Bayesian generalised mixed models with MCMCglmm

Ferran Sayol

8th September 2021

Previous 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/fsayol/My Drive/1_ResearchOrganization/5_Teaching/2021/2021_GG
BC_SpatialR"

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

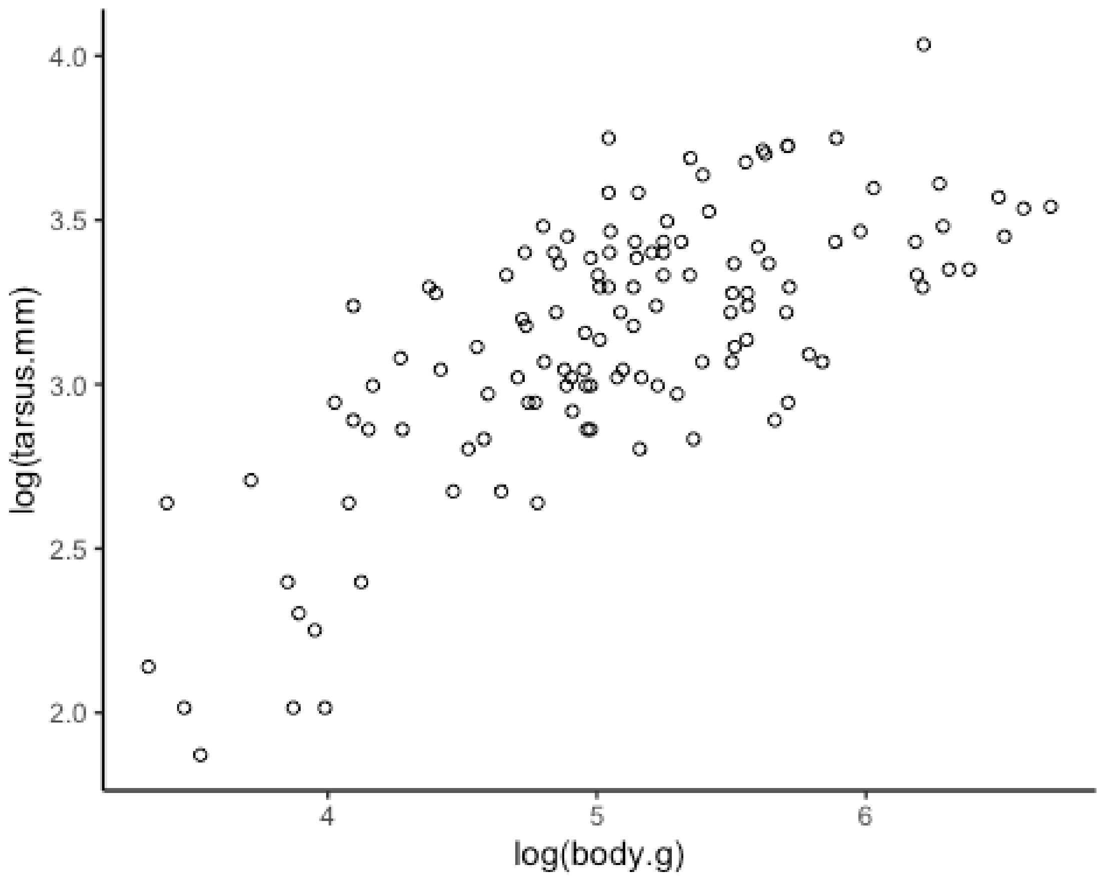
This data file is based on a subset of the data used in an analysis on the relation between foraging behaviour and the evolution of morphological adaptations (Lapiedra et al. 2013).

```
head(cdata)

##                species tarsus.mm tail.mm wing.mm body.g   foraging      re
gion
## 1 Uropelia_campestris       8.5    55.5    64.0     28 terrestrial     Amer
icas
## 2    Geopelia_cuneata      14.0   102.5    91.0     30 terrestrial Austral
asia
## 3 Columbina_passerina       7.5    30.0    78.0     32 terrestrial     Amer
```

```
icas
## 4    Columbina_minuta       6.5     29.5     64.5      34 terrestrial    Amer
icas
## 5       Oena_capensis      15.0    140.5    107.0      41 terrestrial      Af
rica
## 6  Columbina_cruziana      11.0     29.5     83.0      47 terrestrial    Amer
icas
##       measure
## 1 ResearcherA
## 2 ResearcherC
## 3 ResearcherC
## 4 ResearcherB
## 5 ResearcherB
## 6 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()
```



