Processing and Analysis of Biological Data

(Generalized) Linear Mixed Models (GLMM)

Øystein H. Opedal

16 Nov 2022

Mixed-effect models I: Introduction

One very common extension of the linear model is the linear mixed model. The 'mixed' comes from the fact that these models include two variable types: fixed effects and random effects.

$y = \mathbf{X}\beta + \mathbf{Z}\mu + \epsilon$

The fixed effects are the standard predictor variables of a linear model, i.e. variables for which we are interested in their (independent) effect on the response variable. For example, in an ANOVA-type analysis of a factorial experiment, the experimental factors will be treated as fixed effects.

The random effects are variables for which we are not necessarily interested in the mean value of the response for each value of the predictor, but rather the variance in these effects. A common use of random effects is to account for the non-independence of observations that arise, for example, when several measurements are taken from the same individual. Failing to account for this would lead to an artificial inflation of the degrees of freedom of the analysis. This issue is called *pseudoreplication*, because it uses non-independent data points as replicates.

Beyond modelling patterns of non-independence in the data, random-effect models are also often use to estimate variance components that may be of direct interest. A typical application is in quantitative genetics, where the aim of a study can be to estimate the components of the variance in a phenotypic trait. A simple model can be

$y_i = g_i + e_i$

where g is the genetic variance component and e is the environmental variance component. Estimating these variance components exemplifies the general approach of *variance component analysis*.

Variance component analysis using random-effect models

Random-effect models allow us to estimate the variance residing at multiple levels, and thus to ask for example what percentage of variation in a variable is due to differences among populations, and to differences among individuals within populations.

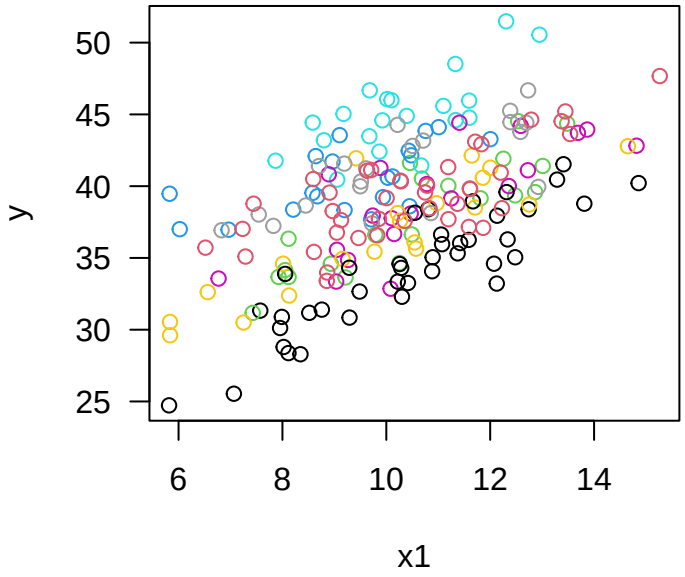
Consider the following simulated data.

```
set.seed(145)
x1 = rnorm(200, 10, 2)

groupmeans = rep(rnorm(10, 20, 4), each=20)
groupID = as.factor(rep(paste0("Group", 1:10), each=20))
```

```
y = 2 + 1.5*x1 + groupmeans + rnorm(200, 0, 2)

plot(x1, y, col=as.numeric(groupID), las=1)
```



There is clearly an overall positive relationship between y and $x1$, but there are also differences among the groups (colors). Here we could partly be interested in quantifying the contribution of group differences to the overall variance, and partly in accounting for group membership when estimating the relationship between y and $x1$.

We will start with the variance component analysis. To specify a random-effect model with the `glmmTMB` package, we use the `(1|pop)` format. The 1 refers to the intercept of the model, and we are therefore specifying random intercepts, but not random slopes. This means that we are estimating the variance in intercepts among groups.

```
library(glmmTMB)

data = data.frame(y, x1, groupID)
head(data)
```

```
##          y       x1 groupID
## 1 35.32047 11.37383  Group1
## 2 37.94114 12.13273  Group1
## 3 35.95843 11.07340  Group1
## 4 38.77994 13.81206  Group1
## 5 33.21379 12.12632  Group1
## 6 38.38936 12.74069  Group1
```

```
m = glmmTMB(y ~ 1 + (1|groupID), data=data)

summary(m)
```

```
##  Family: gaussian  ( identity )
## Formula:          y ~ 1 + (1 | groupID)
## Data: data
##
##      AIC      BIC   logLik deviance df.resid
##   1096.4   1106.3   -545.2   1090.4      197
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  groupID  (Intercept)  9.205   3.034
##  Residual             11.869   3.445
## Number of obs: 200, groups:  groupID, 10
##
## Dispersion estimate for gaussian family (sigma^2): 11.9
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  38.7679     0.9899   39.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The summary table is familiar, and contains some of the same information as we have seen in simple linear models and in GLMs. To extract only the part related to the random effect, we can call the function `VarCorr`, and then extract the variances (recall it´s the variances that are additive, not the standard deviations). Note that we have to extract an "attribute" of the VarCorr object, rather than e.g. a slot in a list.

