Computer Exercise 8 (R)

1: Let's quickly recapitulate a problem from the previous exercise 6, where you analyzed the **Jimson** dataset. Now you will use Bayesian analysis to analyze this dataset to which you previously applied Maximum Likelihood methods. Remember: The data consist of the length/width ratio (LenWid) of second seedling leaves of two types of Jimson weed, called globe (coded as 1) and nominal (coded as 2). Three seeds of each type were planted in 16 pots (Pot is the pot identification number). You might assign G and N as level labels for the types and you also need to make Pot into a factor:

```
dat$Type <- factor(dat$Type,levels=c(1,2),labels=c("G","N"))

dat$Pot <- as.factor(dat$Pot)
```

Before continuing, try using `boxcox` to find a suitable transformation for the response variable. `boxcox` does not accept random effects, so you need to fit a lm model with only fixed factors. Now, assuming you found log transformation to give homogeneous variances, try plotting the difference between the weed types in each pot so you remind yourself of what to expect from the forthcoming analysis:

```
bwplot(logLenWid ~ Type|Pot, data = dat)
```

A Bayesian analysis needs prior distributions of the model parameters, including variance components. These priors are a priori "guesses" of the parameter values the analysis will give information about. Ideally we want to set very weak (non-informative) priors, in order not to influence the results of the analysis. If the data set is rather large and the effect sizes are big this is usually not a problem. So far you have used the default settings of MCMCglmm, which correspond to weak priors. Now we will try setting our own to learn a bit about how priors for the random effects may influence the results (there is usually no reason to worry about the priors for the fixed effects).

To calculate the total variance in our data use the this command:

```
V0 <- var(dat$logLenWid)
```

If you used log-transformation for the LenWid variable, the variance is roughly `V0=0.03`. One approach for setting a prior for variance components is to partition `V0` equally among the random effects in the model. Notice that this is a conservative prior because it assumes that no variance is explained by the main effect of Type, which, is the effect that you will test for significance in your model – that is good, because we can be sure not to have influenced the analysis in a way that would result in false significance of the main effect. Let us split the total variance among the random effects. We have three variance components (residual variance, `Pot`, and `Type:Pot`). Let us first set a weak prior for the variance components:

```
pri1w <- list(R = list(V = V0/3, nu = 0.002), G = list(G1 = list(V
= V0/3, nu = 0.002), G2 = list(V = V0/3, nu = 0.002)))
```

The prior is specified as a list giving the parameters of distributions of the variance components. The `R` part of the list describes the prior distribution of the residual variance, and the `G` part is a list that describes the prior distribution of the other variance components. In this case `G1` and `G2` will correspond to `Pot` and `Type:Pot`, assuming this is the order we define these random effects in our model. For each variance component, a variance parameter `V` and a parameter `nu` describe the prior distribution. The bigger `nu` is, the stronger the prior (for very large `nu`, the prior is that the variance is "a priori known" to be equal to `V`). If `nu` is close to zero, the prior is weak. Thus, the two kinds of parameters describe a value for a variance and how much we believe in this value. The value of `nu=0.002` is often used for variance components and can be considered very weak. Let's also set a stronger prior:

```
pri1s <- list(R = list(V = V0/3, nu = 1), G = list(G1 = list(V =
V0/3, nu = 1), G2 = list(V = V0/3, nu = 1)))
```

Typically, one would prefer the weaker `pri1w` to the stronger `pri1s`, but if the MCMC sampling has problems for the weaker prior, it can be acceptable to use the stronger prior (although, before deciding to rely on a stronger prior, one should first try increasing the number of MCMC iterations and the thinning interval). Now run the MCMCglmm analysis with these priors and compare the quality of the MCMC sampling:

```
fm1w <- MCMCglmm(logLenWid ~ Type, random = ~ Pot + Type:Pot, data
= dat, prior= pri1w, verbose = FALSE)

summary(fm1w)
```

```
fm1s <- MCMCglmm(logLenWid ~ Type, random = ~ Pot + Type:Pot, data
= dat, prior= pri1s)

summary(fm1s)
```

To get some understanding of what MCMCglmm is actually doing, try:

```
plot(fm1w)
```

and:

```
plot(fm1s)
```

The plots show the MCMC simulated sequence of the model parameters (both fixed effects and variance components). Ask your teacher about the plots if you find them difficult to interpret. Note that the analysis is a simulation, and that the output will vary slightly between simulation runs.

