Symbolic Formulae for Linear Mixed Models*

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

[1] The University of Sydney, Camperdown NSW 2008, Australia
dr.emi.tanaka@gmail.com
[2] Australian National University, Acton ACT 2601, Australia
francis.hui@anu.edu.au

Abstract. A statistical model is a mathematical representation of often a simplified or idealised data-generating process. A particular type of statistical model that is widely used is linear mixed models (LMMs), also called multi-level, nested, hierarchical or panel data models. LMMs are used widely in a variety of disciplines, e.g. agriculture, ecology, econometrics, psychology and so on, owing to its flexbility of accounting for complex correlated strucures in data. This flexibility, however, have given rise to a number of ways to specify the LMMs in order to fit it via an application software. In this paper, we review the software design of LMM (and its special case, the linear models) specification, in particular with the use of symbolic model formulae, with focus on the LMM specification in popular but contrasting `lme4` and `asreml` R-packages.

Keywords: multi-level model · hierarchical model · panel data 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 which is ubiquitous in many data analyses. These models are fitted or trained computationally, often with practitioners using some readily available application software or 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 focus user's time on important tasks, e.g. interpreting or using the fitted model.

While there are an abundance of application software for fitting a variety of statistical models, the API is largely inconsistent and restrictive in some fashion. For example, in linear models, the intercept may or may not be included by default; and random error is assumed to be identical and independently distributed (i.i.d) with no option to modify these assumption. These inconsistencies and restrictiveness in the API cause great friction to fit the user's desired models. 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 preditive modelling functions (e.g. random forest, logistic regression, linear regressoin etc) and `scikit-learn` library (Pedregosa et al. 2011)

* Supported by R Consortium

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 linear mixed models (LMMs).

LMMs (also called hierarchical, panel data, nested or multi-level models) are widely used across many disciplines (e.g. ecology, psychology, agriculture, finance etc) due to its flexibility to model complex, correlated structures in the data. This flexibility is primarily achieved by the random effects and its variance-covariance structure - it is this flexibility, however, that results in major difference in model specification. In R, arguably the most popular general purpose package to fit LMMs is `lme4` (Bates et al. 2015). Total downloads from RStudio CRAN mirror from `cranlogs` (Csárdi 2019) indicate there were over 2 million downloads for `lme4` in the whole of 2018 whilst other popular mixed model packages (e.g. `nlme`, Pinheiro et al. 2019; `rstan`, Stan Development Team 2019; `brms`, Bürkner 2017, 2018) in the same year have less than half a million downloads. Another general purpose linear mixed model package is `asreml` (Butler, Cullis, et al. 2009), which wraps the proprietary software ASreml (Gilmour et al. 2009) into the R framework. As this package is not available on CRAN, there is no comparable download logs, although, citations of its technical document indicates popular usage. In this paper, we discuss only `lme4` and `asreml` due to its active maintainance, maturity and contrasting approaches.

Both the `lme4` and `asreml` packages employ a symbolic model formula as part of its API to specfiy the model. 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 model formulae for linear models was applied in Genstat (VSN International 2017) and GLIM (Aitkin et al. 1989) with description by Wilkinson and Rogers (1973). 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 mainly in the representation of random effects, with the additional need to specify the variance-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 model formula in linear models.,We then describe the symbolic model formula employed in the linear mixed model `lme4` and `asreml` R packages 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 `lme4` and `asreml` in Section 4. We conclude with summary and discussion in Section 5.

2 Symbolic Formulae for Linear Models

A special case of linear mixed models is the linear (fixed) models where the model comprises of only fixed effects and a single random term (i.e. the error) given in a matrix

notation as

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

where $\boldsymbol{y}$ is a n-vector of response, $\mathbf{X}$ is the $n \times p$ design matrix for associated p-vector of fixed effects $\boldsymbol{\beta}$ and $\boldsymbol{e}$ is the n-vector of random error. Typically we assume $\boldsymbol{e} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$ although this assumption is unnecessary for fitting the model under least squares.

