Fitting linear mixed models using JAGS and Stan: A tutorial

Tanner Sorensen
Department of Linguistics, University of Potsdam, Germany
Shravan Vasishth
Department of Linguistics, University of Potsdam, Germany
School of Mathematics and Statistics, University of Sheffield, UK

Version dated May 1, 2014

Abstract

This tutorial is aimed at psycholinguists and psychologists interested in fitting linear mixed models using JAGS and Stan.

Keywords: Bayesian linear mixed models, JAGS, Stan

Ever since the arrival of the `nlme` package (Pinheiro & Bates, 2000) and its subsequent version, `lme4` (Bates & Sarkar, 2007), the use of linear mixed models in psychology and linguistics has increased dramatically. In the present tutorial, we show how standard models in psychology, linguistics, and psycholinguistics can be fitted easily using Bayesian tools such as JAGS (Plummer, 2012) and Stan (Stan Development Team, 2013). Our presentation focuses on practical details, in order to allow the reader to quickly start fitting their own models. For simplicity, we focus on two simple designs: a two-condition repeated measures study, and a 2×2 repeated measures factorial design.

There are no prerequisites apart from having some exposure to fitting linear mixed models using `lme4`, and having the relevant software installed: JAGS, Stan, `rjags`, `rstan` in R, and any associated software; see the JAGS (http://mcmc-jags.sourceforge.net) and Stan (mc-stan.org) websites for details.

Stan is the more general programming language for psycholinguistic research because it allows the researcher to flexibly fit fairly complex models. However, we

We thank Martyn Plummer, Bob Carpenter, Andrew Gelman, and members of the Stan mailing list for assistance. Titus von der Malsburg wrote an initial version of the data generation code; Titus also identified a number of typographical errors in the draft. For any questions and comments related to this tutorial, please email the authors: {tsorense,vasishth}@uni-potsdam.de.

chose to introduce JAGS first because it uses BUGS syntax, which is currently widely used in textbooks; anyone learning to do Bayesian analysis would need to understand BUGS syntax in order to read introductory books. We provide some references at the end of the tutorial.

Fitting a linear mixed model with a full variance-covariance matrix by subject and by items

Imagine that we are interested in the effect of ungrammaticality on reading time in a reading task. A typical way to address this kind of question is to prepare several experimental items, each with a grammatical and ungrammatical version. For example, if we decide to use 6 items, each with two versions (grammatical and ungrammatical), then each participant could see six items that are ungrammatical, and six that are grammatical. Thus, we would have 12 data points from each participant. Imagine that we have three participants. We can simulate such a data-set; see Table 1 (we show later how these data are generated). This is one kind of design that is used in psycholinguistic data; we will discuss other designs in case studies using real data.

	subj	item	condition	rt
1	1	1	gram	596
2	1	1	ungram	638
3	1	2	gram	546
4	1	2	ungram	569
5	1	3	gram	615
6	1	3	ungram	584

Table 1
Example of (simulated) repeated measures data (the first six rows of a data frame).

In order to express the fact that the reading times have a common mean with an adjustment for the grammaticality factor, we decompose each reading time into a sum whose terms are the mean of the grammatical condition, β_0, and an adjustment β_1 for the reading time being associated with the ungrammatical condition. So the i^{th} reading time RT_i is given by the sum:

$$\text{RT}_i = \beta_0 + \beta_1 \text{condition}_i + \epsilon_i \qquad (1)$$

The predictor, *condition*$_i$, specifies whether RT_i was in the ungrammatical or grammatical condition. In the data frame shown above, the grammatical condition is written as “gram”, and the ungrammatical condition as “ungram”. In the model shown in 1, we need a way to assign numerical values to the two conditions (multiplying β_1 with a factor level called “grammatical” is an undefined mathematical operation). One way to do this is to say that if RT_i is recorded for the ungrammatical condition, then condition_i is coded as 1. If not, then it is coded as 0. This is called treatment contrast coding, and it has the effect of adjusting the mean for the grammatical condition β_0

by β_1 in the ungrammatical condition. β_1 is now the additional cost of ungrammaticality (or, equivalently, the difference in reading time between the grammatical and ungrammatical condition). Together β_0 and β_1 make up the *fixed* part of the model, which characterizes the effect of the experimental manipulation on RT. The third term ϵ_i is the residual error: from equation 1, it is clear that this is the amount by which the predicted values from the model ($\beta_0 + \beta_1 \text{condition}_i$) differ from RT_i. We assume here that ϵ_i is normally distributed with mean 0 and unknown variance σ^2; this is written as $\epsilon_i \sim N(0, \sigma^2)$. This variance is estimated from the data. For the above data, the estimates of the fixed effects β_0, and β_1, and their standard errors can be computed using the `lm` function in R; see Table 2.

	Estimate	Std. Error
beta0	572.11	8.98
beta1	31.83	12.70

Table 2
Estimated fixed effects and their standard errors.

It is clear that the RT will vary systematically depending on the participant and on the item; some participants will be read fast, some slow, and similarly, some items will be read fast, some slow. Participants and items therefore add variability in the response. We could in principle ignore these sources of variability (or variance components), by assuming only one source of variance, σ^2. An alternative is to take all these variance components into account. We can do this by adding adjustment terms u_{0j} and w_{0k}, which adjust β_0 for participant j and item k. This partially decomposes ϵ_i into a sum of the terms u_{0j} and w_{0k}, which adjust β_0 to incorporate an adjustment to the intercept β_0 for participant j and item k. If participant j is slower than the average of all the participants, u_j would be some positive number, and if item k is read faster than the average reading time of all the items, then w_k would be some negative number. These adjustments u_{0j} and w_{0k} are called random or varying intercepts, and by adjusting the β_0 by these we account for idiosyncratic patterns of speakers and items. We assume that these adjustments are normally distributed around 0: $u_{0j} \sim N(0, \sigma_u^2)$ and $w_{0k} \sim N(0, \sigma_w^2)$. Note that now we have three variance components in this model: σ^2, σ_u^2, and σ_w^2. Note also that all three variance components are assumed to be independent of each other. We can now express the i^{th} reading time — which is associated with subject j reading item k — as the following sum.

$$\text{RT}_{ijk} = \beta_0 + u_{0j} + w_{0k} + \beta_1 \text{condition}_i + \epsilon_i \tag{2}$$

This kind of model, which is called a varying intercepts model, can be fit in R using the `lmer` function available in the package `lme4`, using the command shown below. The estimated coefficients and their standard errors are shown in Table 3.

```
library(lme4)
```

```
m1<-lmer(rt~condition+(1|subj)+(1|item),data)
```

	Estimate	Std. Error
beta0	572.11	9.60
beta1	31.83	12.58

Table 3
Fixed effects estimates of the varying intercepts model.

For the simulated data-set discussed above, the standard deviation of the participants' varying intercepts component is 6.23; for the items varying intercept component, it is 0, and the residual (error) standard deviation is 37.75.

Suppose now that although subject j is a fast reader, she may exhibit greater slowdowns due to the ungrammaticality violation. It might also be that item k is read somewhat more quickly in the ungrammatical condition than the other items are. There may be other participants and items that are impacted to differing degrees by the grammaticality violation. This can be expressed by adjusting β_1 by some quantities u_{1j} and w_{1k}. These are varying slopes, and by adding them we account for the effects for ungrammaticality, which are idiosyncratic to participant j and item k. We now express RT_{ijk} as the following sum.

$$\mathrm{RT}_{ijk} = \beta_0 + u_{0j} + w_{0k} + (\beta_1 + u_{1j} + w_{1k})\mathrm{condition}_i + \epsilon_i \tag{3}$$

The varying intercepts u_{0j} and w_{0k} are adjustments to the fixed intercept β_0, and the varying slopes u_{1j} and w_{1k} are adjustments to the fixed slope β_1. We assume that these adjustments are normally distributed with mean 0 and some unknown variance. Importantly, we also assume that the varying intercepts and slopes for participants are correlated; and that the varying intercepts and slopes for items are also correlated. We elaborate on this point below.

All these random quantities can be regarded as adjustments that reduce the magnitude of the error term ϵ_i. Recall that the expected value of ϵ_i was 0 in equation 1. This is still the case in equations 2 and 3. This constrains the expected value of the random intercepts and random slopes to likewise be 0. As with ϵ_i, if the expected value of the random effects were nonzero, we would add that quantity to β_0 (Searle, Casella, & McCulloch, 2009).

Model 3, which is called a varying intercepts, varying slopes model, is useful in psycholinguistic research because it faithfully reflects all the sources of variance in the experimental design. The experimental design suggests a natural partitioning of RT into groups associated with a given subject or a given item. These groups are specified by considering RT_{ijk} with either the subject index j or the item index k held constant. Groups defined along these lines display systematically different patterns of variance. For example, by-subject variance in language comprehension tasks has been attributed to factors such as individual differences in working memory capacity (Just & Carpenter, 1992) and in processing speed (Kliegl, Masson, & Richter, 2010). Cognitive models of performance in such tasks often make predictions about the relationship between a subject's random intercept and slope. For instance, a fast reader might be expected to take less time resolving an ungrammaticality on the rationale that processing speed in normal reading reflects the processing speed for resolving ungrammaticalities. Such a prediction can be evaluated experimentally by estimating the correlation between u_0 and u_1. A positive correlation between the varying intercepts and varying slopes would support such a prediction.

So far we have assumed several variance components, but we have not explicitly specified how they will covary. If the correlation between the subjects' random intercepts u_{0j} and random slopes u_{1j} is to be estimated, the model must specify that u_{0j} and u_{1j} covary. This simply means that the different variances are not mutually independent. We assume that u_0 and u_1 are normally distributed with mean 0 and with variance and covariance given by the matrix Σ_u, given below. The parameters in Σ_u are unknown, and so we must estimate them. The parameter ρ_u indicates the correlation between u_0 and u_1; it is our estimate of ρ_u by which we evaluate predictions about individual differences in experimental conditions. A variance-covariance matrix Σ_w can likewise be defined for by-item random effects.

$$\Sigma_u = \begin{bmatrix} \sigma^2_{\mathrm{u0}} & \rho_u \, \sigma_{u0}\sigma_{u1} \\ \rho_u \, \sigma_{u0}\sigma_{u1} & \sigma^2_{u1} \end{bmatrix} \tag{4}$$

$$\Sigma_w = \begin{bmatrix} \sigma^2_{\mathrm{w0}} & \rho_w \, \sigma_{w0}\sigma_{w1} \\ \rho_w \, \sigma_{w0}\sigma_{w1} & \sigma^2_{w1} \end{bmatrix} \tag{5}$$

To state all of this more formally, the linear mixed model for this particular example can be specified by assuming that the varying intercepts and slopes by participants and by items have the following bivariate distribution:

$$\begin{pmatrix} u_{0j} \\ u_{1j} \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{\Sigma}_u) \quad \begin{pmatrix} w_{0k} \\ w_{1k} \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{\Sigma}_w) \tag{6}$$

The variance components associated with participants and items are generally treated as nuisance parameters in psycholinguistic work. They are generally included in the model only to discount the possibility that the fixed effects estimates depend on the particular subjects and items used in the experiment. However, as mentioned above, there are situations where these variance components can be of theoretical interest; an example is where we have a theory about how reading speed of a participant affects the magnitude of their ungrammaticality effect. Here, the correlation parameter is of intrinsic theoretical interest.

