Estimating Ground-Motion Models via Bayesian Inference using Stan

Nicolas M. Kuehn*[1] and Peter J. Stafford†[2]

[1]*University of California, Los Angeles*

[2]*Imperial College London*

Abstract

We provide a simple introduction to the estimation of ground-motion models via Bayesian inference and the probabilistic programming language Stan. We show one ca implement a simple ground-motion model in Stan, and how can run the program from the computer environment R. We also show how one can access the results, and plot summaries of estimated parameters. A large number of different Stan models for the development of GMMs is available at https://github.com/pstafford/StanGMMTutorial.

1 Introduction

Empirical ground-motion models (GMMs) are an essential ingredient for probabilistic seismic hazard analysis (PSHA). They are typically estimated from data via regression, often using maximum-likelihood. Recently, Stafford (2019) provided an argument that GMMs should be estimated via Bayesian inference, as it provides a toolchain for integrating new data into existing GMMs. Bayesian methods have been used for estimation of a GMM a couple of times (e.g. Arroyo and Ordaz, 2010,a; Kowsari et al., 2019, 2020; Kuehn and Abrahamson, 2018; Kuehn et al., 2019; Kuehn and Scherbaum, 2015, 2016; Ordaz et al., 1994; Rahpeyma et al.,

*kuehn@ucla.edu

†p.stafford@imperial.ac.uk

2018; Stafford, 2014, 2019; Wang and Takada, 2009), but have not found widespread adoption, which might be due to the fact that Bayesian methods are sometimes viewed as complicated. Here, we want to give a short tutorial how to implement Bayesian regression models in Stan (Carpenter et al., 2017) for the estimation of GMPEs. Stan is a program that implements Markov Chain Monte Carlo inference to estimate the posterior distribution (similar programs are WinBUGS/OpenBUGS (Lunn et al., 2009, 2000) or JAGS (Plummer, 2003), though these are using Gibbs sampling, as opposed to Hamiltonian Monte Carlo (HMC) like Stan). We do not want to go into philosophical arguments about Bayesian statistics; we treat Bayesian inference as a practical tool that helps to combine prior knowledge and complex models. However, we note that GMM development typically requires a lot of judgement and prior information, so that the model behaves well and extrapolates well to large magnitudes and sparse distances (data ranges which are typically sparse). Here, prior distributions are helpful to constrain the model coefficients to reasonable values.

Our goal in this paper is very specific: to provide a collection of models, coded in Stan, that are useful for various aspects of GMM development. This is not a tutorial or introduction to Bayesian statistics. It is also not a tutorial to MCMC. We provide some information about these topics; in particular, model assessment is important in the context of MCMC. However, we recommend that the interested reader delves deeper into the topic. There are textbooks on Bayesian statistics that include Stan code (e.g. Gelman et al., 2013; Kruschke, 2015; McElreath, 2020). For thoughts on a principled Bayesian workflow, see Gelman et al. (2020).

As stated before, in this work we focus on the implementation of GMMs in Stan. We provide a large collection of models at `https://github.com/pstafford/StanGMMTutorial`. Example Models include truncated regression models (Kuehn and Abrahamson, 2020; Kuehn et al., 2020b), models with correlated target variables (Kuehn and Scherbaum, 2015), measurement uncertainty models (Kuehn and Abrahamson, 2018; Stafford, 2014), partially nonergodic models (Kuehn et al., 2020a; Kuehn and Scherbaum, 2016; Stafford, 2014), and models including spatial correlation Huang and Galasso (2019); Jayaram and Baker (2009); Kuehn and Abrahamson (2020); Stafford et al. (2019). We also show one can code models with heteroscedastic variances, or implement Bayesian robust regression (Kuehn et al., 2020a).

2 Bayesian Inference

Broadly speaking, the goal of statistical inference is to estimate the values of parameters of a statistical model from data. Given a data set $\mathcal{D} = \left[\boldsymbol{X}, \vec{Y}\right]$, where $\boldsymbol{X}$ is a matrix of predictor variables and $\vec{Y}$ is a vector of target variables, we are interested to infer the parameters of a model of the form

$$y = f(\vec{x}; \vec{c}) + \varepsilon \tag{1}$$

where $f(\cdot)$ is a function that relates the target variable to the predictor variables, $\vec{c}$ are coefficients of the function, and ε is an error term, typically assumed to be normally distributed with mean zero and standard deviation σ. We will call all parameters of the model $\vec{\theta}$; in this simple model, the parameters comprise the coefficients $\vec{c}$ and the standard deviation σ.

An important quantity in this regard is the *likelihood* function $\mathcal{L}(\vec{\theta})$, which is a function of the parameters $\vec{\theta}$ of the model. The value of the likelihood function for a given observation i is defined as the value of the probability density of the observation, given parameters $\vec{\theta}$:

$$\mathcal{L}(\vec{\theta}|(y_i, \vec{x}_i)) = p(Y = y_i|\vec{x}_i, \vec{\theta}) \tag{2}$$

For a set of observations (a data set $\mathcal{D} = \{(y_1, \vec{x}_1), \ldots, (y_N, \vec{x}_N)\}^T$), one can calculate the likelihood of the full data set

$$\mathcal{L}(\vec{\theta}|\mathcal{D}) = p(\vec{Y} = \vec{y}|\vec{X}, \vec{\theta}) \tag{3}$$

where $\vec{X}$ is a matrix containing the predictor variables for all observations. If the observations are independent, the likelihood factorizes, and one can calculate the joint likelihood as

$$\mathcal{L}(\vec{\theta}|D) = \prod_{i=1}^{N} \mathcal{L}(\vec{\theta}|(y_i, \vec{x}_i)) \tag{4}$$

In a maximum-likelihood (ML) setting, the goal is to find the set of parameters $\vec{\theta}$ that maximizes the joint likelihood, given the data. Note that, given the definition of the likelihood function in Equation (2), one actually maximizes the probability of seeing the data, given the parameters. By contrast, in Bayesian inference the goal is to estimate the so-called *posterior distribution* of the parameters, which is the conditional distribution of the parameters, given

the observed data. The posterior distribution can be calculated by Bayes' theorem as

$$p(\vec{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\vec{\theta})p(\vec{\theta}) \tag{5}$$

where $p(\mathcal{D}|\vec{\theta})$ is the likelihood, and $p(\vec{\theta})$ is called the *prior* distribution. The prior distribution $p(\vec{\theta})$ allows one to incorporate prior information about some parameters, or to constrain the behavior of the model; this is important in GMM development, as data is generally sparse at large magnitudes and short distances, so the data is not very informative. The prior distribution can also serve as some sort of regularization to avoid overfitting to the data. A simple introduction to Bayesian inference can be found in e.g. Spiegelhalter and Rice (2009). There are also many dedicated textbooks, like Kruschke (2015), Gelman et al. (2013) or McElreath (2020).