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 symbolic model formula along with data frame that define the variables in the formula. The input of data frame may be optional if the variables are defined in the parental environment, although we do not recommend such an approach due to larger potential of error (e.g. one variable sorted while others are not).

Symbolic model formulae is heavily used to specfiy 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 - this type of use is beyond 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 (Smith et al. 2018) imitating R's formula was introduced in 2011 and used in `Statsmodels` library (Seabold and Perktold 2010) to fit a statistical model. We will limit the discussion of symbolic model formulae to those implemented in R for the rest of the paper, however, it is important to note that the conceptual framework is not limited to R.

Symbolic model formulae makes use of the variable names defined in the environment (usually through the data frame) in the specification of linear models. The left hand side (LHS) indicate the response; the right hand side (RHS) resemble its mathematical form; and the LHS and RHS is separated by ~ which can be read as "modelled by". E.g. the symbolic model formula `y ~ 1 + x` can be thought of as vector `y` is modelled by a linear predictor consisting of an overall intercept and variable `x`.

When the variables are numerical then the connection between the formula to its regression equation is obvious - LHS is $\boldsymbol{y}$ and RHS correspond to the columns of the design matrix $\mathbf{X}$ in the linear model (1). 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 when the input arguments are the design matrix and the corresponding response – now there is an extra step required by the user to transform the data before model fitting. Such manual tranformation likely results in manual back-tranformation later in the analysis pipeline. 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.

The specification of the intercept by `1` in the formula as done in Figure 1 is unnecessary in R as this is included by default. The removal of intercept is required 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 engrained within R, we recommended 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 discussion on this.