We turn next to the practical details of model fitting in JAGS and Stan. First we introduce the code for estimating ρ_u using the probabilistic programming language JAGS (Plummer, 2012). Then we extend these techniques to more complicated experimental designs, using Stan (Stan Development Team, 2013). In both sections, we assess the accuracy of our estimations using simulated data where the underlying correlations are known.

Writing out a varying intercepts linear mixed model using JAGS

In the previous section we illustrated the linear mixed model and motivated model 3, which has a fixed part to capture planned experimental effects and a random part to capture by-subject and by-item variance. In this section, we implement this model in the Bayesian setting using the probabilistic programming language JAGS. We illustrate this for the simple case of an experimental design with one factor and two levels. We begin by illustrating JAGS syntax before applying our model to a simulated data-set, where we show how to estimate the random effects correlation.

JAGS syntax

In the Bayesian setting, estimates of model parameters such as β_1 or ρ are random variables with a probability distribution, not point values as they are in the frequentist setting. These are called *posterior* probability distributions, and they are derived using Bayes' theorem, which states that the posterior probability distribution of the model parameters θ is proportional to the probability of the observed data given the parameter(s) θ (the *likelihood*), times the *prior* probability distribution of θ. We can restate this as follows:

$$p(\theta \mid y) \propto p(y \mid \theta)p(\theta). \tag{7}$$

The likelihood and the prior are specified directly in our JAGS model. The model specification embodies our assumptions about the underlying processes which generated the observed data. These include both distributional assumptions about certain parameters, e.g., that u_0 is drawn from a normal distribution with mean 0 and unknown variance, and assumptions about the form of the model which generates the data itself. Below we give the relevant JAGS code chunks. Complete model specifications are provided in the appendix.

We first define the fixed effects of our model. Recall that these include an intercept β_0 and an adjustment β_1 for the experimental condition. In JAGS, we write β as a vector with two entries, each of which is drawn from a normal distribution. The prior distribution on β is defined as below:

```
beta[1] ~ dnorm(500,1.0E-5)
beta[2] ~ dnorm(0,1.0E-5)
```

The tilde ($\sim$) means that the parameter to its left is drawn from the probability distribution to its right, which is the normal distribution. Its first argument is the mean and the second is the *precision.* Precision is the inverse of the variance, and is often used in Bayesian settings for reasons that do not concern us here. Low precision means the same thing as high variance in that it places few a priori constraints on the value of a parameter. Such a prior is called vague or “uninformative”. We have chosen such a prior here, but if we already knew a great deal about the parameters, we could have chosen a more informative prior.

We now define the random effects structure of our model. In this article we give an explicit definition only for the by-subject random effects u_0 and u_1, but the same definition is repeated for by-item random effects in the complete model specification. In an experiment with J participants there will be J pairs of by-subject random intercepts u_{0j} and slopes u_{1j}. Recall that our model specifies a covariance structure between the u_0 and u_1. This is expressed in our model by drawing u_{0j} and u_{1j} from a multivariate normal distribution, for $j = 1, 2, \ldots, J$.

```
for( j in 1:J )
{
  u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
}
```

The first argument to the multivariate normal distribution is a vector of two 0s, which specifies the mean values of u_{0j} and u_{1j}. The second argument is the precision matrix Σ_u^{-1}, which is the inverse of the variance-covariance matrix Σ_u from equation 4. A definition of Σ_u^{-1} involves the precisions of the by-subject random intercept and slope, which we denote as τ_{u_0} and τ_{u_1}, respectively. We define the priors for these below (see Chung, Gelman, Rabe-Hesketh, Liu, & Dorie, 2013 for the motivation).

```
tau.u1 ~  dgamma(1.5, 1.0E-4)
tau.u2 ~ dgamma(1.5, 1.0E-4)
```

That is, precision is drawn from a gamma distribution with shape parameter 1.5 and scale parameter small, defining a vague prior. Precision can be transformed to standard deviation, which we do for ease of interpretability. The function `pow` below is the power function: it takes a numerical value and raises it to some power (here, $-1/2$).

```
sigma.u1 <- pow(tau.u1,-1/2)
sigma.u2 <- pow(tau.u2,-1/2)
```

The correlation parameter ρ_u also plays a role in the precision matrix Σ_u^{-1}. This is the correlation between u_0 and u_1. Accordingly, it is constrained to take on values between -1 and 1. To express this, we define as prior for ρ_u a normal distribution which has been truncated at -1 and 1 (the command T(-1,1) implements the truncation).

```
rho.u ~ dnorm(mu_rho.u,tau_rho.u)T(-1,1)
mu_rho.u ~ dunif(-1,1)
tau_rho.u ~ dgamma(1.5,1.0E-4)
```

Notice that the mean and precision of the prior distribution on ρ_u are themselves drawn from a probability distribution. When a parameter of a prior distribution is drawn from a probability distribution, we say that it is drawn from a *hyperprior* distribution. This has the effect of letting the mean and precision of the prior on ρ_u vary probabilistically. We will discuss this in more detail below with some simulation results.

We have now specified τ_{u_0}, τ_{u_1}, and ρ_u. We put these pieces together by defining a prior on Σ_u^{-1}, which determines the pattern of covariance displayed by u_0 and u_1. In particular, we draw Σ_u^{-1} from a Wishart distribution. This takes a scale matrix as a parameter, which we specify as R_u, and a degrees of freedom argument, which we specify to be 2.

```
R.u[1,1] <- pow(sigma.u1,2)
R.u[2,2] <- pow(sigma.u2,2)
R.u[1,2] <- rho.u*sigma.u1*sigma.u2
R.u[2,1] <- rho.u*sigma.u1*sigma.u2
invSigma.u  ~ dwish(R.u,2)
Sigma.u <- inverse(invSigma.u)
```

This completes the definition of the prior on the by-subjects random effects. The same definition also works for the by-item random effects; we leave this as an exercise for the reader, although the solution is provided in the appendix.

The prior on u_0 and u_1 implements certain model assumptions which were discussed in the introduction. In particular, we have specified that the mean of the prior distribution on u_0 and u_1 is zero and that their variances σ_{u_0} and σ_{u_1} covary. This is done explicitly when we draw u_0 and u_1 from a multivariate normal distribution with mean zero and precision matrix Σ^{-1}. This contrasts with the prior on β_0 and β_1, where no such assumptions were made. This is evident in the fact that they are drawn from independent normal distributions.

Finally, we define the prior on the residual variance σ_e, which captures the noise in the data; the upper bound of the uniform prior is simply chosen as a reasonable value based on our knowledge of the research domain. We then transform this to the precision scale, which we need in the model specification.

```
sigma_e  ~ dunif(0,50)
tau.e <- pow(sigma_e,-2)
```

The prior distributions on the model parameters which we have specified combine with the experimental data through the likelihood function to yield a posterior distribution for each parameter. The likelihood function specifies how the various parts of the model generate the dependent variable RT. Note that `subj[i]` and `item[i]` gives the subject and item associated with the i^{th} reading time and that there are N reading times in all. Note also that the predictor $condition_i$ has been replaced by x_i, just for notational ease.

```
for( i in 1:N )
{
mu[i] <- ( beta[1]
          + u[subj[i],1]
          + w[item[i],1] )
          + ( beta[2]
          + u[subj[i],2]
          + w[item[i],2] ) * x[i]
RT[i] ~ dnorm( mu[i], tau.e )
}
```

Compare this with the varying intercepts, varying slopes model specification discussed earlier (equation 3); the above code simply implements the statement in equation 3. The variable `mu[i]` gives the expected value for `RT[i]`. Note that this expected value is conditioned by the fixed effect for grammaticality and random effects for subject and item. The residual variance absorbs whatever variance in RT remains unexplained by the model.

JAGS simulations

In the preceding section, we implemented the linear mixed effects model 3 in JAGS. Here we report the results of a simulation study in which this model was fit to data-sets generated so as to exhibit properties of interest.

We generate data sets from an experimental design with one factor and two levels within this factor. As before, the dependent measure will be reading time and the experimental manipulation will be ungrammaticality. This data-set can be generated quite straightforwardly in R. The relevant code chunks are provided below. Complete code can be found in an online appendix. The fixed effects structure for J subjects and K items is generated as follows.

```
beta_0 <- 600
beta_1 <- 10
x <- rep(0:1, J*K)
```

This corresponds exactly to the fixed effects structure of model 3. The intercept β_0 is 600, the adjustment β_1 is 10, and the treatment coded categorical predictor x_i is coded as a vector of zeros and ones which repeats $J \times K$ times so that each each subject reads each item in both the grammatical and the ungrammatical condition.

The random effects structure of model 3 is defined in terms of the patterns of variance and covariance which the population of subjects and items displays. Once this variance-covariance structure is specified we can randomly draw subjects and items from their respective populations. We now illustrate this for by-subject random effects. First we define the population parameters σ_{u_0}, σ_{u_1}, and ρ_u.

```
sigma_u0 <- 10
sigma_u1 <- 10
rho_u <- 0.6
```

These population parameters stand in direct correspondence to the model parameters which define the variance-covariance matrix Σ_u in equation 4. We now define the variance-covariance matrix Σ_u, which determines the pattern of variance and covariance displayed by the subjects.

```
Sigma_u <- matrix(c(sigma_u0^2,
                    rho_u*sigma_u0*sigma_u1,
                    rho_u*sigma_u0*sigma_u1,
                    sigma_u1^2),nrow=2)
```

Now that we have defined the way in which by-subject random intercepts and slopes will vary, we draw J random intercept and slope pairs from a multivariate normal distribution. For this we use the R package `MASS`.

```
u <- mvrnorm(n=J,c(0,0),Sigma_u)
```

This is a J-by-2 matrix whose first column contains J by-subject random intercepts u_{0j} and whose second column contains J by-subject random slopes u_{1j}. Thus, each row contains the random intercept and slope for one subject. We repeat this same procedure to generate K by-item random intertercepts w_{0k} and slopes w_{1k}.

