

User-friendly Bayesian regression modeling: A tutorial with `rstanarm` and `shinystan`

Chelsea Muth^a, , Zita Oravecz^a & Jonah Gabry^b

^aPennsylvania State University

^bColumbia University

Abstract ■ This tutorial provides a pragmatic introduction to specifying, estimating and interpreting single-level and hierarchical linear regression models in the Bayesian framework. We start by summarizing why one should consider the Bayesian approach to the most common forms of regression. Next we introduce the R package `rstanarm` for Bayesian applied regression modeling. An overview of `rstanarm` fundamentals accompanies step-by-step guidance for fitting a single-level regression model with the `stan_glm` function, and fitting hierarchical regression models with the `stan_lmer` function, illustrated with data from an experience sampling study on changes in affective states. Exploration of the results is facilitated by the intuitive and user-friendly `shinystan` package. Data and scripts are available on the Open Science Framework page of the project. For readers unfamiliar with R, this tutorial is self-contained to enable all researchers who apply regression techniques to try these methods with their own data. Regression modeling with the functions in the `rstanarm` package will be a straightforward transition for researchers familiar with their frequentist counterparts, `lm` (or `glm`) and `lmer`.

Keywords ■ Bayesian modeling, regression, hierarchical linear model. **Tools** ■ Stan, R, `rstanarm`.

muth.ca@gmail.com

CM: n/a; ZO: 0000-0002-9070-3329; JG: n/a

[10.20982/tqmp.14.2.p099](https://doi.org/10.20982/tqmp.14.2.p099)

Acting Editor ■ Denis Cousineau (Université d'Ottawa)

Reviewers

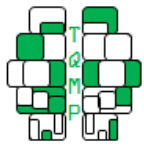
■ One anonymous reviewer.

Introduction

This self-contained tutorial demonstrates how the R package `rstanarm` (Gabry & Goodrich, 2017b) can be used to fit single-level and hierarchical regression models in the Bayesian statistical framework. `rstanarm` is a Stan-based package (Stan Development Team, 2017; Carpenter et al., 2017) that mimics the high-level syntax of R's popular functions `lm` and `lmer` (Bates, Mächler, Bolker, & Walker, 2015), making the Bayesian model specification more succinct than relying on JAGS (Plummer, 2003) or Stan alone (for a tutorial for modeling in Stan alone see e.g., Sorensen, Hohenstein, & Vasishth, 2016). In addition, the graphical user interface based `shinystan` R package (Gabry, 2017) complements `rstanarm`, and provides intuitive ways of assessing model convergence, fit, and results. We will demonstrate these tools via fitting single-level and hierarchical linear regression models, which are typically used to answer common questions in social and behavioral

research. For instance, our applied example centers on self-reported affective states, more specifically on the link between daily self-reports of valence (pleasantness) and arousal (activation) levels. First we will use a single-level regression to predict valence levels (pleasantness) from arousal (activation) levels. We will improve on this model by nesting the valence and arousal data in persons and casting the problem in a hierarchical, or multilevel, modeling framework. The hierarchical regression model simultaneously measures person-level and population-level trends, with the two-levels informing each other and improving estimation accuracy.

While in social and behavioral sciences regression models are most often estimated in the frequentist statistical framework (see e.g., Cohen, Cohen, West, & Aiken, 2013), casting them in the Bayesian statistical framework offers unique advantages, for example in terms of interpretability of estimates and the flexibility of fitting increasingly complex models (Korner-Nievergelt et al., 2015; McEl-



reath, 2016). Bayesian regression modeling has become increasingly accessible and efficient due to advances in statistical software, including generic estimation engines such as Stan (Stan Development Team, 2017; Carpenter et al., 2017) and JAGS (Plummer, 2003), and packaged software such as MPlus (Muthén & Muthén, 2008).

This tutorial shows how to specify and compute the most common regression models in the Bayesian framework. To guide the reader through the fundamentals of single-level and hierarchical linear regression modeling, we give step-by-step instructions for how to fit such models using the `rstanarm` R package (Gabry & Goodrich, 2017b). The code included in the tutorial and supplemental files are free and open-source. Moreover, we also highlight the advantages of Bayesian regression methods. We provide both 1) a concise non-technical introduction to the key statistical concepts behind Bayesian inference and regression modeling; and 2) step-by-step guides to interpreting the results of applied examples with `rstanarm` — a simple regression with `stan_glm`, and a hierarchical regression with `stan_lmer`. Our goal is to ease researchers into Bayesian regression modeling, while preventing the researcher from any harmful shortcuts or oversights. Our intended audience for this tutorial includes applied researchers and graduate-level students who are interested in non-technical discussions of new quantitative methodologies. Moreover, the content is ideally suited for classrooms of learners who come from a wide range of research areas — both quantitative and substantive. Researchers familiar with regression modeling in R will be pleased to find a seamless transition from the R functions `lm` and `lmer` to the `rstanarm` functions `stan_glm` and `stan_lmer`.

First, the principles and fundamentals of the Bayesian statistical framework are provided. Second, the experience sampling study of (Csikszentmihalyi & Larson, 1987) on affective states is introduced to preface our two didactic empirical applications. Third, the fundamentals of single-level regression modeling in the Bayesian framework are presented, with step-by-step guidance for fitting a single-level regression model to the data. Fourth, the extension of the single-level model to a Bayesian hierarchical linear model is also presented with fundamentals and step-by-step guidance on how to specify, run, and assess the model. Lastly, the tutorial concludes by summarizing Bayesian regression modeling.

The Bayesian statistical framework

This section briefly reviews the key components of Bayesian data analysis. We encourage the reader to further explore the literature on Bayesian methods, for example via (Gelman & Hill, 2007; Gelman et al., 2013; Kruschke, 2015; McElreath, 2016).

The Bayesian approach entails using a full probability model that describes not only our uncertainty in the value of an outcome variable y conditional on some unknown parameter(s) θ , but also our a priori uncertainty about the parameter(s) θ themselves. When it comes to regression models, besides the outcome variable y , we also have predictor variables, denoted x . The goal is to update our beliefs about the parameters θ (e.g., the coefficients in a regression model) using our model and data (x and y). The relationship between our prior beliefs about the parameters (before observing the data) and our posterior beliefs about the parameters (after observing the data) is described by Bayes' theorem,

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}, \quad (1)$$

which states that the posterior probability distribution, $p(\theta|y)$, of parameter(s) θ given data y equals the product of a likelihood function $p(y|\theta)$ and prior distribution $p(\theta)$, divided by the marginal distribution of the data $p(y)$. The denominator $p(y)$ is the marginal likelihood: that is the likelihood averaged over the model parameters, weighted by their prior probabilities. It is often referred to as the normalizing constant. It does not depend on the parameter(s) θ and therefore provides no information about which values of θ are more or less probable (since it is averaged over all possible values of θ). Therefore to update our knowledge about θ based on the data y , we need only focus on the numerator $p(y|\theta)p(\theta)$. For this reason it is common to see Equation 1 written as

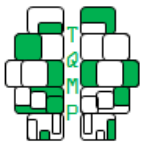
$$p(\theta|y) \propto p(y|\theta)p(\theta),$$

where \propto indicates proportionality.

For regression models we can be more explicit in our statement of Bayes' theorem by including the predictors (regressors) x . We then have

$$p(\theta|y, x) \propto p(y|\theta, x)p(\theta|x), \quad (2)$$

which is the same as before except everything is now conditional on x . In some situations we need to worry about the potential for nontrivial dependence between x and θ , for instance if we are concerned about the potential for sample selection bias (i.e., the sample is not representative in the sense that some people with a particular attribute are systematically excluded). However, in standard regression settings it is assumed that x and θ are independent and so $p(\theta|x)$ simplifies to $p(\theta)$, meaning our prior beliefs about the parameters do not depend on information provided by the predictors. The components of Bayes' theorem introduced above are the fundamental mathematical objects we need to be familiar with when doing Bayesian inference:



- The likelihood, $p(y|\theta, x)$, is a quantity proportional to the probability of the observed data, or more precisely the joint probability of the data y for all possible values of the parameter(s) θ (and given the observed predictors x). For each y_i , $p(y_i|\theta, x_i)$ represents the data generating process that is assumed to have produced the observation. If the observations are conditionally independent given the parameters (i.e., once we know the value of the parameter, the value of one observation no longer improves our prediction of another observation) then the likelihood is the product of individual likelihood contributions from each individual data point. With respect to the parameters, the likelihood does not integrate to 1 and is not a probability density function.
- The prior, $p(\theta)$, is a probability distribution that describes our uncertainty about the parameters θ before observing the data y . Typically we have at least a minimal level of prior information. For example, if all variables (outcome and predictors) are scaled reasonably in relation to each other then we generally know a priori that we should not expect extreme estimates of regression coefficients. Previous research can also be a valuable source of prior information that can be used to develop a so-called informative prior distribution. For further reading on the use of informative priors, and how specifically to incorporate information from previous research into the prior distribution, see for example Korner-Nievergelt et al. (2015).
- The posterior, $p(\theta|y, x)$, is the joint probability distribution of all parameters θ reflecting our updated knowledge about the parameters after we have observed y . The posterior can be thought of as a compromise between the data model (likelihood) and the prior, and describes the relative plausibility of all parameter values conditional on the model. This is the target estimate when we fit a Bayesian model, and facilitates multifaceted and intuitive probabilistic inference.

A fundamental difference between the Bayesian and the frequentist frameworks lies in which quantities are assumed to be fixed, and whether we are allowed to describe uncertainty about parameters using probability theory. The frequentist paradigm is concerned with the probability of the observed data given fixed parameters (i.e., θ does not have a probability distribution) and frequentist inference pertains to a sequence of hypothetical datasets (y vectors) that could have been observed. On the other hand, Bayesian inference pertains to the particular set of N observations that were observed and Bayesians are interested in the probability distribution of the parameters

given this fixed (observed) data.

One of the most important implications of this distinction is that, although the frequentist approach can provide point estimates (and sometimes standard errors) based on long-run properties of estimators, it does not permit making probability statements about parameters. In the frequentist framework we cannot answer questions like *What is the probability that θ is positive?* On the other hand, describing parameter uncertainty using probability theory is fundamental to Bayesian inference. Instead of point estimates we obtain a posterior probability distribution over all possible parameter values conditional on the model and observed data. Using the posterior distribution we can easily make probabilistic statements about parameters (and functions of parameters) of interest, including, if we want, the probability that the parameter value lies in any given interval.