The posterior distribution $p(\vec{\theta}|\mathcal{D})$ contains all relevant information about the parameters, conditioned on the data and taking prior information into account. It is thus a good tool to assess the (epistemic) uncertainties of a model. One can calculate summary statistics such as mean/median or standard deviations of individual parameters, as well as correlations between parameters. With the posterior distribution one can also calculate the *posterior predictive distribution*[1], which encapsulates the full uncertainty in a model prediction. It can be calculated by integrating out the uncertainty of the model parameters.

Except for simple models, the posterior distribution $p(\vec{\theta}|\mathcal{D})$ cannot be calculated analytically, and one has to resort to approximations. One of the most popular tools to perform Bayesian inference is *Markov Chain Monte Carlo* (MCMC) (Neal, 1993) sampling, in which one constructs a Markov chain whose stationary distribution is the posterior distribution $p(\vec{\theta}|\mathcal{D})$. Other approximation methods are variational inference (VI) (Blei et al., 2017), or integrated nested Laplace approximation (INLA) (Rue et al., 2009). In the following, we focus on the program Stan (Carpenter et al., 2017, https://mc-stan.org/). Stan implements both MCMC and variational inference, but we will restrict ourselves to the sampling based methods. It is also possible to perform maximum-a-posteriori (MAP) inference, i.e. find the set of parameters that maximizes the log-posterior density.

There are a few reasons why we focus on Stan. We do not want to discourage people

[1] https://en.wikipedia.org/wiki/Posterior_predictive_distribution

from using other methods; this work is an encouragement to try out Bayesian methods. We use Stan because it allows us to fit models of (almost) arbitrary complexity. Some of these more complex models are described at https://github.com/pstafford/StanGMMTutorial. In addition, since Stan is based on HMC (Betancourt, 2017; Betancourt and Girolami, 2015; Neal, 2011), there are multiple tools to assess the model fit that can be used to diagnose problems with the model and/or model fit (Betancourt, 2016; Vehtari et al., 2020b).

Another advantage of using a program such as Stan is that it is actively developed and has a broad user base. While it is possible to implement their own MCMC scheme (e.g. Rahpeyma et al., 2018), the benefits of using an established software is that one can focus on the modeling part, not the implementation; in addition, maintenance is less of an issue. A dedicated implementation might be faster for a specific model, but Stan allows one great flexibility to try out different models.

3 Bayesian Inference for Ground-Motion Models

In this section, we motivate the use of Bayesian inference and Stan for the development of GMMs. A typical GMM has the form of a mixed effects model, to account for repeated measurements of the same event, or at the same station. This gives rise to a model like the following:

$$Y_{es} = f(\vec{x}_{es}; \vec{c}) + \delta B_e + \delta S_s + \delta WS_{es} \tag{6}$$

where $f(\vec{x}, \vec{c})$ is a function with coefficients $\vec{c}$, which also depends on predictor variables $\vec{x}$. e and s are indices for the event and station. δB_e and δS_s are the event and station term for event e and station s, respectively, and δWS is the remaining residual (tpyically called within-event/within-station residual). The function $f(\vec{x}_{es}; \vec{c})$ is a model for the median prediction for a generic scenario quantified by $\vec{x}_{es}$.

Due to the presence of event and station terms in Equation (6), the observations $\vec{Y}_{es}$ are not independent, and the joint likelihood is a multivariate normal distribution with covariance Σ, with non-zero off-diagonal elements Σ_{ij} if record i and j are from the same event or station. In an ML model, the random effects δB and δS are considered errors in the constant coefficient, and are integrated out of the likelihood; thus, they are not estimated, but rather predicted

given the estimated values for the coefficients and Σ (see e.g. Equation (10) of Abrahamson and Youngs (1992)). By contrast, in a Bayesian model δB and δS are considered parameters of the model that are estimated. We believe that this is a more natural position for GMM development, since one is interested in the difference of ground-motions at a particular site from the average prediction. Hence, the effects δB and δS constitute physical parameters about which inference is of interest in itself.

For a known event term δB and station term δS, the observations become independent, and the likelihood factorizes:

$$p(\vec{Y} = \vec{y}|\vec{X}, \vec{\theta}) = \prod_{i=1}^{N} p(y_i|\vec{x}_i, \vec{c}, \delta B_{eq(i)}, \delta S_{stat(i)}, \phi_{SS}) \tag{7}$$

where $\vec{X}$ is a matrix of predictor variables, whose rows consist of the vectors of predictor variables for each record. On the right hand side of Equation (7), we have partitioned the full set of parameters $\vec{\theta}$ of the model into the components; $\vec{\theta}$ consists of the event and station terms, the coefficients of the functional form $\vec{c}$, and the standard deviation of the within-event/within-station residuals ϕ_{SS} (the "noise" term). The subscripts *eq* and *stat* are indices that connect the ith record to its event and station, respectively.

As stated in Equation (5), the posterior distribution is proportional to the product of prior distribution and likelihood. For event terms and station terms, the prior distributions are normal distributions with mean zero and standard deviation τ and ϕ_{S2S}, respectively. Since these terms are typically assumed to be independent, the prior distribution for all event terms can be written as $p(\delta\vec{B}) = \prod_{i=1}^{N_E} p(\delta B_i|0, \tau)$ (similarly for station terms δS). Thus, we can write the log posterior distribution as

$$\begin{aligned}\ln p(\theta|\vec{\mathcal{D}}) = &\sum_{i=1}^{N} \ln p(y_i|\delta B_{eq(i)}, \delta S_{stat(i)}, \vec{c}, \phi_{SS}) \\ &+ \sum_{i=1}^{N_E} \ln p(\delta B_i|\tau) + \sum_{i=1}^{N_S} \ln p(\delta S|\phi_{S2S}) \\ &+ \ln p(\vec{c}) + \ln p(\tau) + \ln p(\phi_{S2S}) + \ln p(\phi_{SS})\end{aligned} \tag{8}$$

where the first line is the likelihood, the second line comprises the prior distributions for event and station terms, and the prior distributions for coefficients and standard deviations are on

the third line.

As Equation (8) shows, calculating the value of the log posterior consists of summing different log probability values. In Stan, this is how one would specify the model. In a Stan program, the statement `target += normal_lpdf(y | 0, 10);` increments the log density with the value of the expression on the right hand side; in this case, the logarithm of the value of the probability density function (pdf) of a normal distribution with mean zero and standard deviation 10, for a variable y. The construction of a Stan program consists of several statements of the form `target +=` such that they add up to the log density of the full posterior distribution.