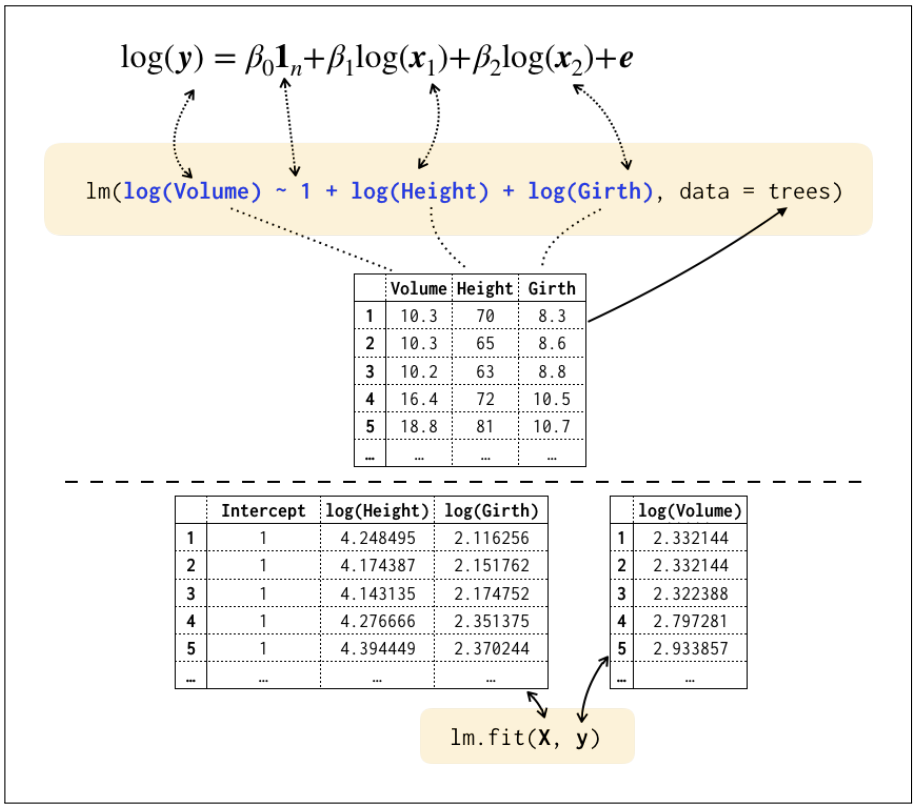

Fig. 1. There are two main approaches to fitting a linear model illustrated above with the fit of a linear model to the `trees` dataset: (1) top uses the `lm` function in R with input argument as a symbolic model formulae (in blue) and (2) bottom uses the `lm.fit` function in R that requires input of design matrix and the response. The latter approach is scarcely used in R, however, it is the common approach in other languages. The connection of the data column names to symbolic model formula and its resemblance to the model equation is immediately obvious. Tranformations may be automatically for later analysis for top approach (e.g. prediction in original scale) however this likely requires manual recovery for the bottom approach. See Section 2.1 about the data and the model.

Categorical variables are typically converted to a set of dummy variables consisting of 0s and 1s indicating whether the corresponding observation belongs to the respective categorical level. For identiability, some contraint is applied, e.g. the treatment constraint will estimate effects in comparison with a particular reference group (the default behaviour in R). 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 ANOVA model form as illustrated in Figure 2 with the fit of two-way ANOVA model with interaction to the `herbicide` data.

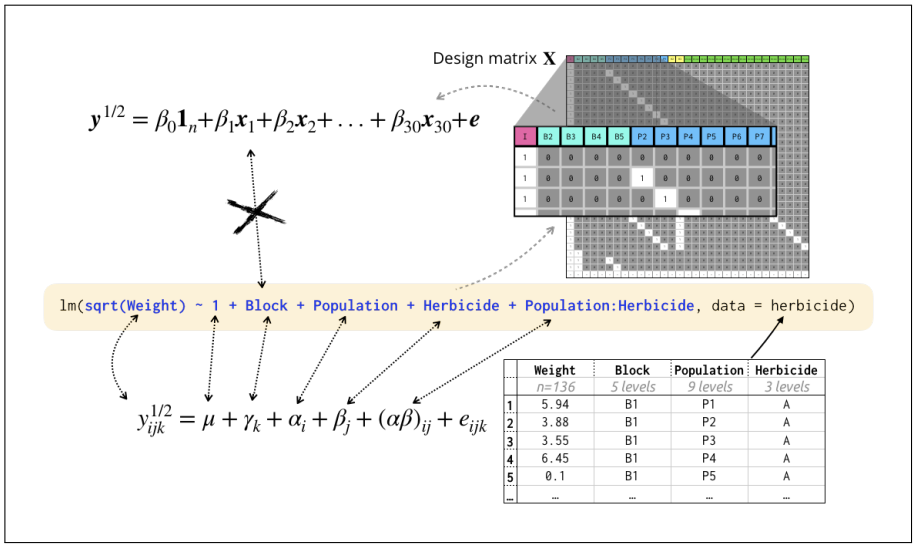

Fig. 2. In the presence of categorical variable, the resemblance of the symbolic model formulae to its regression model form is not immediately obvious. In this case, categorical variables are transformed to a set of dummy variables with constraint applied for identifiability. As such, a single categorical variable span a number of columns in the design matrix. If the model equation is written in the form of ANOVA model (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 specfied easily with symbolic model formula by use of `:` operator as seen in Figure 2. The formula in Figure 2 can also be written more compactly as `sqrt(Weight) ~ 1 + Block + Population * Herbicide` where `*` operator fits is the short hand for main effects and the interaction effects. Further shorthand exists for her 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, 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

