brms: An R Package for Bayesian Generalized Linear Mixed Models using Stan

Paul-Christian Bürkner
University of Münster

Abstract

The **brms** package implements Bayesian generalized linear mixed models in R using the probabilistic programming language **Stan**. A wide range of distributions and link functions are supported, allowing to fit – among others – linear, robust linear, binomial, Poisson, survival, ordinal, zero-inflated, and hurdle models. Further modeling options include multiple grouping factors each with multiple random effects, autocorrelation of the response variable, user defined covariance structures, censored data, as well as meta-analytic standard errors. Prior specifications are flexible and explicitly encourage users to apply prior distributions that actually reflect their beliefs. In addition, model fit can easily be assessed and compared with the Watanabe-Akaike-Information Criterion and leave-one-out cross-validation.

Keywords: Bayesian inference, mixed model, ordinal data, MCMC, **Stan**, R.

1. Introduction

Generalized linear mixed models (GLMMs) offer a great flexibility for researchers across sciences (Brown and Prescott 2015; Pinheiro and Bates 2006; Demidenko 2013) and it is not surprising that many packages for R (R Core Team 2015b) have been developed to fit GLMMs. Possibly the most widely known package in this area is **lme4** (Bates, Mächler, Bolker, and Walker 2015), which uses maximum likelihood or restricted maximum likelihood methods for model fitting. Although alternative Bayesian methods have several advantages over frequentist approaches (e.g., the possibility of explicitly incorporating prior knowledge about parameters into the model), their practical use was limited for a long time because the posterior distributions of more complex models (such as GLMMs) could not be found analytically. Markov chain Monte Carlo (MCMC) algorithms allowing to draw random samples from the posterior were not available or too time-consuming. In the last few decades, however, this has changed with the development of new algorithms and the rapid increase of general computing power. Today, several software packages implement these techniques, for instance **WinBugs** (Lunn, Thomas, Best, and Spiegelhalter 2000; Spiegelhalter, Thomas, Best, and Lunn 2003), **OpenBugs** (Spiegelhalter, Thomas, Best, and Lunn 2007), **JAGS** (Plummer 2013), **MCMCglmm** (Hadfield 2010) and **Stan** (Stan Development Team 2015a; Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancour, Brubaker, Guo, Li, and Ridell 2015) to mention only a few. With the exception of the latter one, all of these programs are primarily using combinations of Metropolis-Hastings updates (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Hastings 1970) and Gibbs-sampling (Geman and Geman 1984; Gelfand and

Smith 1990), sometimes also coupled with slice-sampling (Damien, Wakefield, and Walker 1999; Neal 2003). While being relatively easy to implement, convergence is usually rather slow for high-dimensional models with correlated parameters (Neal 2011; Hoffman and Gelman 2014; Gelman, Carlin, Stern, and Rubin 2014). Furthermore, Gibbs-sampling requires priors to be conjugate to the likelihood of parameters in order to work efficiently (Gelman *et al.* 2014), thus reducing the freedom of the researcher in choosing a prior that reflects his or her beliefs. In contrast, **Stan** implements Hamiltonian Monte Carlo (Duane, Kennedy, Pendleton, and Roweth 1987; Neal 2011) and its extension, the No-U-Turn Sampler (NUTS) (Hoffman and Gelman 2014). These algorithms converge much more quickly especially for high-dimensional models regardless of whether the priors are conjugate or not (Hoffman and Gelman 2014).

Similar to software packages like **WinBugs**, **Stan** comes with its own programming language, allowing for great modeling flexibility (c.f., Stan Development Team 2015b; Carpenter *et al.* 2015). Many researchers may still hesitate to use **Stan** directly, as every model has to be written, debugged and possibly also optimized. This may be a time-consuming and error prone process even for researchers familiar with Bayesian inference. The package **brms**, presented in this paper, aims at closing this gap (at least for GLMMs) allowing the user to benefit from the merits of **Stan** only by using simple, **lme4**-like formula syntax. **brms** supports a wide range of distributions and link functions, allows for multiple grouping factors each with multiple random effects, autocorrelation of the response variable, user defined covariance structures, as well as flexible and explicit prior specifications.

The purpose of the present article is to provide a general overview of the **brms** package (version 0.6.0). We begin by explaining the underlying structure of GLMMs. Next, the software is introduced in detail using recurrence times of infection in kidney patients (McGilchrist and Aisbett 1991) and ratings of inhaler instructions (Ezzet and Whitehead 1991) as examples. We end by comparing **brms** to other R packages implementing GLMMs and describe future plans for extending the package.

2. Model description

The core of every GLMM is the prediction of the response y through the linear combination η of fixed and random effects predictors transformed by the inverse link function f assuming a certain distribution D for y. We write

$$y_i \sim D(f(\eta_i), \theta)$$

to stress the dependency on the i^{th} data point. In many R packages, D is also called the 'family' and we will use this term in the following. The parameter θ describes additional family specific parameters that typically do not vary across data points, such as the standard deviation σ in normal models or the shape α in Gamma or negative binomial models. The linear predictor can generally be written as

$$\eta = \mathbf{X}\beta + \mathbf{Z}u$$

where $\mathbf{X}, \mathbf{Z}$ are the fixed and random effects design matrices respectively and β, u the corresponding fixed and random effects. The design matrices $\mathbf{X}$ and $\mathbf{Z}$ as well as y make up the data, whereas β, u, and θ are the model parameters being estimated. Except for linear

models, we do not incorporate an additional error term for every observation by default. If desired, such an error term can always be modeled using a random effect with as many levels as observations in the data.

2.1. Prior distributions

Fixed effects

In **brms**, fixed effects are not restricted to have normal priors. Instead, every fixed effect can have every one-dimensional prior implemented in **Stan**, for instance uniform, Cauchy or even Gamma priors. As a negative side effect of this flexibility, correlations between fixed effects cannot be modeled as parameters. If desired, point estimates of the correlations can be obtained after sampling has been done. By default, fixed effects have an improper flat prior over the reals.

Random effects

The random effects u are assumed to come from a multivariate normal distribution with mean zero and unknown covariance matrix $\mathbf{\Sigma}$:

$$u \sim N(0, \mathbf{\Sigma})$$

As it is generally the case, covariances between random effects of different grouping factors are assumed to be zero. This implies that $\mathbf{Z}$ and u can be split up into several matrices $\mathbf{Z_k}$ and random effects u_k, where k indexes grouping factors, so that the model can be simplified to

$$u_k \sim N(0, \mathbf{\Sigma_k})$$

Usually, but not always, we can also assume random effects associated with different levels (indexed by j) of the same grouping factor to be independent leading to

$$u_{kj} \sim N(0, \mathbf{V_k})$$

The covariance matrices $\mathbf{V_k}$ are modeled as parameters. In most packages, an Inverse-Wishart distribution is used as a prior for $\mathbf{V_k}$. This is mostly because its conjugacy leads to good properties of Gibbs-Samplers (Gelman *et al.* 2014). However, there are good arguments against the Inverse-Wishart prior (Natarajan and Kass 2000; Kass and Natarajan 2006). The NUTS-Sampler implemented in **Stan** does not require priors to be conjugate. This advantage is utilized in **brms**: $\mathbf{V_k}$ is parameterized in terms of a correlation matrix $\mathbf{\Omega_k}$ and a vector of standard deviations σ_k through

$$\mathbf{V_k} = \mathbf{D}(\sigma_k)\mathbf{\Omega_k}\mathbf{D}(\sigma_k)$$

where $\mathbf{D}(\sigma_k)$ denotes the diagonal matrix with diagonal elements σ_k. Priors are then specified for the parameters on the right hand side of the equation. For $\mathbf{\Omega_k}$, we use the LKJ-Correlation prior with parameter $\zeta > 0$ by Lewandowski, Kurowicka, and Joe (2009)[1]:

$$\mathbf{\Omega_k} \sim LKJ(\zeta)$$

[1]Internally, the Cholesky factor of the correlation matrix is used, as it is more efficient and numerically stable.