The function `normal_lpdf(...|..., ...)` is a built-in function for the logarithm of the PDF of a normal distribution (`_lpdf` stands for logarithm of pdf). Most commonly used distributions are implemented in Stan in the same way. One can also increment the log density with the statement `y ~ normal(0, 10);`. This statement is equivalent to `target += normal_lpdf(y | 0, 10)` except that the former leaves out constant additive terms. For inference, it is only necessary to calculate the value of the log density up to an additive constant.

4 Stan Implementation of Simple GMM

In this section, we provide code for a Stan program of a simple GMM and discuss the implementation. We will use a model with the following functional form as an example

$$f(\vec{x}, \vec{c}) = c_1 + c_2 * M + c_3 * (8 - M)^2 + c_4 \ln(R + h) + c_5 M \ln(R + h) + c_6 R + c_7 \ln \frac{V_{S30}}{400} \quad (9)$$

If the finite-fault term h is fixed, then the model becomes linear, and can be written as

$$f(\vec{x}, \vec{c}) = \vec{c} \cdot \vec{x} \quad (10)$$

with $\vec{x} = \left[1, M, (8 - M)^2, \ln(R + h), M \ln(R + h), R, \ln \frac{V_{S30}}{400}\right]$. For the whole data set, the predictors can be placed into a matrix $\boldsymbol{X}$ of size $N \times K$, where K is the number of (linear) predictors (i.e. $K = 7$ in this example).

The Stan code for the model of Equation (9) is shown below:

```
data {
```

```
  int<lower=1> N;       // number of records
  int<lower=1> N_eq;    // number of earthquakes
  int<lower=1> N_stat; // number of stations
  int<lower=1> K;       // number of predictors

  matrix[N, K] X;       // matrix of predictors
  vector[N] Y;          // target variables

  int<lower=1, upper=N_eq> idx_eq[N];     // event index for each record
  int<lower=1, upper=N_stat> idx_stat[N]; // station index for each record
}

parameters {
  vector[K] c;              // coefficients

  real<lower=0> phi_SS;    // standard deviation for within-event residuals
  real<lower=0> phi_S2S;   // standard deviation of between-event residuals
  real<lower=0> tau;       // standard deviation of site-to-site residuals

  vector[N_eq] deltaB;      // event terms
  vector[N_stat] deltaS;    // station terms
}

model {
  // prior distributions
  c ~ normal(0,10);
  phi_SS ~ normal(0,0.5);
  tau ~ normal(0,0.5);
  phi_S2S ~ normal(0,0.5);
```

```
    deltaB ~ normal(0,tau);
    deltaS ~ normal(0,phi_S2S);

    // likelihood
    Y ~ normal(c * X + deltaB[idx_eq] + deltaS[idx_stat], phi_SS);
  }
```

The basic structure of the program consists of three blocks, the `data {...}` block, the `parameters {...}` block, and the `model {...}` block. In the `data {...}` block, the input parameters are defined. In the case of the GMM, these are the number of data, evens, stations, and linear predictors, the predictor matrix X, the target variable Y, and indices linking records to events and stations. Variables in Stan are typed, meaning that there is a difference between a real and an integer. An integer variable is defined as `int N;`, while a real is defined as `real a;`. The target variable is read in as `vector[N] Y;`, where notation `[]` denotes the size of the vector (similar for matrices). Indexing a vector starts with 1 (i.e., the first element in a vector has index 1). The indices for events and stations are declared as arrays of integers, `int idx_stat[N];`. Note that arrays are declared differently than vectors; the size of the array is declared after the variable name. Constraints on variables are denoted by `<lower=L, upper=U> a;`, which means that the variable a must be larger than L and smaller than U. If a constraint is declared in the data block, Stan will check the values, and throw an error if the constraint is not met. Here, we declared constraints for the indices (they cannot be smaller than 1 or larger than the total number of events/stations), and the number of data/events/stations, since we assume that we will have at least one data point in the data.

In the `parameters {...}` block, the parameters to be estimated are declared. As stated before, these are the coefficients, the standard deviations, and the event and station terms. The parameters are declared similar as in the data block. Since the standard deviations are positive quantities, there must be a constraint `<lower=0>` in their declaration.

The model `model {...}` block contains the probability model, which contains mainly the statements incrementing the log posterior. We have first declared the prior distributions for the coefficients and standard deviations, then for the event and station terms, and finally the likelihood. Note that the order of the statements is not important. Many functions in Stan are

vectorized, which means that they can be applied to vectors. The vectorization is a shorthand and more efficient than an explicit loop. Hence the statement `c ~ normal(0,10);` is equivalent to

```
for(i in 1:K) {
  c[i] ~ normal(0, 10);
}
```

Similarly, the prior distributions for event/station terms, as well as the likelihood are vectorized.

Next, we show an example how one could define the design matrix `X` inside a Stan program. Here, we read in the predictor variables $\mathbf{M}$, R, and V_{S30} in the `data {...}` block, and then define the matrix `X` in the `transformed data {...}` block. The individual columns are assigned the linear predictors. Since most of the functions in Stan are vectorized, one can often assign the whole column at once. In the declaration of the linear prdictor accounting for magnitude dependent geometrical spreading, the term `.*` is used to indicate elementwise multiplication of two vectors. The rest of the model is unchanged.

```
data {
  ...
  vector[N_eq] M;
  vector[N] R;
  vector[N_stat] VS;
  ...
}


transformed data {
  int K = 7;
  real h = 6;
  matrix[N,K] X;


  X[:,1] = rep_vector(1, N);
  X[:,2] = M[eq];
  X[:,3] = square(8 - M[eq]);
```

```
    X[:,4] = log(sqrt(square(R) + square(h)));
    X[:,5] = M[eq] .* log(sqrt(square(R) + square(h)));
    X[:,6] = R;
    X[:,7] = log(VS[stat] / 400);
  }

  parameters {
    ...
  }

  model {
    ...
  }
```

4.1 Running the Stan Model

There are many different ways to execute the Stan model, depending on the working environment. Stan can be run from the command line, or from computing environments like R, Python, Julia, Matlab, Mathematica via dedicated packages (see https://mc-stan.org/users/interfaces/). For `R` (R Core Team, 2020), one can use the package `Rstan` (Stan Development Team, 2020), or `cmdstanR` (Gabry and Cesnovar, 2020), which is a wrapper for the command line version. Similarly, for Python there exist the packages `PyStan` (Riddell et al., 2021) or `cmdstanpy` (Stan Development Team, 2021a). `Rstan` and `Pystan` are more deeply integrated into `R` and Python. We will show how to run the model from `R` using `cmdstanR`. The reason for `cmdstanR` is that it can use the latest version of Stan, with the latest improvements. By contrast, `Rstan` lags behind in the versions of Stan used, but offers more advanced features of analysis in R.

