Primer on Non-Linear Mixed-Effects Model (NLMM) Analysis in R

In the case of the two continuous dance parameters that we analysed (waggle phase duration and return phase duration), we fitted non-linear mixed-effects models. Even though we tried to fit linear mixed-effects models, we could not make inferences from these models, as the model assumption of homoscedasticity of residuals was not validated. On closer inspection of the raw data, we found that the distribution of the two parameters at the level of both, individual runs and mean values per dance, were not normally distributed (e.g., see Fig. 2 and 3).

Fitting simple non-linear mixed effects models in R is straightforward and we describe the process below. We have also attached an accompanying R file containing code that can be used to replicate our model fitting process. We do not go into details regarding the type of non-linear function to fit, as this will depend on the data and the underlying biological phenomenon. One option would also be to fit multiple non-linear functions to the data and choose between these models. Once a non-linear function is chosen, then the next steps are:

1. Obtain reasonable starting values to implement the NLMM
2. Fit the NLMM
3. Perform model diagnostics
4. Obtain the results of the model and plot the predicted fits

While steps 2-4 are common to fitting (generalized) linear mixed-effects models ((G)LMMs), step 1 is different and vital for fitting NLMMs. The R stats package (installed by default) implements several self-start functions that can be used to obtain starting values (like SSmicmen() for Michaelis-Menten models, SSlogis() for logistic models etc.). Additionally, several other packages provide self-start functions for models not covered in the stats package. The aomisc package provides several self-start functions of biological interest (the following blogpost provides additional details on fitting these self-starting functions: https://www.statforbiology.com/2020/stat_nls_usefulfunctions/). To obtain the starting values (step 1), we fitted a simplified non-linear model (without mixed-effects). This can be done either through the nls() function in the stats package in case there is only one continuous predictor, or through the nlsList() from the nlme package when there are additional categorical predictors (code for obtaining these starting values are provided in the accompanying R file).

Once starting values are obtained, NLMMs can be fitted (step 2) using the nlme() function from the nlme package. The function consists of three main arguments, specifying the non-linear model formula, the fixed effects and the random effect. The first argument, specifying the non-linear function, contains the response variable and how it is linked to the continuous predictor, as well as the parameters associated with the function. In our case, we used a logarithmic regression of the form:

$$Y = a + b * \log(X)$$

Where Y is the response variable (in this case the dance parameter of either waggle phase duration, circuit duration or return phase duration), X is the distance and *a* and *b* are the parameters associated with the logarithmic regression. We were specifically interested in the value of *b*, as *b*/100 represents the change in waggle phase duration (Y) for a 1% change in distance (X).

To understand this, consider the logarithmic regression formula given above. Suppose the distance changes by 1% from 100 m to 101 m. In this case, the change in waggle phase duration is:

$$(a + b * \log(101)) - (a + b * \log(100))$$

$$(b * \log(101)) - (b * \log(100))$$

$$b * (\log(101) - \log(100))$$

$$b * (\log\left(\frac{101}{100}\right))$$

log(101/100) is equal to 0.00995 or approximately 0.01. Thus, a 1% change in distance corresponds to a change in waggle phase duration equivalent to b*0.01 or b/100. Similarly, a 10% change in distance would correspond to a change in waggle phase duration approximately equal to b/10 (since log(1.1) is equivalent to 0.095 or approximately 0.1).

Thus, the first part of the model is written as:

$$(Dance\ Parameter) \sim \text{NLS.logcurve}(distance, a, b)$$

indicating that the non-linear function linking the parameter and distance is NLS.logcurve with the parameters distance, *a* and *b*. Here NLS.logcurve is a self-start function from the package aomisc which is a wrapper for the logarithmic regression model.

In the second argument, we specified the fixed effects on which *a* and *b* depend. We built two different versions of the model for each parameter to compare the fixed effects:

1. Neither *a* nor *b* depend on the vegetation condition (implemented as a + b ~ 1), representing the situation where the vegetation condition had no effect on the non-linear relationship between the dance parameter and distance.
2. Both *a* and *b* depend on the vegetation condition (implemented as a + b ~ condition), representing the situation where the vegetation condition determines the non-linear relationship between the dance parameter and distance.

In terms of conventional (G)LMMs, these are analogous to a model in which only distance is a fixed effect and a model in which distance and vegetation condition has an interaction effect, respectively. We then compared between these 2 models using the AICc value to determine the better fitting model for each dance parameter.

The third argument of the function, the random effect, was the same in both models. We allowed *b* to vary with individual bee ID, thereby implementing a model analogous to random slopes in (G)LMMs.

In addition to this, we provided the starting values for the required parameters in the NLMMs. For the first model, since there was no covariate of condition, only one value each of *a* and *b* needs to be estimated and we provided these two starting values. In the case of the second model, two values each of *a* and *b* have to be estimated (corresponding to the two vegetation conditions), and hence we provided four starting values. We used the default 'nlminb' optimizer with 100 iterations of the optimizer to fit the models.

The model assumptions were verified (step 3) similar to the case of (G)LMMs, and we checked for homoscedasticity and normality of residuals as well as normality of random effects. Since we used the same generic methods, we don't go into detail here (the

accompanying R file has the code used for checking these assumptions). In the case of waggle phase duration for cerana, even after fitting the NLMM we still faced the issue of heteroscedasticity in the residuals. To deal with such cases, the nlme() function implements an optional weights argument to accommodate the within-group heteroscedasticity structure. We used trial and error to determine the best class of variance functions from the available standard classes. We did this by fitting different models each with its own variance function and then visually inspecting the homoscedasticity plot. We found that, for our data, the varPower function, corresponding to power variance function structure with a function coefficient of 10 on the variance covariate *distance,* was able to deal with the heteroscedasticity in the data.

Finally, obtaining the results of the NLMMs (Table S3), as well as obtaining predictions for plotting are similar to methods used in (G)LMMs implemented via lmer and glmmTMB packages in R. The methods are fairly standard in statistical analysis using mixed-effects models in R and are provided in the accompanying R file.