Supplementary material: Accounting for individual variation in cognitive testing in farmed species

Emily V. Bushby[1], Mary Friel[1], Conor Goold[1], Helen Gray[1], Lauren Smith[1] & Lisa M. Collins[1]

[1]Faculty of Biological Sciences, University of Leeds, Leeds, LS2 9JT, UK

Contents

1 Introduction

In section 4, we consider a simple example study to demonstrate the utility of multilevel modelling for analysing, and investigating predictors of, individual variation in cognitive abilities. The example considers giving animals a series of 10 trials to initially learn a cognitive task, followed by 10 trials where the previously learned contingency has been reversed. For instance, Erhard et al. [1] used a T-maze paradigm with sheep, where one side of the T-maze was open and led to a food/social reward, while the other exit was closed. After initially learning which side was open, the sheep experienced two reversal conditions where the previously-closed side of the T-maze was open and the previously-open side closed. The authors compared sheep who had experienced prenatal undernutrition to control animals on their rate of learning the reversal task across the first and second reversal conditions. Among a number of different analyses, the authors used paired-samples t-tests to compare the different groups on their rates of learning the reversal learning tasks. However, there could be individual variation in a number of aspects in behaviour that are of interest. For instance, individual sheep may vary in their average probability of getting a trial correct, the change in the probability of getting a trial correct across trials, and in the interaction between trial number and cognitive task (initial learning task or reversal). This individual variation could, in addition, be predicted by the prenatal diet experienced. Multilevel models would allow for the combined analysis of these aspects within one statistical model, rather than several separate models.

1.1 Additional files

Here, we provide a more formal description of this example model, and demonstrate how the model can be fit with simulated data in `R` using both frequentist and Bayesian estimation. The script is located in the file `Bushby-et-al_R-script.R` and the Bayesian model written in the Stan language is located in the file `Bushby-et-al_Bernoulli-model.stan` .

1.2 Audience

This Supplementary Material file presumes some familiarity with the `R` language, and knowledge of the concepts and basic mathematical representation of general and generalised linear models. For those wanting to learn more about Bayesian analysis, we recommend the books by Kruschke [2] and McElreath [3], or the article by Kruschke and Liddell [4]. For those wanting to learn more about the Stan modelling language for statistical inference, we direct readers to the Stan modelling manual [5].

2 Model specification

The statistical model we consider is a hierarchical Bernoulli regression model (also known as a logistic or logit regression), which can be written formally as:

$$y_{ij} \sim Bernoulli(\pi_{ij}) \tag{1}$$

$$\begin{aligned} logit(\pi_{ij}) = log(\frac{\pi_{ij}}{1-\pi_{ij}}) =& \alpha + \nu_{1_j} + \gamma_1 \cdot Predictor_{ij} + \\ & (\beta_1 + \nu_{2_j} + \gamma_2 \cdot Predictor_{ij}) \cdot trial_i + \\ & (\beta_2 + \nu_{3_j} + \gamma_3 \cdot Predictor_{ij}) \cdot condition_i + \\ & (\beta_{1\text{x}2} + \nu_{4_j} + \gamma_4 \cdot Predictor_{ij}) \cdot trial_i \cdot condition_i \end{aligned} \tag{2}$$

In this model, the binary dependent variable denoting the outcome of the trial (correct or incorrect) for observation i of individual j, y_{ij}, is modelled as a Bernoulli random variable (taking values 0 or 1) with probability correct π_{ij}. The probability π_{ij} is a linear function of a number of parameters and variables:

