Bayesian linear mixed models using Stan: A tutorial for psychologists, linguists, and cognitive scientists

Tanner Sorensen

Signal Analysis and Interpretation Laboratory, University of Southern California, Los Angeles, CA, USA

Sven Hohenstein

Department of Psychology, University of Potsdam, Potsdam, Germany

Shravan Vasishth

Department of Linguistics, University of Potsdam, Potsdam, Germany, and CEREMADE, Université Paris-Dauphine, Paris, France

July 26, 2016

Abstract

With the arrival of the R packages `nlme` and `lme4`, linear mixed models (LMMs) have come to be widely used in experimentally-driven areas like psychology, linguistics, and cognitive science. This tutorial provides a practical introduction to fitting LMMs in a Bayesian framework using the probabilistic programming language Stan. We choose Stan (rather than WinBUGS or JAGS) because it provides an elegant and scalable framework for fitting models in most of the standard applications of LMMs. We ease the reader into fitting increasingly complex LMMs, using a two-condition repeated measures self-paced reading study.

Keywords: Bayesian data analysis, linear mixed models, Stan

Introduction

Linear mixed models, or hierarchical/multilevel linear models, have become the main workhorse of experimental research in psychology, linguistics, and cognitive science, where repeated measures designs are the norm. Within the programming environment R (R Development Core Team, 2006), the `nlme` package (Pinheiro & Bates, 2000) and its successor, `lme4` (Bates, Mächler, Bolker, & Walker, 2015) have revolutionized the use of linear mixed models (LMMs) due to their simplicity and speed: one can fit fairly complicated models relatively quickly, often with a single line of code. A great advantage of LMMs over traditional approaches such as repeated measures ANOVA and paired t-tests is that there is no need to aggregate over subjects and items to compute two sets of F-scores (or several t-scores) separately; a single model can take all sources of variance into account simultaneously. Furthermore, comparisons between conditions can easily be implemented in a single model through appropriate contrast coding.

Other important developments related to LMMs have been unfolding in computational statistics. Specifically, probabilistic programming languages like WinBUGS (Lunn, Thomas, Best, & Spiegelhalter, 2000), JAGS (Plummer, 2012) and Stan (Stan Development Team, 2014), among others, have made it possible to fit Bayesian LMMs quite easily. However, one prerequisite for using these programming languages is that some background statistical knowledge is needed before one can define the model. This difficulty is well-known; for example, Spiegelhalter, Abrams, and Myles (2004, p. 4) write: "Bayesian statistics has a (largely deserved) reputation for being mathematically challenging and difficult to put into practice...".

The purpose of this paper is to facilitate a first encounter with model specification in

Please send correspondence to tsorense@usc.edu and {sven.hohenstein,vasishth}@uni-potsdam.de.
The authors' orcid IDs are: 0000-0002-3111-9974 (Sorensen), 0000-0002-9708-1593 (Hohenstein), and 0000-0003-2027-1994 (Vasishth).
All authors have read and approved the final manuscript, and they have no conflicts of interest with respect to their authorship or the publication of this article. Furthermore, the authors did not benefit from funding.

one of these programming languages, Stan. The tutorial is aimed primarily at psychologists, linguists, and cognitive scientists who have used `lme4` to fit models to their data, but who may have only a basic knowledge of the underlying LMM machinery. By "basic knowledge" we mean that they may not be able to answer some or all of these questions: what is a design matrix; what is contrast coding; what is a random effects variance-covariance matrix in a linear mixed model; what is the Cholesky decomposition? Our tutorial is not intended for statisticians or psychology researchers who could, for example, write their own Markov Chain Monte Carlo (MCMC) samplers in R or C++ or the like; for them, the Stan manual is the optimal starting point. The present tutorial attempts to ease the beginner into their first steps towards fitting Bayesian linear mixed models. More detailed presentations about linear mixed models are available in several textbooks; references are provided at the end of this tutorial. For the complete newcomer to statistical methods, the articles by Vasishth and Nicenboim (2016) and Nicenboim and Vasishth (2016) should be read first, as they provide a grounds-up preparation for the present article.

We have chosen Stan as the programming language of choice (over JAGS and WinBUGS) because it is possible to fit arbitrarily complex models with Stan. For example, it is possible (if time consuming) to fit a model with 14 fixed effects predictors and two crossed random effects by subject and item, each involving a 14×14 variance-covariance matrix (Bates, Kliegl, Vasishth, & Baayen, 2015); as far as we are aware, such models cannot be fit in JAGS or WinBUGS.[1]

In this tutorial, we take it as a given that the reader is interested in learning how to fit Bayesian linear mixed models. The tutorial is structured as follows. After a short introduction to Bayesian modeling, we begin by successively building up increasingly complex LMMs using the data-set reported by Gibson and Wu (2013), which has a simple two-condition design. At each step, we explain the structure of the model. The next section takes up inference for this two-condition design.

This paper was written using a literate programming tool, `knitr` (Xie, 2015); this

[1] Whether it makes sense in general to fit such a complex model is a different issue; see Gelman et al. (2014), and Bates, Kliegl, et al. (2015) for recent discussion.

integrates documentation for the accompanying code with the paper. The `knitr` file that generated this paper, as well as all the code and data used in this tutorial, can be downloaded from our website:

`https://www.ling.uni-potsdam.de/~vasishth/statistics/BayesLMMs.html`

In addition, the source code for the paper, all R code, and data are available on github at:

`https://github.com/vasishth/BayesLMMTutorial`

We start with the two-condition repeated measures data-set (Gibson & Wu, 2013) as a concrete running example. This simple example serves as a starter kit for fitting commonly used LMMs in the Bayesian setting. We assume that the reader has the relevant software installed; specifically, the RStan interface to Stan in R. For detailed instructions, see

`https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started`

Bayesian statistics

Bayesian modeling has two major advantages over frequentist analysis with linear mixed models. First, information based on pre-existing knowledge can be incorporated into the analysis using different priors. Second, complex models with a large number of random variance components can be fit. In the following, we will provide a short introduction to Bayesian statistics which highlights these two advantages of the Bayesian approach to data analysis.

The first advantage of the Bayesian approach is a consequence of Bayes' Theorem, the fundamental rule of Bayesian statistics. It can be seen as a way of understanding how the probability that a hypothesis is true is affected by new data. In mathematical notation, Bayes' Theorem states

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)},$$

where H is the hypothesis we are interested in and D represents new data. Since D is fixed for a given data-set, the theorem can be rephrased as

$$P(H \mid D) \propto P(D \mid H)P(H).$$

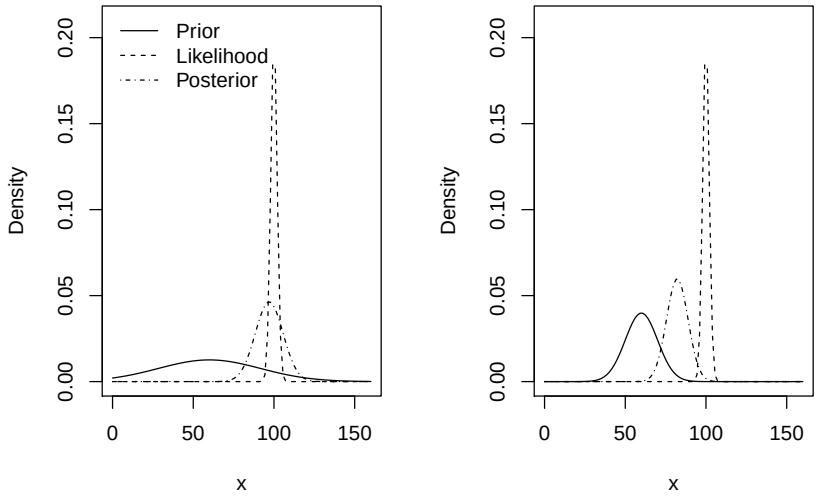

Figure 1. Prior, likelihood, and posterior normal distributions. The likelihood is based on $n = 20$ observations with sample mean $\mu = 100$ and standard deviation $\sigma = 40$. The prior (identical in both panels) has mean $\mu_0 = 60$ and variance $\sigma_0^2 = 1000$ (left-hand panel) or $\sigma_0^2 = 100$ (right-hand panel), respectively.

The *posterior* probability that the hypothesis is true given new data, $P(H \mid D)$, is proportional to the product of the *likelihood* of the new data given the hypothesis, $P(D \mid H)$, and the *prior* probability of the hypothesis, $P(H)$.

For the purposes of this paper, the goal of a Bayesian analysis is simply to derive the posterior distribution of each parameter of interest, given some data and prior knowledge about the distributions of the parameters. The following example illustrates how the posterior depends on the likelihood and prior. Before collecting data, a researcher has some hypothesis concerning the distribution of the response variable X in an experiment. The reseacher expresses his or her belief in a prior distribution, say, a normal distribution with a mean value of $\mu = 60$ and variance $\sigma^2 = 1000$ (solid density in left-hand panel of Figure 1). The large variance reflects the researcher's uncertainty concerning the true mean of the distribution. Alternatively, if the researcher were very certain that $\mu = 60$, then he or she might choose the much lower variance $\sigma^2 = 100$ (solid density in right-hand panel of the right-hand panel of Figure 1).

The researcher starts to collect data. In our example, there are $n = 20$ values with a sample mean $\bar{x} = 100$ and sample standard deviation $s = 40$. The corresponding likelihood distribution is displayed in Figure 1 (dashed line). The resulting posterior distribution (dash-dot line) combines the prior and likelihood. Given the prior with the larger variance (left-hand panel), the posterior is largely influenced by the data. Given the prior with the smaller variance (right-hand panel), its influence on the posterior is much stronger, resulting in a smaller shift towards the data mean.

This toy example illustrates the central idea of Bayesian modeling. The prior reflects our knowledge of past results. In most cases, we will use so-called vague flat priors such that the posterior distribution is mainly affected by the data. The resulting posterior distribution allows for making inferences about model parameters.

The second advantage of Bayesian modeling concerns variance components (random effects). Fitting a large number of random effects in non-Bayesian settings requires a large amount of data. Often, the data-set is too small to reliably estimate variance component parameters (Bates, Kliegl, et al., 2015; Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2016). However, if a researcher is interested in differences between individual subjects or items (random intercepts and random slopes) or relationships between differences (correlations between variance components), Bayesian modeling can be used even if there is not enough data for inferential statistics. The resulting posterior distributions might have high variance but they still allow for calculating probabilities of true parameter values of variance components. Note that we do not intend to criticize classical LMMs, but rather to highlight the possibilities of Bayesian modeling concerning random effects. For further explanation of the advantages this approach affords beyond the classical frequentist approach, the reader is directed to the rich literature relating to a comparison between Bayesian versus frequentist statistics (such as the provocatively titled paper by Lavine, 1999, and the highly accessible textbooks by McElreath, 2016 and Kruschke, 2014).

Example: A two-condition repeated measures design

This section motivates the LMM with the self-paced reading data-set of Gibson and Wu (2013). We introduce the data-set, state our modeling goals here, and proceed to build up increasingly complex LMMs, starting with a fixed effects linear model before adding varying intercepts, adding varying slopes, and finally modeling the correlation between the varying intercepts and slopes (the "maximal model" of Barr, Levy, Scheepers, & Tily, 2013). We explain these new model parameters as we introduce them. Models of varying complexity such as these three can be generalized as described in Appendix B. The result of our modeling is a probability model that expresses how the dependent variable, the reading time labeled `rt`, was generated in the experiment of Gibson and Wu (2013). The model allows us to derive the *posterior probability distribution* of the model parameters from a *prior probability distribution* and a *likelihood function*. Stan makes it easy to compute this posterior distribution for each model parameter of interest. The resulting posterior distribution reflects what we should believe about the value of that parameter, given the experimental data.

