Symbolic Formulae for Linear Mixed Models[⋆]

Emi Tanaka[1,2] and Francis K. C. Hui[3]

[1] The University of Sydney, Camperdown NSW 2008, Australia
[2] Monash University, VIC 3800, Australia
emi.tanaka@monash.edu
[3] Australian National University, Acton ACT 2601, Australia
francis.hui@anu.edu.au

Abstract. A statistical model is a mathematical representation of an often simplified or idealised data-generating process. In this paper, we focus on a particular type of statistical model, called linear mixed models (LMMs), that is widely used in many disciplines e.g. agriculture, ecology, econometrics, psychology. Mixed models, also commonly known as multi-level, nested, hierarchical or panel data models, incorporate a combination of fixed and random effects, with LMMs being a special case. The inclusion of random effects in particular gives LMMs considerable flexibility in accounting for many types of complex correlated structures often found in data. This flexibility, however, has given rise to a number of ways by which an end-user can specify the precise form of the LMM that they wish to fit in statistical software. In this paper, we review the software design for specification of the LMM (and its special case, the linear model), focusing in particular on the use of high-level symbolic model formulae and two popular but contrasting R-packages in lme4 and asreml.

Keywords: multi-level model · hierarchical model · model specification · model formulae · model API · fixed effects · random effects

1 Introduction

Statistical models are mathematical formulation of often simplified real world phenomena, the use of which is ubiquitous in many data analyses. These models are fitted or trained computationally, often with practitioners using some readily available application software package. In practice, statistical models in its mathematical (or descriptive) representation would require translation to the right input argument to fit using an application software. The design of these input arguments (called application programming interface, API) can help ease the friction in fitting the user's desired model and allow focus on important tasks, e.g. interpreting or using the fitted model for purposes downstream.

While there are an abundance of application software for fitting a variety of statistical models, the API is often inconsistent and restrictive in some fashion. For example, in linear models, the intercept may or may not be included by default; and the random error typically assumed to be identical and independently distributed (i.i.d) with no option to

[⋆] Supported by R Consortium

modify these assumptions straightforwardly. Some efforts have been made in this front such as by the `parsnip` package (Kuhn 2018) in the `R` language (R Core Team 2018) to implement a tidy unified interface to many predictive modelling functions (e.g. random forest, logistic regression, linear regression etc) and the `scikit-learn` library (Pedregosa et al. 2011) for machine learning in the `Python` language (Van Rossum and Drake Jr 1995) that provides consistent API across its modules (Buitinck et al. 2013). There is, however, little effort on consistency or discussion for the software specification of many other types of statistical models, including the class of linear mixed models (LMMs), which is the focus of this article.

LMMs (a special case of mixed models in general, which are also sometimes referred to as hierarchical, panel data, nested or multi-level models) are widely used across many disciplines (e.g. ecology, psychology, agriculture, finance etc) due to their flexibility to model complex, correlated structures in the data. This flexibility is primarily achieved via the inclusion of random effects and their corresponding covariance structures. It is this flexibility, however, that results in major differences in model specification between software for LMMs. In `R`, arguably the most popular general purpose package to fit LMMs is `lme4` (Bates et al. 2015) – total downloads from RStudio Comprehensive R Archive Network (CRAN) mirror from `cranlogs` (Csárdi 2019) indicate there were over two million downloads for `lme4` in the whole of 2018, while other popular mixed model packages (e.g. `nlme`, Pinheiro et al. 2019; `rstan`, Stan Development Team 2019; `brms`, Bürkner 2017; Bürkner 2018) in the same year have less than half a million downloads, albeit `rstan` and `brms` are younger packages. Another general purpose LMM package is `asreml` (Butler, Cullis, et al. 2009), which wraps the proprietary software ASreml (Gilmour, Gogel, et al. 2009) into the `R` framework. As this package is not available on CRAN, there are no comparable download logs, although, citations of its technical document indicates popular usage particularly in the agricultural sciences. In this paper, we discuss only `lme4` and `asreml` due to their active maintenance, maturity and contrasting approaches to LMM specification.

The functions to fit LMM in `lme4` and `asreml` are `lmer` and `asreml`, respectively. Both of these functions employ high-level symbolic formulae as part of their API to specify the model. In brief, symbolic model formulae define the structural component of a statistical model in an easier and often more accessible terms for practitioners. The earlier instance of symbolic formulae for linear models was applied in Genstat (VSN International 2017) and GLIM (Aitkin et al. 1989), with a detailed description by Wilkinson and Rogers (1973). Later on, Chambers and Hastie (1993) describe the symbolic model formulae implementation for linear models in the `S` language, which remains much the same in the `R` language. While the symbolic formula of linear models generally have a consistent representation and evaluation rule as implemented in `stats::formula`, this is not the case for LMMs (and mixed models more generally) – the inconsistency of symbolic formulae arises primarily in the representation of the random effects, with the additional need to specify the covariance structure of the random effects as well as structure of the associated model matrix that governs how the random effects are mapped to (groups of) the observational units.

In Section 2, we briefly describe the symbolic formula in linear models. We then describe the symbolic model formula employed in the LMM functions `lmer` and `asreml` in Section 3. We follow by illustrating a number of statistical models motivated by the analysis of publicly available agricultural datasets, with corresponding API for `lmer` and `asreml` in Section 4. We limit the discussion of symbolic model formulae to mostly those implemented in `R`, however, it is important to note that the conceptual framework is not limited to this language. We conclude with a discussion and some recommendations for future research in Section 5.

2 Symbolic Formulae for Linear Models

A special case of LMMs is linear models, which comprises of only fixed effects and a single random term (i.e. the error or noise), given in a matrix notation as

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{e}, \tag{1}$$

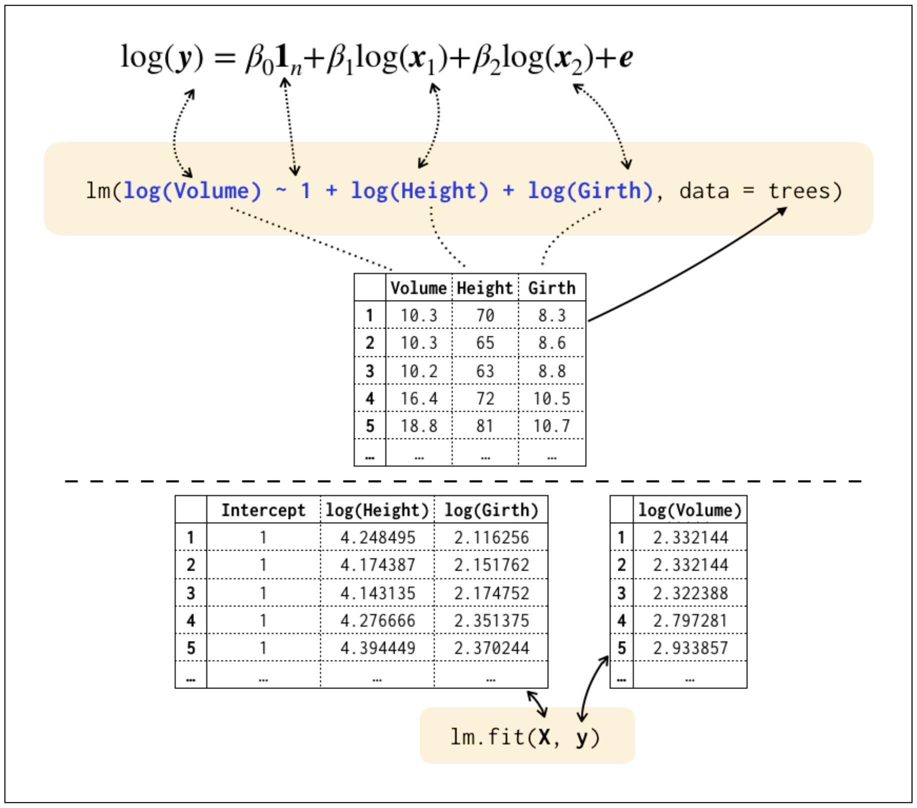
where $\boldsymbol{y}$ is a n-vector of responses, $\mathbf{X}$ is the $n \times p$ design matrix with an associated p-vector of fixed effects coefficients $\boldsymbol{\beta}$, and $\boldsymbol{e}$ is the n-vector of random errors. Typically we assume $\boldsymbol{e} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$.