- The intercept parameter (α) represents the the average probability correct across the trials and, in the example in the text, across the training and reversal learning task.
- The β coefficients describe the influence, across individuals, of trial number (1 to 10; β_1), changing from the training to the reversal condition (β_2), and the interaction between trial number and reversal ($\beta_{1\text{x}2}$), respectively. Trial number is standardised by subtracting its mean and dividing by its standard deviation, and the reversal variable is coded as -0.5 (training task) and 0.5 (reversal task) to ensure the intercept is the average probability correct across trials and conditions.
- The ν_j parameters denote the random effects, or the deviations from α and β parameters for each individual j.
- The γ regression coefficients denote the effect of an 'individual-level' predictor (predictor varies across individuals but is constant across observations within individuals, opposed to 'observation-level' predictors; [6]), such as prenatal diet or sex, on variation across the ν_j parameters. Thus, the γ coefficients capture the influence of diet on individual variation.
- Finally, the linear function is mapped to the binary response variable using the logit link function: $log(\frac{\pi_{ij}}{1-\pi_{ij}})$.

We have included random effects for each of the 'fixed' β coefficients (a 'maximal' model), but this should be informed by the study. In general, the more random effect parameters in the

model, the larger the sample size needed for accurate parameter estimates. However, rather than think of parameters estimated in models with small samples as 'wrong', we can think of the estimates as just inherently *uncertain*. This requires a shift of focus from black-and-white decision rules (e.g. significant vs. not significant) to accurately estimating uncertainty in the parameters. Bayesian analysis provides estimation of the uncertainty in parameters, and also allows easier convergence when fitting multilevel models to small datasets than more classical methods. For this reason, an increasing number of applied statisticians are turning to Bayesian estimation.

The random effect parameters $\nu_{1,\ldots,4_j}$ are considered multivariate normally distributed with means of 0 and covariance matrix Σ:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\rho_{12}\sigma_2 & \sigma_1\rho_{13}\sigma_3 & \sigma_1\rho_{14}\sigma_4 \\ \sigma_2\rho_{21}\sigma_1 & \sigma_2^2 & \sigma_2\rho_{23}\sigma_3 & \sigma_2\rho_{24}\sigma_4 \\ \sigma_3\rho_{31}\sigma_1 & \sigma_3\rho_{32}\sigma_2 & \sigma_3^2 & \sigma_3\rho_{34}\sigma_4 \\ \sigma_4\rho_{41}\sigma_1 & \sigma_4\rho_{42}\sigma_2 & \sigma_3\rho_{34}\sigma_4 & \sigma_4^2 \end{bmatrix} \tag{3}$$

where $\sigma^2_{1,\ldots,4}$ represent the standard deviations of the random effect parameters, and $\rho_{x,y}$ represent the correlations between random effect parameters x, y. For instance, $\rho_{1,2}$ is the correlation between ν_{1_j} and ν_{2_j}, or the correlation between individuals' average probability correct across trials and conditions, and the rate of learning across trials (independent of condition). If the correlation was positive, for example, individuals with a greater overall probability of getting a trial correct would also learn more quickly across trials.

2.1 Data generation

In the associated R script file `Bushby-et-al_R-script.R` we generate some data using a function called `generate_data()`, which simulates data from the mathematical model above. The function returns a list of data that can then be used for analysis. To set up the function, the user can enter their own desired parameter values (as described in the `R` script).

2.2 Frequentist estimation & non-parametric bootstrapping

This model can be fit using the convential frequentist approach using the `lme4` package's `glmer()` function in R with the following code:

```
fit_lme4 <- glmer(

  y ~ x_trial_z * x_cond_c * j_pred_z[j] +
      (x_trial_z * x_cond_c | j ),

  data = d, family = binomial(link="logit")
    )
```

This code is R shorthand for writing out all the single, non-interaction terms first and then including the interaction terms, i.e.

```
fit_lme4 <- glmer(

  y ~ x_trial_z + x_cond_c + j_pred_z[j] +
      x_trial_z * x_cond_c    +
      x_trial_z * j_pred_z[j] +
      x_cond_c  * j_pred_z[j] +
      x_trial_z * x_cond_c * j_pred_z[j] +
      (x_trial_z + x_cond_c + x_trial_z * x_cond_c | j ),

  data = d, family = binomial(link="logit")
    )
```

The predictors are either standardised (suffixed by `_z`) or centered around zero (suffixed by `_c`). Observation-level predictors are prefixed with `x_`, and individual level predictors are prefixed by `j_`.