Compare the effective sample sizes for the variance components (if these are much smaller than the actual MCMC sample size, there is a problem with serial correlations. You can also use commands like

`autocorr.plot(fm1s$VCV)` to examine if successive MCMC samplings of the variance components are correlated. As a rule of thumb, the autocorrelation at the lag given by the thinning (10 in the default case) should be smaller than around 0.1 to have acceptably independent MCMC samplings. You can look directly at the MCMC output, using a command like `plot(mod1w$VCV)` for the variance components or `plot(mod1w$Sol)` for the fixed effects.

Is either of the two analyses acceptable, why so?

From the output, you may notice that the variance component for the `Type:Pot` interaction is much smaller than the one for `Pot`. Try an analysis without this random effect using the weak prior:

```
pri2w = list(R = list(V = V0/2, nu = 0.002), G = list(G1 = list(V =
V0/2, nu = 0.002)))

fm2w = MCMCglmm(logLenWid ~ Type, random = ~ Pot, data = dat,
prior= pri2w, verbose=FALSE)

summary(fm2w)

summary(fm1w)
```

One way of deciding if the simpler model is preferable is to compere the DIC values and to select the model with the smallest value. DIC is the Deviance Information Criterion, which is a generalization of AIC. In this case the second, simpler model has the smaller DIC (at least in the MCMC run your teacher performed), so we prefer that one. What do you conclude about the effect of `Type` from this model? Does the result fit with your expectation?

2: In this example you will investigate if there is sexual size dimorphism in sepsis flies. You may recognize this problem from test exercise 3. Now you will analyze the same problem with more data, using the Bayesian approach. Read in the file **flies23.txt**. The file contains information on line identity, Sex, and body size of each fly. In the experiment, flies from 74 different inbreed lines where reared in the laboratory and adult body size was measured. You may recollect from test exercise 3 that you modeled sex-specific line effects as a random factor. If we want to try this again we have to set Bayesian priors for our random effects that then would include both a common effect of line, and a Sex:line interaction variance in addition. Let's use MCMCglmm to test if males and females differ in size.

First we set weak and unbiased priors for our residual variance (R) and our random effects (G1 = line, and G2 = Sex:line), all equal to 1 (this is the typical non-informative prior) :

```
prior23 <- list(R=list(V=1, nu=0.002), G = list(G1=list(V=1,
 nu=0.002), G2=list(V=1, nu=0.002)))
```

Then let's run our model:

```
fm23 <- MCMCglmm(size ~ Sex, random = ~ line + line:Sex, data = dat,
          prior = prior23, verbose = FALSE)
```

You can have a quick look at the autocorrelation between stored samples of the main effects (Sol) and the random effects (VCV):

```
autocorr.plot(fm23$Sol)

autocorr.plot(fm23$VCV)
```

The correlations seem small (<0.1 is a good rule) and thus there seems to be no big problem with autocorrelation.

Let's look at the results:

```
summary(fm23)
```

It seems that the effective sample sizes for the random effects are a bit below the number of stored iterations (1000), but acceptable. Try making the prior a little bit stronger and setting the variance to something closer to the actual observed variance:

```
prior23b <- list(R=list(V=0.05, nu=0.1), G =
list(G1=list(V=0.05, nu=0.1), G2=list(V=0.05, nu=0.1)))
```

Then run your model again:

```
fm23b <- MCMCglmm(size ~ Sex, random = ~ line + line:Sex, data =
dat, prior = prior23b, verbose = FALSE)
```