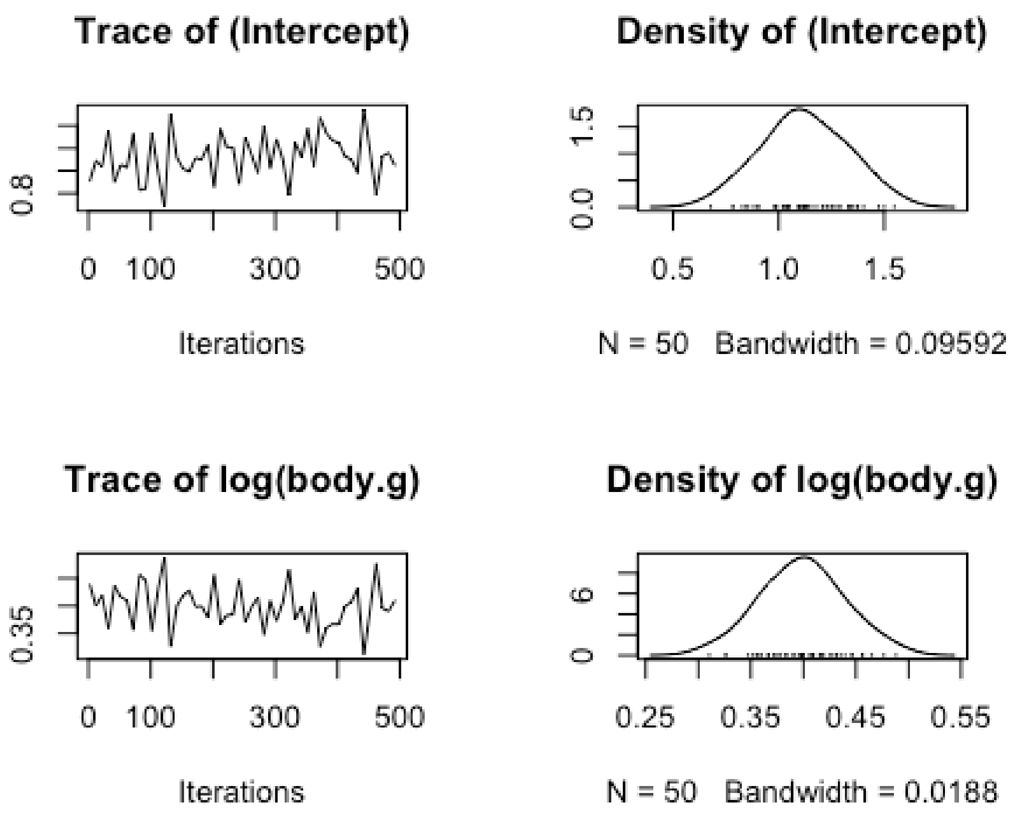
Now let's run a simple model with tarsus.mm as response of body size (body.g).

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 practive in Bayesian analyses.

