Avoiding the misuse of BLUP in behavioral ecology: II. Multivariate modelling for individual plasticity (MCMCglmm tutorial)

Thomas M. Houslay & Alastair J. Wilson

9 January 2017

Introduction

Overview

This tutorial accompanies our 2017 Behavioral Ecology paper, "Avoiding the misuse of BLUP in behavioral ecology". Here, we focus on **individual variation in plasticity**, studying (behavioural) traits measured repeatedly across some environmental gradient. We will:

- Use random regression to test for individual variation in plasticity under a reaction norm perspective;
- Add an additional response variable to the random regression, and test for an association between individual variation in intercepts/slopes and, for example, a single measure of lifetime fitness;
- Show how 'character state' models, set up in terms of environment-specific subtraits, can equally be used to model individual variation in plasticity;
- Add the fitness measure to the character state model to demonstrate why thinking about how selection acts on a trait in different environments may sometimes be a more intuitive way of looking at things than trying to estimate selection on "plasticity" (although both perspectives are valid).

In this version of the tutorial, we illustrate these models using the `R` package `MCMCglmm`, developed by Jarrod Hadfield. Visit the CRAN page for `MCMCglmm` here for links and citation info: https://cran.r-project.org/web/packages/MCMCglmm/index.html.

MCMCglmm fits generalised linear mixed modes (GLMMs) in a Bayesian framework, using Markov chain Monte Carlo techniques. We have also provided a separate tutorial for the `R` interface for `ASReml`, which fits GLMMs using maximum likelihood (and so is likely more familiar to `lme4` users) but is commercially licensed software.

Updates, data sets and further tutorials associated with this paper can be found at https://tomhouslay.com/tutorials/.

Aims

As with the previous tutorial ('I. Multivariate modelling for individual variation'), we assume readers are familiar with the general principles of specifying univariate mixed effects models, and in particular with the use of `MCMCglmm` for univariate mixed effects models. Readers unfamiliar with `MCMCglmm` should look at Jarrod Hadfield's excellent course notes, available at the MCMCglmm CRAN page.

There are various papers that cover the fundamentals of mixed models for individual variation in behavioural plasticity, and we recommend Dingemanse *et al* (2010) 'Behavioural reaction norms: animal personality meets individual plasticity' and Brommer (2013) 'Variation in plasticity of personality traits implies that the ranking of personality measures changes between environmental contexts: calculating the cross-environmental correlation'. We also provide further citations later in the tutorial for more specific topics.

We also use various methods for manipulating and visualising data frames using the `tidyverse` package (including `tidyr`, `dplyr`, `ggplot2` etc) — more details on their use can be found at http://r4ds.had.co.nz/.

In our tutorial, we aim to teach the following:

- Specifying, fitting and interpreting a random regression model;
- Adding a second response variable to a random regression model;
- Restructuring data in a format suitable for character state models;
- Fitting and interpreting character state models;
- Interpreting MCMC credible intervals.

Packages required

There are several packages that you must have installed in `R` prior to starting this tutorial:

- `MCMCglmm`
- `lme4`
- `nadiv`
- `tidyverse`
- `broom`
- `gridExtra`

Behavioural plasticity

In this section, we will look at how one can test for and estimate individual variation in behavioural plasticity — which is also known as individual by environment interaction (IxE). A common approach to studying IxE is to utilise reaction norms which are typically assumed to be linear (first order function of the environmental variable, E). Under this perspective IxE is present if individuals differ in the slope of their phenotypic relationship to E. This variation in slopes (i.e. plasticities) can be modelled using random regression mixed models.

Individual variation in plasticity

As in our previous tutorial, we use wild haggis (*Haggis scoticus*) as our study organism.

Here, we have measured aggression in another population of wild haggis (using individually tagged males). Male haggis are territorial, and previous studies have suggested that aggressive behaviour tends to increase with the size of a rival male. For this experiment, we used model 'intruders' to test for individual variation in aggressive behaviour. These intruders were made up of 3 size classes: average male haggis size (calculated as the population mean), 1 standard deviation below the population mean, and 1 standard deviation above. We measured each focal male against each model size, and repeated this in 2 blocks (with test order randomised). The body size of the focal individual was measured at the beginning of each block. We also have a fitness proxy (a single measure of mating success for the season) for each male.

We want to test whether individuals vary in the extent of their aggression towards intruders, whether there really is a dependence of aggression on intruder size (on average), and – if so – whether the plastic response to intruder size varies among individuals (IxE). Finally, we want to test whether behavioural variation is associated with fitness as we'd expect if aggression is under natural selection.

Load libraries and inspect data

Figure 1: A male haggis in the wild (thanks to Emma Wood, http://www.ewood-art.co.uk/)

```
library(lme4)
library(MCMCglmm)
library(tidyverse)
library(broom)
library(nadiv)
library(gridExtra)
library(lattice)
```

```
df_plast <- read_csv("aggression.csv")
```

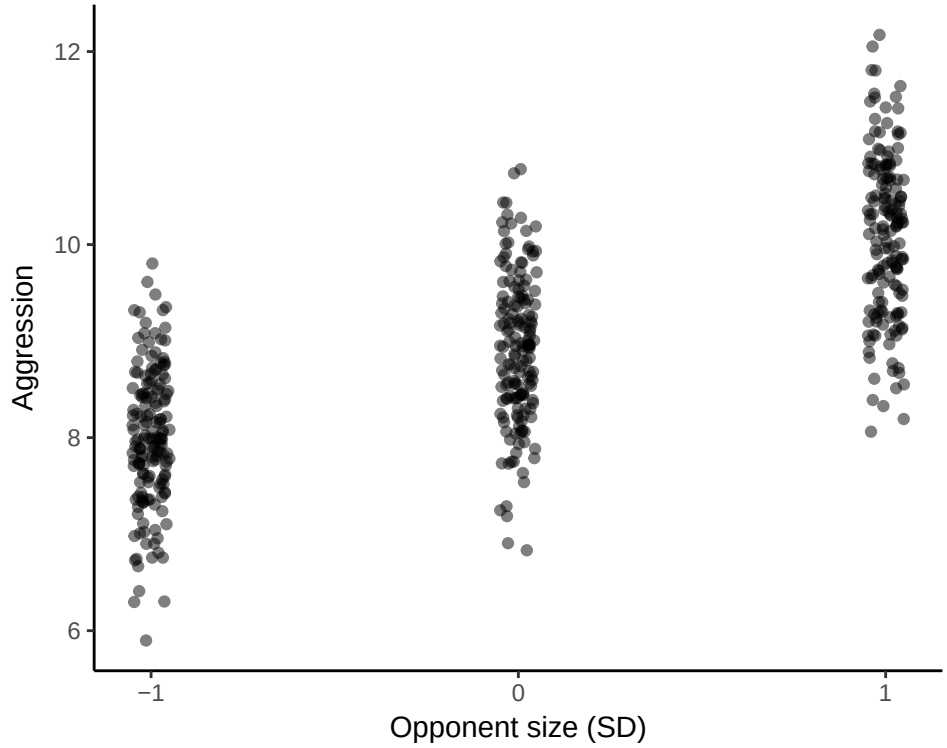
This data frame has 6 variables:

- Individual **ID**
- Experimental **Block**
- Individual **body_size**, as measured for each block
- The repeat number for each behavioural test, **assay_rep**
- Opponent size (**opp_size**), in standard deviations from the mean (i.e., -1,0,1)
- **aggression**, measured 6 times in total per individual (2 blocks of 3 tests)
- **fitness**, our measure of mating success, with a single value for each individual

As in the previous tutorial, we want to convert fitness to **relative fitness** by dividing each individual's fitness value by the mean population fitness. We will not be using the raw fitness value at all, so we can simply overwite it in our data frame with this standardised version (of course, we do not change the csv file in which our collected data is stored, so we can always get the raw values again if needed):

```
df_plast <- df_plast %>%
  mutate(fitness = fitness/mean(fitness, na.rm=TRUE))
```

An initial plot of the phenotypic data indicates that - as expected - there is a general increase in aggression with opponent size (points are lightly jittered on the x-axis to show the spread of data a little better):



From the above plot, and also from a quick look at the population mean for aggression at each opponent size (below), we expect that our models should show a roughly linear increase in aggression (1 unit increase per standard deviation of opponent size).

opp_size	mean_aggr
-1	8.00
0	8.91
1	10.09

We will begin this tutorial by running through some quick models in `lme4`, which is probably familiar to most readers, before moving on to more complex methods with `MCMCglmm`.

Random intercepts model (lme4)

We can quickly run a mixed model with only random intercepts, to see how it fits the data. Here we fit fixed effects of opponent size (our predictor of interest), focal male body size (mean-centred and scaled), assay repeat (mean-centred), and experimental block. As we only had 2 blocks, we fit this as a fixed rather than random effect. Our random effect here is individual ID. So here we allow random intercepts to vary among males, and we also allow population-level plasticity (change in mean aggression with opponent size), but we do not model any IxE:

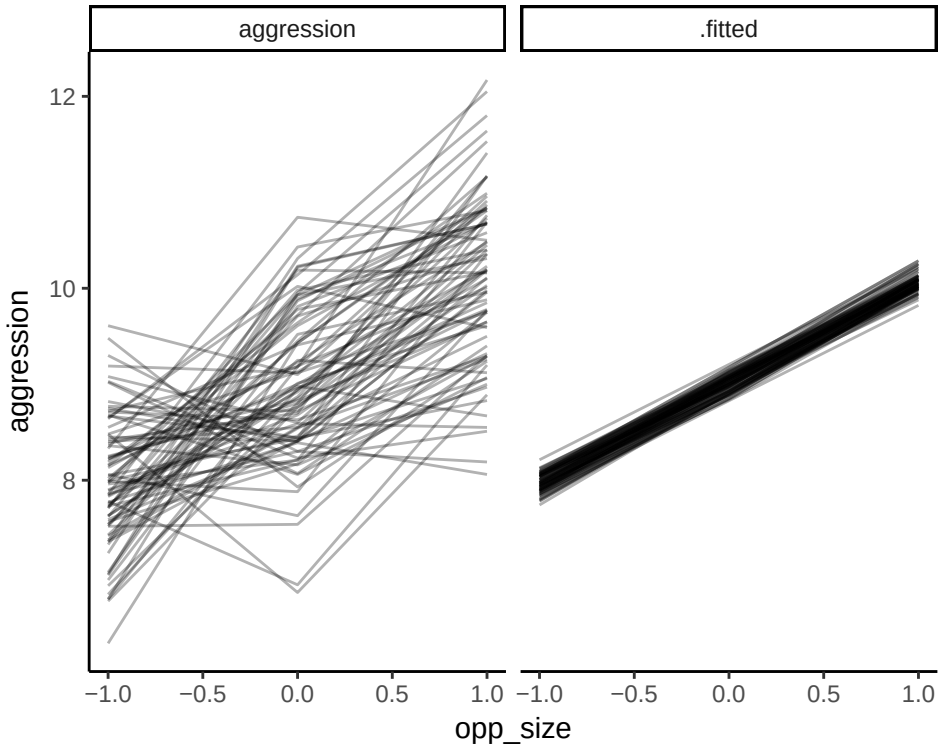
```
lmer_a <- lmer(aggression ~ opp_size + scale(body_size) +
                 scale(assay_rep, scale = FALSE) + block +
                  (1|ID),
                data = df_plast)
```

Diagnostic plots look fine:

```
plot(lmer_a)
qqnorm(residuals(lmer_a))
```

We then use the handy `augment` function from the `broom` package to get predictions for each of our observations. Below, we plot the raw data for each individual in one panel, with the fitted slopes in a second panel. Because we have 2 blocks as fixed effects, for ease of presentation we have selected only one of the blocks for this plot (if you like, you can check the other blocks to reassure yourself that there is little overall difference - another way would be to calculate predictions while averaging over block effects).

```
augment(lmer_a) %>%
  select(ID, block, opp_size, .fitted, aggression) %>%
  filter(block == -0.5) %>%
  gather(type, aggression,
         `.fitted`:aggression) %>%
  ggplot(., aes(x = opp_size, y = aggression, group = ID)) +
  geom_line(alpha = 0.3) +
  theme_classic() +
  facet_grid(.~type)
```



This illustrates the importance of using model predictions to see whether the model actually fits the individual-level data well or not — while the diagnostic plots looked fine, and the model captures mean plasticity, here

we can see that the model really doesn't fit the actual data very well at all. The code below provides a different (and slightly more in-depth) look at this same combination of fitted slope / real data, and indicates that the fitted slopes systematically under- (eg, `ID_42`, `ID_43`) and over-estimate (eg, `ID_5`, `ID_25`) plasticity in aggression at the individual level (we have not shown the figure as it consists of 80 panels!).

```
# Create 'dummy' data frame for prediction
# - all IDs, all opponent sizes,
# - block set at 0 (as blocks are coded as -0.5, 0.5)
# - mean body size and assay repeats
df_ri_ind <- expand(df_plast,
                    ID, opp_size,
                    block = 0,
                    body_size = mean(body_size),
                    assay_rep = mean(assay_rep))

# Get predicted values based on RR model and dummy data frame
#  and using random effects structure as in the model
df_ri_ind$fit <- predict(lmer_a, newdata = df_ri_ind, re.form = NULL)

# Plot predictions and overlay original data points
ggplot(df_ri_ind, aes(x = opp_size, y = fit, group = ID)) +
  geom_line() +
  geom_point(data = df_plast,
             aes(y = aggression),
             alpha = 0.3) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  theme_classic() +
  facet_wrap(~ID)
```

Now, let's move on and test whether random slopes provide a better fit to the data...