The scientific question. Subject and object relative clauses have been widely used in reading studies to investigate sentence comprehension processes. A subject relative is a sentence like *The senator who interrogated the journalist resigned* where a noun (*senator*) is modified by a relative clause (*who interrogated the journalist*), and the modified noun is the grammatical subject of the relative clause. In an object relative, the noun modified by the relative clause is the grammatical object of the relative clause (e.g., *The senator who the journalist interrogated resigned*). In both cases, the noun that is modified (*senator*) is called the head noun.

A typical finding for English is that subject relatives are easier to process than object relatives (Just & Carpenter, 1992). Natural languages generally have relative clauses, and the subject relative advantage has until recently been considered to be true cross-linguistically. However, Chinese relative clauses apparently represent an interesting counter-example to this generalization; recent work by Hsiao and Gibson (2003) has suggested that

in Chinese, *object* relatives are easier to process than subject relatives at a particular point in the sentence (the head noun of the relative clause). We now present an analysis of a subsequently published data-set (Gibson & Wu, 2013) that evaluates this claim.

The data. The dependent variable of the experiment of Gibson and Wu (2013) was the reading time `rt` in milliseconds of the head noun of the relative clause. This was recorded in two conditions (subject relative and object relative), with 37 subjects and 15 items, presented in a standard Latin square design. There were originally 16 items, but one item was removed, resulting in $37 \times 15 = 555$ data points. However, eight data points from one subject (id 27) were missing. As a consequence, we have a total of $555 - 8 = 547$ data points. The first few lines from the data frame are shown in Table 1; "o" refers to object relative and "s" to subject relative.

row	subj	item	so	rt
1	1	13	o	1561
2	1	6	s	959
3	1	5	o	582
4	1	9	o	294
5	1	14	s	438
6	1	4	s	286
⋮	⋮	⋮	⋮	
547	9	11	o	350

Table 1
First six rows, and the last row, of the data-set of Gibson and Wu (2013), as they appear in the data frame.

Fixed Effects Model

We begin by making the working assumption that the dependent variable of reading time `rt` on the head noun is approximately log-normally distributed (Rouder, 2005). This assumes that the logarithm of `rt` is approximately normally distributed. The logarithm of the reading times, $\log \texttt{rt}$, has some unknown grand mean β_0. The mean of the log-normal distribution of `rt` is the sum of β_0 and an adjustment $\beta_1 \texttt{so}$ whose magnitude depends on the categorical predictor `so`, which has the value -1 when `rt` is from the subject relative

condition, and 1 when `rt` is from the object relative condition. One way to write the model in terms of the logarithm of the reading times is as follows:

$$\log \texttt{rt}_i = \beta_0 + \beta_1 \texttt{so}_i + \varepsilon_i \tag{1}$$

This is a *fixed effects model*. The index i represents the i-th row in the data-frame (in this case, $i \in \{1, \ldots, 547\}$); the term ε_i represents the error in the i-th row. With the above ± 1 contrast coding, β_0 represents the grand mean of $\log \texttt{rt}$, regardless of relative clause type. It can be estimated by simply taking the grand mean of $\log \texttt{rt}$. The parameter β_1 is an adjustment to β_0 so that the mean of $\log \texttt{rt}$ is $\beta_0 + 1\beta_1$ when $\log \texttt{rt}$ is from the object relative condition, and $\beta_0 - 1\beta_1$ when $\log \texttt{rt}$ is from the subject relative condition. Notice that $2\beta_1$ will be the difference in the means between the object and subject relative clause conditions. Together, β_0 and β_1 make up the part of the model which characterizes the effect of the experimental manipulation, relative clause type (`so`), on the dependent variable `rt`. We call this a fixed effects model because we estimate the parameters β_0 and β_1, which do not vary from subject to subject or from item to item. In R, this would correspond to fitting a simple linear model using the `lm` function, with `so` as predictor and $\log \texttt{rt}$ as dependent variable.

The error ε_i is positive when $\log \texttt{rt}_i$ is greater than the expected value $\mu_i = \beta_0 + \beta_1 \texttt{so}_i$ and negative when $\log \texttt{rt}_i$ is less than the expected value μ_i. Thus, the error is the amount by which the expected value differs from actually observed value. We assume that the ε_i are independently and identically distributed as a normal distribution with mean zero and unknown standard deviation σ_e. Stan parameterizes the normal distribution by the mean and standard deviation, and we follow that convention here by writing the distribution of ε as $\mathcal{N}(0, \sigma_e)$. (This is different from the standard notation in statistics, where the normal distribution is defined in terms of mean and variance.) A consequence of the assumption that the errors are identically distributed is that the distribution of ε should, at least approximately, have the same shape as the normal distribution. Independence implies that

```
1  # read in data:
2  rDat <- read.table("gibsonwu2012data.txt", header = TRUE)
3  # subset critical region:
4  rDat <- subset(rDat, region == "headnoun")
5
6  # convert subjects and items to factors
7  rDat$subj <- factor(rDat$subj)
8  rDat$item <- factor(rDat$item)
9  # contrast coding of type (-1 vs. 1)
10 rDat$so <- ifelse(rDat$type == "subj-ext", -1, 1)
11
12 # create data as list for Stan, and fit model:
13 stanDat <- list(rt = rDat$rt, so = rDat$so, N = nrow(rDat))
14 library(rstan)
15 fixEfFit <- stan(file = "fixEf.stan", data = stanDat,
16                  iter = 2000, chains = 4)
17
18 # plot traceplot, excluding warm-up:
19 traceplot(fixEfFit, pars = c("beta", "sigma_e"),
20           inc_warmup = FALSE)
21
22 # examine quantiles of posterior distributions:
23 print(fixEfFit, pars = c("beta", "sigma_e"),
24       probs = c(0.025, 0.5, 0.975))
25
26 # examine quantiles of parameter of interest:
27 beta1 <- unlist(extract(fixEfFit, pars = "beta[2]"))
28 print(quantile(beta1, probs = c(0.025, 0.5, 0.975)))
```

Listing 1: R code for the fixed effects model.

there should be no correlation between the errors—this is not the case in the data, since we have multiple measurements from each subject and multiple measurements from each item. This introduces correlation between errors.

Setting up the data. We now fit the fixed effects model. For the following discussion, refer to the code in Listings 1 (R code) and 2 (Stan code). First, we read the Gibson and Wu (2013) data into a data frame `rDat` in R, and then subset the critical region (Listing 1, lines 2 and 4). Next, we create a data list `stanDat` for Stan, which contains the data (line 13). Stan requires the data to be of type list; this is different from the `lm` and `lmer` functions, which assume that the data are of type data-frame.

Defining the model. The next step is to write the Stan model in a text file with extension `.stan`. A Stan model consists of several *blocks*. A block is a set of statements

```
1  data {
2    int<lower=1> N;                //number of data points
3    real rt[N];                    //reading time
4    real<lower=-1,upper=1> so[N];  //predictor
5  }
6  parameters {
7    vector[2] beta;             //intercept and slope
8    real<lower=0> sigma_e;      //error sd
9  }
10 model {
11   real mu;
12   for (i in 1:N){                    // likelihood
13     mu = beta[1] + beta[2] * so[i];
14     rt[i] ~ lognormal(mu, sigma_e);
15   }
16 }
```

Listing 2: Stan code for the fixed effects model.

surrounded by brackets and preceded by the block name. We open up a file `fixEf.stan` in a text editor and write down the first block, the *data block*, which contains the declaration of the variables in the data object `stanDat` (Listing 2, lines 1–5). The strings `real` and `int` specify the data type for each variable. A `real` variable is a real number, and an `int` variable is an integer. For instance, `N` is the integer number of data points. The variables `so` and `rt` are arrays of length `N` whose entries are `real`. We constrain a variable to take only a subset of the values allowed by its type (e.g., `int` or `real`) by specifying in brackets lower and upper bounds (e.g. `<lower=-1,upper=1>`). The variables in the data block, `N`, `rt`, and `so`, correspond to the values of the list `stanDat` in R. The list `stanDat` must match the variables of the data block in case, but the order of variable declarations in the data block does not necessarily have to match the order of values in the list `stanDat`.

Next, we turn to the *parameters block*, where the parameters are defined (Listing 2, lines 6–9). These are the model parameters, for which posterior distributions are of interest. The fixed effects model has three parameters: the fixed intercept β_0, the fixed slope β_1, and the standard deviation σ_e of the error. We store the fixed effects β_0 and β_1 in a vector, which contains variables of type `real`. Although we called our parameters β_0 and β_1 in the fixed effects model, in Stan, these are contained in the vector `beta` with indices 1 and 2.

Thus, β_0 is in `beta[1]` and β_1 in `beta[2]`. The third parameter, the standard deviation σ_e of the error (`sigma_e`), is also defined here, and is constrained to have lower bound zero (Listing 2, line 8).

Finally, the *model block* specifies the prior distribution and the likelihood (Listing 2, lines 10–16). To understand the Stan syntax, compare the Stan code above to the specification of the fixed effects model. The Stan code literally writes out this model. The block begins with a local variable declaration for `mu`, which is the mean of `rt` conditional on whether `so` is -1 for the subject relative condition or 1 for the object relative condition.

The for-loop assigns to `mu` the mean for the log-normal distribution of `rt[i]`, conditional on the value of the predictor `so[i]` for relative clause type. The statement `rt[i] ~ lognormal(mu, sigma_e)` in a for-loop means that the logarithm of each value in the vector `rt` is normally distributed with mean `mu` and standard deviation `sigma_e`.[2]

The prior distributions on the parameters `beta` and `sigma_e` would ordinarily be declared in the model block. If we don't declare any prior, it is assumed that they have a uniform prior distribution. Note that the distribution of `sigma_e` is truncated at zero because `sigma_e` is constrained to be positive (see the declaration `real<lower=0> sigma_e;` in the parameters block). This means that the error has a uniform prior with lower bound zero.[3]

Running the model. We save the file `fixEf.stan` which contains the Stan code and fit the model in R with the function `stan` from the package `rstan` (Listing 1, lines 15–16). This call to the function `stan` will compile a C++ program which produces samples from the joint posterior distribution of the fixed intercept β_0, the fixed slope β_1, and the standard deviation σ_e of the error.

The function generates four *chains* of samples. A *Markov chain* is a stochastic process,

[2] One could have equally well log-transformed the reading time and assumed a normal distribution instead of the lognormal.

[3] This is an example of an improper prior, which is not a probability distribution. Although all the improper priors used in this tutorial produce posteriors which are probability distributions, this is not true in general, and care should be taken in using improper priors (Gelman, 2006). In the present case, a Cauchy prior truncated to have a lower bound of 0 could alternatively be defined for the standard deviation. For example code using such a prior, see the KBStan vignette in the `RePsychLing` package (Baayen, Bates, Kliegl, & Vasishth, 2015).

in which random values are sequentially generated. Each sample depends on the previous one. Different chains are independent of each other such that running a Stan model with four chains is equivalent to running four (identically specified) Stan models with one chain each. For the model used here, each of the four chains contains 2000 samples of each parameter.