```
VarCorr(m)
```

```
##
## Conditional model:
##  Groups   Name        Std.Dev.
##  groupID  (Intercept) 3.0340
##  Residual             3.4451
```

```
VarAmongGroups = attr(VarCorr(m)$cond$groupID, "stddev")^2
VarWithinGroups = attr(VarCorr(m)$cond, "sc")^2
```

```
VarAmongGroups
```

```
## (Intercept)
##    9.204912
```

```
var(groupmeans)
```

```
## [1] 9.805633
```

The among-group variance is a little bit smaller than the actual variance of the group means. This illustrates that mixed models are good at estimating variance components while taking into account the uncertainty associated with variation within groups, which will tend to inflate the among-group variance (because the latter will include the uncertainty in the estimated means within each group). To get a feeling for how this work, we can compute the average sampling variance of each group mean, and subtract it from the among-group variance.

```
mean_sampling_variance = mean(tapply(y, groupID, var)/20)
var(groupmeans) - mean_sampling_variance
```

```
## [1] 9.212205
```

The result is close to the variance estimated from the mixed model.

To calculate the percent of the variance explained by groups, we divide by the total estimated variance.

```
VarAmongGroups/(VarAmongGroups+VarWithinGroups)*100
```

```
## (Intercept)
##    43.68009
```

To interpret the actual variances, we could e.g. scale the standard deviations by the trait mean to obtain a coefficient of variation (CV). To maintain the additivity of the variances, an even better approach is to scale the variances themselves by the square of the trait mean, thus obtaining a squared CV, CV^2. We scale by the square of the mean because variances also have squared units, and we thus obtain a unitless number.

```
CV2_Among = VarAmongGroups/mean(x1)^2
CV2_Within = VarWithinGroups/mean(x1)^2
CV2_Total = CV2_Among + CV2_Within
```

Finally it is nice to organize all these values in a table.

```
df = data.frame(Mean = mean(x1), SD = sd(x1),
                Among = VarAmongGroups/(VarAmongGroups+VarWithinGroups)*100,
                Within = VarWithinGroups/(VarAmongGroups+VarWithinGroups)*100,
                CV2_Among, CV2_Within, CV2_Total)
df = apply(df, MARGIN=2, FUN=round, digits=2)
df
```

```
##       Mean         SD      Among     Within  CV2_Among CV2_Within  CV2_Total
##      10.28       1.93      43.68      56.32       0.09       0.11       0.20
```

Data exercise: Variance partitioning with random-effects models.

Pick any of the datasets we have worked with in the course that includes at least one grouping variable, and perform a random-effect variance partitioning. Produce a neat table and interpret the results biologically and statistically.

Nested and crossed random effects

When we have more than one random effect, we can make further choices when specifying the random-effect structure of the model. *Nested random effects* have, as the name implies, a hierarchical structure, so that specific levels of the lower-level grouping factor occurs only for one level of the upper-level grouping factor. A typical example is hierarchical sampling designs, where we for example take several samples from a set of individuals from a set of families. In this case we could treat individual nested within family as a nested random effect. In this case we can extract variance components for family (variance among family means), individual (variance among individuals *within families*), and the residuals, which will represent variance within individuals.

Such nesting is already a property of the data, but we can formulate this explicitly when fitting the model by using `(1|family/individual)`.

In other cases the levels of the different random factors do not have such a nested structure, and are then considered *crossed random effects*. An example is plots and years in a long-term study (each plot occurs across several years, and several plots are recorded within the same year). In this case we can forumate the random-effect structure as `(1|plot) + (1|year)`. In this case the residual variance component will represent unexplained variance, and can not be interpreted directly as e.g. variance within plots (or within years).

As an example, we return to the data on blossom traits.