Modeling in the Bayesian framework: four key steps

We summarize the Bayesian approach to data analysis in four key steps:

Step 1. Specify the data model and prior. The model is a collection of probability distributions conditional on different values for the parameters. The prior provides the a priori plausibility of the different parameter values. The product of the prior and the likelihood is proportional to the posterior distribution of the parameters (Equation 1).

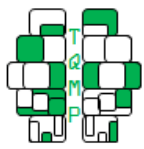
Step 2. Estimate the model parameters. Bayes' theorem as presented in Equation 1 is used to update beliefs about the parameters based on knowledge obtained from observing the data. Estimating regression models in the Bayesian framework typically involves using a numerical algorithm to draw a representative sample from the posterior distribution.

Step 3. Check sampling quality and model fit.¹ Graphical and numerical checks are necessary to confirm that the sample obtained in the previous step adequately describes the posterior and that the observed data is plausible under the model. If not, the model needs to be revised, and we might want to explore alternative model formulations.

Step 4. Summarize and interpret results. We assess estimates, interpret results, and make posterior predictions.

Compared to frequentist approaches, which typically only require likelihood specification, Step 1 can be more involved for Bayesian inference. As mentioned earlier, the most unfamiliar task for a researcher first studying Bayesian modeling is specifying a prior distribution for all unknown parameters. As for the rest of the steps, the algorithms and model checks presented in this tutorial are unique to the Bayesian framework, but while these

¹In addition, it is good practice to generate datasets with different parameter settings, based on the proposed model, and to check how well the parameter values can be recovered. See more details in Stan Development Team (2017), section 65.6. (Fit Simulated Data).



specifics may be unfamiliar, the approach is conceptually straightforward. Bayesian parameter estimation is based on learning about the posterior distribution of the parameters. For some simple models, posterior distributions (and quantities of interest based on posterior distributions) can be calculated analytically (closed-form solutions), but for almost all non-trivial models the full posterior has to be approximated numerically by sampling (simulating draws) from the posterior distribution. Based on the Monte Carlo principle (Metropolis & Ulam, 1949), if we draw a large enough sample from any probability distribution we can use it to accurately describe the distribution and compute quantities of interest, such as point estimates or intervals of any kind. Before moving to our instructive regression applications, we first review the main concepts that a researcher should understand about the sampling component of Bayesian estimation – in this case, sampling with Stan.

Estimating models using Stan and rstanarm

Stan was designed to be efficient and reliable for complex models with high dimensional posterior distributions while also capable of fitting simpler models easily (Hoffman & Gelman, 2014; Carpenter et al., 2017; Stan Development Team, 2017). Stan is available through a variety of interfaces including RStan in R and PyStan in Python, but using these interfaces requires knowledge not only of R or Python but also the Stan modeling language.² In contrast, in this tutorial we show how to fit models via *rstanarm*, an R package written by Stan developers that makes a subset of Stan's functionality available using only programming techniques already familiar to R users. With *rstanarm*, it is easy to use Stan to fit many commonly applied regression models, and *rstanarm* users are at an advantage over other programmers for various reasons. Firstly, *rstanarm* is user-friendly: although the model fitting relies on the Stan estimation engine, *rstanarm* is generally more easy-to-use than Stan and offers comprehensive default settings that allow the user to easily specify models. In addition to its ease of use, *rstanarm* offers the huge benefit of Stan's estimation power: compared to Bayesian estimation engines such as JAGS and BUGS (Spiegelhalter, Thomas, Best, & Gilks, 1996), the estimation algorithms in Stan were developed to handle sampling from the posterior of high-dimensional multilevel models. Compared to the user-friendly, graphical user interface based JASP (JASP Team, 2017), *rstanarm* can fit a wider range of models, including hierarchical models; however it does not calculate the Bayes Factor by default. Lastly, *rstanarm* has the advantage of being freely accessible, in contrast to other generic software engines with intuitive

syntax, such as MPlus (Muthén & Muthén, 2008) and Amos (Arbuckle, 1999).

Most commonly, Bayesian software packages employ simulation techniques such as Markov chain Monte Carlo (MCMC) to obtain a sample consisting of many draws from the target posterior distribution. An MCMC algorithm uses a Markov chain – a sequence of states in which the location of the next state depends on the current state of the chain – to explore the shape of the desired posterior distribution. The most powerful algorithm available in Stan is the Hamiltonian Monte Carlo algorithm (see, e.g., Betancourt, 2017, for an accessible conceptual introduction).

In theory the time a chain spends in any region of the parameter space will be proportional to its posterior probability, in which case a sufficiently long and well behaved chain will nicely approximate the posterior distribution. In order to check that the algorithm is behaving properly, and because in practice we can only run a chain for a finite number of iterations, we run multiple chains and check that they all converge to the same distribution even if initialized at different starting values.

To adequately estimate Bayesian models via MCMC, we need a sample that contains reliable information about the posterior distribution. In the subsequent sections, we introduce several diagnostics that help investigate whether the MCMC algorithm successfully obtained a sample from the target posterior distribution and whether that sample is sufficiently informative for the particular inference tasks of interest to the user. Several important MCMC diagnostic checks are implemented in the R packages *rstanarm* and *shinystan* (discussed later). For a more formal and thorough treatment, see Gelman et al. (2013) and Stan Development Team (2017).

Assessing convergence. First, we should check whether the chains converge to the same area. Recommended convergence checks include monitoring the \hat{R} statistic and visual checks Gelman et al. (2013, Chapter 11). The \hat{R} statistic (also referred to as the potential scale reduction factor) is based on comparing the variation between the chains to the variation within the chains. If all chains converge to the same region and behave similarly, then the variance between the chains should be approximately equal to the average variance within chains and the estimated \hat{R} will be close to 1. An example of converged chains is shown in Figure 1. The different colors indicate different chains, each of which started at a randomly selected initial value. This type of plot is called a trace plot. The trace plot in Figure 1 is for the regression coefficient (slope) of the arousal variable. The chains appear to be indistinguishable except for random noise and the \hat{R} value is close to 1. In practice, a commonly used heuristic is that $\hat{R} < 1.1$ for all parameters

²See <http://mc-stan.org/users/interfaces/> for details on the many available interfaces to Stan.

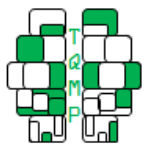
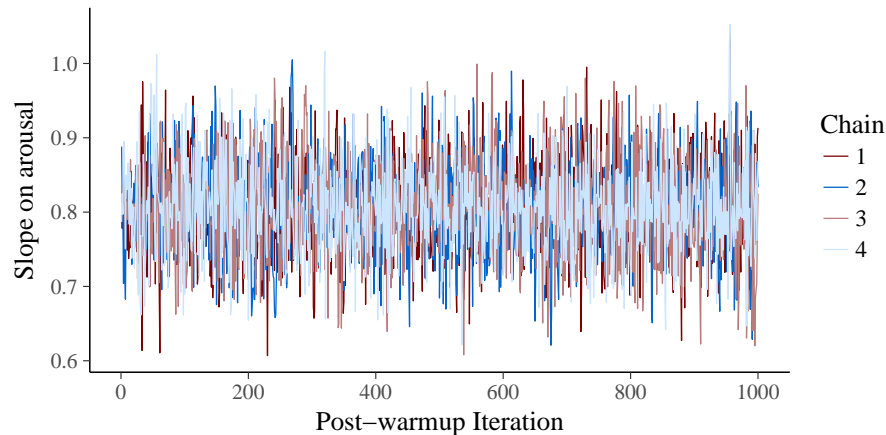


Figure 1 ■ Post-warmup trace plot for the regression coefficient (slope) on Arousal. The four chains appear to be the same except for noise with no discernible pattern, a strong sign of convergence.



is a strong (though not infallible) indicator of convergence. Before we interpret results and make inferences, it is necessary to make sure that all \hat{R} values are below 1.1. We can also examine trace plots, although this becomes impractical for models that have a large number of parameters. Many other types of MCMC visualizations are available in the `bayesplot` R package (Gabry & Mahr, 2017), which is easy to use with `rstanarm` models.³

Effective posterior sample size (ESS). Because MCMC does not return independent draws, the chains will exhibit some degree of autocorrelation. The strength of this dependence varies depending on the model as well as properties of the particular algorithm used. The lower the autocorrelation, the more independent pieces of information we have about the posterior. The approximate number of independent draws with the same estimation accuracy as our sample of correlated draws is referred to as the effective sample size (or n_{eff}). How large of an effective sample size a researcher needs depends on the amount of precision required for the inferential goals at hand. ESS larger than 1000 is generally more than sufficient for the types of problems studied by many social scientists.

Monte Carlo standard error (MCSE). Another way of assessing the error introduced by the MCMC approximation to the posterior is by calculating the Monte Carlo standard error. This diagnostic is also reported in the summary output from `rstanarm`. MCSE relates to ESS, and can be approximated by dividing the posterior standard deviation

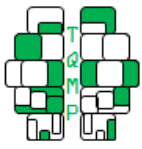
by the square root of the ESS. A low MCSE relative to the estimated posterior standard deviation will result in higher number of effective samples, which is desired. If the MCSE is large relative to the posterior standard deviation, then this sampling error variation masks the posterior standard deviation that is used to quantify the uncertainty in our estimate.

Posterior predictive checking (PPC). It is also important to evaluate whether our model adequately fits the data. Posterior predictive checking is the process of simulating data according to the fitted model and comparing the simulations to the observed data to look for important discrepancies. If the model fits the data well we should be able to replicate important features of the observed data in the simulations. To generate these simulations, we need to sample from the posterior predictive distribution, which is the distribution of the outcome variable implied by the posterior distribution of the model parameters. Each time the MCMC draws from the posterior distribution, we generate a new dataset according to the data generating process used in our model. When we fit models later in the tutorial we will demonstrate some of the various useful comparisons that can be made between the observed data and the posterior predictive simulations.⁴

Now that we have reviewed the fundamentals of the Bayesian framework and estimation via Stan, we proceed to Bayesian regression modeling. What follows is a didac-

³In fact, the plotting functions included with `rstanarm` also use `bayesplot` internally. We also recommend the `bayesplot` vignettes at <http://mc-stan.org/bayesplot>, which provide many worked examples of using visualization for MCMC diagnostics and in other stages of a Bayesian analysis.

⁴Posterior predictive checking is useful for assessing the fitted model in terms of predicting the observed data set (in-sample prediction). For out-of-sample prediction we also use the posterior predictive distribution but with new values of the predictor variables.



tic introduction to the basics of Bayesian regression – offering both conceptual review and practical step-by-step guidance for both single-level and hierarchical models – through two applied examples using `rstanarm`.