The software specification of linear model is largely divided into two approaches: (1) input of arrays for the response $\boldsymbol{y}$ and design matrix for fixed effects $\mathbf{X}$, and (2) input of a symbolic model formula along with a data frame that define the variables in the formula. The input of the data frame may be optional if the variables are defined in the parental environment, although such approach is not recommended due to larger potential for error (e.g. one variable is sorted while others are not).

Symbolic model formulae have been heavily used to specify linear models in `R` since its first public release in 1993, inheriting most of its characteristic from `S`. In `R`, formulae have a special class `formula`, and can be used for other purposes other than model specification, such as `case_when` function in `dplyr` `R`-package (Wickham et al. 2019), which uses the left hand side (LHS) to denote cases to substitute with given value on the right hand side (RHS) - these type of use is not within the scope of this paper. The history of the `formula` class in `R` (and `S`) is considerably longer than other popular languages, e.g. the `patsy` `Python` library (N. J. Smith et al. 2018), which imitates `R`'s formula, was introduced in 2011 and used in `Statsmodels` library (Seabold and Perktold 2010) to fit a statistical model.

Symbolic model formulae makes use of the variable names defined in the environment (usually through the data frame) for specifying the precise model formulation. With linear models, the LHS indicate the response; the RHS resembles its mathematical form; and the LHS and RHS are separated by `~` which can be read as "modelled by". For example, the symbolic model formula `y ~ 1 + x` can be thought of as the vector `y` is modelled by a linear predictor consisting of an overall intercept term and the variable `x`.

When the variables are numerical then the connection between the formula to its regression equation is obvious – the LHS is $\boldsymbol{y}$, while the RHS corresponds to the columns of the design matrix $\mathbf{X}$ in the linear model (1). One advantage of this symbolic model formula approach is that any transformation to the variable can be parsed in the model formula and may be used later in the pipeline (e.g. prediction in its original scale). This contrasts to when the input arguments are the design matrix and the corresponding response vector – there is now an additional step required by the user to transform the data before model fitting and subsequently afterwards for extrapolation. Such manual transformation also likely results in manual back-transformation later in the analysis pipeline for interpretation reasons. This no doubt creates extra layer of friction for the practitioner in their day-to-day analysis. Figure 1 illustrates this connection using the `trees` dataset.



$$\log(\boldsymbol{y}) = \beta_0\mathbf{1}_n+\beta_1\log(\boldsymbol{x}_1)+\beta_2\log(\boldsymbol{x}_2)+\boldsymbol{e}$$

```
lm(log(Volume) ~ 1 + log(Height) + log(Girth), data = trees)
```

	Volume	Height	Girth
1	10.3	70	8.3
2	10.3	65	8.6
3	10.2	63	8.8
4	16.4	72	10.5
5	18.8	81	10.7
...	...	...	...

	Intercept	log(Height)	log(Girth)
1	1	4.248495	2.116256
2	1	4.174387	2.151762
3	1	4.143135	2.174752
4	1	4.276666	2.351375
5	1	4.394449	2.370244
...	...	...	...

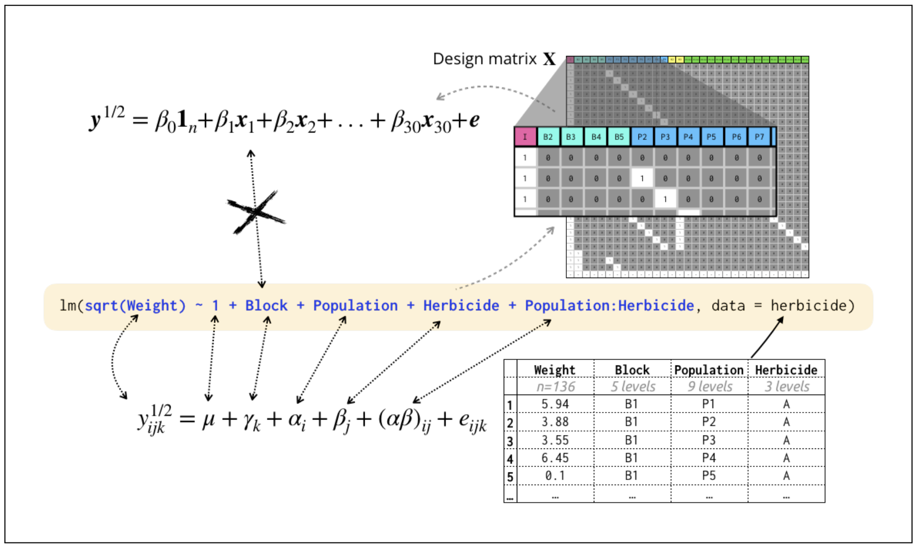
	log(Volume)
1	2.332144
2	2.332144
3	2.322388
4	2.797281
5	2.933857
...	...

```
lm.fit(X, y)
```

Fig. 1. There are two main approaches to fitting a linear model illustrated above with the fit of a linear model to the `trees` dataset: (1) the top half uses the `lm` function with the input argument as a symbolic model formulae (in blue); (2) the bottom half uses the `lm.fit` function which requires input of design matrix and the response. The latter approach is not commonly used in `R`, however, it is the common approach in other languages; see Section 2.1 about the data and the model.

The specification of the intercept by `1` in the formula, as done in Figure 1, is unnecessary in `R` since this is included by default. In turn, the removal of the intercept can be done by including `-1` or `+0` on the RHS. In this paper, the intercept is explicitly included as the resemblance to its model equation form is lost without it. While the omission of `1` is long ingrained within `R`, we recommend to explicitly include `1` and do not recommend designing software to require explicit specification to remove intercept as currently required in R; see Section 2.3 for further discussion on this.