Random regression (lme4)

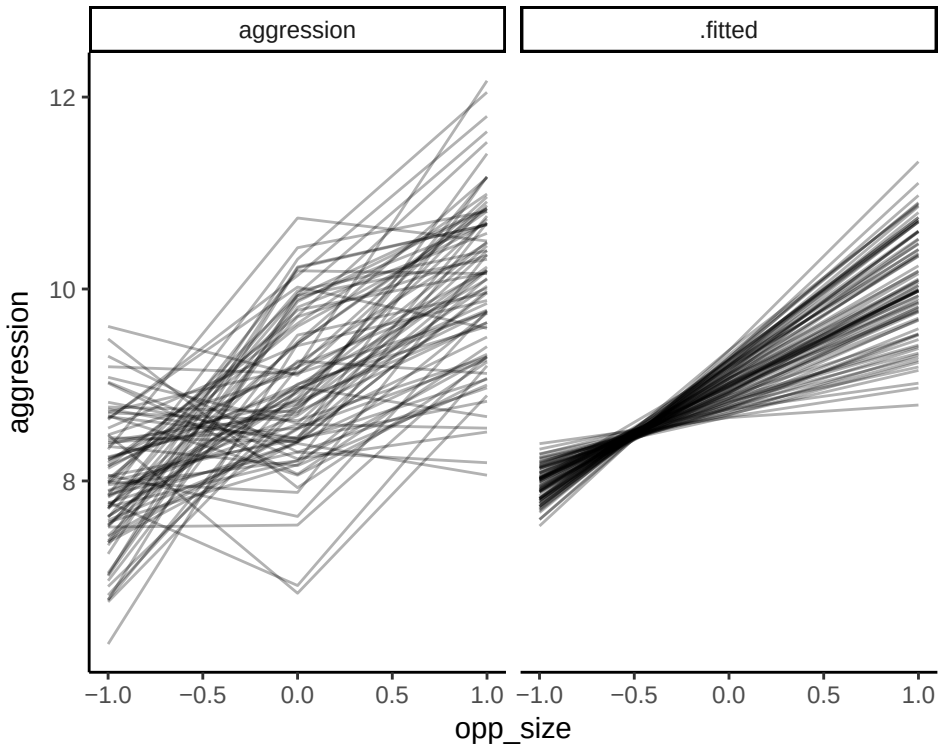
```
lmer_a_rr <- lmer(aggression ~ opp_size + scale(body_size) +
                  scale(assay_rep, scale = FALSE) + block +
                   (opp_size|ID),
                 data = df_plast)
```

```
plot(lmer_a_rr)
qqnorm(residuals(lmer_a_rr))
```

Diagnostic plots for the random regression model also look good — let's go ahead and check our predicted fit (again, looking at a single block - simply change the `filter` specification to check predictions with the other):

```
augment(lmer_a_rr) %>%
  select(ID, block, opp_size, .fitted, aggression) %>%
  filter(block == -0.5) %>%
  gather(type,aggression,
         `.fitted`:aggression) %>%
  ggplot(., aes(x = opp_size, y = aggression, group = ID)) +
  geom_line(alpha = 0.3) +
```

```
  theme_classic() +
  facet_grid(.~type)
```



This certainly seems better, as it is capturing both mean plasticity and variation at the individual level. Again, we have also provided code for a more in-depth look at individual predictions against the raw data:

```
# Create 'dummy' data frame for prediction
# - all IDs, all opponent sizes,
# - block set at 0 (as blocks are coded as -0.5, 0.5)
# - mean body size and assay repeats
df_rr_ind <- expand(df_plast,
                    ID, opp_size,
                    block = 0,
                    body_size = mean(body_size),
                    assay_rep = mean(assay_rep))

# Get predicted values based on RR model and dummy data frame
#  and using random effects structure as in the model
df_rr_ind$fit <- predict(lmer_a_rr, newdata = df_rr_ind, re.form = NULL)

# Plot predictions and overlay original data points
ggplot(df_rr_ind, aes(x = opp_size, y = fit, group = ID)) +
  geom_line() +
  geom_point(data = df_plast,
             aes(y = aggression),
             alpha = 0.3) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  theme_classic() +
  facet_wrap(~ID)
```

Compared to the individual-level plots of the random intercepts model, this appears to be doing a much better job of fitting the data.

We can test the improvement of the model fit using the overloaded `anova` function in R to perform a likelihood ratio test (LRT):

```
anova(lmer_a_rr, lmer_a)
```

```
## refitting model(s) with ML (instead of REML)
```

	Df	AIC	BIC	logLik	deviance	Chisq	Chi Df	Pr(>Chisq)
lmer_a	7	1128.709	1157.925	-557.3544	1114.709	NA	NA	NA
lmer_a_rr	9	1070.269	1107.833	-526.1345	1052.269	62.43989	2	0

We can see here that the LRT uses a chi-square test with 2 degrees of freedom, and indicates that the random slopes model shows a statistically significant improvement in model fit. The 2df are because there are two additional (co)variance terms estimated in the random regression model: a variance term for individual slopes, and the covariance (or correlation) between the slopes and intercepts. Let's look at those values, and also the fixed effects parameters, via the model summary:

```
summary(lmer_a_rr)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## aggression ~ opp_size + scale(body_size) + scale(assay_rep, scale = FALSE) +
##     block + (opp_size | ID)
##    Data: df_plast
##
## REML criterion at convergence: 1074.4
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.04930 -0.59780 -0.02004  0.59574  2.68010
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  ID       (Intercept) 0.05043  0.2246
##           opp_size    0.19166  0.4378   0.96
##  Residual             0.42817  0.6543
## Number of obs: 480, groups:  ID, 80
##
## Fixed effects:
##                                 Estimate Std. Error t value
## (Intercept)                      9.00181    0.03902  230.71
## opp_size                         1.05033    0.06123   17.15
## scale(body_size)                 0.02725    0.03377    0.81
## scale(assay_rep, scale = FALSE) -0.04702    0.03945   -1.19
## block                           -0.02169    0.05973   -0.36
##
## Correlation of Fixed Effects:
##             (Intr) opp_sz scl(_) s(_=F
```

```
## opp_size     0.495
## scl(bdy_sz)  0.000  0.000
## s(_,s=FALSE  0.000 -0.064 -0.006
## block        0.000  0.000  0.002  0.000
```

From the fixed effects, we see that there is a strong effect of opponent size on aggression (1.05 SE 0.06, t = 17.15) — so there is population-level plasticity at a rate anticipated from our earlier plots of raw data, whereby aggression increases (on average) at a rate of 1 unit per standard deviation increase in opponent size. Significance of fixed effects can be determined using likelihood ratio tests (you should find that testing `opp_size` returns $P<0.001$). The other fixed effects in this model — individual body size, the order of assays, and block effects — are small and non-significant.

Interpreting the random effects requires some care. Remember that the intercept variance is the among-individual variance in aggression where x (here, opponent size) = 0. Here this corresponds to an average sized opponent because of the way we scaled the environmental covariate. This is very important! **A different scaling would change the number (but not the model predictions!)**. There is also a highly positive correlation between intercepts and slopes **at the intercept x = 0, i.e., where opponent size is at the mean population value**. This positive correlation indicates that individuals with more positive intercept deviations also have more positive slope deviations **at x = 0**.

When looking at our (above) plot of predicted individual slopes, it is clear that the intercept-slope correlation is highly dependent on the positioning of the intercept. If the intercept was positioned elsewhere then we might get a very different intercept-slope correlation. This is absolutely as it should be — for example, if we had called our opponent sizes 1:3 then x=0 would be off to the far left, and the intercept-slope correlation would be closer to -1 as it would be the least positive intercept deviations that had the most positive slope deviations at that value of x. Failure to understand the dependence of the random effect parameters on the scaling and centring of x means a high risk of interpreting the biological meaning of your model incorrectly.

Similarly, if there is IxE we have to be careful about drawing conclusions from the among-individual variation in intercepts in our model summary. To reiterate, it is the variance among-individuals at x=0, but in the presence of IxE (i.e., individual slopes are allowed to vary) then the among-individual variation changes as a function of opponent size. **So, it is not correct to view variation in intercepts as behavioural variation that is "independent of the environment" as is sometimes stated**. Furthermore, consider again the hypothetical situation above where we coded our opponent sizes as 1:3 (rather than -1:1). In such a situation, the variation in intercepts would still be calculated at x = 0, so not even in the range of where we collected the data!

From the above plots, it is clear that (in the case of our data) here the among-individual variation in aggression is greater at large opponent size than at small opponent size. Later in the tutorial we will look at different methods that enable us to investigate changes in among-individual variance more explicitly. A more thorough discussion of such issues when modelling IxE can be found in Nussey *et al* (2007) 'The evolutionary ecology of individual phenotypic plasticity in wild populations'.

This section was simply a quick reminder of how a random regression works – and some important notes on interpretation – using what's probably a familiar setup. For more advanced modelling techniques we will use `MCMCglmm`, so let's start by seeing how to run this same random regression model using that.

Random regression (MCMCglmm)

Specifying a random regression (also known as 'random slopes' models) in `MCMCglmm` is very similar to in `lme4`, but the `random` term is slightly more involved because it can be used to fit different covariance structures for the random effects.

To fit a random regression, in the `random` effects structure we use the code `random =~ us(1 + opp_size):ID`. This means that we interact `ID`, the grouping variable, with a covariance matrix of effects: intercepts and slopes for opponent sizes. The `us` keyword indicates that we want to fit an unstructured covariance matrix

for these effects (i.e., we fit variance in intercepts, variance in slopes, and the covariance between them), and interacting it with `ID` shows that we want random intercepts and random slopes for individuals. We do not need to set up any specific structure for the residuals, so these are simply modelled as `rcov =~ units`.

We do need to specify a **prior** for our model – we recommend reading up on the use of priors, but (briefly) we use a parameter-expanded prior here that should be uninformative for our model. One of the model diagnostic steps that should be used later is to check that the model is robust to multiple prior specifications.

Other parts of the model specification should be familiar if you have been through the first of our `MCMCglmm` tutorials already. Briefly, we provide the name of the object we set up as the model prior, and values for the total number of iterations (`nitt`), the 'burn-in' of initial iterations to be discarded as the model starts to converge (`burnin`), and the number of iterations to discard in between successive stored samplles (`thin`, which helps to reduce autocorrelation in sampling). Finally, we provide the name of the data frame — we enclose this in the `as.data.frame` function as `MCMCglmm` does not work with the `tbl_df` format used in the **tidyverse** group of packages.

```
# Parameter-expanded prior should be uninformative
#  for variances and covariance
prior_RR <- list(R = list(V = 1, nu = 0.002),
                 G = list(G1 = list(V = diag(2), nu = 2,
                                    alpha.mu = rep(0, 2),
                                    alpha.V= diag(25^2, 2, 2))))


mcmc_A_RR <- MCMCglmm(aggression ~ opp_size +
                        scale(assay_rep, scale=FALSE) +
                        scale(body_size) +
                        block,
                      random =~ us(1 + opp_size):ID,
                      rcov =~ units,
                      family = "gaussian",
                      prior = prior_RR,
                      nitt=750000,
                      burnin=50000,
                      thin=350,
                      verbose = TRUE,
                      data = as.data.frame(df_plast),
                      pr = TRUE,
                      saveX = TRUE, saveZ = TRUE)
```

After the model has been fit by `MCMCglmm` (which will take some time!), we can check some model diagnostics using plots of the MCMC samples. The following code allows you to look at the trace/density plots of the posterior distributions for the (co)variances; we don't show these here as there are a number of panels, but they should indicates no obvious problems (these plots are also available for fixed effects, using `Sol`):

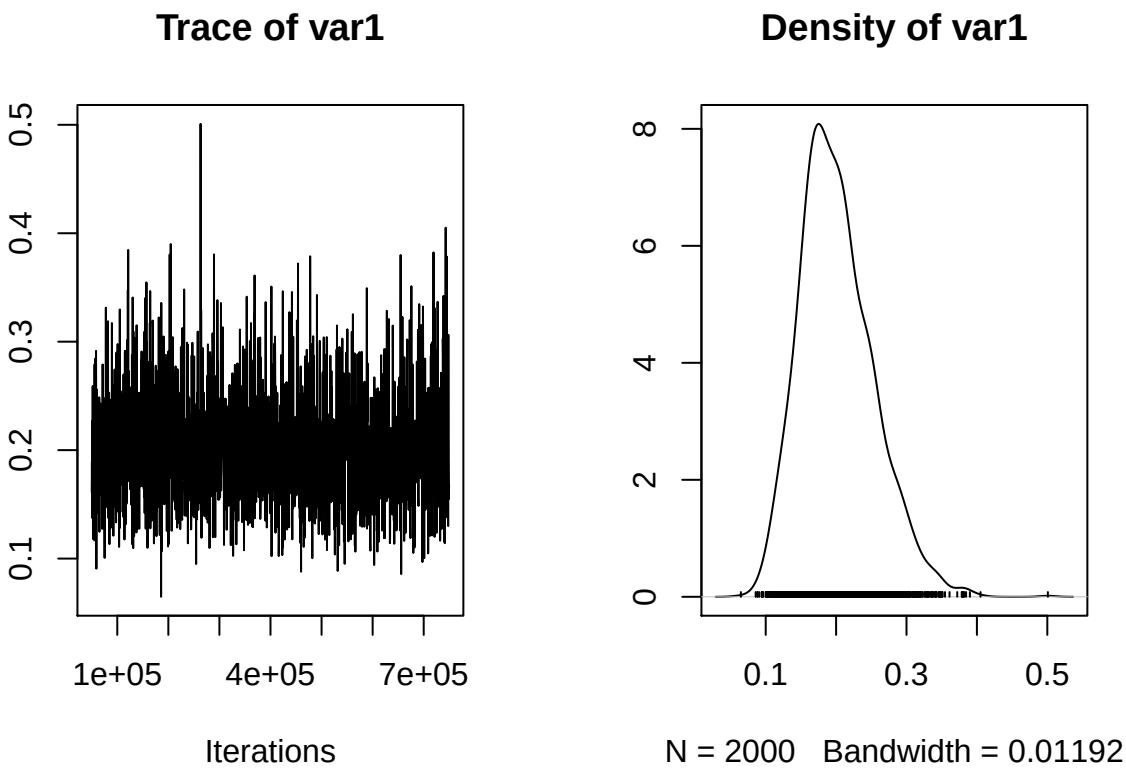
```
plot(mcmc_A_RR$VCV)
```

As noted in the first tutorial, for current purposes these are fine to proceed with (assuming you have used our simulated data and the settings above). Note, however, that for any real analysis then various other checks and tests (e.g. of autocorrelation, robustness to different priors, and good model convergence using the `geweke.diag` and `gelman.diag` diagnostic functions) should be used before accepting final results.