Application: Changes in levels of pleasantness and activation in everyday life

For our demonstration of Bayesian regression and hierarchical modeling, we take as a running example an experience sampling study on well-being, in which participants reported on their momentary levels of valence (pleasant feelings) and arousal (level of activation) for two weeks. This study sought to gain insight into multiple elements and dynamics of well-being at the population-level, as well as at the person-specific level. The momentary experience of valence and arousal is defined as one's core affect (Russell, 2003). In this tutorial, we take the core affect data as a toy example for demonstrating various aspects of Bayesian applied regression modeling, with less emphasis on exploring substantive implications.

Subjects in this dataset are 20 individuals who participated in a 14-day study, reporting on their valence and arousal levels (and other items related to their well-being) via six daily smartphone surveys. Participants were recruited at an east coast university in the United States and were mainly undergraduate students (8 male, 12 female, mean age 22.1 years). All participant interactions were overseen by the Institutional Review Board of the Pennsylvania State University (STUDY00001017). Participants were asked to indicate how active they feel 'right now' on a 0-100 continuous sliding scale, labeled as Not at all and Very much at the two endpoints. We transform this data into daily aggregates of valence and arousal levels, and model 272 self-reports of valence and arousal scores (20 subjects, each measured on 14 occasions, 8 data points missing). While this approach is sufficient as a toy example for fitting the basic simple and hierarchical regression models showcased in this tutorial, note that the richness of such intensive longitudinal data would normally call for advanced models that can dynamically capture the change over time (e.g., stochastic differential equation models).

The following applications incorporate step-by-step guidelines and computer scripts to fit a single-level linear regression and a two-level hierarchical linear model (HLM) in the Bayesian framework using R and `rstanarm`. With the single-level regression, we simply predict valence levels from arousal levels; in the hierarchical model, we do so while also nesting valence and arousal levels in persons. We provide commentary to guide the reader through the 4 steps for Bayesian estimation for both models, as well as code chunks with the necessary `rstanarm` commands.

We also show and interpret the output obtained from both models. The reader can copy our code from this paper directly and/or access the corresponding R files and data set via the Open Science Framework (OSF) website of the project.⁵

Fitting a Bayesian single-level regression to predict valence from arousal

A single-level linear regression model describes how an outcome variable is related to a linear function of one or more predictor variables. We can use the coefficients from a fitted regression model to help study to what degree changes in one variable are associated with changes in another. As noted earlier, this first, single-level model serves as a didactic example that is sufficient to demonstrate simple regression, but it performs a statistical error by ignoring the within-person dependency in the data. Later we will improve on this model when we introduce hierarchical regression.

In what follows we introduce the basic mathematical formulation of the single-level regression model, first using entirely mathematical notation, and then with the particular variables from our empirical example. The single-level regression model is specified as

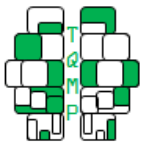
$$y_i = \beta_0 + X_i\beta + \epsilon_i. \quad (3)$$

In this representation, X_i is the i th row of the N by K matrix of predictors X (N = sample size, K = number of predictors), and the outcome for the i th observation y_i is predicted by $\beta_0 + X_i\beta = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} \dots + \beta_K X_{iK}$. The parameters β_0 and β are the intercept and vector of K regression coefficients, respectively. The errors ϵ_i are normally distributed with mean 0 and variance σ^2 , that is $\epsilon_i \sim \text{Normal}(0, \sigma^2)$. The parameter σ^2 represents the variability with which the outcomes deviate from their predictions based on the model. This model may also be written as $y_i \sim \text{Normal}(\beta_0 + X_i\beta, \sigma^2)$, where all terms are the same as defined above. In our example, this equation translates to

$$\text{Valence}_i \sim \text{Normal}(\beta_0 + \beta_1 \text{Arousal}_i, \sigma^2). \quad (4)$$

The outcome Valence_i is linearly predicted by the combination of an overall mean β_0 , the intercept, and a coefficient (slope) β_1 multiplied by the value of the predictor variable Arousal_i . In the single-level regression model β_0 and β_1 do not vary by person. This model addresses the question: *on average, do people report feeling more pleasant if they report feeling more active?*

⁵https://osf.io/ebz2f/?view_only=3fe201fd40d34fbaf4a3e801c40df5e



***stan_glm* model fitting in four steps**

The `stan_glm` function in `rstanarm` can be used to easily fit Bayesian linear or generalized linear regression models. Models specified with `stan_glm` use syntax similar to the `lm` and `glm` functions in R, however `stan_glm` models are fit using MCMC instead of variants of least squares estimation.

Step 0. Install software and prepare data.

First install R and optionally RStudio, as a user friendly environment to use R (though code provided below can also be executed in R).⁶ Then install the necessary R packages, including `rstanarm` and accompanying packages for plotting and tidying output. The installations using the `install.packages` function are only required the first time you run the code.

```
install.packages(c("rstanarm",  
  "bayesplot", "ggplot2", "broom"))  
library("rstanarm")  
library("bayesplot")  
library("ggplot2")  
library("broom")
```

Next, set the working directory to where the provided example dataset “sampledata.csv” is located and load the data.

```
# Set directory and read in data  
setwd("~/work/RSTANARM/Tutorial")  
dat <- read.csv("sampledata.csv",  
  header=TRUE)
```

It is also useful to plot the data to check that there were no errors reading the external file. An example is shown in Figure 2. R code to reproduce all figures in the paper can be found in the accompanying R script on the OSF project page. Once the data is loaded, we are ready to specify the model.

Step 1. Specify the model

The `rstanarm` code for the single-level Bayesian regression from Equation 4, with default prior specification, is:

```
SingleLevelModel <- stan_glm(  
  valence ~ arousal, data = dat)
```

***stan_glm* syntax.** The first part of the call to `stan_glm` specifies the outcome and predictors in the familiar `lm` formula syntax, with the outcome `Valence` to the left of the ‘~’ symbol and the predictor `Arousal` on the right-hand

side. An intercept is included by default. Unless the ‘family’ argument is specified, `stan_glm` assumes that the likelihood is normal (Gaussian) with an identity link function.⁷

***stan_glm* priors.** All parameters in a Bayesian model need prior distributions that approximately describe the researcher’s uncertainty in the parameters before observing the data. Uninformative priors place equal or nearly equal prior weight on all possible values of a parameter. Weakly informative priors provide some information on the relative a priori plausibility of the possible parameter values, for example when we know enough about the variables in our model that we can essentially rule out extreme positive or negative values. Relative to uninformative priors, weakly informative priors can reduce our posterior uncertainty (which is desirable when warranted) and also help stabilize computations. If we have a lot of prior knowledge to bring to a problem, for instance from previous research, then we can use an informative prior.

The default priors in `rstanarm` are intended to be weakly informative and, in general, unless a lot of prior information is available, we recommend weakly informative priors for the parameters of a regression model. A weakly informative prior that reflects the expected magnitude of the parameters based on the scales of the variables will not strongly impact the posterior, but will provide regularization to stabilize computation and avoid overfitting, while still allowing for extreme values when warranted by the data (Gelman, Jakulin, Pittau, & Su, 2008; Stan Development Team, 2017). The weakly regularizing default priors currently used by `stan_glm` are tailored to each parameter type. In the Bayesian framework the priors are part of the model specification and should be reported. After fitting a model, the priors used in the analysis can be viewed using the command `prior_summary(SingleLevelModel)`:

```
Priors for model 'SingleLevelModel'  
-----  
Intercept (after predictors centered)  
  ~ normal(location = 0, scale = 10)  
  **adjusted scale = 212.68  
  
Coefficients  
  ~ normal(location = 0, scale = 2.5)  
  **adjusted scale = 3.29  
  
Auxiliary (sigma)  
  ~ half-cauchy(location = 0, scale = 5)
```

⁶See r-project.org and rstudio.com, respectively

⁷With `stan_glm`, like `glm`, the functional form of the likelihood and the link function can be changed using the ‘family’ argument. The default is equivalent to setting `family=gaussian(link='identity')`, which results in a linear regression. The `rstanarm` package vignettes and help pages have many examples of fitting models with different settings for the ‘family’ argument.

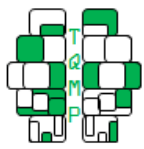
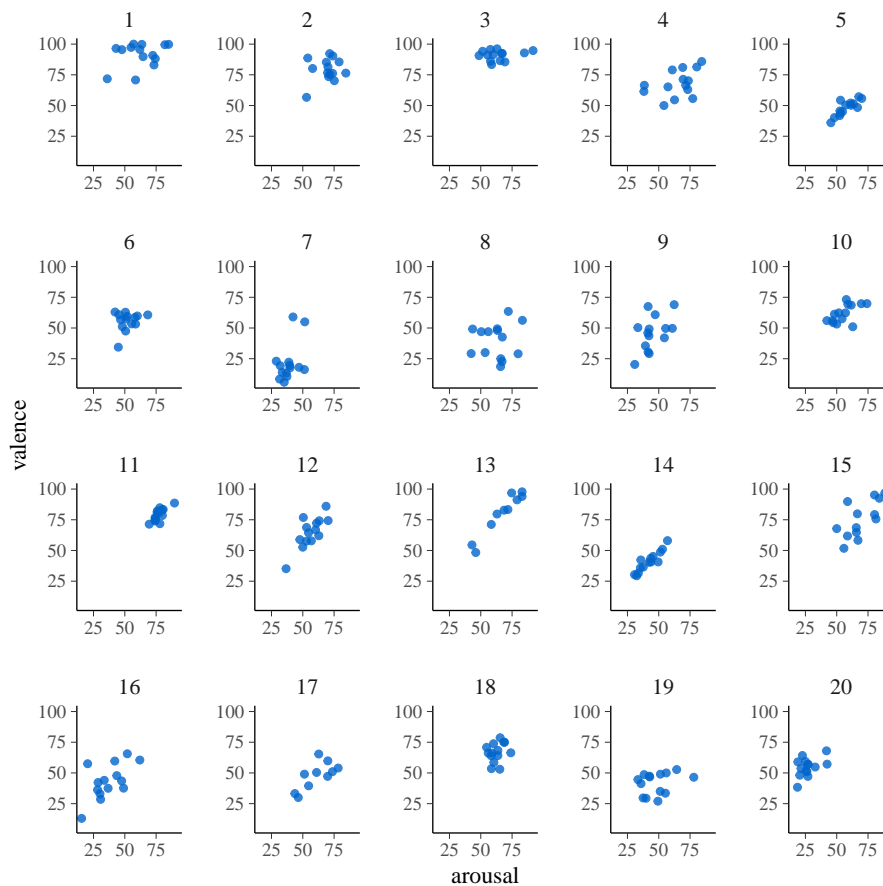


Figure 2 ■ Visual summary of the data, with each subplot corresponding to each person's observed data.



```

**adjusted scale = 106.34
-----
See help('prior_summary.stanreg') for
more details

```

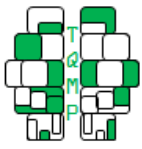
As priors are integral parts of Bayesian models, it is good practice to include information on the prior settings in research papers, based on the above output. We can see that normal distributions are set as priors for the regression parameters β_0 and β_1 , with the argument location corresponding to the mean, and the argument scale to the standard deviation. The default prior for the intercept has location 0 and standard deviation 10, but by default `rstanarm` rescales the standard deviation to ensure that a weakly informative prior is loosely based on the actual range of the outcome variable. In this case the actual prior

used for the intercept ends up being normal with mean 0 and standard deviation roughly 250. The prior standard deviation for the slope coefficient and the prior scale of the half-Cauchy prior on σ were also rescaled.⁸ If the user wants to set informative priors, the rescaling can be turned off and specific location and scale values can be specified, an example is provided below in the *Changing the default settings* section.