The individual-level predictor `j_pred[j]` is indexed with `j`, which is a vector in the data list indicating individual ID, because `lmer()` requires all data vectors to be of the same length.

Running the frequentist version of the model results in convergence warnings, although the parameter estimates are largely consistent with the Bayesian model below. In general, large multilevel models with potentially small numbers of grouping factors may be difficult to fit using frequentist approaches. The Bayesian approach outlined below is often more adaptable in these cases.

Finally, the `R` script includes code to generate confidence intervals on the regression parameters using non-parametric bootstrapping. In this code, a new data set is first generated by sampling individuals with replacement from the original data set. Then, the model is fit to this data set. We do this 100 times, which takes approximately 10 minutes, but for formal analyses it would be better to increase this number of 'bootstraps' to > 1000, although the code could be made more efficient for this purpose. Note, we use the `glmmTMB` package for this purpose, which is quicker than `lme4` .

2.3 Bayesian estimation using *brms*

A Bayesian version of this model can be computed using the `brms` package [7] using the `brm()` function:

```
fit_brms <- brm(

  y ~ x_trial_z * x_cond_c * j_pred_z[j] +
      (x_trial_z * x_cond_c | j ),

  data = d, family = "bernoulli"
    )
```

What is different about the Bayesian analysis? Most simply, Bayesian estimation provides a different method of estimating the parameters of the model. However, the basic model is the same. In the frequentist version, we obtain point-estimates of each parameter (the maximum likelihood estimates) and their standard errors. In the Bayesian version, we obtain a complete distribution of plausible values for each parameter, known as the *posterior distribution*. The posterior distribution is exactly the probability of the different parameter values given the data we provide the model with. Those parameter values more likely in the posterior distribution are most likely given the data. For example, the mean of the posterior distribution is the most probable parameter describing the data.

Calculating the posterior distribution is done via Bayes' rule. If we denote our hypothesis H and the data we collect D, then Bayes' rule tells us the probability of our hypothesis given the data using the formula:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \qquad \text{(Bayes' rule)}$$

Bayes' rule tells us that the posterior probability of the hypothesis given the data is proportional to the probability of the data given the hypothesis, $P(D|H)$ or the *likelihood*, multiplied by the *prior* probability of the hypothesis, $P(H)$, which is the probability of the hypothesis before accounting for the data. This product is divided by the total probability of the data, $P(D)$, which is the probability of the data under all the different hypotheses being considered. In practice, Bayes' rule can be difficult or impossible to calculate for large models due to tricky integrals in the denominator, so Bayesian software employs Markov chain Monte Carlo (MCMC) algorithms to approximate the posterior distribution.

The `brms` package interfaces with the Stan probabilistic programming language ([8]), which estimates model parameters using Hamiltonian Monte Carlo, a type of Markov chain Monte Carlo algorithm. The code above uses default prior distributions for the parameters provided by the `brms` package, which can be inspected using `get_prior()` function or, alternatively, adjusted within the `brm()` function using the `set_prior = ...` command.

2.4 Bayesian estimation using Stan & *Rstan*

For the analyses reported in the main text, we wrote the Bayesian model directly in the Stan language, run through the `Rstan` package [9] which provides greater flexibility for specifying hierarchical models of arbitrary complexity. The code is provided on page 10, and a more heavily commented version is provided in the file `Bushby-et-al_Bernoulli-model.stan`.

Parts of the Stan code may be more opaque than others, particularly two modelling nuances that are used to improve convergence. While the specific details of the Stan code can be safely skipped, these nuances are important in practice because they help fit multilevel models to sparse data structures, where there is not enough data to efficiently estimate the parameters - a potentially common occurrence in studies of farm animal cognition, and applied ethology in general.