If $\zeta = 1$ (the default in **brms**) the density is uniform over correlation matrices of the respective dimension. If $\zeta > 1$, non-zero correlations become less likely, whereas $0 < \zeta < 1$ results in higher probabilities for non-zero correlations. For every element of σ_k, any prior can be applied that is defined on the non-negative reals only. As default in **brms**, we use a half Student-t prior with 3 degrees of freedom. This prior often leads to better convergence of the models than a half Cauchy prior, while still being relatively weakly informative.

Sometimes – for instance when modeling pedigrees – different levels of the same grouping factor cannot be assumed to be independent. In this case, the covariance matrix of u_k becomes

$$\mathbf{\Sigma_k} = \mathbf{V_k} \otimes \mathbf{A_k}$$

where A_k is the known covariance matrix between levels and $\otimes$ is the Kronecker product.

Family specific parameters

For some families, additional parameters need to be estimated. In the current section, we only name the most important ones. Normal, Student and Cauchy distributions need the parameter σ to account for residual error variance. By default, σ has a half Cauchy prior with a scale parameter that depends on the standard deviation of the response variable to remain only weakly informative regardless of response variable's scaling. Furthermore, Student's distributions needs the parameter ν representing the degrees of freedom. By default, ν has a wide gamma prior as proposed by Juárez and Steel (2010). Gamma, Weibull, and negative binomial distributions need the shape parameter α that has a half cauchy prior by default.

3. Parameter estimation

The **brms** package does not fit models itself but uses **Stan** on the back-end. Accordingly, all samplers implemented in **Stan** can be used to fit **brms** models. Currently, these are the static Hamiltonian Monte-Carlo (HMC) Sampler sometimes also referred to as Hybrid Monte-Carlo (Neal 2011, 2003; Duane *et al.* 1987) and its extension the No-U-Turn Sampler (NUTS) by Hoffman and Gelman (2014). HMC-like algorithms produce samples that are much less autocorrelated than those of other samplers such as the random-walk Metropolis algorithm (Hoffman and Gelman 2014; Creutz 1988). The main drawback of this increased efficiency is the need to calculate the gradient of the log-posterior, which can be automated using algorithmic differentiation (Griewank and Walther 2008) but is still a time-consuming process for more complex models. Thus, using HMC leads to higher quality samples but takes more time per sample than other algorithms typically applied. Another drawback of HMC is the need to pre-specify at least two parameters, which are both critical for the performance of HMC. The NUTS Sampler allows to set these parameters automatically thus eliminating the need for any hand-tuning, while still being at least as efficient as a well tuned HMC (Hoffman and Gelman 2014). For more details on the sampling algorithms applied in **Stan**, see the **Stan** user's manual (Stan Development Team 2015b) as well as Hoffman and Gelman (2014).

In addition to the estimation of model parameters, **brms** allows to draw samples from the posterior predictive distribution as well as from the pointwise log-likelihood. Both can be used to assess model fit. The former allows a comparison between the actual response y and the response $\hat{y}$ predicted by the model. The pointwise log-likelihood can be used, among others, to calculate the Watanabe-Akaike information criterion (WAIC) proposed by Watanabe (2010)

and leave-one-out cross-validation (LOO; Gelfand, Dey, and Chang 1992; Vehtari, Gelman, and Gabry 2015a; see also Ionides 2008) both allowing to compare different models applied to the same data (lower WAICs and LOOs indicate better model fit). The WAIC can be viewed as an improvement of the popular deviance information criterion (DIC), which has been criticized by several authors (Vehtari *et al.* 2015a; Plummer 2008; van der Linde 2005; see also the discussion at the end of the original DIC paper by Spiegelhalter, Best, Carlin, and Van Der Linde 2002) in part because of problems arising from fact that the DIC is only a point estimate. In **brms**, WAIC and LOO are implemented using the **loo** package (Vehtari, Gelman, and Gabry 2015b) also following the recommendations of Vehtari *et al.* (2015a).

4. Software

The **brms** package provides functions for fitting GLMMs using **Stan** for full Bayesian inference. To install the latest release version of **brms** from CRAN, type `install.packages("brms")` within R. The current developmental version can be downloaded from GitHub via

```
library(devtools)
install_github("paul-buerkner/brms")
```

Additionally, a C++ compiler is required. This is because **brms** internally creates **Stan** code, which is translated to C++ and compiled afterwards. The program **Rtools** (R Core Team 2015a) comes with a C++ compiler for Windows[2]. On OS X, one should use **Xcode** (Apple Inc. 2015) from the App Store. To check whether the compiler can be called within R, run `system("g++ -v")` when using **Rtools** or `system("clang++ -v")` when using **Xcode**. If no warning occurs and a few lines of difficult to read system code are printed out, the compiler should work correctly. For more detailed instructions on how to get the compilers running, see the prerequisites section on https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started.

Models are fitted in **brms** using the following procedure, which is also summarized in Figure 1. First, the user specifies the model using the `brm` function in a way typical for most model fitting R functions, that is by defining `formula`, `data`, and `family`, as well as some other optional arguments. Second, this information is processed and the `make_stancode` and `make_standata` functions are called. The former generates the model code in **Stan** language and the latter prepares the data for use in **Stan**. These two are the mandatory parts of every **Stan** model and without **brms**, users would have to specify them themselves. Third, **Stan** code and data as well as additional arguments (such as the number of iterations and chains) are passed to functions of the **rstan** package (the R interface of **Stan**; Stan Development Team, 2015a). Fourth, the model is fitted by **Stan** after translating and compiling it in C++. Fifth, after the model has been fitted and returned by **rstan**, the fitted model object is post-processed in **brms** among others by renaming the model parameters to be understood by the user. Sixth, the results can be investigated in R using various methods such as `summary`, `plot`, or `predict` (for a complete list of methods type `methods(class = "brmsfit")`).

[2]During the installation process, there is an option to change the system `PATH`. Please make sure to check this options, because otherwise **Rtools** will not be available within R.

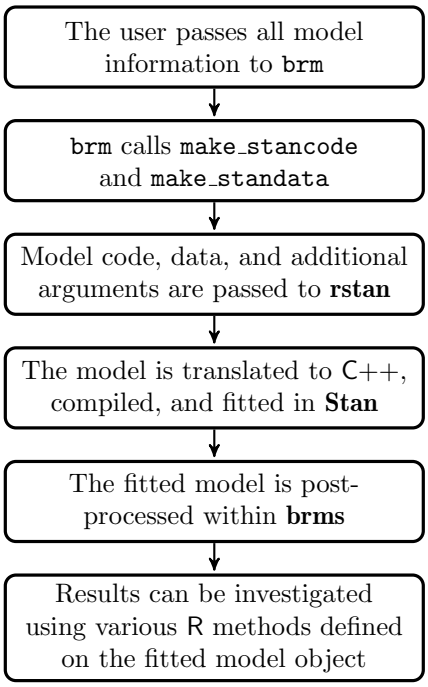

Figure 1: High level description of the model fitting procedure used in **brms**.