Step 2. Estimate the model parameters

To estimate the model parameters, we need to run the code from Step 1. By default, `rstanarm` uses four Markov chains with 2000 iterations each, half of which are discarded as “warm-up”. A warm-up sampling phase is used to give the algorithm time to find the target posterior area, and generally produces non-representative samples. If neces-

⁸For details on the exact computations involved and for how to turn off the default rescaling the user can consult `help("priors", package = "rstanarm")` and also the *Prior Distributions for rstanarm Models* vignette at <http://mc-stan.org/rstanarm/articles/index.html>. In the next release of `rstanarm` the default prior on sigma will be an exponential distribution rather than a half-Cauchy, which is also demonstrated in the vignette.



sary, these defaults may be modified using the `chains` and `iter` arguments. For situations in which the estimation algorithm does not converge (e.g., $\hat{R} > 1.1$) due to an insufficient number of samples, it may be necessary to draw more samples by increasing the `iter` argument. For some complex models, we may need to do more iterations than the default to get a sufficient number of effective samples (e.g., $n_{\text{eff}} > 1000$), which can be done also by increasing the number of iterations or the number of chains (`chains` argument). Increasing the number of chains to get more samples is especially efficient if these chains can be in run parallel.⁹ For our example, and in many cases, the default settings work well. When we execute the code from Step 1, the R console monitors the progress of sampling, and any errors or warnings regarding problems with the sampling will be displayed. If a warning about non-convergence is produced (e.g., 'Markov chains did not converge'), users can increase the number of iterations and re-fit the model. In general, models fit using MCMC may take longer than users are accustomed to; however the current example took around 1 second (the precise timing depends on the user's system). The single-level model is a simple model and runs quickly without warnings.

Changing the default settings. When using `stan_glm`, the prior for the intercept and regression coefficients can be changed for example via the `prior_intercept` and `prior` arguments, respectively. Specifying `autoscale=FALSE` in the call to `normal()` disables the default rescaling of the standard deviation parameters. This is most useful when we want to use an informative prior based on problem-specific knowledge. In the code below we give an example of setting a tighter prior on the intercept (with a smaller standard deviation) that is centered around the middle of the scale (50) and the slope coefficient is set to have a standard normal prior. When encountering convergence warnings, it is often sufficient to increase the number of iterations (e.g., from the default of 2000 to 4000), which can be done via the `iter` argument. In addition, `rstanarm` may suggest that the user increase the parameter controlling the sampler step size, which can be done via the `adapt_delta` argument, which accepts a proportion in (0,1) as its input (the default depends on the prior but it's typically 0.95). As a result of increasing `adapt_delta`, the MCMC algorithm will take smaller steps and be better able to maneuver through the posterior geometry. The following code shows an example of adjusting the prior specifications for the slope and intercept parameters, increasing the number of iterations, and

increasing `adapt_delta`:

Example of adapting priors and sampler settings

```
SingleLevelModelMod <- stan_glm(
  valence ~ arousal, data = dat,
  prior = normal(0,1, autoscale=FALSE),
  prior_intercept=normal(50,100,
    autoscale=FALSE),
  iter = 4000, adapt_delta = 0.99)
```

Step 3. Check sampling quality

After estimation, researchers should look for signs that the chains might not have converged and check that there is a large enough effective sample size for the analysis. The `summary` function provides both a summary of parameter estimates as well as diagnostic information about the sampling quality. The output consists of a short description of the analysis that was carried out followed by a summary of the parameter estimates (interpreted in Step 4) and the log-posterior, and finally three quantities related to the performance of the sampler: Monte Carlo standard error (MCSE), \hat{R} (\hat{R}^2) and effective sample size (n_{eff}).¹⁰

```
summarySingleLevelModel <- summary(
  SingleLevelModel)
print(summarySingleLevelModel)
```

The output of this command is seen in Listing 1.

Numerical checks. For numerical checks of sampling quality, we reference `rstanarm` summary tables, called via the `summary` function as illustrated above. The \hat{R}^2 and n_{eff} columns of the table show that \hat{R}^2 is less than 1.1 and the effective sample size is larger than 2000 for all parameters. This is good evidence in favor of convergence and typically a sufficiently large effective sample size.

Visual checks. Using the `SingleLevelModel` object we created to store the results from `stan_glm`, the code below draws the trace plot for β_1 :

```
plot(SingleLevelModel, "trace",
  pars = "Valence")
```

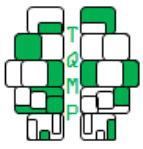
This is the plot displayed in Figure 1 (shown earlier), in which we can see that the chains seem to have converged to the same distribution.

Exploring the posterior with *shinystan*

The user can easily explore the posterior further through the `shinystan` R package, which provides a graphical user interface for interactively exploring `rstanarm` mod-

⁹The following lines can be executed for allowing parallel computation:
`rstan_options(auto_write = TRUE)`
`options(mc.cores = parallel::detectCores())`

¹⁰The log-posterior represents the logarithm of the prior times the likelihood, up to a constant. This value may interpreted, similar to likelihood values, for evaluating predictive accuracy

**Listing 1 ■** Output from the print summary command.

Model Info:

```

function:  stan_glm
family:    gaussian [identity]
formula:   valence ~ arousal
algorithm: sampling
priors:    see help('prior_summary')
sample:    4000 (posterior sample size)
num obs:   272

```

Estimates:

	mean	sd	2.5%	25%	50%	75%	97.5%
(Intercept)	14.7	3.7	7.4	12.2	14.7	17.1	22.0
arousal	0.8	0.1	0.7	0.8	0.8	0.8	0.9
sigma	16.9	0.7	15.6	16.4	16.9	17.4	18.4
mean_PPD	59.7	1.4	56.9	58.7	59.7	60.6	62.4
log-posterior	-1164.8	1.2	-1167.8	-1165.4	-1164.5	-1164.0	-1163.5

Diagnostics:

	mcse	Rhat	n_eff
(Intercept)	0.1	1.0	3687
arousal	0.0	1.0	3721
sigma	0.0	1.0	3932
mean_PPD	0.0	1.0	4000
log-posterior	0.0	1.0	1929

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on split chains (at convergence Rhat=1).

els (or any other models fit using MCMC). Visual and numeric checks are both available from *shinystan* via a user-friendly graphical user interface. With *shinystan* researchers can look at both estimates and diagnostics, and it is easy to customize exportable tables and graphics. *shinystan* provides a wide variety of tools for visualizing and summarizing the posterior distribution and diagnosing MCMC problems. Demonstrating the full capability of *shinystan* is not possible here because of space limitations, therefore here we only highlight its most important features. If the *shinystan* is installed and loaded, to open the interface in the default web browser simply run the following code.

```
launch_shinystan(SingleLevelModel)
```

The summary statistics shown earlier are displayed in an easily readable format in *shinystan*, as shown in Figure 3. The table can be exported in many useful formats.

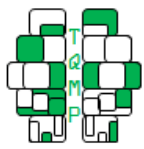
Figure 4 shows several visual assessments of the slope parameter. At the top there is a small table with numerical

summaries of the selected parameter. In the second row, the plot on the left shows the posterior distribution of the slope parameter (see more detail on a similar plot in Step 4). The plot on the right summarizes how autocorrelation among the samples at different lags decreases: the value is already rather low at lag 1, which means the effective sample size should be large. The plot under these two is a stretched out version of the trace plot in Figure 1.

Posterior predictive checks. To visually assess the fit of the model to the data, we can compare the observed data to datasets simulated according to our assumed data generating process and the posterior draws of the model parameters. The code below uses the `pp_check` function to plot a smoothed kernel density estimate of the original data, overlaying the density estimates from 100 generated data sets from the posterior predictive distribution:

```
pp_check(SingleLevelModel, nreps = 100)
+ xlab("valence")
```

The resulting plot is displayed in Figure 5, left side (the

**Figure 3** ■ Posterior summary statistics from `shinystan`.

The screenshot shows the `shinystan` web interface. At the top, there are tabs for `SHINYSTAN`, `DIAGNOSE`, `ESTIMATE`, `EXPLORE`, and `MORE`. Below these, there are buttons for `Parameters plot`, `Posterior summary statistics` (which is active), and `Generate LaTeX table`. On the left, there is a `Digits` input field set to `1`, and buttons for `Copy`, `Print`, `Download`, and `Column visibility`. On the right, there is a `Glossary` button and a `Regex searching` input field. The main table displays posterior summary statistics for the following parameters: (Intercept), arousal, sigma, mean_PPD, and log-posterior. The columns are: n_eff, Rhat, mean, mcse, sd, 2.5%, 25%, 50%, 75%, and 97.5%.

	n_eff	Rhat	mean	mcse	sd	2.5%	25%	50%	75%	97.5%
(Intercept)	3687	1	14.7	0.1	3.7	7.4	12.2	14.7	17.1	22
arousal	3721	1	0.8	0	0.1	0.7	0.8	0.8	0.8	0.9
sigma	3932	1	16.9	0	0.7	15.6	16.4	16.9	17.4	18.4
mean_PPD	4000	1	59.7	0	1.4	56.9	58.7	59.7	60.6	62.4
log-posterior	1929	1	-1164.8	0	1.2	-1167.8	-1165.4	-1164.5	-1164	-1163.5

same plot for the hierarchical model is displayed on the right, see later). Note that the second part of the code that reads as `xlab("valence")` labels the x-axis as “valence”. For models that fit the data well, this type of plot will show that the draws from the posterior predictive distribution (thin light blue lines) and the observed data (thick dark blue line) have similar distributions. In this case, it looks like the model does reasonably well (the dark blue line stays in the region designated by the light blue lines), but there is some room for improvement, especially for predicting valence values between 50 and 70. The `shinystan` package can also be used to perform various posterior predictive checks.