Categorical or factor variables are typically converted to a set of dummy variables consisting of 0s and 1s indicating whether the corresponding observation belongs to the respective level. For parameter identifiability, a constraint needs to be applied, e.g. the treatment constraint will estimate effects in comparison with a particular reference group (the default behaviour in R). Note that in the presence of categorical variables, the direct mapping of the symbolic formula to the regression equation is lost. However, the mapping is clear in converting the model equation to the so-called Analysis of Variance (ANOVA) model specification as illustrated in Figure 2, which represents the fit of a two-way factorial ANOVA model to the `herbicide` data.


Design matrix $\mathbf{X}$
$\boldsymbol{y}^{1/2} = \beta_0 \mathbf{1}_n + \beta_1 \boldsymbol{x}_1 + \beta_2 \boldsymbol{x}_2 + \ldots + \beta_{30} \boldsymbol{x}_{30} + \boldsymbol{e}$
I B2 B3 B4 B5 P2 P3 P4 P5 P6 P7
$y_{ijk}^{1/2} = \mu + \gamma_k + \alpha_i + \beta_j + (\alpha\beta)_{ij} + e_{ijk}$

```
lm(sqrt(Weight) ~ 1 + Block + Population + Herbicide + Population:Herbicide, data = herbicide)
```

	Weight	Block	Population	Herbicide
	n=136	*5 levels*	*9 levels*	*3 levels*
1	5.94	B1	P1	A
2	3.88	B1	P2	A
3	3.55	B1	P3	A
4	6.45	B1	P4	A
5	0.1	B1	P5	A
...	...	...	...	...

Fig. 2. In the presence of categorical variables, the resemblance of the symbolic model formulae to its regression model form is not immediately obvious. In this case, categorical variables are transformed to a set of dummy variables with constraint applied for parameter identifiability. As such, a single categorical variable span a number of columns in the design matrix. On the other hand, if the model equation is written using the ANOVA model specification (with index notation), then the categorical variables have an immediate connection to the fixed effects in the model; see 2.2 for more information about the data and the model.

Interaction effects are specified easily with symbolic model formula by use of the `:` operator as seen in Figure 2. More specifically, the formula in Figure 2 can also be written more compactly as `sqrt(Weight) ~ 1 + Block + Population * Herbicide` where the `*` operator is a shorthand for including both main effects and the interaction effects. Further shorthand exists for higher order interactions, e.g. `y ~ 1 + (x1 + x2 + x3)^3` is equivalent to `y ~ 1 + x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:x2:x3`, a model that contains main effects as well as two-way and three-way interaction effects. The `1` can be included in the bracket as `y ~ (1 + x1 + x2 + x3)^3` to yield the same result. Perhaps surprisingly, `y ~ (0 + x1 + x2 + x3)^3` does not include the intercept in the fitted model, since `0` is converted to `-1` and carried outside the bracket and power operator. The formula simplification rule, say for `y ~ (0 + x1 + x2 + x3)^3`, in `R` can be found by

```
formula(terms(y ~ (0 + x1 + x2 + x3)^3, simplify = TRUE))

## y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:x2:x3 - 1
```

2.1 Trees Volume: Linear Model

The `trees` data set (original data source from T. A. Ryan, Joiner, and B. F. Ryan 1976, built-in data in R) contain 31 observations with 3 numerical variables. The model shown in Figure 1 is a linear model in (2) with the 31×3 design matrix $\mathbf{X} = \left[\mathbf{1}_{31}\ \log(\boldsymbol{x}_1)\ \log(\boldsymbol{x}_2)\right]$, where $\boldsymbol{x}_1$ is the tree height and $\boldsymbol{x}_2$ is the tree diameter (named `Girth` in the data). Finally, $\boldsymbol{y}$ is the log of the volume of the tree.

In Figure 1, the connection of the data column names to symbolic model formula and its resemblance to the model equation is immediately obvious. As discussed before, transformations may be saved for later analysis using the symbolic model formulae (e.g. prediction in original scale), however, this likely requires manual recovery when the API requires design matrix as input.

2.2 Herbicide: Categorical Variable

The `herbicide` data set (original source from R. Hull, Rothamsted Research, data sourced from Welham et al. 2015) contains 135 observations with 1 numerical variable (weight response) and 3 categorical variables: block, herbicide, and population of black-grass with 5, 3 and 9 levels respectively. The experiment employed has a factorial treatment structure (i.e. 27 treatments which are combinations of herbicide and population), with the complete set of treatment randomised within each of the five blocks (i.e. it employs a randomised complete block design).

The model in Figure 2 is a linear model to the square root of the weight of the black-grass with the design matrix $\mathbf{X} = \left[\mathbf{1}_{135}\ \boldsymbol{x}_1\ \cdots\ \boldsymbol{x}_{30}\right]$, where $\boldsymbol{x}_1, ..., \boldsymbol{x}_4$ are dummy variables for `Block` `B2`, `B3` and `B4`, $\boldsymbol{x}_5, ..., \boldsymbol{x}_{12}$ are dummy variables for `Population` `P2` to `P9`, $\boldsymbol{x}_{13}$ and $\boldsymbol{x}_{14}$ are dummy variables for `Herbicide` `B` and `C`, and $\boldsymbol{x}_{15}, ..., \boldsymbol{x}_{30}$ are dummy variables for the corresponding interaction effects. Alternatively, the model can be written via the ANOVA model specification,

$$y_{ijk} = \mu + \gamma_k + \alpha_i + \beta_j + (\alpha\beta)_{ij} + e_{ijk},$$

where index i denotes for level of population, index j for level of herbicide and index k for the replicate block. With dummy variables, the relevant constraints are $\alpha_1 = \beta_1 =$

$\gamma_1 = (\alpha\beta)_{1j} = (\alpha\beta)_{i1} = 0$. This form is equivalent to the linear regression model given in equation (1) with the fixed effects vector

$$\boldsymbol{\beta} = (\mu, \gamma_2, ..., \gamma_5, \alpha_2, \alpha_3, ..., \alpha_9, \beta_2, \beta_3, (\alpha\beta)_{22}, (\alpha\beta)_{23}, ..., (\alpha\beta)_{93})^\top.$$

2.3 Specification of intercept

Wilkinson and Rogers (1973) described many of the operators and evaluation rules associated with symbolic model formulae, that to this day remain a mainstay of `R` as well as other languages. These include simplification rules such as `y ~ x + x` and `y ~ x:x` to `y ~ x`. Their description however did not include any discussion about the intercept. The symbolic evaluation rules governing the intercept are classified as special cases in the current implementation of `R`, although they may not be as overly intuitive on first glance, e.g.

- `y ~ 1:x` simplifies to `y ~ 1`, although one may expect `y ~ x`;
- `y ~ 1*x` simplifies to `y ~ 1`, which may be surprising in light of the proceeding point;
- `y ~ x*1` simplifies to `y ~ x`, which makes the cross operator unsymmetric for this special case.

Further ambiguity arises when we consider cases where we wish to explicitly remove the intercept, e.g.

