag 0      1.000000000  1.000000000
## Lag 10    -0.015605377 -0.015048358
## Lag 50    -0.154307982 -0.131006929
## Lag 100   0.009059353 -0.007210664
```

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

```
##           units
## Lag 0      1.00000000
## Lag 10     0.05206419
## Lag 50     0.01704753
## Lag 100    -0.13711003
```

Another way is to look at autocorrelation plots for each of the traces. For example, let's check the autocorrelation 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 low (i.e. less than 0.1 is recommended). The thinning is used to 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 thinning 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 thinning 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(body.g),
  data = cdata, prior = prior1, verbose=F,
  nitt = 101000, thin=100, burnin = 1000)
```

---

Now, let's explore the output of model.

```
summary(mod1.1)

##
## Iterations = 1001:100901
## Thinning interval = 100
## Sample size = 1000
##
## DIC: 37.41337
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units  0.07815  0.05941  0.09751      887
##
## Location effects: log(tarsus.mm) ~ log(body.g)
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)  1.1256  0.7755  1.4770     1000 <0.001 ***
## log(body.g)   0.3987  0.3258  0.4634     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(body.g)
##  1.1529137  0.3977108
```

Posterior mean of fixed factors:

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

```
##      lower      upper
## (Intercept) 0.7754561 1.4770454
## log(body.g) 0.3257862 0.4634093
```

```
## attr("Probability")
## [1] 0.95
```

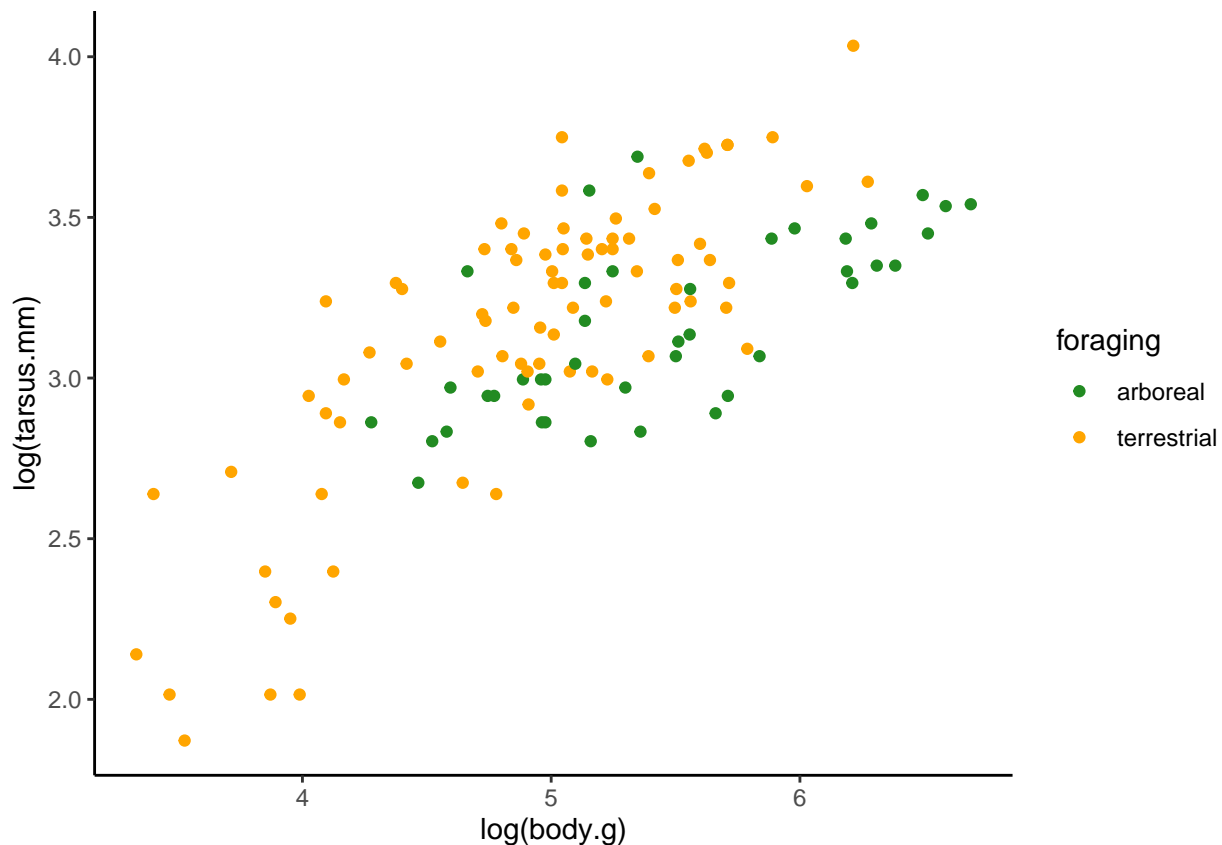
We also have the effective sample size (*eff.samp*).