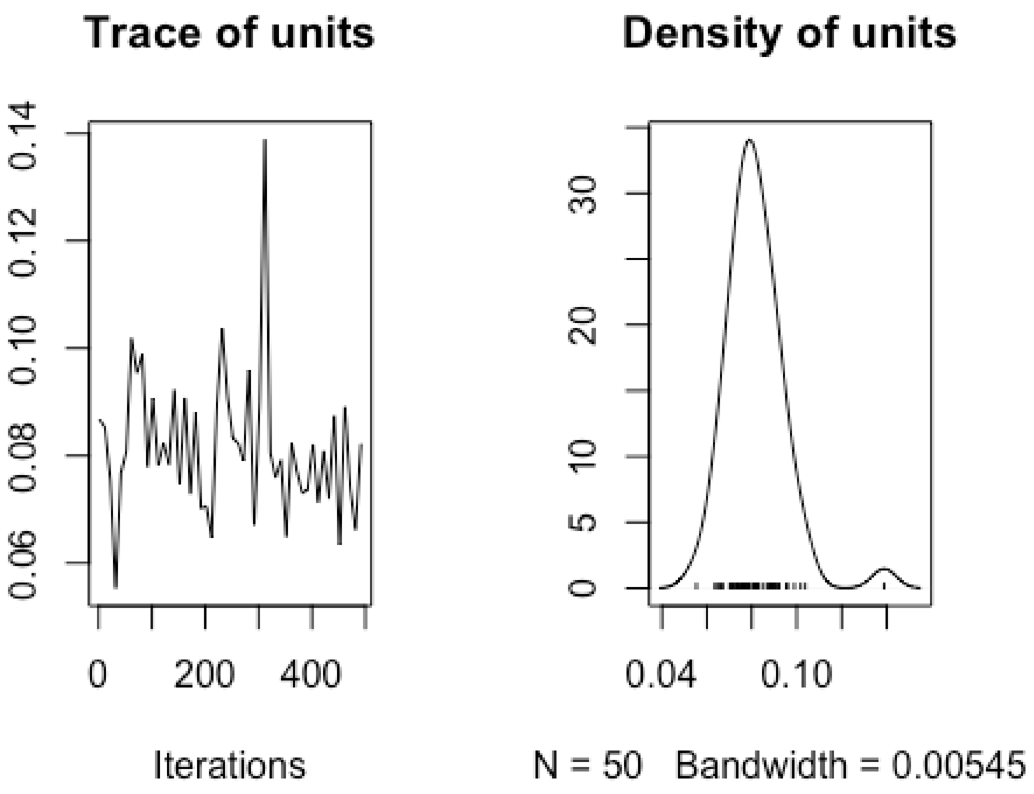
```
prior1 <- list(R=list(V = 1,nu = 0.002)) #We will see this later

mod1.1 <- MCMCglmm(log(tarsus.mm) ~ log(body.g),
                   data = cdata, prior = prior1,verbose=F,
                   nitt = 500, thin=10, burnin = 1)
```

Before we take a 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. We can extract the full chains by checking the parameters *Sol* (for fixed effects) and *VCV* (from the random effects). For instance, model$*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. (We will return to this function tomorrow for another purpose).

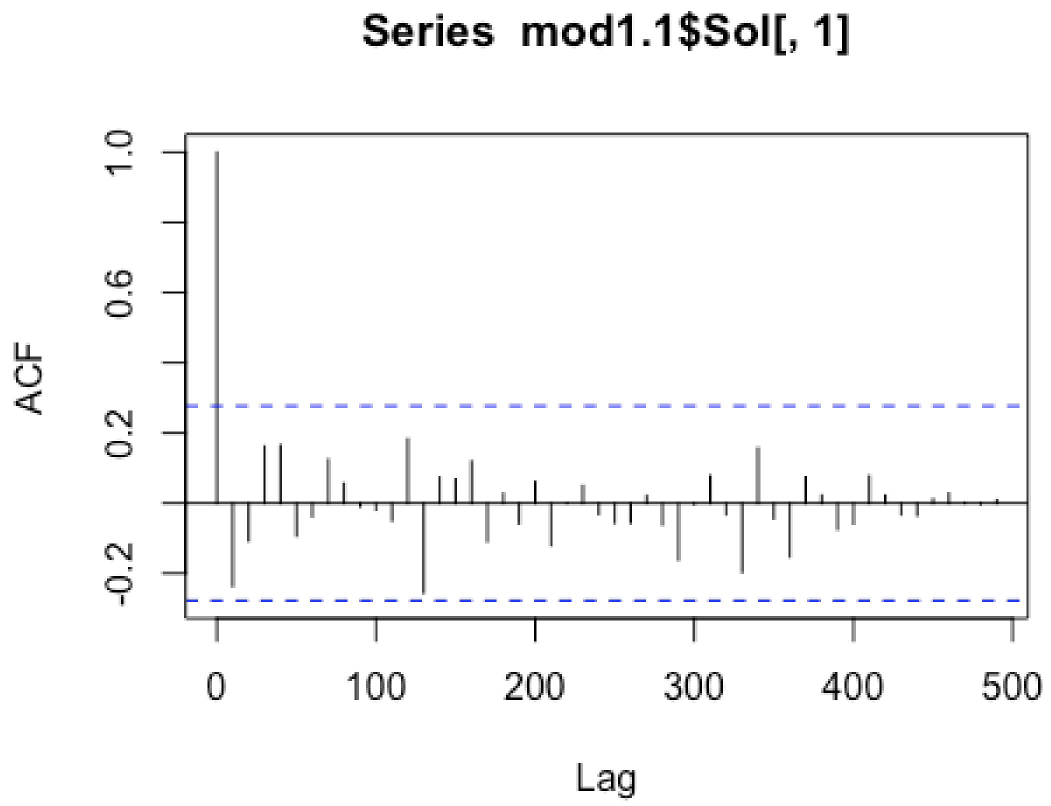
Let's see some diagnosis (Autocorrelation)