- `y ~ -1:x` simplifies to the nonsensical `y ~ 1 - 1`, which is equivalent to `y ~ 0`,
- `y ~ 1 + (-1 + x)` simplifies to `y ~ x - 1`.

The last point was raised by N. J. Smith et al. (2018), and subsequently the formula evaluation differs in the `patsy` `Python` library on this particular aspect. These complications arise due to the explicit specification for removing the intercept. Furthermore, the symbolic model formulae that includes `-1` or `0` removes the resemblance to the model equation, detracting from the aim of symbolic model formula to make model formulation straightforward and accessible for practitioners. It should be noted, however, that these cases are all somewhat contrived and would rarely be used in practice.

3 Linear Mixed Models

Consider a n-vector of response $\boldsymbol{y}$, which is modelled as

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{b} + \boldsymbol{e}, \tag{2}$$

where the $\mathbf{X}$ is the design matrix for the fixed effects coefficients $\boldsymbol{\beta}$; $\mathbf{Z}$ is the design matrix of the random effects coefficients $\boldsymbol{b}$, and $\boldsymbol{e}$ is the vector of random errors. We typically assume that the random effects and errors are independent of each other and both multivariate normally distributed,

$$\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{e}\end{bmatrix} \sim N\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}\end{bmatrix}\right)$$

where **G** and **R** are the covariance matrices of $\boldsymbol{b}$ and $\boldsymbol{e}$, respectively.

In Section 4, we present examples with different variables and structures for model (2). In the next sections, we briefly describe and contrast the fitting functions `lmer` and `asreml` from the `lme4`, `asreml` `R`-packages, respectively.

3.1 `lme4`

The `lme4` `R` package fits a LMM with the function `lmer`. The API consists of a *single* formula that extends the linear model formula as follows – the random effects are added by surrounding the term in round brackets with grouping structure specified on the right of the vertical bar, and the random terms within each group on the left of the vertical bar, e.g. `(formula | group)`. The `formula` is evaluated under the same mechanism for symbolic model formula as linear models in Section 2, with `group` specific effects from `formula`. These `group` specific effects are assumed to be normally distributed with zero mean and unstructured variance, as given above in (2). Examples of its use are provided in Section 4.

3.2 `asreml`

In `asreml`, the random effects are specified as another formula to the argument `random`. One of the main strength of LMM specification in `asreml`, in contrast to `lme4` in wide array of flexible covariance structures. The full list of covariance structures available in `asreml` Version 3 are given in Butler, Cullis, et al. (2009); `asreml` version 4 has some slight differences as outlined in Butler, Gogel, et al. (2018), although the main concept is similar: variance structures are specified with function-like terms in the model formulae, e.g. `us(factor)` will fit a `factor` effect with unstructured covariance matrix; `diag(factor)` will fit a `factor` effect with diagonal covariance matrix, i.e. zero off-diagonal and different parameterisation in the diagonal elements. Note `factor` corresponds to a categorical variable in the data; see Section 4 for examples of its usage.

4 Motivating Examples for LMMs

This section presents motivating examples with model specification by `lmer` or `asreml`. It should be noted that the models are not advocated to be "correct", but rather a plausible model that a practitioner may consider in light of the data and design. For succinctness, we omit all `data` argument to model fit functions. Also, this paper uses `lme4` version 1.1.21; `pedigreemm` version 0.3.3 and `asreml` version 3.

4.1 Chicken Weight: Longitudinal Analysis

The chicken weight data is originally sourced from Crowder and Hand (1990) and found as a built-in data set in `R`. It consists of the weights of 50 chickens tracked over regular time intervals (not all weights at each time points are observed). Each chicken are fed one of 4 possible diets.

In this experiment, we are interested in the influence of different diets on chicken weight. We can model the weight of each chicken over time that includes diet effect, overall intercept and slope for time. Fitting these effects as fixed and assuming that the error is i.i.d. means that the observations from same chicken are uncorrelated and there is no variation for the intercept and slope between chickens. This motivates the inclusion of random intercept and random slope for each chicken. More explicitly, and using an ANOVA model specification, the weight may be modelled as

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \alpha_{T(i)} + b_{0i} + b_{1i} x_{ij} + e_{ij}, \tag{3}$$

where y_{ij} is the weight of the i-th chicken at time index j, x_{ij} is the days since birth at time index j for the i-th chicken, b_{0i} and b_{1i} are the random intercept and random time slope effects for the i-th chicken, β_0 and β_1 are the overall fixed intercept and fixed time slope, and e_{ij} is the random error.

The above model is incomplete without distributional assumptions for the random effects. As intercept and slope clearly measure different units, the variance will be on different scales. Furthermore, we make an assumption that the random intercept and random slope are correlated within the same chicken, but independent across chickens. With the typical assumption of mutual independence of random effects and random error, and normally and identically distributed (NID) effects, we thus have the distribution assumptions,

$$\begin{bmatrix} b_{0i} \\ b_{1i} \end{bmatrix} \sim NID\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_0^2 & \sigma_{01} \\ \sigma_{01} & \sigma_1^2 \end{bmatrix}\right) \qquad \text{and} \qquad e_{ij} \sim NID(0, \sigma^2). \tag{4}$$

If the effects in model (3) are vectorised as in model (2) with

$$\boldsymbol{b} = (b_{01}, b_{02}, ..., b_{0,50}, b_{11}, b_{12}, ..., b_{1,50})^\top \quad \text{and} \quad \boldsymbol{e} = (e_{ij}),$$

then the model assumption (4) can also be written as

$$\boldsymbol{b} \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_0^2 & \sigma_{01} \\ \sigma_{01} & \sigma_1^2 \end{bmatrix} \otimes \mathbf{I}_m\right) \quad \text{and} \quad \boldsymbol{e} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

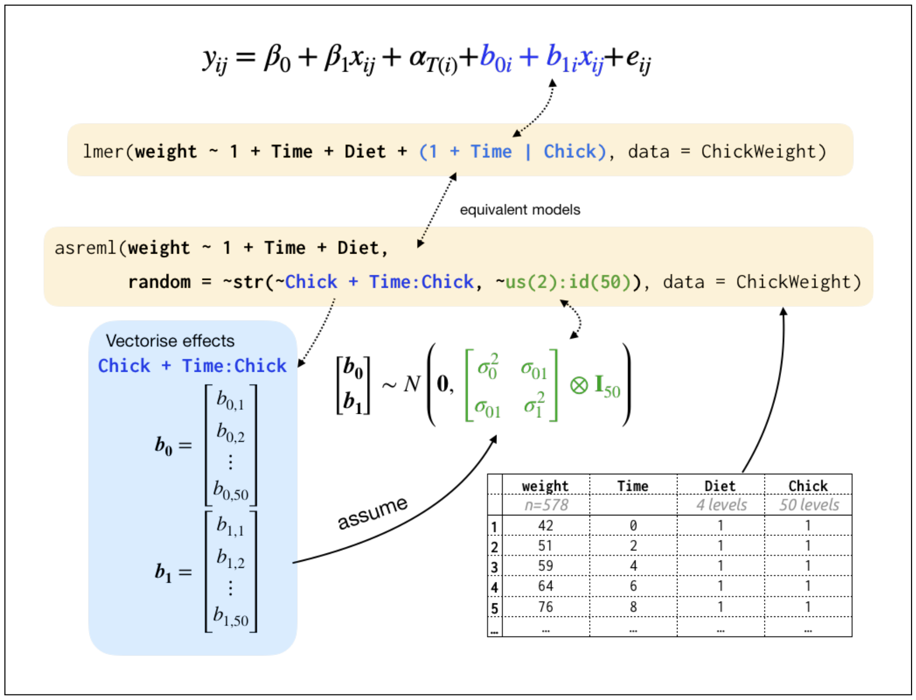
where $\otimes$ is the Kronecker product. In `asreml`, a separable covariance structure, $\boldsymbol{\Sigma}_1 \otimes \boldsymbol{\Sigma}_2$, is specified by the use of an interaction operator where the dimensions and structures of $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are specified by the `factor` input or its number of levels and the function that wraps the `factor`, e.g. `us(2):id(50)` is equivalent to $\boldsymbol{\Sigma}_{2\times2} \otimes \mathbf{I}_{50}$ where $\boldsymbol{\Sigma}_{2\times2}$ is a 2×2 unstructured covariance matrix.