Given that a random regression allows IxE, we want to investigate whether there is support for the hypothesis that individuals vary in the slope of aggression against opponent size. For fixed effects, statistical significance of a variable can be determined by evaluating simply whether its 95% credible intervals cross zero (given that

your model stands up to scrutiny of the diagnostic checks mentioned above!). However, things are slightly trickier when we come to the random effects, as the posterior distribution for variance components should never include zero. Let's take a look at the trace and density plots for the posterior distribution of the among-individual variation in slopes term:

```
plot(mcmc_A_RR$VCV[,"opp_size:opp_size.ID"])
```




```
mean(mcmc_A_RR$VCV[,"opp_size:opp_size.ID"])
```

```
## [1] 0.2016104
```

```
HPDinterval(mcmc_A_RR$VCV[,"opp_size:opp_size.ID"])
```

```
##          lower     upper
## var1 0.1097826 0.3047987
## attr(,"Probability")
## [1] 0.95
```

The posterior distribution for slope variance looks good, and the credible intervals show that the lower bound is not close to zero (although bear in mind that this isn't always so easy – given that this distribution should not include zero, care must be taken not to automatically dismiss variance that is significant but small...). Unlike `lme4` and `ASReml`, `MCMCglmm` does not include formal model comparison tools, which can make evaluating competing models difficult for those of us used to P-values! Let's also fit a random intercepts model, and then we can use another couple of techniques for comparing model fits.

```
# Parameter-expanded prior for only a single random effect
prior_RI <- list(R = list(V = 1, nu = 0.002),
                 G = list(G1 = list(V = 1, nu = 1,
                                    alpha.mu = 0,
                                    alpha.V = 25^2)))

mcmc_A_RI <- MCMCglmm(aggression ~ opp_size +
                        scale(assay_rep, scale=FALSE) +
                        scale(body_size) +
                        block,
                      random =~ ID,
                      rcov =~ units,
                      family = "gaussian",
                      prior = prior_RI,
                      nitt=750000,
                      burnin=50000,
                      thin=350,
                      verbose = TRUE,
                      data = as.data.frame(df_plast),
                      pr = TRUE,
                      saveX = TRUE, saveZ = TRUE)
```

First off, it's worth looking at diagnostic plots to check model convergence etc here before we start thinking about comparing our two different models.

While `MCMCglmm` does provide DIC (Deviance Information Criterion) for model fits, the author of the package has noted that this should not be used in formal model testing (see mailing list thread here). We might use this as an informal guide, however, along with looking at the credible intervals and model predictions. Here we see that the random slopes model has a much lower DIC score, indicating that it provides a better fit to the data (even after penalising the additional parameters):

```
mcmc_A_RI$DIC
```

```
## [1] 1126.747
```

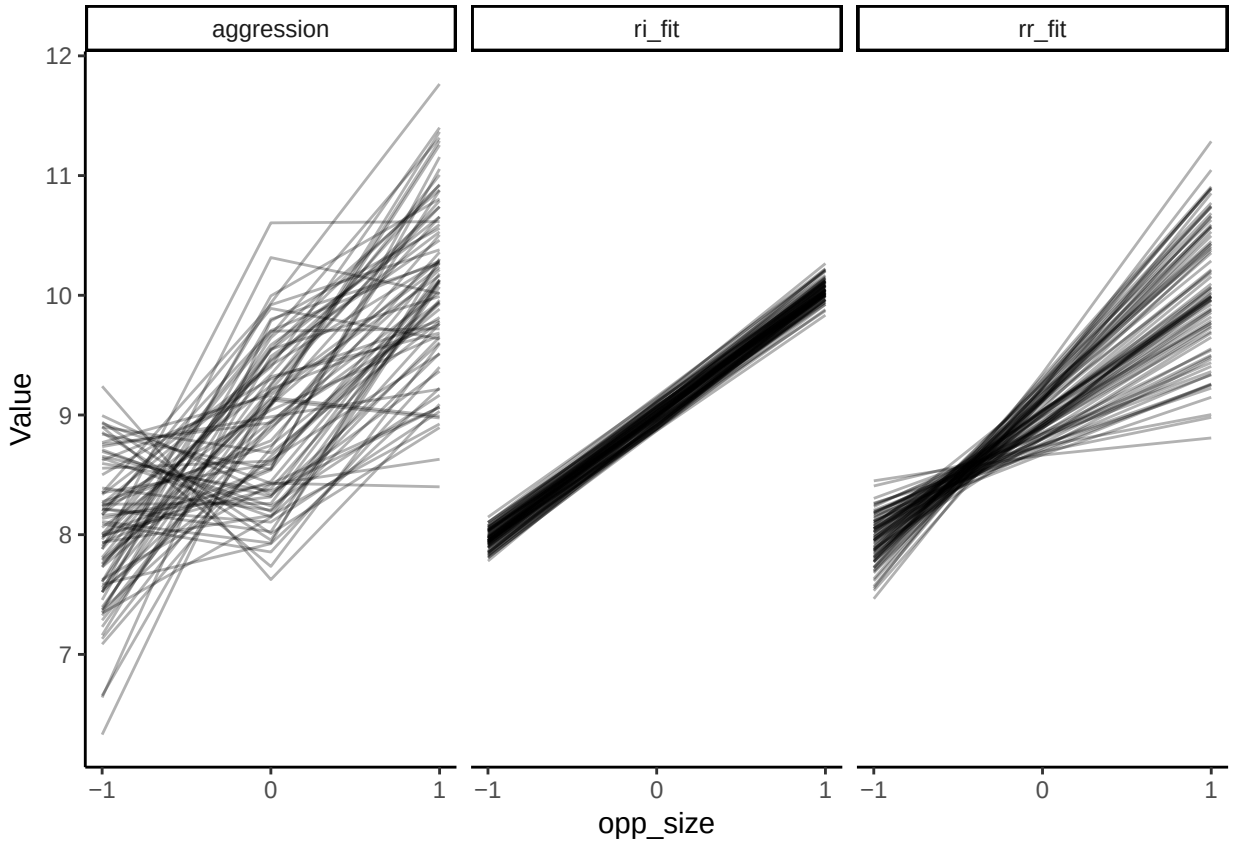
```
mcmc_A_RR$DIC
```

```
## [1] 1025.653
```

A very important part of assessing model fit is to look at how well the different models actually fit the data (in this case, with a particular focus on what's going on at the individual level). Here we'll quickly create a data frame that adds predictions from both the random intercepts and random regression models to the original data. Ideally we would like to have predicted where block = 0, but for this quick look we can simply take the mean of the predicted values for each block at each opponent size. Let's take a side-by-side look at the data, predictions from the random intercepts model, and predictions from the random regression model:

```
# Bind predictions of the random intercepts model
#  and random regression model to the original data;
# Select variables of interest;
# Get mean values at each opp_size for each individual
#  (within each 'type' of data), so that we are
#  essentially averaging across block effects;
```

```
# Convert to a 'long' format for plotting.
df_rand <- cbind(df_plast,
                 ri_fit = predict(mcmc_A_RI, marginal = NULL),
                 rr_fit = predict(mcmc_A_RR, marginal = NULL)) %>%
  select(ID, opp_size, ri_fit, rr_fit, aggression) %>%
  group_by(ID, opp_size) %>%
  summarise(ri_fit = mean(ri_fit),
            rr_fit = mean(rr_fit),
            aggression = mean(aggression)) %>%
  gather(Type, Value,
         ri_fit:aggression)

# Plot separate panels for individual lines of each type
ggplot(df_rand, aes(x = opp_size, y = Value, group = ID)) +
  geom_line(alpha = 0.3) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  theme_classic() +
  facet_grid(.~Type)
```



These figures clearly illustrate the importance of using model predictions to see whether the model actually fits the individual-level data well or not. For the random intercepts model, while the diagnostic plots looked fine and the model captures mean plasticity (i.e., at the population-level), we can see that the model really doesn't fit the actual individual-level data very well at all. In contrast, the random regression model does a much better job of fitting the individual variation (in addition to capturing mean plasticity in the fixed effects).

Just as with the `lmer` models we looked at earlier, you can also look at panels of predictions and data for

each individual, using both models:

```
df_ri_ind <- cbind(df_plast,
                   fit = predict(mcmc_A_RI, marginal = NULL)) %>%
  select(ID, opp_size, fit) %>%
  group_by(ID, opp_size) %>%
  summarise(fit = mean(fit))

# Plot predictions and overlay original data points
ggplot(df_ri_ind, aes(x = opp_size, y = fit, group = ID)) +
  geom_line() +
  geom_point(data = df_plast,
             aes(y = aggression),
             alpha = 0.3) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  theme_classic() +
  facet_wrap(~ID)
```

```
df_rr_ind <- cbind(df_plast,
                   fit = predict(mcmc_A_RR, marginal = NULL)) %>%
  select(ID, opp_size, fit) %>%
  group_by(ID, opp_size) %>%
  summarise(fit = mean(fit))

# Plot predictions and overlay original data points
ggplot(df_rr_ind, aes(x = opp_size, y = fit, group = ID)) +
  geom_line() +
  geom_point(data = df_plast,
             aes(y = aggression),
             alpha = 0.3) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  theme_classic() +
  facet_wrap(~ID)
```

Overall, our various checks indicate that the random slopes model provides a big improvement in model fit compared to the random intercepts model. Let's take a look at the model summary, and how to interpret the output of a `MCMCglmm` random regression model:

```
summary(mcmc_A_RR)
```

```
##
##  Iterations = 50001:749651
##  Thinning interval  = 350
##  Sample size  = 2000
##
##  DIC: 1025.653
##
##  G-structure:  ~us(1 + opp_size):ID
##
##                             post.mean l-95% CI u-95% CI eff.samp
## (Intercept):(Intercept).ID   0.05778  0.02371   0.1001     2000
## opp_size:(Intercept).ID      0.08460  0.03893   0.1305     2000
## (Intercept):opp_size.ID      0.08460  0.03893   0.1305     2000
```

```
## opp_size:opp_size.ID         0.20161  0.10978   0.3048     2059
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.4241   0.3641   0.4865     2000
##
##  Location effects: aggression ~ opp_size + scale(assay_rep, scale = FALSE) + scale(body_size) + bloc
##
##                                 post.mean l-95% CI u-95% CI eff.samp
## (Intercept)                       9.00246  8.92611  9.08057     2000
## opp_size                          1.05098  0.93574  1.17227     2000
## scale(assay_rep, scale = FALSE)  -0.04732 -0.12315  0.03023     2000
## scale(body_size)                  0.02874 -0.03862  0.09883     2000
## block                            -0.02167 -0.13497  0.09947     2000
##                                  pMCMC
## (Intercept)                      <5e-04 ***
## opp_size                         <5e-04 ***
## scale(assay_rep, scale = FALSE)  0.251
## scale(body_size)                 0.418
## block                            0.728
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Taking the `Location effects` first, just as with the `lmer` fits then the fixed effects show a significant effect of opponent size on aggression (the 95% credible intervals clearly distinct from zero) — so there is population-level plasticity at a rate anticipated from our earlier plots of raw data, whereby aggression increases (on average) at a rate of just over 1 unit per standard deviation increase in opponent size. Significance of fixed effects can be determined from the MCMC credible intervals, and in the fixed effects you also get a pMCMC value. The other fixed effects in this model — individual body size, the order of assays, and block effects — are small and non-significant.

The `G-structure` part gives us information about the random effects. The labels correspond to how we would set up a covariance matrix for the intercept and slope variance:

	Intercepts	Slopes
Intercepts	V	COV
Slopes	COV	V

. . . i.e., `(Intercept):(Intercept).ID` refers to the among-individual variation in intercepts, `opp_size:opp_size.ID` to the among-individual variation in slopes, and the other labels refer to the covariances (the off-diagonals in the above matrix). In addition to the estimate of the variance (here given as the mean of the posterior distribution), `MCMCglmm` (unlike in `lme4`) also provides us with a sense of the uncertainty around each estimate, in the form of the 95% credible intervals.

So, just as we found fitting the models with `lmer`, this result from `MCMCglmm` tells us there is among-individual variation in behavioural plasticity (IxE) — i.e., individuals change their aggression at different rates in response to opponent size.

For interpretation of the intercept-slope covariance, it is often easier to convert this to a correlation. Here we can use the formula for a correlation with the posterior distributions of our (co)variance components, giving us a distribution of correlation values that we can use to calculate estimates and 95% credible intervals:

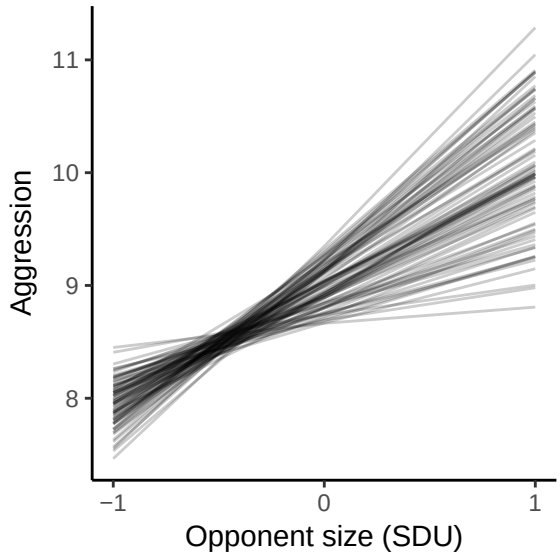
```
mcmc_cor_RR <- mcmc_A_RR$VCV[,"opp_size:(Intercept).ID"]/
  (sqrt(mcmc_A_RR$VCV[,"(Intercept):(Intercept).ID"])*
     sqrt(mcmc_A_RR$VCV[,"opp_size:opp_size.ID"]))