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

##         (Intercept)  log(body.g)
## Lag 0    1.00000000  1.000000000
## Lag 10  -0.23644594 -0.220241388
## Lag 50  -0.09301315 -0.111805511
## Lag 100 -0.01902086  0.006078279

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

```
##                units
## Lag 0    1.00000000
## Lag 10   0.09494951
## Lag 50  -0.08837314
## Lag 100 -0.08898798
```

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 low (i.e. less than 0.1 is reccomended). 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 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.*

Now, let's explore the output of model.

```
summary(mod1.1)

##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 37.3822
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.07813   0.0601  0.09954     1000
##
##  Location effects: log(tarsus.mm) ~ log(body.g)
##
##             post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)    1.1134   0.7603   1.4971     1000 <0.001 ***
## log(body.g)    0.4010   0.3279   0.4679     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)
##   1.0833535   0.4033603
```

Posterior mean of fixed factors:

```
HPDinterval(mod1.1$Sol)

##                 lower     upper
## (Intercept) 0.7603448 1.4970615
## log(body.g) 0.3278635 0.4678904
## attr(,"Probability")
## [1] 0.95
```

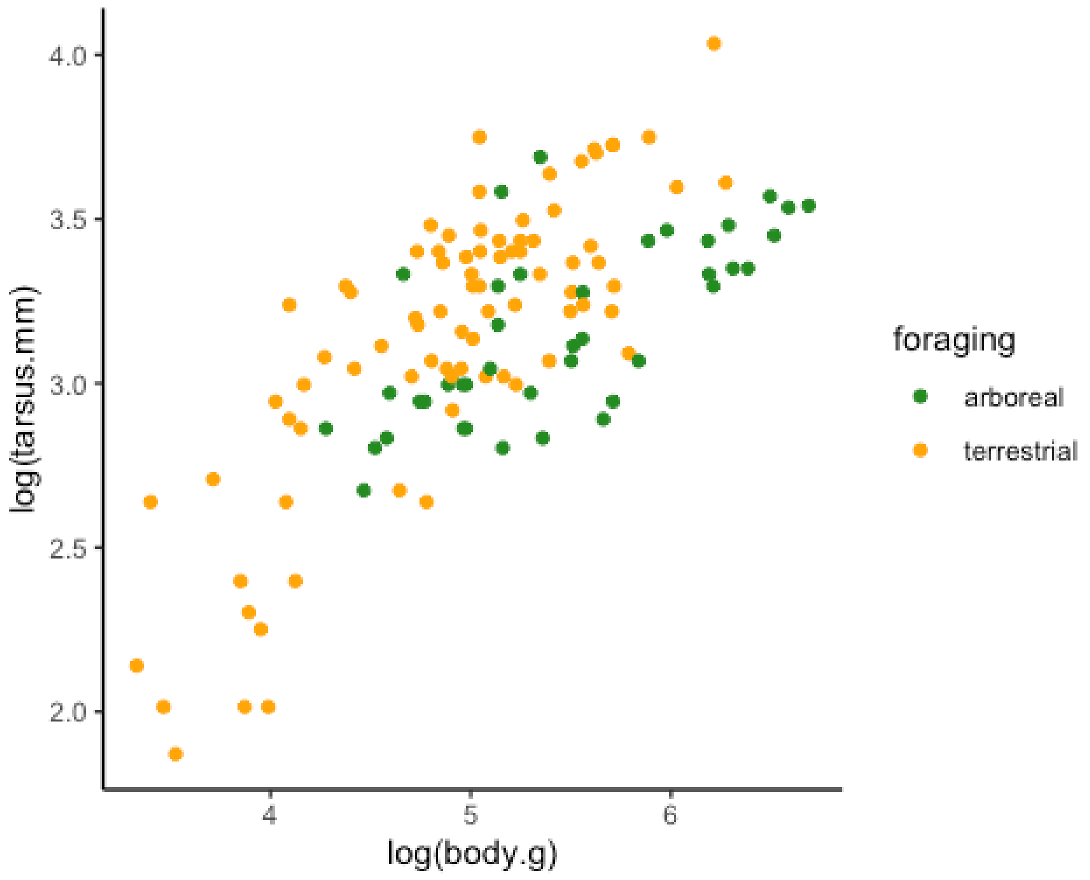
We also have the effective sample size (*eff.samp*).