Now that we have defined the fixed and random effects structure of our model, we turn next to the error component, ϵ. For a data set with two conditions, J subjects, and K items there will be $2 \times J \times K$ observations. We generate one error term for each observation by drawing $2 \times J \times K$ numbers from a normal distribution with mean zero and standard deviation 40.

```
sigma_e <- 40
epsilon  <- rnorm(2*J*K, 0, sigma_e)
```

Each dependent measure RT_i will be determined by the sum $\beta_0 + u_{0j} + w_{0k} + (\beta_1 + u_{1j} + w_{1j})x_i + \epsilon_i$. This is what the following command achieves for each row i in the data frame:

```
RT[i] ~ dnorm( mu[i], tau.e )
```

Given that we have just generated a data-set whose structure corresponds exactly to that which is assumed by model 3, we should be able to find reasonable estimates for these population parameters using the JAGS implementation specified above. In practice, whether we can obtain reasonable estimates depends on the subject and item sample sizes J and K, as well as the stochastic process which generates

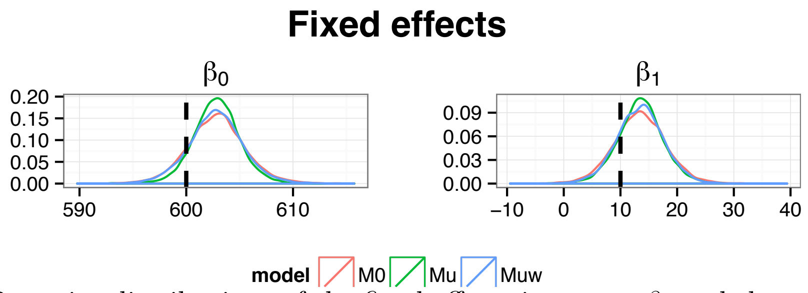

Figure 1. Posterior distributions of the fixed effects intercept β_0 and slope β_1 plotted against the known true values as vertical dashed lines.

a given sample of J subjects and K items. In order to assess the goodness of fit, we choose a moderate sample size of 25 subjects and 16 items. This is a fairly typical sample size in psycholinguistic experiments.

Three models will be fit to the same data. The first model M_0 places no hyperprior on the mean and precision of the prior distributions of ρ_u and ρ_w; instead, the prior distributions have mean zero and low precision. M_0 is included to assess whether a correlation parameter with hyperpriors yields a better estimate than a correlation parameter which has none. The second model M_u has no by-item random slope. It is included to demonstrate that the goodness of fit will deteriorate when the structure of the data is richer than the model assumes. The varying intercepts, varying slopes model 3 is referred to as model M_{uw} here.

Assessing the goodness-of-fit in the Bayesian setting involves drawing samples of model parameters from the posterior as a means of approximating the posterior probability density function. To illustrate this, we sample β_0 and β_1 from their respective posteriors and examine these distributions. The fact that we know the true values of β_0 and β_1 enables a direct comparison of the posterior distribution with these point values. In figure 1 we see that the three models essentially agree on their estimates $\hat{\beta}_0$ and $\hat{\beta}_1$. This is expected because the fixed effects structure of all three models exactly mirrors the way in which the data were generated.

The visual impression conveyed by figure 1 can be sharpened numerically. In table 4 we report the *credible intervals* for β_0 and β_1 as derived under different models. The credible interval is the range in which the β_0 and β_1 lie with probability 0.95. This inference can be made directly, although here we make the usual caveat that the result is contingent upon model assumptions, which include the form of the model and the covariance structure which its parameters exhibit. With this in mind, we could infer from model M_{uw} that β_1 is between 5.02 and 21.72 with 95% certainty.

The posterior distribution of the fixed effects coefficients is not very revealing about the differences among models M_{uw}, M_u, and M_0. Recall that we fit M_u and M_0 in order to assess the adequacy of the random effects structure assumed in M_{uw}. To evaluate this we sample the random effects correlation parameters ρ_u and ρ_w from their respective posterior distributions. Recall that the random effects were generated

	true value	posterior mean	CrI: 2.5th% ile	CrI: 97.5th% ile
$M_{uw} : \beta_0$	600	602.78	597.57	608.02
$M_{uw} : \beta_1$	10	13.47	5.02	21.72
$M_u : \beta_0$	600	602.84	598.51	607.25
$M_u : \beta_1$	10	13.53	5.54	21.52
$M_0 : \beta_0$	600	602.81	597.42	608.40
$M_0 : \beta_1$	10	13.45	4.38	22.52

Table 4

Posterior statistics on β_0 and β_1. The row names give the model and the parameter. The columns give the known population parameter value, the posterior mean, and the upper and lower bounds on the 95% credible intervals.

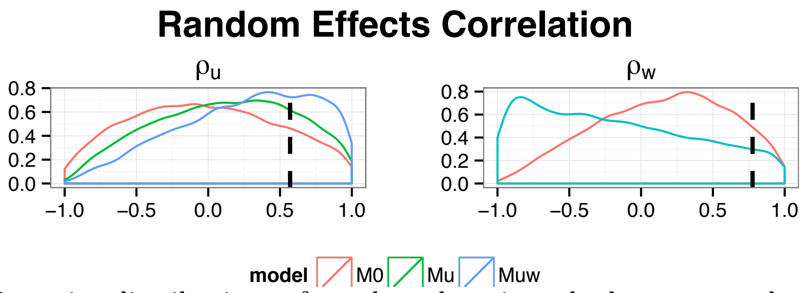

Figure 2. Posterior distributions of ρ_u plotted against the known sample correlation r_u and r_w as vertical dashed lines.

such that both the by-subject and by-item random intercepts and slopes are correlated with ρ_u and ρ_w equal to 0.6. The correlation of the random intercepts and slopes which we drew from multivariate normal distributions, however, deviated somewhat from this, due to sampling variability. The sample correlations r_u and r_w for the subject and item random effects are 0.57 and 0.776, respectively. Figure 2 shows the posterior distributions of ρ_u and ρ_w plotted against these sample correlations.

The posterior density function of ρ_u is visibly better for the full model M_{uw}. We see that the inclusion of a hyperprior has the effect of heightening the posterior's sensitivity to the likelihood function, which is the distribution of the data given the parameters. In contrast, the posterior density derived by M_0 is far more conservative. Most of its probability density lies near zero. This makes sense considering that the prior on ρ_u has mean zero in this model. For model M_0 we see that, although there is a hyperprior on ρ_u, the estimate is less precise. Here we see that the failure to account for by-item variance in the ungrammatical condition increases the noise present in the estimate of by-subject variance in the ungrammatical condition. M_{uw} recovers the underlying parameters best from among the three models considered. Table 5 corroborates this graphical argument with the posterior means and credible intervals, which we derive from the posterior density functions.

The posterior density function of ρ_w is poor for both models which have by-item random slopes. The estimate is particularly bad for the model with a hyperprior on ρ_w. This is an indication that a sample size of 16 is too small to estimate correlation. Nevertheless, even in cases where we do not care to include enough items to estimate ρ_w we should not leave out the random slopes entirely, as we saw that this affects the accuracy of our estimate $\hat{\rho}_u$.

	sample correlation	posterior mean	CrI: 2.5%ile	CrI: 97.5%ile
$M_{uw} : \rho_u$	0.57	0.26	-0.72	0.96
$M_{uw} : \rho_w$	0.78	-0.18	-0.97	0.91
$M_u : \rho_u$	0.57	0.11	-0.79	0.93
$M_0 : \rho_u$	0.57	-0.02	-0.90	0.91
$M_0 : \rho_w$	0.78	0.15	-0.74	0.91

Table 5
Posterior statistics on ρ_u and ρ_w. The rows names give the model and parameter. The columns give the known population parameter value, the posterior mean, and the lower and upper bounds on the 95% credible interval. Note that M_u has no by-item random slope and therefore no correlation parameter ρ_w.

Fitting more complex Bayesian models

In this section we extend the model to more complicated experimental designs. This gives us an opportunity to introduce the probabilistic programming language Stan, which excels at fitting data sets with a multi-factorial, multi-level design. We begin by introducing such a model and implement it in Stan. Then we demonstrate how to generate more involved data sets for the purposes of simulation before evaluating the Stan-implemented model. There we introduce the notion of posterior predictive checking and related test statistics (Gelman et al., 2014).

Stan syntax

The model which we fit here is the same as model 3 except that it has a second predictor variable x_1 with a fixed slope β_2, a random slope u_2 for by-subject variance, and a random slope w_2 for by-item variance.

$$\mathrm{RT}_i = \beta_0 + u_{0j} + w_{0k} + (\beta_1 + u_{1j} + w_{1k})x_{0i} + (\beta_2 + u_{2j} + u_{2k})x_{1i} + \epsilon_i \tag{8}$$

We are now representing two factors with the indicator variables x_0 and x_1. They are coded using centered constrasts: -0.5 and $+0.5$ for each level of each predictor. For a 2×2 design, this will results in an ANOVA contrast coding.

As before, our model assumes that the by-subject and by-item random intercepts and slopes covary. Each subject and item has one intercept and two slopes, and so the variance-covariance matrices Σ_u and Σ_w are now 3×3 matrices. We define Σ_u as follows, and Σ_w has the same structure.

$$\Sigma_u = \begin{bmatrix} \sigma_{u0}^2 & \rho_{u01}\,\sigma_{u0}\,\sigma_{u1} & \rho_{u02}\,\sigma_{u0}\,\sigma_{u2} \\ \rho_{u10}\,\sigma_{u1}\,\sigma_{u0} & \sigma_{u1}^2 & \rho_{u12}\,\sigma_{u1}\,\sigma_{u2} \\ \rho_{u20}\,\sigma_{u2}\,\sigma_{u0} & \rho_{u21}\,\sigma_{u2}\,\sigma_{u1} & \sigma_{u2}^2 \end{bmatrix} \tag{9}$$

There are more parameters now, but the interpretation which we give to them is similar to the simpler case discussed earlier. For instance, ρ_{u12} is the correlation between the by-subject slopes u_{1j} and u_{2j}. The parameter σ_{u0} is the variance of the by-subject intercepts u_0. We generated the random intercepts and slopes by drawing them from a multivariate normal distribution with a 3×3 variance-covariance matrix. We specified this matrix by plugging in constants for all the paramters in 9. Thus, the covariance structure 9 which our model assumes corresponds to the structure in the data.

We now proceed to specify a model in Stan. A Stan model file is split into four code blocks. The first is called the *data* block. Here we assign types and lower and upper bounds to the variables in the data that are passed to the model. The data will contain each of these variables as a list; see appendix for details.

```
data{
  real<lower=-0.5,upper=0.5> x0[N];
  real<lower=-0.5,upper=0.5> x1[N];
  real RT[N];

  int<lower=1> N; // number of observations
  int<lower=1> J; // number of subjects
  int<lower=1> K; // number of items

  int<lower=1,upper=J> subj[N]; // subject ID
  int<lower=1,upper=K> item[N]; // item ID

  vector[3] zero; // a vector of zeros
}
```

The strings `real`, `int`, and `vector` specify the data type for each variable. The variables `x0`, `x1`, and `RT` are vectors which are defined in the data. Each of their N elements is declared to be of the `real` data type. The variables `N`, `J`, and `K` are integers passed from R, so they are assigned the `int` data type. The vector `subj` contains N integer elements which indicate the subject associated with the n^{th} reading time. So if the n^{th} element is 3, then the n^{th} reading time is associated with subject 3. The same goes for `item`. We also declare a vector `zero` of three zeros.

Next we have the *parameters* block and the *transformed parameters* block. Here we merely declare model parameters. We do not yet define priors for them. The difference between these two blocks is that the parameters block declares parameters which are defined without reference to other model parameters. In the transformed parameters block we declare parameters in terms of the simpler parts which were declared in the parameters block.

