**Generalized Linear Mixed Models (GLMM)**

You can use GLMMs to fit any (sensible) number of IVs to your data, similarly to GLMs. You can include categorical and/or continuous independent variables and interactions as needed. The difference with GLMMs is that you incorporate a random factors(s) into the model which accounts for lack of independence in your data. The random factor is commonly used when repeated samples are taken on the same individual or when multiple samples are taken at the same spatial location (e.g., repeated surveys on the same transect or many samples collected in close proximity). They can also be used to model any other hierarchical structure or non-independence in the data.

Note: It’s best to keep your models as simple as possible as complex models may not converge. As always before modelling explore the dataset fully.

This example shows a binomial GLMM, but very similar adaptations from a GLM can be applied for poisson or gamma GLMMs (or any other suitable distribution) using glmmTMB().

**Check variables in dataset**

str(dataset)

class(dataset$IV1)

class(dataset$IV2)

class(dataset$DV)

To change categorical variables to factors if appropriate:

dataset$IV1<-factor(dataset$IV1)

**Binomial GLMM where dependent variable is proportional**

> install.packages('glmmTMB')

> library(glmmTMB)

**Fit the full model**

Fit the full (most complex) model:

**a. If you have one IV, and:**

> mod.1 <- glmmTMB(DV ~ IV + (1|RANDOM.IV), weights = DV.total.observations, family = binomial, data = dataset)

…………………………………………………………………………………………………………………………………………….Go to validate

In this example we have fitted a random intercept to the model (1|RANDOM.IV) which allows the model intercept to vary by each level of the RANDOM.IV (this is the most common use of a random factor). It may be the case that the relationship DV ~ IV is different at each level of the RANDOM.IV. In this case we may want to add a random slope to the model. We would add that to the left-hand side of the random factor (1+RANDOM.IV|RANDOM.IV). Be careful with this, your model may become too complex and not converge.

**b. If you have more than one IV**

> mod.1 <- glmmTMB (DV ~ IV.1 \* IV2 + (1|RANDOM.IV), family = binomial, weights = DV.total.observations, data = dataset)

The default plot function which creates a QQ plot of the normal distribution isn’t very helpful for GLMMs therefore we will validate our model by simulating residuals at the end.

………………………………………………………………………………………………………………………………Go to term selection

**Term selection**

Start with the most complex (‘full’) model, then drop IV terms one at a time. Compare the AIC

scores and select the model with the lowest score. For example,

> mod.1 <- glmmTMB(DV ~ IV.1 \* IV2+ (1|RANDOM.IV), family = binomial, weights = DV.total.observations, data = dataset)

> mod.2 <- glmmTMB (DV ~ IV.1 + IV2+ (1|RANDOM.IV), family = binomial, weights = DV.total.observations, data = dataset)

> mod.3 <- glmmTMB (DV ~ IV.1+ (1|RANDOM.IV), family = binomial, weights = DV.total.observations, data = dataset)

> mod.4 <- glmmTMB (DV ~ IV2+ (1|RANDOM.IV), family = binomial, weights = DV.total.observations, data = dataset)

Then, compare the models with the AIC:

> AIC(mod.1, mod.2, mod.3, mod.4) # select the best model (lowest AIC) …………………………………………………………………………………………………………………………………………….Go to validate

#Note: there is currently no option to fit a quasibinomial GLMM and the advice from model developers is that overdispersion for binomial GLMMs is not a major concern. If a poisson GLMM is overdispersed you can fit a negative binomial model.

**Validate the fitted model**

As GLMMs are more complex than GLMs, we are going to use a specialized package to validate them and calculate the approx. R2 value.

> summary(model) # Shows summary model outputs

>install.packages(“DHARMa”)

>library(DHARMa)

The assumptions of linearity and homogeneity of variance for a GLMM are based upon the residuals, so we must extract them and then perform validation (as we did for GLMs).