Samples 1 to 1000 are part of the *warmup*, where the chains settle into the posterior distribution. We analyze samples 1001 to 2000. The result is saved to an object `fixEfFit` of class `stanFit`.

The warmup samples, also known as the *burn-in* period, are intended to allow the MCMC sampling process to converge to the posterior distribution. Once a chain has converged, the samples remain quite stable.[4] Before the MCMC sampling process, the number of interations necessary for convergence is unknown. Therefore, all warmup samples are discarded. This is necessary since the initial values of the parameters might have low posterior probability and might therefore bias the result.

Besides the number of samples, we specified sampling in four different chains. Each chain is independent from the others and starts with different random initial values. Running multiple chains has two advantages over a single chain. First, the independent chains are helpful for diagnostics. If all chains have converged to the same region of the parameter space, it is more likely that they converged to the posterior distribution. Second, running multiple chains allows for parallel simulations on multiple cores.

Evaluating model convergence. The number of iterations necessary for convergence to the posterior distribution depends on the number of parameters. The probability to reach convergence increases with the number of iterations. Hence, we generally recommend using a large number of iterations although the process might converge after a smaller number of iterations. In the examples in the present paper, we use 1000 iterations for warmup and another 1000 iterations for analyzing the posterior distribution. For more complex models, more iterations might be necessary before the MCMC sampling process

[4] See, Gelman et al. (2014) for a precise discussion of convergence.

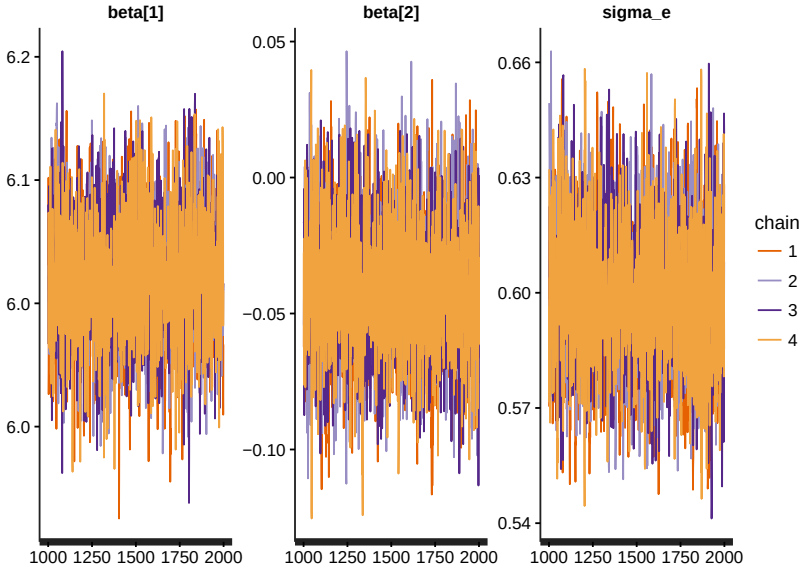

Figure 2. Trace plots of the fixed intercept β_0 (`beta[1]`), the fixed slope β_1 (`beta[2]`), and the standard deviation σ_e (`sigma_e`) of the error for the fixed effects model. Different colours denote different chains.

converges to the posterior distribution. Although there are ways to determine how long the simulation needs to be run and the number of warmup iterations given the type of posterior distribution (Raftery & Lewis, 1992), we illustrate below practical convergence diagnostics for the evaluation of convergence in the samples.

The first step after running the function `stan` should be to look at the *trace plot* of each chain after warmup, using the command shown in Listing 1, lines 13 and 14 (function `traceplot`). We choose the parameters β_i and σ_e (`pars = c("beta", "sigma_e")`) and omit the warmup samples (`inc_warmup = FALSE`). A trace plot has the chains plotted against the sample number. In Figure 2, we see three different chains plotted against sample number going from 1001 to 2000. If the trace plot looks like a "fat, hairy caterpillar" (Lunn, Jackson, Spiegelhalter, Best, & Thomas, 2012) which does not bend, this suggests that the chains have converged to the posterior distribution.

The second diagnostic which we use to assess whether the chains have converged to the posterior distribution is the statistic `Rhat`. Each parameter has the `Rhat` statistic associated with it (Gelman & Rubin, 1992); this is essentially the ratio of between-chain variance to

within-chain variance (analogous to ANOVA). The `Rhat` statistic should be approximately 1 ± 0.1 if the chain has converged. This is shown in the rightmost column of the model summary, printed in Table 2. The information can be otained with `print(fixEfFit)`, where `fixEfFit` is the object of type `stan.model` returned by the function `stan`. For example, see Listing 1, lines 23–24.

Having satisfied ourselves that the chains have converged, next we turn to examine this posterior distribution. If there is an indication that convergence has not happened, then, assuming that the model has no errors in it, increasing the number of samples usually resolves the issue.

parameter	mean	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	6.06	6.01	6.12	1
$\hat{\beta}_1$	−0.04	−0.09	0.02	1
$\hat{\sigma}_e$	0.60	0.56	0.64	1

Table 2
Credible intervals and R-hat statistic in the Gibson and Wu data.

Summarizing the result. The result of fitting the fixed effects model is the *joint posterior probability distribution* of the parameters β_0, β_1, and σ_e. The distribution is joint because each of the 4000 (4 chains $\times 1000$ post-warmup iterations) posterior samples which the call to `stan` generates is a vector $\theta = (\beta_0, \beta_1, \sigma_e)^\intercal$ of three model parameters. Thus, the object `fixEfFit` contains 4000 parameter vectors θ which occupy a three dimensional space. Already in three dimensions, the posterior distribution becomes difficult to view in one graph. Figure 3 displays the joint posterior probability distribution of the elements of θ by projecting it down onto planes. In each of the three planes (lower triangular scattergrams) we see how one parameter varies with respect to the other. In the diagonal histograms, we visualize the marginal probability distribution of each parameter separately from the other parameters.

Of immediate interest is the marginal distribution of the slope β_1. Figure 3 suggests that most of the posterior probability density of β_1 is located below zero. One quantitative way to assess the posterior probability distribution is to examine its quantiles; see Table 2.

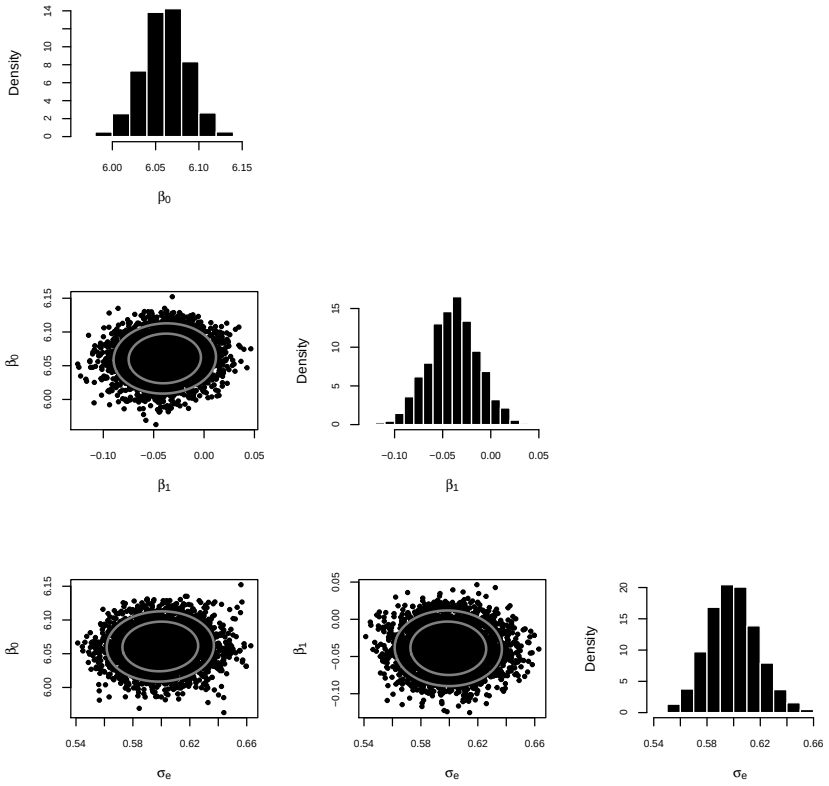

Figure 3. Samples and level curves of the bivariate joint posterior probability distribution of each element of θ with each other element (lower triangular) and marginal posterior probability distribution of each element of θ separately (diagonal). All parameters are on the log scale, but note the difference in length scale between β_1 on the one hand and β_0 and σ_e on the other.

Here, it is useful to define the concept of the *credible interval*. The $(1-\alpha)\%$ credible interval contains $(1-\alpha)\%$ of the posterior probability density. Unlike the $(1-\alpha)\%$ confidence interval from the frequentist setting, the $(1-\alpha)\%$ credible interval represents the range within which we are $(1-\alpha)\%$ certain that the true value of the parameter lies, given the prior and the data (see Morey, Hoekstra, Rouder, Lee, & Wagenmakers, 2015 for further discussion on confidence intervals vs credible intervals). A common convention is to use the interval ranging from the 2.5th to 97.5th percentiles. We follow this convention to obtain 95% credible intervals in Table 2. Lines 27–28 of Listing 1 illustrate how these quantiles of the posterior distribution of β_1 (`beta[2]`) can be computed.

The sample distribution of β_1 indicates that approximately 94% of the posterior probability density is below zero, suggesting that there is some evidence that object relatives are easier to process than subject relatives in Chinese, given the Gibson and Wu data. However, since the 95% credible interval includes zero, we may be reluctant to draw this conclusion. We will say more about the evaluation of research hypotheses further on, but it is important to note here that the fixed effects model presented above is in any case not appropriate for the present data. The independence assumption is violated for the errors because we have repeated measures from each subject and from each item. Linear mixed models extend the linear model to solve precisely this problem.

Varying Intercepts Mixed Effects Model

The fixed effects model is inappropriate for the Gibson and Wu data because it does not take into account the fact that we have multiple measurements for each subject and item. As mentioned above, these multiple measurements lead to a violation of the independence of errors assumption. Moreover, the fixed effects coefficients β_0 and β_1 represent means over all subjects and items, ignoring the fact that some subjects will be faster and some slower than average; similarly, some items will be read faster than average, and some slower.

In linear mixed models, we take this by-subject and by-item variability into account by adding adjustment terms u_{0j} and w_{0k}, which adjust β_0 for subject j and item k. This partially decomposes ε_i into a sum of the terms u_{0j} and w_{0k}, which are adjustments to the intercept β_0 for the subject j and item k associated with $\texttt{rt}_i$. If subject j is slower than the average of all the subjects, 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. Each subject j has their own adjustment u_{0j}, and each item its own w_{0k}. These adjustments u_{0j} and w_{0k} are called *random intercepts* by Pinheiro and Bates (2000) and *varying intercepts* by Gelman and Hill (2007), and by adjusting β_0 by these we account for the variability by speaker and by item.

We assume that these adjustments are normally distributed around zero with un-

known standard deviation: $u_0 \sim \mathcal{N}(0, \sigma_u)$ and $w_0 \sim \mathcal{N}(0, \sigma_w)$. We now have three sources of variance in this model: the standard deviation of the errors σ_e, the standard deviation of the by-subject random intercepts σ_u, and the standard deviation of the by-item varying intercepts σ_w. We will refer to these as *variance components*.