4.1. A worked example

In the following, we use an example about the recurrence time of an infection in kidney patients initially published by McGilchrist and Aisbett (1991). The data set consists of 76 entries of 7 variables:

```
R> library("brms")
R> data("kidney")
R> head(kidney, n = 3)
  time censored patient recur age    sex disease
1    8        0       1     1  28   male   other
2   23        0       2     1  48 female      GN
3   22        0       3     1  32   male   other
```

Variable `time` represents the recurrence time of the infection, `censored` indicates if `time` is right censored (`1`) or not censored (`0`), variable `patient` is the patient id, and `recur` indicates if it is the first or second recurrence in that patient. Finally, variables `age`, `sex`, and `disease` make up the predictors.

4.2. Fitting models with brms

The core of the **brms** package is the `brm` function and we will explain its argument structure using the example above. Suppose we want to predict the (possibly censored) recurrence time using a log-normal model with a random intercept and a random slope for `age` nested within patients. Then, we may use the following code:

```
fit1 <- brm(formula = time | cens(censored) ~ age + sex + disease
            + (1 + age|patient),
            data = kidney, family = gaussian("log"),
            prior = c(set_prior("normal(0,5)", class = "b"),
                      set_prior("cauchy(0,2)", class = "sd"),
                      set_prior("lkj(2)", class = "cor")),
            warmup = 1000, iter = 2000, chains = 4,
            control = list(adapt_delta = 0.95))
```

4.3. `formula`: Information on the response, fixed and random effects

Without doubt, `formula` is the most complicated argument, as it contains information on the response variable, fixed effects, and random effects at the same time. Everything before the $\sim$ sign relates to the response part of `formula`. In the usual and most simple case, this is just one variable name (e.g., `time`). However, to incorporate additional information about the response, one can add one or more terms of the form `| fun(variable)`. `fun` may be one of a few functions defined internally in **brms** and `variable` corresponds to a variable in the data set supplied by the user. In this example, `cens` makes up the internal function that handles censored data, and `censored` is the variable that contains information on the censoring. Other available functions in this context are `weights` for weighted regression, `se` to specify known standard errors primarily for meta-analysis, `trunc` to define truncation boundaries, `trials` for binomial models[3], and `cat` to specify the number of categories for categorical and ordinal models.

Everything on the right side of $\sim$ specifies predictors. Here, the syntax exactly matches that of **lme4**. For both, random and fixed effects terms, the `+` is used to separate different effects from each other. Random terms are of the form `(random | group)`, where `random` contains one or more variables whose effects are assumed to vary with the levels of the grouping factor given in `group`. Multiple grouping factors each with multiple random effects are possible. In the present example, only one random term is specified in which `1 + age` are the random effects and the grouping factor is `patient`. This implies that the intercept of the model as well as the effect of age is supposed to vary between patients. By default, random effects within a grouping factor are assumed to be correlated. Correlations can be set to zero by using the `(random || group)` syntax[4]. Everything on the right side of `formula` that is not recognized as part of a random term is treated as a fixed effect. In this example, the fixed effects are `age`, `sex`, and `disease`.

4.4. `family`: Distribution of the response variable

Argument `family` should usually be a family function, a call to a family function or a character string naming the family. If not otherwise specified, default link functions are applied.

[3] In functions such as `glm` or `glmer`, the binomial response is typically passed as `cbind(success, failure)`. In **brms**, the equivalent syntax is `success | trials(success + failure)`.

[4] In contrast to **lme4**, the `||` operator in **brms** splits up the design matrix computed from `random` instead of decomposing `random` in its terms. This implies that columns of the design matrix originating from the same factor are also assumed to be uncorrelated, whereas **lme4** estimates the correlations in this case. For a way to achieve **brms**-like behavior with **lme4**, see the `mixed` function of the **afex** package by Singmann, Bolker, and Westfall (2015).

brms comes with a large variety of families. Linear and robust linear regression can be performed using the `gaussian`, `student`, or `cauchy` family combined with the `identity` link. For dichotomous and categorical data, families `bernoulli`, `binomial`, and `categorical` combined with the `logit` link, by default, are perfectly suited. Families `poisson`, `negbinomial`, and `geometric` allow for modeling count data. Families `gaussian`, `gamma`, `exponential`, and `weibull` can be used (among others) for survival regression when combined with the `log` link. Ordinal regression can be performed using the families `cumulative`, `cratio`, `sratio`, and `acat`. Finally, families `zero_inflated_poisson`, `zero_inflated_negbinomial`, `hurdle_poisson`, `hurdle_negbinomial`, and `hurdle_gamma` can be used to adequately model excess zeros in the response. In our example, we use `family = gaussian("log")` implying a log-normal[5] "survival" model for the response variable `time`.

4.5. `prior`: Prior distributions of model parameters

Every fixed effect has its corresponding regression parameter. These parameters are named as `b_<fixed>`, where `<fixed>` represents the name of the corresponding fixed effect. The default prior for fixed effects parameters is an improper flat prior over the reals. Suppose, for instance, that we want to set a normal prior with mean `0` and standard deviation `10` on the fixed effect of `age` and a Cauchy prior with location `1` and scale `2` on `sexfemale`[6]. Then, we may write

```
prior <- c(set_prior("normal(0,10)", class = "b", coef = "age"),
           set_prior("cauchy(1,2)", class = "b", coef = "sexfemale"))
```

To put the same prior (e.g., a normal prior) on all fixed effects at once, we may write as a shortcut `set_prior("normal(0,10)", class = "b")`. This also leads to faster sampling, because priors can be vectorized in this case. Note that we could also omit the `class` argument for fixed effects, as it is the default class in `set_prior`.

A special shrinkage prior to be applied on fixed effects is the horseshoe prior (Carvalho, Polson, and Scott 2009, 2010). It is symmetric around zero with fat tails and an infinitely large spike at zero. This makes it ideal for sparse models that have many regression coefficients, although only a minority of them is non-zero. The horseshoe prior can be applied on all fixed effects at once (excluding the intercept) by using `set_prior("horseshoe(1)")`. The 1 implies that the student-t prior of the local shrinkage parameters has 1 degrees of freedom. In **brms** it is possible to increase the degrees of freedom (which will often improve convergence), although the prior no longer resembles a horseshoe in this case[7]. For more details see Carvalho *et al.* (2009, 2010).

Each random effect of each grouping factor has a standard deviation parameter, which is restricted to be non-negative and, by default, has a half Student-t prior with 3 degrees of

[5] For reasons of numerical efficiency and stability of the sampling algorithm, family `gaussian` with link `log` is interpreted as a log-normal distribution with identity link.

[6] When factors are used as predictors, parameter names will depend on the factor levels. To get an overview of all parameters and parameter classes for which priors can be specified, use function `get_prior`. For the present example, `get_prior(time | cens(censored) ~ age + sex + disease + (1 + age|patient), data = kidney)` does the desired.