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 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.

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.

Comming back to the initial question, we want to see if foraging behavioir 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("forestgre
en","orange"))
```



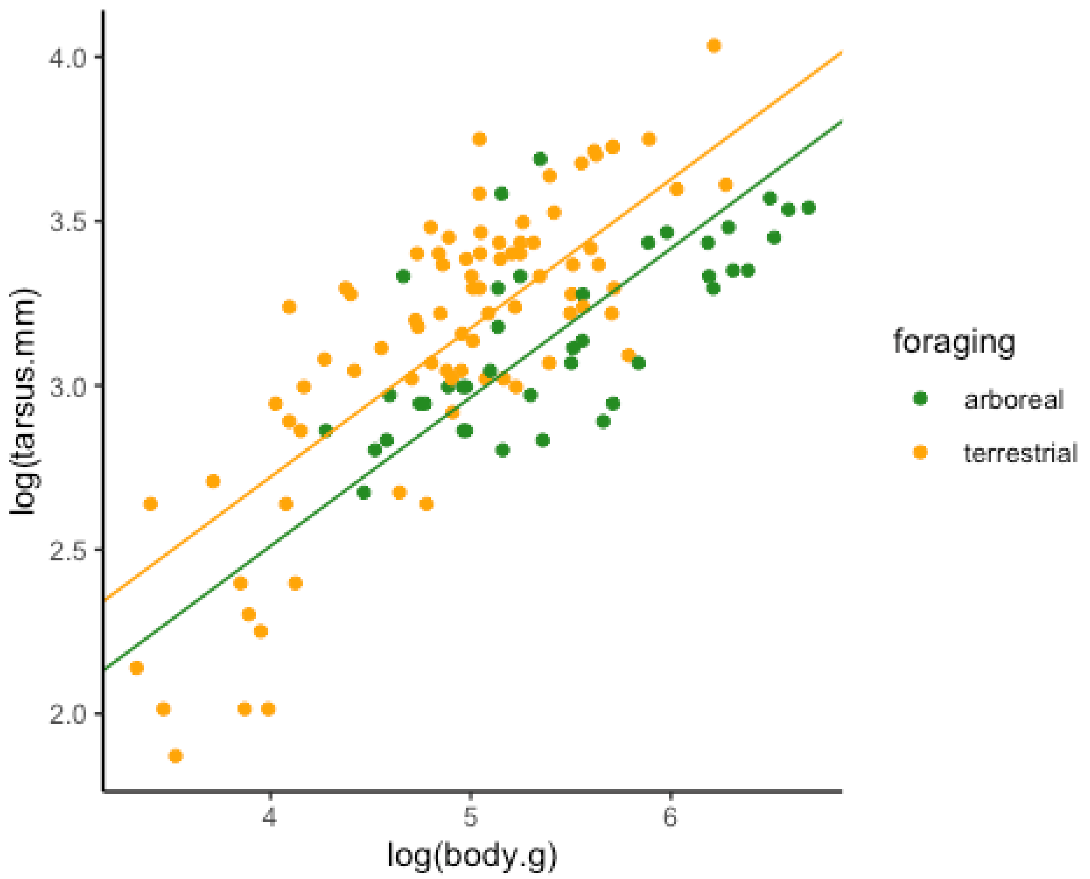
It looks like relative length of the tarsus is influenced by foraging behaviour. But we need to formally test this:

- 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: 20.19632
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06711  0.05131  0.08528     1000
##
##  Location effects: log(tarsus.mm) ~ log(body.g) + foraging
##
##                      post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)             0.6680   0.3101   1.0310     1000  0.006 **
## log(body.g)             0.4573   0.3859   0.5178     1000 <0.001 ***
## foragingterrestrial     0.2397   0.1339   0.3480     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: 20.18396
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06763   0.0527  0.08746    780.3
##
##  Location effects: log(tarsus.mm) ~ log(body.g) + foraging - 1
##
##                      post.mean l-95% CI u-95% CI eff.samp  pMCMC
## log(body.g)             0.4559   0.3849   0.5194   1019.2 <0.001 ***
## foragingarboreal        0.6759   0.3048   1.0560    996.7  0.002 **
## foragingterrestrial     0.9134   0.5886   1.2739   1022.7 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

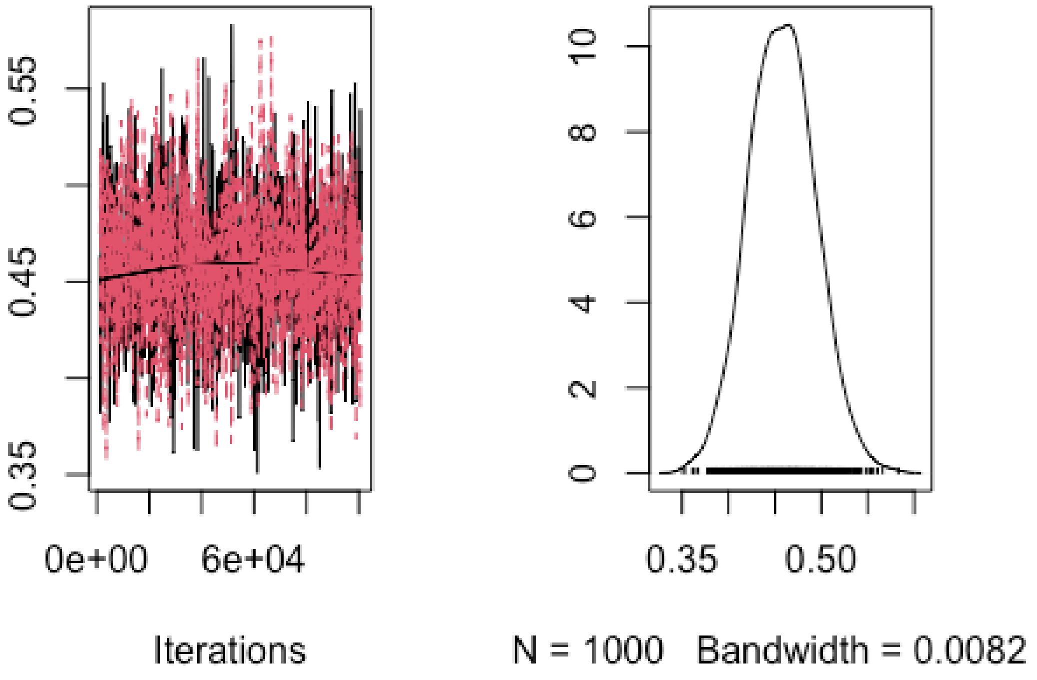
```
# Now we can plot the output of the models:

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



```
summary(mod1.3b)

##
##  Iterations = 1001:100901
##  Thinning interval  = 100
##  Sample size  = 1000
##
##  DIC: 20.1995
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06784  0.05215  0.08629     1000
##
##  Location effects: log(tarsus.mm) ~ log(body.g) + foraging
##
##                        post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)               0.6617   0.2973   1.0725   1000.0  0.002 **
## log(body.g)               0.4586   0.3920   0.5298   1000.0 <0.001 ***
## foragingterrestrial       0.2389   0.1345   0.3410    909.6 <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="gaussia
n",
                     nitt = 1100, thin=10, burnin = 100)
summary(mod2.1)
```

```
## 
##  Iterations = 101:1091
##  Thinning interval  = 10
##  Sample size  = 100 
## 
##  DIC: 19.98019 
## 
##  R-structure:  ~units
## 
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06713  0.04742  0.08047      139
## 
##  Location effects: log(tarsus.mm) ~ foraging + log(body.g) 
## 
##                     post.mean l-95% CI u-95% CI eff.samp pMCMC   
## (Intercept)            0.6978   0.3242   1.0167    100.0 <0.01 **
## foragingterrestrial    0.2356   0.1303   0.3332    255.6 <0.01 **
## log(body.g)            0.4528   0.3979   0.5144    100.0 <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="gaussia
n",
                     nitt = 1100, thin=10, burnin = 100)
summary(mod2.1)

## 
##  Iterations = 101:1091
##  Thinning interval  = 10
##  Sample size  = 100 
## 
##  DIC: 20.29814 
## 
##  R-structure:  ~units
## 
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06704  0.04641  0.08311    141.8
## 
##  Location effects: log(tarsus.mm) ~ foraging + log(body.g) 
## 
##                     post.mean l-95% CI u-95% CI eff.samp pMCMC   
## (Intercept)            0.6829   0.2916   1.0454    100.0 <0.01 **
## foragingterrestrial    0.2355   0.1460   0.3210    110.8 <0.01 **
## log(body.g)            0.4542   0.3765   0.5133    100.0 <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

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 fixed. 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 varience 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: 18.14845
##
##  G-structure:  ~measure
##
##         post.mean  l-95% CI u-95% CI eff.samp
## measure    0.0147 0.0003821  0.04879    70.22
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06474  0.04941  0.08276      100
##
##  Location effects: log(tarsus.mm) ~ foraging + log(body.g)
##
##                       post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)              0.6423   0.2820   1.0967      100 <0.01 **
## foragingterrestrial      0.2594   0.1352   0.3526      100 <0.01 **
## log(body.g)             0.4604   0.3796   0.5311      100 <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.*

  If successful you should get results similar to this:

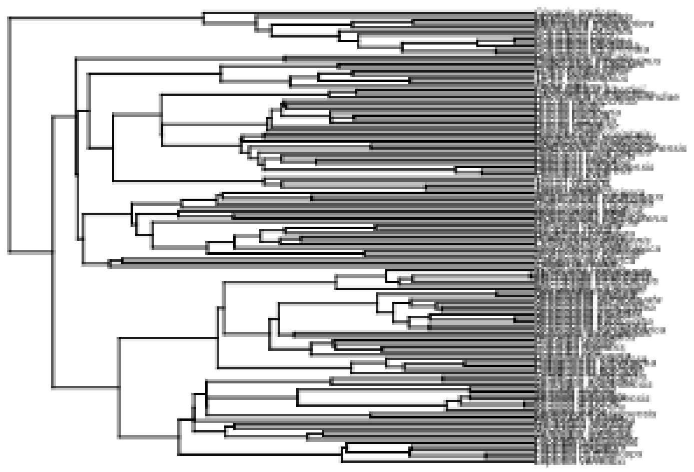
```
## 
##      Africa    Americas Australasia     Eurasia Indomalayan 
##          17          40          41           8          17