In this section, we provide example code to run and assess a simple GMM using Stan. The full example is found at https://github.com/pstafford/StanGMMTutorial/blob/master/gmm_stan_tutorial.html. Code to run the model using `cmdStanR` is shown below.

```
data_list <- list(N = length(data_y),
```

```
                    N_eq = neq,
                    N_stat = nstat,
                    K = 7,
                    X = data_x,
                    Y = data_y,
                    idx_eq = eqid,
                    idx_stat = statid
)

# sample
file <- file.path('path/to/stan-file/', 'gmm.stan')
mod <- cmdstan_model(file)

fit <- mod$sample(
  data = data_list,
  seed = 123,
  chains = 4,
  iter_sampling = 1000,
  iter_warmup = 1000,
  refresh = 500
)
```

The Stan code is stored in a file called `gmm.stan`, and the command `mod <- cmdstan_model(file)` compiles the file. Then, with `mod$sample()` the MCMC sampler is called. In this case, we have specified a seed value for reproducibility. The number of chains is set to 4, with 1000 warm-up samples to be discarded, and 1000 samples post warm-up that can be used for inference. Every 500 iterations progress is printed. For an exhaustive list of arguments, see https://mc-stan.org/cmdstanr/reference/model-method-sample.html.

The outcome of the model fit is stored as a `CmdStanMCMC` object. The `posterior` package (Bürkner et al., 2021) provides parctical tools to wok with the outcomes of Bayesian models, such as the `draws` format. The `bayesplot` package (Gabry et al., 2019) provides many plotting

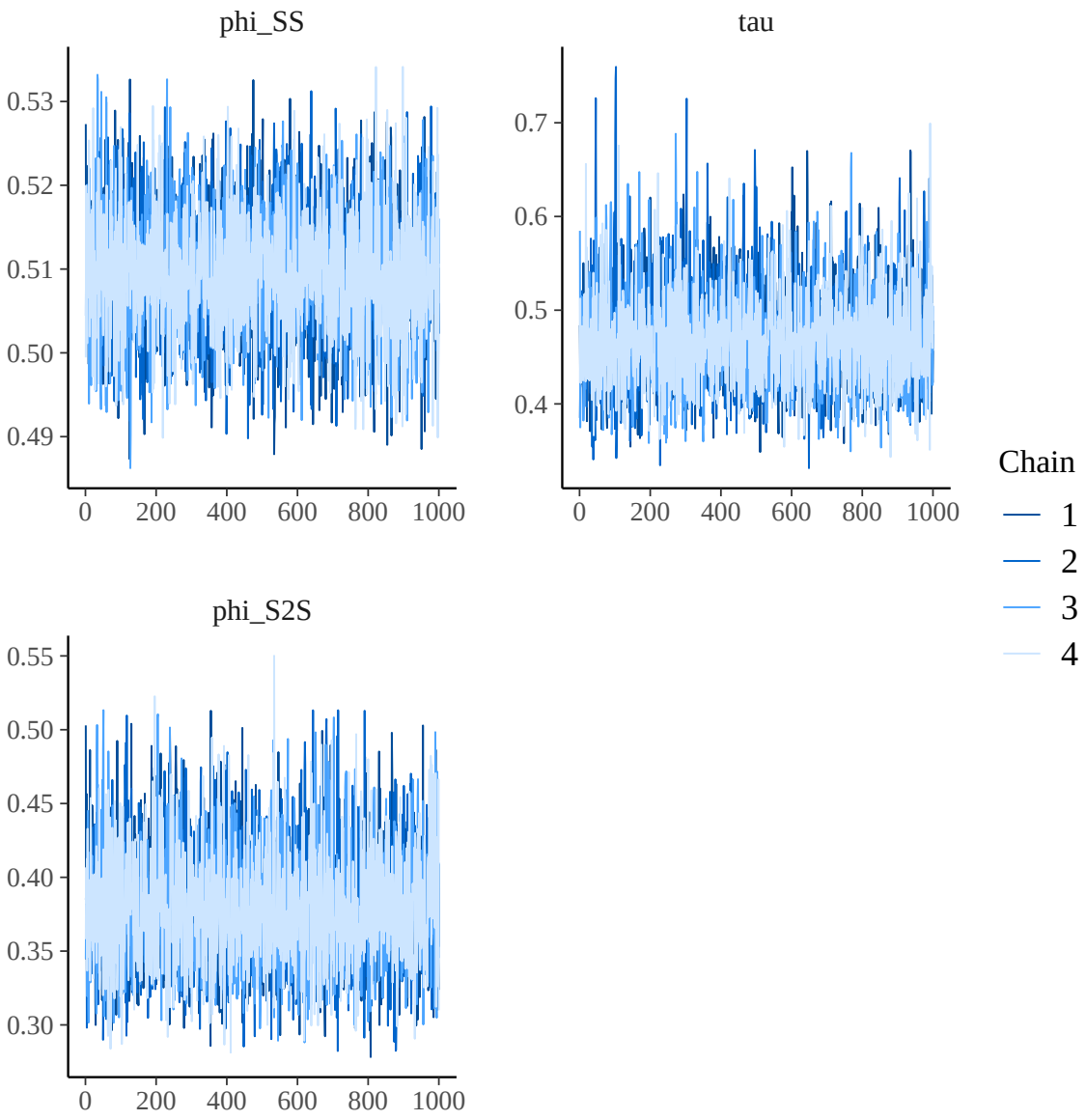

Figure 1: Traceplots of the posterior distributions of the standard deviation parameters.

functions to assess the fits. For example, one can make a traceplot of the different chains for the standard deviation parameters using the following command

```
posterior <- fit$draws()
mcmc_trace(posterior, regex_pars = c("phi_SS", "tau", "phi_S2S"),
                                   facet_args = list(nrow = 2)) +
  xaxis_title(size = 30, family = "sans") +
  yaxis_title(size = 30, family = "sans")
```

which produces a plot like in Figure 1.