Non-centered parameterisation: First, the Stan model parameterises the random effects using the 'non-centered' format. A simple case is the model:

$$
\begin{aligned}
y_{ij} &\sim N(\mu_{ij}, \sigma) && (4)\\
\mu_i &= \alpha + \nu_j && (5)\\
\nu_j &\sim N(0, \tau) && (6)\\
\tau &\sim Unif(0, 100) && (7)\\
\sigma &\sim Unif(0, 100) && (8)
\end{aligned}
$$

Imagine that y_{ij} represents the i^{th} response from individual j, which is assumed to come from a normal distribution with mean μ_{ij} and standard deviation σ. The mean is a linear function of a group-level intercept, α, and a deviation from the group-level intercept for each individual ν_j. The ν_js are assumed normally distributed with mean 0 and standard deviation τ. The standard deviation parameters in the model are both given vague uniform prior distributions.

In some cases, estimating the parameters in this model can be difficult because of their mathematical depenedency. Instead, the same model can be written equivalently as:

$$
\begin{aligned}
y_{ij} &\sim N(\mu_{ij}, \sigma) && (9)\\
\mu_i &= \alpha + \tau z_j && (10)\\
z_j &\sim N(0, 1) && (11)\\
\tau &\sim Unif(0, 100) && (12)\\
\sigma &\sim Unif(0, 100) && (13)
\end{aligned}
$$

The only thing that has changed is that the random effects from the previous model (the ν_js) have been transformed to z-scores. That is, we can write the ν_j as z-scores by $z_j = \frac{\nu_j - 0}{\tau} = \frac{\nu_j}{\tau}$.

Via a little algebra, we can write the model equivalently by expressing the random effects as $z_j\tau = \nu_j$. In practice, this formulation can lead to more efficient sampling of multilevel models, and therefore is a valuable practical tool.

Line 23 in the Stan code demonstrates the definition of the random effects using this uncentered format. However, lines 23 and 24 further require understanding the generation of correlated random effects, which uses the Cholesky factorisation of the correlation matrix.

Cholesky factorisation: Modelling covariances between random effect parameters can be difficult in Bayesian models due to the non-obvious choice of suitable prior distribution. The Stan modelling language allows users to put a prior distribution on what is known as the Cholesky factorisation of the correlation matrix and the standard deviations of the random effect parameters, rather than trying to find a suitable prior for the covariance matrix. We can use this Cholesky factorisation to generate correlated random variables like the correlated random effect parameters.

For those interested in the details, a correlation matrix $\boldsymbol{P}$ can be factored into the product of its lower triangular square root, $\boldsymbol{L}$, multipled by the transpose of the lower triangular square root:

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^T \tag{14}$$

This Cholesky factorisation can be used in the generation of the vector of correlated random effects, $\boldsymbol{\nu}$, by first multiplying the vector of random effect standard deviations, $\boldsymbol{\Omega}$, with the lower triangular matrix of the correlation martix, $\boldsymbol{L}$ to form a matrix $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda} = \boldsymbol{\Omega}\boldsymbol{L} \tag{15}$$

and then multiplying $\boldsymbol{\Lambda}$ with a vector of unit normal variates $\boldsymbol{Z}$ to derive the correlated random effects $\boldsymbol{\nu}$:

$$\boldsymbol{\nu} = \boldsymbol{\Lambda}\boldsymbol{Z} \tag{16}$$

The correlation matrix between the random effects can be obtained using equation 14 (line 24 of the Stan model). This formulation means that prior distributions can be placed on the lower triangular matrix and the standard deviations of the random effects, without needing to specify a prior for the covariance matrix. Luckily, packages such as `brms` do this automatically behind the scenes.