We now express the logarithm of reading time, which was produced by subjects $j \in \{1, \ldots, 37\}$ reading items $k \in \{1, \ldots, 15\}$, in conditions $i \in \{1, 2\}$ (1 refers to subject relatives, 2 to object relatives), as the following sum. Notice that we are now using a slightly different way to describe the model, compared to the fixed effects model. We are using indices for subject, item, and condition to identify unique rows. Also, instead of writing $\beta_1 \texttt{so}_i$, we index β_1 by the condition i. This follows the notation used in the textbook on linear mixed models, written by the authors of `nlme` (Pinheiro & Bates, 2000), the precursor to `lme4`.

$$\log \texttt{rt}_{ijk} = \beta_0 + \underbrace{\beta_{1i}}_{\beta_1 \texttt{so}_i} + u_{0j} + w_{0k} + \varepsilon_{ijk} \tag{2}$$

This is an LMM, and more specifically a *varying intercepts model*. The coefficient β_{1i} is the one of primary interest; it will have some mean value $-\beta_1$ for subject relatives and β_1 for object relatives due to the contrast coding. So, if our posterior mean for β_1 is negative, this would suggest that object relatives are read faster than subject relatives.

We fit the varying intercepts model in Stan in much the same way as the fixed effects model. For the following discussion, consult Listing 3 for the R code used to run the model, and Listing 4 for the Stan code.

Setting up the data. The data which we prepare for passing on to the function `stan` now includes subject and item information (Listing 3, lines 2–8). The data block in the Stan code accordingly includes the number `J`, `K` of subjects and items, respectively, as well as subject and item identifiers `subj` and `item` (Listing 4, lines 5–8).

Defining the model. The random intercepts model, shown in Listing 4, still has the fixed intercept β_0, the fixed slope β_1, and the standard deviation σ_e of the error, and

```
# format data for Stan:
stanDat <- list(subj = as.integer(rDat$subj),
                item = as.integer(rDat$item),
                rt = rDat$rt,
                so = rDat$so,
                N = nrow(rDat),
                J = nlevels(rDat$subj),
                K = nlevels(rDat$item))

# Sample from posterior distribution:
ranIntFit <- stan(file = "ranInt.stan", data = stanDat,
                  iter = 2000, chains = 4)
# Summarize results:
print(ranIntFit, pars = c("beta", "sigma_e", "sigma_u", "sigma_w"),
      probs = c(0.025, 0.5, 0.975))

beta1 <- unlist(extract(ranIntFit, pars = "beta[2]"))
print(quantile(beta1, probs = c(0.025, 0.5, 0.975)))

# Posterior probability of beta1 being less than 0:
mean(beta1 < 0)
```

Listing 3: R code for running the random intercepts model, the varying intercepts model. Note that lines 1–10 and 14 of Listing 1 must be run first.

we specify these in the same way as we did for the fixed effects model. In addition, the varying intercepts model has by-subject varying intercepts u_{0j} for $j \in \{1, \ldots, J\}$ and by-item varying intercepts w_{0k} for $k \in \{1, \ldots, K\}$. The standard deviation of u_0 is σ_u and the standard deviation of w_0 is σ_w. We again constrain the standard deviations to be positive.

The model block places normal distribution priors on the varying intercepts u_0 and w_0. We implicitly place uniform priors on `sigma_u`, `sigma_w`, and `sigma_e` by omitting them from the model block. As pointed out earlier for `sigma_e`, these prior distributions have lower bound zero because of the constraint `<lower=0>` in the variable declarations.

The statement about how each row in the data is generated is shown in Listing 4, lines 26–29; here, both the fixed effects and the varying intercepts for subjects and items determine the expected value `mu`. The vector `u` has varying intercepts for subjects. Likewise, the vector `w` has varying intercepts for items. The for-loop in lines 26–29 now adds `u[subj[i]] + w[item[i]]` to the mean `beta[1]` of the distribution of `rt[i]`. These are subject- and item-specific adjustments to the fixed-effects intercept `beta[1]`. The term `u[subj[i]]` is

```
data {
  int<lower=1> N;                   //number of data points
  real rt[N];                       //reading time
  real<lower=-1, upper=1> so[N];    //predictor
  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
}

parameters {
  vector[2] beta;            //fixed intercept and slope
  vector[J] u;               //subject intercepts
  vector[K] w;               //item intercepts
  real<lower=0> sigma_e;     //error sd
  real<lower=0> sigma_u;     //subj sd
  real<lower=0> sigma_w;     //item sd
}

model {
  real mu;
  //priors
  u ~ normal(0, sigma_u);    //subj random effects
  w ~ normal(0, sigma_w);    //item random effects
  // likelihood
  for (i in 1:N){
    mu = beta[1] + u[subj[i]] + w[item[i]] + beta[2] * so[i];
    rt[i] ~ lognormal(mu, sigma_e);
  }
}
```

Listing 4: Stan code for running the random intercepts model, the varying intercepts model.

the identifier of the subject for row i in the data-frame; thus, if $i = 1$, then `subj[1] = 1`, and `item[1] = 13` (see Table 1).

Running the model. In R, we pass the list `stanDat` of data to `stan`, which compiles a C++ program to sample from the posterior distribution of the random intercepts model. Stan samples from the posterior distribution of the model parameters, including the varying intercepts u_{0j} and w_{0k} for each subject $j \in \{1, \ldots, J\}$ and item $k \in \{1, \ldots, K\}$.

It may be helpful to rewrite the model in mathematical form following the Stan syntax (Gelman & Hill, 2007 use a similar notation); the Stan statements are slightly different from the way that we expressed the random intercepts model. Defining i as the row number in the data frame, i.e., $i \in \{1, \ldots, 547\}$, we can write:

$$
\begin{aligned}
&\text{Likelihood :}\\
&\mu_i = \beta_0 + u_{[subj[i]]} + w_{[item[i]]} + \beta_1 \cdot \texttt{so}_i\\
&\texttt{rt}_i \sim \text{LogNormal}(\mu_i, \sigma_e)\\
&\text{Priors :}\\
&u \sim \text{Normal}(0, \sigma_u) \quad w \sim \text{Normal}(0, \sigma_w)\\
&\sigma_e, \sigma_u, \sigma_w \sim \text{Uniform}(0, \infty)\\
&\beta \sim \text{Uniform}(-\infty, \infty)
\end{aligned}
\tag{3}
$$

Here, notice that the i-th row in the statement for μ identifies the subject identifier (j) ranging from 1 to 37, and the item identifier (k) ranging from 1 to 15.

Summarizing the results. The posterior distributions of each of the parameters is summarized in Table 3. The $\hat{R}$ values suggest that model has converged because they equal one. Note also that compared to Model 1, the estimate of σ_e is smaller; this is because the other two variance components are now being estimated as well. Note that the 95% credible interval for the estimate $\hat{\beta}_1$ includes zero; thus, there is some evidence that object relatives are easier than subject relatives, but we cannot exclude the possibility that there is no difference in the reading times between the two relative clause types.

parameter	mean	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	6.06	5.92	6.20	1
$\hat{\beta}_1$	−0.04	−0.08	0.01	1
$\hat{\sigma}_e$	0.52	0.49	0.56	1
$\hat{\sigma}_u$	0.26	0.19	0.34	1
$\hat{\sigma}_w$	0.20	0.12	0.33	1

Table 3
The quantiles and the $\hat{R}$ statistic in the Gibson and Wu data, the varying intercepts model.

Varying Intercepts, Varying Slopes Mixed Effects Model

The varying intercepts model accounted for having multiple measurements from each subject and item by introducing random intercepts by subject and by item. This reflects that some subjects will be faster and some slower than average, and that some items will be read faster than average, and some slower. Consider now that not only does reading speed differ by subject and by item, but also the slowdown in the object relative condition may differ in magnitude by subject and item. This amounts to a different effect size for `so` by subject and item. Although such individual-level variability was not of interest in the original paper by Gibson and Wu, it could be of theoretical interest (see, for example, Kliegl, Wei, Dambacher, Yan, & Zhou, 2010). Furthermore, as Barr et al. (2013) point out, it is in principle desirable to include a fixed effect factor in the random effects as a varying slope if the experiment design is such that subjects see both levels of the factor (cf. Baayen, Vasishth, Bates, & Kliegl, 2016; Bates, Kliegl, et al., 2015; Matuschek et al., 2016).

Adding varying slopes. In order to express this structure in the LMM, we must introduce varying slopes. The first change is to let the size of the effect for `so` vary by subject and by item. The goal here is to express that some subjects exhibit greater slowdowns in the object relative condition than others. We let effect size vary by subject and by item by including in the model by-subject and by-item varying slopes which adjust the fixed slope β_1 in the same way that the by-subject and by-item varying intercepts adjust the fixed intercept β_0. This adjustment of the slope by subject and by item is expressed by adjusting β_1 by adding two terms u_{1j} and w_{1k}. These are *random* or *varying slopes*, and by adding them we account for how the effect of relative clause type varies by subject j and by item k. We now express the logarithm of reading time, which was produced by subject j reading item k, as the following sum. The subscript i indexes the conditions.

$$\log \mathtt{rt}_{ijk} = \underbrace{\beta_0 + u_{0j} + w_{0k}}_{\text{varying intercepts}} + \underbrace{\beta_{1i} + u_{1ij} + w_{1ik}}_{\text{varying slopes}} + \varepsilon_{ijk} \tag{4}$$

This is a *varying intercepts, varying slopes model.*

```
# 1. Compile and fit model
ranIntSlpNoCorFit <- stan(file="ranIntSlpNoCor.stan", data = stanDat,
                      iter = 2000, chains = 4)

# posterior probability of beta 1 being less
# than 0:
beta1 <- unlist(extract(ranIntSlpNoCorFit, pars = "beta[2]"))
print(quantile(beta1, probs = c(0.025, 0.5, 0.975)))
mean(beta1 < 0)
```

Listing 5: R code for running the varying intercepts, varying slopes model. Note that lines 1-10 and 14 of Listing 1 and lines 2–8 of Listing 3 must be run first.

Setting up the data. Listing 5 contains the R code for fitting the varying intercepts, varying slopes model. The data which we pass to the function `stan` is the same as for the varying intercepts model. This contains subject and item information (Listing 3, lines 2–8).

Defining the model. Listing 6 contains the Stan code for the varying intercepts, varying slopes model. The data block is the same as in the varying intercepts model, but the parameters block contains several new parameters. This time we have the vector `sigma_u`, which contains the standard deviations $(\sigma_{u0}, \sigma_{u1})^\intercal$ of the by-subject random intercepts and slopes. The by-subject random intercepts are in the first row of the $2\times$`J` matrix `u`, and the by-subject random slopes are in the second row of `u`. Similarly, the vector `sigma_w` contains the standard deviations $(\sigma_{w0}, \sigma_{w1})^\intercal$ of the by-item random intercepts and slopes. The by-item random intercepts are in the first row of the $2\times$`K` matrix `w`, and the by-item random slopes are in the second row of `w`.