posterior.mode(mcmc_cor_RR)
```

```
##      var1
## 0.8264322
```

```
HPDinterval(mcmc_cor_RR)
```

```
##          lower     upper
## var1 0.5710141 0.9956851
## attr(,"Probability")
## [1] 0.95
```

Just as in the `lmer` model, we find a strong positive correlation between among-individual variance in intercepts and slopes, **at the intercept (x = 0)**. This is a good time to revisit the earlier notes (at the end of the `lme4` section) about how we need to take a lot of care in how we interpret the variance components of a random regression model, particularly because of the positioning of the intercept. It is particularly important to bear that in mind as we move on to the next stage, where we will add a second response variable to our random regression model.



Bivariate model with random regression on one trait

Having established that there is among-individual variation in behavioural plasticity, we want to now test whether there is an association between variation in aggression (intercept and/or slope) and 'fitness' (measured as mating success — recall also that we standardised this to relative fitness earlier in the tutorial).

In this example, we have two response variables, but only one of them has repeated observations at the individual level and is a function of the x variable (opponent size). We can set up the covariance matrix such that we estimate random intercepts of both traits, and random slopes for trait 1 only:

	$T1_{int}$	$T2_{int}$	$T1_{slope}$
$T1_{int}$	V	COV	COV
$T2_{int}$	COV	V	COV
$T1_{slope}$	COV	COV	V

In addition, as fitness is measured only once then we need to constrain the residual/'within-individual' variance such that it is effectively zero (i.e., meaning that all the variance in fitness will be in the among-individual level). We can do this in the `R` section of the prior, using the `fix` keyword. Note that the value of 0.0001 is adequate here – a smaller value, such as 1e-08, can cause problems with the chain mixing (here speaks the voice of bitter experience!). For the among-individual section, `G`, we use an uninformative parameter-expanded prior for a 3x3 covariance matrix:

```
prior_biv_RR_px  <-  list(R = list(V = diag(c(1,0.0001),2,2), nu = 0.002, fix = 2),
                          G = list(G1 = list(V = matrix(c(1,0,0,
                                                          0,1,0,
                                                          0,0,1),3,3,
                                                        byrow = TRUE),
                                             nu = 3,
                                             alpha.mu = rep(0,3),
                                             alpha.V = diag(25^2,3,3))))
```

Within the model specification itself, we standardise aggression by scaling it (to make the multivariate model easier to fit), and we have already standardised fitness as relative fitness earlier on.

The `at.level` keyword specifies fixed effects as relating to only one of our response variables — here, we have fixed effects of opponent size, block, repeat and body size for aggression (but not for fitness).

We set up our `random` effects in a similar way to the univariate random regression, but here we have 3 variance terms – random intercepts for both response traits, and random slopes for only the first response trait, aggression (we use the `at.level` keyword here just as we did in the fixed effects). We then fit residual variances for each of our response variables, but do not model the covariance between them (and remember that we have set residual variance in fitness to be essentially zero).

```
mcmc_biv_RR <- MCMCglmm(cbind(scale(aggression),
                              fitness) ~ trait-1 +
                          at.level(trait,1):opp_size +
                          at.level(trait,1):scale(assay_rep, scale=FALSE) +
                          at.level(trait,1):block +
                          at.level(trait,1):scale(body_size),
                        random =~ us(trait + opp_size:at.level(trait,1)):ID,
                        rcov =~ idh(trait):units,
                        family = c("gaussian","gaussian"),
                        prior = prior_biv_RR_px,
                        nitt=950000,
                        burnin=50000,
                        thin=450,
                        verbose = TRUE,
                        data = as.data.frame(df_plast),
                        pr = TRUE,
                        saveX = TRUE, saveZ = TRUE)

plot(mcmc_biv_RR$VCV)
```

Let's take a look at the among-individual variance components for this model:

```
summary(mcmc_biv_RR)$Gcovariances
```

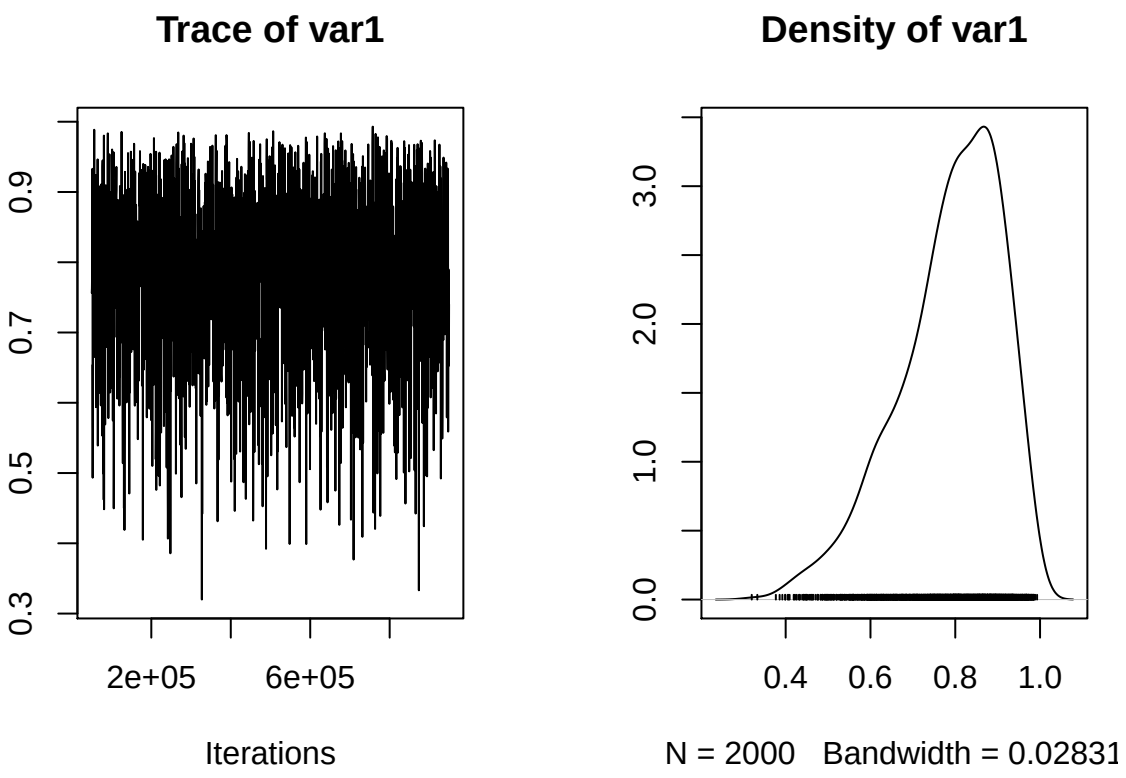
```
##                                                          post.mean
## traitaggression:traitaggression.ID                      0.04368629
## traitfitness:traitaggression.ID                         0.01577026
## opp_size:at.level(trait, 1):traitaggression.ID          0.06276634
## traitaggression:traitfitness.ID                         0.01577026
## traitfitness:traitfitness.ID                            0.02737424
## opp_size:at.level(trait, 1):traitfitness.ID             0.02855723
## traitaggression:opp_size:at.level(trait, 1).ID          0.06276634
## traitfitness:opp_size:at.level(trait, 1).ID             0.02855723
## opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID 0.15120918
##                                                            l-95% CI
## traitaggression:traitaggression.ID                      0.017661095
## traitfitness:traitaggression.ID                         0.004799398
## opp_size:at.level(trait, 1):traitaggression.ID          0.028965667
## traitaggression:traitfitness.ID                         0.004799398
## traitfitness:traitfitness.ID                            0.019536828
## opp_size:at.level(trait, 1):traitfitness.ID             0.010199159
## traitaggression:opp_size:at.level(trait, 1).ID          0.028965667
## traitfitness:opp_size:at.level(trait, 1).ID             0.010199159
## opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID 0.081258602
##                                                            u-95% CI
## traitaggression:traitaggression.ID                      0.07256290
## traitfitness:traitaggression.ID                         0.02764436
## opp_size:at.level(trait, 1):traitaggression.ID          0.09597746
## traitaggression:traitfitness.ID                         0.02764436
## traitfitness:traitfitness.ID                            0.03692282
## opp_size:at.level(trait, 1):traitfitness.ID             0.04769133
## traitaggression:opp_size:at.level(trait, 1).ID          0.09597746
## traitfitness:opp_size:at.level(trait, 1).ID             0.04769133
## opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID 0.22394616
##                                                          eff.samp
## traitaggression:traitaggression.ID                       1870.037
## traitfitness:traitaggression.ID                          2201.828
## opp_size:at.level(trait, 1):traitaggression.ID           1987.031
## traitaggression:traitfitness.ID                          2201.828
## traitfitness:traitfitness.ID                             2000.000
## opp_size:at.level(trait, 1):traitfitness.ID              2151.262
## traitaggression:opp_size:at.level(trait, 1).ID           1987.031
## traitfitness:opp_size:at.level(trait, 1).ID              2151.262
## opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID 2000.000
```

Just as with the random regression model, the summary of variance components gives us estimates (and upper and lower 95% CIs, as well as the effective sample size) of the among-individual variances and covariances that we could use to populate the 3x3 covariance matrix in which we are interested (with `traitaggression` being variance in intercepts for aggression, `opp_size:at.level(trait, 1)` the variance in opponent size-related slopes for aggression, and `traitfitness` the variance in intercepts for relative fitness). We can also find the residual covariances using the `Rcovariances` part of the `MCMCglmm` model summary – you should look at this to check that the residual fitness variance has been fixed at a very low number, as we set up in the prior.

A good 'sanity check' is to ensure that the among-individual correlation between intercepts and slopes for aggression is (approximately) the same as we estimated in our earlier univariate random regression model.

As we can see from our example matrix above, we should find the covariance between variance components 1 and 3 (denoted `opp_size:at.level(trait, 1):traitaggression.ID`), and then calculate the correlation by standardising this by the product of the square root of both variances:

```
mcmc_cor_RR_intslope <- mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):traitaggression.ID"]/
  (sqrt(mcmc_biv_RR$VCV[,"traitaggression:traitaggression.ID"])*
     sqrt(mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID"]))

plot(mcmc_cor_RR_intslope)
```




```
posterior.mode(mcmc_cor_RR_intslope)
```

```
##      var1
## 0.8656797
```

```
HPDinterval(mcmc_cor_RR_intslope)
```

```
##         lower    upper
## var1 0.550797 0.983601
## attr(,"Probability")
## [1] 0.95
```

This does look the same, which is as it should be!

Now, to the more interesting parts – determining the among-individual correlation between aggression and fitness.

The covariance between elements 1 and 2 (line 1 of the summary, `traitfitness:traitaggression.ID`) is the among-individual covariance between fitness and variation in **intercepts** for aggression — here we

convert it to a correlation, and we find it is strongly **positive**. Note that, unlike variances, covariances (and correlations) can take on positive and negative values, so we can use the 95% CIs to think about 'significance'.

```
mcmc_cor_RR_intfit <- mcmc_biv_RR$VCV[,"traitfitness:traitaggression.ID"]/
  (sqrt(mcmc_biv_RR$VCV[,"traitfitness:traitfitness.ID"])*
     sqrt(mcmc_biv_RR$VCV[,"traitaggression:traitaggression.ID"]))

posterior.mode(mcmc_cor_RR_intfit)
```

```
##      var1
## 0.4760913
```

```
HPDinterval(mcmc_cor_RR_intfit)
```

```
##          lower     upper
## var1 0.1750098 0.7085938
## attr(,"Probability")
## [1] 0.95
```

We also find a **positive** among-individual correlation between fitness and variation in **slopes** for aggression (converting the covariance, `opp_size:at.level(trait, 1):traitfitness.ID`, to a correlation).