Prior distributions: As mentioned briefly above, Bayesian estimation requires specifying prior distributions for all parameters to apply Bayes rule. Prior distributions can be specified in numerous ways, from 'vague' or 'broad' where all possible parameter values are considered just as likely (e.g. choosing a uniform prior distribution on a regression coefficient between

-1000 and 1000), to informative or *regularising*, where the prior distributions are focused on a specific range of parameter values (e.g. a normal prior distribution with mean 0 and standard deviation 0.5 for a standardised regression coefficient). The prior distributions in the Bayesian implementation here are chosen to be 'weakly informative' in general, which both improves efficiency and ensures parameter estimates are slightly more conservative than if we were to use vague prior distributions. For the main regression coefficients, the prior distributions are:

$$\alpha \sim N(0, 10) \tag{17}$$
$$\beta_{1,\ldots,3} \sim N(0, 1) \tag{18}$$
$$\gamma_{1,\ldots,4} \sim N(0, 1) \tag{19}$$
$$\sigma_{1,\ldots,4} \sim N(0, 1)^{+} \tag{20}$$

The intercept prior is normally distributed, centered at 0 (corresponding to a probability of 50% correct) and with a standard deviation of 10, which is relatively wide on the log-odds scale. The β and γ regression coefficients also have normally distributed priors centered on zero, with standard deviations of 1, which corresponds to around a 73% change in the dependent variable. The standard deviations of the random effects have half-normal priors (disallowing negative values), with standard deviations of 1, which are relatively tight to aid model convergence.

Finally, the Stan code also includes an LKJ prior distribution on `L_Rho`, the lower triangle or Cholesky factor of the correlation matrix, named after Lewandowski, Kurowicka, and Joe [10] (line 35), with shape parameter 2.

$$L_{Rho} \sim LKJ(2) \tag{21}$$

The matrix Z in equation 16 above is also transformed to a vector of unit normals:

$$\boldsymbol{z} \sim N(0, 1) \tag{22}$$

```
  // Stan model
data{
  int<lower=1> N;
  int<lower=1> N_j;
  int<lower=1> j[N];
  vector[N] x_trial_z;
  vector[N] x_cond_c;
  vector[N_j] j_pred_z;
  int<lower=0, upper=1> y[N];
}

parameters{
  real alpha;
  vector[3] beta;
  vector[4] gamma;
  vector<lower=0>[4] Sigma;
  cholesky_factor_corr[4] L_Rho;
  matrix[r, N_j] Z;
}

transformed parameters{
  matrix[4,4] Rho;
  matrix[N_j,4] nu;
  nu = (diag_pre_multiply(Sigma, L_Rho) * Z)';
  Rho = L_Rho * L_Rho';
}

model{
  vector[N] eta;

  // prior distributions
  to_vector(Z) ~ normal(0, 1);
  alpha ~ normal(0, 10);
  beta ~  normal(0, 1);
  gamma ~ normal(0, 1);
  L_Rho ~ lkj_corr_cholesky(2.0);
  Sigma ~ normal(0, 1);

  // linear predictor (equation 2)
  for(i in 1:N){
    eta[i] = alpha + nu[j[i],1] + gamma[1] * j_pred_z[j[i]]
      + (beta[1] + nu[j[i],2] +
         gamma[2] * j_pred_z[j[i]]) * x_trial_z[i]
      + (beta[2] + nu[j[i],3] +
         gamma[3] * j_pred_z[j[i]]) * x_cond_c[i]
      + (beta[3] + nu[j[i],4] +
         gamma[4] * j_pred_z[j[i]]) * x_trial_z[i] * x_cond_c[i];
  }

  // Bernoulli likelihood statement (equation 1)
    y ~ bernoulli_logit(eta);
}
```

3 Results

The results of the Bayesian models in Stan and `brms` are posterior distributions, held in the matrices `ps_stan` and `ps_brms`, respectively (with `ps` standing for 'posterior samples'). The code then uses the the Stan model results to graph the results shown in Figure 1 of the text.