Finally, the *pMCMC* is an equivalent of the famous *p-value* in Frequentist statistics. In MCMCglmm models, this is calculated as two times the probability that the estimate is either  $>$  or  $<$  0, using whichever 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.

To evaluate the fit of the model, we have the parameter *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.

Coming back to the initial question, we want to see if foraging behaviour can explain tarsus length while accounting for body size. Let’s do a plot first:

```
ggplot(cdata, aes(x=log(body.g), y=log(tarsus.mm), color=foraging)) +
  geom_point(shape=19)+theme_classic()+scale_color_manual(values=c("forestgreen","orange"))
```



It looks like relative length of the tarsus is influenced by foraging behaviour. But we need to formally 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(cdata)

## [1] "species"    "tarsus.mm" "tail.mm"   "wing.mm"   "body.g"    "foraging"
## [7] "region"     "measure"

mod1.2 <- MCMCglmm(log(tarsus.mm) ~ log(body.g)+foraging,
                   data = cdata, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)
# We can remove the intercept:
mod1.3 <- MCMCglmm(log(tarsus.mm) ~ log(body.g)+foraging-1,
                   data = cdata, prior = prior1,verbose=F,
                   nitt = 101000, thin=100, burnin = 1000)

summary(mod1.2)

##
## Iterations = 1001:100901
## Thinning interval = 100
## Sample size = 1000
##
## DIC: 20.185
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.06717  0.04976  0.08347    1000
##
## Location effects: log(tarsus.mm) ~ log(body.g) + foraging
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6687   0.2580   1.0564    1093  0.002 **
## log(body.g)      0.4568   0.3840   0.5264    1000 <0.001 ***
## foragingterrestrial 0.2406   0.1311   0.3456    1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

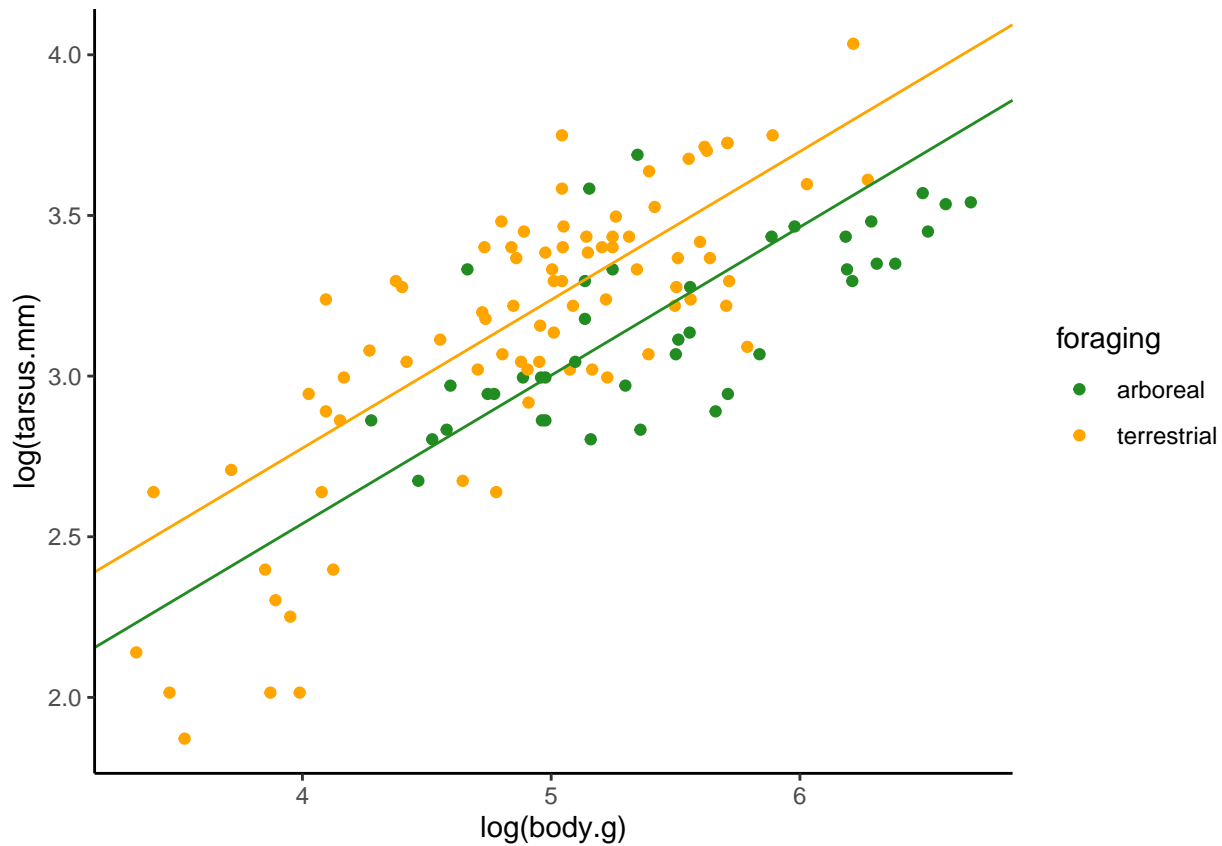
summary(mod1.3)