```
mcmc_cor_RR_slopefit <- mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):traitfitness.ID"]/
  (sqrt(mcmc_biv_RR$VCV[,"traitfitness:traitfitness.ID"])*
     sqrt(mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID"]))

posterior.mode(mcmc_cor_RR_slopefit)
```

```
##      var1
## 0.4804545
```

```
HPDinterval(mcmc_cor_RR_slopefit)
```

```
##          lower    upper
## var1 0.2212651 0.666549
## attr(,"Probability")
## [1] 0.95
```

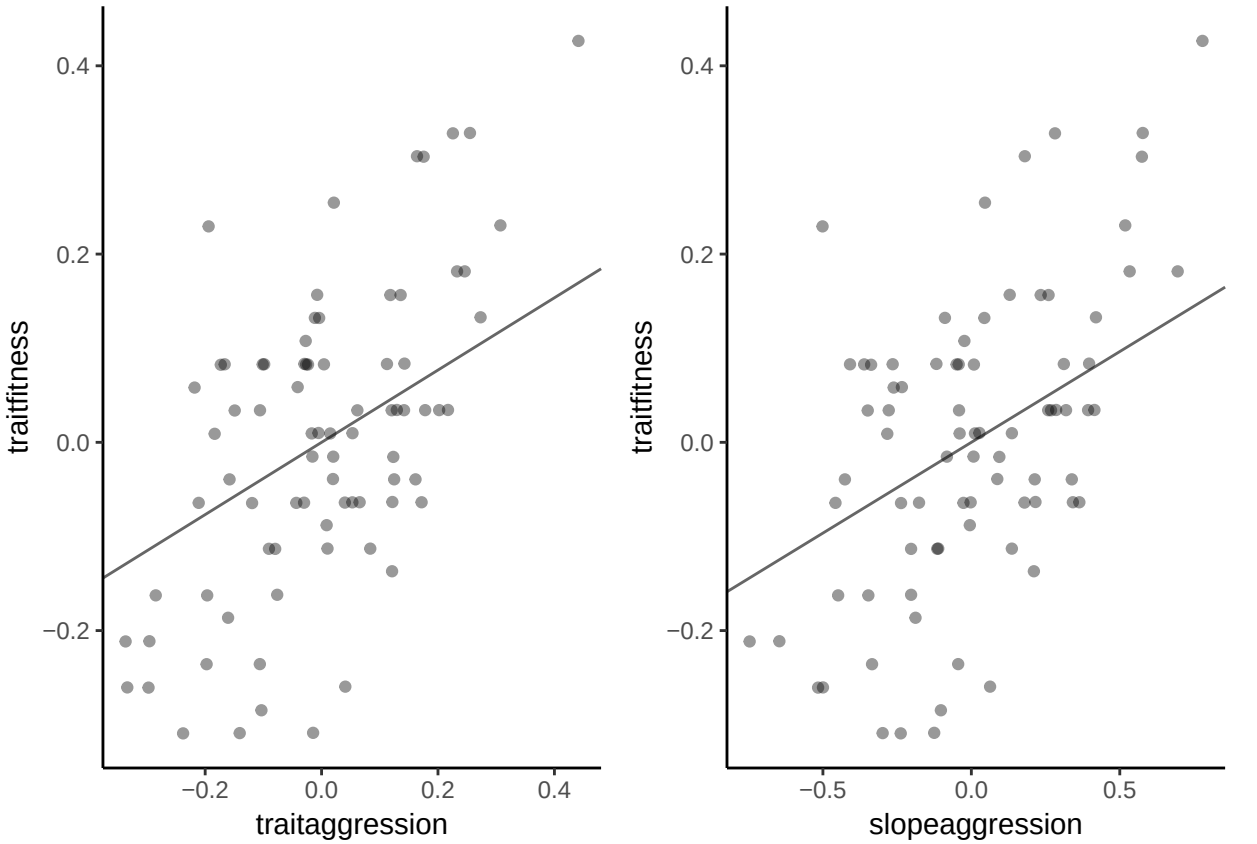
Again, recall that variation in intercepts is among-individual variance for aggression at x=0 in a random regression model, so the correlation between fitness and intercept is interpetable as the fitness-aggressiveness correlation in this 'environment'. It should not be too surprising that the intercept:fitness and slope:fitness correlations are so similar, because we have seen from our earlier univariate random regression models that **at x=0** the intercept:slope correlations are very high. The earlier plots of predictions from random regressions showed that those with higher intercepts at x=0 also tended to have higher slopes at that point.

So we do find a significant association between variation in aggression and in fitness. Below is the variance-covariance matrix (with variances on the diagonal, correlations above, and covariances below):

	Aggression	Fitness	Aggression:Opp
Aggression	0.04	0.48	0.87
Fitness	0.02	0.03	0.48
Aggression:Opp	0.06	0.03	0.15

We can also visualise this result by extracting the BLUPs for each individual and plotting them. Note that as in the previous tutorial, we think this is a reasonable use of BLUP. We are not running statistical analyses on them, we are using the BLUPs (ie the model predictions) only as a visual aid for the interpretation of a statistical model we have fitted. Remember that each of these points has a fair amount of uncertainty around it! We also use the (co)variance estimates from our model to calculate the regression slope directly from the bivariate random regression model:

```
# Get coefficients
df_biv_rr_coefs <- data_frame(Trait = attr(colMeans(mcmc_biv_RR$Sol), "names"),
                              Value = colMeans(mcmc_biv_RR$Sol)) %>%
  separate(Trait, c("tmp","Trait","Type","ID"), sep = "\\.", fill = "left") %>%
  filter(Type == "ID") %>%
  filter(Trait %in% c("traitaggression",
                      "traitfitness",
                      "level(trait, 1)")) %>%
  mutate(Trait = ifelse(Trait == "level(trait, 1)", "slopeaggression", Trait)) %>%
  select(ID, Trait, Value) %>%
  spread(Trait, Value)

# Calculate regression lines from the model fit (co)variances

ai_fit_slope <- mean(mcmc_biv_RR$VCV[,"traitaggression:traitfitness.ID"]/
                       mcmc_biv_RR$VCV[,"traitaggression:traitaggression.ID"])

as_fit_slope <- mean(mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):traitfitness.ID"]/
                       mcmc_biv_RR$VCV[,"opp_size:at.level(trait, 1):opp_size:at.level(trait, 1).ID"])

# Create plots of fitness values against BLUPs of:
# (i) aggression intercepts,
# (ii) aggression slopes

gg_ai_fit <- ggplot(df_biv_rr_coefs,
                    aes(x = traitaggression,
                        y = traitfitness)) +
  geom_abline(intercept = 0, slope = ai_fit_slope,
              colour = "grey40") +
  geom_point(alpha = 0.4) +
  theme_classic()

gg_as_fit <- ggplot(df_biv_rr_coefs,
                    aes(x = slopeaggression,
                        y = traitfitness)) +
  geom_abline(intercept = 0,
              slope = as_fit_slope,
              colour = "grey40") +
  geom_point(alpha = 0.4) +
  theme_classic()

grid.arrange(gg_ai_fit,
             gg_as_fit,
             ncol = 2)
```



We can see that those with higher average aggression (at the intercept) tend to have higher fitness, but also those that have higher slope values have higher fitness. These are often characterised as (for example) correlations between fitness and average aggression, and fitness and plasticity. However, interpreting (and generalising) this kind of result is actually more difficult than it seems. For instance, could we infer that plasticity is under selection? **Maybe, but not necessarily**. For example, we know that slopes and intercepts are strongly positively correlated (at x=0), so we would at least need to try and separate out the "direct" effects of each "trait" (i.e. intercept and slope) on fitness. This could be done, following the conceptual approach of classical selection analysis (e.g. Lande and Arnold 1983) by calculating the partial regressions (interpretable as selection gradients) on intercept and slope on fitness.

We also know that significant among-individual variation in slopes really means that the among-individual variation in aggression changes as a function of our x variable (here, opponent size): in other words, individual-by-environment interactions are occurring. Furthermore, we know that the intercept-fitness covariance could be used as a measure of selection on aggressiveness at x=0. So instead of trying to separate selection on the trait (aggressiveness) mean from selection on plasticity, an alternative way of thinking about this scenario is to ask **how fitness relates to aggression at different values of opponent size**.

In other words, how does selection on a trait change with environment? Getting a more concrete interpretation of such results is where random regression models start to fall down a little. However, if our environmental variable (here, opponent size) can be considered a set of distinct environments, we can move into an alternative (and in our view better for many scenarios) way of modelling IxE...

Character state models

In fact, reaction norms as modelled here by 'random regression' can be viewed as (normally) reduced parameter versions of what's called a **'character state'** model — for more details on this, we recommend the chapter 'Quantifying Genotype-by-Environment Interactions in Laboratory Systems' by Roff & Wilson in Hunt &

Hosken's 2014 textbook 'Genotype-by-Environment Interactions and Sexual Selection'. A character state approach posits that observations for each 'environment' (in this case, opponent size) are best modelled as distinct environment-specific sub-traits using multivariate analyses. With a lot of environments you have a lot of subtraits, so a lot of model parameters, and thus need a lot of data! This is one of the reasons why random regression is popular - you reduce a potentially infinite number of sub traits to a line described by just two parameters (intercept and slope). However, this saving comes at the cost of assuming a reaction norm shape (and if your assumptions are poor you cannot expect your model to fit well) and sometimes – as we have argued above – a lack of easy interpretation of your results.

Luckily, our experimental design here enables us to readily use a character state model. We have only 3 environments and plenty of data — moreover, because we have repeated measures within each 'environment', we can partition among- from within-individual (co)variation across each (see Brommer (2013) 'On between-individual and residual (co)variances in the study of animal personality' for further discussion of this topic).

We'll start by fitting just our aggression data in a character state model, meaning that we will estimate the among-individual variance in (and covariance between) each of our 'sub-traits': aggression against opponents 1SD below the mean body size, aggression against opponents of mean body size, and aggression against opponents 1SD above mean body size.

Modelling IxE for aggression

We need to do some rearranging of our data frame to structure it for this type of model. We want to end up with an individual's aggression values for each 'environment' (for a single block) on a single row in the data frame, with each 'sub-trait' in a different column (let's call these sub-traits `agg_S`, `agg_M`, and `agg_L`). Each row will also have the observation of block, and the opponent body size measurement for that block.

In this case, we have already done the 'data wrangling' for you, and provided the rearranged data frame in a new CSV file that we can load up straight away. However, at the end of this tutorial you will find the code for how we did this from the original data.

```
df_plast_CS <- read_csv("aggression_CS.csv")
```

Check this new data frame to understand the format and how it relates to our original data frame:

```
head(df_plast_CS)
```

```
## # A tibble: 6 × 8
##      ID block body_size agg_L agg_M agg_S opp_order  fitness
##   <chr> <dbl>     <dbl> <dbl> <dbl> <dbl>     <chr>    <dbl>
## 1  ID_1  -0.5     20.58 10.67 10.22  7.02     S_L_M 1.033846
## 2  ID_1   0.5     20.73 10.51  8.95  8.44     M_L_S       NA
## 3 ID_10  -0.5     28.32 10.81  9.43  7.73     L_S_M 0.960000
## 4 ID_10   0.5     28.80 10.67  9.46  8.08     L_S_M       NA
## 5 ID_11  -0.5     22.89  9.77  7.63  8.06     S_L_M 1.083077
## 6 ID_11   0.5     23.62 10.84  8.23  8.16     S_L_M       NA
```

We will first use a **trivariate model** to investigate (co)variances for our three 'sub-traits' (aggression for each opponent size class). We bind these as three response variables, interacting the `trait` keyword with the fixed effects of interest. Doing this means we estimate the effect of (for example) an individual's body size on aggression at each opponent size.

We use the `us` structure to estimate variances and covariances at both the among-individual (`random`) and within-individual/residual (`rcov`) level. We'll use an uninformative prior, parameter-expanded for the

among-individual variances, with a long run (`nitt`), lengthy `burnin` and with a large thinning interval (`thin`). Remember to check diagnostic plots once the model has finished (which will take quite some time!), and that for acceptance of any final results we would need to check convergence of multiple runs, robustness to different priors, and for the absence of autocorrelation among consecutive samples.

```
prior_triv_px  <-  list(R = list(V = diag(3), nu = 2.002),
                        G = list(G1 = list(V = matrix(c(1,0,0,
                                                        0,1,0,
                                                        0,0,1),3,3,
                                                      byrow = TRUE),
                                           nu = 3,
                                           alpha.mu = rep(0,3),
                                           alpha.V = diag(25^2,3,3))))

mcmc_triv_CS <- MCMCglmm(cbind(agg_S,
                           agg_M,
                           agg_L) ~ trait +
                            trait:opp_order +
                            trait:block +
                            trait:scale(body_size),
                         random =~ us(trait):ID,
                         rcov =~ us(trait):units,
                         family = c("gaussian","gaussian","gaussian"),
                         prior = prior_triv_px,
                         nitt=950000,
                         burnin=50000,
                         thin=450,
                         verbose = TRUE,
                         data = as.data.frame(df_plast_CS),
                         pr = TRUE,
                         saveX = TRUE, saveZ = TRUE)
```

The summary of fixed effects terms enable us to calculate the mean aggression for each trait: `(Intercept)` is aggression against small opponents, while `traitagg_M` is the average difference between aggression against small and medium opponents, and `traitagg_L` is the average difference between small and large. Aggression seems to increase in a roughly linear fashion across these three increasing opponent 'environments', which is what we expected from our initial plot of the data and from the random regression models. The pMCMC values for `traitagg_M` and `traitagg_L` show whether they are significantly different from the overall `(Intercept)`, i.e. aggression against small opponents. You could, of course, change the order of traits if you were particularly interested in this (e.g., with `agg_M` as the first trait such that you test for significant differences of both `agg_S` and `agg_L` from this). We haven't shown the results here as they are quite lengthy, but use the following code to look at the fixed effects output:

```
summary(mcmc_triv_CS)$solutions
```

You'll see that there are various values for order now because it is a many-level factor, with effects fitted for each of our response variables — we don't have a single p-value for this parameter, but we can see that (as expected) there do not seem to be any large or significant fixed effects.

Let's move on to the variance components, looking at both among- and within-individual (co)variances for all traits:

```
summary(mcmc_triv_CS)$Gcovariances
```

```
##                            post.mean    l-95% CI    u-95% CI eff.samp
## traitagg_S:traitagg_S.ID  0.1883781  0.06429334  0.31903840 2009.924
## traitagg_M:traitagg_S.ID -0.1423395 -0.23571596 -0.04153836 2000.000
## traitagg_L:traitagg_S.ID -0.1329159 -0.23755768 -0.02521122 2007.987
## traitagg_S:traitagg_M.ID -0.1423395 -0.23571596 -0.04153836 2000.000
## traitagg_M:traitagg_M.ID  0.2348929  0.09228423  0.38284429 2130.234
## traitagg_L:traitagg_M.ID  0.1478959  0.03029400  0.27338211 2000.000
## traitagg_S:traitagg_L.ID -0.1329159 -0.23755768 -0.02521122 2007.987
## traitagg_M:traitagg_L.ID  0.1478959  0.03029400  0.27338211 2000.000
## traitagg_L:traitagg_L.ID  0.2898116  0.10909500  0.48897790 2000.000
```