3.1 Interpreting Bayesian posteriors

Remember that the Bayesian models return a full posterior distribution for each of the parameters. The columns in `ps_stan` and `ps_brms` are the parameters, and the rows are equal to the total length of the MCMC chains we ran (10,000 iterations in the Stan model, but only 4000 iterations using `brms`'s default settings). Each row contains jointly-credibly samples from the posterior distribution.

Exploration of the posterior distribution can be conducted easily by computing the mean and Bayesian credible or highest density intervals (HDI) of each of the parameters, or the mean and credible intervals of the differences between the parameters, depending on the purpose of the analysis. For instance, to investigate the effect of trial we can obtain the mean and 89% HDI, by running:

```
beta1 <- ps_stan[,"beta[1]"]
mean(beta1)
HPDI(beta1, 0.89)
```

By obtaining a full posterior distribution for each of the parameters, Bayesian analyses make comparing parameters and detecting significance easy. For example, we may be interested in whether a parameter is significantly different from zero. To do so, we can assess whether the 89% HDI (or some other percentile - the conventional choice of 95% is often arbitrary) includes zero or not. For β_1, the mean is 0.26 and its 89% HDI is [0.18, 0.35]. Therefore, it is credibly greater than zero. This parameter is show in Figure 1.

3.2 Parameter comparison

We may want to compare the results across the different models we have computed. The last section of code achieves this using the `ggplot2` package, shown in Figure 2. In the figure, the dots represent the mean posterior estimates and the line segments reflect the 89% confidence intervals (frequentist) or highest density intervals (Bayesian) estimated by the models. The real simulated values are shown as red dots. In most cases, the parameters recover the simulated values, although many parameters are widely uncertain. For the standard deviation parameters (denoted `SD_i`), there was some difficulty recovering the true parameters, but note that the Bayesian models' HDIs are often more accurate than the `glmmTMB` results. With only

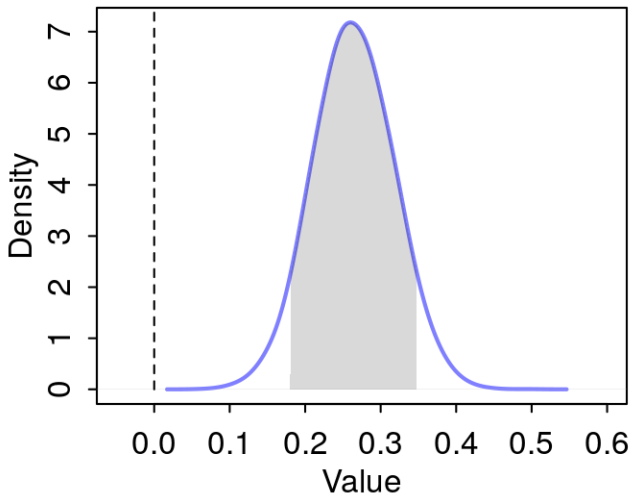

Figure 1: Density of β_1. Zero is marked by a vertical dashed line, and the shaded region denotes the 89% HDI.

100 individuals in the data set, there will be greater simulation variance or error compared to the generating values.

References

[1] Hans W Erhard et al. "Effects of prenatal undernutrition on emotional reactivity and cognitive flexibility in adult sheep". In: *Behavioural Brain Research* 151.1 (2004), pp. 25–35. ISSN: 0166-4328. DOI: https://doi.org/10.1016/j.bbr.2003.08.003.

[2] John Kruschke. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan*. en. Google-Books-ID: FzvLAwAAQBAJ. Academic Press, Nov. 2014. ISBN: 978-0-12-405916-0.

[3] Richard McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Dec. 2015.

[4] John K Kruschke and Torrin M Liddell. "Bayesian data analysis for newcomers". In: *Psychonomic bulletin & review* 25.1 (2018), pp. 155–177.

[5] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. 2016.

[6] Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*.

[7] Paul-Christian Bürkner. "brms: An R Package for Bayesian Multilevel Models Using Stan". In: *Journal of Statistical Software* 80.1 (2017), pp. 1–28. DOI: 10.18637/jss.v080.i01.