```
dat = read.csv("datasets/blossoms/blossoms.csv")
names(dat)
```

```
##  [1] "pop"   "patch" "ASD"   "GAD"   "GSD"   "LBL"   "LBW"   "UBL"   "UBW"
## [10] "GW"    "GA"
```

```
dat$pop = as.factor(dat$pop)
dat$patch = as.factor(paste(dat$pop, dat$patch, "_"))
```

```
m = glmmTMB(UBW ~ 1 + (1|pop/patch), data=dat)
summary(m)
```

```
##  Family: gaussian  ( identity )
## Formula:          UBW ~ 1 + (1 | pop/patch)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##    980.8    994.1   -486.4    972.8      200
##
## Random effects:
##
## Conditional model:
##  Groups    Name        Variance Std.Dev.
##  patch:pop (Intercept) 3.238    1.800
##  pop       (Intercept) 1.143    1.069
##  Residual              4.463    2.113
## Number of obs: 204, groups:  patch:pop, 96; pop, 9
##
## Dispersion estimate for gaussian family (sigma^2): 4.46
##
## Conditional model:
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  18.9254     0.4467   42.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case around half of the variance is residing at the levels of population and patch, and the rest within patches. Let us now include a fixed predictor in the model to estimate the scaling of upper and lower bract sizes.

```
m = glmmTMB(UBW ~ LBW + (1|pop/patch), data=dat)
summary(m)
```

```
##  Family: gaussian  ( identity )
## Formula:          UBW ~ LBW + (1 | pop/patch)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##    626.9    643.4   -308.5    616.9      197
##
## Random effects:
##
## Conditional model:
##  Groups    Name        Variance Std.Dev.
##  patch:pop (Intercept) 0.2255   0.4749
##  pop       (Intercept) 0.1045   0.3233
##  Residual              1.0128   1.0064
## Number of obs: 202, groups:  patch:pop, 95; pop, 9
##
## Dispersion estimate for gaussian family (sigma^2): 1.01
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.83757    0.58859   4.821 1.43e-06 ***
## LBW          0.82670    0.02924  28.277  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is now much less variance explained by population and patch. This is because a large portion of the variance is explained by the fixed predictor.

Random-intercept regression

We can also use random intercepts to "account for" the non-independence of observations for each group. The model syntax for the fixed effects (here $x1$) is like for linear models, and for the random effects (groupID) we use the same syntax as above.

```
m = glmmTMB(y ~ x1 + (1|groupID), data=data)

summary(m)
```

```
##  Family: gaussian  ( identity )
```

```
## Formula:          y ~ x1 + (1 | groupID)
## Data: data
##
##      AIC      BIC   logLik deviance df.resid
##    896.5    909.7   -444.2    888.5      196
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  groupID  (Intercept) 10.576   3.252
##  Residual              4.081   2.020
## Number of obs: 200, groups:  groupID, 10
##
## Dispersion estimate for gaussian family (sigma^2): 4.08
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  23.8343     1.3018   18.31   <2e-16 ***
## x1            1.4528     0.0764   19.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated slope is close to what we used to simulate the data (1.5), and it therefore seems like the model has adequately estimated the mean within-group relationships.

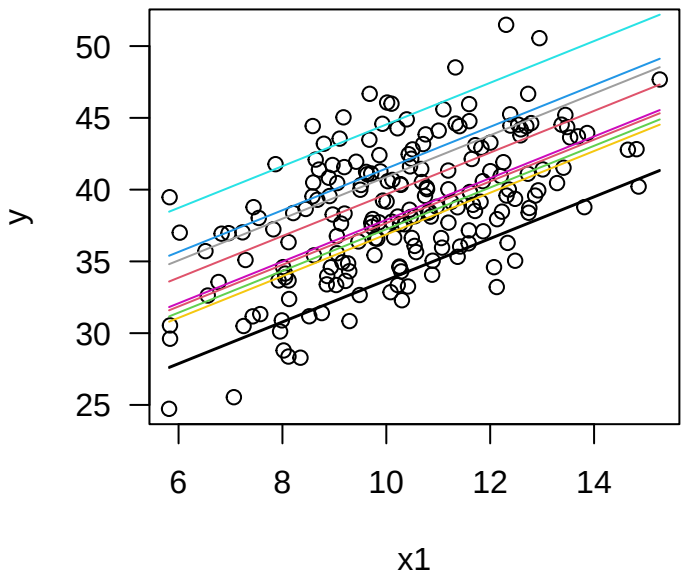
The estimated random-effect variance for `GroupID` is associated with a set of estimated intercepts for each level of the Group factor. These estimates are called *best linear unbiased predictors* (BLUPs), and can be extracted and used for example to produce plots.

```
coef(m)
```

```
## $groupID
##         (Intercept)      x1
## Group1     19.19438 1.45279
## Group10    25.14684 1.45279
## Group2     22.71307 1.45279
## Group3     26.94286 1.45279
## Group4     30.01120 1.45279
## Group5     23.36571 1.45279
## Group6     22.34922 1.45279
## Group7     26.35297 1.45279
## Group8     19.12443 1.45279
## Group9     23.14223 1.45279
```

```
newx = seq(min(x1), max(x1), length.out=200)

plot(x1, y, las=1)
for(i in 1:length(levels(groupID))){
  y_hat = coef(m)$cond$groupID[i,1] + coef(m)$cond$groupID[i,2]*newx
  lines(newx, y_hat, col=i)
}
```