>simulationOutput <- simulateResiduals(model, n = 1000, re.form = (1|RANDOM.IV))

>plot(simulationOutput)

>plotResiduals(simulationOutput, dataset$IV)

The left plot assesses linearity of the residuals (they should be in a straight line). It also tests for overdispersion. If you don’t have linearity or overdispersion you may be missing an IV, there may be a more appropriate distribution for your data or your data may not be linear, in which case look into a GAMM. The plot on the right tests for homogeneity of variance. The 3 lines indicate whether the variance is homogenous across the dataset. Again, violation of homogeneity may indicate you are missing an IV, the distribution you have chosen is inappropriate or your DV may have heterogeneity in the residuals. A small deviation may not be a big problem, but you may want to consider an alternative method where you can fit a variance structure which allows heterogeneity at different levels of an IV (e.g., generalized least squares).

#Note: if you have a large dataset then you may need to recalculate the residuals by a grouping factor (e.g., at the site level) to appropriately assess them. To do this:

>resids <- recalculateResiduals(simulationOutput, group = dataset $IV)

>plot(resids)

If you have time series data and a time based IV is in your fitted model then you also want to test for temporal autocorrelation. To do this, you will need to group your residuals by the time scale (e.g., by year). Note that if your fitted model indicates no significant temporal variation you don’t need to consider this.

>res <- recalculateResiduals(simulationOutput, group = dataset$IV)

>testTemporalAutocorrelation(simulationOutput = res, time = unique(dataset$IV))

Temporal autocorrelation occurs if you have a lag in the data which has not been appropriately accounted for in the fitted model. The temporal autocorrelation suggests that the next time period is significantly influenced by the previous time period. Significance will be noted if this is the case. If temporal autocorrelation is found you can fit a temporal autocorrelation structure (e.g., AR1()) to your model.

Now calculate the R2:

>install.packages(“performance”)

>library(performance)

>r2(model)

You will get a conditional R2 and a marginal R2. The conditional R2 is the variation in the DV explained by the fixed and random factors. The marginal R2 is the variation in the DV explained by the fixed factors alone.

To perform post hoc analysis …………………………………………………………………………Go to post hoc analysis

**Post hoc analysis**

We can perform pairwise tests of the categorical IV(s) or interactions in our fitted model using the estimated marginal means.

library(emmeans) #package which extracts the estimated marginal means from a model for post hoc analysis of significant factors

1. **If we have one categorical IV:**

**#Note:** If you have multiple categorical IVs but no interaction, repeat these steps.

#extracts the estimated marginal means of the IV based upon the model

>emm.mod <- emmeans(model, ~ IV1)

#shows the estimated effect size of each level of the factor (note estimates are on the link scale)

>contrast(emm.mod)

#pairwise analysis between each factor level (note: negative estimate suggests that level1 < level2; positive estimate level1 > level2)

>pairs(emm.mod)

#plot the estimates marginal means (estimates on link scale)

>plot(emm.mod)

#Create a summary plot of the estimated marginal means of your IVs (particularly useful if you have an interaction).

#You can customize the plot using ggplot

>library(ggplot2) #package which allows you to create custom plots

#Here, we use Geom\_jitter to add the actual values from your dataset around the levels of each IV.

>emmip(model, ~ IV1, "response")+

> geom\_jitter(aes(x = IV1, y = DV, colour = IV1), data = dataset)

**b. If we have an interaction between two categorical IVs**

#test the effect of IV1 at each level of IV2.

>emm.mod <- emmeans(model, ~ IV1|IV2)

#**Note:** you can reverse the factor order to test the effect of IV2 at each level of IV1

>contrast(emm.mod)

>pairs(emm.mod)

>plot(emm.mod)

>library(ggplot2) #package which allows you to create custom plots

>emmip(model, ~ IV1|IV2, "response")+

> geom\_jitter(aes(x = IV1, y = DV, colour = IV1), data = dataset)

**c. If we have an interaction between a categorical IV (IV1) and a continuous IV (IV2)**

> emtrends(model, pairwise ~ IV1, var = "IV2")

> emmip(model, IV1 ~ IV2, cov.reduce = range)