Histograms of the posterior distribution can be plotted with the function `mcmc_hist()`. An example is shown in Figure 2, which shows the posterior distributions of the coefficients

$c_1, \ldots, c_7$. This plot is created via

```
mcmc_hist(posterior, regex_pars = c("c"))
```

The plots from `bayesplot` are ggplot objects, and can be customized using functions from the `ggplot2` package (Wickham, 2016). In Figure 3, we show the histograms of the posterior distributions of the standard deviation parameters, together with the true values and the values estimated by `lmer`, using the following code

```
df_lmer_sd <- as.data.frame(VarCorr(fit_lmer))
sd_names <- c('tau', 'phi_S2S', 'phi_SS')
size_text <- 20
size_title <- 30
i <- 1 # plot histogram for tau
mcmc_dens(posterior, pars = sd_names[i]) +
  vline_at(c(sd_list[i], df_lmer_sd$sdcor[i]), size = c(1,0.75), linetype = c(1,2)) +
  xaxis_text(size = size_text, family = "sans") +
  yaxis_text(size = size_text, family = "sans") +
  xaxis_title(size = size_title, family = "sans") +
  yaxis_title(size = size_title, family = "sans") +
  grid_lines(color = "gray60")
```

At https://github.com/pstafford/StanGMMTutorial, we provide some additional plots assessing the posterior distribution, such as pair plots and interval plots. For more possible options and examples, see https://mc-stan.org/bayesplot/index.html. We can also save the fitted object with

```
filename <- 'fit_gmm_stan.RDS'
fit$save_object(file = filename)
```

At https://github.com/pstafford/StanGMMTutorial, we provide more on how one can retrieve subsets of the parameters from the estimated posterior (in the form of a `draws` object), which can then be used for further analysis.

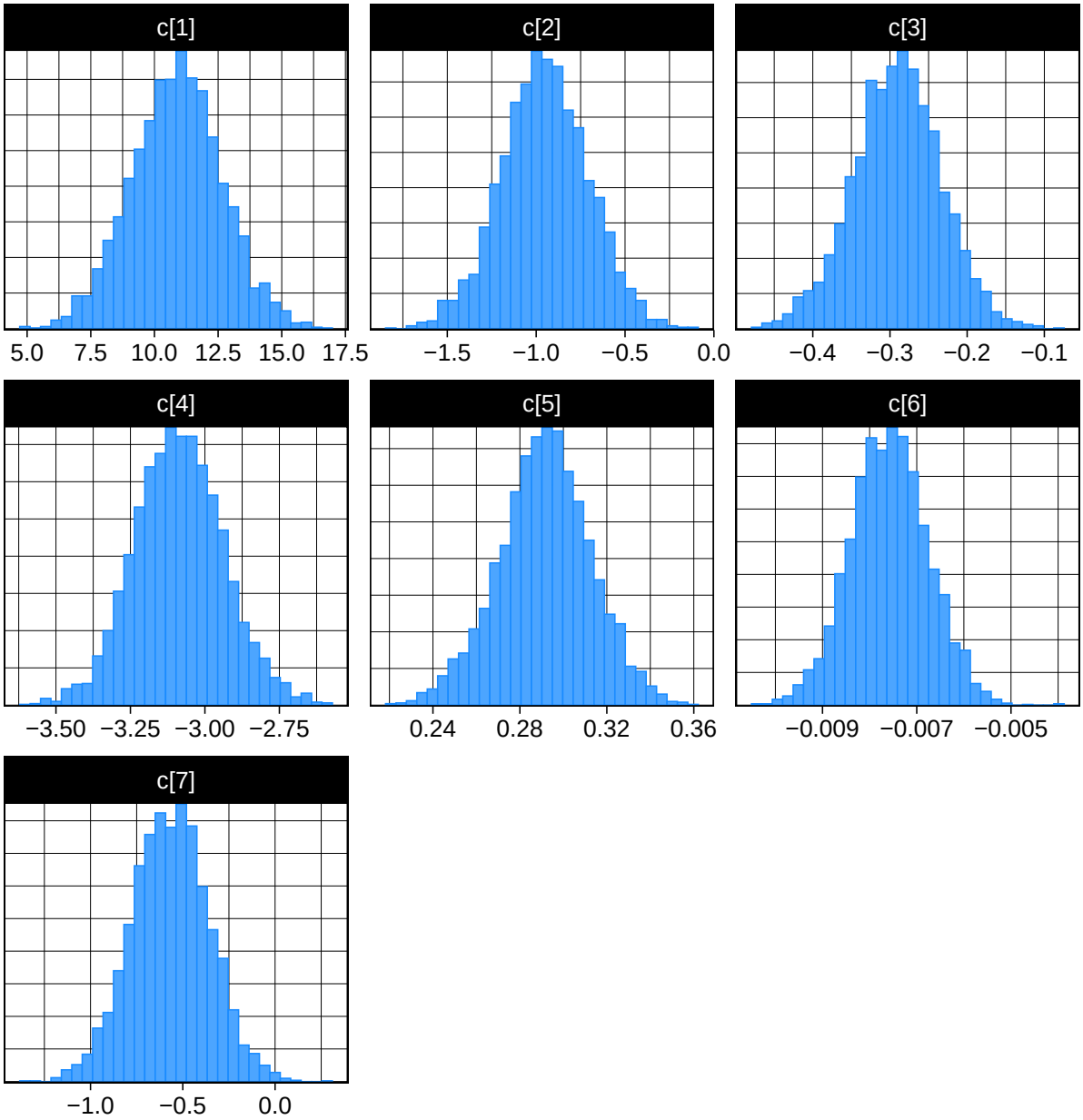

Figure 2: Posterior histograms of the coefficients.

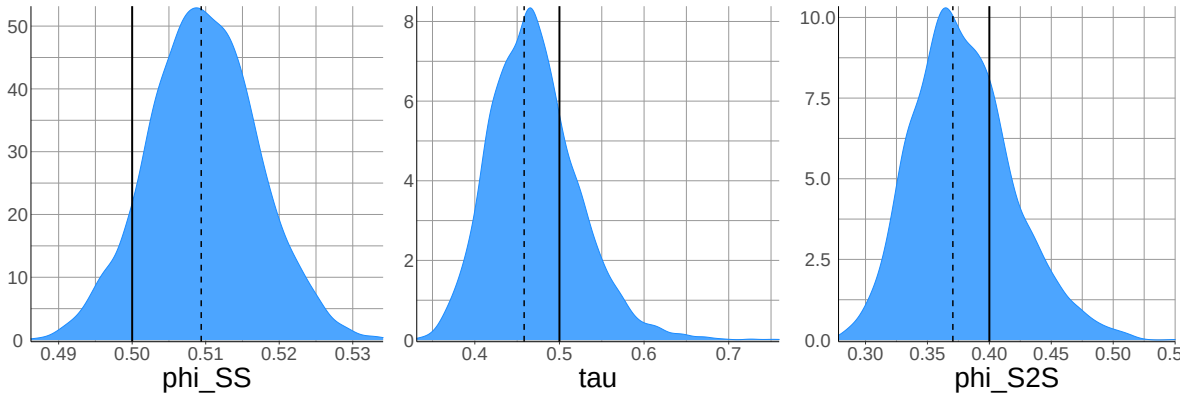

Figure 3: Posterior densities of standard deviation parameters, together with true values (solid vertical line) and values estimated by lmer (dashed vertical line)

4.2 Other Models

At https://github.com/pstafford/StanGMMTutorial, we have compiled a large list of Stan models that showcase the rich variety of models that can be fit using Stan. We also provide the code to generate the data, run the model, and generate the Figures in this article. Models that can be fit easily with Stan include truncated regression models (Kuehn et al., 2020b), models with correlated target variables (Kuehn and Scherbaum, 2015), measurement uncertainty models (Kuehn and Abrahamson, 2018; Stafford, 2014), and models including spatial correlation Huang and Galasso (2019); Jayaram and Baker (2009); Kuehn and Abrahamson (2020); Stafford et al. (2019). Examples of these kinds of models can be found at https://github.com/pstafford/StanGMMTutorial, together with more information about implementation. In addition to these models, Kuehn (2021) provides code to estimate non-ergodic GMMs based on varying coefficient models (Bussas et al., 2017; Gelfand et al., 2003; Landwehr et al., 2016).