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

## 
##  Iterations = 101:1091
##  Thinning interval  = 10
##  Sample size  = 100 
## 
##  DIC: 17.96177 
## 
##  G-structure:  ~measure
## 
##         post.mean  l-95% CI u-95% CI eff.samp
## measure   0.02867 0.0001657  0.07799      100
## 
##                ~region
## 
##        post.mean  l-95% CI u-95% CI eff.samp
## region   0.01036 0.0003469  0.03807    134.4
## 
##  R-structure:  ~units
## 
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.06329  0.04696  0.07844      100
## 
##  Location effects: log(tarsus.mm) ~ foraging + log(body.g) 
## 
##                     post.mean l-95% CI u-95% CI eff.samp pMCMC   
## (Intercept)            0.5830   0.2271   1.0376      100  0.02 * 
## foragingterrestrial    0.2828   0.1890   0.4069      100 <0.01 **
## log(body.g)            0.4660   0.3773   0.5197      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 temporal autocorrelation and arrises because closely related species are expected to be more similar to each other than random species.

We open the tree and plot it:

```r
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.

```r
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: -147.8319
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal   0.04864  0.03212  0.06666    771.6
##
##                ~measure
##
##         post.mean  l-95% CI u-95% CI eff.samp
## measure   0.01388 0.0002477  0.04033     1090
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units  0.009877 0.003993  0.01611    653.9
##
##  Location effects: log(tarsus.mm) ~ 1 + log(body.g)
##
##             post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)    1.5783   1.1560   1.9515     1090 <9e-04 ***
## log(body.g)    0.3103   0.2441   0.3790     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     measure       units
## 0.043256901 0.003534981 0.007580025
```

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 explanied 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
## 0.7955745
```

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
##  79.56
```

This is a useful propierty 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 mutliple measurements for each specimen or species.

FINAL EXERCISE (4): SpatialR and MCMCglmm into practise.

Now, you can do a final exercise to put into practise what you have learned in this tutorial, in the context of spatial analysis. For the spatial part, you will need to load these R-packages:

```
library(sp)
library(raster)
library(sf)
library(rgeos)
library(Imap)
```

We will continue to use the pigeon's dataset example. Now, you can open an additional dataset with some geographical information: *"data/ColumbidaeGeo.txt"*. This data includes coordinates for the breeding and non-breeding coordinates for each species. You can merge this dataset with the previous one, using *merge()*

```
gdata <- read.table("data/ColumbidaeGeo.txt",h=T,stringsAsFactors = F)
cdata <- merge(cdata,gdata,by="species")
```

Now, you can do two different exercises:

EXERCISE 4A: Migratory behaviour.

Classify each species into a migratory or non-migratory species. To do so, you can calculate the migratory distance of each species (i.e. the distance between breeding and non-breeding coordiantes). For this, you can use the function gdist() from the package *imap*.

I will here show how *gdist()* work on. Based on this you should be able to apply it to this specific case.

```
# generate a set of random coordinates
Longitude_1 =  c(110,110,110,110)
Latitude_1= c(10,20,30,40)
cords1 <- cbind(Longitude_1,Latitude_1)
# and another set of random coordinates
Longitude_2 =  c(100,110,120,130)
Latitude_2= c(15,20,25,30)
cords2 <- cbind(Longitude_2,Latitude_2)
#This line will calculate the distance between the the first set of points, t
he second set of points e.t.c.c
gdist(Longitude_1,Latitude_1,Longitude_2,Latitude_2)

## [1]  658.3227    0.0000  611.2998 1149.4887
```

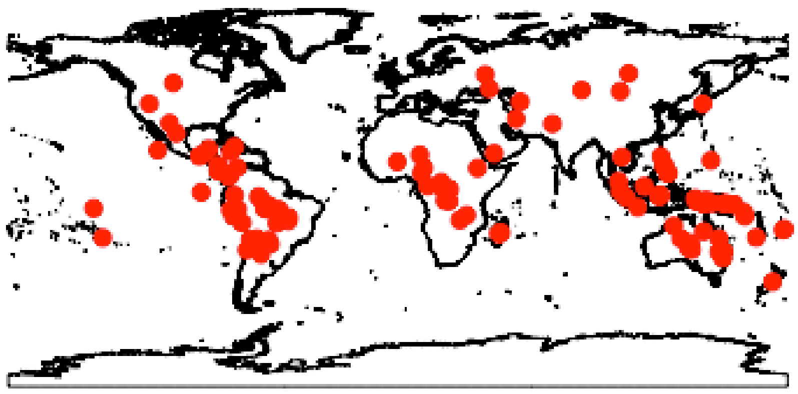
Once you get the migratory distance, you can classify species into resident (distance=0) and migratory (distance>0).

```
## 
##  no yes 
## 114   9
```

EXERCISE 4B: Insularity.

For this exercise, we will be using a world map with different attrributes for islands(1) vs continents(0). This is a **shapefile** called *"MapIslands.shp"*. First, try to open and plot the map and the breeding coordinates.

```
## Reading layer `MapIslands' from data source 
##   `/Users/fsayol/My Drive/1_ResearchOrganization/5_Teaching/2021/2021_GGBC
_SpatialR/data/MapIslands/MapIslands.shp' 
##   using driver `ESRI Shapefile'
## Simple feature collection with 4063 features and 1 field
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -90 xmax: 180 ymax: 83.6341
## Geodetic CRS:  WGS 84
```



Now, you can classify each species into a island or continental species. We can you use the function *over()* to know which breeding coordiantes fall inside islands or continents.

Note 1: Some coordinates might fall in the sea, with value=NA. To solve that we can try to know which is the closest polygon using the function gDistance() and then get the attributes (i.e. island or continent) of the closest polygon.

Note 2: Be aware this is an advanced exercise. If you do not succeed after several trials, go directly to Exercise 4C and try to include "migration" in the MCMCglmm model.

EXERCISE 4C: MCMCglmm into practise

Use the information you calculates in exercises **5A** and **5B** (i.e. insularity and migratory behavior) as a predictor in a *MCMCglmm()*.

- Does the tarsus length vary between migratory and resident species?
- Does the tarsus length vary between island and continental species?
- Can you do a scatterplot to show the data, using *ggplot()*? You can also try to plot the model results, as we did before.