Look at the autocorrelations again for this new model. Also look at the effective sample size for the random effect. The stronger prior improves the simulations compared to the first model. You could increase the `nu` even more, but this may influence the posterior estimates which is not what you want in this case as you have decided to specify relatively low variance in your random effects, making for a less conservative test of the fixed effect. A way to check the influence of the prior is to increase both V and nu and look at point estimates and confidence limits for the random effects. For example, try running a model with a prior using: `V = 1, nu = 1`, and one using `V = 10, nu = 1`, and compare the estimates of the variance components to those of model `fm23b` **a)** When do the priors start to affect estimates based on this rather large dataset? Do you find sexual size dimorphism in your chosen model?

To get a better idea of your data you can try a plot:

```
bwplot(size ~ line|Sex, data = dat)
```

The lines were started from singly mated females and have then been kept isolated for many generations (only brother-sister matings). Flies from the same line are thus more closely related than flies originating from different lines, and the differences between lines should be due to genetic effects. First of all, how do you go about interpreting the effect of `line` and `Sex:line` in model `fm23b`? Would you say they are statistically important? Variance components in MCMCglmm are a priori defined to be non-negative, which is logical. However, this means that their confidence limits will never overlap zero. A better way to evaluate the importance of such variance components is therefore to compare them to each other, or to calculate how much of the total variance they explain. In this example it may be interesting to calculate the proportion of the remaining variance explained by genetic differences (lines) after controlling for the main effect of Sex, for general differences in body size between lines: $VARIANCE_{line}$ / [$VARIANCE_{line}$ + $VARIANCE_{line:Sex}$ + $VARIANCE_{error}$] and for variance in sexual size dimorphism between lines; $VARIANCE_{line:Sex}$ / [$VARIANCE_{line}$ + $VARIANCE_{line:Sex}$ + $VARIANCE_{error}$].

To get this from your stored MCMC results you would write for the overall variance between lines (notice that the formula is exactly the same as above):

```
prop_line <- fm23b$VCV[,"line"] /(fm23b$VCV[,"line"] +
fm23b$VCV[,"line:Sex"] + fm23b$VCV[,"units"])

prop_sex_by_line <- fm23b$VCV[,"line:sex"] /(fm23b$VCV[,"line"] +
fm23b$VCV[,"line:Sex"] + fm23b$VCV[,"units"])
```

`posterior.mode()` gives the mode of the posterior distribution for the requested parameter (the mode will be equal to the mean given that data is symmetrically distributed around the mode). With Bayesian statistics it is also easy to calculate confidence limits for variance components and their ratios since we estimate all parameters and store them during multiple runs. `HPDinterval()` gives the upper and lower values of within which 95% (or some other user-specified percentage) of the parameter estimates lie. Try to calculate the mode (the most frequent parameter estimate) and the 95% confidence limit for the proportion of variation that is genetic:

```
posterior.mode(prop_line)
HPDinterval(prop_line, 0.95)
plot(prop_line)
```

Note that in this case it was important to set relatively weak priors for the random effects as you here were directly interested in their relative importance. Splitting the variance equally among the random effects was also important to get an unbiased prior for this question. If you work more with Bayesian statistics you will learn that what is an appropriate and unbiased prior really depends on your particular question and goal of the analysis. **b)** What proportion of the variance in body size within the sexes is due to differences between lines? Do the same for the interaction variance – the variance in sexual size dimorphism between lines. What can we conclude from this?

c) Finally, compare your parameter estimates for the fixed and random effect in the MCMCglmm model to what you get with lmer() in the lme4 package:

```
library(lme4)

fmLMER <- lmer(size ~ Sex + (1|line) + (1|Sex:line), data = dat)

summary(fmLMER)
```