5 Prior Distributions

In our example, we have not put much emphasis on the prior distributions for the parameters. This is in part because we simulated enough data and the functional form is simple, so that the parameters are well constrained by the simulated data. In the development of GMMs, however, prior distributions can become important, as often data is sparse for some data ranges (e.g. at large magnitudes or very close to the fault), and in this case one needs to constrain the model behavior to avoid unphysical behavior. In GMMs estimated with maximum likelihood, some coefficients are fixed if they cannot be reliably estimated from data; however, this ignores possible uncertainty in the fixed parameter. In terms of a Bayesian model, one can think of a fixed parameter as a parameter with a very narrow prior distribution.

It is difficult to give general advice about setting prior distributions that work in every case, but we believe that this is a par of the model that should no be neglected. Some general advice on prior distributions can be found at https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations.

To set informative prior distributions, one can use for example models that were developed

for a different region (but a similar tectonic environment), making the assumption that ground-motion scaling for the same tectonic envionment is broadly similar. This approach has been taken in e.g. Kuehn and Scherbaum (2016) and Kowsari et al. (2020), where it was shown that the use of informative prior distributions stabilized the regression results for regions with sparse data.

Another approach would be to use simulations to inform the prior distributions. Simulations have been used to constrain model parts that are not well informed by data, such as hanging wall effects in the NGA West-2 models (Abrahamson et al., 2014; Boore et al., 2014; Campbell and Bozorgnia, 2014; Chiou and Youngs, 2014), based on Donahue and Abrahamson (2014). Similarly, the nonlinear site amplification in these models is based on simulations as well (Kamai et al., 2014; Walling et al., 2008).

A similar approach was taken to constrain the large magnitude scaling in the NGA Subduction model of Kuehn et al. (2020a), which is based on the models of Gregor et al. (2002) and Atkinson and Macias (2009). This is not well constrained by data, given that there are very few large magnitude events in the NGA Subduction database (Bozorgnia and Stewart, 2020), and thus it needs to be constrained based on external information. The near-fault term in the NGA subduction model of (Parker et al., 2020) is also based on simulations, though their model is not developed via Bayesian inference. Ordaz et al. (1994) derive prior distributions for a GMM applicable in Mexico City from seismological theory, and discuss in detail how they arrive at the values for the expectation and range of their prior distributions.

In our experience, prior distributions are important for the standard deviations of partially nonergodic adjustment terms. Typically, a partially nonergodic model comprises few regions, so the standard deviations of the adjustment terms are not well constrained. In this case, it makes sense to use a prior distribution that discourages large values of the standard deviation, so that only a strong regional signal in the data leads to significant deviations of the regional parameters from the global average model. Kuehn et al. (2020a) compared Half-Cauchy and exponential prior distributions for ther regional standard deviations. They found that the Cauchy distributions, which has heavy tails, can lead to spurious regional deviations for some coefficients, and concluded that the exponential distribution is a better choice. In https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations, sometimes

also a Half-normal or a Half-Student-T distribution with low degrees of freedom is recommended for standard deviations of hierarchical models.

The notion discussed in the previous paragraph is informed by Simpson et al. (2017), who develop the penalized complexity (PC) prior approach, which sees a hierarchical model (such as a regional random effects model) as a complex extension of a simpler base model. The more complex model is penalized (shrunk towards the base model) unless there is strong information in the data that supports it. This approach has been extended to priors for spatial models (Fuglstad et al., 2019), which becomes important for nonergodic models based on varying coefficient models (Franco-Villoria et al., 2019; Landwehr et al., 2016).

In the end, as stated before it is difficult to give general advice, but we believe that one should at least think about the prior distributions for each parameter. To get an idea about the impact of prior distributions, one can also simulate data based on the prior distributions (i.e. sample coefficients from the prior distribution, then generate synthetic data based on the model). One can then assess whether the simulated data "makes sense", or show unrealistic values. This is a example of (qualitative) prior predictive checking (Gabry et al., 2019; Gelman et al., 2020).

6 Discussion

We have provided a simple introduction to the estimation of GMMs with Stan, using MCMC. The main contribution, though, is the list of models compiled at `https://github.com/pstafford/StanGMMTutorial`, which can serve as a starting point for anyone who is interested to esimate GMMs with Bayesian inference. Throughout the article, we have mentioned that we believe the Bayesian approach has advantages in GMM development: one can set informative prior distributions, and the posterior distribution allows for an easy quantification of epistemic uncertainty. There are disadvantages as well: a Stan model estimated with MCMC will generally take longer to run than an ML model. For example, the simple example model took about 5 minutes with Stan, but only about a second using `lmer`. For more complicated models, the runtime will typically be even longer. It depends on the actual application and model whether this is a concern.

This article is only a brief introduction to Stan, and touches upon Bayesian statistics very

lightly. For more thorough exposition, some introductory textbooks, such as Kruschke (2015) or McElreath (2020) are recommended. For a better understanding of Stan, one should also look at the Stan documentation (`https://mc-stan.org/users/documentation/`). There are different ways to code the same model in Stan, and it is often not immediately obvious which parameterization is most efficient. An example is the difference between a centered/non-centered parameterization in hierarchical models (Betancourt, 2020; Betancourt and Girolami, 2015).

For model comparison in MCMC models, one can use the widely applicable information criterion (WAIC) (Watanabe, 2013), cross-validation (CV), or approximate leave-one-out CV (LOO-CV) (Vehtari et al., 2017). Many of these are easily available for Stan models, and can be computed via the `loo` R-package (Vehtari et al., 2020a).

We have focused on running the Stan models in `R`, but Stan can be run from many interfaces. For Python users, there exist the packages `PyStan` or `cmdstanpy`. For assessment of the posterior dstribution, the Python package `ArviZ` (Kumar et al., 2019) is very useful, and provides many of the same tools and capabilities as the R-packages used in this work.