##
## Iterations = 1001:100901
## Thinning interval = 100
## Sample size = 1000
##
## DIC: 20.18213
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.06672  0.05126  0.08297    1244
##
## Location effects: log(tarsus.mm) ~ log(body.g) + foraging - 1
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
```

```
## log(body.g)          0.4579   0.3895   0.5220    1000 <0.001 ***
## foragingarboreal     0.6669   0.2973   1.0290    1000 <0.001 ***
## foragingterrestrial  0.9042   0.5880   1.2499    1000 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

*# Now we can plot the output of the models:*

```
ggplot(cdata, aes(x=log(body.g), y=log(tarsus.mm), color=foraging)) +
  geom_point(shape=19) + theme_classic() + scale_color_manual(values=c("forestgreen", "orange")) +
  geom_abline(intercept = posterior.mode(mod1.3$Sol)[2], slope = posterior.mode(mod1.3$Sol)[1], color="forestgreen") +
  geom_abline(intercept = posterior.mode(mod1.3$Sol)[3], slope = posterior.mode(mod1.3$Sol)[1], color="orange")
```



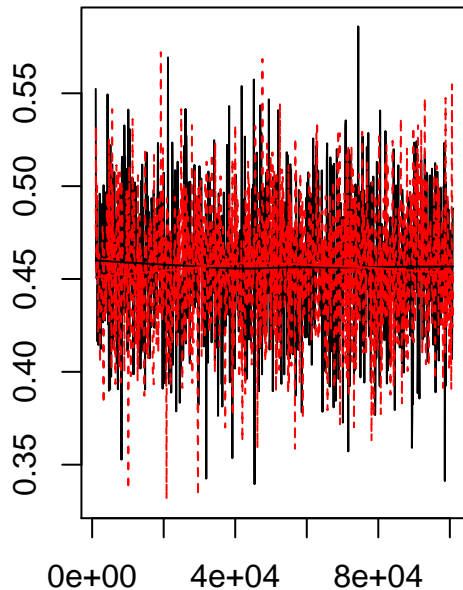
## Model convergence

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.

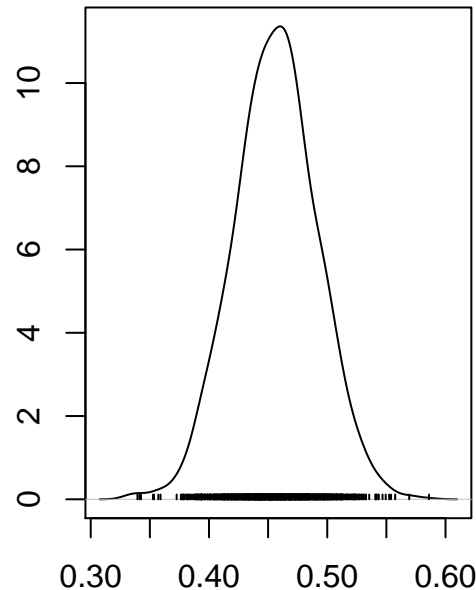
```
mod1.3a <- MCMCglmm(log(tarsus.mm) ~ log(body.g)+foraging,
  data = cdata, prior = prior1, verbose=F,
  nitt = 101000, thin=100, burnin = 1000)
mod1.3b <- MCMCglmm(log(tarsus.mm) ~ log(body.g)+foraging,
  data = cdata, prior = prior1, verbose=F,
  nitt = 101000, thin=100, burnin = 1000)
```



```
#They have reached the same solution?
plot(mcmc.list(mod1.3a$Sol[,2], mod1.3b$Sol[,2]))
```



Iterations



N = 1000 Bandwidth = 0.007911

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

```
##
## Iterations = 1001:100901
## Thinning interval = 100
## Sample size = 1000
##
## DIC: 20.15376
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.06774  0.05004  0.08523    897.2
##
## Location effects: log(tarsus.mm) ~ log(body.g) + foraging
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6762  0.3233  1.0523    1000  0.002 **
## log(body.g)      0.4559  0.3871  0.5205    1000 <0.001 ***
## foragingterrestrial 0.2358  0.1405  0.3466    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.

For fixed effects (B), MCMCglmm uses a normal distribution. The terms mu and V give the mean and variance of this normal distribution. Here we set mu as 0 and the variance as a large number to make these priors effectively uninformative (These are called non-informative priors).

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.