[7] This class of priors is often referred to as hierarchical shrinkage family, which contains the original horseshoe prior as a special case.

freedom and a scale parameter that scales with the standard deviation of the response variable. **Stan** implicitly defines this prior by using a Student-t prior on a restricted parameter (Stan Development Team 2015b). For other reasonable priors on standard deviations see Gelman (2006). In **brms**, standard deviations are named as `sd_<group>_<random>` so that `sd_patient_Intercept` and `sd_patient_age` are the parameter names in the example. If desired, it is possible to set a different prior on each parameter, but statements such as `set_prior("student_t(3,0,5)", class = "sd", group = "patient")` or even `set_prior("student_t(3,0,5)", class = "sd")` may also be used and are again faster because of vectorization.

If there is more than one random effect per grouping factor, correlations between random effects are estimated. As mentioned in Section 2, the LKJ-Correlation prior with parameter $\zeta > 0$ (Lewandowski *et al.* 2009) is used for this purpose. In **brms**, this prior is abbreviated as `"lkj(zeta)"` and correlation matrix parameters are named as `cor_<group>`, (e.g., `cor_patient`), so that `set_prior("lkj(2)", class = "cor", group = "patient")` is a valid statement. To set the same prior on every correlation matrix in the model, `set_prior("lkj(2)", class = "cor")` is also allowed, but does not come with any efficiency increases.

Other model parameters such as the residual standard deviation `sigma` in normal models or the `shape` in Gamma models have their priors defined in the same way, where each of them is treated as having its own parameter class. A complete overview on possible prior distributions is given in the **Stan** user's manual (Stan Development Team 2015b). Note that **brms** performs no checks if the priors are written in correct **Stan** language. Instead, **Stan** will check their correctness when the model is parsed to C++ and returns an error if they are not.

4.6. `control`: Adjusting the sampling behavior of Stan

In addition to choosing the number of iterations, warmup samples, and chains, users can control the behavior of the NUTS sampler by using the `control` argument. The most important reason to use `control` is to decrease (or eliminate at best) the number of divergent transitions that cause a bias in the obtained posterior samples. Whenever you see the warning `"There were x divergent transitions after warmup."`, you should really think about increasing `adapt_delta`. To do this, write `control = list(adapt_delta = <x>)`, where `<x>` should usually be value between `0.8` (current default) and `1`. Increasing `adapt_delta` will slow down the sampler but will decrease the number of divergent transitions threatening the validity of your posterior samples.

Another problem arises when the depth of the tree being evaluated in each iteration is exceeded. This is less common than having divergent transitions, but may also bias the posterior samples. When it happens, **Stan** will throw out a warning suggesting to increase `max_treedepth`, which can be accomplished by writing `control = list(max_treedepth = <x>)` with a positive integer `<x>` that should usually be larger than the current default of `10`.

4.7. Analyzing the results

The example model `fit1` is fitted using 4 chains, each with 2000 iterations of which the first 1000 are warmup to calibrate the sampler, leading to a total of 4000 posterior samples[8]. For

[8] To save time, chains may also run in parallel when using argument `cluster`.

researchers familiar with Gibbs or Metropolis-Hastings sampling, this number may seem far to small to achieve good convergence and reasonable results, especially for hierarchical models. However, as **brms** utilizes the NUTS sampler (Hoffman and Gelman 2014) implemented in **Stan**, even complex models can often be fitted with not more than a few thousand samples. Of course, every iteration is more computationally intensive and time-consuming than the iterations of other algorithms, but the quality of the samples is way higher.

After the posterior samples have been computed, the `brm` function returns an R object, containing (among others) the fully commented model code in **Stan** language, the data to fit the model, and the posterior samples themselves. The model code and data for the present example can be extracted through `stancode(fit1)` and `standata(fit1)` respectively[9]. A model summary is readily available using

```
R> summary(fit1, waic = TRUE)
 Family: gaussian (log)
Formula: time | cens(censored) ~ age + sex + disease + (1 + age | patient)
   Data: kidney (Number of observations: 76)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: 672.58

Random Effects:
~patient (Number of levels: 38)
                  Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)         0.38      0.27     0.02     1.00       1253    1
sd(age)               0.01      0.01     0.00     0.02        880    1
cor(Intercept,age)   -0.12      0.46    -0.87     0.77       2498    1

Fixed Effects:
           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept      3.39      0.57     2.27     4.52       4000    1
age            0.00      0.01    -0.03     0.03       2703    1
sexfemale      1.54      0.38     0.78     2.28       4000    1
diseaseGN     -0.27      0.50    -1.26     0.70       2787    1
diseaseAN     -0.47      0.49    -1.40     0.53       2102    1
diseasePKD     0.74      0.68    -0.60     2.12       2445    1

Family Specific Parameters:
            Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma(time)     1.14      0.13     0.91     1.43       4000    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

On the top of the output, some general information on the model is given, such as family,

[9]Both model code and data may be amended and used to fit new models. That way, **brms** can also serve as a good starting point in building more complicated models in **Stan**, directly.

formula, number of iterations and chains, as well as the WAIC. Next, random effects are displayed separately for each grouping factor in terms of standard deviations and correlations between random effects. On the bottom of the output, fixed effects are displayed. If incorporated, autocorrelation and family specific parameters (e.g., the residual standard deviation `sigma`) are also given.

In general, every parameter is summarized using the mean (`Estimate`) and the standard deviation (`Est.Error`) of the posterior distribution as well as two-sided 95% Credible intervals (`l-95% CI` and `u-95% CI`) based on quantiles. The `Eff.Sample` value is an estimation of the effective sample size. That is the number of independent samples from the posterior distribution that would be expected to yield the same standard error of the posterior mean as is obtained from the dependent samples returned by the MCMC algorithm. The `Rhat` value provides information on the convergence of the algorithm. If `Rhat` is considerably greater than 1 (i.e., > 1.1), the chains have not yet converged and it is necessary to run more iterations and / or set stronger priors.

To visually investigate the chains as well as the posterior distribution, the `plot` method can be used (see Figure 2). An even more detailed investigation can be achieved by applying the **shinystan** package (Gabry 2015) through method `launch_shiny`.

With respect to the above summary, `sexfemale` seems to be the only fixed effect with considerable effect on the response. Because the mean of `sexfemale` is positive, the model predicts longer periods without an infection for females than for males. Looking at the random effects, the standard deviation of `age` is suspiciously small. To test whether it is smaller than the standard deviation of `Intercept`, we apply the `hypothesis` method:

```
R> hypothesis(fit1, "Intercept - age > 0", class = "sd", group = "patient")
Hypothesis Tests for class sd_patient:
                 Estimate Est.Error l-95% CI u-95% CI Evid.Ratio
Intercept-age > 0     0.37      0.27     0.02      Inf      89.91 *
---
'*': The expected value under the hypothesis lies outside the 95% CI.
```

The one-sided 95% credibility interval does not contain zero, thus indicating that the standard deviations differ from each other in the expected direction. In accordance with this finding, the `Evid.Ratio` shows that the hypothesis being tested (i.e., `Intercept - age > 0`) is about 90 times more likely than the alternative hypothesis `Intercept - age < 0`.

When looking at the correlation between both random effects, its distribution displayed in Figure 2 and the 95% credibility interval in the summary output appear to be rather wide. This indicates that there is not enough evidence in the data to reasonably estimate the correlation. Together, the small standard deviation of `age` and the uncertainty in the correlation raise the question if the random effect of `age` should be modeled at all. To answer this question, we fit another model (named `fit2`) similar to `fit1` but with `formula = time | cens(censored) ~ age + sex + disease + (1|patient)` and without any prior for `cor`. A good way to compare both models is leave-one-out cross-validation (LOO)[10], which can be called in **brms** using

[10]The WAIC is an approximation of LOO that is faster and easier to compute. However, according to Vehtari *et al.* (2015a), LOO may be the preferred method to perform model comparisons.