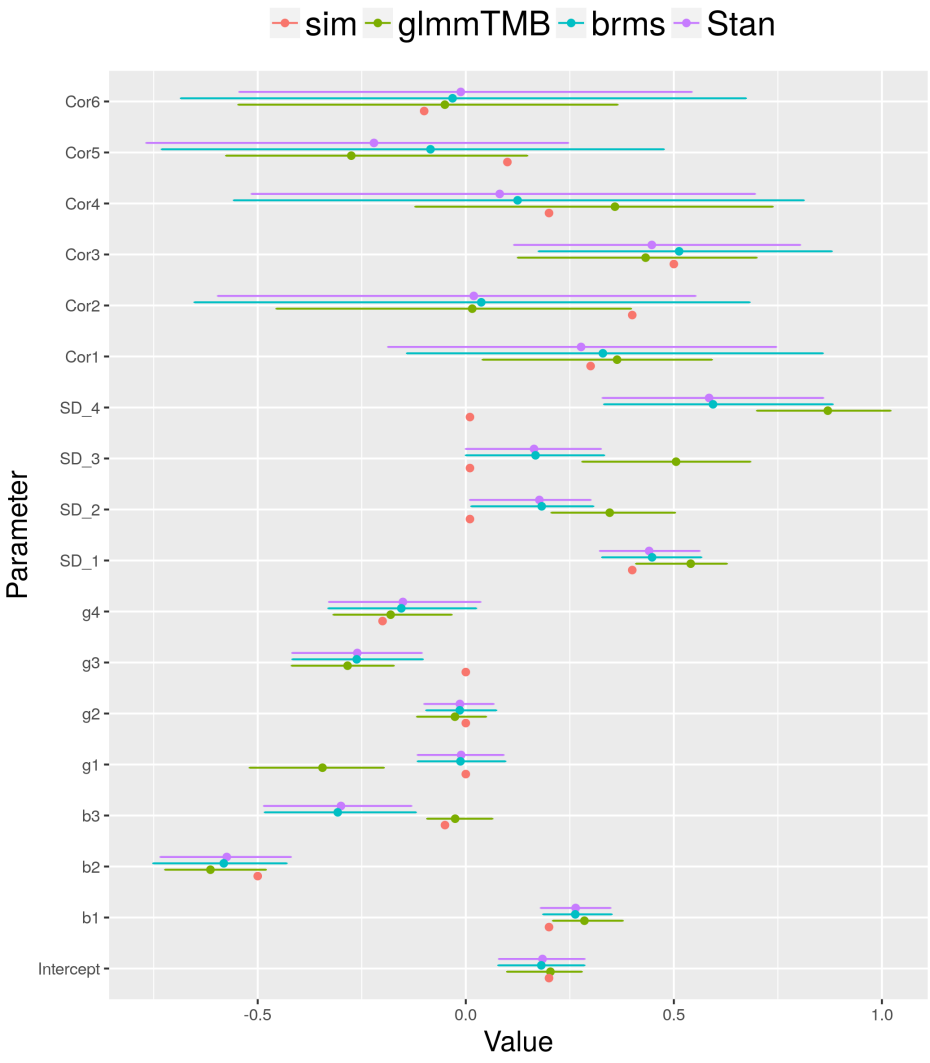

Figure 2: Comparing the frequentist (glmmTMB) and Bayesian (brms & Stan) models' parameter values to the actual simulated values (sim).

[8] Bob Carpenter et al. “Stan: A Probabilistic Programming Language”. In: *J Stat Softw* (2016).

[9] Stan Development Team. *RStan: the R interface to Stan*. R package version 2.17.3. 2018.

[10] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. “Generating Random Correlation Matrices Based on Vines and Extended Onion Method”. In: *Journal of Multivariate Analysis* 100.9 (Oct. 2009), pp. 1989–2001. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2009.04.008.