Step 4. Summarize and interpret results

After assessing convergence and sampling quality via diagnostics of \hat{R} , ESS, and MCSE, and checking the model fit with posterior predictive model, we can focus on interpreting the estimates. As shown before, these can be obtained via the summary function or the graphical user interface in the `shinystan` package. Table 1 shows a summary of these results.

Point estimates. A point estimate is a single value, most often a measure of central tendency (e.g., mean, median), chosen to summarize a posterior distribution. In contrast to the maximum likelihood approach, in which the point estimate (the mode) is the focus, in the Bayesian framework we are interested in the entire posterior distribution. A point estimate is merely one of many ways of summarizing the posterior, and rarely is it the most useful way.

Posterior intervals (PI) and posterior standard deviation (SD). Posterior intervals, also called posterior uncer-

tainty intervals or credible intervals, describe the likely range of the model parameters in probabilistic terms. The 2.5 % and 97.5 % columns in Table 1 show the bounds of the central 95% interval, representing the central 95% of the posterior probability distribution of the parameter. Given our observed data, chosen priors, and assumed data generating process (specified in the likelihood), we can say that the probability that the parameter value lies within the 95% PI is 95%. The posterior standard deviation (SD column in Table 1) is an alternative summary of the uncertainty around the point estimate.

We can see from Table 1 that the point estimate (in this case the mean of the posterior distribution) for the intercept (β_0) is 14.7, with 95 % PI = [7.4, 22.0]. The estimate for the slope parameter (β_1) indicates that a one-unit change in self-reported arousal values is associated with an increase of 0.8 (95 % PI = [0.7, 0.9]) in self-reported valence values (recall that both variables are on a 0-100 scale). Looking back at Figure 4, the middle plot of the (smoothed) posterior probability distribution of the slope shows that all of the probability mass for the slope parameter is on positive values, suggesting that arousal level is remarkably associated with valence. This means that people reporting higher levels of arousal are expected to have higher levels of valence, or put it another way people who report feeling more active also tend to report feeling more pleasant.

It can sometimes be useful to extract the posterior draws from the fitted model object. For this purpose `rstanarm` provides `as.matrix`, `as.data.frame`, and `as.array` methods that return the draws in various formats. For example, if we wanted to estimate the proba-

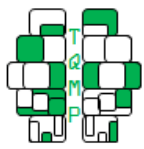


Figure 4 ■ Visual and numerical summary of the slope parameter from `shinystan`. Comprehensive visual diagnostics include: table of numerical summaries (top), posterior distribution of parameter (middle left), autocorrelation among parameter samples (middle right), and trace plot of sample chains (bottom).

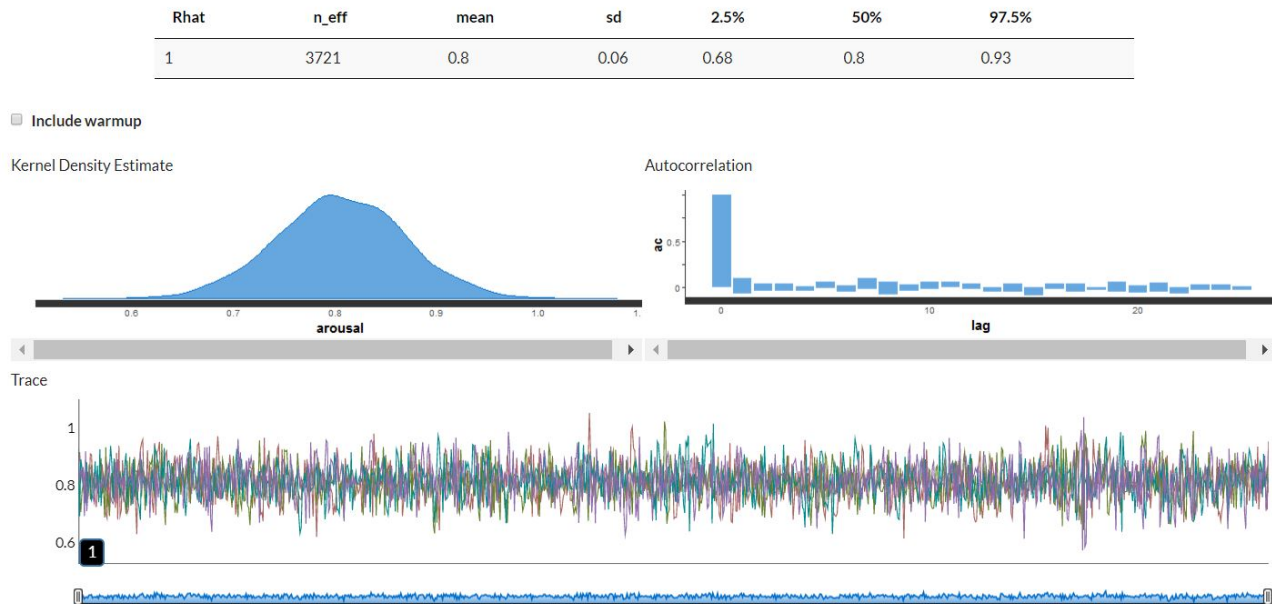


Table 1 ■ Diagnostics and posterior summary statistics of the estimated parameters from the single-level model.

Parameter	\hat{R}	ESS	mean	SD	MCSE	2.5%	97.5%
Intercept (β_0)	1.0	3687	14.7	3.7	0.1	7.4	22.0
Slope (β_1)	1.0	3721	0.8	0.1	0.0	0.7	0.9
Error SD (σ)	1.0	3932	16.9	0.7	0.0	15.6	18.4

bility that the slope (arousal) is greater than some value x we can do this by computing the proportion of draws that are greater than x :

```
posteriorSamples <- as.matrix(  
  singleLevelModel, pars = "arousal")  
mean(posteriorSamples > 0.7)
```

which will return 0.956.

This result tells us that the posterior probability that the slope coefficient is larger than 0.7 is about 96%. By this we illustrate how we can answer questions about parameters or functions of parameters by working directly with the posterior sample, although much of the post-estimation functionality provided by `rstanarm` circumvents the need for the researcher to directly manipulate the draws.

The plot in Figure 6 illustrates the model's prediction of how valence and arousal levels are linked: the blue dots

are the data points, the black regression line is based on the posterior mean estimate of the intercept and slope parameters, and the blue regression lines are computed from a random subset of the posterior draws of these two parameters. The black line shows the positive association between valence and arousal ($\beta_1 \approx 0.8$), while the blue lines illustrate how much uncertainty there is around this average regression line. The spread of the blue lines is somewhat wider at the left of the graph than in the middle, which means that we are a bit less certain about the predictions of valence at low levels of arousal than at medium levels of arousal. The main reason for this is that we have fewer data points that represent the relationship between low arousal values and valence. With more observations at medium arousal levels, the spread (uncertainty) decreases in the middle region of the graph.

Overall, the single-level model captures the positive linear relation between people's valence and arousal scores

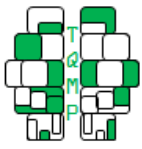
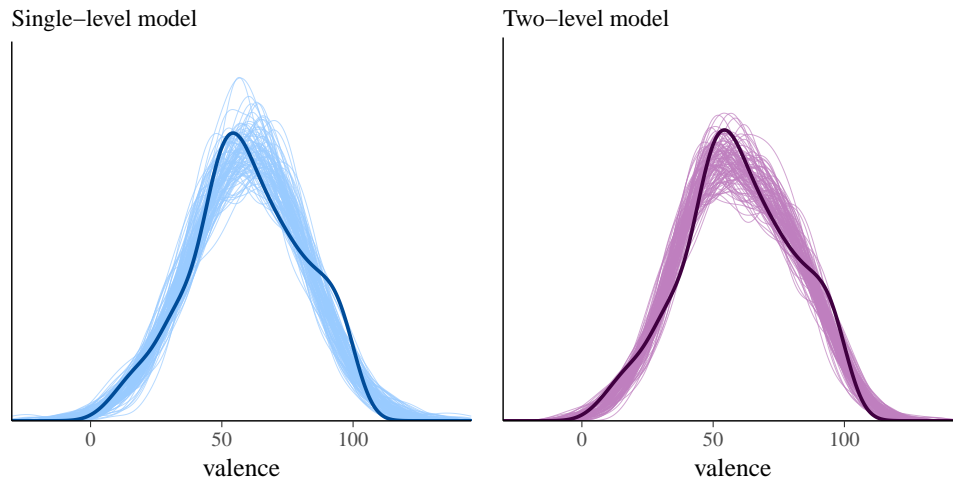


Figure 5 ■ Graphical posterior predictive check comparing the observed distribution of valence scores (dark blue line) to 100 simulated datasets from the posterior predictive distribution (light blue lines), in the single level (left side) and in the hierarchical (right side) models.



well, but there is room for improvement. We are mostly skeptical about the performance of this model because it does not account for the fact that observations are nested in persons. Next we move to hierarchical Bayesian regression, which allows us to better handle this type of nested data.

Fitting a Bayesian two-level model to predict pleasantness from arousal

A hierarchical linear model (HLM) is an extension of single-level linear regression that allows for coefficients to vary by grouping variables. For example, in our case an HLM will let us account for how the link between arousal and time of day varies by person. We introduce Bayesian HLMs by first considering the nature of our data, model, and estimation method in the hierarchical context.