```
R> LOO(fit1, fit2)
            LOOIC    SE
fit1        674.58 45.21
fit2        673.62 45.16
fit1 - fit2   0.96  0.95
```

In the output, the LOO information criterion for each model as well as the difference of the LOOs each with its corresponding standard error is shown. Both, LOO and WAIC, are approximately normal if the number of observations is large so that the standard errors can be very helpful in evaluating differences in the information criteria. However, for small sample sizes, standard errors should be interpreted with care (Vehtari *et al.* 2015a). In addition, model weights are given that represent the posterior probability that each model has the best expected out-of-sample predictive accuracy (Vehtari *et al.* 2015b). For the present example, it is immediately evident that both models have very similar fit. Accordingly, the more parsimonious model `fit2` not containing random effects for `age` may be preferred.

4.8. Modeling ordinal data

In the following, we want to briefly discuss a second example to demonstrate the capabilities of **brms** in handling ordinal data. Ezzet and Whitehead (1991) analyze data from a two-treatment, two-period crossover trial to compare 2 inhalation devices for delivering the drug salbutamol in 286 asthma patients. Patients were asked to rate the clarity of leaflet instructions accompanying each device, using a four-point ordinal scale. Ratings are predicted by `treat` to indicate which of the two inhaler devices was used, `period` to indicate the time of administration, and `carry` to model possible carry over effects.

```
R> data("inhaler")
R> head(inhaler, n = 1)
  subject rating treat period carry
1       1      1   0.5    0.5     0
```

Typically, the ordinal response is assumed to originate from the categorization of a latent continuous variable. That is there are K latent thresholds (model intercepts), which partition the continuous scale into the $K + 1$ observable, ordered categories. Following this approach leads to the cumulative or graded-response model (Samejima 1969) for ordinal data implemented in many R packages. In **brms**, it is available via family `cumulative`. Fitting the cumulative model to the inhaler data, also incorporating a random intercept over subjects, may look this:

```
fit3 <- brm(formula = rating ~ treat + period + carry + (1|subject),
            data = inhaler, family = cumulative)
```

While the support for ordinal data in most R packages ends here[11], **brms** allows changes to this basic model in at least three ways. First of all, three additional ordinal families are implemented. Families `sratio` (stopping ratio) and `cratio` (continuation ratio) are so called sequential models (Tutz 1990). Both are equivalent to each other for symmetric link functions

[11] Exceptions known to us are the packages **ordinal** (Christensen 2015) and **VGAM** (Yee 2010). The former supports only cumulative models but with different modeling option for the thresholds. The latter supports all four ordinal families also implemented in **brms** as well as category specific effects but no random effects.

such as `logit` but will differ for asymmetric ones such as `cloglog`. The fourth ordinal family is `acat` (adjacent category) also known as partial credits model (Masters 1982; Andrich 1978b). Second, restrictions to the thresholds can be applied. By default, thresholds are ordered for family `cumulative` or are completely free to vary for the other families. This is indicated by argument `threshold = "flexible"` (default) in `brm`. Using `threshold = "equidistant"` forces the distance between two adjacent thresholds to be the same, that is

$$\tau_k = \tau_1 + (k-1)\delta$$

for thresholds τ_k and distance δ (see also Andrich 1978a; Andrich 1978b; Andersen 1977). Third, the assumption that predictors have constant effects across categories may be relaxed for non-cumulative ordinal models (Van Der Ark 2001; Tutz 2000) leading to category specific effects. For instance, variable `treat` may only have an impact on the decision between category 3 and 4, but not on the lower categories. Without using category specific effects, such a pattern would remain invisible.

To illustrate all three modeling options at once, we fit a (hardly theoretically justified) stopping ratio model with equidistant thresholds and category specific effects for variable `treat` on which we apply an informative prior.

```
fit4 <- brm(formula = rating ~ period + carry + (1|subject),
            data = inhaler, family = sratio,
            partial = ~ treat, threshold = "equidistant",
            prior = set_prior("normal(1,2)", coef = "treat"))
```

Note that priors are defined on category specific effects as if they were fixed effects. A model summary can be obtained in the same way as before:

```
R> summary(fit4, waic = TRUE)
 Family: sratio (logit)
Formula: rating ~ period + carry + (1 | subject) + partial(treat)
   Data: inhaler (Number of observations: 572)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: 914.73

Random Effects:
~subject (Number of levels: 286)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     1.02      0.24     0.56     1.48        462 1.01

Fixed Effects:
             Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept[1]     0.72      0.13     0.48     0.98       2295 1.00
Intercept[2]     2.57      0.34     1.92     3.28        754 1.01
Intercept[3]     4.42      0.65     3.17     5.77        815 1.01
period           0.26      0.18    -0.10     0.61       4000 1.00
carry           -0.30      0.22    -0.78     0.11       1992 1.00
treat[1]        -0.89      0.29    -1.46    -0.29       2301 1.00
```

```
treat[2]        -0.46      0.47    -1.36     0.44       4000 1.00
treat[3]        -1.79      1.26    -4.29     0.74       4000 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
delta     1.85      0.32     1.24      2.5        960    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Trace and density plots of the model parameters as produced by `plot(fit4)` can be found in Figure 3. We see that three intercepts (thresholds) and three effects of `treat` have been estimated, because a four-point scale was used for the ratings. The treatment effect seems to be strongest between category 3 and 4. At the same time, however, the credible interval is also much larger. In fact, the intervals of all three effects of `treat` are highly overlapping, which indicates that there is not enough evidence in the data to support category specific effects. On the bottom of the output, parameter `delta` specifies the distance between two adjacent thresholds and indeed the intercepts differ from each other by the magnitude of `delta`.

5. Comparison between packages

Over the years, many R packages have been developed that implement GLMMs, each being more or less general in their supported models. Comparing all of them to **brms** would be too extensive and barely helpful for the purpose of the present paper. Accordingly, we concentrate on a brief comparison with two packages that we believe are the most general and widely applied, namely **lme4** by Bates *et al.* (2015) and **MCMCglmm** by Hadfield (2010).

Regarding model families, all three packages support the most common types such as linear and binomial models as well as Poisson models for count data. Currently, **brms** and **MCMCglmm** provide more flexibility when modeling categorical and ordinal data. In addition, **brms** supports robust linear regression using Student's distribution, whereas **MCMCglmm** allows to fit multinomial models that are currently not available in **brms** or **lme4**.

In all three packages, there are quite a few additional modeling options. Variable link functions can be specified in **brms** and **lme4** but not in **MCMCglmm** in which only one link is available per family. **MCMCglmm** generally supports multivariate responses using data in wide format, whereas **brms** currently only offers this option for families `gaussian`, `student`, and `cauchy`. It should be noted that it is always possible to transform data from wide to long format for full compatibility with **brms** or **lme4**. Autocorrelation of the response can only be fitted in **brms**, which supports auto-regressive as well as moving-average effects. For ordinal models in **brms**, effects of predictors may vary across different levels of the response as explained in the inhaler example.