In the model block, we place priors on the parameters declared in the parameters block (Listing 6, lines 23–26), and define how these parameters generate $\log$`rt` (Listing 6, lines 28–32). The statement `u[1] ~ normal(0,sigma_u[1]);` specifies a normal prior for the by-subject random intercepts in the first row of `u`, and the statement `u[2] ~ normal(0,sigma_u[2]);` does the same for the by-subject random slopes in the second row of `u`. The same goes for the by-item random intercepts and slopes. Thus, there is a

```
data {
  int<lower=1> N;                   //number of data points
  real rt[N];                       //reading time
  real<lower=-1,upper=1> so[N];     //predictor
  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
}

parameters {
  vector[2] beta;                   //intercept and slope
  real<lower=0> sigma_e;            //error sd
  matrix[2,J] u;                    //subj intercepts, slopes
  vector<lower=0>[2] sigma_u;       //subj sd
  matrix[2,K] w;                    //item intercepts, slopes
  vector<lower=0>[2] sigma_w;       //item sd
}

model {
  real mu;
  //priors
  u[1] ~ normal(0,sigma_u[1]);       //subj intercepts
  u[2] ~ normal(0,sigma_u[2]);       //subj slopes
  w[1] ~ normal(0,sigma_w[1]);       //item intercepts
  w[2] ~ normal(0,sigma_w[2]);       //item slopes
  //likelihood
  for (i in 1:N){
    mu = beta[1] + u[1,subj[i]] + w[1,item[i]]
          + (beta[2] + u[2,subj[i]] + w[2,item[i]])*so[i];
    rt[i] ~ lognormal(mu,sigma_e);
  }
}
```

Listing 6: Stan code for the varying intercepts, varying slopes model.

prior normal distribution for each of the random effects. These distributions are centered on zero and have different standard deviations.

Running the model. We can now fit the varying intercepts, varying slopes model in R (see Listing 5). We see in the model summary of Table 4, obtained as before using `print(ranIntSlpNoCorFit)`, that the model has converged, and that the credible interval of the parameter of interest, β_1, still includes zero. In fact, the posterior probability of the parameter being less than zero is now 90%.

parameter	mean	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	6.06	5.92	6.20	1
$\hat{\beta}_1$	−0.04	−0.09	0.02	1
$\hat{\sigma}_e$	0.52	0.48	0.55	1
$\hat{\sigma}_{u0}$	0.25	0.18	0.34	1
$\hat{\sigma}_{u1}$	0.06	0.01	0.13	1
$\hat{\sigma}_{w0}$	0.20	0.12	0.32	1
$\hat{\sigma}_{w1}$	0.04	0.01	0.11	1

Table 4
The quantiles and the $\hat{R}$ statistic in the Gibson and Wu data, the varying intercepts, varying slopes model.

Correlated Varying Intercepts, Varying Slopes Mixed Effects Model

Consider now that subjects who are faster than average (i.e., who have a negative varying intercept) may exhibit greater slowdowns when they read object relatives compared to subject relatives. Similarly, it is in principle possible that items which are read faster (i.e., which have a large negative varying intercept) may show a greater slowdown in the object relative condition than in the subject relative condition. The opposite situation could also hold: faster subjects may show smaller SR-OR effects, or items read faster may show smaller SR-OR effects. This suggests the possibility of correlations between random intercepts and random slopes.

In order to express this structure in the LMM, we must model correlation between the varying intercepts and varying slopes. The model equation, repeated below, is the same as before.

$$\log \mathtt{rt}_{ijk} = \underbrace{\beta_0 + u_{0j} + w_{0k}}_{\text{varying intercepts}} + \underbrace{\beta_1 + u_{1ij} + w_{1ik}}_{\text{varying slopes}} + \varepsilon_{ijk}$$

Introducing correlation between the varying intercepts and varying slopes makes this a *correlated varying intercepts, varying slopes model.*

Defining a variance-covariance matrix for the random effects. Modeling the correlation between varying intercepts and slopes means defining a covariance relationship between by-subject varying intercepts and slopes, and between by-items varying

intercepts and slopes. This amounts to adding an assumption that the by-subject slopes u_1 could in principle have some correlation with the by-subject intercepts u_0; and by-item slopes w_1 with by-item intercept w_0. We explain this in detail below.

Let us assume that the adjustments u_0 and u_1 are normally distributed with mean zero and some variances σ^2_{u0} and σ^2_{u1}, respectively; also assume that u_0 and u_1 have correlation ρ_u. It is standard to express this situation by defining a variance-covariance matrix Σ_u, sometimes called simply a variance matrix. This matrix has the variances of u_0 and u_1 respectively along the diagonal, and the covariances on the off-diagonal. The covariance $\text{Cov}(X, Y)$ between two variables X and Y is defined as the product of their correlation ρ and their standard deviations σ_X and σ_Y: $\text{Cov}(X, Y) = \rho\sigma_X\sigma_Y$.

$$\Sigma_u = \begin{pmatrix} \sigma^2_{u0} & \rho_u\sigma_{u0}\sigma_{u1} \\ \rho_u\sigma_{u0}\sigma_{u1} & \sigma^2_{u1} \end{pmatrix} \tag{5}$$

Similarly, we can define a variance-covariance matrix Σ_w for items, using the standard deviations σ_{w0}, σ_{w1}, and the correlation ρ_w.

$$\Sigma_w = \begin{pmatrix} \sigma^2_{w0} & \rho_w\sigma_{w0}\sigma_{w1} \\ \rho_w\sigma_{w0}\sigma_{w1} & \sigma^2_{w1} \end{pmatrix} \tag{6}$$

The standard way to express this relationship between the subject intercepts u_0 and slopes u_1, and the item intercepts w_0 and slopes w_1, is to define a bivariate normal distribution as follows:

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_u\right), \quad \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_w\right) \tag{7}$$

An important point to notice here is that any $n \times n$ variance-covariance matrix has associated with it an $n \times n$ correlation matrix. In the subject variance-covariance matrix Σ_u, the correlation matrix is

$$\begin{pmatrix} 1 & \rho_{01} \\ \rho_{01} & 1 \end{pmatrix} \tag{8}$$

```
ranIntSlpFit <- stan(file = "ranIntSlp.stan", data = stanDat,
                     iter = 2000, chains = 4)

# posterior probability of beta 1 being less
# than 0:
beta1 <- unlist(extract(ranIntSlpFit, pars = "beta[2]"))
print(quantile(beta1, probs = c(0.025, 0.5, 0.975)))
mean(beta1 < 0)

## Use the L matrices the compute the correlation matrices
# L matrices
L_u <- extract(ranIntSlpFit, pars = "L_u")$L_u
L_w <- extract(ranIntSlpFit, pars = "L_w")$L_w

# correlation parameters
cor_u <- apply(L_u, 1, function(x) tcrossprod(x)[1, 2])
cor_w <- apply(L_w, 1, function(x) tcrossprod(x)[1, 2])

print(signif(quantile(cor_u, probs = c(0.025, 0.5, 0.975)), 2))
print(mean(cor_u))
print(signif(quantile(cor_w, probs = c(0.025, 0.5, 0.975)), 2))
print(mean(cor_w))
```

Listing 7: R code for running the correlated varying intercepts, varying slopes model. Note that lines 1–10 and 14 of Listing 1 and lines 2–8 of Listing 3 must be run first.

In a correlation matrix, the diagonal elements will always be 1, because a variable always has a correlation of 1 with itself. The off-diagonal entries will have the correlations between the variables. Note also that, given the variances σ^2_{u0} and σ^2_{u1}, we can always recover the variance-covariance matrix, if we know the correlation matrix. This is because of the above-mentioned definition of covariance.

A correlation matrix can be factored into a *matrix square root*. Given a correlation matrix C, we can obtain its square root matrix L. The square root of a matrix is such that we can square L to get the correlation matrix C back. In the next section, we see that the matrix square root is important for generating the random intercepts and slopes because of its role in generating correlated random variables. Appendix A describes one method for obtaining L, namely, the Cholesky factorization.

Defining the model. With this background, implementing the varying intercepts, varying slopes model is straightforward; see Listing 7 for the R code and Listing 8 for the

```
data {
  int<lower=1> N;                  //number of data points
  real rt[N];                      //reading time
  real<lower=-1, upper=1> so[N];   //predictor
  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
}

parameters {
  vector[2] beta;                  //intercept and slope
  real<lower=0> sigma_e;           //error sd
  vector<lower=0>[2] sigma_u;      //subj sd
  cholesky_factor_corr[2] L_u;
  matrix[2,J] z_u;
  vector<lower=0>[2] sigma_w;      //item sd
  cholesky_factor_corr[2] L_w;
  matrix[2,K] z_w;
}

transformed parameters{
  matrix[2,J] u;
  matrix[2,K] w;

  u = diag_pre_multiply(sigma_u, L_u) * z_u;  //subj random effects
  w = diag_pre_multiply(sigma_w, L_w) * z_w;  //item random effects
}

model {
  real mu;

  //priors
  L_u ~ lkj_corr_cholesky(2.0);
  L_w ~ lkj_corr_cholesky(2.0);
  to_vector(z_u) ~ normal(0,1);
  to_vector(z_w) ~ normal(0,1);
  //likelihood
  for (i in 1:N){
    mu = beta[1] + u[1,subj[i]] + w[1,item[i]]
          + (beta[2] + u[2,subj[i]] + w[2,item[i]]) * so[i];
    rt[i] ~ lognormal(mu, sigma_e);
  }
}
```

Listing 8: The Stan code for the correlated varying intercepts, varying slopes model.

Stan code. The R list `stanDat` is identical to the one of the varying intercepts, varying slopes model, and therefore we will focus on the Stan code. The data block is the same as before. The parameters block contains several new parameters. As before, we have vectors `sigma_u` and `sigma_w`, which are $(\sigma_{u0}, \sigma_{u1})^\intercal$ and $(\sigma_{w0}, \sigma_{w1})^\intercal$. The variables `L_u`, `L_w`, `z_u`, and `z_w`, which have been declared in the parameters block, play a role in the *transformed parameters block*, a block which we did not use in the earlier models. The transformed parameters block generates the by-subject and by-item varying intercepts and slopes using the parameters `sigma_u`, `L_u`, `z_u`, `sigma_w`, `L_w`, and `z_w`. The J pairs of by-subject varying intercepts and slopes are in the rows of the $J \times 2$ matrix `u`, and the K pairs of by-item varying intercepts and slopes are in the rows of the $K \times 2$ matrix `w`.

These varying intercepts and slopes are obtained through the statements `diag_pre_multiply(sigma_u, L_u) * z_u` and `diag_pre_multiply(sigma_w, L_w) * z_w`. This statement generates varying intercepts and slopes from the joint probability distribution of Equation 7. The parameters `L_u`, `L_w` are the matrix square roots (Cholesky factor) of the subject and item correlation matrices, respectively, and `z_u`, and `z_w` are $\mathcal{N}(0, 1)$ random variables. Appendix A has details on how this generates correlated random intercepts and slopes.