This is what the parameters block looks like.

```
parameters{
  vector[3] beta; // fixeffs
  vector[3] u[J]; // subj raneffs
  vector[3] w[K]; // item raneffs
  real<lower=0> sigma_e; // residual variance
  vector<lower=0>[3] sigma_u; // subj variance
  vector<lower=0>[3] sigma_w; // item variance
  corr_matrix[3] Omega_u; // corr matrix for
                          // subj raneffs
  corr_matrix[3] Omega_w; // corr matrix for
                          // item raneffs
}
```

The vector `beta` contains the three fixed effects coefficients for intercept β_0, and the two slopes β_1 and β_2. The expression `vector[3] u[J]` is a 3-by-J matrix which contains the random intercept u_0 and slopes u_1 and u_2 for the J subjects. The same goes for the by-item random effects, which are specified by `vector[3] w[K]`. Recall that we assume the residuals of model 8 to have mean zero and unknown variance. This residual variance is `sigma_e`. Stan has the data type `corr_matrix` for a correlation matrix. This is a matrix of the parameters which determines correlation between random effects coefficients. This is just a handy way to keep track of the ρ_us. The by-subject random effects correlation matrix has the following structure.

$$\Omega_u = \begin{bmatrix} 1 & \rho_{u12} & \rho_{u13} \\ \rho_{u21} & 1 & \rho_{u23} \\ \rho_{u31} & \rho_{u32} & 1 \end{bmatrix} \tag{10}$$

We need this matrix to define the random effects variance-covariance matrix Σ_u. Recall from equation 9 that the variance-covariance matrix Σ_u is defined in terms of variance and correlation parameters. These latter parameters were declared in the parameters block, and we now use these in the transformed parameters block to define Σ_u and Σ_w.

```
transformed parameters{
  cov_matrix[3] Sigma_u; // varcov matrix for subj ranefs
  cov_matrix[3] Sigma_w; // varcov matrix for item ranefs
  for(r in 1:3){
    for(c in 1:3){
      Sigma_u[r,c] <- sigma_u[r] * sigma_u[c] * Omega_u[r,c];
      Sigma_w[r,c] <- sigma_w[r] * sigma_w[c] * Omega_w[r,c];
    }
  }
}
```

So Σ_u and Σ_w are objects of data type `cov_matrix`. They are defined in a control structure which loops over the variance of the by-subject and by-item random effects coefficients. All the parameters used to define the variance-covariance matrices were defined in the paramters block.

We now have defined all the parameters which will be used in the model specification. The fourth block of code in a Stan model file is the *model* block. The block has the form given below.

```
model{
  ...
}
```

This is where priors are placed on the parameters which we declared above, and where the likelihood is defined. We go through this line by line, beginning with the subject random effects. The by-subject random intercept u_0 and slopes u_1, u_2 are drawn from a multivariate normal distribution with mean zero and unknown variance and covariance. Notice that unlike JAGS, in Stan the variance-covariance matrix is specified using the standard specification of the multivariate normal distribution in terms of a variance-covariance matrix, not a precision matrix.

```
for(j in 1:J){
  u[j] ~ multi_normal(zero,Sigma_u);
}
```

Just as in JAGS, drawing a vector of variables from a multivariate distribution imposes covariance structure on the elements of that vector. The variance and covariance for u_0, u_1, and u_2 is determined by the matrix Σ_u. Here we do not place a prior on Σ_u directly. Instead we place priors on its parts. In particular, we place gamma priors on each of σ_{u0}, σ_{u1}, and σ_{u2} with shape parameter 1.5 and scale parameter some small number (Chung et al., 2013); Gelman suggests (p.c.) that this should be replaced with a more informative prior. Figure 3 shows the prior distribution.

```
sigma_u ~ gamma(1.5,1.0E-4);
sigma_w ~ gamma(1.5,1.0E-4);
sigma_e ~ gamma(1.5,1.0E-4);
```

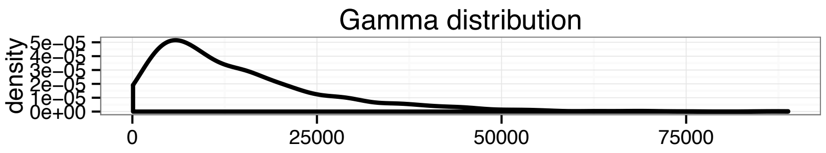

Figure 3. The gamma distribution with shape parameter 1.5 and scale parameter some small number.

The Stan language has a built-in implementation of the lkj-prior, which is a prior on correlation matrices. It takes a positive scalar value η as its parameter. We consider the effect of varying η in detail in a later section. For now it is sufficient to understand that a correlation matrix can have a prior specified in the following manner:

```
Omega_u ~ lkj_corr(2.0);
Omega_w ~ lkj_corr(2.0);
```

We now define the likelihood in terms of a mean `mu` and a residual variance parameter `sigma_e`. Note that `mu` is not a model parameter, but only a variable used to define the likelihood function. As in the JAGS model, the likelihood function determines how the model parameters interact to generate the dependent variable RT. Again, note here that unlike JAGS, Stan defines the normal distribution in terms of the mean and standard deviation, not the mean and precision.

```
real mu[N];
for(i in 1:N){
  mu[i] <-  (beta[1] + u[subj[i],1] + w[item[i],1])
            + (beta[2] + u[subj[i],2] + w[item[i],2])*c1[i]
            + (beta[3] + u[subj[i],3] + w[item[i],3])*c2[i];
}
RT ~ normal(mu,sigma_e);
```

This concludes the specification of the model block. Together, the data, parameters, transformed parameters, and model blocks make up the Stan model specification.

Stan simulations

In the previous section, we implemented the linear mixed effects model 8 in Stan. Here we report the results of a simulation study in which this model is fit to data sets generated to display properties of a 2 × 2 experimental design with one dependent measure and by-subject and by-item random effects. We first demonstrate how these data sets are generated using R before evaluating the fit of model 8 to several data sets which have been generated to display qualitatively different patterns of random effects correlation. In assessing the model, we introduce the procedure of *posterior predictive checking* advocated by Gelman et al. (2014), and implement it in Stan. We then evaluate the accuracy of the model's random effects correlation parameter estimates under different choices of η for the prior distributions on the correlation matrices Ω_u and Ω_w. We conclude the section by fitting the model to a data-set whose by-item random effects all have correlation near zero. We show that the maximal model outperforms a model which assumes no by-item random effects covariance structure even in this case.

We declare the fixed intercept β_0 and slopes β_1 and β_2 as follows.

```
beta <- c(600,10,20)
```

The fixed intercept is of magnitude 600 and the treatment levels of the two categorical predictors are adjusted by 10 and 20 up from the reference levels of the respective factors.

We have now declared the magnitude of the fixed effects and we move on to the random effects. Both model 3 and model 8 assume a "maximal" covariance structure among the random effects (Barr, Levy, Scheepers, & Tily, 2013). Recall that the random effects structure of model 3 consisted of an intercept and a single slope. The random model components which could covary were thus only two. There was only one correlation parameter for each the by-subject and by-item random effects, and it determined how the random intercepts and slopes covaried. Model 8 now includes a second slope, and so the maximal covariance structure is somewhat richer. Not only can both random slopes covary with the random intercept, the two slopes can also covary with each other. Thus, we estimate three correlation parameters for each by-subject and by-item random effects. For by-subject random effects we have the correlation ρ_{u01} between the intercept u_0 and slope u_1, the correlation ρ_{u02} between the intercept u_0 and slope u_1, and the correlation ρ_{u12} between the two slopes u_1 and u_2. In general, we denote the i, j by-subject random effects correlation pararameter as ρ_{uij}. The same goes for by-item random effects w.

We now declare the correlation matrix Ω_u for the by-subject random effects.

```
Omega_u <- matrix(c(1.0,0.6,0.6,
                    0.6,1.0,0.6,
                    0.6,0.6,1.0),nrow=3)
```

This corresponds to the Stan model correlation matrix which we defined in equation 10. It is a square matrix of order 3 whose off-diagonal entries are the correlation parameters. Next, we quite arbitrarily declare the standard deviations σ_{u1}, σ_{u2}, and σ_{u3} of the by-subject random effects to be of magnitude 10. In general these need not be identical.

```
sdev_u <- c(10,10,10)
```

The correlation parameters of Ω_u and the standard deviations σ_{u1}, σ_{u2}, and σ_{u3} define the variance-covariance matrix Σ_u for by-subject random effects.

```
Sigma_u <- matrix(rep(NA,3^2),ncol=3)
for(i in 1:3){
  for(j in 1:3){
    Sigma_u[i,j] <- sdev_u[i]*sdev_u[j]*Omega_u[i,j]
  }
}
```

We use this matrix to draw J by-subject random intercepts and slopes from a multivariate normal distribution. The entries of Σ_u specify the patterns of variance and covariance which the random effects of our J subjects will display.

```
raneff_u <- mvrnorm(n=J,c(0,0,0),Sigma_u)
```

Row j of `raneff_u` contains random draws of an intercept u_{0j} and two random slopes u_{1j} and u_{2j} for subject j. We repeat the same procedure to generate a matrix `raneff_w` whose entries are the random intercepts and slopes by item.

We have now specified the fixed effects β_0, β_1, and β_2 as the vector `beta`, the random effects u_{0j}, u_{1j}, and u_{2j} of subject j in row j of the matrix `raneff_u`, and the random effects w_{0k}, w_{1k}, and w_{2k} of item k in row k of the matrix `raneff_w`. Let us now assemble our data frame from these parts.

There are two factors, and each has two levels. Thus, the number of experimental conditions is four. If each of J subjects sees each each of K items in each of these four conditions, then we have $N = 4 \times J \times K$ observations.

```
n_factor <- 2
n_level <- 2
n_condition <- n_factor*n_level
N <- n_condition*J*K
```

For concreteness, let us take our dependent measure to be reading time RT. So row i of the data frame corresponds to RT_i, where $i = 1, \ldots, N$. We now specify which condition is associated with each RT_i. The vector `condition` takes on the value 1, 2, 3, or 4 depending on which condition the item was read in. Since each subject reads each item in each condition, the length of the vector should be $4 \times J \times K$.

```
condition <- rep(1:n_condition, J*K)
```

Now that we know which condition is associated with each RT_i, we have to determine by how much the fixed slopes will adjust the fixed intercept in each condition. We see jointly the effects of β_1 and β_2 in the first condition, the individual effects of factors β_1 and β_2 in the second and third conditions, respectively, and no effect in the fourth condition.

```
adjustment <- c(beta[1]+beta[2]+beta[3],
                beta[1]+beta[2],
                beta[1]+beta[3],
                beta[1])
fixeff <- rep(0,N)
for(i in 1:N){
  fixeff[i] <- adjustment[condition[i]]
}
```

Next we associate each RT_i with one subject and one item. We define as follows the vectors `subj` and `item`. These are vectors of length N which determine the subject and item associated with each RT_i.

```
item <- rep(1:K, J, each=n_condition)
subj <- rep(1:J, each=K*n_condition)
```

Now that we know which subject, which item, and which condition corresponds to each RT_i, we may determine the magnitude of the random effect on RT_i as follows.

```
ind1 <- c(TRUE,TRUE,FALSE,FALSE)
ind2 <- c(TRUE,FALSE,TRUE,FALSE)
for(i in 1:N){
  adjustment <- rep(0,n_cond)
  adjustment[ind1] <- raneff_u[subj[i],2]
  adjustment[ind2] <- adjustment[ind2] + raneff_u[subj[i],3]
  raneff_u[i] <- raneff_u[subj[i],1] + adjustment[cond[i]]
}
```

We repeat this for the by-item random effects to generate the vector `raneff_w`. We then take the sum `raneff_u + raneff_w` to get the vector `raneff`. Entry i of the vector `raneff` is the sum of the random effects for subject and for item on RT_i. Finally, we generate N random draws from a normal distribution with mean zero and arbitrary standard deviation. This is the noise component of the data.

```
sdev <- 40
epsilon <- rnorm(N,0,sdev)
```

We assemble RT_i as follows.

```
RT <- fixeff + raneff + epsilon
```

We now fit the Stan model presented in section to a data sets which has been generated following the procedure outlined just above. We assess the adequacy of the model through a procedure called *posterior predictive checking* (Gelman et al., 2014).