Information criteria are available in all three packages. The advantage of WAIC and LOO implemented in **brms** is that their standard errors can be easily estimated to get a better sense of the uncertainty in the criteria. Comparing the prior options of the Bayesian packages, **brms** offers a little more flexibility than **MCMCglmm**, as virtually any prior distribution can be applied on fixed effects as well as on the standard deviations of random effects. In addition,

we believe that the way priors are specified in **brms** is more intuitive as it is directly evident what prior is actually applied (see the model specification in Section 4). A more detailed comparison of the packages can be found in Table 1. To facilitate the understanding of the model formulation in **brms**, Table 2 shows **lme4** function calls to fit sample models along with the equivalent **brms** syntax.

So far the focus was only on capabilities. Another important topic is speed, especially for huge and complex models. Of course, **lme4** is usually much faster than the other packages as it uses maximum likelihood methods instead of MCMC algorithms, which are slower by design. As compared to **MCMCglmm**, **brms** needs more time per iteration, but also produces higher quality samples, so that the default of 3000 posterior samples is often more than enough to achieve good results. On the other hand, **MCMCglmm** may require hundreds of thousand iterations, but can manage to get this in the same time **brms** samples a few thousand. For small models, it feels that **MCMCglmm** is faster than **brms** as the latter additionally requires a few seconds to compile the model. For larger models, **brms** can benefit from the possibility of parallelizing chains to drastically improve its efficiency.

6. Conclusion

The present paper is meant to provide a general overview on the R package **brms** implementing GLMMs using the probabilistic programming language **Stan** for full Bayesian inference. Although only a small selection of the modeling options available in **brms** are discussed in detail, we hope that this article can serve as a good starting point to further explore the capabilities of the package.

For the future, we have several plans on how to improve the functionality of **brms**. We want to include multivariate models that can handle multiple response variables coming from different distributions. Also, generalized additive mixed models (Hastie and Tibshirani 1990) may be implemented in future versions of the package. Besides MCMC sampling, **Stan** also provides algorithms for penalized maximum likelihood and variational inference (Stan Development Team 2015b). While providing support for penalized maximum likelihood is probably of less importance, variational inference can serve as a good alternative to MCMC sampling, if the latter is unfeasible for instance because it is not fast enough. At the time of writing this article, variational inference was not yet fully functional and available in **rstan** so that it could not be implemented in the present version of **brms**.

Acknowledgments

First of all, we would like to thank the Stan Development Team for creating the probabilistic programming language **Stan**, which is an incredibly powerful and flexible tool for performing full Bayesian inference. Without it, **brms** could not fit a single model. Furthermore, many users have provided valuable feedback and suggestions since the initial release of the package, thus helping to substantially improve **brms**.

References

	brms	**lme4**	**MCMCglmm**
Supported model types:			
Linear models	yes	yes	yes
Robust linear models	yes	no	no
Binomial models	yes	yes	yes
Categorical models	yes	no	yes
Multinomial models	no	no	yes
Count data models	yes	yes	yes
Survival models	yes[1]	yes	yes
Ordinal models	various	no	cumulative
Zero-inflated and hurdle models	yes	no	yes
Non-linear models	yes	no	no
Additional modeling options:			
Variable link functions	various	various	no
Weights	yes	yes	no
Offset	yes	yes	using priors
Multivariate responses	limited	no	yes
Autocorrelation effects	yes	no	no
Category specific effects	yes	no	no
Standard errors for meta-analysis	yes	no	yes
Censored data	yes	no	yes
Customized covariances	yes	no	yes
Bayesian specifics:			
parallelization	yes	–	no
fixed effects priors	flexible	–	normal
random effects priors	normal	–	normal
covariance priors	flexible	–	flexible
Other:			
Estimator	HMC, NUTS	ML, REML	MH, Gibbs[2]
Information criterion	WAIC, LOO	AIC, BIC	DIC
C++ compiler required	yes	no	no
Modularized	no	yes	no

Table 1: Comparison of the capabilities of the **brms**, **lme4** and **MCMCglmm** package. Notes: (1) Weibull family only available in **brms**. (2) Estimator consists of a combination of both algorithms.

Dataset	**Function call**
cake	
lme4	`lmer(angle ~ recipe * temperature + (1|recipe:replicate), data = cake)`
brms	`brm(angle ~ recipe * temperature + (1|recipe:replicate), data = cake)`
sleepstudy	
lme4	`lmer(Reaction ~ Days + (Days|Subject), data = sleepstudy)`
brms	`brm(Reaction ~ Days + (Days|Subject), data = sleepstudy)`
cbpp[1]	
lme4	`glmer(cbind(incidence, size - incidence) ~ period + (1 | herd), family = binomial("logit"), data = cbpp)`
brms	`brm(incidence | trials(size) ~ period + (1 | herd), family = binomial("logit"), data = cbpp)`
grouseticks[1]	
lme4	`glmer(TICKS ~ YEAR + HEIGHT + (1|BROOD) + (1|LOCATION), family = poisson("log"), data = grouseticks)`
brms	`brm(TICKS ~ YEAR + HEIGHT + (1|BROOD) + (1|LOCATION), family = poisson("log"), data = grouseticks)`
VerbAgg[2]	
lme4	`glmer(r2 ~ (Anger + Gender + btype + situ)^2 + (1|id) + (1|item), family = binomial, data = VerbAgg)`
brms	`brm(r2 ~ (Anger + Gender + btype + situ)^2 + (1|id) + (1|item), family = bernoulli, data = VerbAgg)`

Table 2: Comparison of the model syntax of **lme4** and **brms** using data sets included in **lme4**. Notes: (1) Default links are used to that the link argument may be omitted. (2) Fitting this model takes some time. A proper prior on the fixed effects (e.g., `prior = set_prior("normal(0,5)")`) may help in increasing sampling speed.

Andersen EB (1977). "Sufficient Statistics and Latent Trait Models." *Psychometrika*, **42**(1), 69–81.

Andrich D (1978a). "Application of a Psychometric Rating Model to Ordered Categories which are Scored with Successive Integers." *Applied Psychological Measurement*, **2**(4), 581–594.

Andrich D (1978b). "A Rating Formulation for Ordered Response Categories." *Psychometrika*, **43**(4), 561–573.

Apple Inc (2015). ***Xcode*** *Software, Version 7*. Cupertino, USA. URL https://developer.apple.com/xcode/.

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48.

Brown H, Prescott R (2015). *Applied Mixed Models in Medicine*. John Wiley & Sons.

Carpenter B, Gelman A, Hoffman M, Lee D, Goodrich B, Betancour M, Brubaker MA, Guo J, Li P, Ridell A (2015). "**Stan**: A Probabilistic Programming Language." *Journal of Statistical Software*. URL http://www.stat.columbia.edu/~gelman/research/published/Stan-paper-aug-2015.pdf.

Carvalho CM, Polson NG, Scott JG (2009). "Handling Sparsity via the Horseshoe." In *International Conference on Artificial Intelligence and Statistics*, pp. 73–80.

Carvalho CM, Polson NG, Scott JG (2010). "The Horseshoe Estimator for Sparse Signals." *Biometrika*, pp. 1–16.

Christensen RHB (2015). "**ordinal** – Regression Models for Ordinal Data." R package version 2015.6-28. http://www.cran.r-project.org/package=ordinal/.

Creutz M (1988). "Global Monte Carlo Algorithms for Many-Fermion Systems." *Physical Review D*, **38**(4), 1228.