as `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
## <environment: 0x7fac64b0ef28>
```

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 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 height of the tree and $\boldsymbol{x}_2$ is the diameter (named `Girth` in the data) of the correponding tree; and $\boldsymbol{y}$ is the volume of the tree.

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 crosses 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 fits a linear model to the square root of the weight of the black-grass with design matrix $\mathbf{X} = \left[\mathbf{1}_{135}\ \boldsymbol{x}_1 \cdots \boldsymbol{x}_{30}\right]$ where $\boldsymbol{x}_1, ..., \boldsymbol{x}_4$ [FILL]. Alternatively, the model can be written as the so-called ANOVA model

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

where index i denotes for level of population; index j for level of herbicide and indicate k for the replicate block with constraints $\alpha_1 = \beta_1 = \gamma_1 = (\alpha\beta)_{1j} = (\alpha\beta)_{i1} = 0$. This form is equivalent with the linear regression model given in (1) when fixed effects $\boldsymbol{\beta} = (\mu, \gamma_2, ..., \gamma_5, \alpha_2, \alpha_3, ..., \alpha_9, \beta_2, \beta_3, (\alpha\beta)_{22}, (\alpha\beta)_{23}, ..., (\alpha\beta)_{93})^\top$.

2.3 Specification of intercept

Wilkinson and Rogers (1973) describes many of the operators and evaluation rules with symbolic model formulae that is implemented in 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 about intercept. The symbolic evaluation rules governing intercept are special cases in the current implementation in R. These implementations may not be as intuitive, 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 since
- `y ~ x*1` simplifies to `y ~ x`, which makes the cross operator unsymmetrial for this special case.

Some unambiguity arise from the need to explictly remove intercept, e.g.

- `y ~ 1 + (-1 + x)` simplifies to `y ~ x - 1`,
- `y ~ -1:x` simplifies to `y ~ 1 - 1`.

The first point was raised by Smith et al. (2018) and formula evaluation differ in `patsy` python library in this aspect. These complications arise due to explicit specification of removal of the overall intercept. Furthermore, this removes the resemblance to model equation detracting from the aim of symbolic model formula to make it accessible for practitioners.

3 Linear Mixed Models

Consider a n-vector of response $\boldsymbol{y}$ 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 $\boldsymbol{\beta}$; $\mathbf{Z}$ is the design matrix of the random effects $\boldsymbol{b}$ and $\boldsymbol{e}$ is the vector of random error. We typically assume that

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

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

3.1 `lme4`