```
summary(mcmc_triv_CS)$Rcovariances
```

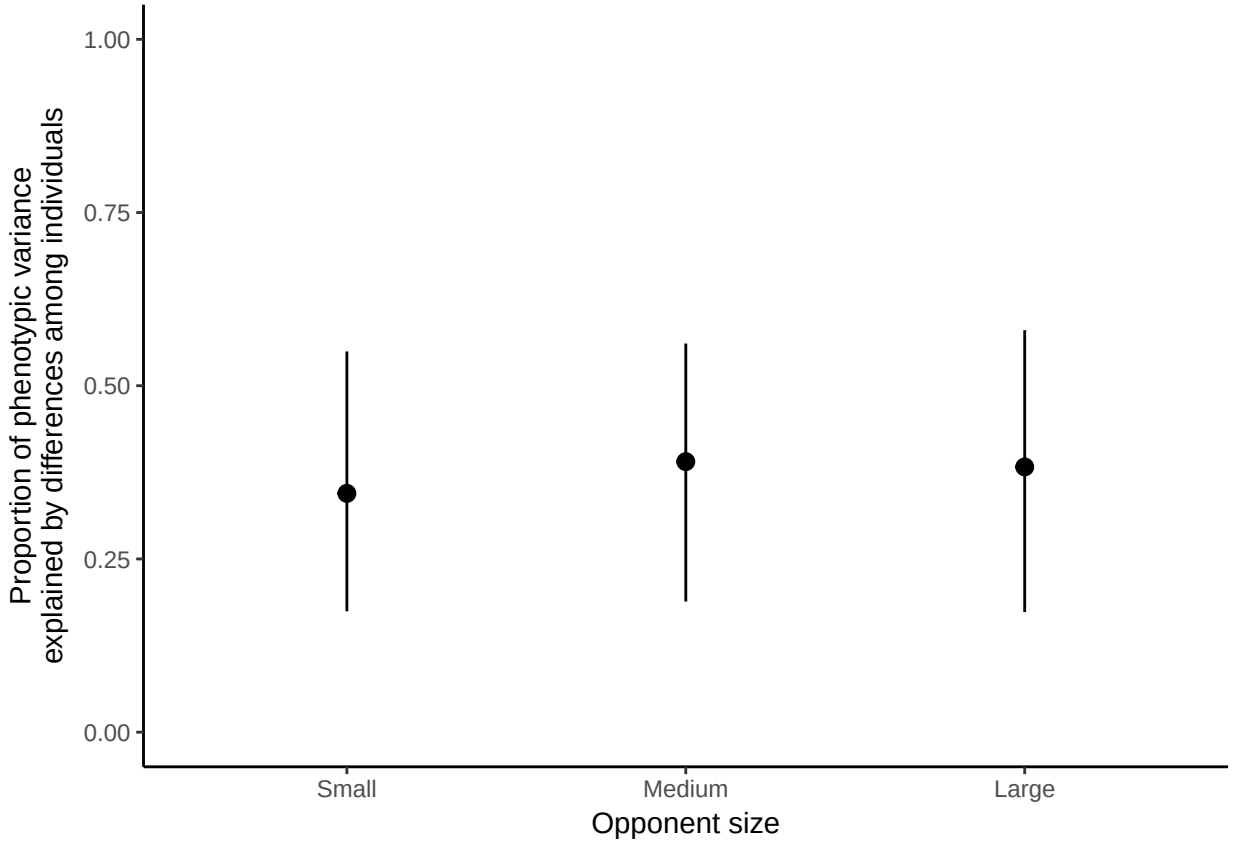
```
##                                post.mean    l-95% CI   u-95% CI eff.samp
## traitagg_S:traitagg_S.units  0.351255358  0.24941379 0.46239079 2000.000
## traitagg_M:traitagg_S.units -0.005402429 -0.09122089 0.06630601 1772.066
## traitagg_L:traitagg_S.units -0.020496590 -0.11329398 0.07275955 2371.019
## traitagg_S:traitagg_M.units -0.005402429 -0.09122089 0.06630601 1772.066
## traitagg_M:traitagg_M.units  0.359559838  0.24945395 0.47016333 1863.637
## traitagg_L:traitagg_M.units  0.166101952  0.07883660 0.27639930 1742.954
## traitagg_S:traitagg_L.units -0.020496590 -0.11329398 0.07275955 2371.019
## traitagg_M:traitagg_L.units  0.166101952  0.07883660 0.27639930 1742.954
## traitagg_L:traitagg_L.units  0.458199509  0.31572030 0.62196184 2000.000
```

Estimating repeatabilities of each trait

Another useful point of character state models is that we can estimate 'repeatability' for each trait (i.e., the proportion of phenotypic variation, conditional on the fixed effects, that is explained by differences among individuals). Similar to how we calculate the posterior distribution of correlations from (co)variances (to then get their estimate and 95% CIs), we can create posterior distributions of repeatabilities for each sub-trait:

```
mcmc_triv_rep_S <- mcmc_triv_CS$VCV[,"traitagg_S:traitagg_S.ID"]/
  (mcmc_triv_CS$VCV[,"traitagg_S:traitagg_S.ID"] +
     mcmc_triv_CS$VCV[,"traitagg_S:traitagg_S.units"])

mcmc_triv_rep_M <- mcmc_triv_CS$VCV[,"traitagg_M:traitagg_M.ID"]/
  (mcmc_triv_CS$VCV[,"traitagg_M:traitagg_M.ID"] +
     mcmc_triv_CS$VCV[,"traitagg_M:traitagg_M.units"])

mcmc_triv_rep_L <- mcmc_triv_CS$VCV[,"traitagg_L:traitagg_L.ID"]/
  (mcmc_triv_CS$VCV[,"traitagg_L:traitagg_L.ID"] +
     mcmc_triv_CS$VCV[,"traitagg_L:traitagg_L.units"])


df_mcmc_reps <- data_frame(Traits = c("Small",
                                      "Medium",
                                      "Large"),
                           Estimate = c(mean(mcmc_triv_rep_S),
                                        mean(mcmc_triv_rep_M),
                                        mean(mcmc_triv_rep_L)),
```

```
                          Lower = c(HPDinterval(mcmc_triv_rep_S)[,"lower"],
                                    HPDinterval(mcmc_triv_rep_M)[,"lower"],
                                    HPDinterval(mcmc_triv_rep_L)[,"lower"]),
                          Upper = c(HPDinterval(mcmc_triv_rep_S)[,"upper"],
                                    HPDinterval(mcmc_triv_rep_M)[,"upper"],
                                    HPDinterval(mcmc_triv_rep_L)[,"upper"]))

ggplot(df_mcmc_reps, aes(x = Traits, y = Estimate)) +
  geom_pointrange(aes(ymin = Lower,
                      ymax = Upper)) +
  scale_x_discrete(limits = c("Small","Medium","Large")) +
  labs(x = "Opponent size",
       y = "Proportion of phenotypic variance\nexplained by differences among individuals") +
  ylim(c(0,1)) +
  theme_classic()
```

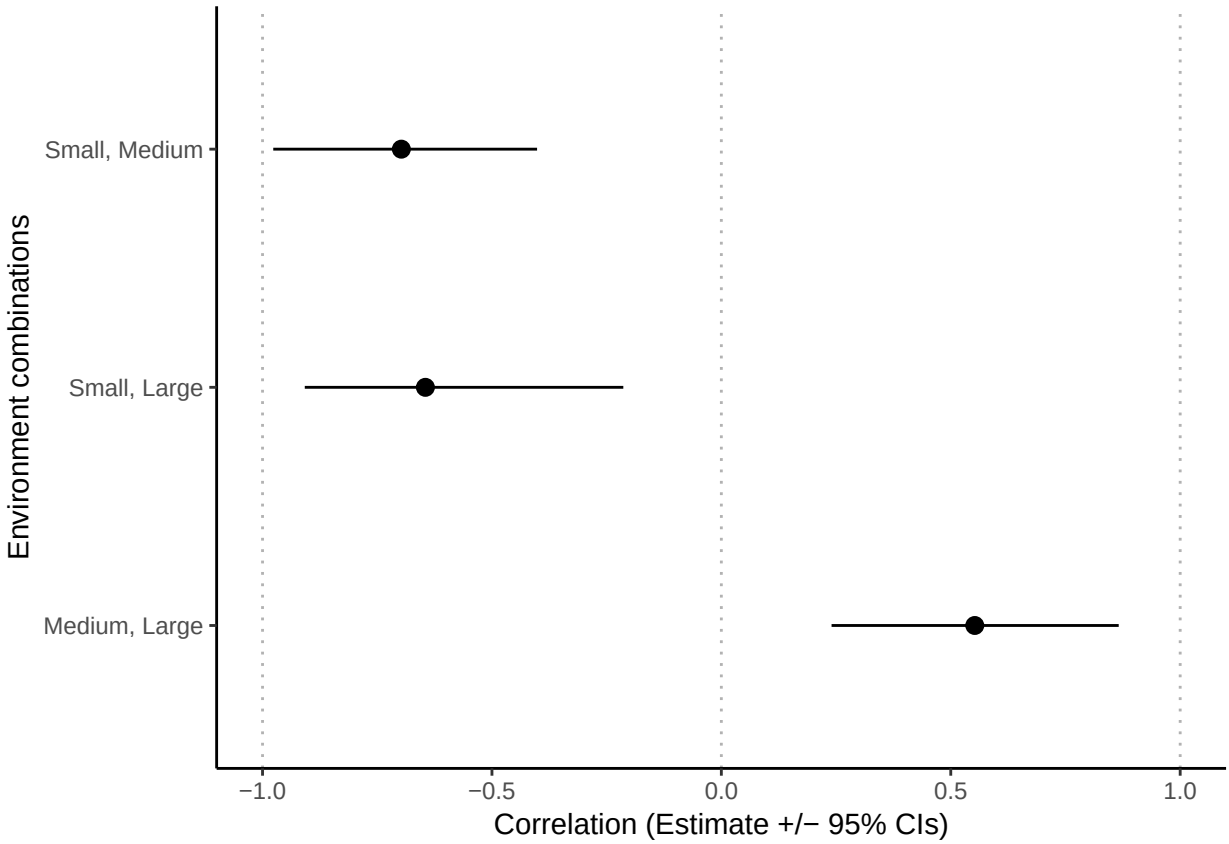


Here the three repeatabilities are quite similar. While we know that the credible intervals for variances will not include zero, we can see that the CIs for each repeatability are quite distinct from zero, and we can interpret these as 'significant' repeatabilities for aggression at each opponent size.

Cross-context correlations

Again, it can be easier to convert the covariances to correlations so that we can compare these 'standardised' associations directly – in this case, we'll calculate the posterior distribution of correlations for aggression across each pair of 'environments', and plot their estimate and associated 95% CIs:

```
mcmc_triv_cor_S_M <- mcmc_triv_CS$VCV[,"traitagg_S:traitagg_M.ID"]/
  (sqrt(mcmc_triv_CS$VCV[,"traitagg_S:traitagg_S.ID"])*
     sqrt(mcmc_triv_CS$VCV[,"traitagg_M:traitagg_M.ID"]))

mcmc_triv_cor_M_L <- mcmc_triv_CS$VCV[,"traitagg_M:traitagg_L.ID"]/
  (sqrt(mcmc_triv_CS$VCV[,"traitagg_M:traitagg_M.ID"])*
     sqrt(mcmc_triv_CS$VCV[,"traitagg_L:traitagg_L.ID"]))

mcmc_triv_cor_S_L <- mcmc_triv_CS$VCV[,"traitagg_L:traitagg_S.ID"]/
  (sqrt(mcmc_triv_CS$VCV[,"traitagg_L:traitagg_L.ID"])*
     sqrt(mcmc_triv_CS$VCV[,"traitagg_S:traitagg_S.ID"]))


df_mcmc_cors <- data_frame(Traits = c("Small, Medium",
                                      "Medium, Large",
                                      "Small, Large"),
                           Estimate = c(posterior.mode(mcmc_triv_cor_S_M),
                                        posterior.mode(mcmc_triv_cor_M_L),
                                        posterior.mode(mcmc_triv_cor_S_L)),
                           Lower = c(HPDinterval(mcmc_triv_cor_S_M)[,"lower"],
                                     HPDinterval(mcmc_triv_cor_M_L)[,"lower"],
                                     HPDinterval(mcmc_triv_cor_S_L)[,"lower"]),
                           Upper = c(HPDinterval(mcmc_triv_cor_S_M)[,"upper"],
                                     HPDinterval(mcmc_triv_cor_M_L)[,"upper"],
                                     HPDinterval(mcmc_triv_cor_S_L)[,"upper"]))