Damien P, Wakefield J, Walker S (1999). "Gibbs Sampling for Bayesian Non-Conjugate and Hierarchical Models by Using Auxiliary Variables." *Journal of the Royal Statistical Society B, Statistical Methodology*, pp. 331–344.

Demidenko E (2013). *Mixed Models: Theory and Applications with R*. John Wiley & Sons.

Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987). "Hybrid Monte Carlo." *Physics Letters B*, **195**(2), 216–222.

Ezzet F, Whitehead J (1991). "A Random Effects Model for Ordinal Responses from a Crossover Trial." *Statistics in Medicine*, **10**(6), 901–907.

Gabry J (2015). ***shinystan***: *Interactive Visual and Numerical Diagnostics and Posterior Analysis for Bayesian Models*. R Package Version 2.0.0, URL http://CRAN.R-project.org/package=shinystan.

Gelfand AE, Dey DK, Chang H (1992). "Model Determination Using Predictive Distributions with Implementation via Sampling-Based Methods." *Technical report*, DTIC Document.

Gelfand AE, Smith AF (1990). "Sampling-Based Approaches to Calculating Marginal Densities." *Journal of the American Statistical Association*, **85**(410), 398–409.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**(3), 515–534.

Gelman A, Carlin JB, Stern HS, Rubin DB (2014). *Bayesian Data Analysis*, volume 2. Taylor & Francis.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 721–741.

Griewank A, Walther A (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Siam.

Hadfield JD (2010). "MCMC Methods for Multi-Response Generalized Linear Mixed Models: the **MCMCglmm** R Package." *Journal of Statistical Software*, **33**(2), 1–22.

Hastie TJ, Tibshirani RJ (1990). *Generalized Additive Models*, volume 43. CRC Press.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and their Applications." *Biometrika*, **57**(1), 97–109.

Hoffman MD, Gelman A (2014). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *The Journal of Machine Learning Research*, **15**(1), 1593–1623.

Ionides EL (2008). "Truncated Importance Sampling." *Journal of Computational and Graphical Statistics*, **17**(2), 295–311.

Juárez MA, Steel MF (2010). "Model-Based Clustering of Non-Gaussian Panel Data Based on Skew-t Distributions." *Journal of Business & Economic Statistics*, **28**(1), 52–66.

Kass RE, Natarajan R (2006). "A Default Conjugate Prior for Variance Components in Generalized Linear Mixed Models (Comment on Article by Browne and Draper)." *Bayesian Analysis*, **1**(3), 535–542.

Lewandowski D, Kurowicka D, Joe H (2009). "Generating Random Correlation Matrices Based on Vines and Extended Onion Method." *Journal of Multivariate Analysis*, **100**(9), 1989–2001.

Lunn DJ, Thomas A, Best N, Spiegelhalter D (2000). "**WinBUGS** a Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**(4), 325–337.

Masters GN (1982). "A Rasch Model for Partial Credit Scoring." *Psychometrika*, **47**(2), 149–174.

McGilchrist C, Aisbett C (1991). "Regression with Frailty in Survival Analysis." *Biometrics*, pp. 461–466.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). "Equation of State Calculations by Fast Computing Machines." *The Journal of Chemical Physics*, **21**(6), 1087–1092.

Natarajan R, Kass RE (2000). "Reference Bayesian Methods for Generalized Linear Mixed Models." *Journal of the American Statistical Association*, **95**(449), 227–237.

Neal RM (2003). "Slice Sampling." *The Annals of Statistics*, pp. 705–741.

Neal RM (2011). "MCMC Using Hamiltonian Dynamics." *Handbook of Markov Chain Monte Carlo*, **2**.

Pinheiro J, Bates D (2006). *Mixed-Effects Models in S and S-PLUS*. Springer Science & Business Media.

Plummer M (2008). "Penalized Loss Functions for Bayesian Model Comparison." *Biostatistics*.

Plummer M (2013). ***JAGS**: Just Another Gibs Sampler*. URL http://mcmc-jags.sourceforge.net/.

R Core Team (2015a). ***Rtools** Software, Version 3.3*. R Foundation for Statistical Computing, Vienna, Austria. URL https://cran.r-project.org/bin/windows/Rtools/.

R Core Team (2015b). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Samejima F (1969). "Estimation of Latent Ability Using a Response Pattern of Graded Scores." *Psychometrika Monograph Supplement*.

Singmann H, Bolker B, Westfall J (2015). *afex: Analysis of Factorial Experiments*. R package version 0.15-2, URL https://CRAN.R-project.org/package=afex.

Spiegelhalter D, Thomas A, Best N, Lunn D (2003). ***WinBUGS** Version - 1.4 User Manual*. URL http://www.mrc-bsu.cam.ac.uk/bugs.

Spiegelhalter D, Thomas A, Best N, Lunn D (2007). ***OpenBUGS** User Manual, Version 3.0.2*.

Spiegelhalter DJ, Best NG, Carlin BP, Van Der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B, Statistical Methodology*, **64**(4), 583–639.

Stan Development Team (2015a). ***Stan**: A C++ Library for Probability and Sampling, Version 2.8.0*. URL http://mc-stan.org/.

Stan Development Team (2015b). ***Stan** Modeling Language: User's Guide and Reference Manual*. URL http://mc-stan.org/manual.html.

Tutz G (1990). "Sequential Item Response Models with an Ordered Response." *British Journal of Mathematical and Statistical Psychology*, **43**(1), 39–55.

Tutz G (2000). *Die Analyse Kategorialer Daten: Anwendungsorientierte Einführung in Logit-Modellierung und Kategoriale Regression*. Oldenbourg Verlag.

Van Der Ark LA (2001). "Relationships and Properties of Polytomous Item Response Theory Models." *Applied Psychological Measurement*, **25**(3), 273–282.

van der Linde A (2005). "DIC in Variable Selection." *Statistica Neerlandica*, **59**(1), 45–56.

Vehtari A, Gelman A, Gabry J (2015a). "Efficient Implementation of Leave-One-Out Cross-Validation and WAIC for Evaluating Fitted Bayesian Models." *Unpublished manuscript*, pp. 1–22. URL http://www.stat.columbia.edu/~gelman/research/unpublished/loo_stan.pdf.

Vehtari A, Gelman A, Gabry J (2015b). ***loo**: Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models*. R Package Version 0.1.3, URL https://github.com/jgabry/loo.

Watanabe S (2010). "Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory." *The Journal of Machine Learning Research*, **11**, 3571–3594.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34.