As we will apply it here, posterior predictive checking is a process whereby a posterior quantity of interest is drawn from the posterior distribution of the model. In this case we are interested in generating random draws of RT from the posterior distribution. This process is predictive in the sense that the draws are the RTs which the model predicts on the basis of the interaction between the likelihood and the prior. Let us denote a vector of N predicted RTs as $\widetilde{\text{RT}}$. We repeatedly sample such vectors, each of which represents a set of RTs which we might observe in an experiment like the one which generated the observed dependent measures RT. We suppose that if the vectors $\widetilde{\text{RT}}$ which our model generates resemble the one which we actually observed, then the model predictions are adequate. Posterior predictive checking is a simple way to check the adequacy of a statistical model, as such a comparison proceeds more or less directly. One comparison which can be made is a comparison of the observed maximum and minimum RT with the distribution of maxima and minuma of $\widetilde{\text{RT}}$.

We generate vectors $\widetilde{\text{RT}}$ of length N by declaring a parameter `RT_tilde` in the parameters block.

```
real RT_tilde[N];
```

We draw $\widetilde{\text{RT}}$ from the normal distribution in the model block using the same sampling statement as was used to draw RT.

```
RT_tilde ~ normal(mu,sigma_e);
```

We would like to find the maximum and the minimum of each vector $\widetilde{\text{RT}}$ in order to derive a posterior distribution for the maxima and minima predicted under the model. In Stan we do this in a separate code block called the *generated quantities* block, which we declare as follows.

```
generated quantities{
  real minimum;
  real maximum;
  minimum <- min(RT_tilde);
  maximum <- max(RT_tilde);
}
```

We generate a data set as described above, letting the number J of subjects equal 35 and the number K of items equal 16. We then estimate the probability density of the minima and maxima predicted by the model using the vectors `minimum` and `maximum`. Figure 4 graphically compares the distributions of these posterior predictive quantities to the observed maximum and minimum RT. Visual inspection suggests that the posterior predictive distributions of MAX($\widetilde{\text{RT}}$) and MIN($\widetilde{\text{RT}}$) have non-negligible probability density over the interval surrounding the observed extrema, which means that the observed values are plausible.

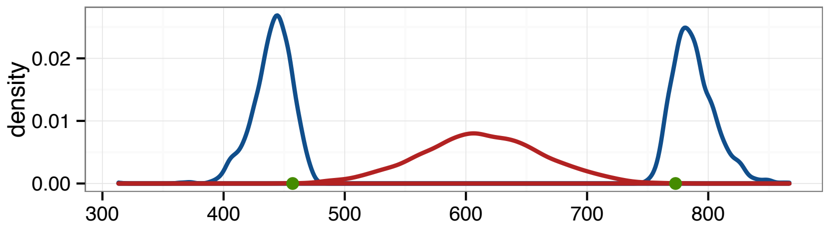

Figure 4. Posterior predictive distribution for extrema (blue) plotted along with the observed maximum and minimum (green dots). The kernel density estimate of the observed RT is given in red.

At this point we introduce a test statistic T, which is the probability that a predicted extremum is more extreme than the observed value. Table 6 gives these probabilities. The use of such a test statistic sharpens the impression gleaned visually from figure 4. The non-negligible probability mass to the left and right of the observed extrema suggests that the model is at least adequate enough to predict similar outcomes.

	observed	posterior mean	CrI: 2.5%ile	CrI: 97.5%ile	$T(\widetilde{\mathrm{RT}})$
minimum	457	440	404	467	0.85
maximum	773	788	761	828	0.81

Table 6
Posterior predictive statistics for minimum and maximum RT under the Stan model. Columns give the observed value, the posterior mean, the upper and lower bounds of the highest posterior density intervals, and the probability that the extrema of $\widetilde{\mathrm{RT}}_i$ *are more extreme than the observed extrema of* RT*.*

Posterior predictive checking is a very flexible method for model evaluation, and accordingly we will avoid specifying a test statistic or procedure which is intended to apply generally. The posterior predictive quantity and the corresponding test statistic should instead be determined by the research question which the data analysis addresses. For instance, a researcher may be interested in the distribution of a dependent measure in a particular experimental condition. In this case the posterior predictive distribution will be conditional on the value of a predictor like x_0 or x_1. These can then be directly compared to the observations within that condition. Posterior predictive distributions for model parameters such as u_0 or Ω_{u01} can also be derived. The comparison here will of course not be with observed quantities, as model parameters — unlike dependent measures — are not observed. Nevertheless, model evaluation can proceed given an explicit model that expresses how the data were generated.

We now turn to the random effects correlation estimates of the Stan model

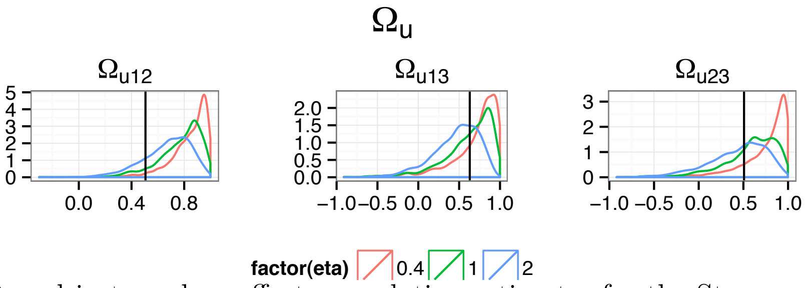

Figure 5. By-subject random effects correlation estimates for the Stan model with η successively being given the values $0.4, 1, 2$.

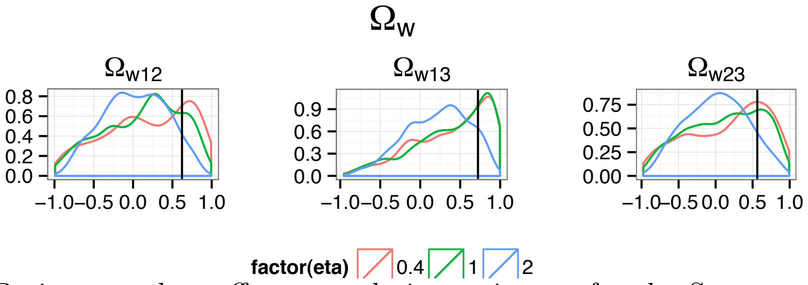

Figure 6. By-item random effects correlation estimates for the Stan model with η successively being given the values $0.4, 1, 2$.

and examine their posterior distribution in more detail. As mentioned in section , we placed lkj prior distributions on the correlation matrices Ω_u and Ω_w. This prior distribution takes as a parameter a positive scalar value η. An lkj prior with η equal to one corresponds to a uniform prior over the space of correlation matrices of a given order (2-by-2, 3-by-3, etc.). When η is between zero and one, there is a trough along the diagonal, meaning that for each correlation parameter the prior probability density is shifted away from zero toward the upper and lower bounds of the interval $[-1, 1]$. When η is greater than one, greater prior probability density lies near zero.

We fit the Stan model to the same data using different values for η. Again, we let the number of participants be $J = 35$ and the number of items $K = 16$. The population parameters Ω_u and Ω_w both uniformly had 0.6 on the off-diagonals. We vary the parameters η_u on the prior for Ω_u, and η_w on the prior for Ω_w successively over three values, $0.4, 1, 2$. In each simulation, η_u and η_w had identical values. Figure 5 demonstrates that the resulting posterior density estimates become more accurate as η_u is increased. Figure 6 shows that there is non-negligible probability density over the entire interval $[-1, 1]$. This suggests that sample size $K = 16$ is too small to estimate $\hat{\Omega}_w$. This is consistent with the results from the JAGS simulations presented earlier.

Case studies with real data

coming soon

Further reading

Gelman and Hill (2007) is an accessible introduction to fitting linear mixed models using BUGS syntax, which is what JAGS uses. The Gelman and Hill book has Stan translations available online, but at least some of this code has serious mistakes in it, so, at least at the present time, it is better to work with the BUGS code when reading the Gelman and Hill book). Lunn, Thomas, Best, and Spiegelhalter (2000) is another important book that provides a lot of good examples, also using BUGS syntax.

In order to learn more about Bayesian methods, Lynch (2007) strikes a nice compromise between formal rigor and accessibility. A less technical book is by Kruschke (2010). For mathematical and probability-theory background we would recommend Gilbert and Jordan (2002) and Kerns (2011) (the latter is available online). A more advanced treatment is provided by Gelman et al. (2014).

For those who can afford the time, we recommend that they do the nine-month graduate certificate in statistics taught online at the University of Sheffield, UK.