```
fixMu <- rep(0,3)
fixV <- diag(3)*10^8
prior2.1 <- list(B=list(mu=fixMu,V=fixV),R=list(V = 1,nu = 0.002))

mod2.1 <- MCMCglmm(log(tarsus.mm) ~ foraging+log(body.g),
  data = cdata, 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: 20.24236
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units      0.0677  0.05231  0.08871      100
##
## Location effects: log(tarsus.mm) ~ foraging + log(body.g)
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6844   0.3984   0.9948     100 <0.01 **
## foragingterrestrial 0.2330   0.1349   0.3467     100 <0.01 **
## log(body.g)       0.4549   0.4047   0.5144     100 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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))

mod2.1 <- MCMCglmm(log(tarsus.mm) ~ foraging+log(body.g),
  data = cdata, 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: 20.2581
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.06727  0.05236  0.08514      100
##
## Location effects: log(tarsus.mm) ~ foraging + log(body.g)
##
##      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6555   0.2533   0.9269     100 <0.01 **
## foragingterrestrial 0.2420   0.1505   0.3398     100 <0.01 **
## log(body.g)      0.4590   0.4075   0.5280     100 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

MCMCglmm uses inverse-Wishart priors for any of the variance terms (R or G). The variance is described by the parameters  $\nu$  and  $V$ .

$V=1$  and  $\nu=0.002$  is frequently used for variance components.

See some information about inverse-Wishart priors here: [https://en.wikipedia.org/wiki/Inverse-Wishart\\_distribution](https://en.wikipedia.org/wiki/Inverse-Wishart_distribution)

## Mixed models: Adding random factors to our MCMCglmm

Just to remember, mixed models are referred to models that contain both fixed and random factors. Depending in our interest a factor might be considered random or fix. Here, you can see a more detailed explanation.

Let's add a random term of measurement ("measure") in the *cdata* example. This variable states which researcher took the morphological measurements (A,B,C). Like before, we need to set up the prior. To add a random term we now add a  $G$  structure that acts just like the other random variance term and is defined using  $\nu$  and  $V$ .

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

Here, we will include "measure" as a random effect:

```
table(cdata$measure)
```

```
##
## ResearcherA ResearcherB ResearcherC
##          40          41          42
```

We can do so by including the random variable in the model in the section `random=` ~.

```
table(cdata$measure)
```

```
##
## ResearcherA ResearcherB ResearcherC
##          40          41          42
```

```
names(cdata)
```

```
## [1] "species" "tarsus.mm" "tail.mm" "wing.mm" "body.g" "foraging"
## [7] "region" "measure"
```