The symbolic model formulae that encompasses the model (3) coupled with assumption in (4) for `lmer` and `asreml` are shown in Figure 3. The two symbolic model formulae share the same syntax for fixed effects, however, in this case the random effects syntax is more verbose for `asreml`.

One may wish to modify their assumption such that now we assume

$$\boldsymbol{b} \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} \otimes \mathbf{I}_m\right),$$

That is, the random slope and random intercept are assumed to be uncorrelated. This uncorrelated model may be specified in `lme4` by replacing `|` with `||` as below.



$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \alpha_{T(i)} + b_{0i} + b_{1i}x_{ij} + e_{ij}$$

```
lmer(weight ~ 1 + Time + Diet + (1 + Time | Chick), data = ChickWeight)
```

```
asreml(weight ~ 1 + Time + Diet,
       random = ~str(~Chick + Time:Chick, ~us(2):id(50)), data = ChickWeight)
```

Vectorise effects
`Chick + Time:Chick`

$$\boldsymbol{b}_0 = \begin{bmatrix} b_{0,1} \\ b_{0,2} \\ \vdots \\ b_{0,50} \end{bmatrix} \quad \boldsymbol{b}_1 = \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,50} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_0^2 & \sigma_{01} \\ \sigma_{01} & \sigma_1^2 \end{bmatrix} \otimes \mathbf{I}_{50}\right)$$

	weight	Time	Diet	Chick
	n=578		*4 levels*	*50 levels*
1	42	0	1	1
2	51	2	1	1
3	59	4	1	1
4	64	6	1	1
5	76	8	1	1
...	...	...	...	...

Fig. 3. This figure shows a longitudinal analysis of the chicken data (see Section 4.1). The index form of the model equation shows direct resemblance for symbolic model formula in `lmer` for the fixed and random effects, however, its covariance form is not as easily inferred. In contrast, the symbolic model formula in `asreml` show resemblance of the covariance structure specified in the second argument of `~str`, however, the corresponding random effects specified in the first argument of `~str` must be vectorised as show in the above figure and requires implicit knowledge of the Kronecker product of relevant matrices.

```
lmer(weight ~ 1 + Time + Diet + (1 + Time || Chick))
```

It should be noted that the effects specified on the LHS of the `||` are uncorrelated if the variables are numerical only; we refer to the example in Table 1 for a case where this does not work when the variable is a factor.

```
lmer(weight ~ 1 + Time + Diet + (1 | Chick) + (0 + Time | Chick))
```

The same model is specified as below for `asreml` where now `us(2)` is replaced with `diag(2)`. The correspondence to the covariance structure is more explicit, but again involves the random effects being (implicitly) vectorised as show in Figure 3 and care is needed with orders of separable structure.

```
asreml(weight ~ 1 + Time + Diet,
     random=~ str(~Chick + Chick:Time, ~diag(2):id(50)))
```

4.2 Field Trial: Covariance Structure

In this example, we consider wheat yield data sourced from the agridat R-package (Wright 2018), which originally appeared in Gilmour, Cullis, and Verbyla (1997). This data set consists of $n = 330$ observations from a near randomised complete block experiment with $m = 107$ varieties, of which 3 varieties have 6 replicates while the rest have 3 replicates. The field trial that the yield data was collected from was laid out in a rectangular array with $r = 22$ rows and $c = 15$ columns. Each of the variety replicates are spread uniformly to $b = 3$ blocks. The columns 1-5, columns 6-10 and columns 11-15 form three equal blocks of contiguous area within the field trial. The data frame `gilmour.serpentine` contains the columns for `yield`, `gen` (variety), `rep` (block), `col` (column) and `row`. Further columns `colf` and `rowf`, which are factor versions of `col` and `row`, have also been added.

We may model the yield observations $\boldsymbol{y}$, ordered by the rows within columns, using the model (2) where here $\boldsymbol{\beta}$ is the b-vector of replicate block effects and $\boldsymbol{b}$ is the m-vector of variety random effects. We consider next a few potential covariance structures for $\boldsymbol{b}$ and $\boldsymbol{e}$.

Scaled identity structure. One of the simplest assumptions to make would be to assume that $var(\boldsymbol{b}) = \mathbf{G} = \sigma_g^2\mathbf{I}_m$ i.e., a scaled identity structure. We may additionally assume that $var(\boldsymbol{e}) = \sigma^2\mathbf{I}_n$. In `lmer`, this is fitted as below.

```
lmer(yield ~ 0 + rep + (1 | gen))
```

To elaborate further, `lmer` specifies a random intercept for each variety. This variety intercept will each be assumed to arise from $NID(0, \mathbf{\Sigma}_{1\times 1})$ where $\mathbf{\Sigma}_{1\times 1}$ is a 1×1 unstructured variance matrix (essentially a single parameter variance component).