Appendix A
Linear mixed models using JAGS

Code for generating data for a two-condition repeated measures design

```
> ## This is the content of the file twocond_gen.R:
> new.df <- function(cond1.rt=600, effect.size=10,
                    sdev=40,
                    sdev.int.subj=10, sdev.slp.subj=10,
                    rho.u=0.6,
                    nsubj=10,
                    sdev.int.items=10, sdev.slp.items=10,
                    rho.w=0.6,
                    nitems=10) {
   library(MASS)

   ncond <- 2

   subj <- rep(1:nsubj, each=nitems*ncond)
   item <- rep(1:nitems, nsubj, each=ncond)

   cond <- rep(0:1, nsubj*nitems)
   err  <- rnorm(nsubj*nitems*ncond, 0, sdev)
   d    <- data.frame(subj=subj, item=item,
                      cond=cond+1, err=err)

   Sigma.u<-matrix(c(sdev.int.subj^2,
                     rho.u*sdev.int.subj*sdev.slp.subj,
                     rho.u*sdev.int.subj*sdev.slp.subj,
                     sdev.slp.subj^2),nrow=2)

   Sigma.w<-matrix(c(sdev.int.items^2,
                     rho.u*sdev.int.items*sdev.slp.items,
                     rho.u*sdev.int.items*sdev.slp.items,
                     sdev.slp.items^2),nrow=2)

   # Adding random intercepts and slopes for subjects:
   ## first col. has adjustment for intercept,
   ## secdon col. has adjustment for slope
   subj.rand.effs<-mvrnorm(n=nsubj,rep(0,ncond),Sigma.u)

   item.rand.effs<-mvrnorm(n=nitems,rep(0,ncond),Sigma.w)

   #  re.int.subj   <- rnorm(nsubj, 0, sdev.int.subj)
```

```
  re.int.subj <- subj.rand.effs[,1]
  d$re.int.subj <- rep(re.int.subj, each=nitems*ncond)
  #  re.slp.subj   <- rnorm(nsubj, 0, sdev.slp.subj)
  re.slp.subj   <- subj.rand.effs[,2]

  d$re.slp.subj <- rep(re.slp.subj,
                       each=nitems*ncond) * (cond - 0.5)

  # Adding random intercepts and slopes for items:
  #  re.int.item   <- rnorm(nitems, 0, sdev.int.items)
  re.int.item <- item.rand.effs[,1]
  d$re.int.item <- rep(re.int.item, nsubj, each=ncond)
  #  re.slp.item   <- rnorm(nitems, 0, sdev.int.items)
  re.slp.item <- item.rand.effs[,2]
  d$re.slp.item <- rep(re.slp.item, nsubj,
                       each=ncond) * (cond - 0.5)

  d$rt <- (cond1.rt + cond*effect.size
           + d$re.int.subj + d$re.slp.subj
           + d$re.int.item + d$re.slp.item
           + d$err)

  return(list(d,cor(re.int.subj,re.slp.subj),
              cor(re.int.item,re.slp.item)))
 }
```

Code for fitting models using JAGS

```
> ## We test whether the full model with a truncated
> ## normal prior on rho was overparameterized.
> ## We fit two models for  comparison.
> ## One has no by-item slopes, the other has
> ## no hyperparameters on rho.w and rho.u.
>
> ############################################################
> ###  generate data sets
> ############################################################
>
> source("./twocond_gen.R")
> dat25<-new.df(nsubj=25,nitems=16,rho.u=0.6,rho.w=0.6)
> d25 <- dat25[[1]]
> d25<-d25[,c(1,2,3,9)]
> d25$x0<-ifelse(d25$cond==1,-0.5,0.5)
> u_corr25 <- dat25[[2]]
> w_corr25 <- dat25[[3]]
> ## note that data is a list of vectors:
> dat25 <- list( subj = sort(as.integer(factor(d25$subj) )),
                item = sort(as.integer(factor(d25$item) )),
                rt = d25$rt,
                x0 = d25$x0,
                N = nrow(d25),
                J = length( unique(d25$subj) ),
                K = length( unique(d25$item) ))
> ###################################################
> ### full model: model M_uw in paper
> ###  truncated normal prior on rho,
> ###  gamma prior on sigma
> ###################################################
> cat("
    data
 {
    zero.u[1] <- 0
    zero.u[2] <- 0
    zero.w[1] <- 0
    zero.w[2] <- 0
 }
    model
 {
    # Intercept and slope for each person,
    # including random effects
    for( j in 1:J )
 {
    u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
    pred_u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
 }
    # Intercepts and slope by item
    for( k in 1:K)
 {
    w[k,1:2] ~ dmnorm(zero.w,invSigma.w)
 ## predicted by item effects:
    pred_w[k,1:2] ~ dmnorm(zero.w,invSigma.w)
 }

    # Define model for each observational unit
    for( i in 1:N )
```

```
{
    mu[i] <- ( beta[1] + u[subj[i],1] + w[item[i],1]) +
    ( beta[2] + u[subj[i],2] + w[item[i],2] ) * ( x0[i] )
    rt[i] ~ dnorm( mu[i], tau.e )
    ## predicted RTs:
    pred[i] ~ dnorm( mu[i], tau.e )
}

    minimum <- min(pred)
    maximum <- max(pred)
    mean <- mean(pred)

    # Fixed intercept and slope (uninformative)
    beta[1] ~ dnorm(0.0,1.0E-5)
    beta[2] ~ dnorm(0.0,1.0E-5)

    # Residual variance
    tau.e <- pow(sigma.e,-2)
    sigma.e  ~ dunif(0,15)

    # variance-covariance matrix of subject ranefs
    invSigma.u  ~ dwish( R.u , 2 )
    R.u[1,1] <- pow(sigma.a,2)
    R.u[2,2] <- pow(sigma.b,2)
    R.u[1,2] <- rho.u*sigma.a*sigma.b
    R.u[2,1] <- R.u[1,2]
    Sigma.u <- inverse(invSigma.u)

    # varying intercepts, varying slopes
    tau.a ~  dgamma(1.5, pow(1.0,-4))
    tau.b ~ dgamma(1.5, pow(1.0,-4))
    sigma.a <- pow(tau.a,-1/2)
    sigma.b <- pow(tau.b,-1/2)

    # variance-covariance matrix of item ranefs
    invSigma.w ~ dwish( R.w, 2 )
    R.w[1,1] <- pow(sigma.c,2)
    R.w[2,2] <- pow(sigma.d,2)
    R.w[1,2] <- rho.w*sigma.c*sigma.d
    R.w[2,1] <- R.w[1,2]
    Sigma.w <- inverse(invSigma.w)
```

```
     # varying intercepts, varying slopes
     tau.c ~ dgamma(1.5, pow(1.0,-4))
     tau.d ~ dgamma(1.5, pow(1.0,-4))
     sigma.c <- pow(tau.c,-1/2)
     sigma.d <- pow(tau.d,-1/2)

     # correlation
     rho.u ~ dnorm(mu_rho.u,tau_rho.u)T(-1,1)
     mu_rho.u ~ dunif(-1,1)
     tau_rho.u ~ dgamma(1.5,10E-4)
     # correlation
     rho.w ~ dnorm(mu_rho.w,tau_rho.w)T(-1,1)
     mu_rho.w ~ dunif(-1,1)
     tau_rho.w ~ dgamma(1.5,10E-4)
 }",file="sim_over.jag")
> ####################################################
> ### no rho hyperparameters model
> ###  truncated normal prior on rho,
> ###  gamma prior on sigma
> ### model M_0 in paper
> ####################################################
> cat("
     data
 {
     zero.u[1] <- 0
     zero.u[2] <- 0
     zero.w[1] <- 0
     zero.w[2] <- 0
 }
     model
 {
     # Intercept and slope for each person, including random effects
     for( j in 1:J )
 {
     u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
     pred_u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
 }
     # Intercepts and slope by item
     for( k in 1:K)
 {
     w[k,1:2] ~ dmnorm(zero.w,invSigma.w)
     pred_w[k,1:2] ~ dmnorm(zero.w,invSigma.w)
 }

     # Define model for each observational unit
     for( i in 1:N )
 {
     mu[i] <- ( beta[1] + u[subj[i],1] + w[item[i],1]) +
     ( beta[2] + u[subj[i],2] + w[item[i],2] ) * ( x0[i] )
     rt[i] ~ dnorm( mu[i], tau.e )
     pred[i] ~ dnorm( mu[i], tau.e )
 }

     minimum <- min(pred)
     maximum <- max(pred)
     mean <- mean(pred)

     # Fixed intercept and slope (uninformative)
     beta[1] ~ dnorm(0.0,1.0E-5)
     beta[2] ~ dnorm(0.0,1.0E-5)

     # Residual variance
     tau.e <- pow(sigma.e,-2)
     sigma.e  ~ dunif(0,15)

     # variance-covariance matrix of subject ranefs
     invSigma.u  ~ dwish( R.u , 2 )
     R.u[1,1] <- pow(sigma.a,2)
     R.u[2,2] <- pow(sigma.b,2)
     R.u[1,2] <- rho.u*sigma.a*sigma.b
     R.u[2,1] <- R.u[1,2]
     Sigma.u <- inverse(invSigma.u)

     # var intercepts, var slopes
     tau.a ~  dgamma(1.5, pow(1.0,-4))
     tau.b ~ dgamma(1.5, pow(1.0,-4))
     sigma.a <- pow(tau.a,-1/2)
     sigma.b <- pow(tau.b,-1/2)

     # variance-covariance matrix of item ranefs
     invSigma.w ~ dwish( R.w, 2 )
     R.w[1,1] <- pow(sigma.c,2)
     R.w[2,2] <- pow(sigma.d,2)
     R.w[1,2] <- rho.w*sigma.c*sigma.d
     R.w[2,1] <- R.w[1,2]
     Sigma.w <- inverse(invSigma.w)

     # var intercepts, var slopes
     tau.c ~ dgamma(1.5, pow(1.0,-4))
     tau.d ~ dgamma(1.5, pow(1.0,-4))
     sigma.c <- pow(tau.c,-1/2)
     sigma.d <- pow(tau.d,-1/2)

     # correlation
     rho.u ~ dnorm(0,1)T(-1,1)
     # correlation
     rho.w ~ dnorm(0,1)T(-1,1)
 }",file="sim_under1.jag")
> ####################################################
> ### no by-item slopes model
> ###  truncated normal prior on rho,
> ###  gamma prior on sigma
> ### Model M_u in paper
> ####################################################
> cat("
     data
 {
     zero.u[1] <- 0
     zero.u[2] <- 0
 }
     model
 {
     # Intercept and slope for each person, including random effects
     for( j in 1:J )
 {
     u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
     pred_u[j,1:2] ~ dmnorm(zero.u,invSigma.u)
 }
     # Intercepts by item
     for( k in 1:K)
 {
     w[k] ~ dnorm(0,tau.w)
     pred_w[k] ~ dnorm(0,tau.w)
 }

     # Define model for each observational unit
     for( i in 1:N )
 {
     mu[i] <- ( beta[1] + u[subj[i],1] + w[item[i]]) +
     ( beta[2] + u[subj[i],2]) * ( c0[i] )
     rt[i] ~ dnorm( mu[i], tau.e )
     pred[i] ~ dnorm( mu[i], tau.e )
 }

     minimum <- min(pred)
     maximum <- max(pred)
     mean <- mean(pred)

     # Fixed intercept and slope (uninformative)
     beta[1] ~ dnorm(0.0,1.0E-5)
     beta[2] ~ dnorm(0.0,1.0E-5)

     # Residual variance
     tau.e <- pow(sigma.e,-2)
     sigma.e  ~ dunif(0,15)

     # by-item itercept variance
     tau.w ~ dgamma(1.5, pow(1.0,-4))
     sigma.w <- pow(tau.w,-1/2)

     # variance-covariance matrix of subject ranefs
     invSigma.u  ~ dwish( R.u , 2 )
     R.u[1,1] <- pow(sigma.a,2)
     R.u[2,2] <- pow(sigma.b,2)
     R.u[1,2] <- rho.u*sigma.a*sigma.b
     R.u[2,1] <- R.u[1,2]
     Sigma.u <- inverse(invSigma.u)

     # var intercepts, var slopes
     tau.a ~  dgamma(1.5, pow(1.0,-4))
     tau.b ~ dgamma(1.5, pow(1.0,-4))
     sigma.a <- pow(tau.a,-1/2)
     sigma.b <- pow(tau.b,-1/2)

     # correlation
     rho.u ~ dnorm(mu_rho.u,tau_rho.u)T(-1,1)
     mu_rho.u ~ dunif(-1,1)
     tau_rho.u ~ dgamma(1.5,10E-4)
```

```
 }",file="sim_under2.jag")
> .libPaths("./")
> require(rjags)
> set.seed(9991)
> ############################################################
> ###  fit models
> ############################################################
>
> track.variables<-c("beta","sigma.e","sigma.a",
                     "sigma.b","rho.u","pred_u","pred_w")
> # full model: M_uw in paper
> sim_25_over.mod <- jags.model(
   file = "sim_over.jag",
   data = dat25,
   n.chains = 4,
   n.adapt = 200000,quiet=T)
> sim_25_over.res <- coda.samples(sim_25_over.mod,
                                   var = track.variables,
                                   n.iter = 50000,
                                   thin = 20 )
> save(list=c("sim_25_over.res"),
       file="RdaFiles/sim_overparam_full.Rda")
> # model lacking hyperprior on the hyperparameters
> # of rho.
> ## M_0 in paper:
> sim_25_under1.mod <- jags.model(
   file = "sim_under1.jag",
   data = dat25,
   n.chains = 4,
   n.adapt = 200000,quiet=T)
> sim_25_under1.res <- coda.samples(sim_25_under1.mod,
                                   var = track.variables,
                                   n.iter = 50000,
                                   thin = 20 )
> save(list=c("sim_25_under1.res"),
       file="RdaFiles/sim_overparam_under1.Rda")
> # model lacking by-item slopes
> ## M_u in paper:
> sim_25_under2.mod <- jags.model(
   file = "sim_under2.jag",
   data = dat25,
   n.chains = 4,
   n.adapt = 200000,quiet=T)
> sim_25_under2.res <- coda.samples(sim_25_under2.mod,
                                    var = track.variables,
                                    n.iter = 50000,
                                    thin = 20 )
> save(list=c("sim_25_under2.res"),
      file="RdaFiles/sim_overparam_under2.Rda")
```