References

Abrahamson, N. A., Silva, W. J., and Kamai, R. (2014). "Summary of the ASK14 Ground Motion Relation for Active Crustal Regions." *Earthquake Spectra*, 30(3), 1025–1055.

Abrahamson, N. A. and Youngs, R. R. (1992). "A stable algorithm for regression analysis using the random effects model." *Bulletin of the Seismological Society of America*, 82(1), 505–510.

Arroyo, D. and Ordaz, M. (2010). "Multivariate Bayesian Regression Analysis Applied to Ground-Motion Prediction Equations, Part 1: Theory and Synthetic Example." *Bulletin of the Seismological Society of America*, 100(4), 1551–1567.

Arroyo, D. and Ordaz, M. (2010a). "Multivariate Bayesian Regression Analysis Applied to Ground-Motion Prediction Equations, Part 2: Numerical Example with Actual Data." *Bulletin of the Seismological Society of America*, 100(4), 1568-1577.

Atkinson, G. M. and Macias, M. (2009). "Predicted Ground Motions for Great Interface Earth-

quakes in the Cascadia Subduction Zone." *Bulletin of the Seismological Society of America*, 99(3), 1552–1578.

Betancourt, M. (2016). "Diagnosing Suboptimal Cotangent Disintegrations in Hamiltonian Monte Carlo. http://arxiv.org/abs/1604.00695

Betancourt, M. (2017). "The Convergence of Markov chain Monte Carlo Methods: From the Metropolis method to Hamiltonian Monte Carlo." http://arxiv.org/abs/1706.01520

Betancourt, M. (2020). "Hierarchical Modeling.." retrieved from https://github.com/betanalpha/knitr_case_studies/tree/master/hierarchical_modeling, commit27c1d260e9ceca710465dc3b02f59f59b729ca43

Betancourt, M. and Girolami, M. (2015). "Hamiltonian Monte Carlo for Hierarchical Models." *Current Trends in Bayesian Methodology with Applications*, Chapman and Hall/CRC, 79–101.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). "Variational Inference: A Review for Statisticians." *Journal of the American Statistical Association*, 112(518), 859–877.

Boore, D. M., Stewart, J. P., Seyhan, E., and Atkinson, G. M. (2014). "NGA-West2 Equations for Predicting PGA, PGV, and 5% Damped PSA for Shallow Crustal Earthquakes." *Earthquake Spectra*, 30(3), 1057–1085.

Bozorgnia, Y. and Stewart, J. P. (2020). "Data Resources for NGA-Subduction Project." *Report No. 2020/02*, Pacific Earthquake Engineering Research Center.

Bürkner, P.-C., Gabry, J., Kay, M., and Vehtari, A. (2021). "posterior: Tools for Working with Posterior Distributions", R package version 1.0.0, https://mc-stan.org/posterior/.

Bussas, M., Sawade, C., Kühn, N., Scheffer, T., and Landwehr, N. (2017). "Varying-coefficient models for geospatial transfer learning." *Machine Learning*, 106(9-10), 1419–1440.

Campbell, K. W. and Bozorgnia, Y. (2014). "NGA-West2 Ground Motion Model for the Average Horizontal Components of PGA, PGV, and 5% Damped Linear Acceleration Response Spectra." *Earthquake Spectra*, 30(3), 1087–1115.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). "Stan : A Probabilistic Programming Language." *Journal of Statistical Software*, 76(1), 1–32.

Chiou, B. S.-J. and Youngs, R. R. (2014). "Update of the Chiou and Youngs NGA Model for the Average Horizontal Component of Peak Ground Motion and Response Spectra." *Earthquake Spectra*, 30(3), 1117–1153.

Donahue, J. L. and Abrahamson, N. A. (2014). "Simulation-Based Hanging Wall Effects." *Earthquake Spectra*, 30(3), 1269–1284.

Franco-Villoria, M., Ventrucci, M., and Rue, H. (2019). "A unified view on Bayesian varying coefficient models." *Electronic Journal of Statistics*, 13(2), 5334–5359.

Fuglstad, G.-A., Simpson, D., Lindgren, F., and Rue, H. (2019). "Constructing Priors that Penalize the Complexity of Gaussian Random Fields." *Journal of the American Statistical Association*, 114(525), 445–452.

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2019). "Visualization in Bayesian workflow." *Journal of the Royal Statistical Society. Series A: Statistics in Society*, 182(2), 389–402.

Gabry, J. and Cesnovar, R. "cmdstanr: R Interface to 'CmdStan'", https://mc-stan.org/cmdstanr,https://discourse.mc-stan.org

Gelfand, A. E., Kim, H.-J., Sirmans, C. F., and Banerjee, S. (2003). "Spatial Modeling With Spatially Varying Coefficient Processes." *Journal of the American Statistical Association*, 98(462), 387–396.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman and Hall/CRC.

Gelman, A., Vehtari, A., Simpson, D., Margossian, C. C., Carpenter, B., Yao, Y., Kennedy, L., Gabry, J., Bürkner, P.-C., and Modrák, M. (2020). "Bayesian Workflow.

Gregor, N. J., Silva, W. J., Wong, I. G., and Youngs, R. R. (2002). "Ground-motion attenuation relationships for Cascadia subduction zone megathrust earthquakes based on a stochastic finite-fault model." *Bulletin of the Seismological Society of America*, 92(5), 1923–1932.

Huang, C. and Galasso, C. (2019). "Ground-motion intensity measure correlations observed in Italian strong-motion records." *Earthquake Engineering and Structural Dynamics*, 48(15), 1634–1660.

Jayaram, N. and Baker, J. W. (2009). "Correlation model for spatially distributed ground-motion intensities." *Earthquake Engineering & Structural Dynamics*, 38(15), 1687–1708.

Kamai, R., Abrahamson, N. A., and Silva, W. J. (2014). "Nonlinear Horizontal Site Amplification for Constraining the NGA-West2 GMPEs." *Earthquake Spectra*, 30(3), 1223–1240.

Kowsari, M., Halldorsson, B., Hrafnkelsson, B., and Jónsson, S. (2019). "Selection of earthquake ground motion models using the deviance information criterion." *Soil Dynamics and Earthquake Engineering*, 117(May 2018), 288–299.

Kowsari, M., Sonnemann, T., Halldorsson, B., Hrafnkelsson, B., Snaebjörnsson, J. P., and Jonsson, S. (2020). "Bayesian inference of empirical ground motion models to pseudo-spectral accelerations of south Iceland seismic zone earthquakes based on informative priors." *Soil Dynamics and Earthquake Engineering*, 132, 106075.