The same model is fitted in `asreml` as below. Particularly, `idv(gen)` signifies a vector of variety effects with `idv` variance structure, i.e. a scaled identity structure. This is the default structure in `asreml`, and so omitting variance structure, `random = ~gen`, results in the same fit.

```
asreml(yield ~ 0 + rep, random = ~idv(gen))
```

Crossed random effects. Field trials often employ rows and/or columns as blocking factors in the experimental design. Furthermore, it is common practice that the management practices of field experiments follow some systematic routine, e.g., harvesting may occur in a serpentine fashion from the first to the last row. These occasionally introduce obvious unwanted noise in the data that are often removed by including random row or column effects assuming that they are i.i.d. for simplicity. These so-called crossed random effects are fitted as below for `lmer` and `asreml`.

```
lmer(yield ~ 0 + rep + (1 | gen) + (1 | rowf) + (1 | colf))
```

```
asreml(yield ~ 0 + rep, random = ~idv(gen) + idv(rowf) + idv(colf))
```

Error covariance structure. A field trial is often laid out in a rectangular array and observations from each plot indexed by row and column within this array. Consequently, the assumption that $var(\boldsymbol{e}) = \sigma^2\mathbf{I}_n$ may be restrictive when there is likely to be some sort of spatial correlation, i.e. plots that are geographically closer would be similar than plots further apart. A range of models may be considered for this potential correlation. In practice, a separable autoregressive process of order one, denoted AR1×AR1, has worked well as a compromise between parsimony and flexibility as a structure (Gilmour, Cullis, and Verbyla 1997). More specifically, we assume $var(\boldsymbol{e}) = \sigma^2\mathbf{\Sigma}_c \otimes \mathbf{\Sigma}_r$ where $\mathbf{\Sigma}_c$ is a $c \times c$ matrix with (i, j)-th entry of $\mathbf{\Sigma}_c$ given as $\rho_c^{|i-j|}$ with autocorrelation parameter ρ_c, and a similar definition holds for $r \times r$ matrix $\boldsymbol{\Sigma}_r$ except the autocorrelation parameter is denoted bvy ρ_r. This model is fitted in `asreml` by supplying a symbolic formula, `ar1(colf):ar1(rowf)`, to the argument `rcov` as below.

```
asreml(yield ~ 0 + rep, random = ~gen, rcov = ~ar1(colf):ar1(rowf))
```

Here, the `ar1` specifies an autoregressive process of order one with dimension given by number of levels in `rowf` and `colf`. It is important to note that `ar1` denotes a correlation matrix and a covariance matrix may be specified by `ar1v`. Care needs to be taken in covariance specification for separable models, as clearly there is a lack of variance parameter(s) where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are both correlation structures only, while if both are covariance structure then the model is over-parameterised and unidentifiable. In the error structure of `rcov`, this is taken care of such that `rcov = ~ar1v(colf):ar1(rowf)`, `rcov = ~ar1(colf):ar1v(rowf)` and `rcov = ~ar1(colf):ar1(rowf)` will fit all the same model. It should be noted that this is not the case for separable covariance structures specified in `random` effects.

In comparison, the more restrictive API of `lmer` function does not allow the assumption on the random effects to be relaxed from $var(\boldsymbol{e}) = \sigma^2\mathbf{I}_n$. One may of course introduce a random effect, $\boldsymbol{b}_e \sim N(\mathbf{0}, \sigma^2\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r)$, and assume $\boldsymbol{e} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$. However, this separable covariance structure also can not be specified within `lmer` function.

Known covariance structure. Often in plant breeding trials, the varieties of interest have some shared ancestry. This is captured in the form of pedigree data that contains 3 columns: individual ID, mother's ID and father's ID. The related structure is commonly captured by the use of a numerator relationship matrix, denoted here has $\mathbf{A}$ (Mrode 2014). For example, suppose that individuals i and j are full-siblings. Then the corresponding (i, j)-th entry in $\mathbf{A}$ is 0.5 (i.e., the average probability that a randomly drawn allele from individual i is identical by descent to the randomly drawn allele at the same autosomal locus from individual j).

With the additional information above, we may assume that $var(\boldsymbol{b}) = \sigma_g^2\mathbf{A}$ to exploit this *known* relatedness structure between varieties. The symbolic model formulae in `lme4` alone is unable to specify this model and, an extension `R` package `pedigreemm` (Vazquez et al. 2010) is required. The pedigree data is parsed to make an object of `pedigree` class, which we refer to here as `ped`. This object `ped` is then included as part of the input in the main fitting function `pedigreemm`, as depicted below.

```
pedigreemm(yield ~ 0 + rep + (1 | gen),
     pedigree = list(gen = ped))
```

In `asreml`, the fit is similar to the above, but the factor with the known covariance structure must be wrapped in `giv` with argument `ginverse` providing a named list with the inverse of the A in a sparse format, i.e. a data frame of 3 columns that consists the row and the column index of A and its corresponding value in A provided that the value is non-zero.

```
asreml(yield ~ 0 + rep, random = ~giv(gen),
     ginverse = list(gen = Ainv))
```

4.3 Multi-Environmental Trial: Separable Structure

In the final example, we consider CIMMYT Australia ICARDA Germplasm Evaluation (CAIGE) bread wheat yield 2016 data (CAIGE 2016), which consists of $t = 7$ sites across Australia, where the overall aim is to select the best genotype (`gen`). There were $m = 240$ genotype tested across seven trials and 252-391 plots, with a total of $n = 2127$ yield observations. Each trial employed a partially replicated (p-rep) design (Cullis, A. B. Smith, and Coombes 2006), with p ranging from 0.23 to 0.39.

Fitting a model to a model should take into account the differential mean yield across sites, and allow for different genotypic variations by site. For simplicity, we ignore other variations for now. In turn, the LMM formulation in equation (2) may be used where $\boldsymbol{y}$ is the vector of yield (ordered rows within columns within sites); $\boldsymbol{\beta}$ is the t-vector of site effects; and $\boldsymbol{b}$ is the mt-vector of genotype-by-site effects (ordered by genotype within site). There are a number of distributions that may be considered for $\boldsymbol{b}$, as explained below.

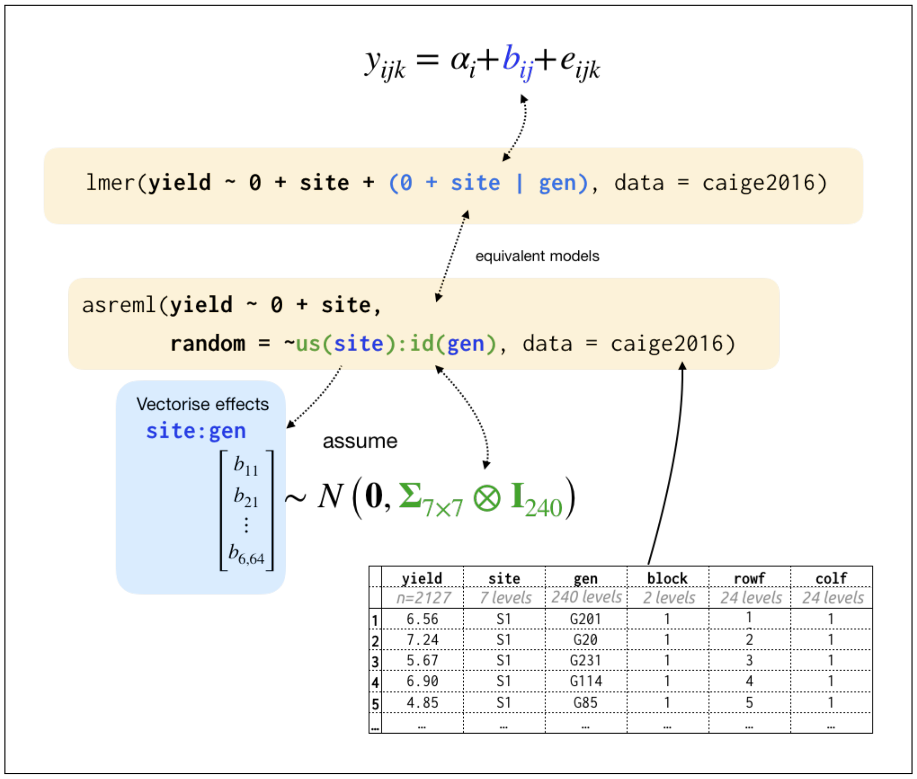
We may consider a separable model such that $\boldsymbol{b} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_s \otimes \boldsymbol{\Sigma}_g)$, where $\boldsymbol{\Sigma}_s$ and $\boldsymbol{\Sigma}_g$ are a $t \times t$ and $m \times m$ matrices, respectively. We may further assume that $\boldsymbol{\Sigma}_g$ has a known structure similar to Section 4.2, but for simple illustration here we will assume that the genotypes are independent, i.e. $\boldsymbol{\Sigma}_g = \mathbf{I}_m$. Also, we may assume that $\boldsymbol{\Sigma}_s = \text{diag}\left(\sigma^2_{g1}, \sigma^2_{g2}, \cdots, \sigma^2_{gt}\right)$, i.e. a diagonal matrix with different variance paramterisation for each site, thus allowing for different genotypic variance at each site. This model can be fitted as below in `asreml`.

```
asreml(yield ~ 0 + site, random = ~ diag(site):id(gen))
```