ggplot(df_mcmc_cors, aes(x = Traits, y = Estimate)) +
  geom_pointrange(aes(ymin = Lower,
                      ymax = Upper)) +
  geom_hline(yintercept = 0,
             linetype = "dotted",
             alpha = 0.3) +
  geom_hline(yintercept = -1,
             linetype = "dotted",
             alpha = 0.3) +
  geom_hline(yintercept = 1,
             linetype = "dotted",
             alpha = 0.3) +
  labs(x = "Environment combinations",
       y = "Correlation (Estimate +/- 95% CIs)") +
  ylim(-1,1) +
  coord_flip() +
  theme_classic()
```



Here we see that the among-individual correlations differ greatly across combinations of 'environments': the correlation is positive between medium and large opponents, while between small:medium and small:large the correlation is negative. Looking at the raw data that we plotted at the beginning of the tutorial, this makes sense — those least aggressive against small opponents tend to be most aggressive against large opponents, and vice versa. We can also see that the 95% credible intervals do not cross zero for any of these correlations, suggesting that they are statistically significant (in whichever direction the correlation is).

Variances, repeatabilities, correlations and IxE

So far we have seen that the character state approach lets us estimate among-individual variance (and repeatabilities) in, and covariance/correlation between, a set of environment-specific sub-traits. But how does this relate to **IxE**?

In the absence of IxE, an individual's random effect size is – by definition – constant with E. It therefore follows that the variance in random effects (i.e., the among-individual variance) is the same in each environment. It also follows that the (among-individual) correlations between environment-specific sub-traits will all be $r = +1$.

We can therefore use the 95% CIs as presented above for the cross-context correlations, and look at whether these include $r = +1$. Clearly we can see that small:medium and medium:large show IxE, as we have significantly negative correlations. For medium:large, while the correlation is significantly positive then it is clear that it is quite far from +1, indicating that IxE also exists across these contexts.

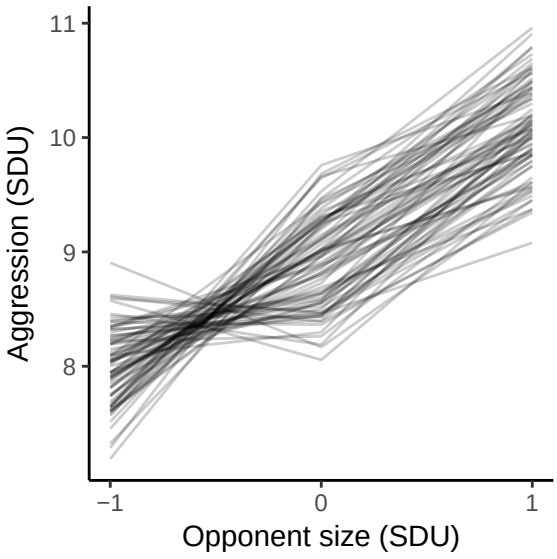
We should also make sure to look at the raw among-individual variances and residual/'within-individual' variances withing each environment, in addition to their repeatabilities. While the repeatabilities here are very similar, if you look back to the summary tables you can see that there is quite a lot more among-individual variation in aggression at large opponent size than at small opponent size; the repeatabilities are similar because there is also greater residual variation at large opponent size than at small. This is important in

the context of IxE because, if there were to be no differences in how individuals respond to environmental change, they should all act in parallel and therefore the among-individual variance should not change across environments/contexts!

Plotting the character state model

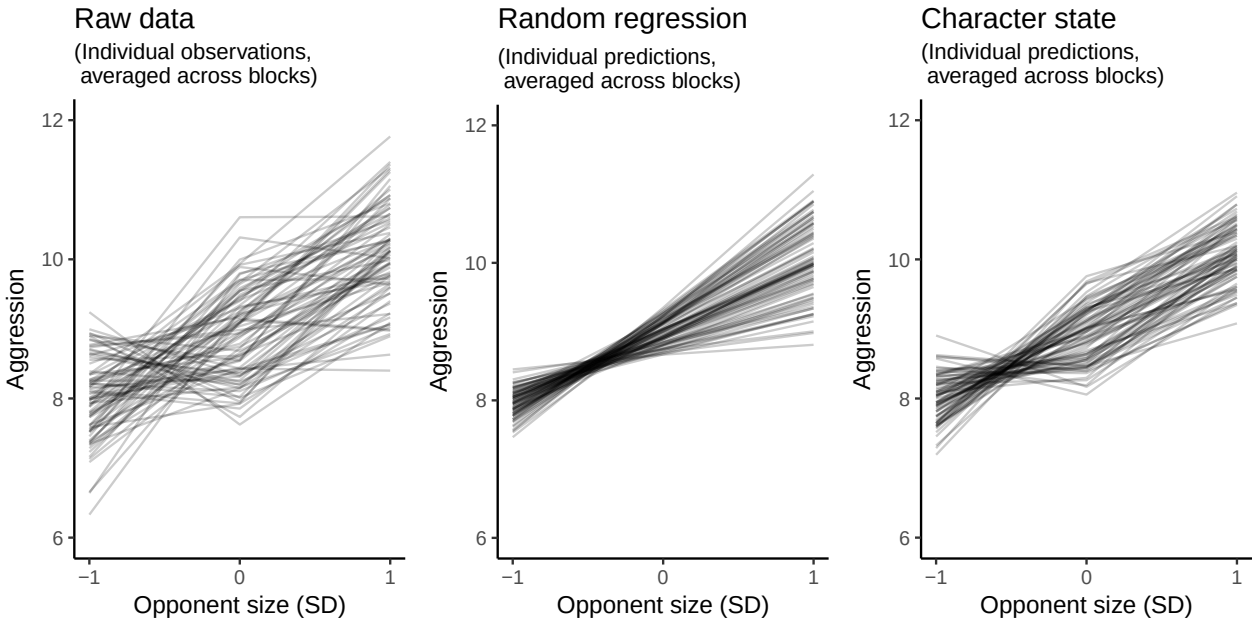
Similar to how we visualised the random regression model, we can use BLUPs to extract individual-level predictions from the model to plot predicted aggression for each individual across each environment. Here we use the `predict` function from `MCMCglmm` to get predicted values for each `trait` and for every `ID`. Because the predict function in `MCMCglmm` is a little confusing, this includes all fixed effects and does give us a predicted value for each block for each individual. We'll simply take the average of an individual's predictions in each environment for this plot. For ease of plotting, we also create a new variable that we set to -1/0/1 depending on which opponent size environment the prediction is for.

```
# Get predictions from MCMCglmm model
aggr_CS_preds <- predict(mcmc_triv_CS, marginal = NULL)

# These come as a single vector, so we need
#  to associate them with the correct individuals
#  and as the right opponent size predictions.
# We also simply average over blocks within each
#  environment for the purposes of plotting.
df_cs_ind <- cbind(df_plast_CS,
                   fit_S = aggr_CS_preds[1:160],
                   fit_M = aggr_CS_preds[161:320],
                   fit_L = aggr_CS_preds[321:480]) %>%
  select(ID, fit_S, fit_M, fit_L) %>%
  group_by(ID) %>%
  summarise(fit_S = mean(fit_S),
            fit_M = mean(fit_M),
            fit_L = mean(fit_L))

# Add numeric value for easier plotting on x-axis
df_cs_ind <- df_cs_ind %>%
  gather(opp_size, aggression,
         fit_S:fit_L) %>%
  mutate(sizeNum = ifelse(opp_size == "fit_S", -1,
                          ifelse(opp_size == "fit_M", 0, 1)))

# Plot values
ggplot(df_cs_ind, aes(x = sizeNum,
                      y = aggression,
                      group = ID)) +
  geom_line(alpha = 0.2) +
  scale_x_continuous(breaks = c(-1,0,1)) +
  labs(x = "Opponent size (SDU)",
       y = "Aggression (SDU)") +
  theme_classic()
```



Here, this brings together all facets of our character state analysis. We can see that (i) there is an increase in average aggression due to opponent size; (ii) correlations differ such that small:medium is negative, while medium:large is positive; (iii) there is greater among-individual variation in aggression when the opponent is large compared to when the opponent is small; (iv) there are changes in the rank order of individual aggression across environments (crossing reaction norms), so among-individual correlations are not perfect and we have evidence of IxE for aggression against different opponent sizes.

We can also compare this model fit to both the original data, and the fit from the `MCMCglmm` random regression:



Adding fitness to character state model

As we discussed earlier, one of the challenges with reaction norm models can be that adding (and, more pertinently, interpreting associations with) another trait (or response variable) can be confusing. Given

suitable data, character state models are an excellent solution because we can simply add a further response variable and fit all correlations. Instead of thinking about how selection acts on plasticity, we will be thinking about how selection on the trait (aggression) changes with environment (opponent size). Crucially (although perhaps not obviously!) **they are the same thing, just seen from alternative viewpoints** (reaction norm vs character state).

Now, let's add `fitness` to create a 4-variable response trait in a new model. Remember that `fitness` has only a single measure per individual, so we use the prior to `fix` its residual (within-individual) variance to essentially zero. This should also have the effect of keeping the residual covariances featuring fitness to essentially zero as well.

Also note that we use the `at.level` keyword to apply the fixed effects only to the aggression observations (i.e., traits 1:3 of the 4-trait response).

```
prior_quad_px  <-  list(R = list(V = diag(c(1,1,1,0.0001)), nu = 2.002, fix = 4),
                        G = list(G1 = list(V = matrix(c(1,0,0,0,
                                                        0,1,0,0,
                                                        0,0,1,0,
                                                        0,0,0,1),4,4,
                                                      byrow = TRUE),
                                           nu = 4,
                                           alpha.mu = rep(0,4),
                                           alpha.V = diag(25^2,4,4))))


mcmc_quad_CS <- MCMCglmm(cbind(agg_S,
                               agg_M,
                               agg_L,
                               fitness) ~ trait +
                           at.level(trait,1:3):opp_order +
                           at.level(trait,1:3):block +
                           at.level(trait,1:3):scale(body_size),
                         random =~ us(trait):ID,
                         rcov =~ us(trait):units,
                         family = c("gaussian","gaussian","gaussian","gaussian"),
                         prior = prior_quad_px,
                         nitt=950000,
                         burnin=50000,
                         thin=450,
                         verbose = TRUE,
                         data = as.data.frame(df_plast_CS),
                         pr = TRUE,
                         saveX = TRUE, saveZ = TRUE)
```

As always, use diagnostic plots etc to check that the model has converged in a reasonable manner (and again, if we were using this model for publication we would need to do various other checks before accepting it). You'll likely get a warning message about not being able to estimate some of the fixed effects (because order is now a factor with many levels) – in this case we can ignore it, as we are keeping those variables in as statistical controls only (and we know from earlier tests that they make little difference to our model).

Here, let's skip to the results. The summary for this model is quite big, so we can just take sections of it in turn. First, let's look at the among-individual (co)variances:

```
summary(mcmc_quad_CS)$Gcovariances
```

```
##                                post.mean     l-95% CI     u-95% CI
## traitagg_S:traitagg_S.ID      0.18235037  0.050695833  0.319084259
## traitagg_M:traitagg_S.ID     -0.13354092 -0.235348171 -0.034987391
## traitagg_L:traitagg_S.ID     -0.12648350 -0.232284328 -0.016221405
## traitfitness:traitagg_S.ID   -0.01331098 -0.036002253  0.008907543
## traitagg_S:traitagg_M.ID     -0.13354092 -0.235348171 -0.034987391
## traitagg_M:traitagg_M.ID      0.22991308  0.083792272  0.381687934
## traitagg_L:traitagg_M.ID      0.14905770  0.029898843  0.278241316
## traitfitness:traitagg_M.ID    0.01695767 -0.006348472  0.041865045
## traitagg_S:traitagg_L.ID     -0.12648350 -0.232284328 -0.016221405
## traitagg_M:traitagg_L.ID      0.14905770  0.029898843  0.278241316
## traitagg_L:traitagg_L.ID      0.30632779  0.128210763  0.514804286
## traitfitness:traitagg_L.ID    0.05150078  0.022615785  0.083310059
## traitagg_S:traitfitness.ID   -0.01331098 -0.036002253  0.008907543
## traitagg_M:traitfitness.ID    0.01695767 -0.006348472  0.041865045
## traitagg_L:traitfitness.ID    0.05150078  0.022615785  0.083310059
## traitfitness:traitfitness.ID  0.02737288  0.019421906  0.036316355
##                               eff.samp
## traitagg_S:traitagg_S.ID     2000.000
## traitagg_M:traitagg_S.ID     2000.000
## traitagg_L:traitagg_S.ID     2000.000
## traitfitness:traitagg_S.ID   2144.143
## traitagg_S:traitagg_M.ID     2000.000
## traitagg_M:traitagg_M.ID     2020.432
## traitagg_L:traitagg_M.ID     2000.000
## traitfitness:traitagg_M.ID   1761.170
## traitagg_S:traitagg_L.ID     2000.000
## traitagg_M:traitagg_L.ID     2000.000
## traitagg_L:traitagg_L.ID     2000.000
## traitfitness:traitagg_L.ID   2000.000
## traitagg_S:traitfitness.ID   2144.143
## traitagg_M:traitfitness.ID   1761.170
## traitagg_L:traitfitness.ID   2000.000
## traitfitness:traitfitness.ID 2000.000
```

As a sanity check, it's a good idea to compare the among-individual (co)variances for the aggression traits here with those found in the previous trivariate model (of course, there will be small differences because of the MCMC sampling, but should be roughly the same).

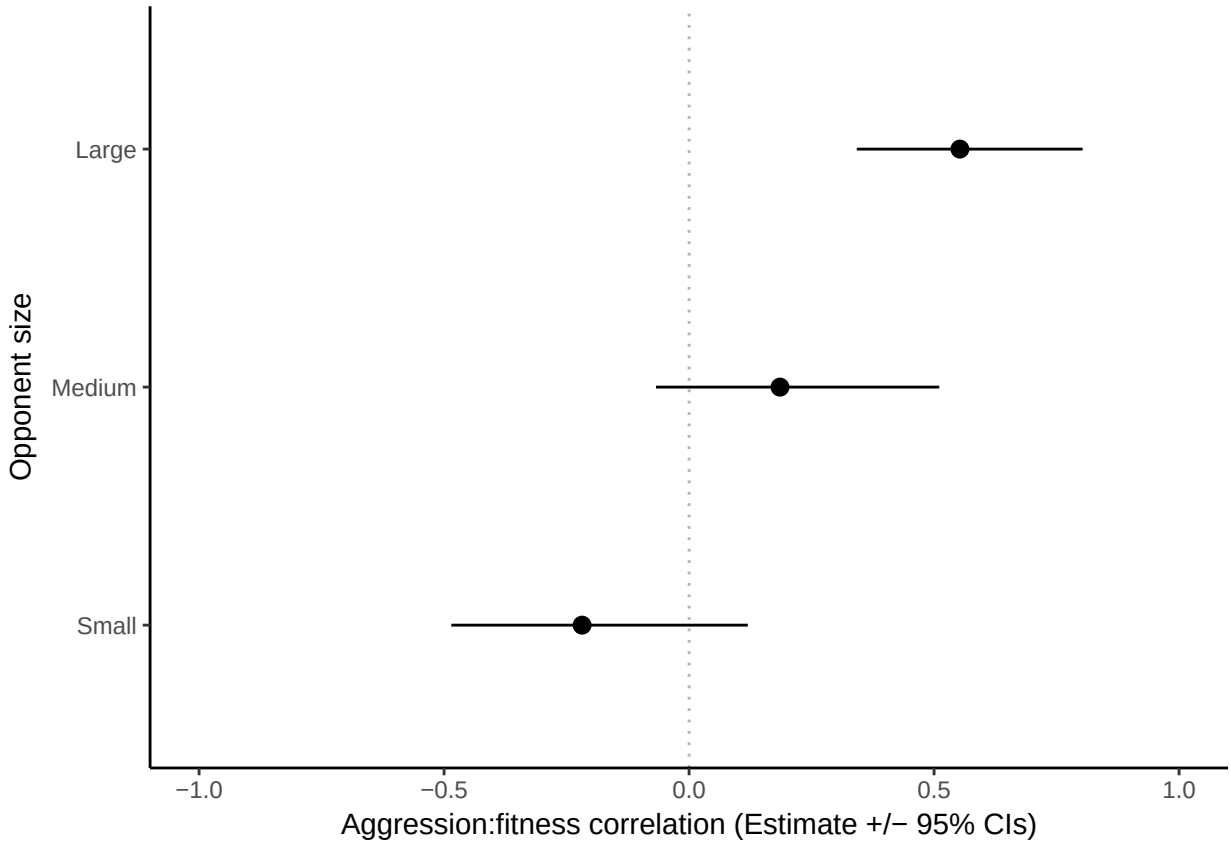
We can also check the residual (co)variances, ensuring that we did constrain the residual (or 'within-individual') variance in fitness to 0.0001. The covariances will have been estimated here, but they should be essentially 0 (which you can see from their estimate and the credible intervals):