In the model block, we place priors on the parameters declared in the parameters block, and define how these parameters generate $\log$ `rt` (Listing 8, lines 30–43). The statement `L_u ~ lkj_corr_cholesky(2.0)` specifies a prior for the square root `L_u` (Cholesky factor) of the correlation matrix. This prior is best interpreted with respect to the square of `L_u`, that is, with respect to the correlation matrix. The statement `L_u ~ lkj_corr_cholesky(2.0)` implicitly places the lkj prior (so-called because it was first described by Lewandowski, Kurowicka, & Joe, 2009) with shape parameter $\nu = 2.0$ on the correlation matrices

$$\begin{pmatrix} 1 & \rho_u \\ \rho_u & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & \rho_w \\ \rho_w & 1 \end{pmatrix}, \tag{9}$$

where ρ_u is the correlation between the by-subject varying intercept σ_{u0} and slope σ_{u1}

(cf. the covariance matrix of Equation 5) and ρ_w is the correlation between the by-item varying intercept σ_{w0} and slope σ_{w1}. The lkj distribution is a probability distribution over correlation matrices. The lkj distribution has one shape parameter ν, which controls the prior correlation. If $\nu > 1$, then the probability density becomes concentrated about the 2×2 identity matrix.[5] This expresses the prior belief that the correlations are not large. If $\nu = 1$, then the probability density function is uniform over all 2×2 correlation matrices. If $0 < \nu < 1$, then the probability density has a trough at the 2×2 identity matrix. In our model, we choose $\nu = 2.0$. This choice implies that the correlations on the off-diagonal are near zero, reflecting the fact that we have no prior information about the correlation between intercepts and slopes.

The statement `to_vector(z_u) ~ normal(0,1)` places a normal distribution with mean zero and standard deviation one on `z_u`.[6] The same goes for `z_w`. The for-loop assigns to `mu` the mean of the log-normal distribution from which we draw `rt[i]`, conditional on the value of the predictor `so[i]` for relative clause type and the subject and item identity.

Running the model. We can now fit the varying intercepts, varying slopes model; see Listing 7 for the code. We see in the model summary in Table 5 that the model has converged,[7] and that the credible intervals of the parameter of interest, β_1, still includes zero. In fact, the posterior probability of the parameter being less than zero is now 90%. This information can be extracted as shown in Listing 7, lines 6–8.

Figure 4 plots the varying slope's posterior distribution against the varying intercept's posterior distribution for each subject. The correlation between u_0 and u_1 is negative, as captured by the marginal posterior distributions of the correlation ρ_u between u_0 and u_1. Thus, Figure 4 suggests that the slower a subject's reading time is on average, the slower they read object relatives. In contrast, Figure 4 shows no clear pattern for the by-item varying intercepts and slopes. The broader distribution of the correlation parameter for

[5] The lkj prior can scale up to correlation matrices larger than 2×2.

[6] The function `to_vector` means that we rearrange the matrix `z_u` as a vector in order to place the normal distribution on a vector. This makes the code run faster.

[7] We do not report the R-hat statistic for parameters ρ_u, ρ_w because these parameters converge when $\hat{R}$ equals one for each entry of the matrices L_u, L_w. This was the case.

parameter	mean	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	6.06	5.92	6.20	1
$\hat{\beta}_1$	−0.04	−0.09	0.02	1
$\hat{\sigma}_e$	0.52	0.48	0.55	1
$\hat{\sigma}_{u0}$	0.25	0.18	0.34	1
$\hat{\sigma}_{u1}$	0.07	0.01	0.13	1
$\hat{\sigma}_{w0}$	0.20	0.12	0.32	1
$\hat{\sigma}_{w1}$	0.04	0.0	0.11	1
$\hat{\rho}_u$	−0.44	−0.91	0.36	
$\hat{\rho}_w$	−0.01	−0.76	0.76	

Table 5
The quantiles and the $\hat{R}$ statistic in the Gibson and Wu data, the varying intercepts, varying slopes model.

items compared to slopes illustrates the greater uncertainty concerning the true value of the parameter. We briefly discuss inference next.

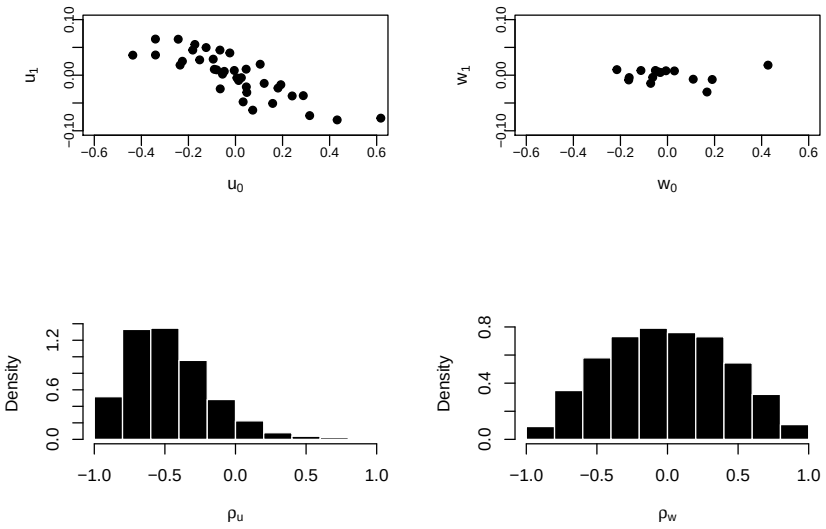

Figure 4. The top row shows the relationship between the posterior mean of the varying slopes (y-axis) and intercepts (x-axis) for each subject (left panel) and item (right panel). The bottom row shows the posterior distribution of the parameter of correlation between the varying slopes and intercepts for each subject (left panel) and item (right panel).

Random effects in a non-Bayesian LMM. We fit the same model also as a classical non-Bayesian LMM with the `lmer` function from the `lme4` package. This allows

us to compare the `lme4` results with the Stan results. Here, we focus on random effects. As illustrated in Figure 5, the estimates of the random-effect standard deviations of the classical LMM are in agreement with the modes of the posterior distributions. The `lmer` function does not show any convergence error, but the correlations between the random intercepts and slopes shows the boundary values -1 and $+1$,: the variance-covariance matrices for the subject and item random effects are degenerate. By contrast, Stan can still estimate posterior distributions for parameters in such an overly complex model (Figure 4). Of course, one may want to simplify the model for reasons of parsimony, or easier interpretability. Model selection can be carried out by evaluating predictive performance of the model, with methods such as Leave One Out (LOO) Cross-validation, or by using information criteria like the Watanabe Akaike (or Widely Available) Information Criterion (WAIC). See Nicenboim and Vasishth (2016) for discussion and example code.

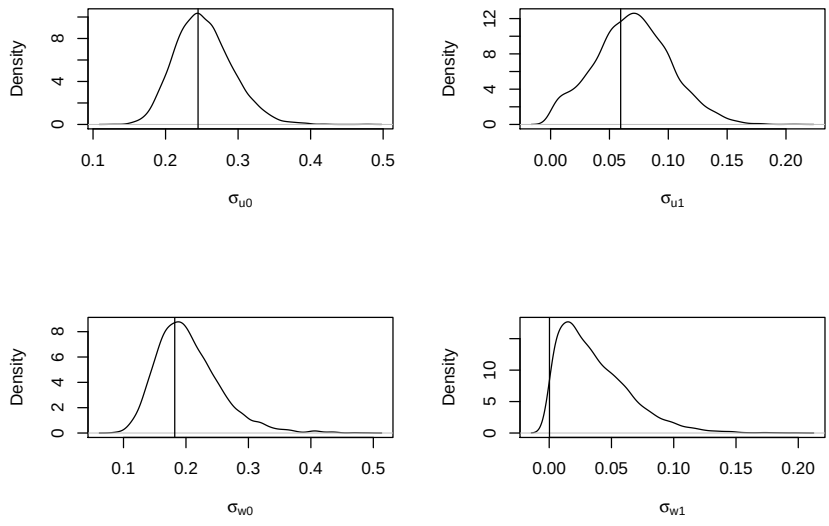

Figure 5. The curves show the density of the posterior distributions of the random-effect standard deviations. The vertical bars indicate the corresponding `lmer` estimates. The top row shows the random effects for subjects, the bottom row shows the random effects for items. Left-hand panels correspond to random intercepts, right-hand panels correspond to random slopes.

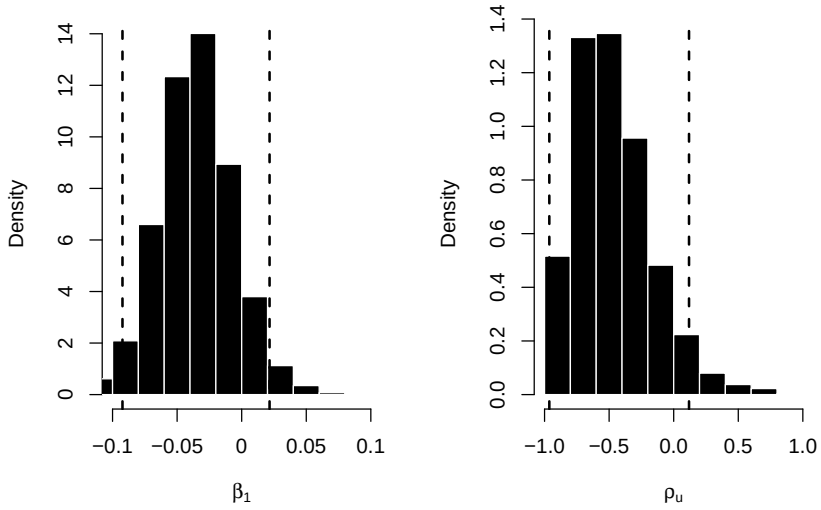

Figure 6. Upper and lower bounds on the highest posterior density credible intervals (dashed lines) plotted over the marginal posterior distribution of the fixed slope β_1 (left) and of the correlation ρ_u between the by-subject varying intercepts and varying slopes (right).

Inference

Having fit a correlated varying intercepts, varying slopes model, we now explain one way to carry out statistical inference, using credible intervals. We have used this approach to draw inferences from data in previously published work (e.g., Frank, Trompenaars, & Vasishth, 2015, Hofmeister & Vasishth, 2014, Safavi, Husain, & Vasishth, 2016). There are of course other approaches possible for carrying out inference. Bayes Factors are an example; see Lee and Wagenmakers (2013) and Rouder and Morey (2012). Another is to define a Region of Practical Equivalence (Kruschke, 2014). The reader can choose the approach they find the most appealing. For further discussion of Bayes Factors, with example code, see Nicenboim and Vasishth (2016).

The result of fitting the varying intercepts, varying slopes model is the posterior distribution of the model parameters. Direct inference from the posterior distributions is possible. For instance, we can find the posterior probability with which the fixed intercept β_1 or the correlation ρ_u between by-subject varying intercepts and slopes take on any given

value by consulting the marginal posterior distributions whose histograms are shown in Figure 6. The information conveyed by such graphs can be sharpened by using the 95% credible interval, mentioned earlier. Approximately 95% of the posterior density of β_1 lies between the 2.5th percentile -0.09 and the 97.5th percentile 0.02. This leads us to conclude that the slope β_1 for relative clause type `so` is less than zero with probability 90% (see Listing 7, line 8). Since zero is included in the credible interval, it is difficult to draw the inference that object relative clauses are read faster than subject relative clauses. However, one could perhaps still make a weak claim that object relatives are easier to process, especially if a lot of evidence has accumulated in other experiments that supports such a conclusion (see Vasishth, Chen, Li, & Guo, 2013 for a more detailed discussion). Meta-analysis of existing studies can help in obtaining a better estimate of the posterior distribution of a parameter; for psycholinguistic examples, see Engelmann, Jäger, and Vasishth (2016); Mahowald, James, Futrell, and Gibson (2016); Vasishth (2015).