The same model in `lmer` is somewhat more involved as shown in Table 1.

The diagonal model assumes that genotype-by-site effects are uncorrelated across sites for the same genotype. However, a more realistic assumption is to assume that these effects are correlated, thus allowing for different correlation of genotype effects between pair of sites, i.e. we assume that $\boldsymbol{\Sigma}_s$ is an unstructured covariance matrix. The specification of such model for `lmer` and `asreml` is shown in Figure 4.

A even more realistic model may consider including site-specific random row or column effects, and assuming an AR1×AR1 process for the error covariance at each site



$$y_{ijk} = \alpha_i + b_{ij} + e_{ijk}$$

```
lmer(yield ~ 0 + site + (0 + site | gen), data = caige2016)
```

```
asreml(yield ~ 0 + site,
       random = ~us(site):id(gen), data = caige2016)
```

$$\begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{6,64} \end{bmatrix} \sim N\left(\mathbf{0}, \mathbf{\Sigma}_{7\times 7} \otimes \mathbf{I}_{240}\right)$$

	yield	site	gen	block	rowf	colf
	n=2127	7 levels	240 levels	2 levels	24 levels	24 levels
1	6.56	S1	G201	1	1	1
2	7.24	S1	G20	1	2	1
3	5.67	S1	G231	1	3	1
4	6.90	S1	G114	1	4	1
5	4.85	S1	G85	1	5	1
...	...	...	...	...	...	...

Fig. 4. Depiction of the fit of a simplified LMM for the analysis of the MET data. In modelling the `site`-by-`gen` random effect, the variance structure are specified differently using `lmer` and `asreml`, where latter shows resemblances of covariance structure written mathematically and when all random effects are vectorised and concatenated, while the former requires some additional computation.

Table 1. The table lists the equivalent symbolic model formula in `lmer` and `asreml` for the site-by-genotype random effect, $\boldsymbol{b}$ and the corresponding mathematical form of the variance structure of $\boldsymbol{b}$. Here, $\boldsymbol{\Sigma}_{t\times t}$ is a $t \times t$ unstructured covariance matrix; $\mathbf{D}_{t\times t} = \text{diag}(\sigma^2_{g1}, ..., \sigma^2_{g7})$, a $t \times t$ diagonal covariance matrix; m is the number of genotypes; t is the number of sites; and `S1` is a n-vector where the entry is one if the corresponding observation belongs to site 1 and zero otherwise (similar definitions hold for `S2`, ..., `S7`). The conversion of the factor `site` to numerical variables `S1`, ..., `S7` is required to have uncorrelated random effects in `lmer` via the `||` operator, as per the last row in the table. The `||` group separation in `lmer` is only effective when variables on LHS are numerical.

`lmer`	`asreml`	$var(\boldsymbol{b})$

`(1 | site:geno)`	`idv(site):id(geno)`
	`id(site):idv(geno)`
	`site:geno`
		$\sigma^2_g \mathbf{I}_{tm}$

`(0 + site | geno)`
`(0 + site || geno)`
`(0 + S1 + S2 + S3 + S4 + S5 + S6 + S7 | geno)`
	`us(site):id(geno)`	$\boldsymbol{\Sigma}_{t\times t} \otimes \mathbf{I}_t$

`(0 + S1 + S2 + S3 + S4 + S5 + S6 + S7 || geno)`	`diag(site):id(geno)`	$\mathbf{D}_{t\times t} \otimes \mathbf{I}_t$

as in Section 4.2. These are easily included in `asreml` using the `at` function within the symbolic model formulae. For example, the inclusion of random row effect at site `S1` only and AR1×AR1 processes for the error covariance at each site is shown below.

```
asreml(yield ~ 0 + site,
       random = ~us(site):id(gen) + at(site, "S1"):idv(rowf),
       rcov = ~at(site):ar1(colf):ar1(rowf))
```

The above model cannot be specified using `lmer`.

5 Discussion

In fitting statistical models, the user may not necessary understand the full intricacies of model fitting process. However, it is essential that the user understands how to specify the model that they wish to fit and the interpretation from the fit. Symbolic model formulae is a way of bridging the gap between software and mathematical representation of the model, and has been extensively employed in R for this purpose.

In this article, we have extensively compared two widely used LMM R-packages with contrasting model specification in functions: `lmer` and `asreml`. Both of these functions use symbolic model formulae to specify the model with `lmer` taking a more hierarchical approach to random effects specification, while `asreml` focuses on the covariance structure of the vectorised random effects (and the data for the matter). There are strength and weakness in both approaches as we discuss next.

It is clear from Section 4.1 that a random intercept and random slope model is verbose using the symbolic model formulae of `asreml`. Specifically, the random effect symbolic formula contains a function `str` that takes input of two other formula: the first input specifying the random effects, and second input specifying the covariance structure of the vectorised form of random effects specified in the first input. The second input also requires the dimension(s) of covariance structure as input. These number may need manual update when the data is subsequently updated, thus making this symbolic model formula clumsy to use.

On the other hand, the flexibility of `asreml` is evident in Sections 4.2 and 4.3, where the LMMs fitted are less easy to pose hierarchically, but the vectorised version of the LMM remains straightforward provided one knows how to establish the set up the structure of the covariance matrices. Put another way, the vast set of in-built pre-defined covariance structures in `asreml` (e.g. scaled identity, diagonal structure, unstructured, autoregressive process), along with the capacity to modify the error covariance structure and incorporate separable structures makes the model specification embedded in `asreml` a superior choice here. There are many more pre-defined covariance structures not demonstrated in this paper, and interested readers may refer to Butler, Cullis, et al. (2009) and Butler, Gogel, et al. (2018). By contrast, the lack of flexibility in `lme4` means that either a more obtuse workaround is required or the precise LMM can not be formulated at all.