Nested data and nested models. Social scientists often work with nested data, which is to say data that have some form of relevant hierarchical structure. For instance, consider a data set from subjects who report their sleeping hours repeatedly over a couple of days, or subjects who complete several test items measuring the same underlying construct such as the reading ability of first grade students. In the latter example, the repeated measures are nested within students, while students could be nested in a higher-order group like classrooms or schools, which can in turn be nested in school districts or states, and so on. In many cases, it can be important to account for this type of structure when we develop a model because data with an underlying nested structure are not exchangeable at the

individual measurement level (Gelman & Hill, 2007). If we expect observations from the same group to be similar to some extent, then that structure should be encoded in the model (Raudenbush & Bryk, 2002). When a hierarchical/multilevel model is formulated for this type of nested data, it both (1) accounts for correlation among observations within the same group, and (2) specifies relations between groups (Littell, Milliken, Stroup, Wolfinger, & Schabenberger, 2006).

In the experience sampling study on affective states, groups (higher level units) correspond to participants, because repeated measures are grouped within person. Moving from a single-level regression model to a hierarchical model is a straightforward extension: the single-level regression specification will be used for the observations of the outcome variable y within a person, and we will add a hierarchical structure to tie the person-specific regression coefficients together with a joint distribution at level-2. Let us denote by y_{ij} the outcome for measure $i = 1, \dots, I_j$ for person $j = 1, \dots, J$, with person j having I_j total observations. For a single predictor x , the model can be written as:

$$y_{ij} \sim \text{Normal}(\beta_{0j} + \beta_{1j}X_{ij}, \sigma^2), \quad (5)$$

$$\begin{bmatrix} \beta_{0j} \\ \beta_{1j} \end{bmatrix} \sim \text{Normal}\left(\begin{bmatrix} \mu_{\beta_0} \\ \mu_{\beta_1} \end{bmatrix}, \begin{bmatrix} \sigma_{\beta_0}^2 & \rho\sigma_{\beta_0}\sigma_{\beta_1} \\ \rho\sigma_{\beta_0}\sigma_{\beta_1} & \sigma_{\beta_1}^2 \end{bmatrix}\right). \quad (6)$$

At level-2, Equation 6, the person-specific parameters are assigned a joint distribution with population-level hyperparameters (i.e., parameters of the prior distribution for

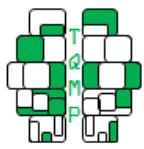
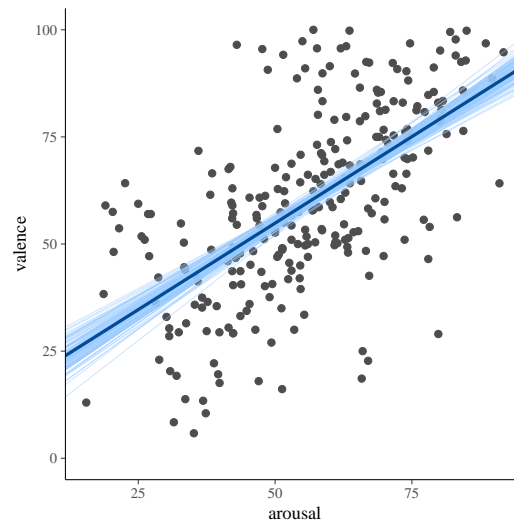


Figure 6 ■ Observed data (blue dots) and average predicted linear relation between arousal and time of day (dark line). Uncertainty is visualized by also plotting the regression lines computed from 100 of the draws from the posterior distribution (thin lines).



the person-specific parameters): a bivariate normal distribution with means ($\mu_{\beta_0}, \mu_{\beta_1}$), variances ($\sigma_{\beta_0}^2, \sigma_{\beta_1}^2$), and correlation (ρ) that are common across all groups at the population level. In terms of our applied example, the two-level model, the measurement level of the HLM with person-level grouping, Equation 5, can be written as

$$\text{Valence}_{ij} \sim \text{Normal}(\beta_{0j} + \beta_{1j}\text{Arousal}_{ij}, \sigma^2), \quad (7)$$

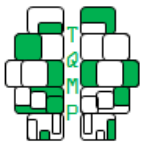
which expresses that we are predicting average levels of valence from average levels of arousal, with measurements nested in persons. The errors at this level now represent within-person variation *Valence* unexplained by *Arousal*. At the second level of the model, which is identical to Equation 6, the variance parameters represent between-person variation and the correlation parameter is the correlation between person-specific slopes and intercepts in the population. Various additional structures of interest for the reader's particular regression applications (e.g., multivariate modeling, latent variable assessment, item response theory; Carpenter et al., 2017; Kruschke, 2015) can already be flexibly specified with *rstanarm* or will become available in future releases.¹¹

Bayesian HLM advantages. Hierarchical models allow us to incorporate more information than single-level regression by explicitly accounting for both within- and between-group structure and variation. The nested structure of

HLMs leads to partial pooling, which is a compromise between the extremes of complete pooling (one single-level regression model combining data from all groups) and no pooling (a separate regression for each group). Partial pooling occurs when the estimates for group-specific parameters inform the shared population-level parameters, and vice versa. Estimates for groups with small sample sizes and a lot of variability are affected most by partial pooling; they are shrunk toward the population mean to a greater degree than are the estimates for larger groups with less variation. This feature of partial pooling helps buffer outlier effects and avoid overfitting, that is, it helps avoid learning characteristics of the sample that are artifacts of measurement noise. In the Bayesian framework when a prior itself depends on higher-order priors, the higher-order prior is referred to as a regularizing prior (McElreath, 2016). Making distributional assumptions about parameters (priors) allows for optimal hierarchical pooling across groups. Group-specific estimates are shrunk more towards population estimates after learning from the prior distribution from other groups via shared hyperparameters. Being able to set prior distributions on the population parameters provides additional shrinkage, this way improving parameter estimation.

Shrinkage is most beneficial for small samples, and simulation studies show that small sample models provide

¹¹There are currently 11 vignettes at <http://mc-stan.org/rstanarm/articles/index.html> that demonstrate how to fit the various models included in *rstanarm*.



more robust estimates in the Bayesian framework than when estimated with frequentist estimation techniques (Kruschke, 2015), as they fully account for the uncertainty at each level, this way improving the accuracy of estimates and predictions. By modeling all parameters in a probabilistic framework, full Bayesian inference appropriately propagates our uncertainty through each level of the model and the resulting posterior distributions give a full picture of our uncertainty in each parameter. In contrast, maximum marginal likelihood approaches to estimating these models often underestimate standard errors for the regression coefficients because they only condition on a point estimate of the so-called “random effects” parameters rather than a full probability distribution.

Fitting a two-level hierarchical model with stan_lmer in four steps

The `stan_lmer` function in `rstanarm` can be used to estimate linear models with intercepts and slopes that are allowed to vary across groups (Gabry & Goodrich, 2017a). Models specified with `stan_lmer` use syntax similar to the `lmer` function in R’s `lme4` package, however `rstanarm` uses Stan’s MCMC algorithm to draw from the posterior distribution of the model parameters rather than estimating them via the frequentist approach taken by `lme4`.

Step 0. Prepare data

We assume here that the necessary software is already installed. We load the packages and data the same way as for the previous example but this time we will use an extra variable that is in the same dataset, which specifies the within-person nesting structure (person identification, PID, variable). This step is identical to the one described for the single-level model, therefore not repeated here.

Step 1. Specify likelihood and prior distributions

The two-level model is specified in Equations 5 and 6 (with variable names explicitly stated in Equation 7). Using the same dataset as for the previous model, we specify the two-level model as follows:

```
TwoLevelModel <- stan_lmer(valence ~
  arousal + (1 + arousal | PID), data=
  dat)
```

stan_lmer syntax. The first part of the `stan_lmer` function specifies a model formula that indicates the outcome variable, predictors, and which coefficients are allowed to vary according to levels of the predictors. For `stan_lmer` a normal (Gaussian) likelihood is the only

option and is implied.¹² In the model formula, in addition to the single-level formula `valence ~ arousal` we now also have a second component `(1 + arousal | PID)` that specifies that both the intercept and the coefficient on the `arousal` variable are to vary by person, that is, by level of the `PID` variable. The model formula for `stan_lmer` is written using the exact same syntax as the formula for the `lmer` function from the `lme4` package (Bates et al., 2015) that does maximum marginal likelihood inference.

stan_lmer priors. To complete the Bayesian model specification, we must specify priors for all model parameters. For the two-level model this means priors for the varying regression parameters (β_{0j} , β_{1j}), the non-varying regression parameters (μ_{β_0} , μ_{β_1}), the covariance matrix Σ , and the measurement level error variance σ^2 . Equation 6 serves as the prior for the varying regression parameters, in our case the person-specific intercept and slope parameters β_{0j} and β_{1j} : this formulation specifies that our uncertainty in the parameters β_{0j} and β_{1j} is described by a bivariate normal distribution with hyperparameters μ_{β_0} , μ_{β_1} , and Σ , the population means and covariance matrix.

Like `stan_glm`, by default `stan_lmer` sets weakly informative priors if the user does not specify otherwise. The default weakly informative priors for the parameters in Equation 7 can be viewed by running the `prior_summary(TwoLevelModel)` command.

```
Priors for model 'TwoLevelModel'
-----
Intercept (after predictors centered)
  ~ normal(location = 0, scale = 10)
  **adjusted scale = 212.68

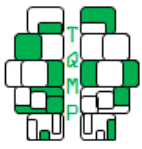
Coefficients
  ~ normal(location = 0, scale = 2.5)
  **adjusted scale = 3.29

Auxiliary (sigma)
  ~ half-cauchy(location = 0, scale = 5)
  **adjusted scale = 106.34

Covariance
  ~ decov(reg. = 1, conc. = 1, shape = 1,
  scale = 1)
-----
See help('prior_summary.stanreg') for
more details
```

Similarly to the output for the single-level model, the default and adjusted priors are reported for the intercept

¹² `rstanarm` also provides the `stan_glm` function with additional options for the form of the likelihood and link function for multilevel generalized linear models.



and coefficients. For the hierarchical model, a prior must also be specified for the level-two covariance matrix of the person-specific regression terms. To reiterate with our example, the covariance matrix shown in Equation 6 quantifies between-person (population-level) variation in the person-specific intercept ($\sigma_{\beta_0}^2$) and slope ($\sigma_{\beta_1}^2$) along the diagonal, and their covariation ($\rho\sigma_{\beta_0}\sigma_{\beta_1}$) in the off-diagonals. `rstanarm` uses a prior on covariance matrices that has been developed to be robust for common applied regression problems. Here we note the unique value of the covariation between intercept and slope: this measure describes how average starting values relate to average trajectories, which may be illustrative for person-specific inference. For example, a negative covariation between person-specific intercepts and person-specific slopes suggests that if a person has a low valence on average, its level will likely be strongly associated with arousal levels. The covariance matrix is decomposed into a vector of variances ($\sigma_{\beta_0}^2, \sigma_{\beta_1}^2$) and a correlation matrix. The correlation matrix summarizes the covariation of the slopes and intercepts in a standardized form and allows us to specify a general setting for these parameters that does not depend on the size of the variance parameters. For the variances themselves, `rstanarm` decomposes them into the product of the trace of the implied covariance matrix (representing the total variance at this level of the model) and a simplex vector (representing the proportion of that total variance attributable to each of the variables). The trace itself is then set equal to the product of the square of an estimated scale parameter and the known order of the matrix. The last line of the output above shows the default settings used for this prior (the `decov` prior). A more thorough description of this prior is beyond the scope of this tutorial but the reader can consult the `rstanarm` package vignettes (Gabry & Goodrich, 2017a) for full details. Modifying this prior on the covariance structure can be done using the `prior_covariance` argument to `stan_lmer`, but the default settings result in a moderately regularizing prior that works very well in most settings. We recommend keeping the defaults for `prior_covariance` when using `stan_lmer` unless the user is comfortable with the explanation of this prior in the documentation and vignettes and has reason to believe that the default settings are inappropriate for the particular application at hand, which should be somewhat rare.