Affiliation:

Paul-Christian Bürkner
Department of Statistics
Faculty of Psychology
University of Münster
48149, Münster
E-mail: paul.buerkner@wwu.de
URL: http://wwwpsy.uni-muenster.de/Psychologie.inst4/AEHolling/personen/buerkner.html

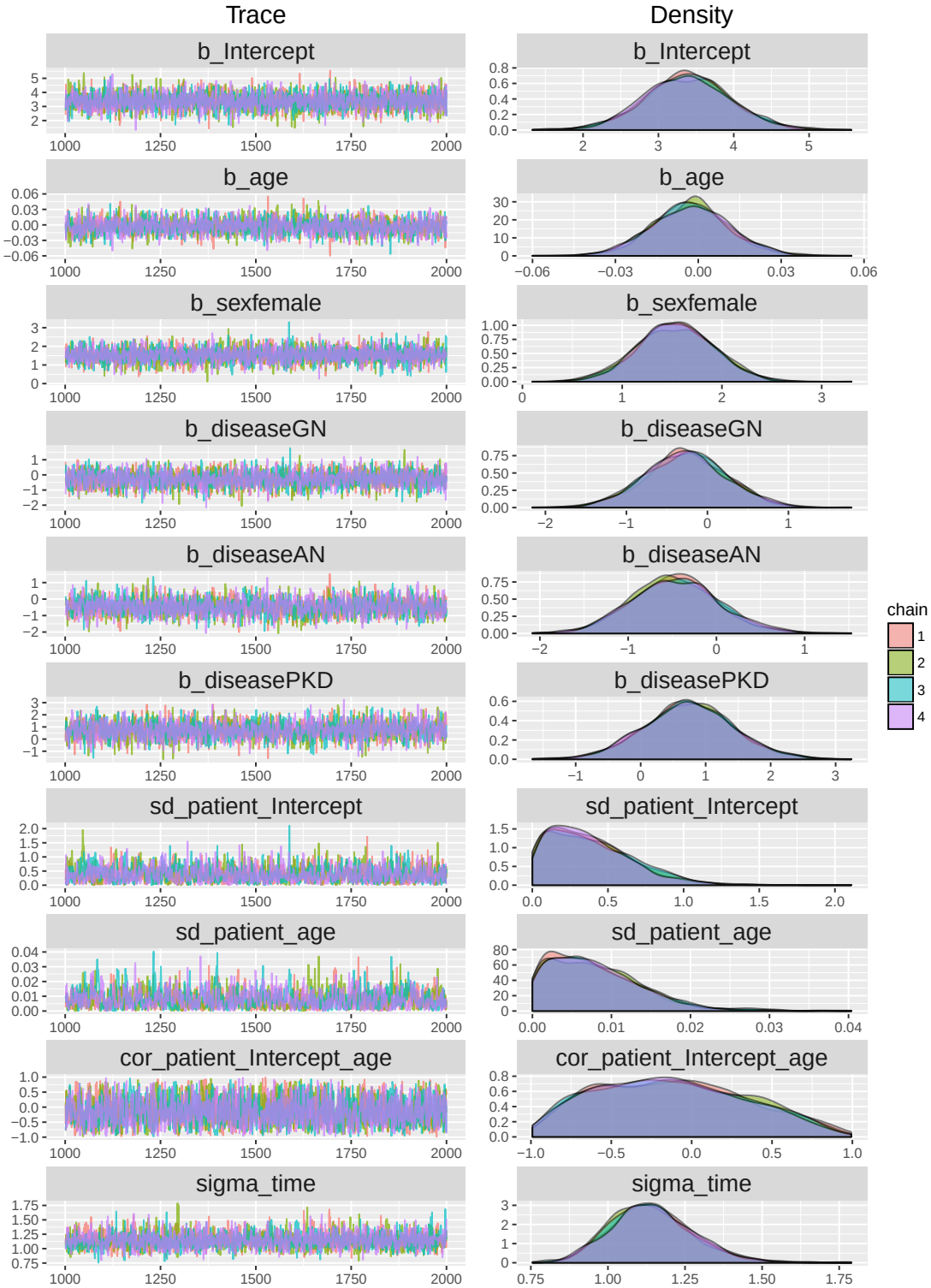

Figure 2: Trace and Density plots of all relevant parameters of the kidney model discussed in Section 4.

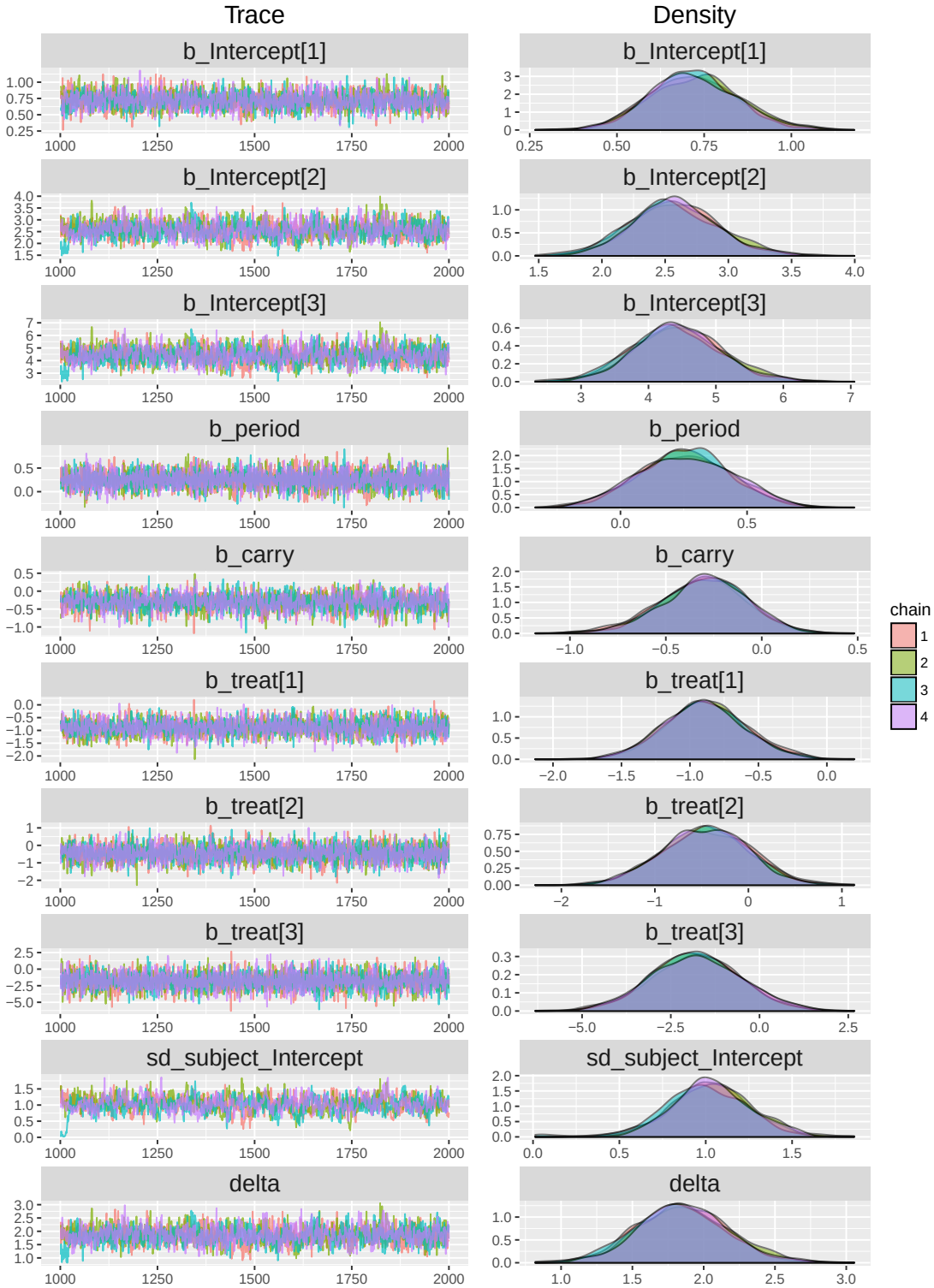

Figure 3: Trace and Density plots of all relevant parameters of the inhaler model discussed in Section 4.