That being said, the Bürkner (2017) and Bürkner (2018) (`brms`) make extensive discussion of symbolic model formula and extends on the framework built on `lme4`. The `brms` model formulae resembles `lme4` and many symbolic model formulae in our examples will be similar. The `brms` `R`-package uses a Bayesian approach to fit its models and model specification require further discussion on specifying priors. These discussions are left for future review, although we acknowledge that such extensions may well resolve some of the current limitations of `lme4` and bridge its gap in flexibility with `asreml`.

Symbolic model formulae in `R` is widely used and frameworks to specify mixed models by `lme4` and `asreml` (version 3) used for many years. This makes drastic changes difficult for these frameworks. Based on our review, we argue that ideally any new framework for symbolic model formulae should require intercepts to be specified explicitly. As discussed in Section 2.3, the default inclusion or explicit removal of intercepts removes the resemblance of symbolic model formulae to the model equation. Currently, the implicit inclusion of intercepts makes certain model formulation unclear and inconsistent across different LMM specifications, e.g. `(Time | Chick)` in `lmer` includes random intercept (and slope) for `Chick`, but the equivalent formulation `str(~Chick + Chick:Time, ~diag(2):id(50))` in `asreml` does not include the random intercept.

There is a trade-off between different types of symbolic model formulae: `lmer` syntax is no doubt less flexible and may be less intuitive to some, however, with a degree of familiarity pertains as a higher level language for symbolic model formula. For many hierarchical models, the formulation is more elegant and simpler than `asreml`. However, `asreml` is more flexible to specify variety of covariance matrices. This strength is predicated on having a deeper understanding of random effects and its covariance structure, and promotes the view of the LMM in a fully vectorised form.

Acknowledgement

This paper benefited from twitter conversation with Thomas Lumley. This paper is made using `R` Markdown (Xie, Allaire, and Grolemund 2018). Huge thanks goes to the teams behind `lme4` and `asreml` `R`-packages that make fitting of general LMMs accessible to wider audiences. All materials used to produce this paper and its history of changes can be found on github `https://github.com/emitanaka/paper-symlmm`.

References

Aitkin, Murray et al. (1989). *Statistical Modelling in GLIM*. Oxford University Press.

Bates, Douglas et al. (2015). “Fitting Linear Mixed-Effects Models using lme4”. In: *Journal of Statistical Software* 67.1. DOI: `10.18637/jss.v067.i01`. eprint: `1406.5823`.

Buitinck, Lars et al. (2013). “API design for machine learning software: experiences from the scikit-learn project”. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Bürkner, Paul-Christian (2017). “brms: An R Package for Bayesian Multilevel Models Using Stan”. In: *Journal of Statistical Software* 80.1, pp. 1–28.

— (2018). “Advanced Bayesian Multilevel Modeling with the R Package brms”. In: *The R Journal* 10.1, pp. 395–411. DOI: `10.32614/RJ-2018-017`. URL: `https://doi.org/10.32614/RJ-2018-017`.

Butler, David Geoffrey, Brian R Cullis, et al. (2009). *Mixed models for S language environments ASReml-R reference manual*.

Butler, David Geoffrey, Beverley J Gogel, et al. (2018). *Navigating from ASReml-R Version 3 to 4*.

CAIGE (2016). *CAIGE Project*. URL: `http://www.caigeproject.org.au`.

Chambers, John M. and Trevor J. Hastie (1993). “Statistical Models in S”. In: 2. Chap. 2, p. 227. DOI: `10.2307/1269676`.

Crowder, M. and D. Hand (1990). *Analysis of Repeated Measures*. Chapman and Hall. URL: `http://www.python.org`.

Csárdi, Gábor (2019). *cranlogs: Download Logs from the ’RStudio’ ’CRAN’ Mirror*. R package version 2.1.1. URL: `https://CRAN.R-project.org/package=cranlogs`.

Cullis, Brian R, Alison B Smith, and Neil Edwin Coombes (2006). “On the design of early generation variety trials with correlated data”. In: *Journal of Agricultural, Biological, and Environmental Statistics* 11.4, pp. 381–393. ISSN: 1085-7117. DOI: `10.1198/108571106X154443`.

Gilmour, Arthur R, Brian R Cullis, and Arūnas P Verbyla (1997). “Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments”. In: *Journal of Agricultural, Biological, and Environmental Statistics* 2.3, pp. 269–293. DOI: `10.2307/1400446`.

Gilmour, Arthur R, Beverley J Gogel, et al. (2009). *ASReml user guide release 3.0*.

Kuhn, Max (2018). *parsnip: A Common API to Modeling and analysis Functions*. R package version 0.0.0.9003. URL: `https://topepo.github.io/parsnip`.

Mrode, Raphael A (2014). *Linear Models for the Prediction of Animal Breeding Values*. Ed. by Sarah Hulbert. 3rd ed. X: CABI. ISBN: 1780643918, 9781780643915. DOI: `10.1017/CBO9781107415324.004`. arXiv: `arXiv:1011.1669v3`.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python ”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pinheiro, Jose et al. (2019). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-140. URL: `https://CRAN.R-project.org/package=nlme`.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Ryan, T. A., B. L. Joiner, and B. F. Ryan (1976). *The Minitab Student Handbook*. Duxbury Press.

Seabold, Skipper and Josef Perktold (2010). “Statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*.

Smith, Nathaniel J. et al. (Oct. 2018). *pydata/patsy: v0.5.1*. Version v0.5.1. DOI: `10.5281/zenodo.1472929`. URL: `https://doi.org/10.5281/zenodo.1472929`.

Stan Development Team (2019). *RStan: the R interface to Stan*. R package version 2.19.2. URL: `http://mc-stan.org/`.

Van Rossum, Guido and Fred L Drake Jr (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands. URL: `http://www.python.org`.

Vazquez, A. I. et al. (2010). “Technical Note: An R package for fitting generalized linear mixed models in animal breeding”. In: *Journal of Animal Science* 88, pp. 497–504.

VSN International (2017). *Genstat for Windows 19th Edition*. VSN International, Hemel Hempstead, UK. URL: `Genstat.co.uk`.

Welham, Sue J et al. (2015). *Statistical Methods in Biology: Design and analysis of experiments and regression*. Chapman and Hall.

Wickham, Hadley et al. (2019). *dplyr: A Grammar of Data Manipulation*. R package version 0.8.3. URL: `https://CRAN.R-project.org/package=dplyr`.

Wilkinson, G N and C E Rogers (1973). “Symbolic Description of Factorial Models for Analysis of Variance”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 22.3, pp. 392–399.

Wright, Kevin (2018). *agridat: Agricultural Datasets*. R package version 1.16. URL: `https://CRAN.R-project.org/package=agridat`.

Xie, Yihui, J.J. Allaire, and Garrett Grolemund (2018). *R Markdown: The Definitive Guide*. ISBN 9781138359338. Boca Raton, Florida: Chapman and Hall/CRC. URL: `https://bookdown.org/yihui/rmarkdown`.