Step 2. Estimate model parameters

To estimate the two-level Bayesian HLM, simply run code from Step 1 above, which will create the object `TwoLevelModel` in the R global environment. Again, the sampling progress will be printed to the console and any errors or warnings from Stan will be displayed, which may

recommend increasing the number of iterations or other actions if necessary. Due to the increased complexity of the two-level model compared to the single-level model, the sampling process will be more computationally demanding and the run time will be a bit longer. We expect less than 5 minutes total, although the precise timing depends on the user's hardware.

Step 3. Check sampling quality

Numerical checks. Before assessing the results of the estimation, we assess sampling quality. The two-level model uses `rstanarm`'s default settings for MCMC and runs without warnings. For numerical checks of sampling quality, we call the summary function (note that the output reported below is truncated to include person-specific estimates for only the first two and the last participants; below we only show a few lines.). As for the single-level model, additional summary tables are accessible through `shinystan`'s following command `summary(TwoLevelModel)` as well. The results of this command are seen in Listing 2.

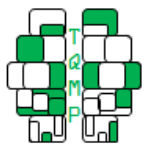
Again there are no red flags in the summary output for the two-level model. We see low \hat{R} values (close to 1), large effective sample sizes, and Monte Carlo standard errors that are much smaller than the posterior standard deviations.

Visual checks. By using the `plot` function we can again create trace plots that visualize the Markov chains in the style of time series plots for a few of the model parameters (example provided in the online R script). We also encourage the user to call `shinystan` for various options for convergence plots. For the two-level model the trace plots of the MCMC chains (not shown here) exhibited good mixing and show no signs of convergence problems. The numerical and visual convergence diagnostics look satisfactory for this model.

Posterior predictive checks. We can again draw 100 samples from the posterior predictive distribution so that we can compare the predictions from the two-level model against the actual data.

```
pp_check(TwoLevelModel, nreps=100) +  
  xlab("Arousal")
```

Results are shown in the right panel of Figure 5. Both plots depict the original data and smoothed kernels based on 100 generated data sets (light purple lines) from the posterior predictive distribution. Notice that when we move from the plot for the single-level model to the plot for the two-level model the small white gaps shrink and the lines better match the height of the smoothed data. Many other graphical posterior predictive checks are also available via the `pp_check` function.

**Listing 2 ■** Output from the summary command.

Model Info:

```

function:  stan_lmer
family:    gaussian [identity]
formula:   valence ~ arousal + (1 + arousal | PID)
algorithm: sampling
priors:    see help('prior_summary')
sample:    4000 (posterior sample size)
num obs:   272
groups:    PID (20)

```

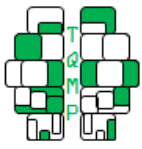
Estimates:

	mean	sd	2.5%	25%	50%	75%	97.5%
(Intercept)	29.8	5.8	18.1	26.1	30.0	33.5	41.2
arousal	0.5	0.1	0.4	0.5	0.5	0.6	0.7
b[(Intercept) PID:1]	37.5	11.0	17.2	29.9	37.1	44.7	60.3
b[arousal PID:1]	-0.2	0.2	-0.5	-0.3	-0.2	-0.1	0.1
b[(Intercept) PID:2]	18.1	13.1	-6.4	9.2	17.4	26.3	45.5
b[arousal PID:2]	-0.1	0.2	-0.5	-0.2	-0.1	0.0	0.2
...							
b[(Intercept) PID:20]	11.6	7.3	-2.3	6.8	11.5	16.2	26.3
b[arousal PID:20]	-0.1	0.2	-0.4	-0.2	-0.1	0.1	0.3
sigma	9.3	0.4	8.5	9.0	9.3	9.6	10.2
Sigma[PID:(Intercept), (Intercept)]	420.2	200.6	140.9	281.1	382.3	519.3	897.2
Sigma[PID:arousal, (Intercept)]	-3.2	2.5	-9.4	-4.4	-2.7	-1.4	0.2
Sigma[PID:arousal, arousal]	0.1	0.0	0.0	0.0	0.0	0.1	0.2
mean_PPD	59.7	0.8	58.2	59.2	59.7	60.2	61.2
log-posterior	-1075.4	7.0	-1090.2	-1079.9	-1074.8	-1070.4	-1063.2

Diagnostics:

	mcse	Rhat	n_eff
(Intercept)	0.2	1.0	1442
arousal	0.0	1.0	2364
b[(Intercept) PID:1]	0.2	1.0	4000
b[arousal PID:1]	0.0	1.0	4000
b[(Intercept) PID:2]	0.2	1.0	4000
b[arousal PID:2]	0.0	1.0	4000
...			
b[(Intercept) PID:20]	0.1	1.0	2372
b[arousal PID:20]	0.0	1.0	4000
sigma	0.0	1.0	4000
Sigma[PID:(Intercept), (Intercept)]	5.4	1.0	1360
Sigma[PID:arousal, (Intercept)]	0.1	1.0	1168
Sigma[PID:arousal, arousal]	0.0	1.0	1121
mean_PPD	0.0	1.0	4000
log-posterior	0.3	1.0	710

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on split chains (at convergence Rhat=1).



Step 4. Summarize and interpret results

Model parameter estimates. Compared to the single-level model, the two-level model has many more parameters because we have slopes and intercepts for each person in the sample. When dealing with such a large output, summary tables can be tailored to extract only the estimates of interest to the researcher. This can be done with the summary method defined in `rstanarm` itself and/or using the tidy function in the broom package, which now has functionality specifically for `rstanarm` models and makes it easy to group the estimates by parameter type or as before. The `shinystan` package may be called to interactively explore the `rstanarm` model. In Listing 3, we report the three levels of tidy summaries (to conserve space, note that the person-specific tidy summary is truncated to six lines).

Table 2 shows parameter estimates from the two-level model, based on a table generated by `shinystan`. Note that `shinystan` summarizes the level-two spread estimates in terms of variances and covariances, while the output above uses standard deviations and correlation. In terms of our research questions, we consider relations of valence and arousal (both measured on a continuous scale between 0 and 100) at the person-specific and population levels. The population intercept (μ_{β_0}) is 29.9 (95% PI = [18.7, 41.0]) and the person-specific intercepts vary around this mean with variance ($\sigma_{\beta_0}^2$) of 425.6. At the population level a one-unit change in arousal is associated with a .5 increase (μ_{β_1}) in valence (95% PI = [0.4, 0.7]), which is slightly smaller in magnitude than the point estimate of the slope from model 1 ($\beta_1=0.8$), and there is some between-person variation in these values ($\sigma_{\beta_1}^2 = 0.1$). Lastly, the two-level model captures negative covariation between person-specific intercept and person-specific slopes ($\rho\sigma_{\beta_0}\sigma_{\beta_1} = -3.3$), meaning that people with lower levels of valence, on average, have higher associations between valence and arousal, on average.

Compared to single-level model, the two-level model has a lower standard deviation (σ) due to its ability to account for multiple levels of information. The level-1 standard deviation quantifies the variability with which the outcomes deviate from the model's predictions, and its posterior mean was 16.9 in the single-level model and has decreased to 9.3 in the two-level model (its posterior standard deviation dropped from 0.7 to 0.4). The decrease in standard deviation reflects the additional information accounted for in the nested, 2-level structure.

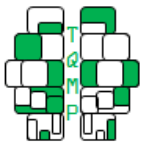
Moreover, the two-level model's population-level variance estimates help us learn the degree to which there is variation between people. Figure 7 illustrates the variation in the estimated valence-arousal relations across the

20 individuals in the sample. Here the points are the observed data for each person, the dark blue lines are posterior mean estimates of the person-specific regression lines, and the light blue lines are based on a random sample of 100 of the posterior draws. The dashed line is the mean regression line from the single-level model and is the same in each panel. The light blue lines help convey the uncertainty we have surrounding the person-specific estimates. The spread of the lines is different for each person because our uncertainty in a person-specific prediction depends on the sample size and variability in that person's data. For instance, because most of the observations for Participant 14 are concentrated at arousal levels around 45, we have very little uncertainty in our predictions for Participant 7 in that range and much greater uncertainty at lower or higher levels of arousal. For other participants (e.g., Participant 13) the pattern is reversed and we have less uncertainty at larger values of arousal.

Overall, both the single-level model and the two-level model capture a positive relation between valence and arousal but only the two-level model adequately accounts for the information contained in the person-specific variation. Consequently, with the two-level model, we can more precisely describe how the average person reports feeling more pleasant when they report greater activity, as well as how individuals tend to differ from that average. And despite the increase in complexity, nuanced posterior inference from the Bayesian HLM is nearly as straightforward as it is for the single-level model.

Discussion

In this tutorial we introduced the R package `rstanarm`, which is a user-friendly tool to fit single-level and hierarchical Bayesian regression models. Bayesian methods are becoming more and more popular for statistical inference, and for communicating current software developments to applied researchers. Users familiar with R will find it much easier to use the `rstanarm` package, compared to Stan's original R interface `RStan`, because the latter presumes knowledge of Stan's unique modeling language while `rstanarm` uses similar syntax as R's popular functions `glm` and `lmer`. Switching directly to Stan as a modeling language may be a barrier for many researchers, because its syntax may be unintuitive for those without a coding background. Thus, `rstanarm` serves as a valuable stepping-stone for researchers to enter the realm of Bayesian inference while still being able to use familiar software and syntax tools. As an added bonus, `rstanarm` works with `shinystan`, which provides rich opportunities of assessing model convergence, fit, and results. We also note that alternative to `rstanarm` is `brsm` (Burkner, 2017), which also uses simplified syntax to carry


Listing 3 ■ Commands to obtain three levels tidy summaries and their results.