Appendix B
Code for fitting a 2×2 design using Stan

```
> ## SENSITIVITY ANALYSIS FOR RANEFFS I
> ##
> ## We vary the shape parameter eta of the lkj correlation. In
> ## particular, we go from small eta (0.1) to large eta (4).
> ##
> ## The other thing we manipulate is whether the same full
> ## variance-covariance structure is assumed for the by-item
> ## random effects.
> ##
> ## In this script, we assume zero by-item raneff correlation.
> ##
> ## Note that, for eta = 1, the prior is uniform over all
> ## correlation matrices of a given order (in our case, the
> ## order is three). There is a trough at the diagonal when
> ## eta < 1 and a peak on the diagonal when eta > 1.
> ## (cf. Stan users manual).
>
> require(rstan)
> source("./two_by_two_gen.R")
> sessionInfo()
> ## This is the loop which fits models with no covariance
> ## structure for item raneffs
> for (eta in c(.4,1,2)){

   ###################################
   ###  DECLARE MODEL AS STRING
   ###################################
   code<-sprintf('
                 data{
                 int<lower=1> N; //no. rows
                 real<lower=-0.5,upper=0.5> c1[N];
                 real<lower=-0.5,upper=0.5> c2[N];
```

```
real rt[N];
int<lower=1> I; //no. subj
int<lower=1,upper=I> subj[N]; //subj id
int<lower=1> K; //no. item
int<lower=1,upper=K> item[N]; //item id
vector[3] zero;
}
parameters{
vector[3] beta; // fixeff
vector[3] u[I]; // by-subj raneff
vector[3] w[K]; // by-item raneff
real<lower=0> sigma_e; //resid
vector<lower=0>[3] sigma_u; //subj var
vector<lower=0>[3] sigma_w; //item var
corr_matrix[3] Omega_u; //corr matrix for
                        // subj ran int and slope
real y[N]; // post. pred.
}
transformed parameters{
cov_matrix[3] Sigma_u; // varcov matrix for
                       // subj ranefs
for(r in 1:3){
for(c in 1:3){
Sigma_u[r,c] <- sigma_u[r] * sigma_u[c] * Omega_u[r,c];
}
}
}
model{
real mu[N];
for(i in 1:N){
mu[i] <-  (beta[1] + u[subj[i],1] + w[item[i],1]) +
(beta[2] + u[subj[i],2] + w[item[i],2])*c1[i] +
(beta[3] + u[subj[i],3] + w[item[i],3])*c2[i];
}
for(i in 1:I){
u[i] ~ multi_normal(zero,Sigma_u); // subj ranef
}
for(k in 1:K){
w[k,1] ~ normal(0,sigma_w[1]); // item int
w[k,2] ~ normal(0,sigma_w[2]); // item slpA
w[k,3] ~ normal(0,sigma_w[3]); // item slpB
}
```

```
                rt ~ normal(mu,sigma_e);
                y ~ normal(mu,sigma_e);

                sigma_u ~ gamma(1.5,10E-4);
                sigma_w ~ gamma(1.5,10E-4);
                sigma_e ~ gamma(1.5,10E-4);
                Omega_u ~ lkj_corr(%f);
                }
                generated quantities{
                real minimum;
                real maximum;
                minimum <- min(y);
                maximum <- max(y);
                }
                ',eta)

###########################################################
###  GENERATE DATA
###########################################################
dat <- two_by_two(n_u=35,n_w=16,
                  Omega_w=matrix(c(1,0,0,
                                   0,1,0,
                                   0,0,1),nrow=3),
                  seed=999)
df <- dat[[1]]
d<-df[,c(1,2,3,4,10)]

# sum coding the two factors, which each have two levels
d$c1<-ifelse(d$factor_a==1,0.5,-0.5)
d$c2<-ifelse(d$factor_b==1,0.5,-0.5)

# u_corr[[1]] is by-subj. int-factor_a raneff correlation
# u_corr[[2]] is by-subj. int-factor_b raneff correlation
# u_corr[[3]] is by-subj. factor_a-factor_b raneff correlation
# and the same goes, mutatis mutandis, for the w_corr35, which
# is by-item raneffs.
u_corr <- dat[[2]]
w_corr <- dat[[3]]
dat <- list( subj = sort(as.integer(factor(d$subj) )),
             item = sort(as.integer(factor(d$item) )),
             rt = d$rt,
                c1 = d$c1,
                c2 = d$c2,
                N = nrow(d),
                I = length( unique(d$subj) ),
                K = length( unique(d$item) ),
                zero = rep(0,3))

   ###########################################################
   ###  FIT MODELS
   ###########################################################
   fit <- stan(model_code = code,
               data = dat,
               iter = 1500, warmup=1000, chains = 2)
   ###########################################################
   ###  SAVE
   ###########################################################
   save(list=c('fit','d','u_corr','w_corr'),
        file=sprintf('./w_nocov_eta%.2f.Rda',eta))
 }
> ## This is the loop which fits models with full covariance
> ## structure for both item and subject raneffs
> for (eta in c(.4,1,2)){

   ##################################
   ###  DECLARE MODEL AS STRING
   ##################################
   code<-sprintf('
   data{
   int<lower=1> N; //no. rows
     real<lower=-0.5,upper=0.5> c1[N];
     real<lower=-0.5,upper=0.5> c2[N];
     real rt[N];
     int<lower=1> I; //no. subj
     int<lower=1,upper=I> subj[N]; //subj id
     int<lower=1> K; //no. item
     int<lower=1,upper=K> item[N]; //item id
     vector[3] zero;
   }
   parameters{
   vector[3] beta; // fixeff
   vector[3] u[I]; // by-subj raneff
   vector[3] w[K]; // by-item raneff
   real<lower=0> sigma_e; //resid
   vector<lower=0>[3] sigma_u; //subj var
   vector<lower=0>[3] sigma_w; //item var
   corr_matrix[3] Omega_u; //corr matrix for
                           //subj ran int and slope
   corr_matrix[3] Omega_w; //corr matrix for
                           //item ran int and slope
   real y[N]; // post. pred.
   }
   transformed parameters{
   cov_matrix[3] Sigma_u; // varcov matrix for subj ranefs
   cov_matrix[3] Sigma_w; // varcov matrix for item ranefs
   for(r in 1:3){
   for(c in 1:3){
   Sigma_u[r,c] <- sigma_u[r] * sigma_u[c] * Omega_u[r,c];
   Sigma_w[r,c] <- sigma_w[r] * sigma_w[c] * Omega_w[r,c];
   }
   }
   }
   model{
   real mu[N];
   for(i in 1:N){
   mu[i] <-  (beta[1] + u[subj[i],1] + w[item[i],1]) +
   (beta[2] + u[subj[i],2] + w[item[i],2])*c1[i] +
   (beta[3] + u[subj[i],3] + w[item[i],3])*c2[i];
   }
   for(i in 1:I){
   u[i] ~ multi_normal(zero,Sigma_u); // subj ranef
   }
   for(k in 1:K){
   w[k] ~ multi_normal(zero,Sigma_w); // item ranef
   }

   rt ~ normal(mu,sigma_e);
   y ~ normal(mu,sigma_e);

   sigma_u ~ gamma(1.5,10E-4);
   sigma_w ~ gamma(1.5,10E-4);
   sigma_e ~ gamma(1.5,10E-4);
   Omega_u ~ lkj_corr(%f);
   Omega_w ~ lkj_corr(%f);
   }
```

```
generated quantities{
real minimum;
real maximum;
minimum <- min(y);
maximum <- max(y);
}
',eta,eta)

############################################################
###  GENERATE DATA
############################################################
dat <- two_by_two(n_u=35,n_w=16,
                  Omega_w=matrix(c(1,0,0,
                                   0,1,0,
                                   0,0,1),nrow=3),
                  seed=999)
df <- dat[[1]]
d<-df[,c(1,2,3,4,10)]

# sum coding the two factors, which each have two levels
d$c1<-ifelse(d$factor_a==1,0.5,-0.5)
d$c2<-ifelse(d$factor_b==1,0.5,-0.5)

# u_corr[[1]] is by-subj. int-factor_a raneff correlation
# u_corr[[2]] is by-subj. int-factor_b raneff correlation
# u_corr[[3]] is by-subj. factor_a-factor_b raneff correlation
# and the same goes, mutatis mutandis, for the w_corr35, which
# is by-item raneffs.
u_corr <- dat[[2]]
w_corr <- dat[[3]]
dat <- list( subj = sort(as.integer(factor(d$subj) )),
             item = sort(as.integer(factor(d$item) )),
             rt = d$rt,
             c1 = d$c1,
             c2 = d$c2,
             N = nrow(d),
             I = length( unique(d$subj) ),
             K = length( unique(d$item) ),
             zero = rep(0,3))

############################################################
###  FIT MODELS
```

```
   ############################################################
   fit <- stan(model_code = code,
               data = dat,
               iter = 1500, warmup=1000, chains = 2)
   ############################################################
   ###  SAVE
   ############################################################
   save(list=c('fit','d','u_corr','w_corr'),
        file=sprintf('./w_cov_eta%.2f.Rda',eta))
 }

> ## SENSITIVITY ANALYSIS FOR RANEFFS II
> ##
> ## We vary the shape parameter eta of the lkj correlation.
> ## In particular, we go from small eta (0.1) to large eta (4).
> ##
> ## The other thing we manipulate is whether the same full
> ## variance-covariance structure is assumed for the by-item
> ## random effects.
> ##
> ## In this script, we assume by-item correlation of 0.6.
> ##
> ## Note that, for eta = 1, the prior is uniform over all
> ## correlation matrices of a given order (in our case, the
> ## order is three). There is a trough at the diagonal when
> ## eta < 1 and a peak on the diagonal when eta > 1.
> ## (cf. Stan users manual).
>
> require(rstan)
> source("./two_by_two_gen.R")
> sessionInfo()
> ## This is the loop which fits models with no covariance
> ## structure for item raneffs
> for (eta in c(.4,1,2)){

   ###################################
   ###  DECLARE MODEL AS STRING
   ###################################
   code<-sprintf('
                 data{
                 int<lower=1> N; //no. rows
                 real<lower=-0.5,upper=0.5> c1[N];
                 real<lower=-0.5,upper=0.5> c2[N];
```

```
real rt[N];
int<lower=1> I; //no. subj
int<lower=1,upper=I> subj[N]; //subj id
int<lower=1> K; //no. item
int<lower=1,upper=K> item[N]; //item id
vector[3] zero;
}
parameters{
vector[3] beta; // fixeff
vector[3] u[I]; // by-subj raneff
vector[3] w[K]; // by-item raneff
real<lower=0> sigma_e; //resid
vector<lower=0>[3] sigma_u; //subj var
vector<lower=0>[3] sigma_w; //item var
corr_matrix[3] Omega_u; //corr matrix for
                        // subj ran int and slope
real y[N]; // post. pred.
}
transformed parameters{
cov_matrix[3] Sigma_u; // varcov matrix for
                       // subj ranefs
for(r in 1:3){
for(c in 1:3){
Sigma_u[r,c] <- sigma_u[r] * sigma_u[c] * Omega_u[r,c];
}
}
}
model{
real mu[N];
for(i in 1:N){
mu[i] <-  (beta[1] + u[subj[i],1] + w[item[i],1]) +
(beta[2] + u[subj[i],2] + w[item[i],2])*c1[i] +
(beta[3] + u[subj[i],3] + w[item[i],3])*c2[i];
}
for(i in 1:I){
u[i] ~ multi_normal(zero,Sigma_u); // subj ranef
}
for(k in 1:K){
w[k,1] ~ normal(0,sigma_w[1]); // item int
w[k,2] ~ normal(0,sigma_w[2]); // item slpA
w[k,3] ~ normal(0,sigma_w[3]); // item slpB
}
```

```
             rt ~ normal(mu,sigma_e);
             y ~ normal(mu,sigma_e);

             sigma_u ~ gamma(1.5,10E-4);
             sigma_w ~ gamma(1.5,10E-4);
             sigma_e ~ gamma(1.5,10E-4);
             Omega_u ~ lkj_corr(%f);
             }
             generated quantities{
             real minimum;
             real maximum;
             minimum <- min(y);
             maximum <- max(y);
             }
             ',eta)

############################################################
###  GENERATE DATA
############################################################
dat <- two_by_two(n_u=35,n_w=16,seed=999)
df <- dat[[1]]
d<-df[,c(1,2,3,4,10)]

# sum coding the two factors, which each have two levels
d$c1<-ifelse(d$factor_a==1,0.5,-0.5)
d$c2<-ifelse(d$factor_b==1,0.5,-0.5)

# u_corr[[1]] is by-subj. int-factor_a raneff correlation
# u_corr[[2]] is by-subj. int-factor_b raneff correlation
# u_corr[[3]] is by-subj. factor_a-factor_b raneff correlation
# and the same goes, mutatis mutandis, for the w_corr35, which
# is by-item raneffs.
u_corr <- dat[[2]]
w_corr <- dat[[3]]
dat <- list( subj = sort(as.integer(factor(d$subj) )),
             item = sort(as.integer(factor(d$item) )),
             rt = d$rt,
             c1 = d$c1,
             c2 = d$c2,
             N = nrow(d),
             I = length( unique(d$subj) ),
             K = length( unique(d$item) ),
                zero = rep(0,3))

   ###########################################################
   ###  FIT MODELS
   ###########################################################
   fit <- stan(model_code = code,
               data = dat,
               iter = 1500, warmup=1000, chains = 2)
   ###########################################################
   ###  SAVE
   ###########################################################
   save(list=c('fit','d','u_corr','w_corr'),
        file=sprintf('./w_nocov_eta%.2f_wcor6.Rda',eta))
 }
> ## This is the loop which fits models with full covariance
> ## structure for both item and subject raneffs
> for (eta in c(.4,1,2)){

   ###################################
   ###  DECLARE MODEL AS STRING
   ###################################
   code<-sprintf('
                 data{
                 int<lower=1> N; //no. rows
                 real<lower=-0.5,upper=0.5> c1[N];
                 real<lower=-0.5,upper=0.5> c2[N];
                 real rt[N];
                 int<lower=1> I; //no. subj
                 int<lower=1,upper=I> subj[N]; //subj id
                 int<lower=1> K; //no. item
                 int<lower=1,upper=K> item[N]; //item id
                 vector[3] zero;
                 }
                 parameters{
                 vector[3] beta; // fixeff
                 vector[3] u[I]; // by-subj raneff
                 vector[3] w[K]; // by-item raneff
                 real<lower=0> sigma_e; //resid
                 vector<lower=0>[3] sigma_u; //subj var
                 vector<lower=0>[3] sigma_w; //item var
                 corr_matrix[3] Omega_u; //corr matrix for
                                         //subj ran int and slope
```

```
corr_matrix[3] Omega_w; //corr matrix for
                        //item ran int and slope
real y[N]; // post. pred.
}
transformed parameters{
cov_matrix[3] Sigma_u; // varcov matrix for subj ranefs
cov_matrix[3] Sigma_w; // varcov matrix for item ranefs
for(r in 1:3){
for(c in 1:3){
Sigma_u[r,c] <- sigma_u[r] * sigma_u[c] * Omega_u[r,c];
Sigma_w[r,c] <- sigma_w[r] * sigma_w[c] * Omega_w[r,c];
}
}
}
model{
real mu[N];
for(i in 1:N){
mu[i] <-  (beta[1] + u[subj[i],1] + w[item[i],1]) +
(beta[2] + u[subj[i],2] + w[item[i],2])*c1[i] +
(beta[3] + u[subj[i],3] + w[item[i],3])*c2[i];
}
for(i in 1:I){
u[i] ~ multi_normal(zero,Sigma_u); // subj ranef
}
for(k in 1:K){
w[k] ~ multi_normal(zero,Sigma_w); // item ranef
}

rt ~ normal(mu,sigma_e);
y ~ normal(mu,sigma_e);

sigma_u ~ gamma(1.5,10E-4);
sigma_w ~ gamma(1.5,10E-4);
sigma_e ~ gamma(1.5,10E-4);
Omega_u ~ lkj_corr(%f);
Omega_w ~ lkj_corr(%f);
}
generated quantities{
real minimum;
real maximum;
minimum <- min(y);
maximum <- max(y);
```

```
             }
             ',eta,eta)

############################################################
###  GENERATE DATA
############################################################
dat <- two_by_two(n_u=35,n_w=16,seed=999)
df <- dat[[1]]
d<-df[,c(1,2,3,4,10)]

# sum coding the two factors, which each have two levels
d$c1<-ifelse(d$factor_a==1,0.5,-0.5)
d$c2<-ifelse(d$factor_b==1,0.5,-0.5)

# u_corr[[1]] is by-subj. int-factor_a raneff correlation
# u_corr[[2]] is by-subj. int-factor_b raneff correlation
# u_corr[[3]] is by-subj. factor_a-factor_b raneff correlation
# and the same goes, mutatis mutandis, for the w_corr35, which
# is by-item raneffs.
u_corr <- dat[[2]]
w_corr <- dat[[3]]
dat <- list( subj = sort(as.integer(factor(d$subj) )),
             item = sort(as.integer(factor(d$item) )),
             rt = d$rt,
             c1 = d$c1,
             c2 = d$c2,
             N = nrow(d),
             I = length( unique(d$subj) ),
             K = length( unique(d$item) ),
             zero = rep(0,3))

############################################################
###  FIT MODELS
############################################################
fit <- stan(model_code = code,
            data = dat,
            iter = 1500, warmup=1000, chains = 2)
############################################################
###  SAVE
############################################################
save(list=c('fit','d','u_corr','w_corr'),
     file=sprintf('./w_cov_eta%.2f_wcor6.Rda',eta))
```

```
}
```