```
summary(mcmc_quad_CS)$Rcovariances
```

```
##                                    post.mean     l-95% CI    u-95% CI
## traitagg_S:traitagg_S.units      3.581584e-01  0.252594492 0.476143484
## traitagg_M:traitagg_S.units     -1.058100e-02 -0.093522953 0.068361734
## traitagg_L:traitagg_S.units     -2.727021e-02 -0.119320318 0.061872066
## traitfitness:traitagg_S.units    9.611125e-05 -0.003148960 0.003805668
## traitagg_S:traitagg_M.units     -1.058100e-02 -0.093522953 0.068361734
## traitagg_M:traitagg_M.units      3.683274e-01  0.257539309 0.489049277
## traitagg_L:traitagg_M.units      1.696161e-01  0.077484113 0.267932036
## traitfitness:traitagg_M.units    1.891748e-04 -0.003662057 0.003762664
```

```
## traitagg_S:traitagg_L.units     -2.727021e-02 -0.119320318 0.061872066
## traitagg_M:traitagg_L.units      1.696161e-01  0.077484113 0.267932036
## traitagg_L:traitagg_L.units      4.515262e-01  0.315904287 0.596351696
## traitfitness:traitagg_L.units    5.788416e-04 -0.003872096 0.004507275
## traitagg_S:traitfitness.units    9.611125e-05 -0.003148960 0.003805668
## traitagg_M:traitfitness.units    1.891748e-04 -0.003662057 0.003762664
## traitagg_L:traitfitness.units    5.788416e-04 -0.003872096 0.004507275
## traitfitness:traitfitness.units  1.000000e-04  0.000100000 0.000100000
##                                 eff.samp
## traitagg_S:traitagg_S.units     2000.000
## traitagg_M:traitagg_S.units     2000.000
## traitagg_L:traitagg_S.units     2271.433
## traitfitness:traitagg_S.units   2000.000
## traitagg_S:traitagg_M.units     2000.000
## traitagg_M:traitagg_M.units     1767.183
## traitagg_L:traitagg_M.units     1870.145
## traitfitness:traitagg_M.units   1809.804
## traitagg_S:traitagg_L.units     2271.433
## traitagg_M:traitagg_L.units     1870.145
## traitagg_L:traitagg_L.units     1658.481
## traitfitness:traitagg_L.units   1815.984
## traitagg_S:traitfitness.units   2000.000
## traitagg_M:traitfitness.units   1809.804
## traitagg_L:traitfitness.units   1815.984
## traitfitness:traitfitness.units    0.000
```

Of course, we are most interested in how among-individual variation in aggression in each 'environment' is associated with fitness variation. For ease of interpretation, we shall convert these aggression:fitness covariances to correlations. Just as we visualised the cross-environment correlations for aggression in our trivariate model, a nice way to look at (and assess the significance of) these fitness:environment-specific aggression correlations is to plot their estimates and 95% credible intervals:

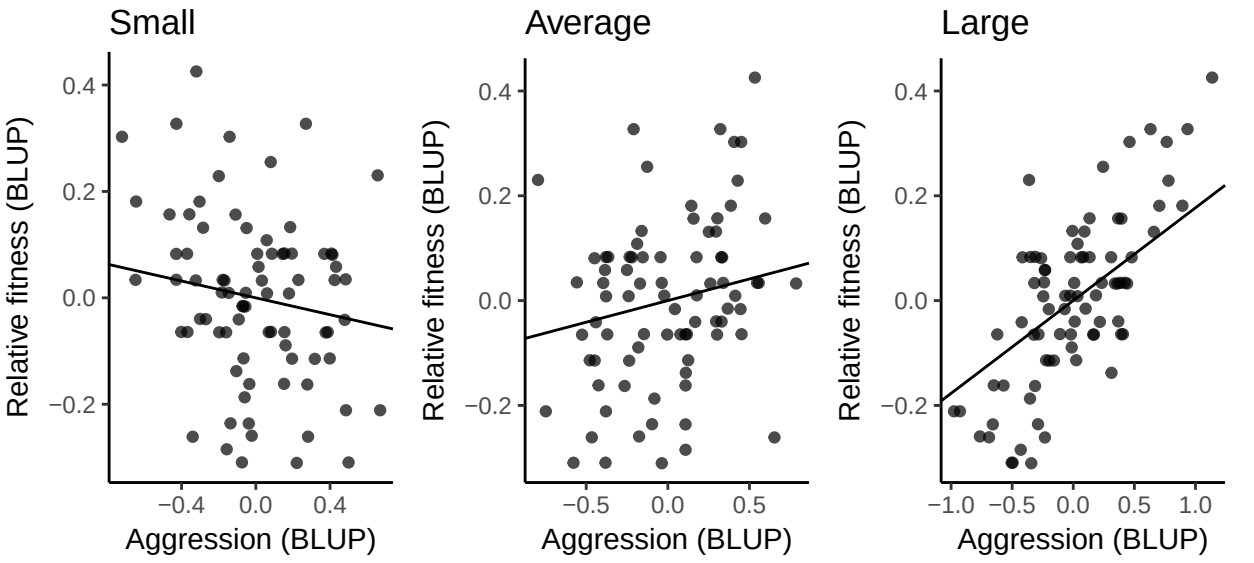


We can see from this figure that the correlations between fitness and opponent size-specific aggression varies widely across opponent size classes: from negative (and non-significant) against small opponents to positive (and non-significant) against average-sized opponents, up to **positive and significant** against large opponents. So qualitatively at least, it seems that estimated selection on individual aggressiveness goes from weakly negative with small opponents, through to more strongly positive for large opponents.

Let's finish by plotting the association between relative fitness and individual deviations (BLUPs) for aggression at each opponent size 'environment'. We've provided the code below to do this at the level of small opponents, and you can rework some of it to do the same for the other two opponent size 'environments':

```
# Get BLUPs for all traits
df_aggr_fit_coefs <- data_frame(Trait = attr(colMeans(mcmc_quad_CS$Sol), "names"),
                                Value = colMeans(mcmc_quad_CS$Sol)) %>%
  separate(Trait, c("Trait","Type","ID"), sep = "\\.", fill = "right") %>%
  filter(Type == "ID")

# Subset for small, fitness
#  and reform for easier plotting
df_S_fit_coefs <- df_aggr_fit_coefs %>%
  filter(Trait %in% c("traitagg_S", "traitfitness")) %>%
  select(-Type) %>%
  spread(Trait, Value)

# Find the regression line for this opponent size:
#  the covariance of aggression(S), relative fitness
#  divided by the variance in aggression(S)
s_fit_slope <-  mean(mcmc_quad_CS$VCV[,"traitagg_S:traitfitness.ID"]/
                     mcmc_quad_CS$VCV[,"traitagg_S:traitagg_S.ID"])
```

```
gg_s_fit <- ggplot(df_S_fit_coefs, aes(x = traitagg_S,
                                       y = traitfitness,
                                       group = ID)) +
  geom_point(alpha = 0.7) +
  geom_abline(intercept = 0, slope = s_fit_slope) +
  labs(x = "Aggression (BLUP)",
       y = "Relative fitness (BLUP)",
       title = "Small") +
  theme_classic()
```



Conclusions

Using both random regression and character state models, we find evidence of significant among-individual variance in aggressiveness. We also show that, on average, there is a plastic response manifest as higher aggression toward larger opponents. Since IxE is present we can also say that individuals differ in their plastic responses to opponent size. In our view however, the magnitude and nature of the IxE is more readily interpreted under the character state model because this enables us to estimate environment-specific among-individual variances and covariances (in addition to their derived repeatabilities and correlations). Importantly, the fact that under a null hypothesis of no IxE we expect the variances to be equal and the cross-environment correlations to be +1 means we have a clear point of comparison against which to both test and interpret the IxE.

We found strong associations between higher levels of aggression and fitness using both types of model. However, while the results of the random regression model can be interpreted in terms of selection on plasticity and intercept, there are dangers of doing so if one doesn't fully understand (i) the dependence of model outputs on the scaling/centring of the X axis, and/or (ii) the need to fully account for slope-intercept covariance, if the goal is to test for "selection on plasticity". In contrast, the character state model yields estimates of selection in each environment that are easily interpreted. Using this method, we found that the only significant covariance between fitness and aggression was when aggression was measured against large opponents. Here, greater aggression was significantly linked to higher fitness. Aggression against average and small opponents was not significantly associated with fitness.

Data wrangling

Below, we demonstrate how to do some 'data wrangling' to turn the standard long format into a different ('wide') format that can be used for character state models.

We need to do some rearranging of our data frame to structure it for this type of model. We want to end up with the aggression values for each 'environment' (for a single block) on a single row in the data frame, with each sub-trait in a different column. Each row will also have the observation of block, and the opponent body size measurement for that block.

First, we should name these traits `agg_S`, `agg_M`, and `agg_L` respectively (and we are reloading the original data first into a new data frame):

```
df_plast_2 <- read_csv("aggression.csv")
```

```
df_plast_2 <- df_plast_2 %>%
  mutate(opp_type = ifelse(opp_size == -1, "agg_S",
                           ifelse(opp_size == 0, "agg_M",
                                  "agg_L")))
```

This does mean that we need to turn our assay repeat variable into an 'order' variable, so that we can test for any systematic effects of opponent size order in the model:

```
# Create a new data frame, 'df_order',
#  which holds the 'order' of opponents for each individual in each block
df_order <- df_plast_2 %>%
  select(ID, block, assay_rep, opp_type) %>%
  arrange(ID, block, assay_rep) %>%
  separate(opp_type, into = c("tmp","opp_type")) %>%
  select(-tmp) %>%
  spread(assay_rep, opp_type) %>%
  unite(opp_order, `1`:`3`)
```

We now want to 'spread' our aggression observations across separate columns, such that each row has a value for aggression in each of the three opponent size classes (where 1 row represents an individual's observations for one block). Having spread these values across distinct columns, we then join the data frame of 'order' values to it.

Note: here we are **not** going to standardise aggression measurements to (for example) standard deviation units, but if you wanted to do that you must do it **before** spreading them across multiple rows. That would not only scale them by the overall phenotypic SD, but if you centred them as well you would retain differences in the mean values across opponent size classes (so you could still model population-level plasticity).

```
# Spread aggression values across different
#  opponent sizes
df_plast_CS <- df_plast_2 %>%
  select(ID, block, aggression, body_size, opp_type) %>%
  spread(opp_type, aggression)

# Add values for order of opponent sizes
df_plast_CS <- left_join(df_plast_CS,
                         df_order,
                         by = c("ID","block"))
```

. . . and quickly check to make sure the new data frame looks the way we wanted:

```
head(df_plast_CS)
```

We did not include **fitness** here because it makes the spreading of our aggression columns difficult, but we can add this back to our data frame now — remembering to make sure it is only repeated once per individual:

```
# Create new df containing non-NA fitness values for each individual:
df_fit <- df_plast_2 %>%
  select(ID, fitness) %>%
  filter(!is.na(fitness))

# Join to the CS data frame
df_plast_CS <- left_join(df_plast_CS,
                         df_fit,
                          by = "ID")

# Delete duplicates in fitness
# - easy way here is to keep only those with one of the blocks
df_plast_CS <- df_plast_CS %>%
  mutate(fitness = ifelse(block == -0.5, fitness, NA))
```

Finally, let's standardise fitness to 'relative fitness' by dividing by the population mean:

```
df_plast_CS <- df_plast_CS %>%
  mutate(fitness = fitness/mean(fitness, na.rm=TRUE))
```

. . . and check the results of our wrangling efforts:

```
head(df_plast_CS)
```

Again, http://r4ds.had.co.nz/ is the place to go to learn more about the `tidyverse` *packages used for this kind of 'data wrangling'.*