What about the correlations between varying intercepts and varying slopes for subject and for item? What can we infer from the analysis about these relationships? The 95% credible interval for ρ_u is $(-1, 0.1)$. Our belief that ρ_u is less than zero is rather uncertain, although we can conclude that ρ_u is less than zero with probability 90%. There is only weak evidence that subjects who read faster than average exhibit greater slowdowns at the head noun of object relative clauses than subjects who read slower than average. For the by-item varying intercepts and slopes, it is pretty clear that we do not have enough data (15 items) to draw any conclusions. For these data, it probably makes sense to fit a simpler model (Bates, Kliegl, et al., 2015), with only varying intercepts and slopes for subject, and only varying intercepts for items; although there is no harm done in this particular example if we fit a model with a full variance-covariance matrix for both subjects and items.

In sum, regarding our main research question, our conclusion here is that we cannot say that object relatives are harder to process than subject relatives, because the credible interval for β_1 includes zero. However, one could argue that there is *some* weak evidence in favor of the hypothesis, since the posterior probability of the parameter being negative

is approximately 90%.

Further reading

We hope that this tutorial has given the reader a flavor of what it would be like to fit Bayesian linear mixed models. There is of course much more to say on the topic, and we hope that the interested reader will take a look at some of the excellent books that have recently come out. We suggest below a sequence of reading that we found helpful. A good first general textbook is by Gelman and Hill (2007); it begins with the frequentist approach and only later transitions to Bayesian models. The book by McElreath (2016) is also excellent. For those looking for a psychology-specific introduction, the books by Kruschke (2014) and Lee and Wagenmakers (2013) are to be recommended, although for the latter the going might be easier if the reader has already looked at Gelman and Hill (2007). As a second book, Lunn et al. (2012) is recommended; it provides many interesting and useful examples using the BUGS language, which are discussed in exceptionally clear language. Many of these books use the BUGS syntax (Lunn et al., 2000), which the probabilistic programming language JAGS (Plummer, 2012) also adopts; however, Stan code for these books is slowly becoming available on the Stan home page (`https://github.com/stan-dev/example-models/wiki`). For those with introductory calculus, a slightly more technical introduction to Bayesian methods by Lynch (2007) is an excellent choice. Finally, the textbook by Gelman et al. (2014) is the definitive modern guide, and provides a more advanced treatment.

Acknowledgements

We are grateful to the developers of Stan (in particular, Andrew Gelman, Bob Carpenter) and members of the Stan mailing list for their advice regarding model specification. Douglas Bates and Reinhold Kliegl have helped considerably over the years in improving our understanding of LMMs from a frequentist perspective. We also thank Edward Gibson for releasing his published data. Titus von der Malsburg, Lena Jäger, and Bruno Nicenboim provided useful comments on previous drafts.

References

Baayen, R. H., Bates, D., Kliegl, R., & Vasishth, S. (2015). RePsychLing: Data sets from psychology and linguistics experiments [Computer software manual]. Retrieved from `https://github.com/dmbates/RePsychLing` (R package version 0.0.4)

Baayen, R. H., Vasishth, S., Bates, D., & Kliegl, R. (2016). *The cave of shadows: Addressing the human factor with generalized additive mixed models.* (arXiv preprint)

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*(3), 255–278. doi: 10.1016/j.jml.2012.11.001

Bates, D. M., Kliegl, R., Vasishth, S., & Baayen, H. (2015). *Parsimonious mixed models.* Retrieved from `http://arxiv.org/abs/1506.04967` (ArXiv e-print; submitted to *Journal of Memory and Language*)

Bates, D. M., Mächler, M., Bolker, B. M., & Walker, S. C. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*(1), 1–48. doi: 10.18637/jss.v067.i01

Engelmann, F., Jäger, L. A., & Vasishth, S. (2016). *The determinants of retrieval interference in dependency resolution: Review and computational modeling.* (Manuscript re-submitted (30 April 2016))

Frank, S. L., Trompenaars, T., & Vasishth, S. (2015). Cross-linguistic differences in processing double-embedded relative clauses: Working-memory constraints or language statistics? *Cognitive Science*, n/a–n/a. doi: 10.1111/cogs.12247

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian analysis*, *1*(3), 515–534.

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.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple

sequences. *Statistical science*, 457–472.

Gibson, E., & Wu, H.-H. I. (2013). Processing Chinese relative clauses in context. *Language and Cognitive Processes*, *28*(1-2), 125–155. doi: 0.1080/01690965.2010.536656

Hofmeister, P., & Vasishth, S. (2014). Distinctiveness and encoding effects in online sentence comprehension. *Frontiers in Psychology*, *5*, 1237. doi: 10.3389/fpsyg.2014.01237

Hsiao, F. P.-F., & Gibson, E. (2003). Processing relative clauses in Chinese. *Cognition*, *90*, 3–27. doi: 10.1016/S0010-0277(03)00124-0

Just, M., & Carpenter, P. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, *99*(1), 122–149. doi: 10.1037/0033-295X.99.1.122

Kliegl, R., Wei, P., Dambacher, M., Yan, M., & Zhou, X. (2010). Experimental effects and individual differences in linear mixed models: Estimating the relationship between spatial, object, and attraction effects in visual attention. *Frontiers in Psychology*, *1*, 238. doi: 10.3389/fpsyg.2010.00238

Kruschke, J. (2014). *Doing Bayesian Data Analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

Lavine, M. (1999). What is Bayesian statistics and why everything else is wrong. *The Journal of Undergraduate Mathematics and Its Applications*, *20*, 165–174. Retrieved from `https://www2.stat.duke.edu/courses/Spring06/sta114/whatisbayes.pdf`

Lee, M. D., & Wagenmakers, E.-J. (2013). *Bayesian cognitive modeling: A practical course*. Cambridge University Press.

Lewandowski, D., Kurowicka, D., & Joe, H. (2009). Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis*, *100*(9), 1989–2001.

Lunn, D., Jackson, C., Spiegelhalter, D. J., Best, N., & Thomas, A. (2012). *The BUGS book: A practical introduction to Bayesian analysis* (Vol. 98). CRC 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. doi: 10.1023/A:1008929526011

Lynch, S. M. (2007). *Introduction to applied Bayesian statistics and estimation for social scientists*. Springer. doi: 10.1007/978-0-387-71265-9

Mahowald, K., James, A., Futrell, R., & Gibson, E. (2016). A meta-analysis of syntactic priming in language production. *Journal of Memory and Language*.

Matuschek, H., Kliegl, R., Vasishth, S., Baayen, R. H., & Bates, D. (2016). *Balancing Type I Error and Power in Linear Mixed Models.* (arXiv preprint)

McElreath, R. (2016). *Statistical rethinking: A bayesian course with examples in r and stan* (Vol. 122). CRC Press.

McElreath, R. (2016). *Statistical rethinking: A Bayesian course with examples in R and Stan*. Chapman and Hall.

Morey, R. D., Hoekstra, R., Rouder, J. N., Lee, M. D., & Wagenmakers, E.-J. (2015). The fallacy of placing confidence in confidence intervals. *Psychonomic Bulletin & Review*, *23*(1), 103–123. doi: 10.3758/s13423-015-0947-8

Nicenboim, B., & Vasishth, S. (2016). *Statistical methods for linguistics research: Foundational ideas – Part II.* (arXiv preprint)

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS*. New York: Springer-Verlag. doi: 10.1007/b98882

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

R Development Core Team. (2006). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org` (ISBN 3-900051-07-0)

Raftery, A. E., & Lewis, S. (1992). How many iterations in the Gibbs sampler? In J. Bernardo, J. Berger, A. Dawid, & A. Smith (Eds.), *Bayesian statistics 4* (pp. 763–773). Oxford University Press.

Rouder, J. N. (2005). Are unshifted distributional models appropriate for response time? *Psychometrika*, *70*, 377–381. doi: 10.1007/s11336-005-1297-7

Rouder, J. N., & Morey, R. D. (2012). Default Bayes factors for model selection in regression. *Multivariate Behavioral Research*, *47*(6), 877–903. doi: 10.1080/00273171.2012.734737

Safavi, M. S., Husain, S., & Vasishth, S. (2016). Dependency resolution difficulty increases with distance in persian separable complex predicates: Implications for expectation and memory-based accounts. *Frontiers in Psychology*, *7*. (Special Issue on Encoding and Navigating Linguistic Representations in Memory) doi: 10.3389/fpsyg.2016.00403

Spiegelhalter, D. J., Abrams, K. R., & Myles, J. P. (2004). *Bayesian approaches to clinical trials and health-care evaluation* (Vol. 13). John Wiley & Sons. doi: 10.1002/0470092602

Stan Development Team. (2014). Stan modeling language users guide and reference manual, version 2.4 [Computer software manual]. Retrieved from `http://mc-stan.org/`

Vasishth, S. (2015). *A meta-analysis of relative clause processing in Mandarin Chinese using bias modelling.* Sheffield, UK. Retrieved from `http://www.ling.uni-potsdam.de/~vasishth/pdfs/VasishthMScStatistics.pdf`

Vasishth, S., Chen, Z., Li, Q., & Guo, G. (2013, 10). Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PLoS ONE*, *8*(10), e77006. doi: 10.1371/journal.pone.0077006

Vasishth, S., & Nicenboim, B. (2016). Statistical methods for linguistic research: Foundational ideas – Part I.

(accepted, Language and Linguistics Compass)

Xie, Y. (2015). knitr: A general-purpose package for dynamic report generation in R [Computer software manual]. (R package version 1.11)

Appendix A

Cholesky factorization

A correlation matrix can be factored into a *square root of the matrix*; one method is the Cholesky factorization. Given a correlation matrix C, we can obtain its square root L. The square root of a matrix is such that we can square L to get the correlation matrix C back. We illustrate the matrix square root with a simple example. Suppose we have a correlation matrix:

$$C = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \tag{10}$$

We can use the Cholesky factorization function in R, `chol`, to derive the lower triangular square root L of this matrix. This gives us:

$$L = \begin{pmatrix} 1 & 0 \\ -0.5 & 0.8660254 \end{pmatrix} \tag{11}$$

We confirm that this is a square root by multiplying L with itself to get the correlation matrix back (squaring a matrix is done by multiplying the matrix by its transpose):

$$LL^{\intercal} = \begin{pmatrix} 1 & 0 \\ -0.5 & 0.8660254 \end{pmatrix} \begin{pmatrix} 1 & -0.5 \\ 0 & 0.8660254 \end{pmatrix} = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \tag{12}$$

The reason that the Cholesky factorization is useful for LMMs is that we use it to generate the by-subject and by-item random intercepts and slopes.