The `lme4` R package fits a linear mixed model with the function `lmer`. The API consists of a *single* formula that extends the linear model formula. More specifically, 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 for linear models in Section 2 with `group` specific effects from `formula`. The `group` specific effects is assumed to be Gaussian distributed with zero mean and unstructured variance. Examples are provided in Section 4.

3.2 `asreml`

The strength of linear mixed model specification in `asreml` lies in its flexible covariance structure. The full list of covariance strucures available for `asreml` Version 3 is 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` correspond to a categorical variable in the data. See more examples in Section 4.

4 Motivating Examples

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.

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 the 4 diets. A possible model that a user may fit to this chicken weight data is illustrated in Figure 3 and elaborated next.

The model fitted in Figure 3 is commonly referred to as *random intercept and random slope model* with the diet treatment effect. More specifically, the model

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

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 random intercept and random time slope effects for the i-th chicken; β_0 and β_1 are the overall intercept and slope for time covariate.

4.2 Field Trial: Covariance Structure

4.3 Multi-Environmental Trial: Separable Structure

5 Discussion

Software packages that fit statistical models have varying input arguments.

Linear (fixed) models are special cases of linear mixed models. As such it is important that the model formula evaluation rules specified in linear models hold true for linear mixed models.

The `brms` papers make extensive discussion of symbolic model formula and extends on the framework built on `lme4`. These are noteworthy.

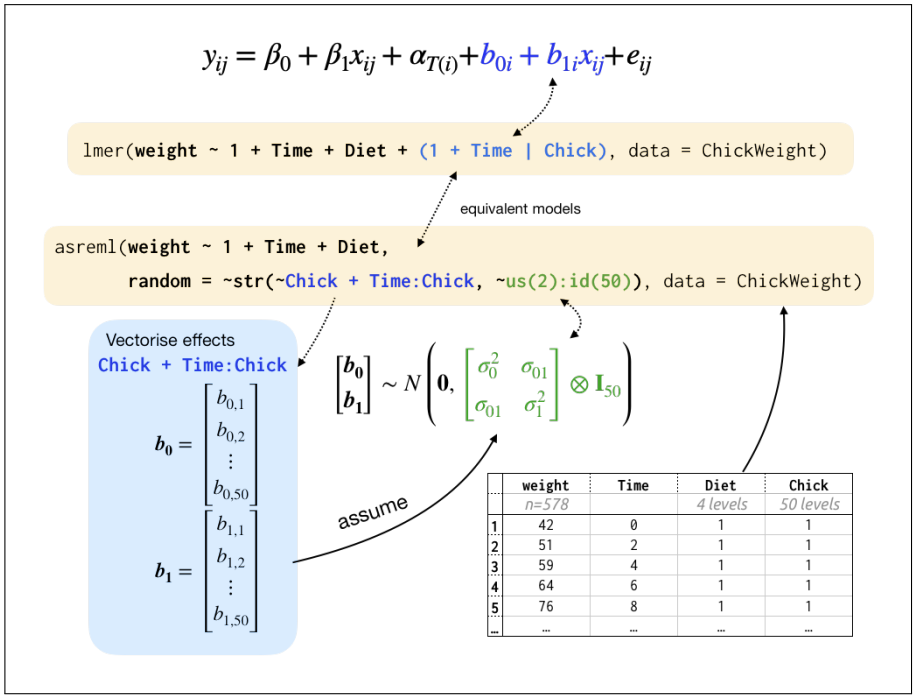

Fig. 3. This figure shows a longtiduinal analysis of the chicken data (see Section 4.1). The index form of the model equation shows direct resemblance for symbolic model formula in `lmer` for the fixed and random effects, however, the its covariance form is not as easily inferred. In contrast, the symbolic 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. This results in the loss of one-to-one correspondence between effects and symbolic terms for `asreml`.

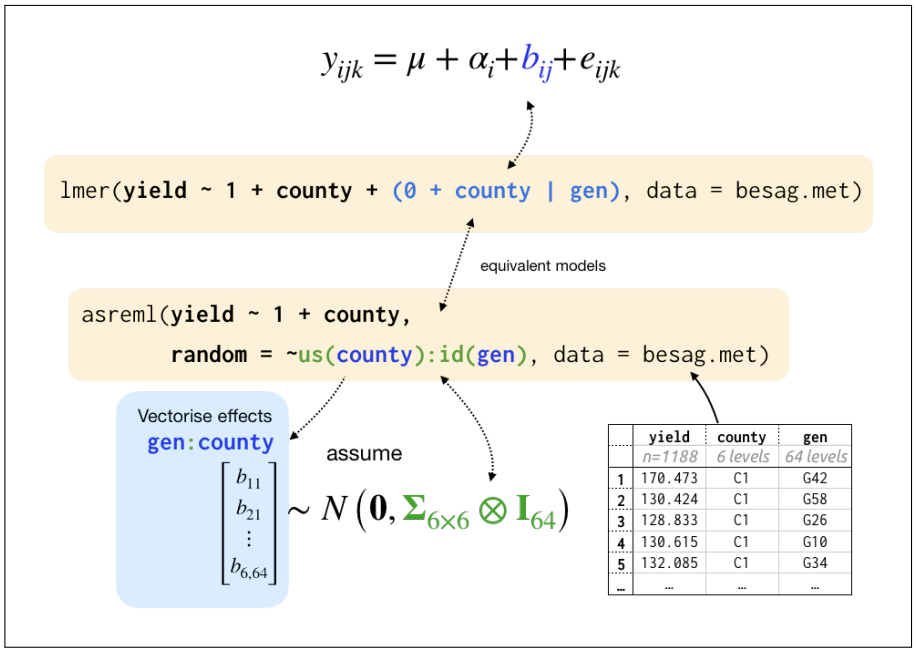

Fig. 4. This figure show the fit of a simple mixed mdoel for the analysis of MET data. In modelling the `county` by `gen` random effect, the variance structure are specified differently.

Acknowledgement

This paper benefited from twitter conversation with Thomas Lumley. This paper is made using R Markdown (Xie, Allaire, and Grolemund 2018). 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*.

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.

Gilmour, Arthur R 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.

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.

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.

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.

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.