References

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure in mixed-effects models: Keep it maximal. *Journal of Memory and Language*.

Bates, D., & Sarkar, D. (2007). lme4: Linear mixed-effects models using s4 classes [Computer software manual]. (R package version 0.9975-11)

Chung, Y., Gelman, A., Rabe-Hesketh, S., Liu, J., & Dorie, V. (2013). Weakly informative prior for point estimation of covariance matrices in hierarchical models. *Manuscript submitted for publication*.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (Third ed.). Chapman and Hall/CRC.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models.* Cambridge, UK: Cambridge University Press.

Gilbert, J., & Jordan, C. (2002). *Guide to mathematical methods.* Macmillan.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, *99(1)*, 122–149.

Kerns, G. (2011). *Introduction to probability and statistics using r.*

Kliegl, R., Masson, M. E., & Richter, E. M. (2010). A linear mixed model analysis of masked repetition priming. *Visual Cognition*, *18*(5), 655–681.

Kruschke, J. (2010). *Doing bayesian data analysis: A tutorial introduction with r.* Academic Press.

Lunn, D., Thomas, A., Best, N., & Spiegelhalter, D. (2000). Winbugs-a bayesian modelling framework: concepts, structure, and extensibility. *Statistics and computing*, *10*(4), 325–337.

Lynch, S. M. (2007). *Introduction to applied bayesian statistics and estimation for social scientists.* Springer.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS.* New York: Springer-Verlag.

Plummer, M. (2012). Jags version 3.3.0 manual. *International Agency for Research on Cancer. Lyon, France.*

Searle, S. R., Casella, G., & McCulloch, C. E. (2009). *Variance components* (Vol. 391). John Wiley & Sons.

Stan Development Team. (2013). *Stan: A C++ library for probability and sampling, version 2.1.* Available from `http://mc-stan.org/`