Generating correlated random variables using the Cholesky factor. The by-subject and by-item adjustments are generated using the following standard procedure for generating correlated random variables $\mathbf{x} = (x_1, x_2)$:

1. Given a vector of standard deviations (e.g., σ_{u0}, σ_{u1}), create a diagonal matrix:

$$\tau = \begin{pmatrix} \sigma_{u0} & 0 \\ 0 & \sigma_{u0} \end{pmatrix} \tag{13}$$

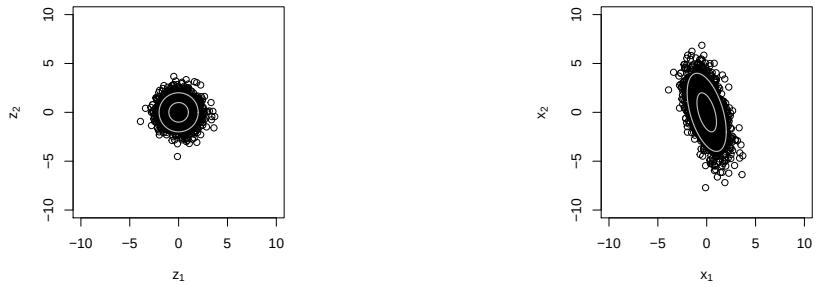

Figure A1. Uncorrelated random variables $\mathbf{z} = (z_1, z_2)^\intercal$ (left) and correlated random variables $\mathbf{x} = (x_1, x_2)^\intercal$ (right).

2. Premultiply the diagonalized matrix τ with the Cholesky factor L of the correlation matrix C to get a matrix Λ.

3. Generate values from a random variable $\mathbf{z} = (z_1, z_2)^\intercal$, where z_1 and z_2 each have independent $\mathcal{N}(0, 1)$ distributions (left panel of Figure A1).

4. Multiply Λ with $\mathbf{z}$; this generates the correlated random variables $\mathbf{x}$ (right panel of Figure A1).

It is helpful to walk through steps 1 to 4 of the procedure described above for generating correlated random intercepts and random slopes. These are carried out in lines 26 and 36 of Listing 8. The statement `diag_pre_multiply(sigma_u, L_u)` in line 26 computes the transpose matrix product (steps 1 and 2). The statement `to_vector(z_u) ~ normal(0,1);` in line 36 generates `z_u` as samples from the unit normal distribution (step 3). In line 26, the right multiplication of `diag_pre_multiply(sigma_u, L_u)` by `z_u`, a matrix of normally distributed random variables, yields the varying intercepts and slopes (step 4).

$$\begin{aligned}\begin{pmatrix} u_{01} & u_{11} \\ u_{02} & u_{12} \\ \vdots & \vdots \\ u_{0J} & u_{1J} \end{pmatrix} &= (\text{diag}(\sigma_{u0}, \sigma_{u1}) L_u \mathbf{z}_u)^\intercal \\ &= \left(\begin{pmatrix} \sigma_{u0} & 0 \\ 0 & \sigma_{01} \end{pmatrix} \begin{pmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{pmatrix} \begin{pmatrix} z_{11} & z_{12} & \dots & z_{1J} \\ z_{21} & z_{22} & \dots & z_{2J} \end{pmatrix} \right)^\intercal \end{aligned} \tag{14}$$

Appendix B

Matrix formulation of the linear mixed model

In the body of the text, we fit four models of increasing complexity to the data-set of Gibson and Wu (2013). In all specifications, there was an explicit vector `so` for the predictor variable in Stan. However, if we want to fit more complex models with many categorical and continuous predictors and interactions, this approach requires increasingly complex specifications in Stan code. Alternatively, we can use the matrix formulation of the linear mixed model that allows for using the same code for models of different complexity. In the following, we will apply this approach for an alternative version of the correlated varying intercepts, varying slopes model, which includes random intercepts and slopes for subjects and items.

We build up the model specification by first noting that, for each subject, the by-subject varying intercept u_0 and slope u_1 have a multivariate normal prior distribution with mean zero and covariance matrix Σ_u. Similarly, for each item, the by-item varying intercept w_0 and slope w_1 have a multivariate normal prior distribution with mean zero and covariance matrix Σ_w. The error ε is assumed to have a normal distribution with mean zero and standard deviation σ_e.

We proceed to implement the model in Stan. Instead of passing the predictor `so` to `stan` as vector, as we did earlier, we make `so` into a design matrix `X` us-

ing the function `model.matrix` available in R (see Listing 9, line 2).[8] The command `model.matrix(~ 1 + so, rDat)` creates a model matrix with two fixed effects, the intercept (`1`) and a factor (`so`), based on the data frame `rDat`. The first column of the design matrix `X` consists of all ones; this column represents the intercept. The second column is the predictor `so` and consists of values in $\{-1, 1\}$. The model matrix thus consists of a two-level factorial design, with blocks of this design repeated for each subject. For the full data-set, we could write it very compactly in matrix form as follows:

$$\log \mathtt{rt} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_u\mathbf{u} + \mathbf{Z}_w\mathbf{w} + \boldsymbol{\varepsilon} \tag{15}$$

Here, $\mathbf{X}$ is the $N \times P$ model matrix (with $N = 547$, since we have 547 data points; and $P = 2$ since we have the intercept plus another fixed effect), $\boldsymbol{\beta}$ is a vector of length P including fixed effects parameters, $\mathbf{Z}_u$ and $\mathbf{Z}_w$ are the subject and item model matrices ($N \times P$), and $\mathbf{u}$ and $\mathbf{w}$ are the by-subject and by-item adjustments to the fixed effects estimates; these are identical to the design matrix $\mathbf{X}$ in the model with varying intercepts and varying slopes included. For more examples of similar model specifications in Stan, see the R package `RePsychLing` on github (`https://github.com/dmbates/RePsychLing`).

Note that we remove the column names and the attributes of the model matrix `X` in order to use it for Stan; refer to Listing 9. Having defined the model, we proceed to assemble the list `stanDat` of data, relying on the above matrix formulation. The number `N` of observations, the number `J` of subjects and `K` of items, the reading times `rt`, and the subject and item indicator variables `subj` and `item` are familiar from the previous models presented. The integer `P` is the number of fixed effects (two including the intercept). Model 4 includes a varying intercept u_0 and a varying slope u_1 for each subject, and so the number `n_u` of by-subject random effects equals `P`. Likewise, Model 4 includes a varying intercept w_0 and a varying slope w_1 for each item, and so the number `n_w` of by-item random effects

[8] Here, we would like to acknowledge the contribution of Douglas Bates in specifying the model in this general matrix form.

```
# Make design matrix
X <- unname(model.matrix(~ 1 + so, rDat))
attr(X, "assign") <- NULL
# Make Stan data
stanDat <- list(N = nrow(X),
                P = ncol(X),
                n_u = ncol(X),
                n_w = ncol(X),
                X = X,
                Z_u = X,
                Z_w = X,
                J = nlevels(rDat$subj),
                K = nlevels(rDat$item),
                rt = rDat$rt,
                subj = as.integer(rDat$subj),
                item = as.integer(rDat$item))
# Fit the model
matrixFit <- stan(file = "matrixModel.stan", data = stanDat,
                  iter = 2000, chains = 4)
```

Listing 9: Matrix formulation code for running the varying intercepts, varying slopes model.

also equals `P`.

We also have to adapt the Stan code to the model formulation (see Listing 10). The data block contains the corresponding variables. Using the command `row_vector[P] X[N]`, we declare the fixed effects design matrix `X` as an array of `N` row vectors of length `P` whose components are the predictors associated with the `N` reading times. Likewise for the subject and item random effects design matrices `Z_u` and `Z_w`, which correspond to $\mathbf{Z}_u$ and $\mathbf{Z}_w$ respectively in Equation 15. The vector `beta` contains the fixed effects β_0 and β_1. The matrices `L_u`, `L_w` and the arrays `z_u`, `z_w` of vectors (not to be confused with the design matrices `Z_u` and `Z_w`) will generate the varying intercepts and slopes u_0, u_1 and w_0, w_1, using the procedure described for the varying intercepts, varying slopes model. For example, the command `vector[n_u] u[J]` specifies `u` as an array of `J` vectors of length `n_u`; hence, there is one vector per subject. The vector `sigma_u` contains the standard deviations of the by-subject varying intercepts and slopes u_0, u_1, and the vector `sigma_w` contains the standard deviations of the by-item varying intercepts and slopes w_0, w_1. The variable `sigma_e` is the standard deviation σ_e of the error ε. The transformed parameters block generates the by-subject intercepts and slopes u_0, u_1 and the by-item intercepts and slopes

```
data {
  int<lower=0> N;               //n trials
  int<lower=1> P;               //n fixefs
  int<lower=0> J;               //n subjects
  int<lower=1> n_u;             //n subj ranefs
  int<lower=0> K;               //n items
  int<lower=1> n_w;             //n item ranefs
  int<lower=1,upper=J> subj[N]; //subject indicator
  int<lower=1,upper=K> item[N]; //item indicator
  row_vector[P] X[N];           //fixed effects design matrix
  row_vector[n_u] Z_u[N];       //subj ranef design matrix
  row_vector[n_w] Z_w[N];       //item ranef design matrix
  vector[N] rt;                 //reading time
}
parameters {
  vector[P] beta;               //fixed effects coefs
  cholesky_factor_corr[n_u] L_u;  //cholesky factor of subj ranef corr matrix
  cholesky_factor_corr[n_w] L_w;  //cholesky factor of item ranef corr matrix
  vector<lower=0>[n_u] sigma_u; //subj ranef std
  vector<lower=0>[n_w] sigma_w; //item ranef std
  real<lower=0> sigma_e;        //residual std
  vector[n_u] z_u[J];           //subj ranef
  vector[n_w] z_w[K];           //item ranef
}
transformed parameters {
  vector[n_u] u[J];             //subj ranefs
  vector[n_w] w[K];             //item ranefs
  {
    matrix[n_u,n_u] Sigma_u;    //subj ranef cov matrix
    matrix[n_w,n_w] Sigma_w;    //item ranef cov matrix
    Sigma_u = diag_pre_multiply(sigma_u, L_u);
    Sigma_w = diag_pre_multiply(sigma_w, L_w);
    for(j in 1:J)
      u[j] = Sigma_u * z_u[j];
    for(k in 1:K)
      w[k] = Sigma_w * z_w[k];
  }
}
model {
  //priors
  L_u ~ lkj_corr_cholesky(2.0);
  L_w ~ lkj_corr_cholesky(2.0);
  for (j in 1:J)
    z_u[j] ~ normal(0,1);
  for (k in 1:K)
    z_w[k] ~ normal(0,1);
  //likelihood
  for (i in 1:N)
    rt[i] ~ lognormal(X[i] * beta +
                      Z_u[i] * u[subj[i]] +
                      Z_w[i] * w[item[i]],
                      sigma_e);
}
```

Listing 10: Stan code for the matrix formulation of the varying intercepts, varying slopes model.

w_0, w_1.

We place lkj priors on the random effects correlation matrices through the `lkj_corr_cholesky(2.0)` priors on their Cholesky factors `L_u` and `L_w`. We implicitly place uniform priors on the fixed effects β_0, β_1, the random effects standard deviations σ_{u0}, σ_{u1}, and σ_{w0}, σ_{w1} and the error standard deviation σ_e by omitting any prior specifications for them in the model block. We specify the likelihood with the probability statement that `rt[i]` is distributed log-normally with mean `X[i] * beta + Z_u[i] * u[subj[i]] + Z_w[i] * w[item[i]]` and standard deviation `sigma_e`. The next step towards model-fitting is to pass the list `stanDat` to `stan`, which compiles a C++ program to sample from the posterior distribution of the model parameters.

A major advantage of the above matrix formulation is that we do not need to write a new Stan model for a future repeated measures design. All we have to do now is define the design matrix $\mathbf{X}$ appropriately, and include it (along with appropriately defined $\mathbf{Z}_u$ and $\mathbf{Z}_w$ for the subjects and items random effects) as part of the data specification that is passed to Stan.