Population-level estimates

```
summaryTwoLevelModelPop <- tidy(TwoLevelModel, intervals=TRUE, prob=.95,
                                parameters = "non-varying")
print(summaryTwoLevelModelPop, digits = 2)
```

	term	estimate	std.error	lower	upper
1	(Intercept)	29.85	5.341	18.72	41.0
2	arousal	0.54	0.077	0.39	0.7

Variance estimates

```
summaryTwoLevelModelVar<- tidy(twoLevelModel, intervals=TRUE, prob=.95,
                                parameters = "hierarchical")
print(summaryTwoLevelModelVar, digits = 2)
```

	term	group	estimate
1	sd_(Intercept).PID	PID	20.63
2	sd_arousal.PID	PID	0.24
3	cor_(Intercept).arousal.PID	PID	-0.65
4	sd_Observation.Residual	Residual	9.29

Person-specific estimates

```
summaryTwoLevelModelPerson <- tidy(TwoLevelModel, intervals=TRUE, prob=.95,
                                    parameters = "varying")
print(summaryTwoLevelModelPerson, digits = 2)
```

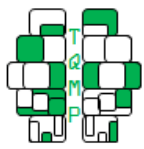
	level	group	term	estimate	std.error	lower	upper
1	1	PID	(Intercept)	36.752	10.87	17.611	59.51
2	1	PID	arousal	-0.161	0.17	-0.500	0.14
3	2	PID	(Intercept)	16.860	12.24	-5.952	45.72
4	2	PID	arousal	-0.077	0.17	-0.479	0.25
...							
39	20	PID	(Intercept)	11.504	7.09	-2.376	26.22
40	20	PID	arousal	-0.047	0.16	-0.422	0.30

out Bayesian parameter estimation in Stan.

Using the `rstanarm` package, we presented Bayesian estimation of single-level and two-level hierarchical linear regression modeling and we used simple applied examples to highlight some of the advantages of multilevel modeling. In terms of broader impact, we hope that this tutorial will encourage researchers to use Bayesian modeling in their work and help facilitate the transition from software packages implementing frequentist methods to new packages like `rstanarm` that make applied Bayesian regression modeling more accessible than ever before. While model specification in `rstanarm` is straightforward, we caution the reader to always carefully consider how the model specifications translate to the applied problem to avoid mechanically fitting models to data.

For readers new to Bayesian methodology, learning to

work in this unfamiliar statistical framework entails learning a new set of foundational principles that pertain to model specification, estimation, and inference. We think this fundamental shift is worthwhile for many reasons that include: the introduction of prior distributions allows for valuable structure to be incorporated at each level of a model and regularization helps manage extreme estimates and avoid overfitting; the posterior distribution is a complete representation of our uncertainty in the model parameters, given the data and modeling assumptions; the posterior predictive distribution is easy to work with and correctly propagates parameter uncertainty into predictions; and the language with which we can describe our inferences is intuitive and permits probability statements about parameters of interest. Finally, in addition to learning new concepts, in order to begin using Bayesian meth-

**Table 2** ■ Level-2 parameter estimates from the two-level model.

Parameter	\hat{R}	ESS	mean	SD	MCSE	2.5%	97.5%
Intercept (μ_{β_0})	1.0	2056	29.9	5.6	0.1	18.7	41.0
Slope (μ_{β_1})	1.0	2809	0.5	0.1	0.0	0.4	0.7
Error SD (σ)	1.0	4000	9.3	0.4	0.0	8.5	10.2
Between-person VAR Intercept ($\sigma_{\beta_0}^2$)	1.0	2094	425.6	207.6	4.5	140.6	927.1
Covariance Intercept-Slope ($\rho\sigma_{\beta_0}\sigma_{\beta_1}$)	1.0	1719	-3.3	2.6	0.1	-9.3	0.2
Between-person VAR Slope ($\sigma_{\beta_1}^2$)	1.0	1495	0.1	0.0	0.0	0.0	0.2

ods a researcher must also become familiar with a new set of software tools. For R users, `rstanarm` makes the transition straightforward.¹³

References

- Arbuckle, J. L. (1999). Amos 4.0 [computer software]. Chicago: Smallwaters, 1.
- Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1), 1–48. doi:10.18637/jss.v067.i01
- Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. Retrieved from <https://arxiv.org/abs/1701.02434>
- Burkner, P.-C. (2017). Brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1–28. doi:10.18637/jss.v080.i01
- Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., ... Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1), 1–32. doi:10.18637/jss.v076.i01
- Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2013). *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge.
- Csikszentmihalyi, M. & Larson, R. (1987). Validity and reliability of the experience sampling method. *The Journal of Nervous and Mental Disease*, 175, 526–536.
- Gabry, J. (2017). shinystan: interactive visual and numerical diagnostics and posterior analysis for Bayesian models. R package version 2.4.0. Retrieved from <http://mc-stan.org/shinystan/>
- Gabry, J. & Goodrich, B. (2017a). Estimating generalized linear models with group-specific terms with rstanarm. Retrieved from <http://mc-stan.org/rstanarm/articles/glmer.html>
- Gabry, J. & Goodrich, B. (2017b). rstanarm: Bayesian applied regression modeling via Stan. R package version 2.15.3. Retrieved from <http://mc-stan.org/rstanarm/>
- Gabry, J. & Mahr, T. (2017). *Bayesplot: plotting for Bayesian models*. R package version 1.4.0. Retrieved from <http://mc-stan.org/bayesplot>
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis, Third edition*. Boca Raton (FL): Chapman & Hall/CRC.
- Gelman, A. & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge: Cambridge University Press.
- Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *Ann. Appl. Stat.* 2(4), 1360–1383. doi:10.1214/08-AOAS191
- Hoffman, M. D. & Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1), 1593–1623. Retrieved from <http://dl.acm.org/citation.cfm?id=2627435.2638586>
- JASP Team. (2017). JASP (Version 0.8.5)[Computer software]. Retrieved from <https://jasp-stats.org/>
- Korner-Nievergelt, F., Roth, T., von Felten, S., Guélat, J., Almasi, B., & Korner-Nievergelt, P. (2015). *Bayesian data analysis in ecology using linear models with R, BUGS, and Stan*. Academic Press.
- Kruschke, J. (2015). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.
- Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., & Schabenberger, O. (2006). *SAS for mixed models 2nd ed.* Cary, NC: SAS Institute.
- McElreath, R. (2016). *Statistical rethinking: a Bayesian course with examples in R and Stan*. CRC Press.
- Metropolis, N. & Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247), 335–341. doi:10.1080/01621459.1949.10483310
- Muthén, L. K. & Muthén, B. O. (2008). Mplus (version 5.1). Los Angeles, CA: Muthén & Muthén.
- Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In

¹³For questions about `rstanarm` or Stan in general visit the Stan Forums at <http://discourse.mc-stan.org>.

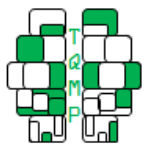
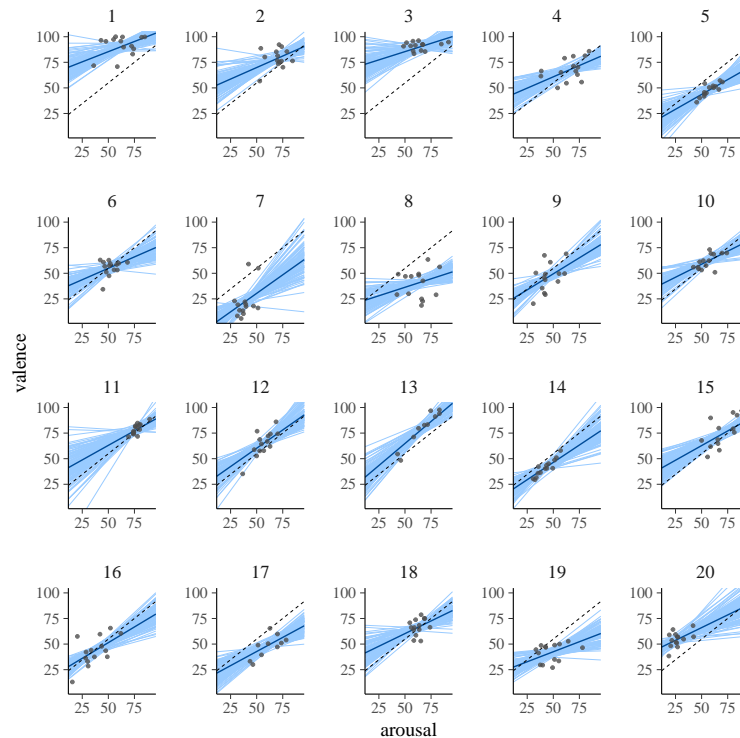


Figure 7 ■ Graphical illustration of the person-specific results: observed data (dots), posterior mean regression lines (dark lines), and uncertainty (light lines).



Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003) (pp. 20–22).

Raudenbush, S. W. & Bryk, A. S. (2002). *Hierarchical linear models: applications and data analysis methods*. Newbury Park, CA: Sage.

Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological Review*, 110, 145–172.

Sorensen, T., Hohenstein, S., & Vasishth, S. (2016). Bayesian linear mixed models using Stan: A tutorial for psy-

chologists, linguists, and cognitive scientists. *The Quantitative Methods for Psychology*, 12(3), 175–200. doi:10.20982/tqmp.12.3.p175

Spiegelhalter, D., Thomas, A., Best, N., & Gilks, W. (1996). BUGS 0.5: bayesian inference using Gibbs sampling manual (version ii). *MRC Biostatistics Unit, Institute of Public Health, Cambridge, UK*, 1–59.

Stan Development Team. (2017). Stan modeling language users guide and reference manual, version 2.17.0. Retrieved from <http://mc-stan.org/>

Citation

Muth, C., Oravecz, Z., & Gabry, J. (2018). User-friendly Bayesian regression modeling: A tutorial with rstanarm and shinystan. *The Quantitative Methods for Psychology*, 14(2), 99–119. doi:10.20982/tqmp.14.2.p099

Copyright © 2018, Muth, Oravecz, and Gabry. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Received: 21/12/2017 ~ Accepted: 11/03/2018