3: The experiment with the flies was actually carried out at two temperatures. One day old fly larvae from each line were split between 23˚C which can be considered benign, and 31˚C which can be considered very hot and sub-lethal. Your job will be to investigate if temperature affects body size of these flies and if males and females respond differently to temperature. First read in the file **fliesTEMP.txt.** The file is similar to before but now there are also flies reared in 31˚C included along with an additional column giving information on which temperature treatment each fly belongs to. As there are only two temperatures used we could consider it a factor with two levels, so convert temperature to a factor before continuing. You can get an idea of your data by plotting (you need a big screen window, or you may choose to plot a subset of the data):

```
bwplot(size ~ temp:Sex |line, data = dat)
```

To test for different effects of the temperature treatment in males and females it seems reasonable to incorporate the crossed random effects temp:line, Sex:line and Sex:temp:line, in addition to the main effect of line. Let's try to set weak and unbiased priors for these effects based on the amount of residual variance in an anova not including the interaction:

```
fm <- lm(size ~ Sex + temp, data = dat)

anova(fm)
```

The residual variance is `V0 = 0.11`, so we will split that equally among our variance components. This should serve as a conservative prior as we a priori assume that the fixed effect of interest; the interaction between Sex and temperature, explains no variance in the data. Note that you now have five components to your prior; R giving the residual variance, G1 giving the prior for the random effect for the intercept of "line", and G2 giving the prior for the random effect of "temp:line", G3 giving the effect of "Sex:line", and G4 giving the effect of "temp:Sex:line". What this last random effect, G4, estimates, is if there is variation between lines in the difference in response to temperature between males and females.

```
prior2331 = list(R=list(V=V0/5, nu=0.002), G = list(G1=list(V=
V0/5,nu=0.002), G2=list(V= V0/5, nu=0.002), G3=list(V= V0/5,
nu=0.002), G4 =list(V= V0/5, nu=0.002)))
```

Then run your model:

```
fmTEMP <- MCMCglmm(size ~ Sex + temp + Sex:temp, random = ~ line +
temp:line + Sex:line + temp:Sex:line, data = dat, prior =
prior2331, verbose = FALSE)
```

Try some diagnostics to see if everything looks ok:

```
autocorr.plot(fmTEMP$VCV)
```

There seems to be strong correlations between simulation runs for the random effects. While fixed effects seem robust, these effects are not estimated independently of the variance components for the random effects that therefore need to be accurately estimated. Let's have a look at the model output:

```
summary(fmTEMP)
```

We can see that the numbers of effective samples for some of the random effects are much fewer than the number of stored MCMC-runs. MCMCglmm has problems estimating the random effects for this more complicated model. You previously learned that one way of dealing with difficulties in estimation of random effects is to change the priors. Another, often better, way that may spare you of inferring strong a priori assumptions about your data is to increase the interval for storing simulations in MCMCglmm. Here is the same model again, but now let's store runs every 100th iteration rather than every 10th. To get a proper number of stored runs, we also need to let our simulations run for a little bit longer:

```
fmTEMP2 <- MCMCglmm(size ~ Sex + temp + Sex:temp, random = ~ line +
temp:line + Sex:line + temp:Sex:line, data = dat, prior =
prior2331,  nitt=110000, burnin=10000, thin=100, verbose = FALSE)
```

We can specify many things in our MCMC-models. You can try typing: `?MCMCglmm` to get an idea. For your purpose here, notice: `nitt` = the number or runs of the MCMC-chain (the more the better up until a certain point), `burnin` = the number of runs before values are being stored (sometimes the MCMC-chain needs some time to stabilize), and `thin` = the interval for storing values (if you store values each run these will more likely be correlated and non-independent). Let's compare the autocorrelations for random effects in our two models with different thinning-intervals:

```
autocorr.plot(fmTEMP$VCV)
```

then try:

```
autocorr.plot(fmTEMP2$VCV)
```

Increasing the thinning interval seems to have helped. Look at the results:

```
summary(fmTEMP2)
```

Try to interpret the coefficients for the fixed effects. You can also try a plot that might help you with the interpretation:

```
interaction.plot(dat$temp, dat$Sex, dat$size)
```

What do you conclude about the effects of temperature on body size in the two sexes?