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

```
##
## Iterations = 101:1091
## Thinning interval = 10
## Sample size = 100
##
## DIC: 18.33814
##
## G-structure: ~measure
##
##          post.mean 1-95% CI u-95% CI eff.samp
## measure  0.01713 0.0003811 0.09656      100
##
## R-structure: ~units
##
##          post.mean 1-95% CI u-95% CI eff.samp
## units    0.06502 0.04666 0.08098      100
##
## Location effects: log(tarsus.mm) ~ foraging + log(body.g)
##
##          post.mean 1-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6698  0.3032  1.1025    226.1 <0.01 **
## foragingterrestrial 0.2494  0.1278  0.3354    100.0 <0.01 **
## log(body.g)       0.4570  0.3898  0.5249    100.0 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
prior2.3 <- list(G = list(G1 = list(nu=0.002, V=1), G2 = list(nu=0.002, V=1)), R = list(nu=0.002, V=1))
table(cdata$region)
```

```
##
##      Africa  Americas Australasia  Eurasia Indomalayan
##          17         40          41         8          17
```

```
names(cdata)
```

```
## [1] "species" "tarsus.mm" "tail.mm" "wing.mm" "body.g" "foraging"
## [7] "region" "measure"
```

```
mod2.2 <- MCMCglmm(log(tarsus.mm) ~ foraging+log(body.g),
  random= ~measure+region,
  data = cdata, prior = prior2.3, verbose=F,
  nitt = 1100, thin=10, burnin = 100)
summary(mod2.2)
```

```
##
```

```

## Iterations = 101:1091
## Thinning interval = 10
## Sample size = 100
##
## DIC: 18.34891
##
## G-structure: ~measure
##
##          post.mean  l-95% CI u-95% CI eff.samp
## measure    0.01664 0.0006464 0.06204    81.26
##
##          ~region
##
##          post.mean  l-95% CI u-95% CI eff.samp
## region    0.01244 0.0003491 0.05304     100
##
## R-structure: ~units
##
##          post.mean l-95% CI u-95% CI eff.samp
## units    0.06333 0.04953 0.07839     100
##
## Location effects: log(tarsus.mm) ~ foraging + log(body.g)
##
##          post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.6079  0.1720  1.0929     100 <0.01 **
## foragingterrestrial 0.2736  0.1472  0.3649     100 <0.01 **
## log(body.g)       0.4646  0.3992  0.5608     100 <0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

---

### PART 3: Phylogenetic effects and random variance.

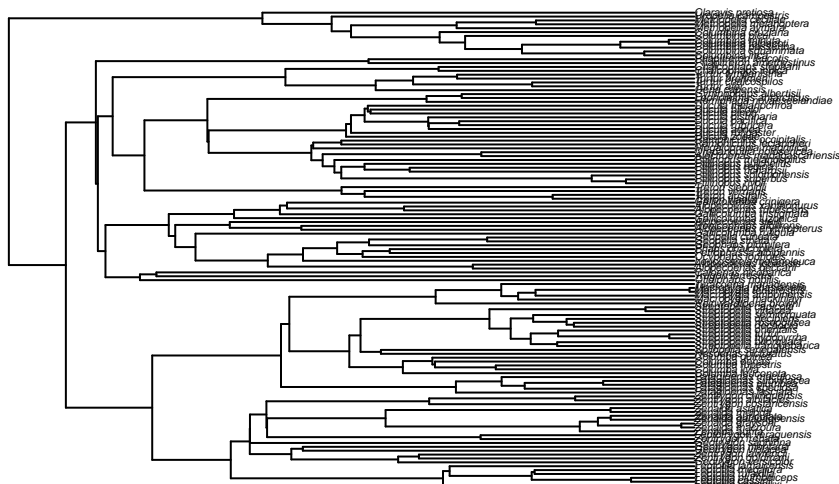
As species are not independent of each other due to shared ancestry, we need to take this into account. MCMCglmm allows to include phylogenetic similarity as a random effect. For this, we only need a phylogenetic tree and a column in our data called 'animal' that corresponds to the phylogenetic tips of the tree. Phylogenetic effects can be seen as another type of autocorrelation in addition to spatial and temporal autocorrelation and arises because closely related species are expected to be more similar to each other than random species.

We open the tree and plot it:

```

ctree <- read.tree("data/ColumbidaeTree.tre")
plot(ctree,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.

```

cdata$animal <- cdata$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(body.g),
                  random= ~animal + measure,
                  data = cdata, prior = prior3.1, verbose=F,
                  pedigree=ctree,
                  nitt = 11000, thin=10, burnin = 100)
summary(mod3.1)

```

```

##
## Iterations = 101:10991
## Thinning interval = 10
## Sample size = 1090
##
## DIC: -146.0802
##
## G-structure: ~animal
##
##          post.mean l-95% CI u-95% CI eff.samp
## animal    0.04891  0.03155  0.06762    832.2
##
##          ~measure
##
##          post.mean l-95% CI u-95% CI eff.samp
## measure    0.02144  0.0001034  0.04012    1090
##
## R-structure: ~units
##
##          post.mean l-95% CI u-95% CI eff.samp
## units    0.009806  0.004539  0.01561    808.6
##
## Location effects: log(tarsus.mm) ~ 1 + log(body.g)
##
##          post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)   1.5827   1.1704   1.9772    1090 0.00183 **

```

```
## log(body.g)    0.3087    0.2446    0.3809      1227 < 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          measure          units
## 0.0469304325 -0.0002461996  0.0095643219
```

However, it's more useful to report the intraclass correlation in relative terms, as the proportion of variance explained by each random factor. This is because we are interested in the proportion of variance explained by each factor, but not by the absolute variance explained (Which will vary a lot between variables, studies, samples,...).

The proportion of variance explained can be calculated by dividing the variance of factor X by the sum of all variances. 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
## 0.8343402
```

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
## 83.43
```

This is a useful property of the MCMCglmm models, as we can see which is the phylogenetic effect of traits or can calculate the repeatability of measurements, in case we have multiple measurements for each specimen or species.