Kruschke, J. K. (2015). *Doing Bayesian Data Analysis, Second Edition: A Tutorial with R, JAGS, and Stan*. Academic Press / Elsevier, 2nd editio edition.

Kuehn, N. (2021). "Comparison of Bayesian Varying Coefficient Models for the Development of Nonergodic Ground-Motion Models." *engrXiv*, doi:10.31224/osf.io/tjxa3

Kuehn, N., Bozorgnia, Y., Campbell, K. W., and Gregor, N. (2020a). "Partially Non-Ergodic Ground-Motion Model for Subduction Regions using the NGA-Subduction Database." *Report No. 2020/04*, Pacific Earthquake Engineerig Research Center.

Kuehn, N. M. and Abrahamson, N. A. (2018). "The Effect of Uncertainty in Predictor Variables on the Estimation of Ground-Motion Prediction Equations." *Bulletin of the Seismological Society of America*, 108(1), 358–370.

Kuehn, N. M. and Abrahamson, N. A. (2020). "Spatial correlations of ground motion for non-ergodic seismic hazard analysis." *Earthquake Engineering & Structural Dynamics*, 49(1), 4–23.

Kuehn, N. M., Abrahamson, N. A., and Walling, M. A. (2019). "Incorporating Nonergodic Path Effects into the NGA-West2 Ground-Motion Prediction Equations." *Bulletin of the Seismological Society of America*, 109(2), 575–585.

Kuehn, N. M., Kishida, T., AlHamaydeh, M., Lavrentiadis, G., and Bozorgnia, Y. (2020b). "A Bayesian model for truncated regression for the estimation of empirical ground-motion models." *Bulletin of Earthquake Engineering*, 18(14), 6149–6179.

Kuehn, N. M. and Scherbaum, F. (2015). "Ground-motion prediction model building: a multilevel approach." *Bulletin of Earthquake Engineering*, 13(9), 2481–2491.

Kuehn, N. M. and Scherbaum, F. (2016). "A partially non-ergodic ground-motion prediction equation for Europe and the Middle East." *Bulletin of Earthquake Engineering*, 14(10), 2629–2642.

Kumar, R., Carroll, C., Hartikainen, A., and Martin, O. (2019). "ArviZ a unified library for exploratory analysis of Bayesian models in Python." *Journal of Open Source Software*, 4(33), 1143.

Landwehr, N., Kuehn, N. M., Scheffer, T., and Abrahamson, N. (2016). "A Nonergodic Ground-Motion Model for California with Spatially Varying Coefficients." *Bulletin of the Seismological Society of America*, 106(6), 2574–2583.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). "The BUGS project: Evolution, critique and future directions." *Statistics in Medicine*, 28(25), 3049–3067.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000). "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility." *Statistics and Computing*, 10(4), 325–337.

McElreath, R. (2020). *A Bayesian Course with Examples in R and Stan*. Chapman and Hall/CRC, 2nd edition.

Neal, R. (2011). "MCMC Using Hamiltonian Dynamics." *Handbook of Markov Chain Monte Carlo*, 113–162.

Neal, R. M. (1993). "Probabilistic Inference Using Markov Chain Monte Carlo Methods." *Report no.*, Department of Computer Science, Universiy of Toronto.

Ordaz, M., Arciniega, A., and Singh, S. K. (1994). "Bayesian Attenuation Regressions: an Application to Mexico City." *Geophysical Journal International*, 117(2), 335–344.

Parker, G. A., Stewart, J. P., Boore, D. M., Atkinson, G. M., and Hassani, B. (2020). "NGA-Subduction Global Ground-Motion Models with Regional Adjustment Factors." *Report No. August*.

Plummer, M. (2003). "JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." *3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, K. Hornik, F. Leisch, and A. Zeileis, eds., Vienna (1).

R Core Team (2020). "R: A language and environment for statistical computing. R Foundation for Statistical Computing." Vienna, Austria. https://www.R-project.org/

Rahpeyma, S., Halldorsson, B., Hrafnkelsson, B., and Jónsson, S. (2018). "Bayesian hierarchical model for variations in earthquake peak ground acceleration within small-aperture arrays." *Environmetrics*, 29(3), 1–19.

Riddell, A., Hartikainen, A., and Carter, M. (2021). " PyStan (3.0.0)." https://pypi.org/project/pystan

Rue, H., Martino, S., and Chopin, N. (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2), 319–392.

Simpson, D., Rue, H., Riebler, A., Martins, T. G., and Sørbye, S. H. (2017). "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors." *Statistical Science*, 32(1), 1–28.

Spiegelhalter, D. and Rice, K. (2009). "Bayesian statistics." *Scholarpedia*, 4(8), 5230.

Stafford, P. J. (2014). "Crossed and Nested Mixed-Effects Approaches for Enhanced Model Development and Removal of the Ergodic Assumption in Empirical Ground-Motion Models." *Bulletin of the Seismological Society of America*, 104(2), 702–719.

Stafford, P. J. (2019). "Continuous integration of data into ground-motion models using Bayesian updating." *Journal of Seismology*, 23(1), 39–57.

Stafford, P. J., Zurek, B. D., Ntinalexis, M., and Bommer, J. J. (2019). "Extensions to the Groningen ground-motion model for seismic risk calculations: component-to-component variability and spatial correlation." *Bulletin of Earthquake Engineering*, 17(8), 4417–4439.

Stan Development Team (2020). "RStan: the R interface to Stan". R package version 2.21.2 https://mc-stan.org/rstan/index.html

Stan Development Team (2021a). "CmdStanPy (0.9.76)", https://pypi.org/project/cmdstanpy

Vehtari, A., Gabry, J., Magnusson, M., Yao, Y., Bürkner, P.-C., Paananen, T., and Gelman, A. (2020a). "loo: Efficient leave-one-out cross-validation and WAIC for Bayesian models, R package version 2.4.1 https://mc-stan.org/loo/

Vehtari, A., Gelman, A., and Gabry, J. (2017). "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC." *Statistics and Computing*, 27(5), 1413–1432.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., and Bürkner, P.-C. (2020b). "Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC." *Bayesian Analysis*, 1–23.

Walling, M., Silva, W., and Abrahamson, N. (2008). "Nonlinear site amplification factors for constraining the NGA models." *Earthquake Spectra*, 24(1), 243–255.

Wang, M. and Takada, T. (2009). "A Bayesian Framework for Prediction of Seismic Ground Motion." *Bulletin of the Seismological Society of America*, 99(4), 2348–2364.

Watanabe, S. (2013). "A widely applicable bayesian information criterion." *Journal of Machine Learning Research*, 14(1), 867–897.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York https://ggplot2.tidyverse.org.