We can also use the predict function to make predictions for a specific level of the random effect.

```
y_hat = predict(m, newdata=list(x1=newx, groupID=rep("Group5",200)), re.form=NULL)
```

EXERCISE: Fit a naïve linear model to the same data and evaluate how the two fitted models differ.

Data exercise: random-intercept models

Pick any of the datasets we have worked with in the course that includes at least one grouping variable, and perform a random-intercept analysis (regression, ANCOVA or ANOVA). Produce relevant summary statistics and interpret the results biologically and statistically.

Generalized linear mixed models

Generalized linear mixed models extend linear mixed models in the same way that simple GLMs extend linear models. We can generally fit mixed-effect models with any common error distribution and link function, including binomial, Poisson and negative binomial errors. There are several packages for fitting such data, including the `glmmTMB` package we used above, and the `lme4` package. The syntax is very similar, and they tend to give very similar results for simple models.

The data on bee distribution in the Brazilian Atlantic forest were collected from sampling units (sites) belonging to 72 study areas. We can include study area (SA) as a random factor estimating variation in the mean abundance of bees across space.

```
dat = read.csv("datasets/Eulaema.csv")
dat$SA = as.factor(dat$SA)
```

```
m = glmmTMB(Eulaema_nigrita~MAP + (1|SA), family="nbinom2", data=dat)
summary(m)
```

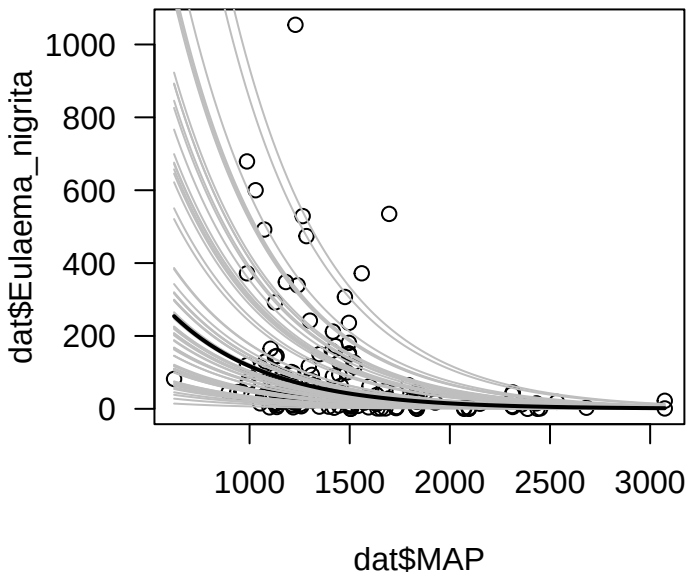
```
##  Family: nbinom2  ( log )
## Formula:          Eulaema_nigrita ~ MAP + (1 | SA)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##   1790.8   1803.5   -891.4   1782.8      174
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  SA     (Intercept) 1.382    1.176
## Number of obs: 178, groups:  SA, 72
##
## Dispersion parameter for nbinom2 family (): 1.87
##
## Conditional model:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.820199   0.542558  12.570  < 2e-16 ***
## MAP         -0.002057   0.000353  -5.827 5.65e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
newMAP = seq(min(dat$MAP), max(dat$MAP), length.out=200)

plot(dat$MAP, dat$Eulaema_nigrita, las=1)

for(i in 1:length(levels(dat$SA))){
  y_hat = exp(coef(m)$cond$SA[i,1] + coef(m)$cond$SA[i,2]*newMAP)
  lines(newMAP, y_hat, col="grey")
}

y_hat = exp(summary(m)$coef$cond[1,1] + summary(m)$coef$cond[2,1]*newMAP)
lines(newMAP, y_hat, lwd=2)
```


dat$Eulaema_nigrita
1000
800
600
400
200
0
1000
1500
2000
2500
3000
dat$MAP